

Forum:[Hack Forums](#)

Thread: [Summary of PHP 7 features](#)

Sakuto - January 12, 2016, 5:41 pm

Summary of PHP 7 features

Hello everybody, as you all know, PHP 7 is finally available in every good hosting service, and it's for this reason that we will take a look at the major features of this new version. I will try to explain everything in a clear way with example, and I hope you will enjoy reading it :)

[img]http://nsa37.casimages.com/img/2015/12/30/151230044751619202.png[/img]

Scalar typePHP 7 introduce a whole new feature, as you were already able to do it before with class, you are now able to use scalar type to define the expected type for parameter, but also the return type of a function. Here's a quick example:

PHP Code

```
<?php
function testFunction(float $myNumber) {
    echo "Your number is: $myNumber";
}
```

```
testFunction(12.5); // will work perfectly
```

```
testFunction(12);
```

```
// with a boxing method, PHP will handle this int passed while a float was expected
```

You have two level of verification for scalar type, strict mode and non-strict. In the non-strict mode (the default mode), PHP will try to convert the given variable into the right type with a boxing system. Contrariwise, in strict mode, if the given variable doesn't match the type, PHP will return a Fatal Error. For example, if we are taking back our previous code, the second call of our function would make the script return a Fatal Error. You can easily switch from one to the other by typing this in the beginning of your page.

PHP Code

```
<?php
declare(strict_types=1);
```

But what would've been scalar type without return type? Nothing really interesting. You are now, in addition, able to define a return type for your function, let's take our previous code again and implement it!

PHP Code

```
<?php
function testFunction(float $myNumber): string {
    return "Your number is: $myNumber";
}
```

```
testFunction(12.5); // will work perfectly
```

```
testFunction(12); // with a boxing method, PHP will handle this int passed in a float
```

PHP actually support eight primitive type: Four scalar types boolean

integer

float (or double)

string

Two compound types array

object

Two special types resource

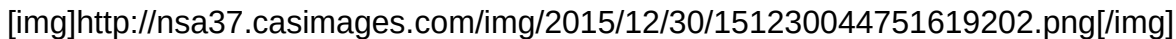
NULL

But also pseudo-type mixed

number

callback (or Callable)

void



Null coalescing Operator Great addition to this version, the null coalescing Operator, this simple operator will allow you to drastically reduce your code! It will return take the first parameter if it work, otherwise the second. Here's an example:

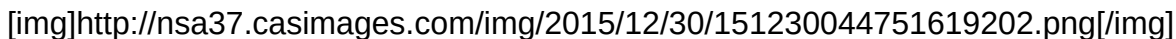
PHP Code

```
<?php
```

```
$pageld = isset($_GET['pageld']) ? $_GET['pageld'] : 1; // Before
```

```
$pageld = $_GET['pageld'] ?? 1; // After
```

```
$pageld = $_GET['pageld'] ?? $_SESSION['lastPage'] ?? 1; // We can chain with this operator
```



Tie Fighter Operator New little feature that will allow you to compare two variables. It will return:

- 0: if both variables are identical

- -1: if the first variable is the greatest

- 1: if the second variable is the greatest

PHP Code

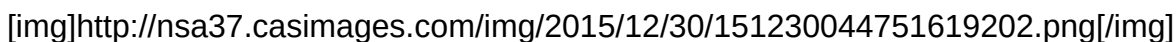
```
<?php
```

```
1 <=> 1 // return 0
```

```
0 <=> 1 // return -1
```

```
1 <=> 0 // return 1
```

Very simple addition, but could be useful and allow you to avoid multiple condition with if, elseif, else...



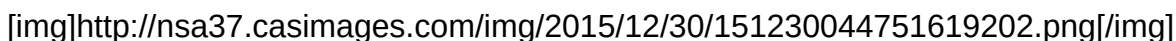
Array constants You are now able to declare array in constant, nothing really hard to explain here so let me quickly show you an example:

PHP Code

```
<?php
```

```
const BESTUSER = ['Sakuto', 'StackOverlowin'];
```

```
var_dump(BESTUSER);
```



Group use declaration The new version give us the ability to group multiple use in one instruction, particularly useful when working with Framework or a lot of library.

PHP Code

```
<?php
// Before
use App\HF\Forum;
use App\HF\User;
use App\HF\God as Sakuto;

// Now
use App\HF\{Forum, User, God as Sakuto}
```

[img]http://nsa37.casimages.com/img/2015/12/30/151230044751619202.png[/img]

Closure Call We're now able to inject closure into existing class with the method `->call()`. This method will be executed as a method of this class and so will have access to the context of the current class instance.

PHP Code

```
<?php
class myClass {
    public $firstVar;
}

$foo = new myClass();
$foo->firstVar = 'bar';

$showValue = function() { return $this->firstVar; };

echo $showValue->call($foo);
```

A bit of explanation for this piece of code. We are creating a new class with a public attribute `firstVar`. Instantiate and assign a value to your instance of class `myClass`. We are then creating a closure that just return the variable `firstVar`.

When calling the closure on the `foo` instance, the current context will be equals to the context, the closure will get access to the attribute `firstVar` and will correctly return `"bar"`;

[img]http://nsa37.casimages.com/img/2015/12/30/151230044751619202.png[/img]

Assert The `assert` function is fully changed, in the last version of PHP, we were only able to return a string as the second parameter. In the version 7, you can now return custom class to avoid typing again and again the same message, a short example to make it clearer.

PHP Code

```
<?php
// In PHP 5.6
assert(1 == 2, "They are not equals");
assert(1 == 3, "They are not equals");

// In PHP 7
assert(1 == 2, new AssertionError());
assert(1 == 3, new AssertionError());
```

The main advantage is that you don't need to write again and again the same message, you can now return any class that implement the `Throwable` interface. Making it easier to test your script.

[img]http://nsa37.casimages.com/img/2015/12/30/151230044751619202.png[/img]

Mysql_* is deadTo conclude this short presentation of the new feature, we will talk about the best decision made on this release. The suppression of mysql_* function.

You are now forced to switch to PDO or MySQLi, and it make me feel good, really good.

[img]http://nsa37.casimages.com/img/2015/12/30/151230044751619202.png[/img]

And way more...Of course this list was not exhaustive, they are a lot of new feature / addition to current feature to explore like [Yield, Anonymous Class, Reflection](#) and much more... But it's up to you read to this if you're interested. I tried to make a short list of the features that I think, you'll use very soon.

Thanks to [Haxalot](#) for the reference guide linked in Yield, Anonymous class and Reflection

Nipsey Hussle - January 12, 2016, 6:35 pm

Nice share man! Really like the features php 7 has brought on make things easier and other nice features they've added! Namespace short cut is a feature I will be using a ton most the time when I'm working with frameworks or packages with composer which I do a ton! Great thread to help anyone also trying to conver code from 5 to 7 if they want too :)

Alex™ - January 12, 2016, 6:45 pm

PHP Code

```
<?php
```

```
$pageld = $_GET['pageld'] ?? $_SESSION['lastPage'] ?? 1; // We can chain with this operator  
Best. Feature. Ever.
```

LilDurk - January 12, 2016, 10:05 pm

Nice share Sakuto, always like your threads.

Fkn - January 12, 2016, 10:58 pm

Great thread Sakuto, thanks for the time you've put into this.

ShizzleFly - January 13, 2016, 2:49 am

Great thread here buddy. +repped for making these threads

Hash G. - January 13, 2016, 5:10 am

Great and informative thread Sakuto, thanks.

newQuery - January 13, 2016, 5:45 am

Awesome man, very useful thank you ! :)

FocalBanger - January 13, 2016, 6:32 am

Thanks for the share man! Very useful and well written!

SIM-2434055 - January 13, 2016, 11:06 am

Nice job on the thread!

?

\$lick - January 13, 2016, 12:03 pm

(Jan 12, 2016 07:45 PM)Alex™ Wrote: [\[CITE\]](#) PHP Code

```
<?php
```

```
$pageld = $_GET['pageld'] ?? $_SESSION['lastPage'] ?? 1; // We can chain with this operator
```

Best. Feature. Ever.
Fucking finally, it was killing me to not be able to do this.

noname1 - January 13, 2016, 12:21 pm
nice write up Sakuto, thank you

Fox™ - January 13, 2016, 1:39 pm
mysql_* has been deprecated since 5.6 though.

Sakuto - January 13, 2016, 1:41 pm
(Jan 13, 2016 02:39 PM)Fox™ Wrote: [\[CITE\]](#) mysql_* has been deprecated since 5.6 though.
It's not just deprecated, it's fully removed now.

Thanks everybody for your kind comments :)

Fox™ - January 13, 2016, 1:45 pm
(Jan 13, 2016 02:41 PM)Sakuto Wrote: [\[CITE\]](#) It's not just deprecated, it's fully removed now.

Thanks everybody for your kind comments :)
mysql_* functions for me didn't work in 5.6 anyway when I was trying out a really old script that I wrote, but if they weren't removed fully then, I'm glad it is now.

Remix - January 13, 2016, 2:58 pm
(Jan 13, 2016 02:45 PM)Fox™ Wrote: [\[CITE\]](#) mysql_* functions for me didn't work in 5.6 anyway when I was trying out a really old script that I wrote, but if they weren't removed fully then, I'm glad it is now.
They actually should've been working in 5.6
But I'm wondering who still tries to use it ?

Fox™ - January 13, 2016, 3:55 pm
(Jan 13, 2016 03:58 PM)+Remix Wrote: [\[CITE\]](#) They actually should've been working in 5.6
But I'm wondering who still tries to use it ?
I fired up an old script to see about improving it and updating to PDO, but 5.6 totally broke the mysql_*, so I had to update to PDO before I could even check out the script again. It could have something within my installation.

MineSQL - January 13, 2016, 10:33 pm
Pretty decent thread, your english was getting a little broken here and there, so maybe read through it again when you get the chance.

I am quite happy about type hinting being added to the code, much easier to troubleshoot/test code than in the past.

Sakuto - January 14, 2016, 7:21 am
(Jan 13, 2016 04:55 PM)Fox™ Wrote: [\[CITE\]](#) I fired up an old script to see about improving it and updating to PDO, but 5.6 totally broke the mysql_*, so I had to update to PDO before I could even check out the script again. It could have something within my installation. Deprecated function are usually throwing Notice, not Exception / FE, strange.

(Jan 13, 2016 11:33 PM)MineSQL Wrote: [\[CITE\]](#) Pretty decent thread, your english was getting a little broken here and there, so maybe read through it again when you get the chance.

I am quite happy about type hinting being added to the code, much easier to troubleshoot/test code

than in the past.I'll try to improve the grammar / text, but my knowledge is pretty limited, so I'm not sure about my capacity to find the mistakes :B

Thanks for your comment by the way (:

SIM-1947090 - January 14, 2016, 12:30 pm

Null coalescing Operator

and

The Fighter Operater

made my day :)

Thanks for the info ;)

False King - January 15, 2016, 12:50 pm

PHP7 really has some great updates, besides that it's overall performance/speed has also had a major boost. Really nice thread, love your threads in the PHP section.

#Skyline - January 16, 2016, 12:23 am

I'm glad the `mysql_*` function is dead.

And this is a very informational thread about PHP 7, good stuff !

DashPHP - January 19, 2016, 12:25 pm

I love the new features, especially the new ternary type one.

Though to be fair, most web servers are still running on the 5.6 stable release, so if you were to write code on localhost with php 7.0, I'd be careful not to use these new features unless you are sure your server/web host has php 7.

Very interesting thread, well done!

sarciszewski - January 19, 2016, 10:25 pm

You forgot about the CSPRNG functions, `random_bytes()` and `random_int()` :(

??e rando? d??c? g?y - January 20, 2016, 4:47 pm

I though using `mysql_` was already death tbh

DrWhatHF - January 21, 2016, 6:16 am

OP's Closure Call is wrong.

you could always doing closure call since php5.3 when they added `std::class` interface

and you forgot to mention Anonymous Classes.

Sakuto - February 17, 2016, 2:39 pm

Bump, still interesting to read too.

itsMeCherry - February 17, 2016, 2:56 pm

nooo, `mysql_` where you go ;(

Haxalot - February 18, 2016, 8:38 am

Just a quick review...

(Jan 12, 2016 06:41 PM)Sakuto Wrote: [\[CITE\]](#) You have two level of verification for scalar type, strict

mode and non-strict.

The official terminology is strict mode and coercive mode.

(Jan 12, 2016 06:41 PM)Sakuto Wrote: [\[CITE\]](#) Contrariwise, in strict mode, if the given variable doesn't match the type, PHP will return a Fatal Error. For example, if we are taking back our previous code, the second call of our function would make the script return a Fatal Error.

You've found [the only case](#) where strict mode accepts a type different to the one that has been declared. In strict mode, float type declarations will accept either integers or floats. This isn't exactly a good thing, since converting integers to floats will result in precision loss in some cases. For 64bit platforms, integers in PHP (which are stored internally as longs) will have their precision lost for some numbers greater than 2^{53} .

For example:

PHP Code

```
<?php
```

```
var_dump(PHP_INT_MAX - 511.0 === PHP_INT_MAX - 512.0); // bool(true)
```

(<https://3v4l.org/3MF0k>)

(Jan 12, 2016 06:41 PM)Sakuto Wrote: [\[CITE\]](#) PHP actually support eight primitive type:Four scalar typesboolean

integer

float (or double)

string

Two compound typesarray

object

Two special typesresource

NULL

But also pseudo-typesmixed

number

callback (or Callable)

void

I assume you're referring to that list in the context of return type declarations? If so, it should be the following:bool (shorthand alias only for booleans)

int (shorthand alias only for integers)

float (PHP uses the term "float" for fractional numbers, even though internally they're stored as doubles)

string

array

classes or interfaces (invariance subtyping)

callable

void (as of PHP 7.1, to be released later this year)

(Jan 12, 2016 06:41 PM)Sakuto Wrote: [\[CITE\]](#) Tie Fighter Operator

Never heard it called that before. The official terminology used is typically the combined comparison operator (or spaceship operator).

(Jan 12, 2016 06:41 PM)Sakuto Wrote: [\[CITE\]](#) Array constantsYou are now able to declare array in constant, nothing really hard to explain here so let me quickly show you an example:

PHP Code

```
<?php
const BESTUSER = ['Sakuto', 'StackOverlowin'];

var_dump(BESTUSER);
```

Support for what you've just shown was actually added in PHP 5.6. Unfortunately, const's counterpart, define(), was forgotten about, and so it was rectified in PHP 7 to enabled array values to be defined:

PHP Code

```
<?php
define('ALLOWED_IMAGE_EXTENSIONS', ['jpg', 'jpeg', 'gif', 'png']);
```

(Jan 12, 2016 06:41 PM)Sakuto Wrote: [\[CITE\]](#) Group use declarationThe new version give us the ability to group multiple use in one instruction, particularly useful when working with Framework or a lot of library.

PHP Code

```
<?php
// Before
use App\HF\Forum;
use App\HF\User;
use App\HF\God as Sakuto;

// Now
use App\HF{Forum, User, God as Sakuto}
```

Cool (but ugly) fact: you can import functions and constants too in the same group use declaration:

PHP Code

```
<?php

use A\SomeClass;
use function A\someFunction;
use const A\SOME_CONST;

// same as
use A{SomeClass, function someFunction, const SOME_CONST};
```

(Jan 12, 2016 06:41 PM)Sakuto Wrote: [\[CITE\]](#) AssertThe assert function is fully changed, in the last version of PHP, we were only able to return a string as the second parameter. In the version 7, you can now return custom class to avoid typing again and again the same message, a short example to make it clearer.

// redacted

The main advantage is that you don't need to write again and again the same message, you can now return any class that implement the Throwable interface. Making it easier to test your script.

The predominant reason was not about not having to retype the same error message each time - it was about having a more flexible error handling system with easier debugging (since exceptions produce a stack trace).

(Jan 12, 2016 06:41 PM)Sakuto Wrote: [\[CITE\]](#) Thanks to [Haxalot](#) for the reference guide linked in Yield, Anonymous class and Reflection
Thanks! I hope you enjoyed my PHP 7 Reference article :)

Hash G. - February 18, 2016, 8:41 am

(Feb 17, 2016 03:56 PM)Fire First Wrote: [\[CITE\]](#) nooo, mysql_ where you go ;(
Honestly if you're missing mysql_ you should really update yourself on how PHP has changed in the last years.

itsMeCherry - February 18, 2016, 10:15 am

(Feb 18, 2016 09:41 AM)Hash G. Wrote: [\[CITE\]](#) Honestly if you're missing mysql_ you should really update yourself on how PHP has changed in the last years.
No need to, i am fine with my perfectly working mysql which is barely any slower than mysqli

Remix - February 18, 2016, 1:05 pm

(Feb 18, 2016 11:15 AM)Fire First Wrote: [\[CITE\]](#) No need to, i am fine with my perfectly working mysql which is barely any slower than mysqli
Are you seriously still using mysql_?????????????
??

SIM-1947090 - February 18, 2016, 2:01 pm

(Feb 18, 2016 02:05 PM)+Remix Wrote: [\[CITE\]](#) Are you seriously still using mysql_????????????????
??
And i thought mysqli_ was "outdated/old_fashioned" and "not modern" lol people still use mysql_ or someone is being sarcastic xD

Haxalot - February 18, 2016, 2:47 pm

(Feb 18, 2016 11:15 AM)Fire First Wrote: [\[CITE\]](#) No need to, i am fine with my perfectly working mysql which is barely any slower than mysqli
IIRC, the original MySQL API (mysql_ functions) has the best performance out of the MySQLi API and PDO. Unfortunately though, it is no longer supported, and so general bug fixes (and even security updates) are no longer considered important. The API won't be updated to support any of the new functionality introduced into MySQL (and hasn't been updated to support any of its new features in the past decade or so). So even though it works fine, it is virtually dead and code you write will not be future proof as platforms stop including support for it.

(Feb 18, 2016 03:01 PM)H1gh-T3ch Wrote: [\[CITE\]](#) And i thought mysqli_ was "outdated/old_fashioned" and "not modern" lol people still use mysql_ or someone is being sarcastic xD

The MySQLi API (mysqli_ functions) are absolutely not outdated, nor are they old fashioned. The API is superior to PDO when wanting to really leverage MySQL's features. It's the only MySQL-specific API that directly exposes the capabilities of the MySQL Native Driver (mysqlnd). That's something PDO simply cannot do, because it's not a full-blown abstraction layer - it does not aim to emulate missing functionality across all supported databases, only simply support the subset of functionalities that are common across all databases (i.e. the baseline features). (The exception to this is prepared statements, where it'll emulate them on the client if the DB does not support them, since this is a critical feature for the safety of DB querying.)

itsMeCherry - February 18, 2016, 3:10 pm

(Feb 18, 2016 03:47 PM)Haxalot Wrote: [\[CITE\]](#) IIRC, the original MySQL API (mysql_ functions) has the best performance out of the MySQLi API and PDO. Unfortunately though, it is no longer supported, and so general bug fixes (and even security updates) are no longer considered important. The API won't be updated to support any of the new functionality introduced into MySQL

(and hasn't been updated to support any of its new features in the past decade or so). So even though it works fine, it is virtually dead and code you write will not be future proof as platforms stop including support for it.

The MySQLi API (mysqli_ functions) are absolutely not outdated, nor are they old fashioned. The API is superior to PDO when wanting to really leverage MySQL's features. It's the only MySQL-specific API that directly exposes the capabilities of the MySQL Native Driver (mysqlnd). That's something PDO simply cannot do, because it's not a full-blown abstraction layer - it does not aim to emulate missing functionality across all supported databases, only simply support the subset of functionalities that are common across all databases (i.e. the baseline features). (The exception to this is prepared statements, where it'll emulate them on the client if the DB does not support them, since this is a critical feature for the safety of DB querying.)

And finally we accross a guy who can agree with what i say because he actually knows his facts, unlike the kids who just read and join the hype of "omg mysql is unsecure dont use it". No kids, learn to secure your code and you will be fine.

(Feb 18, 2016 02:05 PM)+Remix Wrote: [\[CITE\]](#) Are you seriously still using mysqli_?????????????
??

(Feb 18, 2016 03:01 PM)H1gh-T3ch Wrote: [\[CITE\]](#) And i thought mysqli_ was
"outdated/old_fashioned" and "not modern" lol people still use mysqli_ or someone is being sarcastic
xD

"Old fashioned", pffft. You clearly learned PHP using online videos last week

SIM-1947090 - February 18, 2016, 3:23 pm

(Feb 18, 2016 04:10 PM)Fire First Wrote: [\[CITE\]](#) "Old fashioned", pffft. You clearly learned PHP
using online videos last week

Imfao didnt know sarcasm doesnt exist on hf, chill

Remix - February 19, 2016, 2:24 am

(Feb 18, 2016 03:47 PM)Haxalot Wrote: [\[CITE\]](#) IIRC, the original MySQL API (mysqli_ functions) has
the best performance out of the MySQLi API and PDO. Unfortunately though, it is no longer
supported, and so general bug fixes (and even security updates) are no longer considered
important. The API won't be updated to support any of the new functionality introduced into MySQL
(and hasn't been updated to support any of its new features in the past decade or so). So even
though it works fine, it is virtually dead and code you write will not be future proof as platforms stop
including support for it.

The MySQLi API (mysqli_ functions) are absolutely not outdated, nor are they old fashioned. The
API is superior to PDO when wanting to really leverage MySQL's features. It's the only
MySQL-specific API that directly exposes the capabilities of the MySQL Native Driver (mysqlnd).
That's something PDO simply cannot do, because it's not a full-blown abstraction layer - it does not
aim to emulate missing functionality across all supported databases, only simply support the subset
of functionalities that are common across all databases (i.e. the baseline features). (The exception
to this is prepared statements, where it'll emulate them on the client if the DB does not support
them, since this is a critical feature for the safety of DB querying.)
is mysql faster than php7 pdo/mysqli?

(Feb 18, 2016 04:23 PM)H1gh-T3ch Wrote: [\[CITE\]](#) Imfao didnt know sarcasm doesnt exist on hf,
chill
acting like it was sarcasm when you realize that you were wrong yy

SIM-1947090 - February 19, 2016, 2:52 am

(Feb 19, 2016 03:24 AM)+Remix Wrote: [\[CITE\]](#) is mysql faster than php7 pdo/mysqli?

acting like it was sarcasm when you realize that you were wrong yy

I dont think i was wrong :)

But if that makes you happy, let it be

(i put it between quotation marks for a reason you dumb fuck kid, cuz i knew there are too many retarded 15 years old kids on this forum/s and i was just waiting for another one)

Remix - February 19, 2016, 3:00 am

(Feb 19, 2016 03:52 AM)H1gh-T3ch Wrote: [\[CITE\]](#) I dont think i was wrong :)

But if that makes you happy, let it be

(i put it between quotation marks for a reason you dumb fuck kid, cuz i knew there are too many retarded 15 years old kids on this forum/s and i was just waiting for another one)

Ok cool story

"people still use mysql_ or someone is being sarcastic"

My mistake

You're more mature than these 15 y/o kids here

Haxalot - February 19, 2016, 3:43 am

(Feb 19, 2016 03:24 AM)+Remix Wrote: [\[CITE\]](#) is mysql faster than php7 pdo/mysqli?

Do you mean running the MySQL extension on PHP 5.x VS running the PDO or MySQL Improved extensions on PHP 7? If so, then the MySQL extension will be less performant, just simply because of the optimisations in PHP 7. If you run the MySQL extension on PHP 7 (it can still be found in PECL), then not sure. It's something that could be benchmarked.