PHP 8 errors that stop old code

The PHP 8 changes that turn old extension code into fatal errors - bare-word constants, static calls, wrong argument types, count() on null, incompatible signatures - with how to find and fix each one.

Source file:
`doc/eng-us/help/extension-development/part-4-upgrading/php-8-errors-that-stop-old-code.md` Last changed 2026-11-04
https://doc.exponential.earth/learninminutes/exphelp/viewer/help/extension-development/part-4-upgrading/php-8-errors-that-stop-old-code.md

Part 4, chapter 4. This chapter is about the changes in PHP 8.0 that turn code which used to run with a warning, or silently, into an `Error` or `TypeError` that stops the request. They are the first thing to fix, because nothing else can be tested while the page is blank. Every example is from a real extension and names the commit that fixed it.

How these errors show up

On a development copy with the debug output on, a fatal error gives you a message, a file and a line. In production it gives you a blank page or the kernel's error page, and the message is in the PHP error log and in `var/log/error.log`. The message always starts with the kind of error:

Message starts with	Meaning
`Uncaught Error: Undefined constant`	a bare word that PHP 7 took for a string
`Uncaught Error: Non-static method ... cannot be called statically`	a static call to an instance method
`Uncaught Error: Call to undefined method`	the method does not exist (often it never did, and the line was never reached before)
`Uncaught TypeError: ... must be of type ..., ... given`	a built-in function got the wrong type
`Uncaught ArgumentCountError`	too few arguments to a function or method
`Fatal error: Declaration of ... must be compatible with ...`	a child class's method does not match its parent

1. Bare words are no longer strings

PHP 7 turned an undefined constant into the string of its name, with a warning. PHP 8 stops. Old `ezinfo.php` files are the most common place. From se7enxweb/expsite_installer, commit 0c771fe:

PHP

```php
// before: Name, Version ... are undefined constants
return array( Name => "expsite_installer",
              Version => "1.0.1",
              info_url => "https://github.com/se7enxweb/expsite_installer" );

// after
return array( 'Name' => "expsite_installer",
              'Version' => "1.0.1",
              'Info_url' => "https://github.com/se7enxweb/expsite_installer" );
```

Note the second fix in the same lines: the key the about page reads is `Info_url`, not `info_url`. Search for bare words in array keys and in `$array[key]` access:

Shell

```shell
grep -rnE '\$[A-Za-z_][A-Za-z0-9_]*\[\s*[A-Za-z_][A-Za-z0-9_]*\s*\]' --include='*.php' .
grep -rnE '^\s*(return array\(\s*)?[A-Za-z_][A-Za-z0-9_]*\s*=>' --include='*.php' .
```

Each hit is a line to look at: a real constant such as `PHP_EOL` or a class constant is fine, a word that is meant as a string needs quotes.

2. Instance methods called statically

PHP 7 allowed `ClassName::method()` on a method without `static`, with a deprecation notice, and `$this` was undefined inside. PHP 8 throws. From se7enxweb/sevenx_themes_media, commit f7d93c0, in a template operator:

PHP

```php
// before: classAttributeContent() is an instance method of eZSelectionType
$classContent = eZSelectionType::classAttributeContent( $classAttr );

// after: ask the class attribute for its content, which calls the datatype for you
$classContent = $classAttr->attribute( 'content' );
```

The general fix is to get an instance first, or, as here, to use the higher-level call the kernel offers. Read the kernel method before you add `static` to anything: if it uses `$this`, it cannot be static.

3. Built-in functions check their argument types

Most built-in functions now throw a `TypeError` when given a type they cannot use. The ones old code hits most:

Call	PHP 7	PHP 8	Fix
`count( null )`, `count( 'text' )`	warning, returned 0 or 1	`TypeError`	`is_countable( $x ) ? count( $x ) : 0`
`method_exists( false, 'm' )`	returned false	`TypeError`	`is_object( $x ) && method_exists( $x, 'm' )`
`implode( $array, ', ' )` (pieces first)	worked	`TypeError`	`implode( ', ', $array )`
`array_merge( $a, null )`	warning, returned null	`TypeError`	`array_merge( $a, $b ?? array() )`

Call	PHP 7	PHP 8	Fix
`in_array( $x, null )`	warning	`TypeError`	check `is_array()` first
`strlen( array() )`	warning	`TypeError`	find out why an array arrived

From se7enxweb/explayouts_ui_api, commit 2a95edc: a block whose definition had no settings left the handler `false`, and `method_exists()` stopped the whole request, so the editor's sidebar loaded forever:

PHP

```
// before
$itemViewTypes = method_exists( $handler, 'getItemViewTypes' ) ? $handler->getItemViewTypes() : a
rray();

// after
$itemViewTypes = is_object( $handler ) && method_exists( $handler, 'getItemViewTypes' )
    ? $handler->getItemViewTypes()
    : array();
```

`implode()` with its arguments the old way round was found in the kernel's OAuth module, in ezstarrating and in swark during the PHP 8.0 review (PHP 8.0 support).

4. Too few arguments

Calling a user function with fewer arguments than it requires was a warning; it is an `ArgumentCountError` now. So is calling a built-in function without an argument it needs: `crypt()` without a salt was one of the findings in the kernel's LDAP user class, fixed with a random password from `random_bytes()`.

The opposite also matters: a function that gained an argument in a later PHP is an `ArgumentCountError` on the older one. `fputcsv()` with its `$eol` argument (added in PHP 8.1) stops on PHP 8.0, so the kernel branches on `PHP_VERSION_ID` there.

5. Methods that do not match their parent

PHP 8 checks every method that overrides a parent's method when the class loads. A mismatch is a fatal error even if the method is never called. Two from the PHP 8.0 review:

Extension	Class and method	Problem	Fix
syndication	`eZSyndicateType::onPublish()`	parameters by reference, the parent's by value	& removed
ezstarrating	`ezsrRatingObjectTreeNode::makeObjectsArray()`	fewer parameters than `eZContentObjectTreeNode::makeObjectsArray()`	parameters added

To fix one, open the parent class in the kernel and copy its signature exactly: the same parameters, the same defaults, the same by-reference marks. You may add optional parameters at the end; you may not remove any.

6. Methods that never existed

Old code sometimes calls a method that does not exist, on a branch that was never reached, or reached only with a warning swallowed by `@`. PHP 8 makes the branch fatal. From se7enxweb/explayouts_site_api, commit 031d380:

PHP

```php
// before: eZURLAliasML::fetchByNodeID() does not exist
return '/' . eZURLAliasML::fetchByNodeID( $value->mainLocationId() );

// after
$node = $value->mainLocationId() > 0 ? eZContentObjectTreeNode::fetch( (int)$value->mainLocationI
d() ) : null;
return $node instanceof eZContentObjectTreeNode ? '/' . $node->attribute( 'url_alias' ) : '';
```

Whenever you see a kernel class name in old code, check that the method exists before you trust it:

Shell

```shell
grep -n "function fetchByNodeID" kernel/classes/ezurlaliasml.php
```

7. Removed functions

Calling a function PHP removed (`each()`, `create_function()`, `split()`, `ereg()`, `mysql_query()` ...) is `Call to undefined function`. They have their own chapter: Removed PHP functions and their replacements.

8. Constructors that are not called

This one is not an error at all, which makes it worse: PHP 8 no longer runs a method named after its class as the constructor, so the object is built without being set up, and the error comes later, somewhere else. See Old-style constructors and static calls.

A test that finds most of them

Run each entry point once with the debug output on and every error reported:

Shell

```shell
php -d error_reporting=-1 -d display_errors=1 bin/php/ezexec.php extension/myoldext/bin/php/somes
cript.php
```

For web pages, open each one with the debug output on. For classes, a small script that only loads them finds signature mismatches at once:

PHP

```php
<?php
// extension/myoldext/tests/load_all_classes.php: run with php bin/php/ezexec.php
$autoload = include 'var/autoload/ezp_extension.php';
foreach ( $autoload as $class => $file )
{
    if ( strpos( $file, 'extension/myoldext/' ) === 0 && !class_exists( $class ) && !interface_ex
ists( $class ) )
    {
        echo "Not loaded: $class ($file)\n";
    }
}
echo "Done\n";
```

Loading a class compiles all its methods, so every "must be compatible" error appears here, whether the method is ever called or not.

Related pages

- Previous: Taking inventory of an old extension
- Next: PHP 8.1 to 8.5 deprecations
- PHP 8.0 support
- PHP 8.0 incompatible changes in the PHP manual
- Debugging extensions
- Free documentation: this page is free to copy, translate and improve under the GNU FDL 1.3.

Become an Exponential Partner

Partners share their information, provide funding, contribute back and develop the products with us, and that is how Exponential grows. A partner website and dashboard are planned for mid-2027: please become a partner today to speed up our growth rate and land new features faster.

Partners: start here | How to apply

Exponential Copyright © 1998-2026 Exponential Foundation and 7x
Permission is granted to copy, distribute and/or modify this document under the terms of the GNU Free Documentation License, Version 1.3 or any later version published by the Free Software Foundation; with no Invariant Sections, no Front-Cover Texts, and no Back-Cover Texts. A copy of the license is included in the section entitled "GNU Free Documentation License".
This documentation is free documentation, licensed under the GNU Free Documentation License 1.3 or later:
https://www.gnu.org/licenses/fdl-1.3.html.