The coding standard (phpcs.xml.dist)

How PHP code in Exponential is written and checked - every rule of phpcs.xml.dist with an example, the forbidden functions, the PHP 8.0 compatibility check, how only new violations fail, and the commands that run the...

Source file: `doc/eng-us/help/php-developers/the-coding-standard.md` Last changed 2026-11-04
https://doc.exponential.earth/learninminutes/exphelp/viewer/help/php-developers/the-coding-standard.md

This page is for PHP developers who write code for the kernel or for extensions that follow the kernel's style. The style is described in `phpcs.xml.dist` and checked by PHP_CodeSniffer. Each rule was measured against the existing kernel and library code before it was added, so the standard describes how the code already looks; it is not PSR-12 (Coding standards).

Contents

1. An example in the house style
2. The rules
3. Forbidden functions
4. Type declarations and PHP 8.0
5. Only new violations fail
6. Running the check
7. Beyond what the tool checks

An example in the house style

PHP

```php
<?php
/**
 * File containing the acmeReportService class.
 *
 * @package acmereport
 */

class acmeReportService
{
    /**
     * The totals of the items, by status.
     *
     * @param array $items list of acmeReportItem
     * @return array status => total
     */
    public static function totals( array $items )
    {
        $totals = array();
        foreach ( $items as $item )
        {
            $status = (string)$item->attribute( 'status' );
            if ( !isset( $totals[$status] ) )
            {
                $totals[$status] = 0;
            }
```

```
            $totals[$status] += (int)$item->attribute( 'amount' );
        }
        return $totals;
    }
}
```

The rules

Rule	Right	Wrong
Long array syntax (`Generic.Arrays.DisallowShortArraySyntax`)	`array( 1, 2 )`	`[ 1, 2 ]`
One space inside control structure parentheses	`if ( $a )`, `foreach ( $list as $x )`	`if ($a)`
One space inside call parentheses, none before `(`	`foo( $a, $b )`	`foo($a, $b)`, `foo ( $a )`
One space inside declaration parentheses, `=` with spaces	`function foo( $a, $b = 1 )`	`function foo($a, $b=1)`
Function brace on its own line (`OpeningFunctionBraceBsdAllman`)	`function foo()` then `{` on the next line	`function foo() {`
No space after `!`	`!$a`	`! $a`
Lower case keywords and constants	`true`, `false`, `null`, `foreach`	`TRUE`, `NULL`
Full open tag	`<?php`	`<?`
Unix line endings, spaces not tabs, no trailing spaces		

The layout of a call over several lines is free; only the spaces inside the parentheses are checked.

Forbidden functions

`Generic.PHP.ForbiddenFunctions` refuses functions that PHP removed or that Exponential replaces:

Forbidden	Use
`each`	`foreach`
`create_function`	a closure
`split`	`explode` or `preg_split`
`ereg`, `eregi`	`preg_match`
`ereg_replace`	`preg_replace`
`mysql_query`, `mysql_connect`	eZDB
`sizeof`	`count`
`delete`	`unset`

Call-time pass by reference (`foo( &$a )`) is refused too.

Type declarations and PHP 8.0

Two more parts belong to the standard:

- **No type declarations of PHP 7 and later**, checked by the project's own sniff `bin/phpcs/Exponential/Sniffs/TypeDeclarations/NoTypeDeclarationsSniff.php`: no `declare( strict_types=1 )`, no scalar, pseudo, union or intersection parameter types, no return, property or constant types. Allowed are the hints PHP 5 had: `array`, `callable`, `self` and class names, also nullable (`?array`, `?eZINI`). Files under `tests/` may declare the `: void` PHPUnit requires. The reasons are on Why the code has no type declarations.
- **PHPCompatibility** with `testVersion 8.0-`: no syntax or functions newer than PHP 8.0, and none that later versions removed or deprecate (Writing code that runs on PHP 8.0).

Only new violations fail

Old files do not follow every rule, and the project never reformats old code for style. `bin/php/checkcodestyle.php` therefore compares the number of violations of each changed file with its previous version and fails only when your change adds violations. You may leave the old style of the lines you did not touch alone; you may not add to it.

Running the check

Shell

```
make devtools                                          # once: PHP_CodeSniffer and PHPCompatibility
 into .devtools/
make phpcs                                             # changed files since origin/main; only new
violations fail
make phpcs BASE=origin/release-branch                  # against another base
php bin/php/checkcodestyle.php --staged                # staged files against HEAD (what the pre-co
mmit hook runs)
.devtools/vendor/bin/phpcs --standard=phpcs.xml.dist extension/acmereport/   # a whole directory,
 all findings
```

In CI the code style step is a warning, not a failure, for now; the pre-commit hook refuses a commit that adds violations unless you use `--no-verify`. The sniff has its own tests: `vendor/bin/phpunit --testsuite quality` (Git hooks and make targets).

Beyond what the tool checks

These are the conventions of the code base that no sniff checks; reviewers do:

- One class per file, file name in lower case (`acmereportservice.php` for `acmeReportService`).
- A docblock on each class and public method with `@param` and `@return` types, since the code has no type declarations (the example above shows the form).
- New identifiers carry your prefix; new project code uses `exp`; kernel `eZ*` names are never renamed (Class naming and the exp prefix).
- `eZDebug::writeError()` and friends with `__METHOD__` as the label.
- American English in comments and messages.

Related pages

- Coding standards
- Quality checks
- Static analysis with PHPStan
- 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.