Static analysis and code style

How Exponential checks code without running it - PHPStan at level 1 against a baseline of inherited findings, PHP_CodeSniffer with the project's own style and sniffs (no PHP 7+ type declarations, PHP 8.0 compatibility)...

Source file: `doc/eng-us/help/software-engineers/static-analysis-and-code-style.md` Last changed 2026-11-04
https://doc.exponential.earth/learninminutes/exphelp/viewer/help/software-engineers/static-analysis-and-code-style.md

This page is for engineers who run, review or improve the static checks. The approach suits an old, large code base: the checks were introduced without reformatting existing code, and they fail only on what a change adds. The reference is Quality checks.

Contents

1. Setting up the tools
2. PHPStan and its baseline
3. The code style
4. Why no type declarations
5. Only what a change adds
6. Improving the baseline

Setting up the tools

Shell

```
make devtools   # PHP_CodeSniffer and PHPStan into .devtools/ (composer.dev.json), never into vendor/
make hooks      # git config core.hooksPath .githooks
make check      # lint, manifest-check, phpcs and phpstan, as CI does
```

The development tools live in `.devtools/`, resolved from `composer.dev.json` for PHP 8.0.30, so they never change the installation's `vendor/`.

PHPStan and its baseline

Setting	**Value (`phpstan.neon.dist`)**
Level	1
PHP version	8.1 (`phpVersion: 80100`)
Paths	`kernel`, `lib`
Excluded	`vendor/*`, `var/*`, tests, the command classes in `kernel/private/classes/commands/` (they declare global helper functions), `dfs_filter_iterator.php` (needs tentative return types first)
Baseline	`phpstan-baseline.neon`: the findings the code had when the check was introduced

Only findings missing from the baseline fail. `make phpstan` runs it; the pre-push hook runs it when the tools are installed; CI runs it on PHP 8.3 and blocks.

The code style

`phpcs.xml.dist` describes the Exponential 6.0.15 style, each rule measured against the existing code before it was added:

- `array( ... )` instead of `[ ... ]`;
- one space inside the parentheses of control structures, calls and declarations: `if ( $a )`, `foo( $a, $b );`
- the opening brace of a function on its own line;
- `!$a` without a space; lower case keywords and constants;
- Unix line endings, spaces instead of tabs, no trailing spaces;
- no functions removed from PHP.

PHPCompatibility checks for `testVersion 8.0-`: no syntax or functions newer than PHP 8.0, unless guarded by `function_exists()` or provided in `lib/phpcompat.php` (PHP 8.0 support). The project's own sniffs live in `bin/phpcs/Exponential/` and have tests (`php vendor/bin/phpunit --testsuite quality`).

In CI the code style is a warning for now; the pre-commit hook refuses a commit that adds violations.

Why no type declarations

The sniff `bin/phpcs/Exponential/Sniffs/TypeDeclarations/` reports `declare( strict_types=1 )`, scalar, pseudo and union parameter types, return types, property types and constant types. The parameter hints PHP 5 already had stay allowed: `array`, `callable`, `self` and class names, also nullable (`?array`, `?eZINI`). Files under `tests/` may declare the `: void` PHPUnit requires.

The reason is compatibility: the kernel must run on PHP 8.0, and adding types to existing signatures would break every extension that overrides or extends them. Engineers used to strictly typed PHP should document types in docblocks instead, which PHPStan reads.

Only what a change adds

Check	How it compares
Code style	the number of violations of each changed file against its previous version
PHPStan	the findings against `phpstan-baseline.neon`
Syntax	every changed file, with PHP 8.3 and PHP 8.0

A change that touches an old file is not asked to fix the whole file; it must not make it worse.

Shell

```
make lint                # php -l on the PHP files changed since origin/main
make lint80              # the same with PHP 8.0
make phpcs BASE=origin/main
make phpstan
```

Improving the baseline

Raise the PHPStan level or shrink the baseline in small, separate changes, and regenerate it with `make phpstan-baseline`. Never regenerate the baseline to make a new finding pass: that hides the finding as if it had always been there. The same goes for code style: fix the code rather than weakening a sniff.

Related pages

- Coding standards
- PHP, PHP-FIG and Composer standards
- Handling technical debt
- 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.