Coding standards

The code style of Exponential and how it is checked - the phpcs ruleset, PHPCompatibility for PHP 8.0, PHPStan with a baseline, PHP 8.0 lint, file headers, naming, and the template, CSS and JavaScript rules.

Source file: `doc/eng-us/standards/coding-standards.md` Last changed 2026-11-04
https://doc.exponential.earth/exphelp/viewer/standards/coding-standards.md

This page is for developers who change the kernel, its templates or its extensions, and for reviewers who check such changes. It answers: what the code style is, which tools check it and how strictly, how files are headed and named, and which rules apply to templates, stylesheets and scripts. The day-to-day use of the tools is described in Quality checks.

Summary

Standard	Version	Status	Evidence (short)
Exponential code style (PHP_CodeSniffer)	`phpcs.xml.dist`, "Exponential 6.0.15 code style"	**Partly complies**	only new violations fail; warning only in CI
PHP 8.0 compatibility (PHPCompatibility)	`testVersion 8.0-`	**Complies**	`phpcs.xml.dist`, CI job `php80` lints every file with PHP 8.0
Static analysis (PHPStan)	PHPStan 2.1, level 1	**Partly complies**	`phpstan.neon.dist`, baseline of about 3,000 findings, blocking in CI
PHP syntax (`php -l`)	PHP 8.3 and 8.0	**Complies**	`.githooks/pre-commit`, `quality.yml`, `make lint80-all`
File header docblock	project header	**Partly complies**	1,454 of 1,862 PHP files in `kernel/` and `lib/` carry the current copyright line
Naming (sevenx and exp prefixes, file = lower-case class name)	written rule	**Complies** for new code	375 `exp*.php` files in `kernel/` and `lib/`
No inline CSS in admin templates	project rule	**Partly complies**	`tests/tests/design/AdminStylingRulesTest.php` with a shrinking allowlist
No inline script (Content Security Policy)	CSP Level 3 (W3C)	**Partly complies**	`AdminContentSecurityPolicyTest.php`, `ExtensionInlineScriptsTest.php`
JavaScript syntax	ECMAScript, Node.js 22 `--check`	**Complies** for syntax	`quality.yml` step "JavaScript syntax of changed files"
EditorConfig	-	**Not applicable**	no `.editorconfig` in the repository

The PHP code style: `phpcs.xml.dist`

What it is. PHP_CodeSniffer (`squizlabs/php_codesniffer` `^3.13`, from `composer.dev.json`) checks code against a ruleset. The repository's ruleset is `phpcs.xml.dist`, named

"Exponential" and described as "Exponential 6.0.15 code style". It does not extend a whole standard (not PSR-12, not PEAR); it picks single sniffs. The comment at its top says every rule "was measured against the code before it was added".

What it checks.

Rule	Sniff	Meaning
Long array syntax	`Generic.Arrays.DisallowShortArraySyntax`	`array( ... )`, never `[ ... ]`
Spaces in control structures	`PSR2.ControlStructures.ControlStructureSpacing` (1 space after `(` and before `)`)	`if ( $a )`, `foreach ( $list as $item )`
Spaces in calls	`PEAR.Functions.FunctionCallSignature` (1 space inside; indent, closing bracket line, content after the open bracket, opening indent, multiple arguments per line and empty lines excluded)	`foo( $a, $b )`, no space before `(`; multi-line layout is free
Spaces in declarations	`Squiz.Functions.FunctionDeclarationArgumentSpacing` (1 space inside, 1 around `=`)	`function foo( $a, $b = 1 )`
Function braces	`Generic.Functions.OpeningFunctionBraceBsdAllman`	the `{` of a function on its own line
Not operator	`Generic.Formatting.SpaceAfterNot` (spacing 0)	`!$a`
Lower case	`Generic.PHP.LowerCaseConstant`, `Generic.PHP.LowerCaseKeyword`	`true`, `false`, `null`, keywords
Files	`Generic.PHP.DisallowShortOpenTag`, `Generic.Files.LineEndings`, `Generic.WhiteSpace.DisallowTabIndent`, `Squiz.WhiteSpace.SuperfluousWhitespace.EndLine`	`<?php`, Unix line endings, spaces not tabs, no trailing spaces
Forbidden functions	`Generic.PHP.ForbiddenFunctions`	`each`, `create_function`, `split`, `ereg`, `eregi`, `ereg_replace`, `mysql_query`, `mysql_connect`, `sizeof`, `delete`, each with the replacement named (`foreach`, a closure, `preg_*`, `eZDB`, `count`, `unset`)
Call-time pass by reference	`Generic.Functions.CallTimePassByReference`	not allowed
No type declarations	`bin/phpcs/Exponential/Sniffs/TypeDeclarations/NoTypeDeclarationsSniff.php`	no `declare( strict_types=1 )`, no scalar, union, return, property or constant types; the PHP 5 hints `array`, `callable`, `self`, class names (also nullable) stay allowed; tests may declare `: void`
PHP 8.0 compatibility	`PHPCompatibility` with `testVersion 8.0-`	no syntax or function newer than PHP 8.0 unless guarded, no removed or deprecated functions

Excluded paths: `*/vendor/*`, `*/.devtools/*`, `*/var/*`, `*/node_modules/*`. Only `.php` files are checked.

How it is enforced.

- `bin/php/checkcodestyle.php` runs phpcs on changed files only and compares the number of violations of each file with its previous version. **Only a change that adds violations**

fails, so editing an old file does not require reformatting it.

  - `--staged` in the pre-commit hook (`.githooks/pre-commit`), `--base=<ref>` for `make phpcs` and CI.
  - Without the development tools (`make devtools`) it prints a hint and passes.

- In CI (`.github/workflows/quality.yml`, step "Code style of changed files (warning only)") it has `continue-on-error: true`: **a violation is a warning, not a failure**.
- The project's own sniff has a test: `tests/tests/bin/phpcs/NoTypeDeclarationsSniffTest.php`, run by the blocking CI job "Code style sniff tests" (`phpunit --testsuite quality`).

Status: Partly complies. New code is held to the style locally; CI does not block on it, and old code is not reformatted.

The kernel's legacy style in short

The style comes from the code base inherited from eZ Publish 4 and is kept for consistency:

PHP

```php
class expExampleThing
{
    /**
     * Returns the names of the items, sorted.
     *
     * @param array $items
     * @return array
     */
    public static function sortedNames( $items )
    {
        $names = array();
        foreach ( $items as $item )
        {
            if ( !$item->attribute( 'is_hidden' ) )
            {
                $names[] = $item->attribute( 'name' );
            }
        }
        sort( $names );
        return $names;
    }
}
```

- Braces on their own line for classes, functions and control structures (only the function brace is checked).
- Four spaces of indentation.
- Types are documented in docblocks (`@param`, `@return`), not declared.

Static analysis: PHPStan

What it is. PHPStan (`phpstan/phpstan` `^2.1`, from `composer.dev.json`) finds type and logic errors without running the code.

How it is configured (`phpstan.neon.dist`):

- `level: 1` (of 0 to 10), `phpVersion: 80100`.

- `paths`: `kernel` and `lib`. Excluded: `vendor/*`, `var/*`, `*/tests/*`, `kernel/private/classes/commands/*` (scripts that declare global helper functions) and `kernel/private/classes/clusterfilehandlers/dfsbackends/dfs_filter_iterator.php` (missing tentative return types).
- `bootstrapFiles`: `autoload.php`; `treatPhpDocTypesAsCertain: false`.
- `phpstan-baseline.neon` holds the findings the code had when the check was introduced: 1,271 entries with a total count of 3,026 (summed for this page; Quality checks gives 3,033). **Only findings missing from the baseline fail.**

How it is enforced. `make phpstan`; the `pre-push` hook (`.githooks/pre-push`) when the development tools are installed; the blocking CI job `phpstan` (PHP 8.3) in `quality.yml`. `make phpstan-baseline` regenerates the baseline.

Status: Partly complies. Level 1 is a low level, and the baseline accepts about 3,000 existing findings.

PHP syntax: `php -l` and PHP 8.0

- `.githooks/pre-commit` runs `php -l` on every staged PHP file as staged, and also with PHP 8.0 when a PHP 8.0 binary is found (`EXP_PHP80`, `php8.0`, `php80` or the PHP 8.0 binary of a hosting panel).
- `make lint` and `make lint80` check the files changed since `origin/main`; `make lint80-all` checks every PHP file in git with PHP 8.0.
- CI: `quality.yml` lints changed files with PHP 8.3 (job `checks`) and every file with PHP 8.0 (job `php80`, `make lint80-all PHP80=php`), both blocking.

Status: Complies. See PHP 8.0 support for how newer PHP functions are guarded.

File header docblock

New PHP files start with a docblock that names the file's content, the copyright, the license and the package:

PHP

```php
<?php
/**
 * File containing the expHelpAudience class.
 *
 * @copyright Copyright (C) 1998 - 2026 Exponential Foundation and 7x. All rights reserved.
 * @license GNU General Public License v2.0 (or any later version)
 * @package kernel
 */
```

- Counted for this page in `kernel/` and `lib/` (1,862 PHP files): 1,454 carry the line `Copyright (C) 1998 - 2026 Exponential Foundation and 7x. All rights reserved.`, 1,818 carry `@package`, 1,041 still carry an inherited `eZ Systems AS` copyright line next to it, and 1,058 still carry the placeholder `@version //autogentag//`.
- The `@license` line has two common forms: "GNU General Public License v2.0 (or any later version)" (684 files) and the older "For full copyright and license information view LICENSE file distributed with this source code" (1,142 files). Both point to the GPL-2.0-or-later license of `composer.json`.

- Inherited copyright lines are kept: they record who wrote the original code.

The old header template: header.txt

header.txt is the file header of the 4.x era, kept for reference. It is a block of `//` comments with `SOFTWARE NAME`, `SOFTWARE RELEASE: 4.1.x`, two `COPYRIGHT NOTICE` lines and the full GPL v2.0 notice. **Do not use it for new files**; use the docblock above. Old files that still have this form keep it.

Status: Partly complies. No tool checks the header; the counts above show the mix.

Naming conventions

The prefix rule is stated in full in Naming prefixes: sevenx and exp: `sevenx` is 7x's official public PHP code prefix (an identifier cannot start with a digit, so "7x" cannot be one), extensions and classes from 7x use `sevenx` or `exp` as 7x needs, and `exp` is used with intent to build the Exponential brand and to assign ownership to the Exponential project and the Exponential Foundation.

- **New code uses the `exp` prefix**: classes `expFoo` (`expHelpAudience`, `expContentSecurityPolicy`), template operators (`kernel/common/expcontentsecuritypolicyoperator.php`), templates (`design/standard/templates/parts/exp_content_direction.tpl`), modules and settings. 375 files in `kernel/` and `lib/` are named `exp*.php`.
- **Existing identifiers are never renamed**: `eZContentObject`, `eZINI`, `eZPersistentObject`, `ezpKernelWeb` and the `ez*` file names stay (667 `ez*.php` files in `kernel/` and `lib/`). Renaming would break every extension.
- **File name = lower-case class name**: `expHelpAudience` lives in `kernel/classes/help/exphelpaudience.php`, `eZMySQLCharset` in `lib/ezdb/classes/ezmysqlcharset.php`. The autoload generator finds classes by scanning, so this is a convention, not a technical requirement.
- **Templates** are lower case with underscores, `.tpl` (no template in `design/` has an upper-case letter in its path, checked for this page); override templates live under `override/templates/`.
- **INI files**: the defaults are `settings/<name>.ini`; overrides are `<name>.ini.append.php` in `settings/override/`, `settings/siteaccess/<siteaccess>/` or an extension's `settings/`. Sections are `[CamelCase]`, settings `CamelCase`, lists `Name[]`.

Status: Complies for new code. The rule is written down in Naming prefixes: sevenx and exp.

Templates, stylesheets and scripts

No inline CSS

The rules are in Styling the administration and `CONTRIBUTING.md`:

- no `<style>` block and no `style=""` attribute in a template of an administration design;
- the shared components (`.a4-card`, `.a4-btn`, `.a4-table` ...) of `stylesheets/components.css` first;
- colors, radius, spacing and fonts only from the `--a4-*` tokens, and every color token has a

value for `html[data-a4-theme="dark"]`;

- module CSS only in `stylesheets/modules/<group>.css`.

`tests/tests/design/AdminStylingRulesTest.php` (suite `design`, no database) enforces them for admin4, admin4l, admin3 and admin. Files still being migrated are on an **allowlist that only shrinks**: a listed file that is clean fails until it is removed from the list. `tests/tests/design/StandardDatatypeEditNoStyleAttributeTest.php` covers the standard design's datatype edit templates, and the suite `admin-extensions` covers the admin templates of `git_manager`, `ezupdate` and `sevenx_dse`.

Status: Partly complies (the allowlist is not empty yet).

No inline script: the Content Security Policy

What it is. Content Security Policy Level 3 (W3C) lets a site forbid inline scripts and styles.

- `settings/site.ini [ContentSecurityPolicy]` configures it (`Mode=auto`, `Nonce=enabled`, `NonceDirectives[]=script-src`); see Content Security Policy.
- `kernel/private/classes/expcontentsecuritypolicy.php` builds the header; a per-request nonce is available to templates through the operator `csp_nonce()` for an inline script that cannot be avoided.
- `tests/tests/design/AdminContentSecurityPolicyTest.php` checks that the templates admin4 and admin4l render carry no inline script, no `on...=` handler, no `javascript:` URL, no `style=""` and no `<style>`.
- `tests/tests/design/ExtensionInlineScriptsTest.php` holds listed extensions to the same rule for scripts, with a shrinking list of known findings.

Status: Partly complies. Some older designs stay report-only (`ReportOnlyDesigns[]`), and site designs are not covered by the tests.

JavaScript

- Syntax: `node --check` on changed `.js` files (not `*.min.js`) in the pre-commit hook when Node.js is installed and in CI (Node.js 22).
- Design tests run admin4 scripts in Node.js against `tests/tests/design/js/mini_dom.js`.
- There is no ESLint, Prettier or Stylelint configuration in the repository.

Status: Complies for syntax; there is no style or lint rule set for JavaScript or CSS beyond the design tests.

EditorConfig

There is no `.editorconfig` file in the repository. **Status: Not applicable.** Editors must be set to four spaces, Unix line endings and UTF-8 by hand; phpcs checks the first two for PHP.

Known gaps

1. Code style is a warning in CI, not a failure.
2. PHPStan runs at level 1 with about 3,000 accepted findings.
3. File headers are mixed (old `@license` wording, `@version //autogentag//`, the 4.x `//` header in old files).

4. `CONTRIBUTING.md` links a `CODING_STYLE.md` that does not exist.
5. No JavaScript or CSS linter; no `.editorconfig`.
6. The inline CSS and inline script rules still have allowlists.

Related pages

- Naming prefixes: sevenx and exp
- Quality checks
- Styling the administration
- Content Security Policy
- PHP, PHP-FIG and Composer standards
- Testing and continuous integration

Copyright and copyleft

Since 29 October 2026 the copyright of Exponential is held by the Exponential Foundation and 7x together, the Foundation named first: "Exponential Copyright © 1998-2026 Exponential Foundation and 7x", in files "Copyright (C) 1998 - 2026 Exponential Foundation and 7x. All rights reserved." Who holds it and why, what the GPL lets you do, the third-party notices that stay and how to write the notice in new files and extensions: Copyright and copyleft.

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.