**1. Overview**

* Files MUST use only <?php and <?= tags.
* Files MUST use only UTF-8 without BOM for PHP code.
  + (standard unicode character set)
* Files SHOULD *either* declare symbols (classes, functions, constants, etc.) *or* cause side-effects (e.g. generate output, change .ini settings, etc.) but SHOULD NOT do both.
* Namespaces and classes MUST follow an “autoloading” PSR: [[PSR-0](https://github.com/php-fig/fig-standards/blob/master/accepted/PSR-0.md), [PSR-4](https://github.com/php-fig/fig-standards/blob/master/accepted/PSR-4-autoloader.md)].
* Class names MUST be declared in StudlyCaps.
* Class constants MUST be declared in all upper case with underscore separators.
* Method names MUST be declared in camelCase.

**2. Files**

PHP code MUST use the long <?php ?> tags or the short-echo <?= ?> tags; it MUST NOT use the other tag variations.

PHP code MUST use only UTF-8 without BOM.

A file SHOULD declare new symbols (classes, functions, constants, etc.) and cause no other side effects, or it SHOULD execute logic with side effects, but SHOULD NOT do both.

The phrase “side effects” means execution of logic not directly related to declaring classes, functions, constants, etc., *merely from including the file*.

“Side effects” include but are not limited to: generating output, explicit use of require or include, connecting to external services, modifying ini settings, emitting errors or exceptions, modifying global or static variables, reading from or writing to a file, and so on.

All PHP files MUST use the Unix LF (linefeed) line ending only.

All PHP files MUST end with a non-blank line, terminated with a single LF.

The closing ?> tag MUST be omitted from files containing only PHP.

Lines SHOULD NOT be longer than 80 characters; lines longer than that SHOULD be split into multiple subsequent lines of no more than 80 characters each.

There MUST NOT be trailing whitespace at the end of lines.

Blank lines MAY be added to improve readability and to indicate related blocks of code except where explicitly forbidden.

There MUST NOT be more than one statement per line.

Code MUST use an indent of 4 spaces for each indent level and MUST NOT use tabs for indenting.

**3. Namespace and Class Names**

Namespaces and classes MUST follow an “autoloading” PSR: [[PSR-0](https://github.com/php-fig/fig-standards/blob/master/accepted/PSR-0.md), [PSR-4](https://github.com/php-fig/fig-standards/blob/master/accepted/PSR-4-autoloader.md)].

This means each class is in a file by itself, and is in a namespace of at least one level: a top-level vendor name.

Class names MUST be declared in StudlyCaps.

Code written for PHP 5.3 and after MUST use formal namespaces.

The header of a PHP file may consist of a number of different blocks. If present, each of the blocks below MUST be separated by a single blank line, and MUST NOT contain a blank line. Each block MUST be in the order listed below, although blocks that are not relevant may be omitted.

* Opening <?php tag.
* File-level docblock.
* One or more declare statements.
* The namespace declaration of the file.
* One or more class-based use import statements.
* One or more function-based use import statements.
* One or more constant-based use import statements.
* The remainder of the code in the file.

**4.Classes, Properties, and Methods**

Whatever naming convention is used SHOULD be applied consistently within a reasonable scope. That scope may be vendor-level, package-level, class-level, or method-level.

Any closing brace MUST NOT be followed by any comment or statement on the same line.

The extends and implements keywords MUST be declared on the same line as the class name.

The opening brace for the class MUST go on its own line; the closing brace for the class MUST go on the next line after the body.

Opening braces MUST be on their own line and MUST NOT be preceded or followed by a blank line.

Closing braces MUST be on their own line and MUST NOT be preceded by a blank line.

Visibility MUST be declared on all methods.

When making a method or function call, there MUST NOT be a space between the method or function name and the opening parenthesis, there MUST NOT be a space after the opening parenthesis, and there MUST NOT be a space before the closing parenthesis. In the argument list, there MUST NOT be a space before each comma, and there MUST be one space after each comma.

**5. Control Structures**

The general style rules for control structures are as follows:

* There MUST be one space after the control structure keyword
* There MUST NOT be a space after the opening parenthesis
* There MUST NOT be a space before the closing parenthesis
* There MUST be one space between the closing parenthesis and the opening brace
* The structure body MUST be indented once
* The body MUST be on the next line after the opening brace
* The closing brace MUST be on the next line after the body

The body of each structure MUST be enclosed by braces. This standardizes how the structures look and reduces the likelihood of introducing errors as new lines get added to the body.

**Example types:**

* Class names:
  + ThisIsAClass = StudleyCaps
* Methods names
  + thisIsAMethod = camelCase
* Class constant must be declared in uppercase with underscore seperators
  + CLASS\_CONSTANT

**Logging:**

* Levels of logging
  + The logger interface exposes eight methods to write logs to the eight levels:
    - Debug
    - Info
    - Notice
    - Warning
    - Error
    - Critical
    - Alert
    - Emergency
  + We should NOT create custom levels.
* Messages
  + Every method accepts a string as the message, or an object with a \_\_ToString().
  + The message may contain placeholders which may be replaced with values from the context array
    - Placeholder names must correspond to keys in the context array