http://www.php-fig.org/psr/psr-1,2,3&4

Basic Coding Standard PSR-1

- Files MUST use only `<?php` and `<?=` tags.
- Files MUST use only UTF-8 without BOM for PHP code.
- 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, PSR-4].
- 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`.

Basic Coding Standard PSR-2

- Code MUST use 4 spaces for indenting, not tabs.
- There MUST NOT be a hard limit on line length; the soft limit MUST be 120 characters; lines SHOULD be 80 characters or less
- There MUST be one blank line after the `namespace` declaration, and there MUST be one blank line after the block of `use` declarations.
- The closing `?>` tag MUST be omitted from files containing only PHP.
- All PHP files MUST end with a single blank line.
- There MUST NOT be more than one statement per line.
- Code MUST use an indent of 4 spaces, and MUST NOT use tabs for indenting.
- N.b.: Using only spaces, and not mixing spaces with tabs, helps to avoid problems with diffs, patches, history, and annotations. The use of spaces also makes it easy to insert fine-grained sub-indentation for inter-line alignment.
- PHP keywords MUST be in lower case.
- Property names SHOULD NOT be prefixed with a single underscore to indicate protected or private visibility.

- Method names SHOULD NOT be prefixed with a single underscore to indicate protected or private visibility.
-
- The `var` keyword MUST NOT be used to declare a property.
-
- Argument lists MAY be split across multiple lines, where each subsequent line is indented once. When doing so, the first item in the list MUST be on the next line, and there MUST be only one argument per line.(argument across multiple line)
- When the argument list is split across multiple lines, the closing parenthesis and opening brace MUST be placed together on their own line with one space between them.

```
<?php
namespace Vendor\Package;

class ClassName
{
    public function aVeryLongMethodName(
        ClassTypeHint $arg1,
        &$arg2,
        array $arg3 = []
    ) {
        // method body
    }
}
```

- 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.

```
<?php
bar();
$foo->bar($arg1);
Foo::bar($arg2, $arg3);
```

- Argument lists MAY be split across multiple lines, where each subsequent line is indented once. When doing so, the first item in the list MUST be on the next line, and there MUST be only one argument per line.

```
<?php
$foo->bar(
    $longArgument,
    $longerArgument,
    $muchLongerArgument
);
```

- 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 closing brace MUST be on the next line after the body

```php
<?php
if ($expr1) {
    // if body
} elseif ($expr2) {
    // elseif body
} else {
    // else body;
}
```

```php
<?php
switch ($expr) {
    case 0:
        echo 'First case, with a break';
        break;
    case 1:
        echo 'Second case, which falls through';
        // no break
    case 2:
    case 3:
    case 4:
        echo 'Third case, return instead of break';
        return;
    default:
        echo 'Default case';
        break;
}
```

```php
<?php
while ($expr) {
    // structure body
}
```

```php
<?php
do {
    // structure body;
} while ($expr);
```

```php
<?php
for ($i = 0; $i < 10; $i++) {
    // for body
}
```

```php
<?php
foreach ($iterable as $key => $value) {
    // foreach body
}
```

```php
<?php
try {
    // try body
} catch (FirstExceptionType $e) {
    // catch body
} catch (OtherExceptionType $e) {
    // catch body
}
```

- 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.

```php
<?php
namespace Vendor\Package;

use FooClass;
use BarClass as Bar;
use OtherVendor\OtherPackage\BazClass;

class ClassName extends ParentClass implements \ArrayAccess, \Countable
{
    // constants, properties, methods
}
```

- Lists of `implements` MAY be split across multiple lines, where each subsequent line is indented once. When doing so, the first item in the list MUST be on the next line, and there MUST be only one interface per line.

```php
<?php
namespace Vendor\Package;

use FooClass;
use BarClass as Bar;
use OtherVendor\OtherPackage\BazClass;

class ClassName extends ParentClass implements
    \ArrayAccess,
    \Countable,
    \Serializable
{
    // constants, properties, methods
}
```

Autoload Standard PSR-4

1. The term "class" refers to classes, interfaces, traits, and other similar structures.
2. A fully qualified class name has the following form:
3. `\<NamespaceName>(\<SubNamespaceNames>)*\<ClassName>`
    1. The fully qualified class name MUST have a top-level namespace name, also known as a "vendor namespace".
    2. The fully qualified class name MAY have one or more sub-namespace names.
    3. The fully qualified class name MUST have a terminating class name.
    4. Underscores have no special meaning in any portion of the fully qualified class name.
    5. Alphabetic characters in the fully qualified class name MAY be any combination of lower case and upper case.
    6. All class names MUST be referenced in a case-sensitive fashion.
4. When loading a file that corresponds to a fully qualified class name ...
    1. A contiguous series of one or more leading namespace and sub-namespace names, not including the leading namespace separator, in the fully qualified class name (a "namespace prefix") corresponds to at least one "base directory".
    2. The contiguous sub-namespace names after the "namespace prefix" correspond to a subdirectory within a "base directory", in which the namespace separators represent directory separators. The subdirectory name MUST match the case of the sub-namespace names.
    3. The terminating class name corresponds to a file name ending in `.php`. The file name MUST match the case of the terminating class name.
5. Autoloader implementations MUST NOT throw exceptions, MUST NOT raise errors of any level, and SHOULD NOT return a value.