Name : Mayur Purushvani

PSR :

PSR – PHP Standard Recommendations .

It is an Extended coding style.

General Rules :

Coding Standards :

The First letter of each word is capitalized including the very first letter.

Files :

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 :

Line length must be 120 characters.

Line should not be longer than 80 characters.

No white space at the end of any statement.

Blank line may be added to improve readability of code.

Not be more than one statement per line.

Indenting :

Must use proper indentation of 4 spaces for each indent level. Not use tab.

Keywords and Types :

Keywords and types must be in lower case.

Short form of type keywords must be – bool instead of Boolean and int instead of integer.

Declare Statements, Namespace, and Import Statements :

Opening tag <?php

One or more class-based ‘use’ import statement.

One or more function-based ‘use’ import statement.

One or more constant-based ‘use’ import statement.

Example :

```
<?php

declare(strict_types = 1);

namespace Vendor/Packages;


use Vendor/Packages/{ClassA as A, ClassB as B};

use Vendor/Packages/AnotherNamespace/AnotherClassA as A;


use function Vendor/Packages/{function A, functionB};

use Another/Vendor/functionD;


use const Vendor/Packages/{CONSTANT_A, CONSTANT_B};

use const Another/Vendor/CONSTANT_C;

?>
```

Declare(strict_types = 1);

This means you have to place declare(strict_types=1); in every file where you want strict types to be imposed.

Classes, Properties and Methods :

The tern classes refers to all classes, interfaces and traits.

Extends and implements :

```
class className extends ParentClass implements \ArrayClass

{

}
```

Using Traits :

What is Traits? :

Traits are a mechanism for code reuse in single inheritance languages such as PHP. A Trait is intended to reduce some limitations of single inheritance by enabling a developer to reuse sets of methods freely in several independent classes living in different class hierarchies.

The ‘use’ keyword is used in the classes to implement traits.

Example :

```
use Vendor/Package/FirstTrait;
```

Properties and Constants :

Visibility must be declared in all projects

The ‘var’ keyword must not be used to declare any property.

There must be not more than one property declare in the single line.

Must be a space declaration between type declarations and property names.

Example :

```
namespace Vendor/Packages;

class Abc

{

        public $foo = null;

        public static int $a = 0;

}
```

Methods and Functions:

Visibility must be declared in all the methods.

Not be declared with space in method or property names.

No space between open and closing statements.

Example :

```
<?php

function foo ($a, $b, $c = [])

{

}
```

Methods ad Function Arguments :

Space before each comma ad as well after each comma.

To declare nullable type declaration You can use '?'.

Must not be a space after the '?'.

Example :

```
<?php

class abc

{

        public  function name(?string $ar) : ?string

        {

                return  'foo';

        }

?>
```

When combining the both the reference operators, use three dots operators.

No space between the two of them

Example :

```
public function process(string $ar, &...$part)
{
}
```

Abstract, Final and Static :

Abstract and final declarations must with visibility declaration.

When static declaration must after the visibility declaration.

Example :

```
<?php
abstract class abc
{
        protected static $foo;
        final public static function bar()
        {
        }
}
```

Methods and Function Calls :

No space between at all.

Example :

```
<?php
bar();
$foo->bar($arg1);
Foo:bar($ar1, $ar2);
?>
```

Control Structures :

Must not be space between at all.

Structure of body must be indented.

All the lines after the next line in body.

If, elseif and else statements :

Example:

```
<?php
If($ex1){
}
elseif($ex2){
}
else{
}
?>
```

Switch case :

The case statement must be indented.

No space between at all.

There must be a comment such as //No break when all all-through is when there is no break statement in switch cases.

Example:

```
switch($abc){
        case 0:
                echo ‘first’;
                break;
        case 1:
                echo ‘second’;
                //no break
}
```

<u>While, do while loop :</u>

while Example :

```
<?php
while($abc){
}
?>
```

do while Example :

```
<?php
do
{
} while($abc);
?>
```

For loop :

Must be split across multiple lines, where each sub lines is indented at least once.

Example :

```
<?php

for($i = 0; $i < 10; $i++) {

}

?>
```

foreach :

example :

```
<?php

foreach($abc as $a => $value){

}

?>
```

try, catch, finally :

Example :

```
<?php

try {

} catch(Exception e) {

} finally {

}

?>
```

<u>**Operators :**</u>

<u>Unary Operators :</u>

To increment or decrement operators must not have any space between the operator and operand.

Example :

```
i++;

j++;
```

For typecasting, must not have any space within the parenthesis.

```
$intValue = (int) $abc;
```

<u>Binary Operators :</u>

All binary like arithmetic, comparison, assignment, bitwise, logical, string and type operators must be followed at least one space.

Example :

```
<?php

If ($a == $b) {

        $foo = $bar;

} else {

        $foo = $car;

}

?>
```

<u>Ternary Operators :</u>

At least one space around the ? or ; characters.

Example :

```
$variable = $foo ? ‘foo’ : ‘bar’ ;
```

Or

```
$variable = $foo ?: ‘foo’ ;
```

Clousers :

It is declared with a space after the ‘function’ keyword.

Space before and after the ‘use’ keyword.

Opening and closing statement must be in the new line.

Example :

```
<?php

$clouser = function ($ar1, $ar2) {

        //with arguments

};

$clouser = function ($ar1, $ar2) use ($var1, $var2) {

        //With argument and var

}

$clouser = function ($ar1, $ar2) use ($var1. $var2) : bool {

        //with return statement

}

?>
```

Anonymous Class :

Example :

```
<?php

$instance = new class {};

?>
```