Static Keyword

Tip This page describes the use of the *static* keyword to define static methods and properties. *static* can also be used to define static variables and for late static bindings. Please refer to those pages for information on those meanings of *static*.

Declaring class properties or methods as static makes them accessible without needing an instantiation of the class. A property declared as static cannot be accessed with an instantiated class object (though a static method can).

For compatibility with PHP 4, if no visibility declaration is used, then the property or method will be treated as if it was declared as *public*.

Static methods

Because static methods are callable without an instance of the object created, the pseudo-variable `$this` is not available inside the method declared as static.

Caution In PHP 5, calling non-static methods statically generates an **`E_STRICT`** level warning.

Warning In PHP 7, calling non-static methods statically is deprecated, and will generate an **`E_DEPRECATED`** warning. Support for calling non-static methods statically may be removed in the future.

Example #1 Static method example

```
<?php
class Foo {
    public static function aStaticMethod() {
        // ...
    }
}

Foo::aStaticMethod();
$classname = 'Foo';
$classname::aStaticMethod(); // As of PHP 5.3.0
?>
```

Static properties

Static properties cannot be accessed through the object using the arrow operator ->.

Like any other PHP static variable, static properties may only be initialized using a literal or constant before PHP 5.6; expressions are not allowed. In PHP 5.6 and later, the same rules apply as *const*expressions: some limited expressions are possible, provided they can be evaluated at compile time.

As of PHP 5.3.0, it's possible to reference the class using a variable. The variable's value cannot be a keyword (e.g. *self*, *parent* and *static*).

Example #2 Static property example

```php
<?php
class Foo
{
    public static $my_static = 'foo';

    public function staticValue() {
        return self::$my_static;
    }
}

class Bar extends Foo
{
    public function fooStatic() {
        return parent::$my_static;
    }
}



print Foo::$my_static . "\n";

$foo = new Foo();
print $foo->staticValue() . "\n";
print $foo->my_static . "\n";      // Undefined "Property" my_static

print $foo::$my_static . "\n";
$classname = 'Foo';
print $classname::$my_static . "\n"; // As of PHP 5.3.0
print Bar::$my_static . "\n";
$bar = new Bar();
print $bar->fooStatic() . "\n";
?>
```