**Static Methods and Properties**

If you are familiar with the basics of OOP in PHP, you should recognize that you often access class data and functionality through objects. For instance, you should be familiar with the syntax:

$Foo = new Foo();

$Foo->Name = "Alex Roxon";

The above snippet alters the public member variable ***Name*** in class Foo. However, with the introduction of the ***static*** keyword in PHP 5, we can now access methods and properties through the context of a class rather than only the object ***(note: these methods/properties need to be declared static first)***. For example:

class Foo {

static public $Name = "Alex Roxon";

static public function helloWorld() {

print "Hello world from " . self::$Name;

}

}

To declare a method or property as static, we must use the ***static*** keyword. When accessing properties from outside the class scope, we use the class name followed by two :: and then the property name. For instance:

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

Foo::helloWorld();

The above would output:

Alex Roxon

Hello world from Alex Roxon

You should be able to see that from within the context of the class, to access member variables or properties you use the self keyword.

As static methods are within the class scope, they cannot access any ***normal*** methods or properties (i.e. those that have not been declared static), as those would belong to the object, and not the class. A by-product of this is you cannot use the $this variable from within a static method, as static methods are not invoked in the context of an object.

**Why use static methods and properties?**

After reading the above, this is a question you are probably asking yourself. You certainly should not get in to the habit of unnecessarily using the static keyword when it is not required, however, static elements do have a number of characteristics that can be useful.

1. When you are working on a large OOP based project, you’ll no doubt be working with many classes (both parent and child classes). An unfortunate consequence of this is that in order to access elements from different classes, they must manually be passed through each class (or worse, storing an instance in a global variable). This can be painstakingly frustrating and can lead to messy code and overall bad project design. Thankfully, static elements are accessible from any context (i.e. anywhere in your script), so you can access these methods without needing to pass an instance of the class from object to object.
2. As you don’t need to declare an object instance to access static elements, you can be saved from unnecessary declarations to access seemingly simple functions.
3. Static elements are available in every instance of a class, so you can set values that you want to be available to all members of a type.

**Conclusion**  
Static methods and properties, introduced in PHP5 can be remarkably useful (see advantages above). If you can wrap your head around the concept, you’ll find yourself implementing code that will in the end save you time and make you a *better* programmer.