Constructor

Constructor Functions are a special type of functions which are called automatically whenever an object is created.

```
class Example {
	public function __construct() {
		// any code in here is automatically run when an
		object is created from this class
	}
}
```

This creates a new object from the class Example

```
$var = new Example
```

When the object is being created, the __construct() function automatically runs.

If a _construct() function has parameters, then these must be passed along when the object is created. E.g.

```
class Person {
	private $name;
	public function __construct($name) {
		$this->name = $name;
	}
}

$bob = new Person("Bob Hanson");
$frank = new Person("Frank Johnson");
$michael = new Person("Michael Bayson");
```

See more: http://www.tutorialspoint.com/php/php_object_oriented.htm

$this

In php the key $this is used inside classes to refer to the class itself. In the example below the $this->name refers to the public $name property defined in the beginning of the class.

```
class Person {
	private $name;
	public function __construct($name) {
		$this->name = name;
	}
}
```

Notice that using $this means that you will not have to include $ for the properties you call, just the name.

This is correct:
`$this->name;`

This is NOT correct:
`$this->$name;`

Public vs Private

When you define a class, you can decide to make properties and methods either public or private.

Public
A public property or method will be accessible from outside the class scope. That means that code written outside the class { } will be able to call the property or method.
If it is a property, not only can outside the class code call it, it can also assign a new value.

E.g.

```
class User {
	public $name;
	public function __construct($name) {
		$this->name = $name;
	}
}

$michael = new User;
echo $michael->name;
$michael->name = "Frank"; // oh oh, now his name is Frank
```

Private
By setting a property or method to private, it can no longer be accessed by code outside the class. This makes it possible to controll how your objects are accessed and used. E.g. the $name property above can no longer be changed from outside the class.

```
class User {
	private $name;
	public function __construct($name) {
		$this->name = $name;
	}
	public function get_name() {
		return $this->name;
	}
}

$michael = new User;
echo $michael->get_name();
$michael->name = "Frank"; // Error. Php will NOT allow you to
                          access this private property.
```

Instances

You create an object (known as an instance of the class) by using the keyword **new**. You can make as many instances of an object as you want.

```
$skoda = new Car;
$bmw = new Car;
$hummer = new Car;
```

You access the properties and call the methods of the object by using the **->** notation.

```
$skoda = new Car;
$skoda->color = "blue";

$bmw = new Car;
$bmw->color = "red";
$bmw->drive();
echo $bmw->color;
```

Class

A class is a blue print for an object. This is where you define all the properties (something that describes the object) and methods (what the object can do) that the object will have when you create an instance of it.

A class is defined using the **class** keyword. E.g.

```
class Car {
	public $color;
	public function drive() {
		// code to make it drive goes here
	}
}
```

In the example above, the class define what a car object should look like and what it can do. Here it would have one property (color) and one method (drive).

The scope of the class is defined by the {} following the class keyword. Any code written between these brackets are considered to be in the class scope and anything before/after to be outside the class scope.