Object Oriented Programming in PHP

Aditya lohia
Mayank Sharma

What is a *class*?

- Conceptually, a class represents an **object**, with associated methods and variables

Class Definition

```
<?php
class dog {
    public $name;
    public function bark() {
        echo 'Woof!';
    }
}
?>
```

An example class definition for a dog. The dog object has a single attribute, the name, and can perform the action of barking.

Class Definition

```
class dog {
    public $name;
    public function bark() {
        echo 'Woof!';
    }
}
?>
```

Define the **name** of the class.

Class Definition

```
<?php
cl
    public $name;
    var $name
    public function bark() {
        echo 'Woof!';
}
}
?>
```

Define an object attribute (variable), the dog's name.

Class Definition

Define an object action (function), the dog's bark.

```
<?php
class dog {
    public function bark() {
        echo 'Woof!';
    }
}
?>
```

Class Definition

```
<?php
class dog {
    public $name;
    public function bark() {
        echo 'Woof!';
    }
}
?>
```

End the class definition

Class Defintion

- The definition **does not do anything by itself**.
- It is a blueprint, or description, of an object.
- To do something, you need to **use** the class.

Class Usage

```
<?php
require('dog.class.php');
$puppy = new dog();
$puppy->name = 'Rover';
echo "{$puppy->name} says ";
$puppy->bark();
?>
```

Class Usage

```php
<?php
require('dog.class.php');
$puppy = new dog();
$puppy->name = 'Rover';
echo "{$puppy->name} says ";
$puppy->bark();
?>
```

Include the class definition

Class Usage

```php
<?php
require('dog.class.php');
$puppy = new dog();
$puppy->name = 'Rover';
echo "{$puppy->name} says ";
$puppy->bark();
?>
```

Create a new **instance** of the class.

Class Usage

```
<?php
require('dog.class.php');
$puppy->name = 'Rover';
$puppy->name = 'Rover';
echo "{$puppy->name} says ";
$puppy->bark();
?>
```

Set the name variable **of this instance** to 'Rover'.

Class Usage

```
<?php
require('dog.class.php');
$puppy = new dog();
$puppy->name = 'Rover';
echo "{$puppy->name} says ";
$puppy->bark();
?>
```

Use the name variable of this instance in an echo statement..

Class Usage

```
<?php
require('dog.class.php');
$puppy = new dog();
$puppy->name = 'Rover';
echo "{$puppy->name} says ";
$puppy->bark();
?>
```

Use the dog object bark method.

One dollar and one only…

```
$puppy->name = 'Rover';
```

The most common mistake is to use more than one dollar sign when accessing variables.

The following means something entirely different.

~~`$puppy->$name = 'Rover';`~~

$this Keyword

If you need to use the class variables within any class actions, use the special variable **$this** in the definition:

```
class dog {
        public $name;
        public function bark()
        {
        echo $this->name.' says Woof!';
        }
}
```

Constructor methods

- A **constructor** method is a function that is automatically executed when the class is first instantiated.

- Create a constructor by including a function within the class definition with the __**construct name**.

- Remember.. if the constructor requires arguments, they must be passed when it is instantiated!

Constructor Example

```
<?php
class dog {
    public $name;
    public function __construct($nametext)
    {
        $this->name = $nametext;
    }
    public function bark() {
        echo 'Woof!';
}
}
?>
```

Constructor function

Class Scope

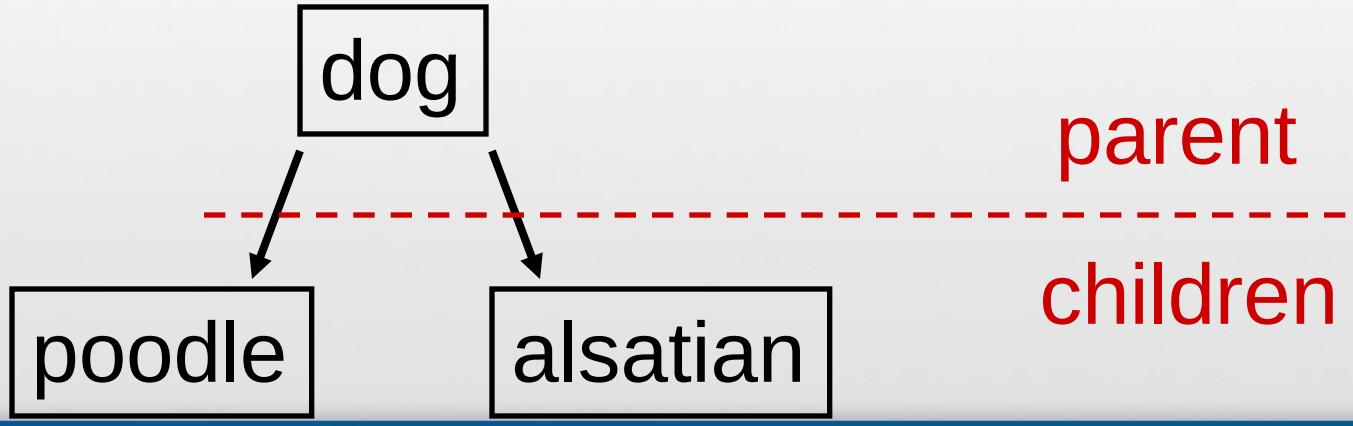
Each instantiated object has its own local scope.

e.g. if 2 different dog objects are instantiated, **`$puppy1`** and **`$puppy2`**, the two dog names **`$puppy1->name`** and **`$puppy2->name`** are entirely independent..

Inheritance

- The real power of using classes is the property of inheritance.
- The child classes ‘inherit’ all the methods and variables of the parent class, and can add extra ones of their own.

e.g. the child classes poodle inherits the variable ‘name’ and method ‘bark’ from the dog class, and can add extra ones.



Inheritance example

There are three sizes of poodle - Standard, Miniature, and Toy…

```
class poodle extends dog {
  public $type;

  public function set_type($height) {

     if ($height<10) {
          $this->type = 'Toy';

     } elseif ($height>15) {
          $this->type = 'Standard';

     } else {
          $this->type = 'Miniature';
          }
     }
}
```

Inheritance example

There are three sizes of poodle - Standard, Miniature, and Toy…

```
class poodle extends dog {
  public $type;

  public function set_type($height) {

      if ($height<10) {
            $this->type = 'Toy';

      } elseif ($height>15) {
            $this->type = 'Standard';

      } else {
            $this->type = 'Miniature';
            }
      }
}
```

Note the use of **extends** keyw
indicate that the
poodle class is
of the dog class

Method Overriding

It is possible to over-ride a parent method with a new method if it is given the same name in the child class..

```
class poodle extends dog {
    …
    public function bark() {
                echo 'Yip!';
            }
    …
}
```

Access Specifiers

Access Specifiers defines the visibility for variables and methods of Class.

There are three Access Specifiers in PHP

1) Public
2) Private
3) Protected

Interface

Used to implement Multiple inheritance.

Method is defined in Interface and implemented in Derived Class

Variable Should be constant.

Thank You