Constructors & Destructors

This lesson introduces the concept of constructors, destructors, how to declare and call them.

We'll cover the following

- Introduction
- Constructor Declaration
- Constructor Definition
  - Calling a constructor
- Introduction to Destructors

Introduction

A *constructor* is used to initialize *member* variables when an *object* is declared. It is automatically called at the time when the *object* of the *class* is declared.

> **Note:** A *constructor* is a *member* function that is usually `public`.

Also keep in mind that unlike other *methods* defined inside a class, a constructor **cannot** *return* a value.

Constructor Declaration

Classes can define a special `__construct()` *method*, which is executed as part of object creation.

Let’s take a look at how it is defined:

```
<?php

class Shape {

public $sides = 0;
```

```
public $name = " ";

 //initializes $name to $Name and $sides to $Sides

public function __construct($Name, $Sides)

?>
```

Constructor Definition

The constructor for `Shape` can be defined as follows:

```
<?php
class Shape
{

    public $sides = 0;

    public $name = " ";

    public function __construct($name, $sides)
    { //defining a constructor
        $this->sides = $sides; //initializing $this->sides to $sides
        $this->name = $name; //initializing $this->name to $name

    }

    public function description()
    { //method to display name and sides of a shape
        echo "A $this->name with $this->sides sides.";
    }

}

$myShape = new Shape("hexagon", 6); //making an object and passing values to the constructor
$myShape->description(); // A shape with 6 sides

?>
```

Calling a constructor

As you can see above in **line 21**, the way to call a *constructor* is not like a normal *member* function.

- It is called in *object* declaration.
- It creates a `Shape` object.
- Then *calls* the *constructor* to initialize *variables*.

In the example, the **constructor** takes as parameters the *name* and *number of sides* and sets their values.

Introduction to Destructors

Destructors are the opposite of **constructors**, as they define the *final* behavior of an *object* and execute when the object is no longer in use.

An object's **destructor**, which takes **no** *parameters*, is called sometime after an *object* is no longer *referenced*, but the complexities of *garbage* collection make the specific timing of *destructors* uncertain.

> **Note: Destructors** are not called but are invoked *automatically*.

```
<?php
class Shape
{

    public $sides = 0;

    public $name = " ";

    public function __construct($name, $sides)
    { //defining a constructor
        $this->sides = $sides; //initializing $this->sides to $sides
        $this->name = $name; //initializing $this->name to $name

    }

    public function __destruct()
    { //destructor for Shape gets called at the end
        echo "Destructor Called!\n";
    }

    public function description()
    { //method to display name and sides of a shape
        echo "A $this->name with $this->sides sides.\n";
    }

}

$myShape = new Shape("hexagon", 6); //making an object and passing values to the constructor
$myShape->description(); // A shape with 6 sides
```

This marks the end of our discussion on **constructors** and **destructors**. In the next lesson, we will discuss different access modifiers used while declaring member variables in classes.

variables in classes.