

\$this and self

This lesson discusses the use of this and self in accessing members in classes.

We'll cover the following

- \$this
 - The Object Operator
- self
 - The Scope Resolution Operator
- this versus self
- Example

\$this

The pseudo-variable `$this` refers to a member of a class according to an instance of that class. It provides a reference to the calling object, that is the object to which the method or property belongs to. Note that the member of the class must be non-static.

The Object Operator

The `->` symbol is a built-in construct in PHP that is used with the `$this` keyword to access contained **methods** and **properties**.

self

In PHP, like many other languages, the `self` keyword refers to properties and methods inside the scope of a class. This pseudo-variable provides a reference to the calling object, that is the object to which the method or property belongs to.

The Scope Resolution Operator

The `::` symbol, known as the *scope resolution operator* is an in-built construct in PHP that is used to access contained **methods** and **properties**. It is used with the `self` keyword.

this versus self

- Use `$this->member` for accessing *non-static* members (methods and properties).
- Use `self::$member` for accessing *static* members (methods and properties).

Example

Run the code below to see how you can use `$this` and `self` in PHP.

```
<?php
class Circle
{
    // properties
    public $radius= 0; //declaring public member
    public static $pi=3.14; //declaring a public static member

    // Method to get the Circumference
    public function getCircumference(){
        return (2 * self::$pi * $this->radius );
    }

    // Method to get the area
    public function getArea(){
        return ($this->radius * $this->radius*self::$pi);
    }

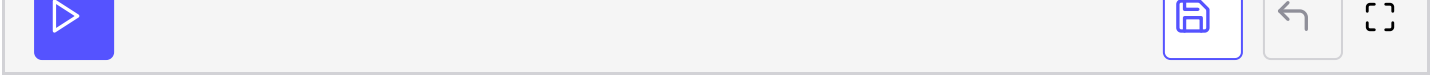
    // Method to get the diameter
    public function getDiameter(){
        return ($this->radius * 2);
    }
}

// Create a new Circle class object
$obj = new Circle;

// Set object properties values
$obj->radius = 4;

// Read the object properties values again to show the change
echo "Radius is ". $obj->radius . "\n";
echo "Diameter is ". $obj->getDiameter() . "\n";

// Call the object methods
echo "Circumference is ". $obj->getCircumference(), "\n";
echo "Area is " . $obj->getArea(). "\n";
echo "Value of pi is " . $obj::$pi;
?>
```



As you can see in the code above, when accessing *non static* members like `radius` we are using `$this` and `->` to access the value, whereas, when accessing the *static* member like `pi` we used `self` and `::` to access the value.

In the next lesson we'll discuss *constructors* and *destructors* as well as their use in classes.