Closures

Introduction

- Closure is a function or a reference for a function together with a table that includes references for each one of the non local variables that exist within the outer scope of the closure.
- Unlike a plain function pointer the closure can use those non local variables that belong to its outer scope even when invoked out of it.

Sample

```php
<?php
class Something
{
    private $number;
    private $person;
    public function __construct($num)
    {
        $this->number = $num;
    }
    public function getFunction()
    {
        return function()
        {
            for($i=1;$i<=$this->number;$i++)
            {
                echo "<br>gaga";
            }
        };
    }
}
```

Sample

```
$obA = new Something(7);
$obB = new Something(3);
$temp = $obB->getFunction();
$temp();

?>
```

The `bindTo` Function

- Using this function we can bind a closure function, that is currently bonded with a specific object, with another object instead.

The `bindTo` Function

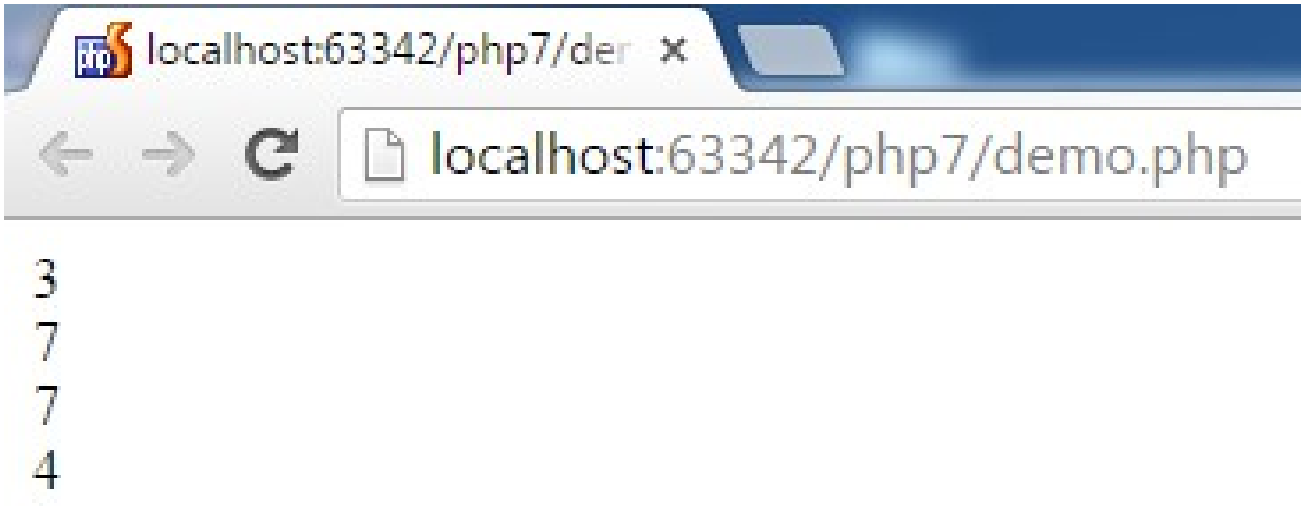
```
<?php
class A {
    private $magic;
    function __construct($number) {
        $this->magic = $number;
    }
    function getClosure() {
        return function() { return $this->magic; };
    }
    function setMagic($number) {
        if($number>0) {
            $this->magic = $number;
        }
    }
}

$ob1 = new A(3);
$ob2 = new A(4);
$ob3 = new A(5);
```

The `bindTo` Function

```
$f1 = $ob1->getClosure();
echo $f1()."<br/>";
$ob1->setMagic(7);
echo $f1()."<br/>";
$f2 = $f1->bindTo($ob2);
echo $f1()."<br/>";
echo $f2()."<br/>";
//echo $f1->call($ob3)."<br/>";
?>
```

The `bindTo` Function



The Closure `call` Function

- Using this function we can invoke a closure function on object which isn't the one the closure function is bounded with.

The Closure `call` Function

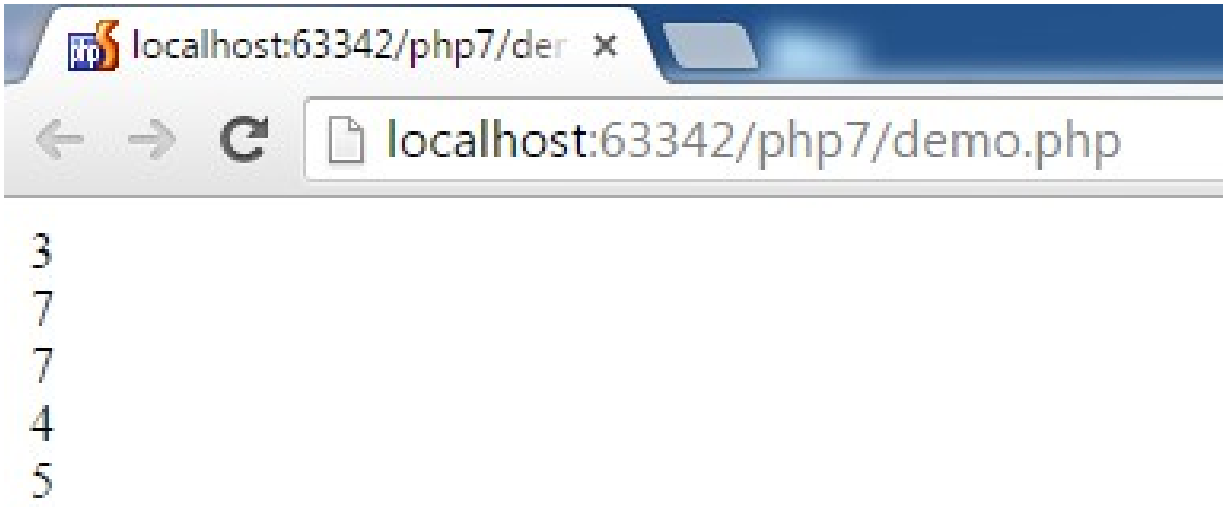
```php
<?php
class A {
    private $magic;
    function __construct($number) {
        $this->magic = $number;
    }
    function getClosure() {
        return function() { return $this->magic; };
    }
    function setMagic($number) {
        if($number>0) {
            $this->magic = $number;
        }
    }
}
```

The Closure `call` Function

```
$ob1 = new A(3);
$ob2 = new A(4);
$ob3 = new A(5);

$f1 = $ob1->getClosure();
echo $f1()."<br/>";
$ob1->setMagic(7);
echo $f1()."<br/>";
$f2 = $f1->bindTo($ob2);
echo $f1()."<br/>";
echo $f2()."<br/>";
echo $f1->call($ob3)."<br/>";
?>
```

The Closure `call` Function



The `use` Keyword

- When defining a closure inner function we can allow that closure inner function to access local variables that belong to its outer scope by adding the `use` keyword in order to specify which of the local variables that belong to the outer scope we want to allow our closure inner function to access.

The `use` Keyword

```
<?php
function f1($number1)
{
    $number2 = 0;
    $f2 = function ($num1, $num2) use ($number1, &$number2)
    {
        $number2 = $number2 + ($num1 + $num2);
        echo "number2=".$number2;
        return $number1 * ($num1+$num2);
    };
    return $f2;
}

$f = f1(7);
$num = $f(2,3);
echo "<h1>".$num."</h1>";
$num = $f(2,3);
echo "<h1>".$num."</h1>";
?>
```

The `use` Keyword