

# **Resumen de Javascript**

# **Semana 6**

# Factory Constructor Pattern

This pattern is special, because it doesn't use "new".

The object is created by a simple function call, similar to *Python-style*:

```
var animal = Animal("fox")
```

```
var rabbit = Rabbit("rab")
```

## Declaration

The constructor is defined as a function which returns a new object:

```
1 function Animal(name) {  
2  
3   return {  
4     run: function() {  
5       alert(name + " is running!")  
6     }  
7   }  
8  
9 }
```

usage:

```
1 var animal = Animal("fox")
2 animal.run()
```

## Inheritance

Rabbit is made by creating an `Animal`, and then mutating it:

```
01 function Rabbit(name) {
02
03   var rabbit = Animal(name) // make animal
04
05   rabbit.bounce = function() { // mutate
06     this.run()
07     alert(name + " bounces to the skies! :)")
08   }
09
10   return rabbit // return the result
11 }
12
13 var rabbit = Rabbit("rab")
14 rabbit.bounce()
```

## Private/protected methods (encapsulation)

Local variables and functions become private:

```
01 function Bird(name) {  
02  
03     var speed = 100 // private prop  
04     function openWings() { /* ... */ } // private method  
05  
06     return {  
07         fly: function() {  
08             openWings()  
09             this.move()  
10         },  
11         move: function() { /*...*/ }  
12     }  
13 }
```

The code above looks simple, but still there is a gotcha.

A public method can be called as `this.move()` from another public method, but *not* from a private method.

A private method like `openWings` can't reference `this`. There's no reference to the new object in a local function.

One way to solve that is to bind the new object to a local variable prior to returning:

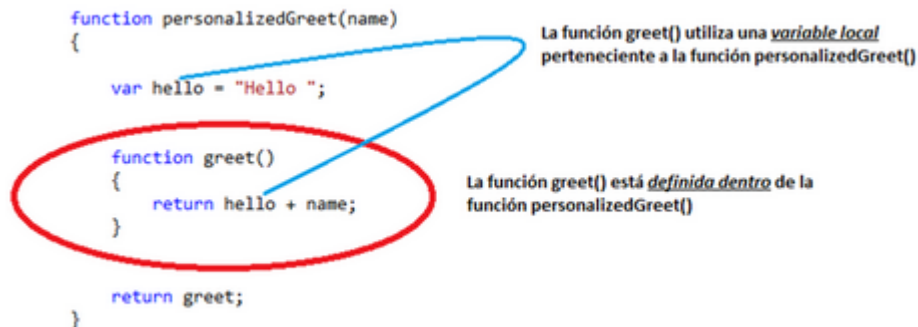
```
01 function Bird(name) {  
02  
03   function doFly() {  
04     openWings()  
05     self.move()  
06   } // private method  
07  
08  
09   var self = {  
10     fly: function() { doFly() },  
11     move: function() { /*...*/ }  
12   }  
13   return self  
14 }
```

## Summary

- The *factory constructor* uses a function which creates an object on it's own without new.
- Inheritance is done by creating a parent object first, and then modifying it.
- Local methods and functions are private. The object must be stored in closure prior to returning if we want to access it's public methods from local ones.

Whenever you see the function keyword within another function, the inner function has access to variables in the outer function.

```
function foo(x) {  
  var tmp = 3;  
  
  function bar(y) {  
    alert(x + y + (++tmp)); // will alert 16  
  }  
  
  bar(10);  
}  
  
foo(2);
```



```
function personalizedGreet(name)  
{  
  var hello = "Hello ";  
  
  function greet()  
  {  
    return hello + name;  
  }  
  
  return greet;  
}
```

La función greet() utiliza una variable local perteneciente a la función personalizedGreet()

La función greet() está definida dentro de la función personalizedGreet()