Chapter 2: closures

Closure definition

A closure is a function holding a reference to local variables that stay allocated even after the function returns. Local

Sample 1:

```
// some scope, but even window
var name = 'Federico';

hello();

// here is the closure
function hello() {
    alert('Hello '  + name + '!');
}
```

Sample 2:

```
var multiplier = function (n){
       return function (p) {
           return p * n;
       }
   },
   m3 = multiplier(3),
   m5 = multiplier(5);

console.log(m3(7), m5(7));
```

Sample 3:

```
var n;
function f() {
   var s = 'secret';
   n = s;
}
```

```
f();
console.log(n)
```

Scope chain

Said that javascript has a function scope will be clear that in the following code:

```
//
//... global
//
var a = 1;
function f () {
    var b = 2;
    function g () {
        var c = 3;
        // ...
    };
};
```

Seen that in javascript the scope is at function level from now on let me refer to the scope of a function *f* with the *S(f)* symbol, and use S for the global scope. Using that notation we note that:

- S holds *a* and *f()*, **not** *b*, *g()* or *c*
- S(f) holds [S, *b*, *g()*], **not** *c*
- S(g) holds [S(f), *c*]

where each function

Told that we can see closure in that fashion: