Ámbito de las variables

El objeto que engloba todas las variables globales es *window*, que es a su vez global también.

Las dos sentencias siguientes son equivalentes:

```
var unaVariable = 'hola';
window.unaVariable = 'hola';
```

Es lógico que si una variable se declara fuera de una función se haga global, hasta aquí todo normal, pero miremos el siguiente caso:

```
function Prueba(){
 this.propiedad='una propiedad';
};

Prueba();
alert(window.propiedad); // "una propiedad"
```

Curiosamente, *propiedad* pasa a ser global aún estando dentro de una función. Ésto pasa porque el operador ***this*** **hace 'públicas' las variables y los métodos** (muy importante recordar esto).

No ocurre lo mismo si declaramos una variable dentro de una función (ámbito local):

```
function UnaFuncion(){
 var variableDentroFuncion = 'una variable interna'; //privada
};

UnaFuncion();
alert(window.variableDentroFuncion); //undefined, ya no es global
```

En el primer ejemplo, al llamar a la función Prueba() lo hemos hecho dentro de un ámbito global y por tanto su propiedad también lo era. Si a esta función constructora / 'clase' la instanciamos dentro de una variable 'p', el objeto global window no tendrá acceso a él, no de una forma directa, con lo que nos ahorramos un poco de memoria:

```
function Prueba(){
 this.propiedad='una propiedad';
};

var p=new Prueba();

alert(window.propiedad); // undefined
alert(window.p.propiedad); // 'una propiedad'
```

Ámbito de los métodos: públicos, privados y privilegiados

* **Métodos Públicos**, los métodos públicos son completamente accesibles dentro del contexto del objeto.

- Desde el constructor. Con el operador this, hacemos públicos los métodos como ya hemos visto.

```
    function Prueba(){
      this.getSaludo=function() {return 'Hola!';}
    };
```

- Desde el objeto prototype. También visto anteriormente, prototype nos permite hacer públicos los métodos:

Utilizando la función prueba anterior...

```
Prueba.prototype.ampliaSaludo = function (cadenaArg) {
 return this.getSaludo().concat(cadenaArg);}

var p=new Prueba();
alert(p.ampliaSaludo(' Como estas?')); //Hola! Como estas?
```

* **Métodos Privados**. Métodos que solamente son accesibles desde el propio objeto, ya sea desde otros métodos privados dentro de él mismo, otras variables privadas que contenga o otros métodos privilegiados (como veremos más tarde). Los métodos privados son muy importantes ya que permiten una correcta gestión del código sin que se produzcan coincidencias con otros objetos.

```
function CapitalizaUnaPalabra(palabraArg){

 var primeraLetra=palabraArg.charAt(0).toUpperCase();  //variable privada
 var segundaLetra=palabraArg.slice(1).toLowerCase();  //variable privada

 // método privado
 function unionLetras() { alert (primeraLetra.concat(segundaLetra));};

 //método privilegiado
 this.capitaliza = function () {return unionLetras();}
};

var c= new CapitalizaUnaPalabra('música');
c.unionLetras(); // error: no accesible
c.capitaliza(); // Música
```

* **Métodos Privilegiados**. Es una combinación de los dos anteriores. Ya que los métodos privados no son accesibles desde el exterior, si queremos obtener sus valores debemos hacer un puente entre lo público y lo privado. Como ya hemos visto en el ejemplo anterior,*this.capitaliza* es un método que se hace público devolviendo una operación privada (*unionLetras*), éste es un método privilegiado.

La mayor diferencia entre los tres es que únicamente nuevos **métodos públicos pueden añadirse una vez creado el objeto. Los privados y privilegiados deben establecerse en la función constructora.**