Patrones de Invocación de Funciones: this

Cada vez que llamamos a una función en javascript se crea un nuevo contexto, en el cual tenemos *this* y *arguments*. El *this* se va a inicializar de distintas formas dependiendo de cómo invoquemos a la función.

Hay 4 patrones para invocar una función:

- como Método
- como Función
- como Constructor
- y con *Apply*

Patrón de invocación como Método

Cuando una función es guardada como una propiedad de un objeto, lo llamamos *método*. En este caso *this* es inicializado con el objeto al que pertenece la función.

```
var obj = {
    valor: 0,
    incrementar: function(incremento){
       this.valor += incremento;
    }
};

obj.incrementar(2);
console.log(obj.valor); // 2
```

Patrón de invocación como Función

Cuando una función no está dentro de un objeto es invocada como función, y *this* es inicializado con el Objeto Global. Esto es un problema, ya que cuando llamamos a una función dentro de otra, *this* sigue referenciando al Objeto Global y si queremos acceder al this de la función padre tenemos que almacenarlo en una variable primero:

```
var obj = {
     valor: 0,
     incrementar: function(incremento){// invocado como método
        var that = this;

        function otraFuncion(unValor){ // invocado como función
               that.valor += unValor;
        }

        otraFuncion(incremento);
     }
};

obj.incrementar(2);
console.log(obj.valor); // 2
```

Patrón de invocación como Constructor

Javascript es un lenguaje de herencia prototipada, lo que significa que un objeto puede heredar directamente propiedades de otro objeto y como su prototipado no es muy convincente, javascript ofrece una sintaxis estilo *creación de objetos* como en los lenguajes clásicos de POO (Programación Orientada a Objetos). Dicho esto, cuando invocamos una función con *new* se creará un objeto con una referencia al valor de su miembro de función prototipada (también llamado *constructor*) y *this* tendrá una referencia a este nuevo objeto.

```
    var Persona = function(){ // nuestro "constructor"
        this.nombre = 'José';
    }
    Persona.prototype.mostrarNombre = function(){
        console.log(this.nombre); //con this accedemos al
constructor
    }

    var p = new Persona();
    p.mostrarNombre(); //imprime 'José'
```

Si invocamos la función Persona() sin el new, this se inicializa distinto y nuestra clase se va a comportar de forma muy extraña, es por eso que como convención llamamos a las clases con la primer letra en mayúscula.

Patrón de invocación con Apply

La función *apply* nos deja construir un arreglo de argumentos para usar al invocar una función y también nos deja elegir el valor que tendrá *this*. *apply* recibe 2 parámetros, el primero es el valor para *this* y el segundo es un arreglo de parámetros.

Usando el ejemplo anterior de prototipado, vamos a cambiar el *this* utilizando el *apply*

```
    var Persona = function(){ // nuestro "constructor"
        this.nombre = 'José';
    }
    Persona.prototype.mostrarNombre = function(){
        console.log(this.nombre); //con this accedemos al
constructor
    }

    //con apply cambiamos el this referenciado al objeto
persona por otroObjeto
    var otroObjeto = {
        nombre: 'Pepe'
    };

    var p = new Persona();
    p.mostrarNombre(); //imprime 'José'
    p.mostrarNombre.apply(otroObjeto); // imprime 'Pepe'
```