Modificando el contexto: call, apply y bind

Partimos del siguiente ejemplo:

```
var obj = {
  delta: 2,
  test: function(num1, num2){
     return (num1 + num2) * this.delta;
  }
};

var resultado = obj.test(2,2);
console.log(resultado); // (2 + 2) * 2 = 8
```

En este caso, simplemente tenemos un objeto con una propiedad y una función, al llamarla con 2 números, los sumamos y le aplicamos el delta de su contexto (en el cual está corriendo la función), que para este caso es el objeto en sí mismo.

Pero qué pasa si ahora queremos cambiarle ese contexto, un nuevo delta, definido por nosotros pero sin modificar el objeto…

apply()

```
var cambio = {
  delta: 5
};

var resultado = obj.test.apply(cambio, [2,2]);
console.log(resultado); // (2 + 2) * 5 = 20
```

Le aplicamos un nuevo contexto en el cual corre la función, por lo que ahora el *this* de *obj* va a contenter nuestro objeto *cambio*, por ende, el delta valdrá 5 al ejecutarse.

¿Y qué es el array que sigue a la llamada?, esos son los argumentos que recibe la función:

```
[funcion].apply([contexto], [parámetros como array]);
```

call()

Otra forma de llamar a una función aplicándole un contexto es con *call*, siguiendo con el ejemplo del apply:

```
var cambio = {
  delta: 5
};

var resultado = obj.test.call(cambio, 2, 2);
console.log(resultado); // (2 + 2) * 5 = 20
```

La única diferencia es la forma de pasar los argumentos, ya que en vez de que sea un array, simplemente toma el primero como el contexto y los siguientes son los argumentos en el orden que los esperamos.

```
[funcion].call([contexto], [param1], [param2], [paramN]);
```

bind()

En este caso hay una vuelta de tuerca más, que nos puede ser bastante útil. El *bind()* no llama a la función con un nuevo contexto, sino que nos devuelve una referencia a la función con ese nuevo contexto:

```
var cambio = {
  delta: 5
};

var funcionConCambio = obj.test.bind(cambio);
var resultado = functionConCambio(2, 2);
console.log(resultado); // (2 + 2) * 5 = 20
```

Ésto aplicado a callbacks puede ser muy interesante. ¿Usas el *self*, *me* o *that*? Supongamos tenemos una llamada ajax y definimos un callback para cuando termina, un ejemplo común sería:

```
// más código donde utilizamos el this
var that = this;
function callback(datos){
```

```
    that.magia(datos);
  }
  ajax(callback);
```

Ahora con bind nos quedaría así:

```
  function callback(datos){
    this.magia(datos);
  }
  ajax(callback.bind(this));
```

¡Mucho más limpio!