Cada día usamos más funciones como JavaScript map . Con el paso del tiempo nos vamos acostumbrando a programar de una forma más funcional. Ahora bien ¿Cómo se implementan realmente estas funciones en JavaScript? . Vamos a ver un ejemplo .Para ello partiremos de un array que ejecuta una función map.

```
var lista = [1, 2, 3, 4];

var resultado = lista.map(function(elemento) {

return elemento * 2

})
console.log(resultado);
```

En este caso tenemos un array de cuatro números y ejecutamos la función map sobre él para imprimir por la consola el mismo array multiplicado por dos.



Acabamos de usar programación funcional para realizar una de las transformaciones más sencillas

JavaScript Map implementación

A muchas personas les cuesta entender en un primer momento como esta implementado en JavaScript este método. Vamos a mostrar una posible implementación:

```
function miMapa(fn) {

var resultado = [];
for (indice = 0; indice < this.length; indice++) {

resultado[indice] = fn(this[indice]);
}
return resultado;

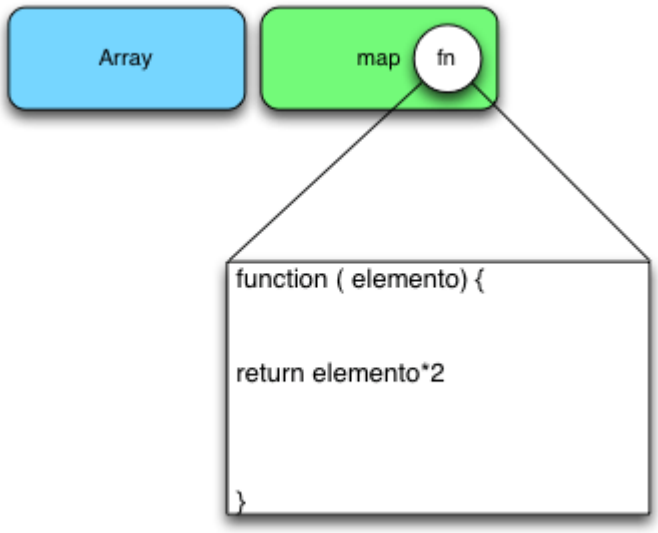
}

Array.prototype.miMapa = miMapa;
var resultado = lista.miMapa(function(elemento) {

return elemento * 2;
})
console.log(resultado);

 
```

En este caso hemos implementado una función que se denomina miMapa y que tiene varias peculiaridades. En primer lugar recibe otra función como parámetro que es la encargada de implementar el algoritmo del mapa.



En segundo lugar hace uso de la variable de contexto this para hacer referencia al array con el que estamos trabajando. Con esta variable y esta función se ejecuta un buble for recorriendo el array y generando un nuevo array aplicando la función que pasemos por parámetro.

JavaScript y Prototipos

Por último nos queda modificar el prototipo del Array en JavaScript para que añada esta nueva función al conjunto ya soportado. A partir de ese momento podemos invocar a la función miMapa como si fuera otra de las funciones que pertenecen al array.

```
var resultado = lista.miMapa(function(elemento) {

return elemento * 2;
})
```

```
console.log(resultado);
```

A veces cuesta entender como funciona la programación funcional pero si vemos el código implementado nos daremos cuenta de que es más sencilla de lo que parece. Esto nos posibilita ampliar las funciones de las que disponemos

Otros artículos relacionados: javascript ES6 Fetch API , javascript promises , JavaScript Reactivo