El uso de JavaScript Array Spread Operator es muy típico en JavaScript moderno. El uso de arrays en JavaScript es continuo y estamos bastante acostumbrados a ellos . Sin embargo a veces se nos hace un poco lioso su uso. Vamos a poner un ejemplo sencillo. Tenemos un array de JavaScript y queremos añadirle un nuevo elemento

```
var lista=[1,2,3,4];
lista.push(5);
console.log(lista);
```

El resultado sale en la consola:

```
[ 1, 2, 3, 4, 5 ]
```

No tiene nada de especial es el código más habitual y es bastante correcto. ¿ Se puede realizar la misma operación de otra forma más óptima?

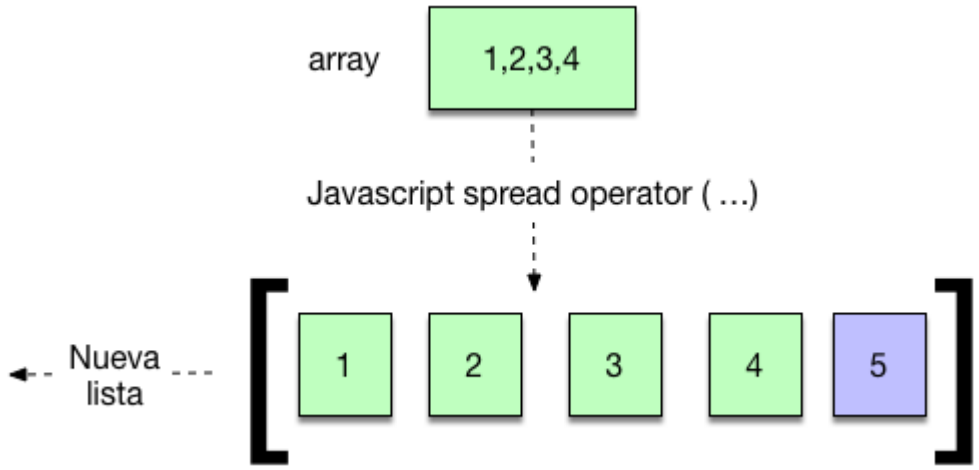
JavaScript Array Spread Operator

La realidad es que si ya que podemos usar el concepto de Spread Operator sobre un array y hacer lo siguiente:

```
var lista=[1,2,3,4];
var lista2=[...lista,5];
console.log(lista2);
```

En este caso caso usamos el Spread operator y convertimos nuestro array en un conjunto de

parámetros para un nuevo array.



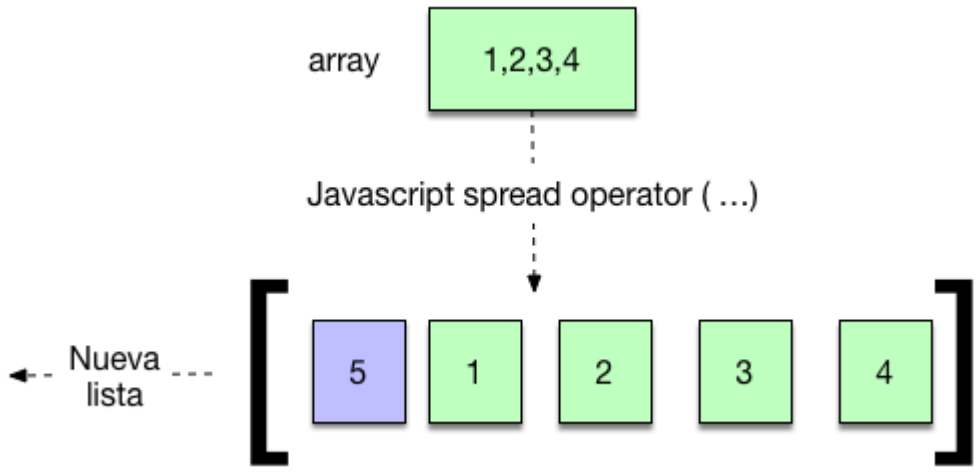
La sintaxis es más clara no cabe duda , sin embargo tampoco el aporte parece muy significativo simplemente es algo más cómodo. La realidad es un poco diferente ya que hay situaciones en las que queremos que el numero 5 sea el primer elemento del array. Uhmm ..¿Cómo hacemos eso? . Bueno tendremos que buscar alguna función de JavaScript que nos permite realizar dicha operación . En nuestro caso se trata de la función unshift … “super conocida por todos”. Así pues el código nos quedará de la siguiente forma si queremos que el elemento aparezca al principio.

```
var lista=[1,2,3,4];
lista.unshift(5);
console.log(lista);
```

El resultado se puede ver en la consola:

```
[ 5, 1, 2, 3, 4 ]
```

¿Lo podríamos haber hecho de una forma más sencilla con el spread operator? . La realidad es que si . Podríamos haber usado el elemento al principio del array y luego aplicar el spread operator:



Veámoslo en código:

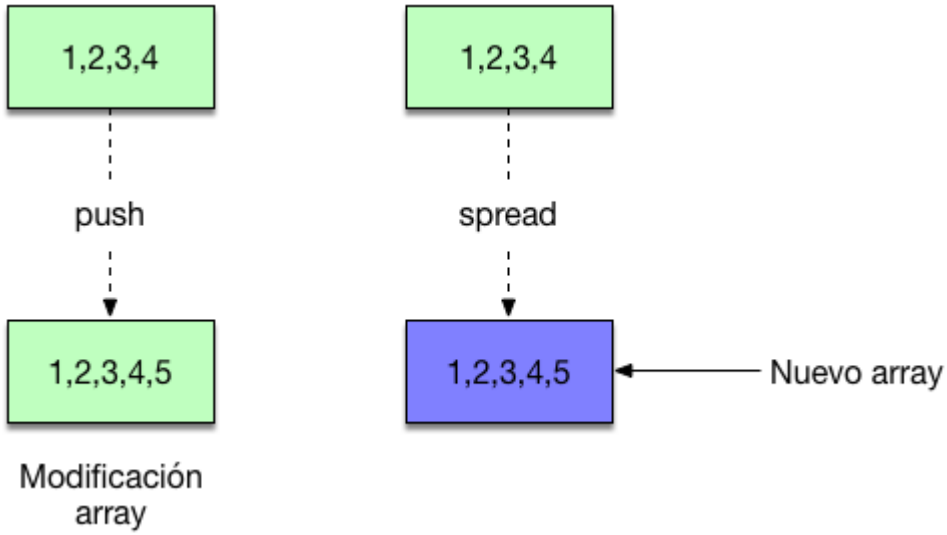
```
var lista=[1,2,3,4]
var lista2=[5,...lista]
console.log(lista2)
```

El resultado lo podemos ver en la consola y es idéntico al anterior:

```
[ 5, 1, 2, 3, 4 ]
```

JavaScript Array Spread Operator y otras ventajas

Otra de las cosas que tiene positivas el uso del spread operator cuando manejamos arrays es que promueve la inmutabilidad. Ya que cuando lo utilizamos se genera un nuevo array y no se modifica el anterior. Algo que si pasaba con el método push.



Veámoslo en código:

```
var lista=[1,2,3,4];
var lista2=[...lista,5];
console.log(lista);
console.log(lista2);
```

El resultado en la consola es :

```
[ 1, 2, 3, 4 ]
[ 1, 2, 3, 4, 5 ]
```

Es algo que también tenemos que tener en cuenta a la hora de manejarlo por las ventajas que aporta.

Otros artículos relacionados

1. JavaScript Template for loop y como usarlo

2. Utilizando JavaScript template String
3. JavaScript Console log y el manejo de Objetos
4. JavaScript setTimeout vs setInterval