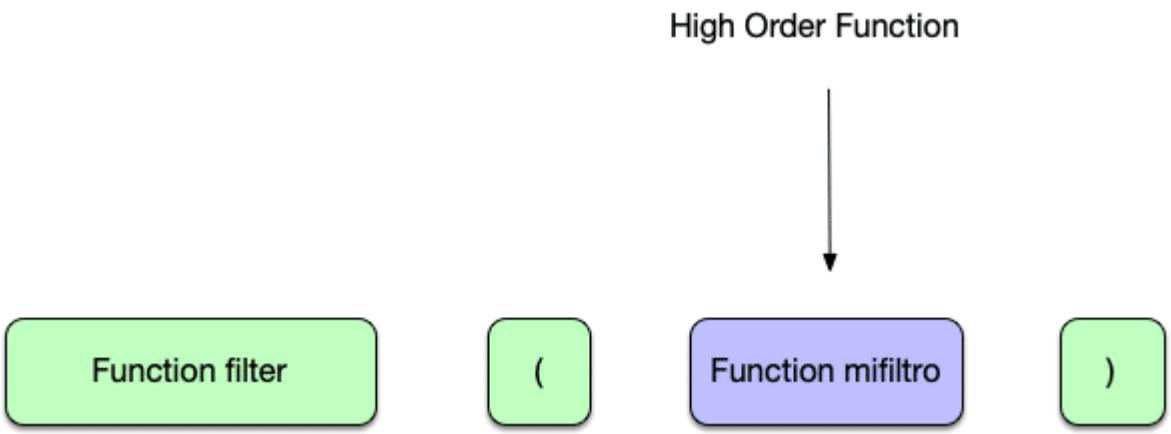
JavaScript filter map reduce . Son las High Order Functions más habituales cuando programamos en JavaScript . Mucha gente esta muy acostumbrada a utilizarlas , pero también hay mucha gente que cuando empieza con la programación funcional estas funciones le resultan cuanto menos costosas de entender . Vamos a explicarlas un poco desde cero .

Javascript filter map reduce (filtrados)

La primera que vamos a abordar es JavaScript filter . Esta función se encarga de filtrar los elementos de un array y quedarnos con aquellos que cumplan la condición.

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

let listaFiltrada=lista.filter(function(item) {

      return item>2;

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

Al principio lo que cuesta mucho entender es como una función como Javascript Filter recibe otra función como parámetro. Esto viene definido por el concepto de High Order Function , que hace referencia a una función que puede recibir entre sus parámetros una función o devolverla.



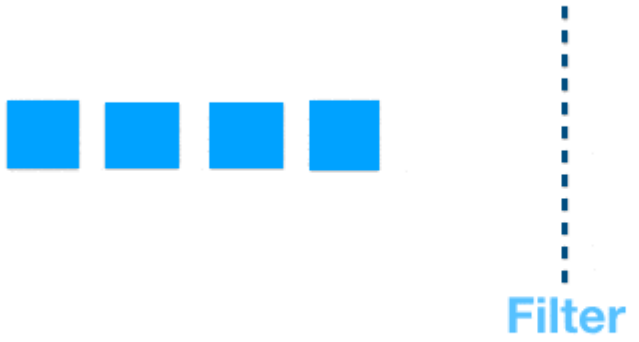
En nuestro caso vamos a ver un ejemplo muy muy sencillo . Partiendo de una lista de números en un array de JavaScript.

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

let listaFiltrada=lista.filter(function(item) {

      return item>2;

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

Como podemos observar el filtro que definimos nos hace quedarnos únicamente con los elementos mayores de 2. Si ejecutamos el código el resultado en la consola será:

```
iMac-de-cecilio-2:010Funcional
[ 3, 4 ]
iMac-de-cecilio-2:010Funcional
```

Por lo tanto esta primera función lo único que hace es definir la condición de filtrado para eliminar elementos de la lista.



JavaScript Map (Transformaciones)

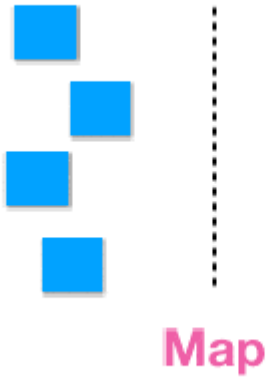
La siguiente operación que nos aparece es la de Javascript Map . Esta operación no filtra los elementos del array sino lo que hace es transformarlos y convertirlos en otros elementos a través del uso de una función.

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

let listaTransformada=lista.map(function(item) {

      return item*2;

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

En este caso lo que hemos hecho es multiplicar por dos cada uno de los elementos . Por lo tanto tendremos la lista con los valores duplicados al imprimirla por la consola.

```
iMac-de-cecilio-2:010Funcional cecilioalvarezcaules$ no
[ 2, 4, 6, 8 ]
iMac-de-cecilio-2:010Funcional cecilioalvarezcaules$
```

Acabamos de realizar una transformación sobre el array.



JavaScript Reduce (Acumulación)

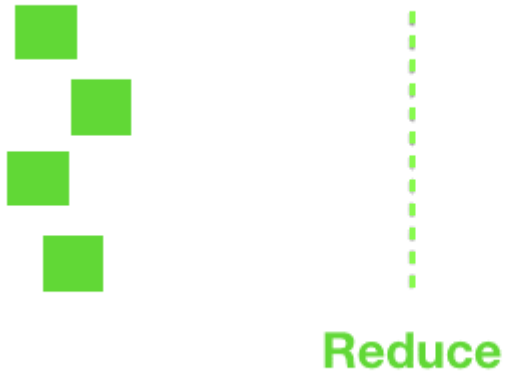
La última operación que nos queda por abordar es la operación de reducción esta operación una un High Order function para ir agrupando los valores según los procesa . Es decir nos vale para recorrer el array e ir acumulando un resultado final como puede ser un sumatorio. Vamos a verlo en código:

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

var suma= lista.reduce(function(acumulador,item) {

  return acumulador+item;

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

El resultado que muestra la consola es :

```
iMac-de-cecilio-2:010Funcional cecilioalvarezcaules$ no
10
iMac-de-cecilio-2:010Funcional cecilioalvarezcaules$
```

La operación de reducción aborda el concepto de acumulación :



Este es un resumen del uso de JavaScript filter map reduce.

Otros artículos relacionados

- Javascript Sincrono o Asíncrono
- JavaScript Console Profile
- JavaScript Destructuring
- High Order Functions