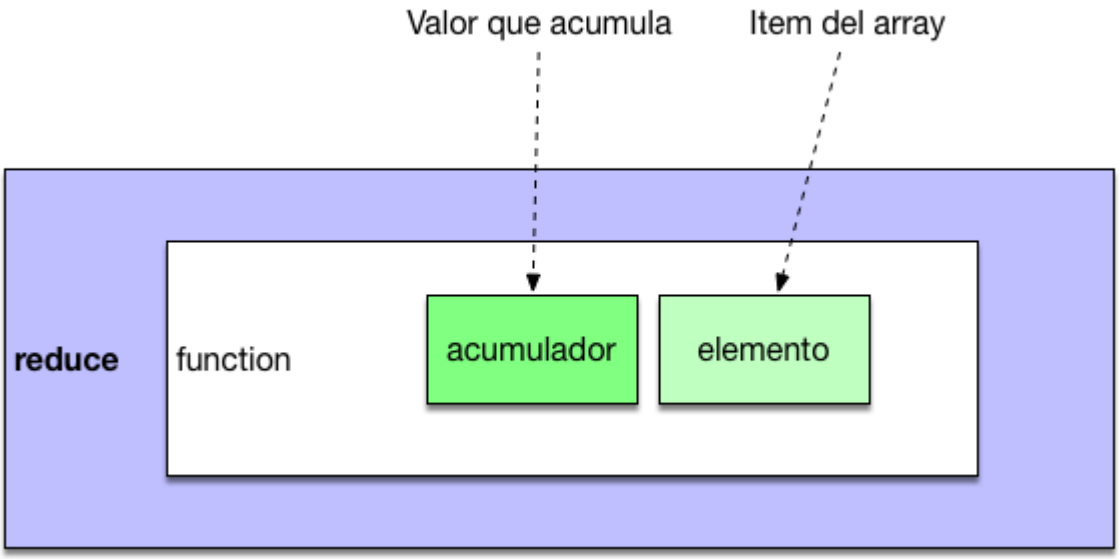
Tabla de Contenidos

El uso de JavaScript reduce es uno de los más habituales cuando hablamos de programación funcional . Por defecto esta función se encarga de “reducir” un conjunto de elementos a un único resultado. Sin embargo soporta bastantes más opciones que le aportan flexibilidad vamos a verlo partiendo del ejemplo más básico:

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

var resultado1= lista.reduce(function(total,valor) {

  return total+valor;

})
console.log (resultado1);
```

En este caso la función recibe dos argumentos. El primer argumento es el acumulador , en este caso le hemos llamado total , y el segundo argumento es el valor de cada elemento del array.

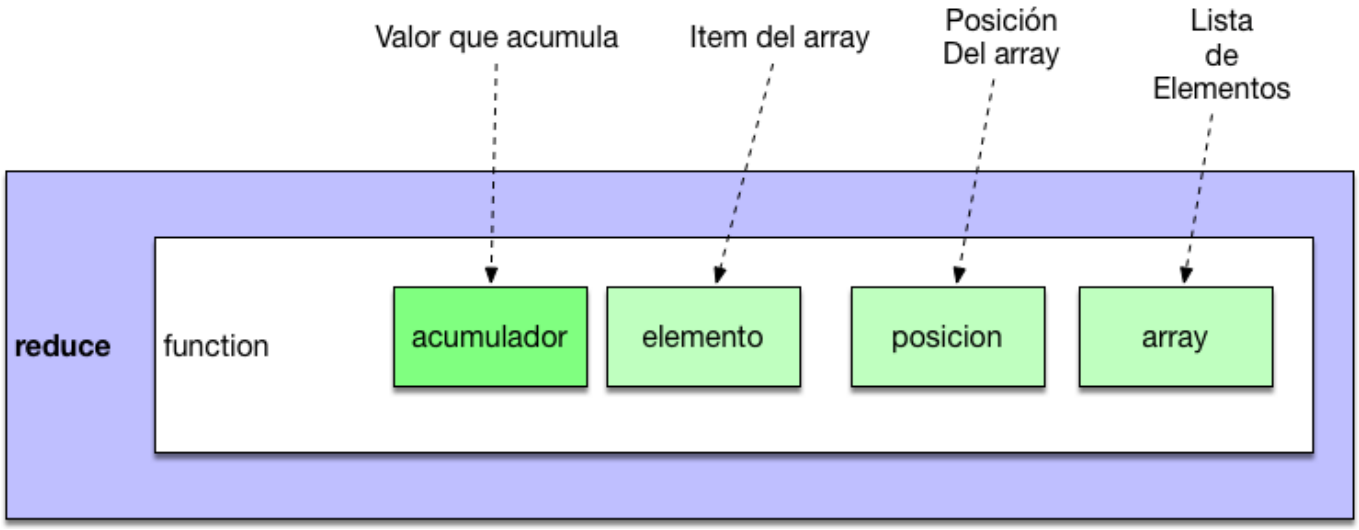


Con eso en principio nos es suficiente para realizar la suma de los primeros n elementos.

```
10
```

JavaScript Reduce y sus parámetros adicionales

Muchas veces nos olvidamos que de javaScript reduce soporta parámetros adicionales que le permiten ganar en flexibilidad y abordar operaciones más complejas que simplemente el sumar un conjunto de elementos. Estos parámetros son el indice del array y el propio array de elementos.



Vamos a usar estos parámetros adicionales para calcular el elemento mayor de un array e imprimirlo en la consola:

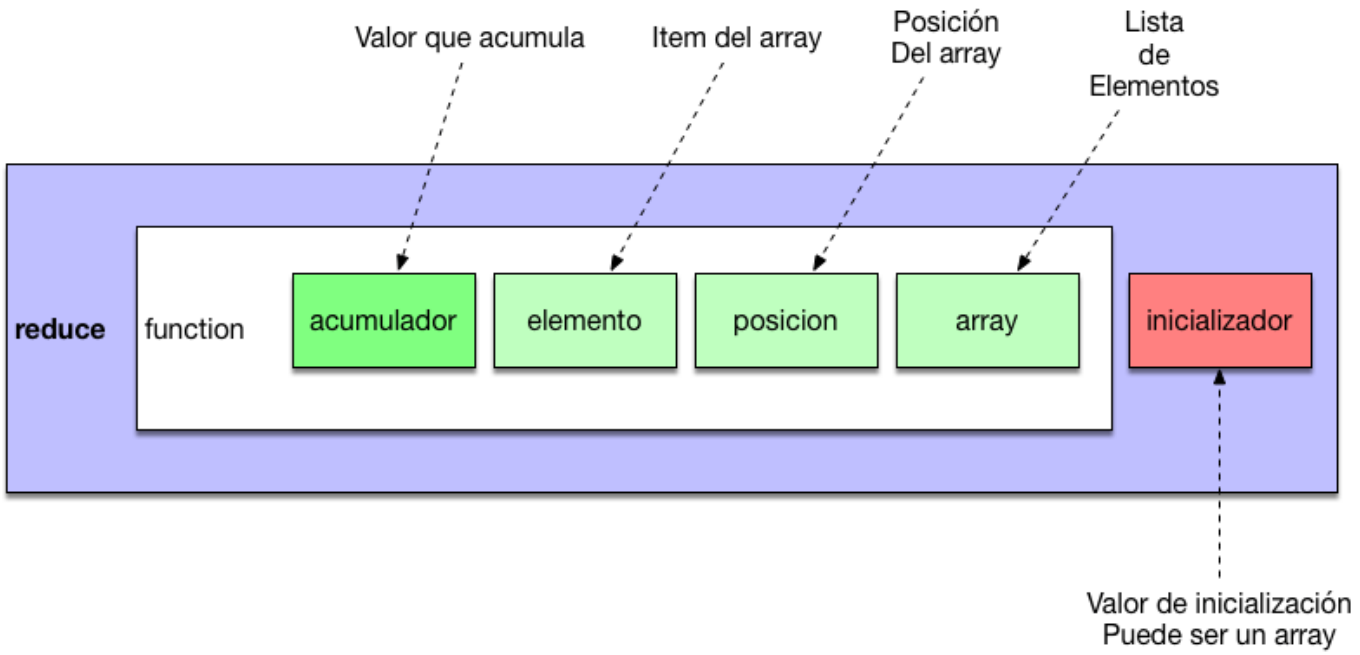
```
var mayor= lista.reduce(function(resultado,valor,indice,array) {


  if (indice< array.length){

    if (resultado<valor) resultado=valor;
  }

  return resultado;

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

El resultado lo podemos ver en la consola nos localiza el elemento mayor del array:

4

JavaScript Reduce e inicializadores

Existe otra opción bastante útil cuando trabajamos con javascript reduce y es definir cual va a ser el valor inicial que la reducción va a tener y en la que se va a apoyar. En muchas ocasiones no hay que ponerlo ya que se trata de un cero .Sin embargo hay situaciones en las que se inicializa con un array .



Esto aporta flexibilidad, veamos un ejemplo que nos convierte nuestro array de números en un array de numeros + el iva.

```
var resultado3= lista.reduce(function(resultado,valor) {


  resultado.push(valor*1.21);

  return resultado;

},[])
console.log(resultado3);
```

Veamos el resultado en la consola:

```
[ 1.21, 2.42, 3.63, 4.84 ]
```

JavaScript Reduce , flatMap

Como se puede observar tenemos el array pero con un nuevo conjunto de valores algo muy práctico y que normalmente realiza la funcion map . Sin embargo esto no es todo al poder acceder al array podemos realizar operaciones como la clásica de flatmap de Java de forma muy sencilla.

```
var lista2=[[1,2],[3,4]];

const aplanar = lista.reduce((total, valor) =&gt; {
  return total.concat(valor);
}, []);

console.log(aplanar);
```

El resultado le vemos aparecer en la consola como un único array.

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

El concepto de reducción es muy flexible y tenemos que habituarnos a utilizarlo.

Otros artículos relacionados:

1. JavaScript Template for loop y como usarlo
2. JavaScript Deferred Objects y peticiones asíncronas
3. JavaScript Array Prototype y extensibilidad
4. Usando Rx Observables en JavaScript
5. El concepto de JavaScript Spread operator

Externos

1. JavaScript Functional programming