# Map

O objeto Map contém pares de chave-valor e lembra a ordem original da inserção das chaves. Qualquer valor (objetos e valores primitivos) podem ser usados como chave ou valor.

* Cria um novo array
* Não modifica o array original
* Realiza operações em ordem

## Descrição

Um objeto Map itera seus elementos na ordem da inserção - um loop for...of retorna um array de [key, value] para cada iteração

Sintaxe:

**array.map(callback,thisArg);**

**calback(item, index, array);**

**Callback –** função a ser executada em cada elemento

**thisArg (opcional):** valor de ‘this’ dentro da função call-back

# Filter

O objeto Filter cria um novo vetor baseado no original que atendem às condições do filtro.

* Cria um novo array
* Não modifica o array original

Sintaxe:

**array.filter(call-back, thisArg);**

**Callback –** função a ser executada em cada elemento

**thisArg (opcional):** valor de ‘this’ dentro da função callBack



# Reduce

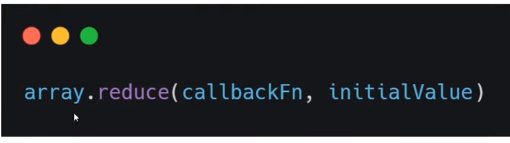
O objeto Reduce executa uma função em todos os elementos do array, retornando um valor único, podendo ser um array ou não.

Sintaxe:

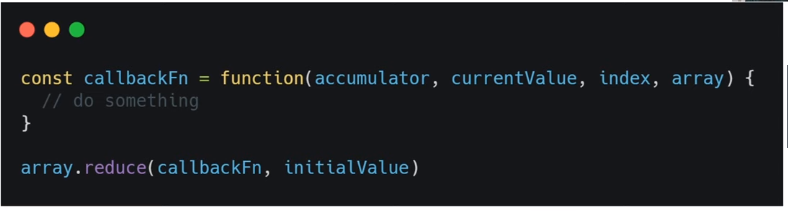
**array.reduce(callbackFn, initialValue);**

Callback – função a ser executada a partir do acumulador.

InitialValue – valor sobre o qual o retorno final irá atuar (opcional)



Exemplo:



**Accumulator / prevValue:** acumulador de todas as chamadas de callbackFn

**currentValue:** elemento atual sendo acessado pela função

neste exemplo funciona como uma função recursiva.