**Map**

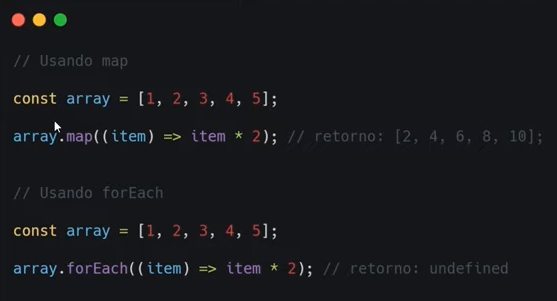
* é quando um conjunto passa por uma operação e se torna outro conjunto
* função **map** cria um novo array
* não modifica o array original
* realiza as operações em ordem

**Sintaxe:**

* **callback**: função a ser executada em cada elemento
* **thisArg (opcional)**: valor de “this” dentro da função de callback
* array.map(callback, thisArg)
* callback(item, index, array)

**Map vs forEach**

* valor de retorno
* considere se o array auxiliar será necessário



**Filter**

* é como se você colocasse todo seu array dentro de um filtro e apenas os itens que corresponderem a determinada condição serão retornados no novo array que será criado
* cria um novo array
* não modifica o array original

**Sintaxe**

* **call-back** – função a ser executada em cada elemento
* **thisArg (opcional)** – valor de “this” dentro da função de call-back
* *Array*.filter(callback, thisArg)
* const frutas = [ "maçã fuji", "maçã verde", 'laranja', 'abacaxi'];
* frutas.filter((*fruta*) => *fruta*.includes('maçã'))
* //retorno: ['maçã fuji', 'maçã verde'];

**Reduce**

* executa uma função em todos os elementos do array, retornando um valor único

**Sintaxe:**

* **callback** – função a ser executada a partir do acumulador
* **initialValue** – valor sobre o qual o retorno final irá atuar
* *Array*.reduce(callbackFn, initialValue)
* **accumulator/prevValue** – acumulador de todas as chamadas de callbackFn
* **currentValue** – elemento atual sendo acessado pela função
* const callbackFn = function (*accumulator*, *currentValue*, *index*, *array*) {
* //do something
* }
* *Array*.reduce(callbackFn, initialValue)