```
// ### Java Script - FUNÇÕES DE ALTA ORDEM  ###
// Capítulo: 03-03 (Funções de Alta Ordem)

const list1 = [1, 2, 3, 4];
const list2 = [];
const nomes = ['MARIA', 'JOAO', 'ANABELA'];

// ------------------------------------------------------------
// map: aplica uma função a cada elemento da coleção original,
// retornando uma nova coleção com os elementos alterados

function dobro(x){
    return x * 2;
}

function triplo(x){
    return x * 3;
}

/*
A chamada da função ".map" no array "list1" que era [1, 2, 3, 4],
passando como argumento a função "dobro", gerou uma nova lista, um
novo array, contendo os valores [2, 4, 6, 8], que é o dobro do array
original.
*/

console.log("MAP_____________________________________:")
const m1 = list1.map(dobro);
console.log(m1);

//Aqui iremos aplicar a função triplo no array "list1", gerando um novo array.
const m2 = list1.map(triplo);
console.log(m2);

//Aqui iremos aplicar a função anônima.
const m3 = list1.map(x => x * 2);
console.log(m3)

// ------------------------------------------------------------
// filter: retorna uma nova coleção contendo apenas
// aqueles elementos da coleção original que
// satisfazem um dado predicado

//Aqui temos uma função "PAR" que irá retornar (true ou false) se um número é par
function par(x){
    return x % 2 == 0;
}

// Então iremos aplicar esta função ".par()" para FILTRAR ".filter()" uma lista
original
// resultando apenas nos números pares dessa lista.
// Como ficaria.
console.log("FILTER_____________________________________:")
const f1 = list1.filter(par);
console.log(f1);
```

```
// A constante f1 "const f1"
// Recebe a "list1" [1, 2, 3, 4]
// ".filter(arg0)" => O ".filter()" irá aplicar o efeito da função que estiver
//dentro do () aos elementos da lista "list1".
//arg0 => Representa a função que será utilizada na função ".filter()".
//Ou seja, a constante f1 irá receber o valor da lista "list1" filtrada pela função
"par"
//com isso irá retornar apenas OS NÚMEROS PARES DA list1. que são [2, 4].

//Também é possível utilizar a função Lambda
//Ou seja, irá testar se um dado valor de X é par.
// Lê-se: Para cada X que leva X mó de 2 igual a 0.
const f2 = list1.filter(x => x % 2 == 0);
console.log(f2);

//Outro exemplo:
const f3 = list1.filter(x => x > 2);
console.log(f3);

// ------------------------------------------------------------
// reduce: aplica cumulativamente uma função aos elementos de
// uma coleção, retornando o resultado final cumulativo.
// * você pode informar, opcionalmente, um valor inicial como
// parâmetro (necessário para coleção vazia).

function soma(x, y){
    return x + y;
}

function produto(x, y){
    return x * y;
}

// Neste caso o REDUCE irá somar os valores da lista "list1"
// retornando o resultado desta expressão em um único algarismo. Veja:
// A list1 = [1, 2, 3, 4]
// A função REDUCE com a função "soma" aplicada a ela irá retornar
// o valor "10" que é a soma de 1, 2, 3 e 4.

console.log("REDUCE_____:")
const r1 = list1.reduce(soma);
console.log(r1);

//A função Reduce possui dois argumentos ".reduce(arg0, arg1)"
//O arg0 é a função que será a origem da expressão reduce. Conforme exemplo acima.
//O arg1 é o ELEMENTO NEUTRO.
//O ELEMENTO NEUTRO, é obrigado no caso da "REDUÇÃO" de uma lista "VAZIA"
// conforme a "list2 = [ ]"
//A função cumulativa ".reduce(arg0)" sob uma lista vazia não faz sentido.
//Por este motivo é necessário um segundo argumento "arg1", como 0 para retornar
//"0" caso a lista seja vazia.
//Ficando assim:
const r2 = list2.reduce(soma, 0);

const r3 = list1.reduce(produto, 1);
console.log(r3)
//list1 = [1, 2, 3, 4]  =  1 x 2 x 3 x 4 = 24
//Observe que o elento neutro do produto é 1.
const r4 = list2.reduce(produto, 1);
console.log(r4)
```

```
118 // ------------------------------------------------------------
119 // sort: ordena a coleção conforme a função de comparação
120 // informada como parâmetro
121
122 function compararPorTamanho(s1, s2) {
123     return s1.length - s2.length;
124 }
125
126 console.log("SORT ----------------------------");
127
128 const s1 = [...nomes].sort();
129 console.log(s1);
130
131 const s2 = [...nomes].sort(compararPorTamanho);
132 console.log(s2);
133
134 const s3 = [...nomes].sort((x, y) => x.length - y.length);
135 console.log(s3);
136
```