* Array: nos permitem que possamos agrupar vários tipos de dados em uma única variável, tendo em conta que cada um desses dados se encontra em uma determinados índices do nosso array
* **.push():** permite que adicione-se um dado no final do array. É executado sobre um array já existente e recebe como um parâmetro um ou mais elementos separados por vírgula que serão inseridos no final do array, aumentando a quantidade de índices
* **.pop():** não recebe nenhum parâmetro e sempre retira o último elemento do array
  + Mas para onde vai o elemento que você retirou do array? O .pop() além de retirou o último elemento do array, ele também retorna esse dado. Então se você precisa do elemento o melhor a se fazer é armazenar esse dado dentro de uma variável utilizando o pop.
* **.shift():** remove do array o elemento localizado no índice “0” e também e retorna esse dado
* **.unshift():** insere um ou mais dados ao início do array.
* **.indexOf():** recebe uma string de texto que será consultada dentro do array
  + Retorna -1 quando não acho o que está sendo procurado
  + Sempre vai retornar a primeira ocorrência que ele encontrar
* **.lastIndexOf():** faz exatamente o mesmo do anterior, porem começa a sua busca pelo último elemento da array.
* **.join():** permite unificar todos os elementos presentes no array em uma única string separando cada um dos valores por uma virgula
  + Nele posso escolher qual o caractere será o utilizado nessa separação .join(“ – “)
* **.include():** similar ao o indexOf mas retorna true ou false
* **.map():** se executa sobre um determinado array e recebe como parâmetro um call-back. O call-back executa um a um os elementos desse nosso array tomando como parâmetro cada um desses elementos e é responsável por retornar um array novo com a mesma quantidade de elementos do array original.
* **.filter():** ele é responsável por filtrar cada um dos elementos do nosso array dependendo de alguma lógica que deverá ser implementada. Dessa forma o filtro passa pelo array especificado tomando cada um dos seus elementos e retorna apenas com os elementos que atendem a nossa condição definida (diferente do map). É importante que nossa função retorne um booleano. Irá retornar um array.
* **.reduce():** é executado sobre um determinado array e recebe como um parâmetro um call-back visando reduzir o array fornecido à mínima expressão, retornando como resultado um único valor. O call-back vai receber dois parâmetros, um acumulador e uma elemento atual. O acumulado é o primeiro valor, e o atual o segundo.
* **.forEach():** iterar sobre um array. Não retorna nada.
* **.sort():** ordena o array em ordem alfabética
* **.reverse():** inverte o array
* **For in** só itera sobre objetos literais
  + let objeto = {nome: “bla”, idade: “bla”}
  + for (let dados in objeto) {

console.log(dados)} -> vai retornar nome, e idade

* + for (dados in objeto) {

console.log(objeto[dado])} -> vai retornar “bla”

* **For of** só itera sobre array e strings
* Objeto date: let dataAtual = new Date()
* getDay(): retorna o número referente ao dia da semana. 0 = domingo, 1 = segunda...
* getDate(): retorna o dia do mês
* getMonth(): retorna o mês. Janeiro = 0, dezembro = 11
* getFullYear()
* Alocação por desestruturação: Extração de dados de arrays ou objetos literais. O javascript irá criar as três variáveis mencionadas e em cada uma delas será atribuído o valor correspondente que existe dentro do nosso array, mantendo a mesma lógica presente no nosso array
  + A alocação por desestruturação não modifica nem altera o array de origem, pois o seu objetivo é só copiar os valores
* O operador Spread serve para copiar ou mover dados de uma array para outro
* Podemos usar o operador spread sobre qualquer elemento iterável
* Podemos usar o operador spread para definir funções, e nesse contexto chamamos de Parâmetro Rest
* Implementando o parâmetro rest podemos definir uma função que aceite qualquer número de parâmetros
* O parâmetro rest se escreve sempre no final como último parâmetro