JAVASCRIPT: ARRAY

AULA 01 ARRAY - ARQUIVO: media.js

```
const notas = [10, 6.5, 8, 7.5];

let media = (notas[0] + notas[1] + notas[2] + notas[3]) / notas.length

console.log(media)
Saida: 8
```

ARRAY

Ao contrário de uma variável, que guarda somente um valor por vez, um array (ou lista) pode armazenar diversos valores. Pode ser usada, por exemplo, para agrupar diversos dados que têm relação entre si.

Um array pode ser definido como uma lista ordenada de valores enumerados em que cada valor é chamado de elemento, e cada elemento se localiza numa posição na lista chamada de índice.

AULA 01 VIDEO 02 - ARQUIVO adicionando_notas.js

UTILIZANDO O METODO PUSH

Ele cria um novo elemento no final da array, neste caso o exemplo abaixo, o número passado foi o 7 entre parênteses, o que chamamos de parâmetros. Se não for passado um parâmetro dentro desse método vai ocorrer o erro de NaN - Not A Number.

```
const notas2 = [10, 6, 8];
notas.push(7)
console.log(notas2)
 saída: [ 10, 6, 8, 7 ]
```

AULA 01 VIDEO 03 - ARQUIVO removendo-nota.js

METODO É O .pop()

remove o último elemento, e não precisa utilizar nenhum parâmetro e ele vai funcionar normalmente, ele não aceita nenhum parâmetro.

```
const notas3 = [10, 7, 8, 5, 10]
notas3.pop()
console.log(notas3)  Saida: [ 10, 7, 8, 5 ]

let media3 = (notas3[0] + notas3[1] + notas3[2] + notas3[3]) /
notas3.length
```

```
console.log(` A média é ${media3}`)  Saida: A média é 7.5
```

OUTROS METODOS PARA ARRAY

concat()
Junta dois arrays, colocando o array passado como argumento, logo depois do primeiro. Em português essa operação é conhecida como concatenação.
Não altera o array no qual foi chamado, então precisamos salvar esse resultado em um novo array.

filter()
Retorna uma lista contando todos os elementos que passaram em um teste, ou seja, uma função escrita por nós.
Não altera o array onde foi chamado, então precisamos salvar esse resultado em um novo array.

find()
Funciona de forma parecida com o filter, porém retorna apenas o primeiro valor que satisfizer o teste, podendo ser uma string ou um número.

findIndex()
Funciona igual o find(), mas retorna o índice em vez do elemento, possibilitando usá-lo em outras partes do código.

indexOf()
Localiza e retorna o índice referente à primeira ocorrência de determinado valor em um array. Caso não exista nenhuma ocorrência do valor, retorna -1.

lastIndexOf()
Funciona da mesma forma que o indexOf(), porém retorna o índice referente à última ocorrência de um valor em um array, varrendo o array de trás para frente.

forEach()
Executa uma função em cada elemento do array de forma individual.
Não altera o array original e nem retorna um valor, deixando esse trabalho a cargo da função escolhida.

shift()
Retira o primeiro elemento do array.
Altera o array original removendo o elemento e trocando o índice de todos os elementos para um a menos do que eram, o índice 1 passa a ser o 0, o 2 passa a ser o 1, e assim por diante.

unshift()
Funciona igual ao push(), porém adiciona na primeira posição e acaba trocando o índice de todos os elementos.

Altera o array original com o novo valor.

reduce()
Utiliza uma função definida pelo usuário em cada um dos elementos, guardando o resultado em uma variável que pode ser acessada dentro da função que foi definida, retornando um único valor no final, reduzindo o array para um único valor.

reduceRight()
Funciona igual o reduce() porém começa do final do array e segue até o início.

reverse()
Inverte a ordem dos elementos do array, então o primeiro vira o último, o segundo o penúltimo e assim por diante.

slice()
Copia uma parte do array para outro array.

sort()
Organiza o array de acordo com a classificação Unicode, onde os números vêm antes das letras, porém não funciona corretamente para números, onde temos que definir uma função que irá auxiliar o comando.

splice()
Consegue remover, um ou mais elementos consecutivos caso o segundo parâmetro tenha um valor maior que 0, e incluir um ou mais elementos a partir de um índice escolhido.

AULA 02 VIDEO 01 - ARQUIVO dividindo-arrays.js

UTILIZANDO O METODO .SLICE() - Cria uma array Nova.
O método vai utilizar dois parâmetros, o primeiro serve para indicar o inicio do corte, e o segundo até onde vai ser esse corte, quando não passamos o segundo parâmetro ele vai até o final.

```
const nomes = [ 'João', ' Juliana', ' Ana', ' Caio', ' Lara', '
Marjorie', ' Bruno', ' Guilherme', ' Aline', ' Fabiana.',
'Andre ', 'Carlos ', 'Paulo ', 'Bia ', 'Viviam ', 'Isabele ', 'Vinicius
', 'Renan ', ' Daisy', ' Camilo.']

const sala1 = nomes.slice(0, nomes.length/2)
const sala2 = nomes.slice(nomes.length/2)

console.log(`Alunos da Sala 1: ${sala1}`)
```

```
saída: Alunos da Sala 1: João ,Juliana ,Ana ,Caio, Lara, Marjorie, Bruno,
Guilherme, Aline, Fabi
a
console.log(`Alunos da Sala 2: ${sala2}`)
saída: Alunos da Sala 2: André ,Carlos ,Paulo ,Bia ,Viviam ,Isabeli
,Vinicius ,Renan ,Daisy ,Camilo
```

AULA 02 VIDEO 02 - ARQUIVO: atualizando-lista.js

Método é o .splice() - Não Cria uma Array Nova.
Esse método permite que possamos remover itens e coloque outro no lugar do qual retiramos, o primeiro parâmetro é o índice onde será o início, o parâmetro dois é a quantidade de elementos que desejamos remover a partir do ponto inicial que definimos, no último parâmetro serve para colocar o elemento que desejamos inserir na array.

Exemplo splice (parâmetro 1 - início da remoção, parâmetro 2 - quantidade de elementos para serem removidos, parâmetro 3 - item a ser colocado.)

OBS:
Podemos passar esse método sem o terceiro parâmetro e ele vai funcionar normalmente, caso não desejarmos inserir nenhum elemento, também podemos inserir algum elemento sem precisar remover algum utilizando zero no parâmetro 2.

```
const listaDeChamada1 = ['Mariazinha', ' Juvelnal', ' Fernando', '
Caio','Karol',' Lara']

listaDeChamada1.splice(1,2, 'Santos')
console.log(`Lista de Chamada: ${listaDeChamada1}`)
//Saida - Lista de Chamada: Mariazinha, Santos, Caio,Karol, Lara

//COLOCANDO ZERO NO SEGUNDO PARAMETRO.

const listaDeChamada2 = ['João', ' Pedro', ' thiago', ' Maria','
Selene',' Ruanito']

listaDeChamada2.splice(2,0, ' Bruno')


console.log(`Lista de Chamada: ${listaDeChamada2}`)
Saida - Lista de Chamada: João, Pedro, Bruno, thiago, Maria, Selene,
Ruanito
```

AULA 02 - VIDEO 03 - ARQUIVO: juntando-turmas.js

Metodo .concat() - Cria uma array Nova.

Primeiro selecionamos uma das arrays e dentro do método inserimos uma outra arrays como parâmetro.

```
const salaDePython = ['Melissa', 'Selene', 'Bruno']

const salaDeJavaScript = [ 'Karol', 'Roberto', 'Rafael']

const salasUnificadas = salaDePython.concat(salaDeJavaScript)

console.log(salasUnificadas)
```

Saída - ['Melissa', 'Selene', 'Bruno', 'Karol', 'Roberto', 'Rafael']

AULA 02 - VIDEO 04 - ARQUIVO: lista-duas-dimensoes.js

MATRIZES

Este recurso se chama lista com duas dimensões porque se trata de uma lista com outras listas dentro dela.

```
const alunos = [ 'Miguel', ' Ruanito', ' Ponga', ' Milena']

const mediasDosAlunos = [ 10, 7, 9, 6]

let listaDeNotasEAlunos = [ alunos, mediasDosAlunos]

console.log( `${listaDeNotasEAlunos[0][0]}, sua media é
${listaDeNotasEAlunos[1][0]}`)
```

Saída: Miguel, sua média é 10

Para ter acesso aos elementos de cada array:
listaDeNotasEAluno
    [0 - Acessa o indice zero da array listaDeNotasEAlunos que é a Array Alunos]
    [0 - Acessa o indice zero da array alunos que é Miguel]
listaDeNotasEAlunos
    [1 - Acessa o indice 1 da array listaDeNotasEAlunos que é mediasDosAlunos]
    [0 - Acessa o indice zero da arrays Alunos = 10]

AULA 03 - VIDEO 01 - ARQUIVO: localizando-aluno.js

METODO .includes() E indexOf()

```
.includes - ele busca uma informação dentro de uma array, e vai retornar
se o que procuramos esta nesta lista, se estiver o resultado será true e
segue-se para dentro do bloco de código, se false ele encerra a busca.
includes -> booleano
indexOf - retorna o número do índice

const alunos = [ 'Miguel', 'Ruanito', 'Ponga', 'Milena']
const mediasDosAlunos = [ 10, 7, 9, 6]

let listaDeNotasEAlunos = [ alunos, mediasDosAlunos]

const exibeNomeNota = (nomeDoAluno) => {
    if (listaDeNotasEAlunos[0].includes(nomeDoAluno)){
        let indice = listaDeNotasEAlunos[0].indexOf(nomeDoAluno)
        return listaDeNotasEAlunos[0][indice] + ', sua média é: '  +
listaDeNotasEAlunos[1][indice]
    } else {
        return 'Aluno(a) não cadastrado.'
    }
}
console.log(exibeNomeNota('Ponga'))
SAIDA: "Ponga, sua média é: 9"

Dentro do if temos como condicional o método includes, este vai fazer uma
busca pelo tipo de dado que passamos para ele como parâmetro, será feita
uma varredura na array listaDeNotasEAlunos no índice [0], que acessa a
array alunos e vai procurar pelo dado desejado, caso esteja, vai entrar
no bloco de códigos do if onde temos uma variável para armazenar o índice
que entrou neste bloco com método indexOf(),  que fará o processo de
trazer este índice para ser armazenado, o que nesse caso vamos receber
como resultado índice 2.

o return dentro do if vai trazer o nome e nota do aluno
[0 - alunos] [índice vai ser o 2]
E a nota do Aluno
[1 - mediasDosAlunos][índice vai ser o 2]
*/
```

AULA 03 - VIDEO 02 - ARQUIVO: for.js

```
Para exibir o número do índice passamos para o console a variável criada
para fazer o loop, no exemplo a let i(variável de controle para o for)
que está armazenado a cada volta o valor do índice incrementando de 1 em
1 Exmplo: 0, 1, 2, 3...
Para ver o conteúdo, utilizamos a array que desejamos acessar e coloca-se
dentro de colchetes o índice que queremos acessar, nesta situação pega-se
a variável do for, que no caso é o (i) que será os índices da array que
estamos utilizando.
Exemplo: números[i] - assim vai ser exibido todos os conteúdos, mediante
a passagem de cada índice, já que o for vai incrementa a variável i de um
em um, índice 0, 1, 2..., vai exibindo o elemento que estiver na posição
do índice atual. No caso da const numero[0] na posição zero, o elemento é
o 100.
*/

const numeros = [ 100, 200, 300, 400, 500, 600, 700, 800]

for (let i = 0; i< numeros.length; i++){
    console.log(i, numeros[i])
}
```

```
SAIDA:
0 100
1 200
2 300
3 400
4 500
5 600
6 700
7 800
```

AULA 03 - VIDEO 03 - ARQUIVO: media-for.js

```
      O for nesta aula vai fazer uma soma!
      Temos uma variável externa ao loop(somaDasNotas) para armazenar o
valor da soma das notas a cada iteração do for, dentro do bloco, temos a
variável (somaDasNotas) realizando a soma e recebendo as informações da
array notas[i], que a cada passagem do for pelos índices, os elementos
estão sendo somados, e armazenado na variável externa somaDasNotas.

const notas = [ 10, 6.5, 8, 7.5];

let somaDasNotas = 0;

for (let i=0; i < notas.length; i++){
    somaDasNotas += notas[i];
```

```
    console.log(somaDasNotas)
                                    SAIDA:
                                      10
                                     16.5
                                     24.5
                                      32

}

let media = somaDasNotas/notas.length;

console.log(media)
SAIDA: "8"
```

AULA 03 - VIDEO 04 - ARQUIVO media-foreach.js

ForEach - passa por todos os elementos de uma array, a principal característica dessa função é não precisar passar os números dos índices de uma array, ela irá percorrer toda a array automaticamente do início ao fim, nesta situação estamos somando todos os elementos e salvando na variável criada fora do bloco de código do ForEach, neste Caso a (somaDasNotas).
Podemos utilizar com uma arrays function:

```
                        notas.forEach( nota => {
                          somaDasNotas += nota
```

Ela é uma função call-back - porque o parâmetro da função não é um dado, por isso ela é uma função que chama outra, o parâmetro passado nela deve ser uma outra função e não uma string, ou number...

```
const notas2 = [ 10, 6.5, 8, 7.5]

let somaDasNotas2 = 0

Utilizando com uma função anonima!
notas2.forEach(function(nota){
 somaDasNotas2 += nota
})
let media2 = somaDasNotas2/notas.length

console.log(media2)
SAIDA: 8
```

AULA 04 - VIDEO 01 - ARQUIVO: callback.js

Só podemos passar como parâmetro para o forEach uma outra função, ele não vai funcionar se tentarmos passar outros tipos de dados, ela só recebe como parâmetro apenas função, podemos passar um segundo parâmetro que é para ter acesso ao índice.

```
const nome = ["Bruno", "Karol", "Ana", "Leo",]
nome.forEach(imprimeNomes)

function imprimeNomes(nome) {
    console.log(nome)
     SAIDA: "Bruno, Karol, Ana, Leo, Joaquim"
                                }
```

AULA 04 - VIDEO 02 - ARQUIVO: ponto-extra.js

o método map() é utilizado para adicionar mais 1 em cada nota, este metodo recebe uma função como parâmetro e vai criar uma outra array atualizada.

```
const notas = [ 10, 9, 8, 7, 6];

const notasAtualizadas = notas.map( nota => nota == 10? nota : ++nota)

console.log(notasAtualizadas)
SAIDA: "[10, 10, 9, 8, 7]"
```

AULA 04 - VIDEO 03 - ARQUIVO: map-nomes.js

Pegamos o .map() com o método toUpperCase() para alterar cada letra dos nomes para maiúscula, já que o método map só recebe como parâmetro uma outra função, o map entra na array varrendo cada índice e faz essa alteração com o método toUpperCase().
toUpperCase() é um método para alterar uma string de minúscula para Maiúscula.

```
let nomes2 = [ "ana Julia", " Caio vinicius", "BIA silva"]
const nomesAtualizados2 = nomes2.map(nome => nome.toUpperCase())

console.log(nomesAtualizados2)
SAIDA: "[ 'ANA JULIA', ' CAIO VINICIUS', 'BIA SILVA' ]"
```

AULA 05 - VIDEO 01 - ARQUIVO: reprovados.js

O método .filter() recebe como parâmetro uma função.
Ele pode receber mais um outro parâmetro que é o índice da array que o método está sendo utilizado, (aluno - primeiro parâmetro acessa o elemento, neste exemplo são as notas, índice - Segundo parâmetro acessa o índice, como por exemplo o 0, 1 ,2, 3...), o primeiro parâmetro não está sendo utilizado, podemos colocar um "_" no lugar de aluno que a função vai rodar normalmente, o método filter está retornando o nome Marcos da arrays nomes porque o método está associado a ela.

```
const nomes = [ 'Ana', 'Marcos', 'Maria', 'Mauro'];
const notas = [ 7, 4.5, 8, 7.5];

const reprovados = nomes.filter ( (aluno, indice) => notas [indice] < 5)
console.log( `reprovados: ${reprovados}`)
SAIDA: "reprovados: Marcos"
```

AULA 05 - VIDEO 02 - ARQUIVO: media-geral.js

Metodo .reduce() para Soma Elementos de uma Array.

o reduce comprime uma array somando os elementos dentro dela, um-a-um, para apenas um único valor, o 0 posiciona o ponto inicial do loop, acum e atual são parâmetros da arrow function, podemos passar qualquer nome a eles.
O primeiro parâmetro é um acumulador, vai acumular as notas de cada elemento, o segundo parâmetro será o valor atual, após cada loop será feita a soma atual + acum.

```
const salaJS = [ 7, 8, 8, 7, 10, 6.5, 4, 10, 7];
const salaJava = [ 6, 5, 8, 9, 5, 6];
const salaPython = [ 7, 3.5, 8, 9.5];


function mediaSala( notaDaSala){
    const somaDasNotas = notaDaSala.reduce((acum, atual) => atual +
acum,0)
    return somaDasNotas/notaDaSala.length
}

console.log( `Média da Sala de JavaScript: ${ mediaSala(salaJS)}`)
console.log( `Média da Sala de Java: ${ mediaSala(salaJava)}`)
console.log( `Média da Sala de Python: ${ mediaSala(salaPython)}`)
SAIDAS:
"Média da Sala de JavaScript: 7.5
Média da Sala de Java: 6.5
Média da Sala de Python: 7"
```