FIAP

GRADUAÇÃO

45697056

TDS

Responsive Web Development

Prof. Alexandre Carlos profalexandre.jesus@fiap.com.br

Prof. Luís Carlos lsilva@fiap.com.br

Prof. Renato Bortolin renatobortolin@fiap.com.br

Manipulação de Array

45697056

Para darmos procedimento e aumentar as possibilidades de uso do nosso código, vamos entender melhor como manipular arrays. Um array nada mais é que uma variável onde é possível armazenar vários valores, isso nos possibilita trabalhar com grandes quantidades de informações de um determinado tipo de forma mais simples, leve e performática.

```
let aluno1 = 'João'
let aluno2 = 'Carlos'
let aluno3 = 'Maria'
```

Usando variáveis simples para armazenar valores do mesmo tipo.

```
let aluno = ['joão','Carlos','Maria']
```

Usando array para armazenar valores do mesmo tipo.

Obs. Imagine se fossem dezenas ou centenas de valores....

Manipulação de Array

45697056

Neste array podemos guardar qualquer tipo de elemento, desde uma simples string, outros arrays ou até objetos.

Array de strings.

```
let aluno = ['joão','Carlos','Maria']
```

Array de arrays.

```
let grupos = [['Laura','Letícia'],['Pedro','Gustavo']]
```

Array de objetos.

```
let carros = [
    {'marca':'Honda','modelo':'Civic'},
    {'marca':'Toyota','modelo':'Corolla'},
    {'marca':'GM','modelo':'Cruze'}
]
```

Manipulação de Array PUSH( )

45697056

Para inserirmos um novo elemento a nosso array, podemos inserir alocar na próxima posição, ou pedirmos para que ele faça isso por nós utilizando o método push().

```
aluno[3] = 'Barbara'
console.log(aluno); // João, Carlos, Maria, Barbara
```

Adicionando uma nova posição ao array

```
aluno.push('Lucas')
console.log(aluno); // João, Carlos, Maria, Barbara, Lucas
```

Ultilizando o método push( ).

Manipulação de Array SORT( ) e POP( )

45697056

Podemos ordenar o conteúdo dos array, utilizando o método sort( ), perceba que agora está em ordem alfabética.

```
aluno.sort()
console.log(aluno); // "Barbara", "Carlos", "Lucas", "Maria", "joão"
```

Se precisarmos remover o último elemento, podemos usar o método pop( ), ao remover o último elemento, nós podemos guarda-lo em outra variável se quisermos.

```
alunoDesistente = aluno.pop()
console.log(aluno); // "Barbara", "Carlos", "Lucas", "Maria"
console.log(alunoDesistente); // João
```

Manipulação de Array UNSHIFT( ) e SHIFT( )

45697056

Podemos inserir um elemento na posição inicial do array com o método unshift( ).

```
aluno.unshift('Igor')
console.log(aluno); // "Igor", "Barbara", "Carlos", "Lucas", "Maria"
```

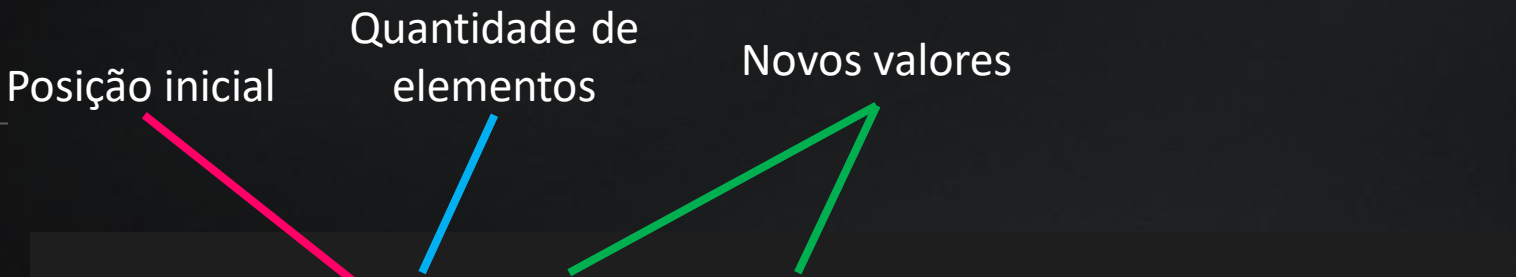
E para remover o elemento da primeira posição usamos o método shift( ).

```
aluno.shift()
console.log(aluno); // "Barbara", "Carlos", "Lucas", "Maria"
```

Manipulação de Array SPLICE( )

45697056

Se quisermos alterar ou remover um ou mais elementos de uma posição específica do array podemos utilizar o método splice( ).

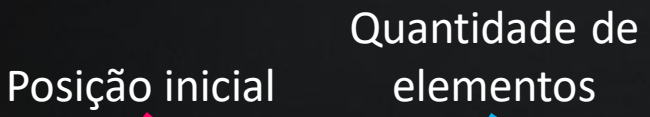



```
aluno.splice(1,2,"Cláudio","Débora")
console.log(aluno); // "Barbara", "Cláudio", "Débora", "Maria"
```

Manipulação de Array SPLICE( )

45697056

Com splice também podemos apagar um ou mais itens do array.




```
aluno.splice(1,1)
console.log(aluno); // "Barbara", "Débora", "Maria"
```

Perceba que como não passamos os valores para substituir ele acaba apagando apenas

Manipulação de Array – método MAP( )

45697056

O método map permite criar um novo array a partir de um array já existente, podendo manipular seu conteúdo através de uma função de callback.

```
const cursos = [
    {'nome':'HTML5','duracao':'3 meses'},
    {'nome':'CSS3','duracao':'4 meses'},
    {'nome':'Javascript','duracao':'5 meses'}
]

console.log(cursos); //exibe todos os objetos do array

const nomeCursos = cursos.map(cursos => cursos.nome)

console.log(nomeCursos); // arrays apenas com os nomes dos cursos

const propgCursos = cursos.map(cursos => `O ${cursos.nome} só dura ${cursos.duracao}`)
//arrays manipulando o conteúdo
for(let cr in propgCursos) console.log(propgCursos[cr]);
```

Manipulação de Array – método MAP( )

45697056

No método map, a função de call-back também pode receber um segundo parâmetro, se é a posição do elemento no array, podendo ser usado para criar uma identificação única do elemento..

```
const indiceCursos = cursos.map((cursos,i) =>
    `O ${cursos.nome} deve ser o ${i+1}º a ser feito.`)

for(let i in indiceCursos) console.log(indiceCursos[i]);
```

Manipulação de Array – método FILTER( )

45697056

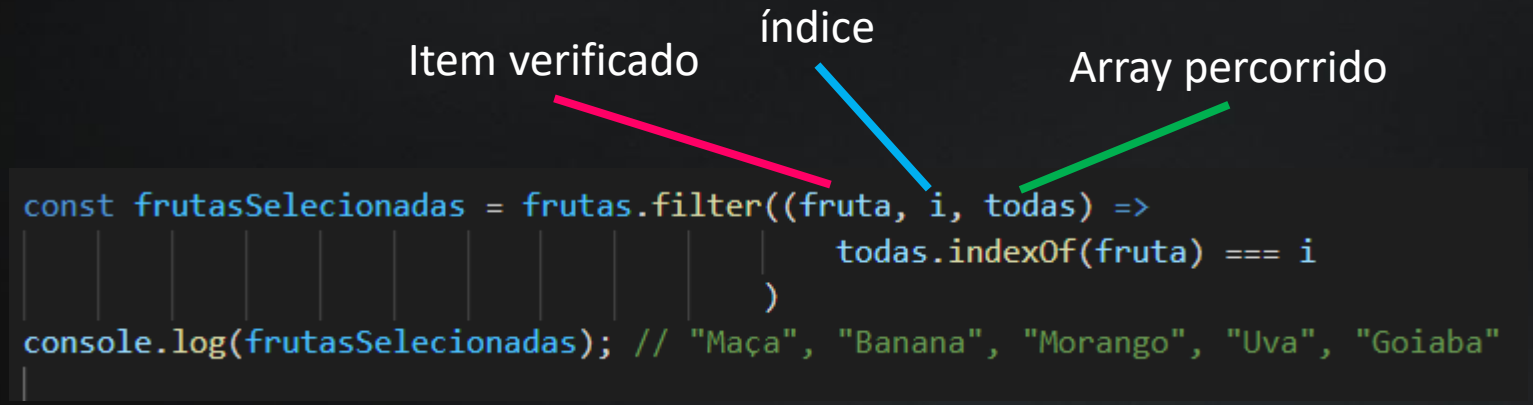
Se precisarmos criar um novo array a partir de um primeiro, mas somente com valores específicos podemos usar o método filter, que percorre o array fazendo a validação contida na função de callback.

```
const notas = [1,2,3,4,5,6,7,8,9,10]
console.log(notas); // 1, 2, 3, 4, 5, 6, 7, 8, 9, 10
const notasAprov = notas.filter(item => item >= 6)
console.log(notasAprov); // 6, 7, 8, 9, 10

const pares = notas.filter(item => item %2 == 0)
console.log(pares); // 2, 4, 6, 8, 10
```

Manipulação de Array – método FILTER( )

Ainda conhecendo o filter, ele pode receber 3 valores como parâmetro: o item do array, o índice do array e o próprio array.



Aqui estamos percorrendo o array, e pegando apenas a primeira fruta de cada tipo.

Manipulação de Array – método REDUCE( )

O método reduce executa uma função de call-back para cada interação da passagem pelo array retornando um único valor. Na função de callback ele espera receber até 4 parâmetros: valor Anterior ou acumulador, valor atual, índice e o array percorrido. No exemplo abaixo só estaremos usando os dois primeiros. Devemos também passar o valor inicial.

```
const vendedores = [
    {'nome':'Janaina','vendas':5},
    {'nome':'Vitória','vendas':7},
    {'nome':'Marcelo','vendas':3},
    {'nome':'Henrique','vendas':9},
]

const totalVendas = vendedores.reduce((valorAnt,vendAtual)=> valorAnt + vendAtual.vendas,0)
console.log(totalVendas); // 24
```

Valor acumulado

Elemento atual

Valor inicial

Manipulação de Array – método EVERY( )

45697056

O método every testa se todos os elementos do array passam pelo teste implementado pela função fornecida, retornando assim um valor boleano.

```
const filaBrinquedo = [
    {'nome':'Sara','altura':1.50},
    {'nome':'Luciana','altura':1.70},
    {'nome':'Kleber','altura':1.65},
    {'nome':'Anderson','altura':1.80}
]

const todaFilaPode = filaBrinquedo.every(pessoas => pessoas.altura >= 1.60)
console.log(todaFilaPode == true ? "Vamos lá" : "Nem todos podem");
```

Manipulação de Array – método SOME( )

45697056

O método some testa se ao menos um dos elementos do array passa no teste lógico, retornando um boleano.

```
const passeio = [
    {'nome':'Sara','idade':17},
    {'nome':'Luciana','idade':16},
    {'nome':'Kleber','idade':15},
    {'nome':'Anderson','idade':21}
]

const verifIdade = passeio.some(pessoa => pessoa.idade >= 18)
console.log(verifIdade);
```

Manipulação de Array – método FIND( )

45697056

O método find retorna o primeiro elemento do array que atender ao teste imposto pela função callback.

```
const candidatos = [
    {'nome':'Reinaldo','nota':65},
    {'nome':'Rita','nota':67},
    {'nome':'Sérgio','nota':78},
    {'nome':'Valter','nota':80}
]

const selecionado = candidatos.find( cand => cand.nota >= 70)
console.log(`${selecionado.nome} teve a nota ${selecionado.nota}!`);
```

Manipulação de Array – método INCLUDES( )

O método includes verifica se um array contém ou não um determinado elemento e retorna um boleano.

```
const selecionado = candidatos.find( cand => cand.nota >= 70)
console.log(`${selecionado.nome} teve a nota ${selecionado.nota}!`);

const convidados = ['prof Allen', 'Lucas', 'Gilberto','prof Luís','prof Alexandre']

const profConvid = convidados.filter( conv => conv.includes('prof'))
console.log(profConvid); // "prof Allen", "prof Luís", "prof Alexandre"
```

Repare que neste exemplo usamos o includes em uma string, que é um array de caracteres.

DUVIDAS