FIAP

GRADUAÇÃO

45697056

TDS

Responsive Web Development

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

Prof. Luís Carlos lsilva@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( ), REVERSE() e POP( )

45697056

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

```
console.log(aluno.sort()); // Barbara, Carlos, João, Lucas, Maria
console.log(aluno.sort().reverse()); // Maria, Lucas, João, Carlos, Barbara
```

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"
```

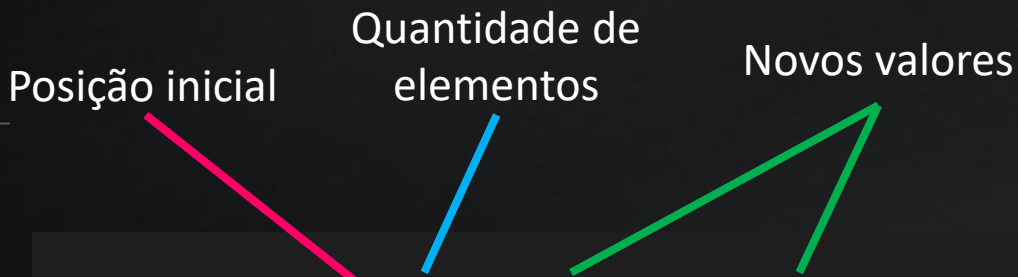
E para remover o elemento da primeira posição usamos o método shift( ).

```
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( ).

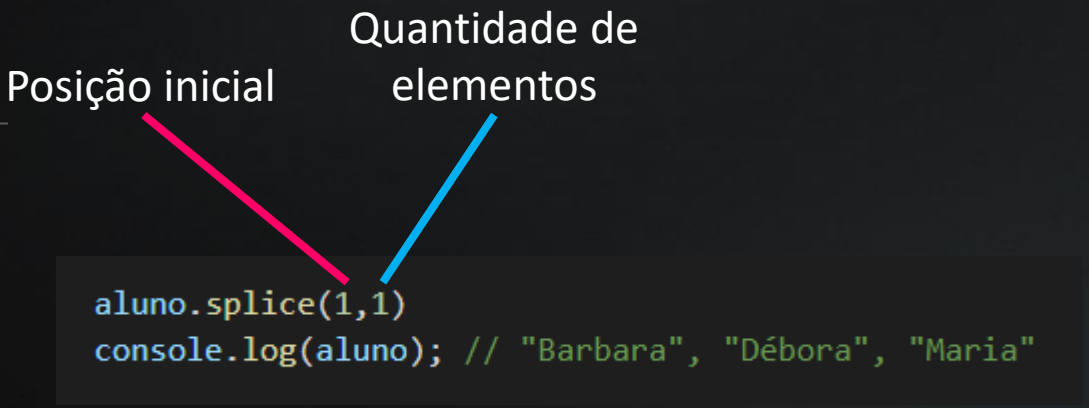



```
aluno.splice(1,2,"Cláudio","Débora")
console.log(aluno); // "Barbara", "Cláudio", "Débora", "Maria"
```

Manipulação de Array SPLICE( )

45697056

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



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

Exercício

45697056

Para exercitarmos nosso conhecimento vamos trabalhar com uma página de tarefas pessoais.

- Crie um projeto HTML/CSS/JS de nome exercicio-01-arrays
- Crie uma página HTML de nome index e adicione os seguintes elementos:
  - Título da página com identificação do aluno.
  - Formulário com um input e um botão para adicionar novas tarefas.
  - Uma lista não ordenada para receber as tarefas inseridas no array, através do formulário(esta lista deve ser preenchida dinamicamente por um loop for).
- É obrigatório a utilização dos métodos de manipulação de array necessários para incluir novas tarefas e excluir as tarefas desejadas. Ex: PUSH, POP, SHIFT, UNSHIFT e SPLICE.

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( )

45697056

Ainda conhecendo o filter podemos, por exemplo, fazer uma validação de login de uma forma bem descomplicada, comparando os valores do objeto.

```
let alunos = [
    {nome:'Luis', senha: 123},
    {nome:'Alexandre', senha: 456}
]

const logado = alunos.filter(aluno => aluno.nome === 'Luis' && aluno.senha === 123)

console.log(...logado)
```

Aqui estamos percorrendo o array e retornando apenas o objeto correto caso a condição seja satisfeita.

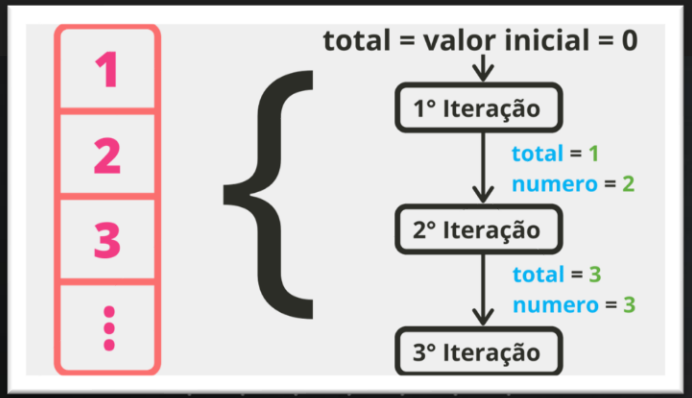
Manipulação de Array – método REDUCE( )

O método reduce executa uma função de call-back para cada interação da passagem pelo array retornando um único valor. Devemos inserir um argumento que será o responsável por acumular os valores e um segundo que representará o valor atual..

```
let numbers = [4, 5, 8, 6, 7]

const soma = numbers.reduce((acumulado, atual)=> acumulado + atual)

console.log(soma);
```



Manipulação de Array – método REDUCE( )

Neste exemplo vamos usar uma função para facilitar a soma dos atributos dos objetos, assim inserimos a função diretamente no reduce e deixamos o valor inicial como 0.

```
const vendedores = [
    {nome:'Luis',vendas: 150},
    {nome:'Alexandre',vendas: 250},
    {nome:'Andréia',vendas: 120}
]

function somaObj(obj1, obj2) {
    return {vendas: obj1.vendas + obj2.vendas}
}
const total = vendedores.reduce(somaObj, {vendas: 0})
console.log(total.vendas);
```

Valor inicial

Elemento atual

Valor acumulado

A função log( ) recebe outra função como parâmetro e a chama dentro de si. Isso é válido no JavaScript e nós chamamos de "callback". Então, a função que é passada para outra função como um parâmetro é uma função de callback.

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( )

45697056

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

```
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.

Exercício

45697056

1 – Crie um array contendo 10 valores de salário e utilizando o método map() atribua um aumento de 15% para salários até 2000 e um aumento de 10% para salários acima de 2000.

2 – Utilizando o array de resultado do exercício anterior, crie um novo array, usando o método filter(), contendo somente os salários superiores a 2500.

3 – Utilizando o array de resultado do exercício anterior, usando o método reduce(), some os valores.

OBJETOS

45697056

Em JavaScript, objetos são estruturas de dados que permitem armazenar informações em pares de chave-valor. Eles são uma coleção de propriedades, onde cada propriedade tem um nome (chave) e um valor associado. Os valores podem ser de qualquer tipo de dado, como números, strings, booleanos, funções e até mesmo outros objetos.

```
//Exemplo criação de um objeto
    const pessoa = {
        nome: 'Luis',
        idade: 30,
        sexo: 'M',
        altura: 1.80,
        peso: 80,
        andar: function(){
            console.log('Andando...')
        }
    }
```

PROPRIEDADES

45697056

As propriedades dos objetos em JavaScript são pares de chave-valor que definem as características e valores associados a estes objetos. Essas propriedades permitem armazenar e acessar informações dentro do objeto. A chave é sempre uma string (ou símbolo) que identifica exclusivamente a propriedade, e o valor pode ser de qualquer tipo de dado válido em JavaScript, como números, strings, booleanos, funções ou até mesmo outros objetos. As propriedades podem ser adicionadas, modificadas ou removidas dinamicamente em tempo de execução.

PROPRIEDADES

45697056

```
//Acessando o atributo nome
console.log(pessoa.nome);

//Alterando o valor de um atributo
pessoa.nome = 'Alexandre';
console.log(pessoa.nome);

//Adicionando um atributo dirteto ao objeto
pessoa.cabelo = 'Castanho';
console.log(pessoa.cabelo);

//Adicionando um atributo ao objeto utilizando colchetes
pessoa['olhos'] = 'Castanhos';
console.log(pessoa['olhos']);

//Acessando um atributo utilizando colchetes
console.log(pessoa['nome']);

//Acessando um atributo utilizando colchetes e uma variável
let atributo = 'idade';
console.log(pessoa[atributo]);
```

PROPRIEDADES

45697056

```
//Deletando um atributo do objeto
delete pessoa.idade;
console.log(pessoa);

//Deletando um atributo utilizando colchetes
delete pessoa['sexo'];
console.log(pessoa);

//Deletando um atributo utilizando colchetes e uma variável
let atributo2 = 'altura';
delete pessoa[atributo2];
console.log(pessoa);
```

MÉTODOS

45697056

Métodos em objetos literais em JavaScript são funções que estão associadas às propriedades de um objeto. Eles permitem que os objetos realizem ações ou executem operações específicas relacionadas aos seus dados. Os métodos são definidos dentro do objeto como valores de propriedades, onde a chave é o nome do método e o valor é a própria função. Essas funções podem acessar e manipular os dados do objeto usando a palavra-chave "this", que se refere ao próprio objeto em que o método está sendo chamado. Os métodos podem ser chamados diretamente através do objeto e podem ter acesso a outras propriedades e métodos do mesmo objeto.

MÉTODOS

45697056

```
//Adicionando Metodo ao objeto
    pessoa.falar = function(){
        console.log('Falando...')
    }
    pessoa.falar();

//Executando a função andar
    pessoa.andar();

//Adicionando Metodo ao objeto utilizando colchetes
    pessoa['correr'] = function(){
    console.log('Correndo...')
    }

    pessoa.correr();

//Adicionando Metodo ao objeto utilizando colchetes e uma variável
    let metodo = 'pular';
    pessoa[metodo] = function(){
        console.log('Pulando...')
    }
    pessoa.pular();
```

MÉTODOS

```
//Acessando um metodo utilizando colchetes
    console.log(pessoa['falar']);

//Acessando um metodo utilizando colchetes e uma variável
    let metodo2 = 'correr';
    console.log(pessoa[metodo2]);

//Deletando um metodo do objeto
    delete pessoa.falar;
    console.log(pessoa);

//Deletando um metodo utilizando colchetes
    delete pessoa['correr'];
    console.log(pessoa);

//Deletando um metodo utilizando colchetes e uma variável
    let metodo3 = 'pular';
    delete pessoa[metodo3];
    console.log(pessoa);
```

MÉTODOS(FUNÇÕES)

45697056

```
//Exemplo de SPREAD
    const pessoa2 = {...pessoa, nome: 'Andréia', idade: 25, sexo: 'F'}
    console.log(pessoa2);

//Exemplo de REST
    const {nome, idade, ...resto} = pessoa2;
    console.log(nome, idade, resto);

//Exemplo de DESTRUCTURING
    const {nome: nome2, idade: idade2, ...resto2} = pessoa2;
    console.log(nome2, idade2, resto2);
```

Exercícios

45697056

Agora que já sabemos manipular objetos, vamos melhorar a nossa lista de tarefas. Nossa nova lista será uma tabela e deve ser feita utilizando objetos com os seguintes atributos:

- Descrição,
- Autor,
- Departamento,
- Importância.

Nossa lista de tarefas deverá ter os seguintes controles:

- Inclusão de nova tarefa;
- Exclusão da tarefa concluída;
- Opção para adicionar o campo valor nos objetos das tarefas que serão pagas à parte.
- Opção para adicionar o campo duração nos objetos das tarefas que serão realizadas à parte.
- Opção para criação de uma lista das tarefas por ordem de importância contendo apenas a descrição.

DUVIDAS