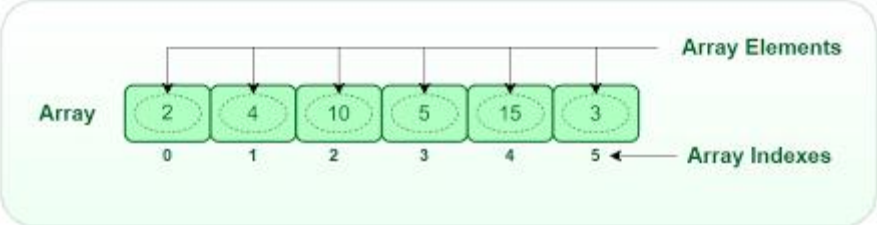
O que é um array?

Arrays são estruturas que servem para guardar dados, e organizá-los.

Seu objetivo é ser um espaço fixo na memória do computador para armazenar varios elementos.

Esses elementos podem ser acessados por um tipo de indicação, que chamamos de índice.

Arrays são variáveis compostas, unidimensionais e indexadas.



Declarando um array

```
let nomes = ['joao', 'jose', 'juca', 'anna']
```

Imprimindo seus dados

```
console.log(nomes[0]);
console.log(nomes[1]);
console.log(nomes[2]);
```

Exemplo: inserindo os dados em um array e exibindo seu valores no browser.

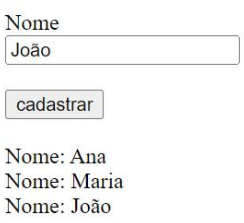
HTML

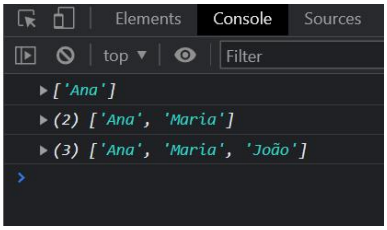
```
<label>Nome</label> <br>
<input type="text" id="name"> <br><br>
<button id="btn">cadastrar</button><br><br>
<div id="app"></div>
```

JS

```
let nome = document.getElementById('name')
let app = document.getElementById('app')
let btn = document.getElementById('btn')

// Declarando um array
let nomes = []
```

```
function cadastrar(){

   // push - insere um valor no final do array
   nomes.push(nome.value)

   imprimir()  // chamando a função imprimir
}

function imprimir(){
   console.log(nomes);
   app.innerHTML = ''

   // percorrendo um array
   for(let i = 0; i < nomes.length; i++){
      app.innerHTML += `Nome: ${nomes[i]} <br>`
   }
}
btn.addEventListener('click', cadastrar)
```





Abstração

Abstração é o processo de identificar as qualidades ou propriedades importantes do problema que está sendo modelado. Através de um modelo abstrato, pode-se concentrar nas características relevantes e ignorar as irrelevantes.

Objeto

Um objeto é um elemento computacional que representa alguma entidade (abstrata ou concreta) do problema sob análise. No paradigma de orientação a objetos, tudo pode ser potencialmente representado como um objeto.

Notação de um objeto literal em JavaScript

Um objeto literal é composto por um par de chaves " { } ", que envolve uma ou mais propriedades. Cada propriedade segue o formato " nome: valor " e devem ser separadas por vírgula.
No exemplo a seguir temos a abstração **aluno** escrito com a notação do JavaScript.

```
let aluno = {
```

```
    nome: 'Cristiano Ronaldo',
    idade: 22,
    sexo: 'M'
}

console.log("Nome: " + aluno.nome);
console.log("Idade: " + aluno.idade);
console.log("Sexo: " + aluno.sexo);
```

Array de Objetos

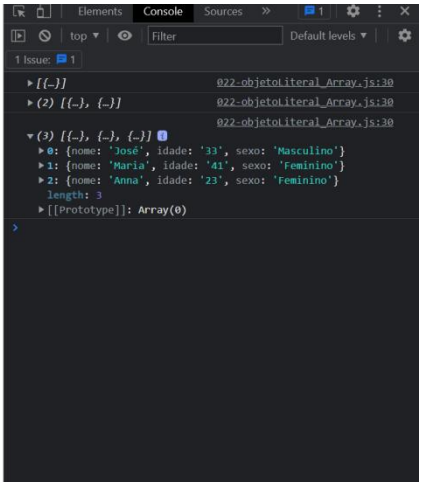
HTML

```
<label>Nome:</label>
<input type="text" id="nome" class="form-control">

<label>Idade:</label>
<input type="number" id="idade" class="form-control">

<label>Sexo:</label>
 <select class="form-control" id="sexo">
    <option value="">Selecione...</option>
    <option value="Masculino">Masculino</option>
    <option value="Feminino">Feminino</option>
 </select>

 <button id="btn" class="btn btn-primary">Cadastrar</button>


 <table class="table">
    <thead>
     <tr>
      <th scope="col">NOME</th>
      <th scope="col">IDADE</th>
      <th scope="col">SEXO</th>
     </tr>
    </thead>
    <tbody id="table">
     <!--  Aqui será inserido os dados do array -->
    </tbody>
   </table>
  </div>
</div>
```

JavaScript

```
let btn = document.getElementById('btn')
let tabela = document.getElementById('table')

// cria um array vazio
let arr = []

function cadastrar() {
    // pega os dados do formulário
    let nome = document.getElementById('nome').value
    let idade = document.getElementById('idade').value
    let sexo = document.getElementById('sexo').value

    // coloca os dados em um objeto literal
    let obj = {
        nome: nome,
        idade: idade,
        sexo: sexo
    }

    // armazena o objeto em um array
    arr.push(obj)

    // chama a função criarTabela
    criarTabela()
}


function criarTabela() {

    console.log(arr);

    tabela.innerHTML = ''

    for (let i = 0; i < arr.length; i++) {
        tabela.innerHTML += `
        <tr>
            <td>${arr[i].nome}</td>
            <td>${arr[i].idade}</td>
            <td>${arr[i].sexo}</td>
        </tr>
        `
    }
    clear()
}

function clear(){
    document.getElementById('nome').value = ''
    document.getElementById('idade').value = ''
    document.getElementById('sexo').value = ''
```

```
    document.getElementById('nome').focus()
}
btn.addEventListener('click', cadastrar)
```



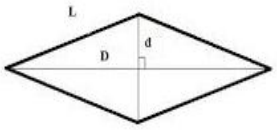
Exercícios

Exercício 82:

Crie uma aplicação para calcular e imprimir a área de um losango.

Se os valores de entrada forem negativos ou iguais a zero a aplicação deverá imprimir uma mensagem de erro - “Valores inválidos”.

Área = (D * d) / 2



Exercício 83:

Conhecendo a diagonal maior e a diagonal menor de um losango regular, crie uma aplicação para calcular o valor de seu lado.

Atenção: Os valores não podem ser negativos ou iguais a zero.

Exercício 84:

Em época de pouco dinheiro, os comerciantes estão procurando aumentar suas vendas oferecendo desconto.

Faça uma aplicação que possa entrar com o valor de um produto e imprimir o novo valor tendo em vista que o desconto foi de 9%.

O valor do produto não pode ser menor ou igual a zero.

Exercício 85:

Construa uma aplicação que receba dois valores numéricos e efetue sua adição.

Caso o resultado da adição seja maior que 10, exibir os números digitados, o valor da adição e a raiz cúbica da adição.

Caso contrário exibir somente os valores digitados e o valor da adição.

Exercício 86

Elabore uma aplicação que faça leitura de vários números inteiros e para cada vez que um número for inserido a aplicação retorne o maior e o menor número lido.

A aplicação também deve exibir todos os números inseridos.

Exercício 87:

Crie uma aplicação para imprimir a soma de todos os números de 0 à 100.

Exercício 88:

Crie uma aplicação que receba um número obrigatoriamente maior que zero e imprima todos os números de zero até o número digitado.

Exercício 89:

Crie uma aplicação que receba um número obrigatoriamente menor que dez e imprima todos os números de vinte até o número digitado.

Exercício 90:

Faça uma aplicação que receba vários números, calcule e mostre:

- Todos os números digitados
- A soma dos números digitados
- A quantidade de números digitados
- A média dos números digitados
- A média dos números pares

Exercício 91:

Faça uma aplicação que some todos os números abaixo de 1000 (até o número zero) que são múltiplos de 3 e de 5 ao mesmo tempo.

Exercício 92

Faça uma aplicação para imprimir em ordem decrescente todos os números múltiplos de 3 e de 5 compreendidos entre 30 e 300.

Exercício 93

Crie uma aplicação para ler dois números e imprimir todos os números pares e múltiplos de 7 (ao mesmo tempo) compreendidos entre os números digitados pelo usuário.

Atenção: o usuário pode digitar o primeiro número menor que o segundo e vice-versa.

Exercício 094:

Desenvolva uma aplicação capaz de receber e armazenar os dados de diversos alunos para consulta posterior. Os dados de cada aluno incluem: nome, nota 01, nota 02 e média aritmética.

Após cada inserção de dados, a aplicação deverá exibir todos os registros cadastrados até o momento, além de apresentar a quantidade de alunos aprovados e reprovados. Para ser considerado aprovado, o aluno deve obter uma média igual ou superior a 7,00.

Exercício 095:

Desenvolva uma aplicação para gerenciar uma biblioteca, onde seja possível cadastrar livros com os seguintes dados: título, autor, ano de publicação e número de páginas.

Após cada inserção de livro, a aplicação deve exibir todos os livros cadastrados até o momento e a quantidade total de livros na biblioteca.

Exercício 096:

Crie um programa para controle de estoque de uma loja, onde seja possível cadastrar produtos com as informações: nome, preço, quantidade em estoque e categoria.

Após cada inserção de produto, o programa deve mostrar todos os produtos cadastrados até o momento e a quantidade total de itens em estoque.

Exercício 097:

Elabore uma aplicação para registrar as despesas mensais de um usuário, com os seguintes dados: descrição da despesa, valor e categoria (alimentação, transporte, moradia).

Após cada registro de despesa, o programa deve exibir todas as despesas cadastradas até o momento e o total gasto no mês.

Exercício 098:

Crie um sistema para acompanhar o desempenho de uma equipe esportiva, onde seja possível cadastrar os jogadores com as informações: nome, posição, número da camiseta e idade.

Após cada inserção de jogador, o sistema deve mostrar todos os jogadores cadastrados até o momento, a média de idade da equipe, a quantidade de jogadores com menos de 20 anos e a quantidade de jogadores com idade maior ou igual a 20 anos.

Exercício 099:

Desenvolva um programa para gerenciar as tarefas de um projeto, onde seja possível cadastrar as atividades com os seguintes dados: descrição da tarefa, responsável, prazo de entrega e status (pendente, em andamento, concluída).

Após cada inserção de tarefa, o programa deve exibir todas as atividades cadastradas até o momento e a quantidade de tarefas pendentes.

Exercício 100:

Crie uma aplicação que receba dois números e imprima a soma dos valores pares compreendidos entre estes números.

Exercício 101:

Crie uma aplicação que receba um número qualquer.

Se o número for positivo, imprimir o número digitado e sua raiz quadrada.

Se o número for negativo, imprimir o número digitado e seu valor elevado ao quadrado.

Exercício 102:

(2.00 pt) Crie uma aplicação web utilizando HTML, CSS e JavaScript que simule o adastro de pessoas.

A aplicação deve conter um formulário com os seguintes campos: Nome e Idade

Ao clicar no botão "Cadastrar", os seguintes passos devem ocorrer:

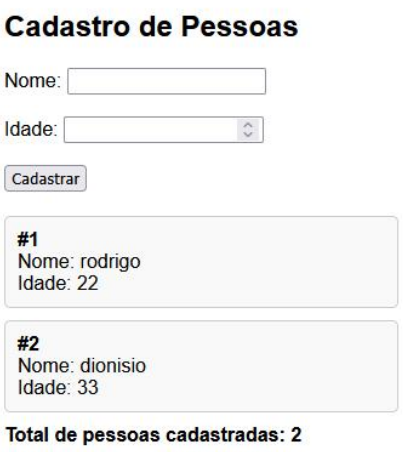
- Os dados preenchidos no formulário devem ser armazenados em um objeto.

- Este objeto deve ser adicionado a um array.

A cada nova interação (ou seja, a cada clique no botão de cadastro), um novo objeto deve ser criado e adicionado ao array (exibir o array no console.log).

(6.00 pt) Após cada cadastro a lista completa de pessoas cadastradas deve ser exibida de forma visual e organizada (por exemplo, como "cartões" ou blocos com as informações).

(2.00 pt) A aplicação também deve exibir um resumo com a quantidade total de pessoas cadastradas até o momento.



Exercício 103:

(1.00 pt) Crie uma aplicação web utilizando HTML, CSS e JavaScript que simule o cadastro de pessoas.

A aplicação deve conter um formulário com os seguintes campos: Nome, Idade, Email e Estado (somente com as opções SP, MG e RJ).

(2.00 pt) Ao clicar no botão "Cadastrar", os seguintes passos devem ocorrer:

Os dados preenchidos no formulário devem ser armazenados em um objeto com as propriedades correspondentes.

Este objeto deve ser adicionado a um array.

A cada nova interação (ou seja, a cada clique no botão de cadastro), um novo objeto deve ser criado e adicionado ao array (exibir o array no console.log).

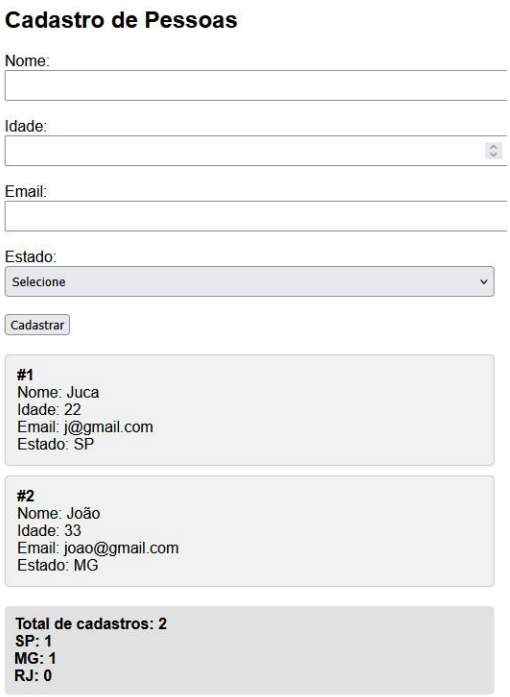
(4.00 pt) Após cada cadastro, a lista completa de pessoas cadastradas deve ser exibida de forma visual e organizada (por exemplo, como "cartões" ou blocos com as informações de nome, idade, email e estado).

(3.00 pt) A aplicação também deve exibir um resumo com:

A quantidade total de pessoas cadastradas até o momento.

A quantidade de pessoas por estado, com a seguinte estrutura:

Total de cadastros: 5
SP: 2
MG: 1
RJ: 2



Exercício 104:

(1.00 pt) Crie uma aplicação web utilizando HTML, CSS e JavaScript que simule o cadastro de carros.

A aplicação deve conter um formulário com os seguintes campos:
Marca, Modelo, Ano e Cor (com as opções: Vermelho, Preto e Branco).

(2.00 pt) Ao clicar no botão "Cadastrar", os seguintes passos devem ocorrer:

Os dados preenchidos no formulário devem ser armazenados em um objeto com as propriedades correspondentes.

Este objeto deve ser adicionado a um array.

A cada nova interação (ou seja, a cada clique no botão de cadastro), um novo objeto deve ser criado e adicionado ao array (exibir o array no console.log).

(4.00 pt) Após cada cadastro, a lista completa de carros cadastrados deve ser exibida de forma visual e organizada.

(3.00 pt) A aplicação também deve exibir um resumo com:

A quantidade total de carros cadastrados até o momento.

A quantidade de carros por cor, com a seguinte estrutura:

Total de cadastros: 4
Vermelho: 2 Preto: 1 Branco: 1

Exercício 105:

(1.00 pt) Crie uma aplicação web utilizando HTML, CSS e JavaScript que simule o cadastro de alunos.

A aplicação deve conter um formulário com os seguintes campos:
Nome, Idade, Curso (somente: Administração, Informática, Engenharia) e Turno (com as opções: Manhã, Tarde, Noite).

(2.00 pt) Ao clicar no botão "Cadastrar", os seguintes passos devem ocorrer:

Os dados preenchidos no formulário devem ser armazenados em um objeto com as propriedades correspondentes.

Este objeto deve ser adicionado a um array.

A cada nova interação (ou seja, a cada clique no botão de cadastro), um novo objeto deve ser criado e adicionado ao array (exibir o array no console.log).

(3.00 pt) Após cada cadastro, a lista completa de alunos cadastrados deve ser exibida de forma visual e organizada (por exemplo, como "cartões" com as informações de nome, idade, curso e turno).

A aplicação também deve exibir um resumo com:

(1.00 pt) A quantidade total de alunos cadastrados.

(1.50 pt) A quantidade de alunos por turno (manhã, tarde, noite).

(1.50 pt) A quantidade de alunos por curso.

Total de cadastros: 5
Por Turno:
Manhã: 2 Tarde: 1 Noite: 2
Por Curso:
Informática: 3 Administração: 1 Engenharia: 1

Cadastro de Alunos

Nome:

Idade:

Curso:
Selecione

Turno:
Selecione

Cadastrar

#1
Nome: João
Idade: 33
Curso: Informática
Turno: Manhã

#2
Nome: Anna
Idade: 33
Curso: Administração
Turno: Noite

Total de cadastros: 2

Por Turno:
Manhã: 1
Tarde: 0
Noite: 1

Por Curso:
Informática: 1
Administração: 1