

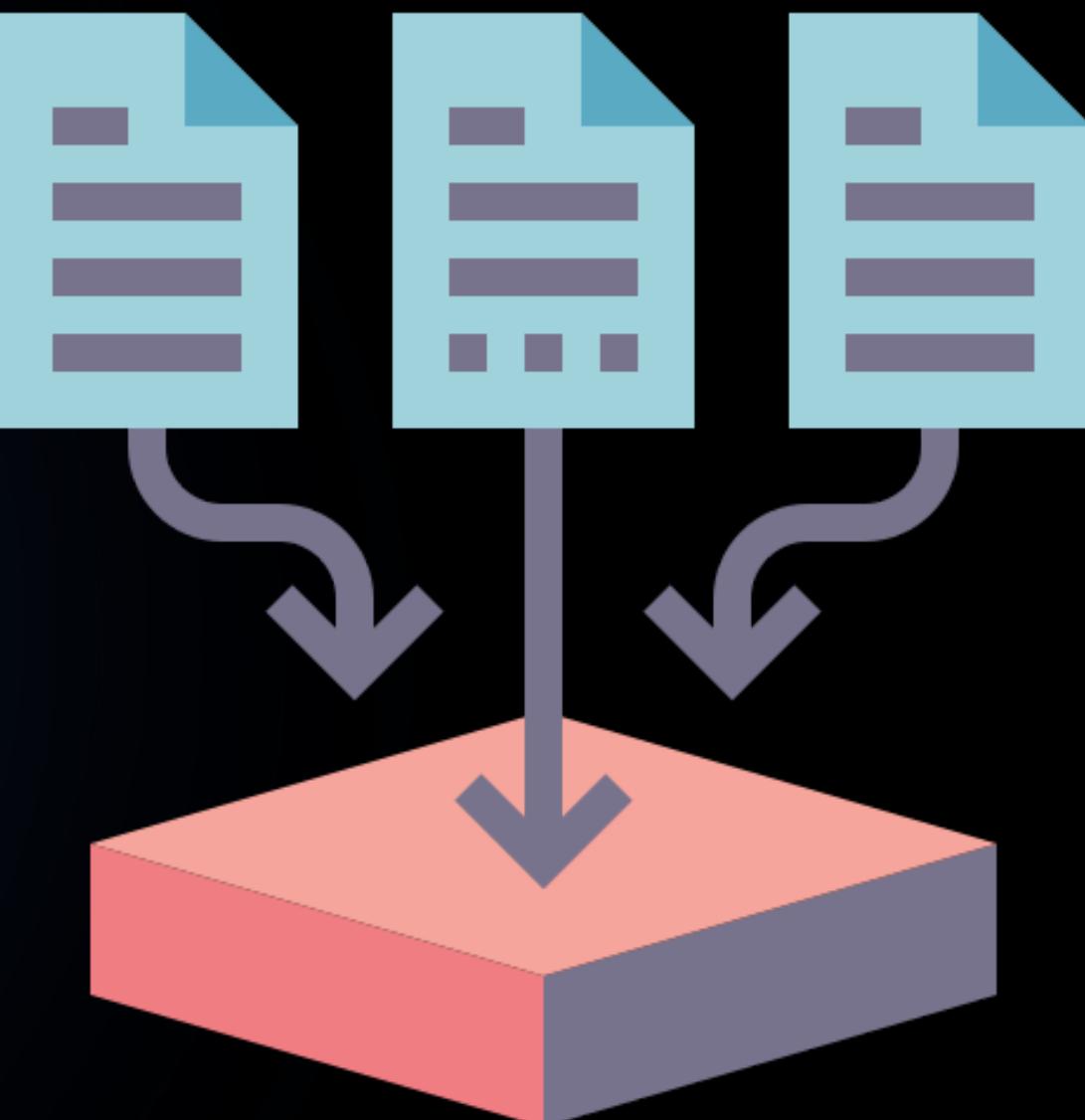
Curso de

# JavaScript

# Persistência de Dados

# O que é persistência de dados?

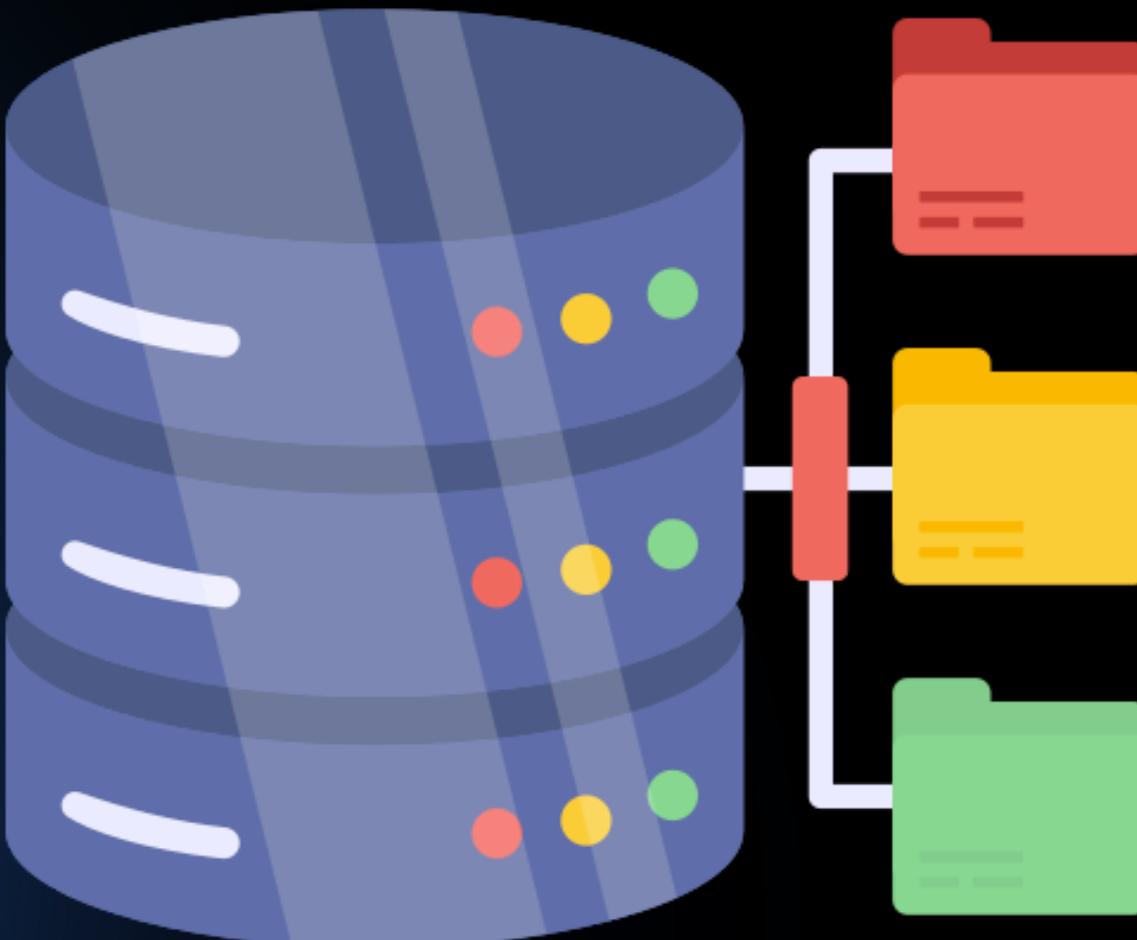
A persistência de dados consiste em salvar e recuperar informações necessárias para operações, por exemplo, o tema de preferência do usuário. Estas informações ficam disponíveis mesmo após o fechamento do navegador ou aplicativo.



# Tipos de persistencia de dados

A persistencia de dados pode ser feita de 3 formas distintas, cada uma com um escopo diferente:

Usando **SessionStorage**;  
Usando **LocalStorage**;  
Usando **IndexDB**.





## sessionStorage

O **sessionStorage** é uma API do próprio navegador que permite gravar dados em um esquema de par **chave:valor**, os dados ficam salvos nela até que a sessão seja encerrada (fechar a aba ou a janela do navegador).

# sessionStorage

```
● ● ●  
1 <!DOCTYPE html>  
2 <html lang="en">  
3 <head>  
4   <meta charset="UTF-8">  
5   <meta name="viewport" content="width=device-width, initial-scale=1.0">  
6   <title>Aula 07</title>  
7 </head>  
8 <body>  
9   <input type="text" id="campoNome" placeholder="Digite seu nome">  
10  <button onclick="salvar()">Salvar</button>  
11  <button onclick="mostrar()">Mostrar</button>  
12  <h1 id="nome"></h1>  
13 </body>  
14 <script src="script.js"></script>  
15 </html>
```

```
● ● ●  
1 function salvar() {  
2   let valor = document.getElementById("campoNome").value;  
3   sessionStorage.setItem("nome", valor);  
4 }  
5  
6 const nome = document.getElementById("nome")  
7  
8 function mostrar() {  
9   nome.innerText = sessionStorage.getItem("nome")  
10 }
```



## Local Storage

O local storage é uma forma de armazenamento de dados persistente, ou seja, os dados não serão deletados se o navegador for fechado, que permite que dados sejam salvos diretamente no navegador.

# localStorage

```
1 <!DOCTYPE html>
2 <html lang="en">
3 <head>
4   <meta charset="UTF-8">
5   <meta name="viewport" content="width=device-width, initial-scale=1.0">
6   <title>Document</title>
7 </head>
8 <body>
9   <input type="text" id="campoNome" placeholder="Digite seu nome">
10  <button onclick="salvar()">Salvar</button>
11  <button onclick="mostrar()">Mostrar</button>
12  <h1 id="nome"></h1>
13 </body>
14 <script src="script.js"></script>
15 </html>
```

```
1 function salvar() {
2   let valor = document.getElementById('campoNome').value;
3   localStorage.setItem('nome', valor);
4 }
5
6 function mostrar() {
7   const nome = document.getElementById('nome');
8   nome.innerText = localStorage.getItem('nome')
9 }
```

## IndexedDB

O indexedDB é uma API do navegador que prove um banco de dados orientado a objetos no proprio navegador.

Isso permite guardar um volume muito maior de dados em comaparação com as alternativas anteriores, mas ele ainda tem limite de espaço.



# Exemplo de uso de IndexedDB



```
1 let request = indexedDB.open("MeuBanco", 1);
2
3 request.onupgradeneeded = function(event) {
4     let db = event.target.result;
5     db.createObjectStore("usuarios", { keyPath: "id" });
6 }
```



```
1 request.onsuccess = function(event) {
2     let db = event.target.result;
3
4     // Inserindo dado
5     let tx = db.transaction("usuarios", "readwrite");
6     let store = tx.objectStore("usuarios");
7     store.add({ id: 1, nome: "Ana", idade: 25 });
8
9     tx.oncomplete = () => console.log("Dados salvos com sucesso");
10 }
```

# Resolução --- Exercícios

# **Exercício**

---

**Exercício já disponível na pasta “Aula 7” na branch master**

# chamada

---

