

# MATERIAL EXTRA

Armazenamento Web

## ARMAZENAMENTO WEB

O armazenamento web, que inclui **LocalStorage** e **Session Storage**, é uma funcionalidade disponibilizada pelo navegador para armazenar dados de forma simples e eficiente no lado do cliente, sem necessidade de servidores. Esses dois tipos de armazenamento fazem parte da API Web Storage, introduzida para superar limitações de métodos antigos, como cookies, oferecendo mais espaço, desempenho melhor e uma interface mais acessível para manipulação de dados.

### LocalStorage

É um armazenamento persistente que mantém os dados mesmo após o fechamento do navegador ou do computador. Isso o torna ideal para guardar informações que precisam estar disponíveis para o usuário durante várias sessões, como preferências de interface ou dados de configurações. Os dados armazenados no Local Storage são acessíveis em todas as abas e janelas do mesmo domínio, até que sejam explicitamente removidos, seja pelo desenvolvedor ou pelo próprio usuário. Ele suporta operações simples para adicionar, recuperar, remover ou limpar dados, sempre no formato chave–valor.



**LocalStorage** é uma propriedade do objeto **window** e pode ser acessado da seguinte forma: **window.localStorage**. Na maioria dos casos e principalmente agora no início vamos adotar apenas **localStorage**.

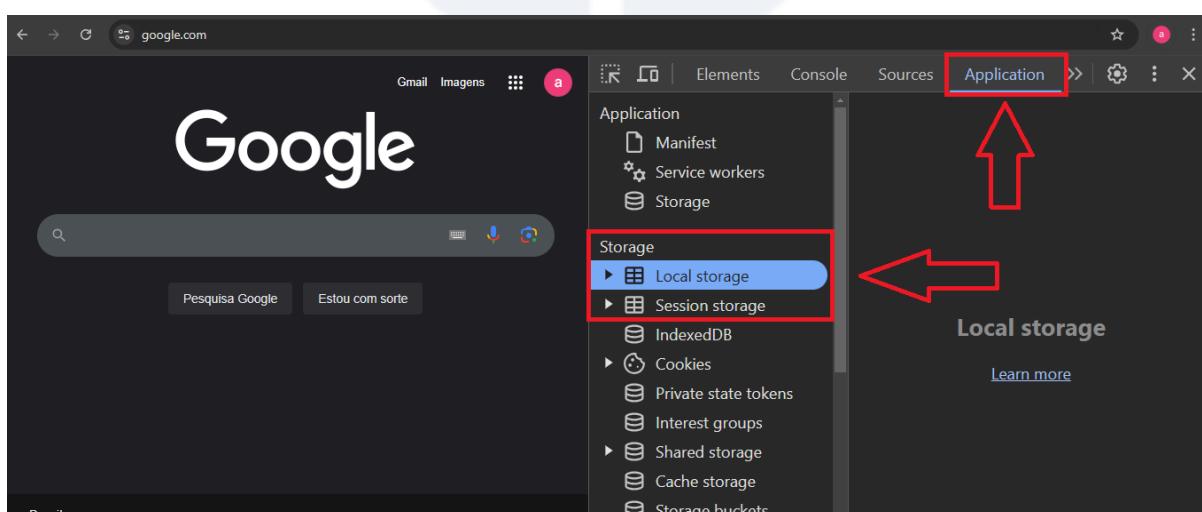
Os métodos que vamos explorar com relação a propriedade localStorage são listados abaixo:

- `localStorage.setItem()` para **criar** um par (ou item) de chave: valor;
- `localStorage.getItem()` para **recuperar** o valor de uma chave;
- `localStorage.removeItem()` para **remover** um par específico;
- `localStorage.clear()` para **remover todos** os itens gravados;

### Sintaxe:

```
localStorage.setItem(key, value);
localStorage.getItem(key);
localStorage.removeItem(key);
localStorage.clear();
```

Para explorar o conteúdo da localStorage, vamos abrir as ferramentas do desenvolvedor: Dentro do Navegador Chrome, **pressione F12**, depois clique em **Application** e por fim em **LocalStorage**, conforme mostra a figura abaixo:



É importante saber que você só pode salvar dados em formato String no localStorage, e isso é um restritivo, porque limita a gravação de dados

mais complexos. Mais adiante veremos como armazenar dados estruturados no localStorage.

## Criando informações na localStorage

Para criar um par ou item na localStorage, podemos usar o método **setItem()**.

Confira abaixo alguns exemplos deste método:

```
window.localStorage.setItem("cidade", "Campo Bom");
localStorage.setItem(1, "teclado");
localStorage.setItem(2, "mouse");
localStorage.setItem(0, "monitor");
window.localStorage.setItem("a", "Quais os processadores com 4 núcleos?");
localStorage.setItem("b", "Qual velocidade máxima da rede 2.4GHz?");
```

No exemplo acima foi apresentado o método **setItem()** sendo acessado diretamente pela propriedade `localStorage` e também por meio de `window.localStorage`.

## Recuperando informações na localStorage

Para recuperar informações da localStorage basta saber qual a chave desejamos e usar o método **getItem("chave")**, confira abaixo um exemplo:

```
let cidade = localStorage.getItem("cidade");
console.log(cidade);
```

## Removendo um item

Para remover informações da localStorage basta informar a chave e usar o método **removeItem("chave")**, confira abaixo um exemplo:

```
localStorage.removeItem("cidade");
```

Depois de não usarmos mais esses dados, é uma boa prática apagar a localStorage, evitando assim acúmulo desnecessário de dados. O método **removeItem()** irá apagar a chave que você definir. Se a chave não existir, ele não fará nada.

## Removendo todos os itens da localStorage

Para remover todas informações da localStorage basta usar o método **clear()**, confira abaixo um exemplo:

```
localStorage.clear();
```

## Como criar e recuperar um objeto no localStorage

Para criar e recuperar **objetos** na localStorage podemos proceder conforme o exemplo abaixo:

```
const produto = {  
    descricao: "Notebook",  
    preco: 1234.99,  
};  
  
// Como criar um objeto no localStorage  
localStorage.setItem("produto", JSON.stringify(produto));  
  
// Como recuperar um objeto do localStorage  
const produtoLocalStorage = JSON.parse(localStorage.getItem("produto"));
```

Note no exemplo acima que estamos gerando um **JSON (JavaScript Object Notation)** com nosso objeto produto. Mesmo que a localStorage aceite apenas String, nada impede de colocarmos objetos do tipo JavaScript, ou seja, JSON sendo convertidos em Strings pelo método **JSON.stringify()**. Desta forma do comando `JSON.stringify(produto)`, faz com que seja gerada uma String da variável produto criada.

Depois para recuperar o produto vamos utilizar o método **parse()** e alimentar uma variável com seu resultado.

### Como criar e recuperar um array no localStorage

Para criar e recuperar **array** na localStorage podemos proceder conforme o exemplo abaixo:

```
const dias = ["segunda", "terca", "quarta", "quinta", "sexta"];  
  
// Como criar um array no localStorage  
localStorage.setItem("dias", JSON.stringify(dias));  
  
// Como recuperar um array do localStorage  
const diasGravados = JSON.parse(localStorage.getItem("dias"));
```

Da mesma forma que convertemos um objeto, vamos converter um array, tanto para criar, no localStorage quanto para recuperar.

## Session Storage

Projetado para dados que devem durar apenas enquanto a aba ou janela do navegador estiver aberta. Quando o usuário fecha a aba, todos os dados armazenados no Session Storage são automaticamente apagados. Isso o torna útil para situações onde as informações precisam ser descartadas rapidamente após a sessão, como dados temporários de um formulário ou um estado transitório de um aplicativo. Ele também opera no formato chave-valor e sua interface é idêntica à do Local Storage.

### Sintaxe:

```
sessionStorage.setItem(key, value);  
sessionStorage.getItem(key);  
sessionStorage.removeItem(key);  
sessionStorage.clear();
```

**Exemplo:**

```
sessionStorage.setItem("theme", "dark");
sessionStorage.getItem("theme");
sessionStorage.removeItem("theme");
sessionStorage.clear();
```

Apesar de suas semelhanças, a principal diferença entre Local Storage e Session Storage está na persistência dos dados. Enquanto o primeiro preserva informações mesmo após o encerramento do navegador, o segundo apaga tudo assim que a sessão é finalizada.

Ambos possuem algumas características em comum: oferecem até 5 MB de armazenamento por domínio na maioria dos navegadores e, por razões de segurança, só permitem acesso de scripts executados no mesmo domínio e protocolo. Além disso, nenhuma dessas formas de armazenamento deve ser usada para guardar dados sensíveis, como senhas, pois os valores não são criptografados e podem ser acessados facilmente via ferramentas de desenvolvedor do navegador.

Portanto, ao decidir entre Local Storage e Session Storage, é importante considerar o tempo de vida desejado para os dados e o comportamento esperado em termos de persistência entre sessões. Essa escolha impacta diretamente a experiência do usuário e a segurança da aplicação.

## REFERÊNCIAS

1. [Introdução ao DOM – APIs da Web | MDN](#)
2. [Window – Web APIs | MDN](#)
3. [Document – Web APIs | MDN](#)
4. [Storage – Web APIs | MDN](#)

# BONS ESTUDOS