

Guia de Arrays e LocalStorage em JavaScript

Este documento reúne os principais métodos para manipulação de **arrays** e o uso do **LocalStorage** no navegador. O objetivo é servir como material introdutório para iniciantes em JavaScript.

◆ O que é o LocalStorage?

O **LocalStorage** é um recurso dos navegadores que permite armazenar informações de forma **persistente** (os dados continuam mesmo depois de fechar o navegador).

- Ele armazena pares **chave** → **valor**.
- Todos os valores são salvos como **strings**.

Principais métodos do LocalStorage

Método	O que faz
<code>localStorage.setItem(chave, valor)</code>	Salva um valor (sempre em string)
<code>localStorage.getItem(chave)</code>	Recupera o valor de uma chave
<code>localStorage.removeItem(chave)</code>	Remove uma chave específica
<code>localStorage.clear()</code>	Apaga tudo do LocalStorage
<code>localStorage.key(n)</code>	Retorna o nome da chave pelo índice
<code>localStorage.length</code>	Número de itens armazenados

Exemplo básico

```
// Salvar
localStorage.setItem("nome", "Ana");

// Ler
let nome = localStorage.getItem("nome");
console.log(nome); // "Ana"

// Remover
localStorage.removeItem("nome");

// Apagar tudo
localStorage.clear();
```

◆ Guardando Objetos e Arrays no LocalStorage

Como o LocalStorage só aceita **strings**, usamos duas funções:

- `JSON.stringify(objeto)` → transforma objeto/array em string.
- `JSON.parse(string)` → transforma string de volta em objeto/array.

```
// Exemplo com objeto
let usuario = { nome: "Carlos", idade: 30 };
localStorage.setItem("usuario", JSON.stringify(usuario));

let recuperado = JSON.parse(localStorage.getItem("usuario"));
console.log(recuperado.nome); // "Carlos"

// Exemplo com array
let frutas = ["maçã", "banana", "uva"];
localStorage.setItem("frutas", JSON.stringify(frutas));

let lista = JSON.parse(localStorage.getItem("frutas"));
console.log(lista[1]); // "banana"
```

♦ Iterando sobre o LocalStorage

```
// Mostrar todas as chaves e valores
for (let i = 0; i < localStorage.length; i++) {
  let chave = localStorage.key(i);
  let valor = localStorage.getItem(chave);
  console.log(chave, valor);
}

// Converter tudo em objeto
let dados = {};
for (let i = 0; i < localStorage.length; i++) {
  let chave = localStorage.key(i);
  dados[chave] = localStorage.getItem(chave);
}
console.log(dados);
```

♦ O que é Callback?

Muitos métodos de arrays em JavaScript recebem uma **função como parâmetro**. Essa função é chamada de **callback**.

➡ Um **callback** é simplesmente **uma função que você passa como argumento para outra função**.

O JavaScript executa essa função em cada item do array (ou em momentos específicos).

Exemplo simples:

```
function digaOla(nome) {
  console.log("Olá " + nome);
}

// Essa função é passada como callback para forEach
let nomes = ["Ana", "Bruno", "Carla"];
nomes.forEach(digaOla);
```

Também podemos escrever callbacks **anônimos**:

```
function digaOla(nome) {  
  console.log("Olá " + nome);  
}  
  
// Essa função é passada como callback para forEach  
let nomes = ["Ana", "Bruno", "Carla"];  
nomes.forEach(digaOla);  
  
nomes.forEach(nome => console.log("Oi " + nome));
```

👉 **Resumo:** um callback é apenas uma função passada como parâmetro, que será chamada durante a execução de outro método.

◆ Métodos de Arrays

Agora que sabemos o que é um **callback**, vamos ver os principais métodos.

⚡ = indica que o método usa callback.

🔍 Busca e Checagem

⚡ **find(callback, thisArg?)**

Retorna o **primeiro elemento** que satisfaz a condição.

```
let numeros = [10, 20, 30, 40];  
let achado = numeros.find(n => n > 25);  
console.log(achado); // 30
```

⚡ **findIndex(callback, thisArg?)**

Retorna o **índice** do primeiro elemento que satisfaz a condição.

```
let indice = numeros.findIndex(n => n > 25);  
console.log(indice); // 2
```

⚡ `some(callback, thisArg?)`

Retorna `true` se **algum elemento** passa na condição.

```
console.log(numeros.some(n => n > 35)); // true
```

⚡ `every(callback, thisArg?)`

Retorna `true` se **todos os elementos** passam na condição.

```
console.log(numeros.every(n => n > 0)); // true
```

`includes(valor, início?)`

Verifica se o array contém o valor indicado.

```
console.log(numeros.includes(20)); // true
```

Iteração

⚡ `forEach(callback, thisArg?)`

Executa a função para cada item do array (não retorna novo array).

```
let frutas = ["maçã", "banana", "uva"];
frutas.forEach((fruta, i) => console.log(i, fruta));
```

⚡ `map(callback, thisArg?)`

Cria um **novo array** aplicando a função a cada elemento.

```
let maiusculas = frutas.map(f => f.toUpperCase());
console.log(maiusculas); // ["MAÇÃ", "BANANA", "UVA"]
```

Filtragem e Transformação

`filter(callback, thisArg?)`

Retorna um novo array com elementos que passam no teste.

```
let maiores = numeros.filter(n => n > 15);  
console.log(maiores); // [20, 30, 40]
```

`reduce(callback, valorInicial)`

Reduz o array a um único valor.

```
let soma = numeros.reduce((acc, n) => acc + n, 0);  
console.log(soma); // 100
```

Ordenação e Manipulação

`sort(compareFn?)`

Ordena os elementos (modifica o array original).

```
let letras = ["d", "a", "c", "b"];  
letras.sort();  
console.log(letras); // ["a", "b", "c", "d"]
```

`reverse()`

Inverte a ordem (modifica o array original).

```
letras.reverse();  
console.log(letras); // ["d", "c", "b", "a"]
```

`slice(início, fim)`

Retorna uma cópia parcial do array.

```
let sub = numeros.slice(1, 3);  
console.log(sub); // [20, 30]
```

splice(início, quantos, ...novosItens)

Adiciona ou remove elementos (modifica o array original).

```
let lista = ["a", "b", "c"];  
lista.splice(1, 1, "x", "y");  
console.log(lista); // ["a", "x", "y", "c"]
```

Concatenação e Junção

concat(...arrays)

Junta arrays em um novo array.

```
let arr1 = [1, 2], arr2 = [3, 4];  
console.log(arr1.concat(arr2)); // [1, 2, 3, 4]
```

join(separador?)

Junta os elementos em uma string.

```
console.log(frutas.join(", ")); // "maçã, banana, uva"
```

Outros

flat(nível?)

Achata arrays aninhados até o nível indicado.

```
let aninhado = [1, [2, [3, 4]]];  
console.log(aninhado.flat(2)); // [1, 2, 3, 4]
```

`at(índice)`

Retorna elemento pelo índice (aceita negativos).

```
console.log(numeros.at(-1)); // 40
```

`fill(valor, início?, fim?)`

Preenche array com o valor indicado.

```
let vazio = new Array(5).fill(0);  
console.log(vazio); // [0, 0, 0, 0, 0]
```

◆ Exemplo Prático: Lista de Tarefas com LocalStorage + Arrays

```
<input id="tarefa" placeholder="Digite uma tarefa">  
<button onclick="adicionar()">Adicionar</button>  
<ul id="lista"></ul>  
  
<script>  
  function carregar() {  
    let tarefas = JSON.parse(localStorage.getItem("tarefas")) || [];  
    let ul = document.getElementById("lista");  
    ul.innerHTML = "";  
    tarefas.forEach((tarefa, i) => {  
      let li = document.createElement("li");  
      li.textContent = tarefa;  
      ul.appendChild(li);  
    });  
  }  
  
  function adicionar() {  
    let input = document.getElementById("tarefa");  
    let tarefas = JSON.parse(localStorage.getItem("tarefas")) || [];  
    tarefas.push(input.value);  
    localStorage.setItem("tarefas", JSON.stringify(tarefas));  
    input.value = "";  
  }  
</script>
```



```
    cargar();  
  }  
  
  cargar();  
</script>
```