Capítulo 5

Vetores (Arrays)

Programação com JavaScript & TypeScript

Ambiente para praticar

Use o TypeScript Playground para testar código direto no navegador

https://www.typescriptlang.org/play/

1. Escreva código TypeScript no painel esquerdo
2. Use console.log() para ver saída na aba "Logs"
3. Clique em "Run" para executar
4. Também funciona com JavaScript puro!

O que são vetores?

Estruturas que armazenam uma lista de dados na memória

Representação de um vetor

0	Arroz
1	Feijão
2	Iogurte
3	Leite
4	Suco
5	Pão

produtos

Conceitos-chave

```
// Declarar um vetor vazio
const produtos: string[] = [];

// Declarar com valores
const produtos = ["Arroz", "Feijão"];

// Índice começa em 0!
produtos[0]  // "Arroz"
produtos[1]  // "Feijão"
```

Inclusão e exclusão de itens

`push()`

Adiciona ao final

```
arr.push("Suco")
```

`pop()`

Remove do final

```
arr.pop()
```

`unshift()`

Adiciona ao início

```
arr.unshift("Café")
```

`shift()`

Remove do início

```
arr.shift()
```

Espaço para exemplo ao vivo

Tamanho e exibição de itens

`.length`

Retorna o número de elementos do vetor

```
const arr = ["A", "B", "C"];

console.log(arr.length); // 3
```

Percorrer com for

```
for (let i = 0; i < arr.length; i++) {
  console.log(arr[i]);
}

// Ou com for...of
for (const item of arr) { ... }
```

Exibir como texto

```
arr.toString()       // "A,B,C"
arr.join(" - ")      // "A - B - C"
arr.join(", ")       // "A, B, C"
```

Localizar conteúdo

`indexOf()`

Busca do início para o fim.
Retorna o índice da 1ª ocorrência
ou -1 se não encontrar.

```
const ids = [5, 6, 8, 3, 6, 9];

ids.indexOf(6)   // 1
ids.indexOf(7)   // -1
```

`includes()`

Retorna true ou false.
Mais simples quando você só
precisa saber se existe.

```
const ids = [5, 6, 8, 3, 6, 9];

ids.includes(6)  // true
ids.includes(7)  // false
```

Espaço para exemplo ao vivo — Jogo descubra o número

Vetores de objetos

Cada elemento pode ter múltiplos atributos

```
// Definindo o tipo (TypeScript)
interface Carro {
  modelo: string;
  preco: number;
}

const carros: Carro[] = [];

// Adicionando objetos
carros.push({ modelo: "Fusca",
              preco: 6500 });

// Acessando atributos
carros[0].modelo  // Fusca
```

`interface`
Define a forma do objeto (TypeScript)

`{ }`
Chaves delimitam os atributos do objeto

`atributo:`
Cada campo seguido de : e seu valor

`.atributo`
Acesse com ponto ex: carros[0].preco

Pesquisar e filtrar dados

Extrair informações de uma lista com condições

Filtrar com for + if

```
const idades = [12, 20, 15, 17];
let achou = false;

for (const idade of idades) {
  if (idade >= 18) {
    console.log(idade);
    achou = true;
  }
}
if (!achou) console.log("Nenhum");
```

Filtrar com .filter()

```
const idades = [12, 20, 15, 17];

const maiores = idades
  .filter(i => i >= 18);

console.log(maiores);
// [20]
```

Espaço para exemplo ao vivo — Filtrando carros por preço

Classificar itens do vetor

Ordenar strings

```
const nomes = ["Pedro","Ana","João"];

nomes.sort();
// ["Ana", "João", "Pedro"]

nomes.reverse();
// ["Pedro", "João", "Ana"]
```

Ordenar números

```
const nums = [50, 100, 2];

nums.sort();  // [100, 2, 50]
// Strings! Errado para números

nums.sort((a, b) => a - b);
// [2, 50, 100] Correto!
```

Ordenar objetos (por atributo)

```
// Ordenar carros por preço (crescente)
carros.sort((a, b) => a.preco - b.preco);
// Ordenar por nome (alfabético)
carros.sort((a, b) => a.modelo.localeCompare(b.modelo));
```

Resumo — Capítulo 5

`Declaração`	const arr = [] ou const arr = [1, 2, 3]
`push / pop`	Adiciona e remove do final
`unshift / shift`	Adiciona e remove do início
`.length`	Retorna o número de elementos
`indexOf`	Retorna índice ou -1 se não existir
`Objetos`	Vetores com { atributo: valor }
`filter()`	Filtra elementos por condição
`sort()`	Ordena — use (a,b) => a-b para números