# Lógica de Programação - JavaScript

## Aula 02

Rafael Neri

# Vetor = uma lista de valores

- Pense em um vetor como uma lista.

| | | | |
|---|---|---|---|
| ***Frutas:*** | Laranja | Uva | Abacaxi |

| | | | |
|---|---|---|---|
| ***Cores:*** | Vermelho | Verde | Azul |

- Um vetor é como uma sequência de caixas numeradas: cada caixa guarda um valor.
- Exemplo prático: um vetor que guarda as notas de um aluno.
  Por exemplo, `[8.0, 7.0, 9.0]` representam três notas.

# Exemplo em JavaScript — média de notas

```
const notas = [8.0, 7.0, 9.0];
const soma = notas[0] + notas[1] + notas[2];
const media = soma / 3;
console.log('Média =', media); // Média = 8
```

# Matriz — uma tabela de valores

- Uma matriz é como uma planilha com linhas e colunas;
  sua dimensão é `m × n` (m linhas, n colunas).
- Cada elemento é identificado por linha e coluna: `matriz[m][n]`, onde `m` é o número da linha e `n` o número da coluna.

Exemplo prático (2 linhas × 2 colunas):

```
[
  [8.0, 7.0, 9.0], // aluno A
  [6.0, 7.0, 8.0], // aluno B
]
```

# Exemplo em JavaScript — média de notas

```
const notas = [
  [8.0, 7.0, 9.0],
  [6.0, 7.0, 8.0],
];

const somaA = notas[0][0] + notas[0][1] + notas[0][2];
const somaB = notas[1][0] + notas[1][1] + notas[1][2];



const mediaA = somaA / 3;
const mediaB = somaB / 3;

console.log('Média Aluno A =', mediaA); // Média = 8
console.log('Média Aluno B =', mediaB); // Média = 7
```

# Laços de Repetição (Loops)

Nem sempre sabemos de antemão quantas vezes uma ação precisa ser executada.
Os **laços de repetição** permitem **executar um bloco de código várias vezes**, enquanto uma condição for verdadeira — ou percorrendo elementos de uma estrutura (como arrays e objetos).

## `for` — repetição com contador

Usado quando sabemos **quantas vezes** o loop deve executar.

```
for (let i = 0; i < 5; i++) {
  console.log("Contagem:", i);
}
```

**Explicação:**

- `let i = 0` → inicialização
- `i < 5` → condição de continuação
- `i++` → incremento a cada volta

Resultado: imprime de 0 a 4.

## `while` — repetição com condição

Usado quando **não sabemos quantas vezes** o loop deve ocorrer, mas temos uma condição que controla a repetição.

```
let contador = 0;

while (contador < 5) {
  console.log("Contagem:", contador);
  contador++;
}
```

O bloco será executado **enquanto a condição for verdadeira**.

⚠️ Cuidado: se esquecer de atualizar a variável, o loop será **infinito**!

## `do...while` — executa pelo menos uma vez

A diferença é que a verificação da condição vem **depois** do bloco.

Assim, o código executa **pelo menos uma vez**, mesmo que a condição seja falsa.

```
let numero = 1;

do {
  console.log("Número:", numero);
  numero++;
} while (numero <= 3);
```

Resultado: imprime `1`, `2`, `3`.

## `for...in` — percorrendo propriedades de um objeto

Serve para **iterar sobre as chaves (nomes das propriedades)** de um objeto.

```
const cliente = { nome: "Joaquim", idade: 54, ativo: true };

for (let chave in cliente) {
  console.log(chave + ":", cliente[chave]);
}
```

**Saída:**

```
nome: Joaquim
idade: 54
ativo: true
```

## `for...of` — percorrendo valores de coleções

Usado para **percorrer os valores** de arrays, strings ou objetos iteráveis.

```
const produtos = ["remédio", "vitamina", "curativo"];

for (let item of produtos) {
  console.log("Produto:", item);
}
```

Saída:

```
Produto: remédio
Produto: vitamina
Produto: curativo
```

## Diferença entre `for...in` e `for...of`

| Estrutura | Itera sobre | Usado com | Exemplo |
| --- | --- | --- | --- |
| `for...in` | Índices / chaves | Objetos e arrays (para índices) | `for (let i in array)` |
| `for...of` | Valores | Arrays, strings, coleções | `for (let valor of array)` |

## Exemplo prático: somando preços

```
const precos = [10.5, 8.0, 12.75];
let total = 0;

for (let preco of precos) {
  total += preco;
}

console.log("Total da compra:", total.toFixed(2));
```

## Controle de fluxo dentro de loops

Você pode **interromper** ou **pular** uma iteração:

- `break` → encerra o loop imediatamente
- `continue` → pula para a próxima iteração

```
for (let i = 1; i <= 10; i++) {
  if (i === 3) continue; // pula o número 3
  if (i === 5) break; // para no 5
  console.log(i);
}
```

Saída: `1, 2, 4`

# Métodos de Array

Arrays em JavaScript têm funções nativas para manipular dados de forma simples e poderosa — o famoso “superpoder” dos arrays.

## Métodos de Adição e Remoção

Esses métodos permitem manipular o tamanho do array, adicionando ou removendo elementos nas extremidades.

- `push()`
  Adiciona um item ao final do array (como colocar um produto novo na última prateleira).
- `pop()`
  Remove o último item (tirar o produto mais distante).
- `shift()`
  Remove o primeiro item (produto logo da frente).
- `unshift()`
  Adiciona um item no início (colocar algo logo na porta da prateleira).

## Métodos de Busca

Esses métodos procuram informações dentro do array.

- `includes()`
  Verifica se o item existe (true/false).
- `indexOf()`
  Encontra a posição de um valor.
- `find()`
  Retorna o primeiro elemento que satisfaz uma condição (ex.: “o primeiro cliente maior de 60 anos")
- `findIndex()`
  Retorna o índice do item que atende a condição.

## Métodos de Transformação

- `map()`

  **Cria um novo array transformando cada item com base em uma regra.**
  **É como pegar uma lista de preços e aplicar 10% de desconto em todos automaticamente.**

  **É muito usado para:**

  - formatar dados recebidos da API
  - converter tipos (strings para números)
  - gerar listas derivadas (ex.: nomes de usuários, preços com taxa, etc.)

## Métodos de Filtragem

- `filter()`
  **Cria um novo array menor, contendo apenas os itens que atendem a uma condição.**
  **É como separar somente as vitaminas dentro de um estoque que tem vários tipos de produtos.**

  **Usado para:**

  - filtrar produtos por categoria
  - buscar usuários ativos/inativos
  - criar listas personalizadas com base em permissões, status ou valores mínimos/máximos

## Métodos de Redução

- `reduce()`
  **Transforma um array inteiro em um único valor — que pode ser um número, string, objeto ou até outro array.**

  **Muito usado para:**

  - somar valores (ex.: total de vendas)
  - contar itens
  - agrupar informações
  - construir objetos a partir de listas
  - calcular médias

## Métodos de Ordenação

- `sort()`
  Organiza alfabeticamente ou numericamente (com função de comparação).
- `reverse()`
  Inverte a ordem dos elementos.

  Usos comuns:

  - ordenar nomes de clientes
  - classificar produtos pelo preço
  - organizar listas em dashboards
  - ordenar dados antes de exibir em tabelas

Rafael Neri