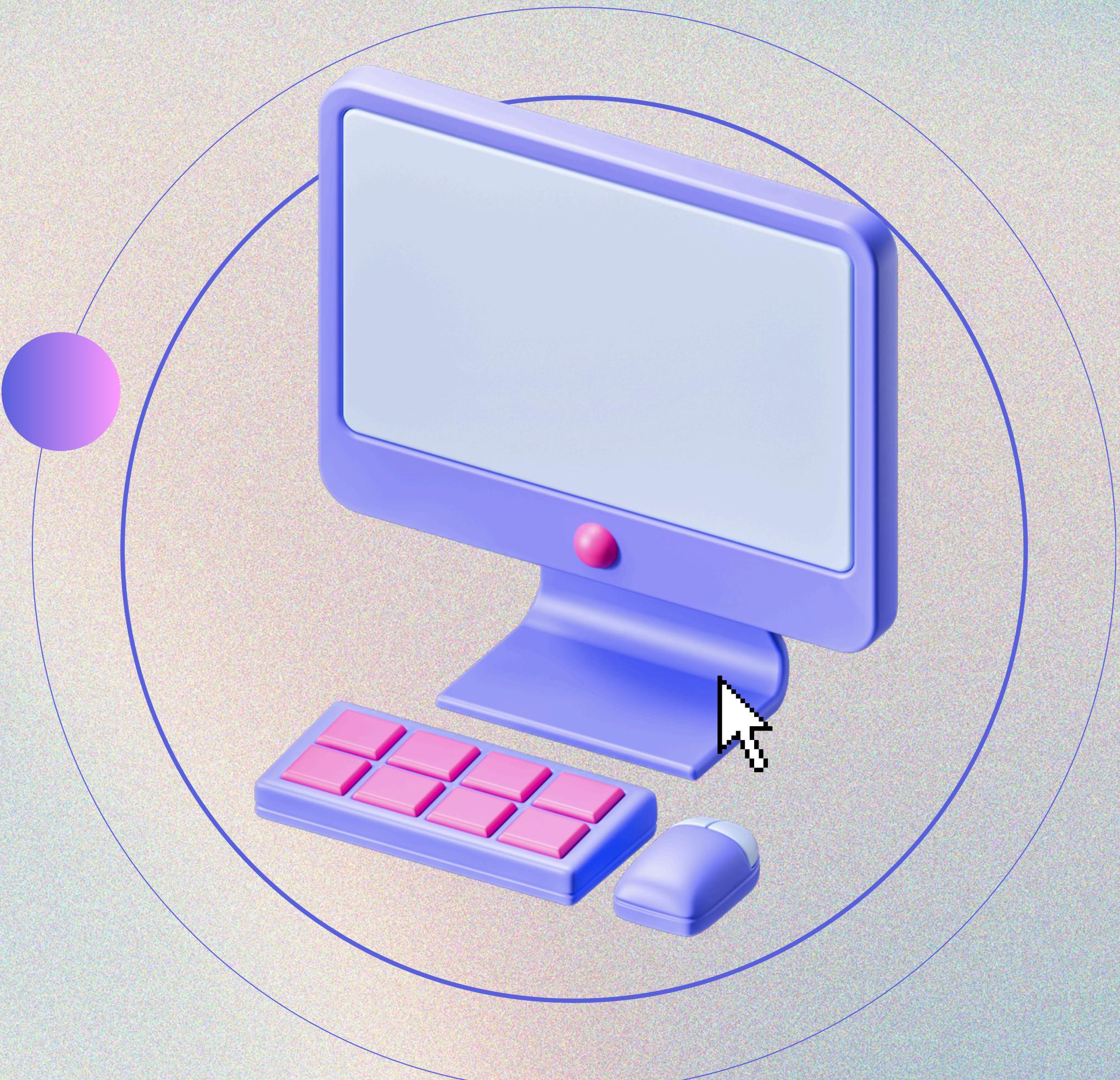


# ARRAYS ITERATION METHODS

Victória Ferreira & Kauã Oliveira



# Array.every()

O método `every()` serve para verificar se TODOS os itens de um array seguem uma condição.

Ex: Verifique se todos os números do array são maiores que 10?

```
//Array.Every
function mostrarEvery() {
    // Obter o valor do input e criar um array de números
    const input = document.getElementById('inputEvery').value;
    const array = input.split(',').map(item => parseInt(item.trim()));

    // Obter os elementos HTML para exibir os resultados
    const resultBox = document.getElementById('every02');
    document.getElementById('every01').innerHTML = `<strong>Lista:</strong> ${array.join(', ')}`;
    resultBox.innerHTML = '';

    // Verificar se todos os números do array são maiores que 0
    const resultado = array.every(item => item > 10);

    // Exibir o resultado
    resultBox.innerHTML += `Todos os números são maiores que 10? ${resultado}`;
    resultBox.classList.remove('hidden');
}
```

# Array.some()

O método some() serve para verifica se PELO MENOS UM item do array segue a regra.

Ex: Verifique se pelo menos 1 dos números do array são maiores que 10?

```
//Array.some
function mostrarSome() {
    // Obter o valor do input e criar um array de números
    const input = document.getElementById('inputSome').value;
    const array = input.split(',').map(item => parseInt(item.trim()));

    // Obter os elementos HTML para exibir os resultados
    const resultBox = document.getElementById('some02');
    document.getElementById('some01').innerHTML = `<strong>Lista:</strong> ${array.join(', ')}`;
    resultBox.innerHTML = '';

    // Verificar se existe algum número maior que 10 no array
    const resultado = array.some(item => item > 10);

    // Exibir o resultado
    resultBox.innerHTML += `Existe algum número maior que 10? ${resultado}`;
    resultBox.classList.remove('hidden');
}
```

# Array.from()

O método `from()` serve para criar um NOVO ARRAY a partir de outra coisa (tipo uma string, ou algo que parece array).

Ex: Transformar uma palavra em um array de letras.

```
function mostrarFrom() {  
    // Obter o valor do input e criar um array de letras  
    const inputFromArray = document.getElementById('inputFromArray').value;  
    const arrayFrom = Array.from(inputFromArray);  
  
    // Elemento para exibir o array original  
    const arrayFrom01 = document.getElementById('arrayFrom01');  
  
    // Exibir o array original  
    arrayFrom01.innerText = `Array.from(): ${arrayFrom}`;  
}
```

# Array.keys()

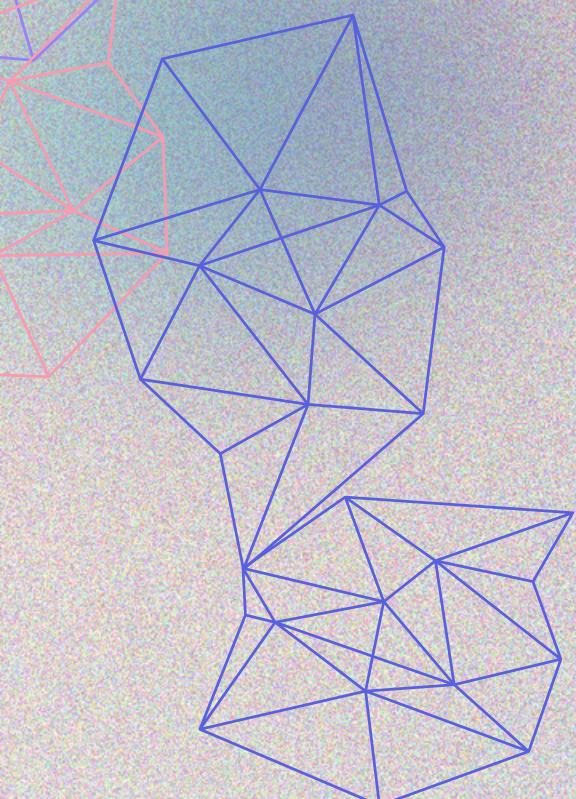
O método `keys()` serve para pegar todos os índices de um array. Quando você usa `array.keys()`, ele não devolve os valores, mas sim as posições (0, 1, 2, etc).

```
function mostrarKeys() {
    const input = document.getElementById('inputKeys').value;
    const array = input.split(',').map(item => item.trim());
    const resultBox = document.getElementById('keys02');
    document.getElementById('keys01').innerHTML = `<strong>Lista:</strong> ${array.join(', ')}`;
    resultBox.innerHTML = '';

    for (let i of array.keys()) {
        resultBox.innerHTML += `Índice ${i} = ${array[i]}<br>`;
    }

    resultBox.classList.remove('hidden');
}

let doces = ['chocolate', 'doce de leite', 'fini'];
for (let i of doces.keys()) {
    console.log(`Fruta na posição ${i}: ${doces[i]}`);
}
```



# Array.reduceRight()

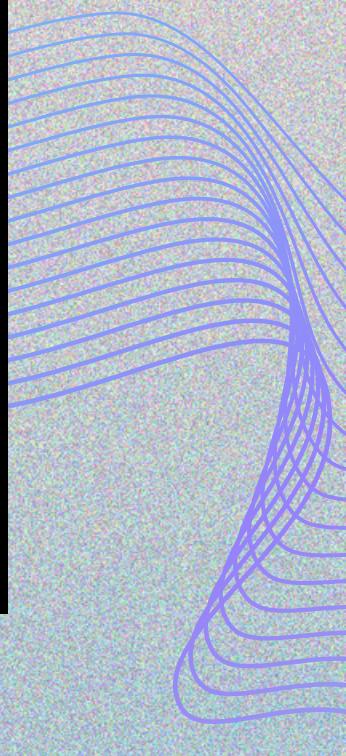
reduceRight() é muito parecido com reduce(), mas percorre o array da direita para a esquerda (do último elemento até o primeiro). Ele aplica uma função em cada elemento, acumulando um resultado final.

```
function mostrarReduceRight() {
  const input = document.getElementById('inputReduceRight').value;
  const arrayPalavras = input.split(',').map(p => p.trim());

  const frase = arrayPalavras.reduceRight((acc, curr) => acc + ' ' + curr);

  document.getElementById('reduceRight01').innerHTML = `<strong>Original:</strong> ${arrayPalavras.join(', ')}`;
  document.getElementById('reduceRight02').innerHTML = `<strong>Frase final:</strong> "${frase}"`;
  document.getElementById('reduceRight02').classList.remove('hidden');
}

const palavras = ['JS', 'é', 'legal'];
const frase = palavras.reduceRight((acc, curr) => acc + ' ' + curr);
console.log(frase); // "legal é JS"
```



# Array.with()

O método .with() serve para substituir um valor em uma posição específica do array, sem modificar o array original.

```
//WITH para HTML
function mostrarWith() {
    const input = document.getElementById('inputWith').value;
    const index = parseInt(document.getElementById('indexWith').value);
    const novoValor = document.getElementById('newValueWith').value;
    const array = input.split(',').map(item => item.trim());

    const novaArray = array.with(index, novoValor);

    document.getElementById('with01').innerHTML = `<strong>Array Original:</strong> ${array.join(', ')}`;
    document.getElementById('with02').innerHTML = `<strong>Nova lista:</strong> ${novaArray.join(', ')}`;
    document.getElementById('with02').classList.remove('hidden');
}

//WITH para exemplo no console
const series = ['The 100', 'Solo Leveling', 'Round 6'];
const novasSeries = series.with(0, 'Greys Anatomy');
console.log(novasSeries); // ['Greys Anatomy', 'Solo Leveling', 'Round 6']
console.log(series);     // ['The 100', 'Solo Leveling', 'Round 6']
```

# VAMOS VER NA PRÁTICA?

[https://victoriafe-sa.github.io/arrays\\_js/](https://victoriafe-sa.github.io/arrays_js/)





**OBRIGADO**