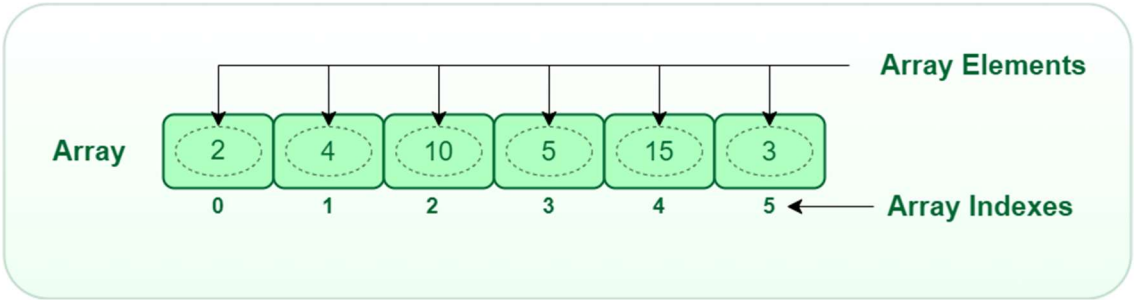
Estrutura de Dados em Javascript (ARRAY)

Um array em JavaScript é uma estrutura de dados que armazena uma coleção ordenada de elementos, como uma lista. Ele é importante porque permite organizar e acessar múltiplos valores de forma eficiente, facilitando operações como iteração, filtragem e manipulação de dados.

Analogia: Pense em um array como uma prateleira de livros. Cada livro (elemento) tem uma posição específica (índice), o que facilita encontrar, adicionar ou remover livros sem bagunçar toda a estante. Assim como a prateleira organiza os livros, o array organiza dados de forma estruturada e acessível.



Arrays em JavaScript têm várias particularidades que os tornam versáteis e poderosos:

1. **Tipagem Dinâmica:**
   Podem armazenar elementos de diferentes tipos (números, strings, objetos, até outros arrays).
2. **Tamanho Flexível:**
   O tamanho do array pode mudar dinamicamente, com métodos como push(), pop(), shift(), e unshift().
3. **Acesso por Índice:**
   Elementos são acessados por índices numéricos, começando do 0.
4. **Métodos Úteis:**
   Possui métodos embutidos como map(), filter(), reduce(), forEach(), que facilitam a manipulação de dados.
5. **Iteráveis:**
   Podem ser percorridos com loops (for, for...of) ou métodos como forEach().
6. **Arrays são Objetos:**
   Em JavaScript, arrays são um tipo especial de objeto, com propriedades e métodos específicos.
7. **Spread Operator (...):**
   Permite copiar, concatenar ou expandir arrays de forma simples.

8. **Arrays Esparsos:**
   Podem ter "buracos" (valores indefinidos) entre os elementos.
9. **Mutabilidade:**
   Arrays são mutáveis, ou seja, seus elementos podem ser alterados diretamente.
10. **Uso de Length:**
    A propriedade length não só informa o tamanho, mas também pode ser usada para redimensionar o array.

Categorias e Métodos de Arrays em JavaScript

1. Adicionar e Remover Elementos

Método	Descrição
push()	Adiciona ao final do array
pop()	Remove do final do array
shift()	Remove do início do array
unshift()	Adiciona ao início do array
splice()	Adiciona/Remove elementos em qualquer posição

2. Manipular e Copiar Arrays

Método	Descrição
slice()	Retorna uma cópia parcial do array
concat()	Une dois ou mais arrays em um novo array

3. Iterar e Transformar Arrays

Método	Descrição
forEach()	Executa uma ação para cada elemento
map()	Cria um novo array com base na transformação aplicada
reduce()	Reduz o array a um único valor (acumulador)

4. Filtrar e Buscar Elementos

Método	Descrição
filter()	Retorna elementos que satisfazem um critério
find()	Retorna o primeiro elemento encontrado
includes()	Verifica se um elemento existe no array

Métodos de Array em JavaScript - Adicionar e Remover Elementos

1. push()

Adiciona elementos ao final do array, retornando o novo tamanho.

Exemplos:

```
const frutas = ['maçã', 'banana'];
frutas.push('laranja');
console.log(frutas); // ['maçã', 'banana', 'laranja']

const numeros = [1, 2];
console.log(numeros.push(3, 4)); // 4
console.log(numeros); // [1, 2, 3, 4]
```

2. pop()

Remove o último elemento do array e retorna o elemento removido.

Exemplos:

```
const frutas = ['maçã', 'banana', 'uva'];
console.log(frutas.pop()); // 'uva'
console.log(frutas); // ['maçã', 'banana']
```

```
const numeros = [1, 2, 3, 4];
numeros.pop();
console.log(numeros); // [1, 2, 3]
```

3. shift()

Remove o primeiro elemento do array e retorna esse elemento.

Exemplos:

```
const frutas = ['maçã', 'banana', 'laranja'];
console.log(frutas.shift()); // 'maçã'
console.log(frutas); // ['banana', 'laranja']

const numeros = [5, 6, 7];
numeros.shift();
console.log(numeros); // [6, 7]
```

4. unshift()

Adiciona elementos ao início do array, retornando o novo tamanho.

Exemplos:

```
const frutas = ['banana', 'laranja'];
frutas.unshift('maçã');
console.log(frutas); // ['maçã', 'banana', 'laranja']

const numeros = [4, 5];
console.log(numeros.unshift(2, 3)); // 4
console.log(numeros); // [2, 3, 4, 5]
```

Métodos de Manipular e Copiar Arrays

5. slice()

Retorna uma cópia de parte do array, sem modificar o original.

Exemplos:

```
const numeros = [1, 2, 3, 4, 5];
```

```
console.log(numeros.slice(1, 3)); // [2, 3]

const frutas = ['maçã', 'banana', 'laranja', 'uva'];

console.log(frutas.slice(2)); // ['laranja', 'uva']
```

6. splice()

Modifica o conteúdo de um array removendo ou adicionando elementos.

Exemplos:

```
const numeros = [1, 2, 3, 4];

numeros.splice(1, 2); // remove a partir da posição 1, 2 elementos

console.log(numeros); // [1, 4]


const frutas = ['maçã', 'banana', 'laranja'];

frutas.splice(1, 1, 'uva', 'manga'); // remove 1 elemento na posição 1 e adiciona novos

console.log(frutas); // ['maçã', 'uva', 'manga', 'laranja']
```

7. concat()

Une dois ou mais arrays, retornando um novo array.

Exemplos:

```
const array1 = [1, 2];

const array2 = [3, 4];

console.log(array1.concat(array2)); // [1, 2, 3, 4]


const frutas = ['maçã'];

console.log(frutas.concat('banana', ['uva', 'laranja'])); // ['maçã', 'banana', 'uva', 'laranja']
```

Iterar e Transformar Arrays

8. Método forEach

O método forEach permite executar uma ação específica em cada elemento de um array. É como um professor passando por cada carteira na sala, verificando a atividade de cada aluno individualmente.

Exemplo passo a passo: