Lógica de Programação com Java Script

Othon Oliveira

SENAC - PROA

Sumário

Arrays e Vetores em JavaScript

- **Arrays e Vetores** são estruturas de dados usadas para armazenar coleções de elementos em JavaScript.

Arrays e Vetores em JavaScript

- **Arrays e Vetores** são estruturas de dados usadas para armazenar coleções de elementos em JavaScript.
- Um **array** é uma estrutura de dados que pode conter elementos de qualquer tipo, acessados por índices numéricos.

Arrays e Vetores em JavaScript

- **Arrays e Vetores** são estruturas de dados usadas para armazenar coleções de elementos em JavaScript.
- Um **array** é uma estrutura de dados que pode conter elementos de qualquer tipo, acessados por índices numéricos.
- Um **vetor** é um array que possui elementos do mesmo tipo, otimizado para operações matemáticas.

Criando Arrays e Vetores

- Em JavaScript, você pode criar um array usando colchetes: `let myArray = []`.

Criando Arrays e Vetores

- Em JavaScript, você pode criar um array usando colchetes: `let myArray = []`.
- Para criar um vetor, você pode usar bibliotecas como `NumPy` em Python ou `TypedArray` em JavaScript.

Acessando Elementos

- Para acessar um elemento em um array, use a notação de colchetes: `myArray[0]`.

Acessando Elementos

- Para acessar um elemento em um array, use a notação de colchetes: `myArray[0]`.
- Vetores geralmente possuem operações otimizadas para acessar elementos.

Propriedades dos Arrays

- `length`: Retorna o número de elementos no array.

Propriedades dos Arrays

- `length`: Retorna o número de elementos no array.
- `push()`: Adiciona um elemento ao final do array.

Propriedades dos Arrays

- `length`: Retorna o número de elementos no array.
- `push()`: Adiciona um elemento ao final do array.
- `pop()`: Remove e retorna o último elemento do array.

Propriedades dos Arrays

- `length`: Retorna o número de elementos no array.
- `push()`: Adiciona um elemento ao final do array.
- `pop()`: Remove e retorna o último elemento do array.
- `splice()`: Adiciona ou remove elementos em posições específicas.

Outras Propriedades dos Arrays

- `join()`: Converte os elementos em uma string, separados por um delimitador.

Outras Propriedades dos Arrays

- `join()`: Converte os elementos em uma string, separados por um delimitador.
- `concat()`: Combina dois ou mais arrays e retorna um novo.

Outras Propriedades dos Arrays

- `join()`: Converte os elementos em uma string, separados por um delimitador.
- `concat()`: Combina dois ou mais arrays e retorna um novo.
- `slice()`: Retorna uma cópia de parte do array, sem modificar o original.

Outras Propriedades dos Arrays

- `join()`: Converte os elementos em uma string, separados por um delimitador.
- `concat()`: Combina dois ou mais arrays e retorna um novo.
- `slice()`: Retorna uma cópia de parte do array, sem modificar o original.
- `indexOf()`: Retorna o índice do primeiro elemento correspondente.

Uma função que mostra os Elementos

```
function imprimirArray(arr) {
  // O código da função vai aqui
}
```

Chame a função

```
//cria um vetor
let meuArray = [1, 2, 3, 4, 5];

// chama a função, passando o array
imprimirArray(meuArray);
```

enfim a função

```
function imprimirArray(arr) {
  // O código da função vai aqui
}
```

Chame a função

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

Propriedade `length`

Length: Retorna o tamanho (número de elementos) do array.

```
let frutas = ["maçã", "banana", "laranja"];
let tamanho = frutas.length;
console.log(`O tamanho do array é ${tamanho}`);
```

Método push()

Push(): Adiciona um ou mais elementos ao final do array

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

Método pop()

Pop(): Remove e retorna o último elemento do array.

```
let frutas = ["maçã", "banana", "laranja"];
let ultimaFruta = frutas.pop();
console.log(ultimaFruta); // "laranja"
console.log(frutas); // ["maçã", "banana"]
```

Método splice()

splice(): Este método é usado para alterar o conteúdo de um array, adicionando ou removendo elementos. Ele recebe três argumentos: o índice inicial (onde a operação deve começar), o número de elementos a serem removidos e, opcionalmente, os elementos que devem ser adicionados ao array

```
let frutas = ["maçã", "banana", "laranja"];
frutas.splice(1, 1, "uva", "pêra");
console.log(frutas); // ["maçã", "uva", "pêra", "laranja"]
```

Método join()

Join(): Converte todos os elementos do array em uma única string, separados por um delimitador especificado

```
let frutas = ["maçã", "banana", "laranja"];
let texto = frutas.join(", ");
console.log(texto); // "maçã, banana, laranja"
```

Método concat()

Concat(): Combina dois ou mais arrays criando um novo array resultante

```
let frutas1 = ["maçã", "banana"];
let frutas2 = ["laranja", "uva"];
let frutasCombinadas = frutas1.concat(frutas2);
console.log(frutasCombinadas); // ["maçã", "banana", "laranja"
```

Método `slice()`

slice(): Este método retorna uma parte de um array, definida por um índice inicial e um índice final. Ele não modifica o array original, mas cria um novo array com os elementos selecionados.

```
let frutas = ["maçã", "banana", "laranja", "uva"];
let fatia = frutas.slice(1, 3);
console.log(fatia); // ["banana", "laranja"]
```

Método indexOf()

IndexOf(): Retorna o primeiro índice em que um elemento especificado pode ser encontrado no array, ou -1 se o elemento não estiver presente

```
let frutas = ["maçã", "banana", "laranja"];
let indice = frutas.indexOf("banana");
console.log(‘O índice de "banana" é ${indice}‘);
```

Exercício 1: Soma dos Elementos

```
// Dado um array de números, calcule a soma deles.
function somaArray(arr) {
  let soma = 0;
  for (let i = 0; i < arr.length; i++) {
    soma += arr[i];
  }
  return soma;
}
```

Exercício 2: Encontrar o Maior Elemento

```
// Encontre o maior elemento em um array.
function maiorElemento(arr) {
  let maior = arr[0];
  for (let i = 1; i < arr.length; i++) {
    if (arr[i] > maior) {
      maior = arr[i];
    }
  }
  return maior;
}
```

Exercício 3: Filtrar Números Pares

```
// Dado um array de números, filtre apenas os números pares.
function numerosPares(arr) {
  return arr.filter(numero => numero % 2 === 0);
}
```

Objetos

Em JavaScript, um objeto é uma coleção de pares chave-valor, onde cada chave (também chamada de propriedade) está associada a um valor. Os objetos são usados para representar estruturas de dados complexas e são uma das principais características da linguagem.

Objetos

Sintaxe: Os objetos são criados usando chaves {}. As propriedades e seus valores são definidos dentro das chaves, separados por dois-pontos :. As propriedades são separadas por vírgulas

Exemplo

```
let pessoa = {
  nome: "João",
  idade: 30,
  cidade: "São Paulo"
};
```

Exemplo

Acesso a Propriedades: Você pode acessar as propriedades de um objeto usando a notação de ponto (objeto.propriedade) ou a notação de colchetes (objeto['propriedade']).

```
console.log(pessoa.nome); // "João"
console.log(pessoa['idade']); // 30
```