**Array**

[**Sumário**](https://developer.mozilla.org/pt-BR/docs/Web/JavaScript/Reference/Global_Objects/Array#sum%C3%A1rio)

O objeto Array do JavaScript é um objeto global usado na construção de 'arrays': objetos de alto nível semelhantes a listas.

**Criando um Array**

var frutas = ['Maçã', 'Banana'];

console.log(frutas.length);

// 2

**Acessar um item (*index*) do Array**

var primeiro = frutas[0];

// Maçã

var ultimo = frutas[frutas.length - 1];

// Banana

**Iterar um Array**

frutas.forEach(function (item, indice, array) {

console.log(item, indice);

});

// Maçã 0

// Banana 1

**Adicionar um item ao final do Array**

var adicionar = frutas.push('Laranja');

// ['Maçã', 'Banana', 'Laranja']

**Remover um item do final do Array**

var ultimo = frutas.pop(); // remove Laranja (do final)

// ['Maçã', 'Banana'];

**Remover do início do Array**

var primeiro = frutas.shift(); // remove Maçã do início

// ['Banana'];

**Adicionar ao início do Array**

var adicionar = frutas.unshift('Morango') // adiciona ao início

// ['Morango', 'Banana'];

**Procurar o índice de um item na Array**

frutas.push('Manga');

// ['Morango', 'Banana', 'Manga']

var pos = frutas.indexOf('Banana');

// 1

**Remover um item pela posição do índice**

var removedItem = frutas.splice(pos, 1); // é assim que se remove um item

// ['Morango', 'Manga']

**Remover itens de uma posição de índice**

var vegetais = ['Repolho', 'Nabo', 'Rabanete', 'Cenoura'];

console.log(vegetais);

// ['Repolho', 'Nabo', 'Rabanete', 'Cenoura']

var pos = 1, n = 2;

var itensRemovidos = vegetais.splice(pos, n);

// Isso é como se faz para remover itens, n define o número de itens a se remover,

// a partir da posição (pos) em direção ao fim da array.

console.log(vegetais);

// ['Repolho', 'Cenoura'] (o array original é alterado)

console.log(itensRemovidos);

// ['Nabo', 'Rabanete']

**Copiar um Array**

var copiar = frutas.slice(); // é assim que se copia

// ['Morango', 'Manga']

[**Sintaxe**](https://developer.mozilla.org/pt-BR/docs/Web/JavaScript/Reference/Global_Objects/Array#sintaxe)

[*element0*, *element1*, ..., *elementN*]

new Array(*element0*, *element1*, ..., *elementN*)

new Array(*arrayLength*)

*element0*, *element1*, ..., *elementN*

Um array JavaScript é inicializado com os elementos contém, exceto no caso onde um único argumento é passado para o construtor do Array e esse argumento é um número (veja o parâmetro arrayLength abaixo). Esse caso especial só se aplica para os arrays JavaScript criados com o construtor Array , e não para literais de array criados com a sintaxe de colchetes [].

*arrayLength*

Se o único argumento passado para o construtor do Array for um número inteiro entre 0 e 232-1 (inclusive), um novo array com o tamanho desse número é retornado. Se o argumento for qualquer outro número, uma exceção [RangeError](https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/RangeError) é lançada.

[**Descrição**](https://developer.mozilla.org/pt-BR/docs/Web/JavaScript/Reference/Global_Objects/Array#descri%C3%A7%C3%A3o)

Arrays são objetos semelhantes a listas que vêm com uma série de métodos embutidos para realizar operações de travessia e mutação. Nem o tamanho de um array JavaScript nem os tipos de elementos são fixos. Já que o tamanho de um array pode ser alterado a qualquer momento e os dados podem ser armazenados em posições não contíguas, arrays JavaScript não tem a garantia de serem densos; isso depende de como o programador escolhe usá-los. De uma maneira geral, essas são características convenientes, mas, se esses recursos não são desejáveis para o seu caso em particular, você pode considerar usar arrays tipados.

Arrays não podem usar strings como índices (como em um [array associativo](https://pt.wikipedia.org/wiki/Vetor_associativo)), devem ser usados números inteiros. Definir ou acessar não-inteiros usando [notação de colchetes (ou notação de ponto)](https://developer.mozilla.org/pt-BR/docs/Web/JavaScript/Reference/Operators/Property_Accessors) não vai definir ou recuperar um elemento do array em si, mas sim definir ou acessar uma variável associada com a [coleção de propriedades de objeto](https://developer.mozilla.org/pt-BR/docs/Web/JavaScript/Data_structures#propriedades) daquele array. As propriedades de objeto do array e a lista de elementos do array são separados, e as operações de travessia e mutação não podem ser aplicadas a essas propriedades nomeadas.

[**Accessando elementos de um array**](https://developer.mozilla.org/pt-BR/docs/Web/JavaScript/Reference/Global_Objects/Array#accessando_elementos_de_um_array)

Arrays JavaScript começam com índice zero: o primeiro elemento de um array está na posição 0 e o último elemento está na  posição equivalente ao valor da propriedade [length](https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/Array/length) (tamanho) menos 1.

var arr = ['este é o primeiro elemento', 'este é o segundo elemento'];

console.log(arr[0]); // exibe 'este é o primeiro elemento'

console.log(arr[1]); // exibe 'este é o segundo elemento'

console.log(arr[arr.length - 1]); // exibe 'este é o segundo elemento'

Elementos de um array são somente propriedades de objetos, da forma que [toString](https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/Date/toString) é uma propriedade. Contudo, note que tentando acessar o primeiro elemento de um array da seguinte forma causará um erro de sintaxe, pois o nome da propriedade é inválido:

console.log(arr.0); // um erro de sintaxe

Não há nada de especial a respeito de arrays JavaScript e suas propriedades que causam isso. As propriedades JavaScript que começam com um dígito não podem ser referenciadas com notação de ponto. Elas necesitam usar notação de colchetes para poderem ser acessadas. Por exemplo, se você tivesse um objeto com a propriedade "3d", também teria que ser referenciá-la usando notação de colchetes. Por exemplo:

var anos = [1950, 1960, 1970, 1980, 1990, 2000, 2010];

console.log(anos.0); // um erro de sintaxe

console.log(anos[0]); // funciona corretamente

renderer.3d.setTexture(model, 'personagem.png'); // um erro de sintaxe

renderer['3d'].setTexture(model, 'personagem.png'); //funciona corretamente

Note que no exemplo 3d, '3d' teve de ser colocado entre aspas. É possivel também colocar entre aspas os índices de arrays JavaScript (ou seja, years['2'] ao invés de years[2]), contudo isto não é necessário. O valor 2 em years[2] eventualmente será convertido a uma string pela engine do JavaScript  através de uma conversão explicita com o método toString. E é por esta razão que  '2' e '02' irão referenciar dois slots diferentes no objeto anos e o seguinte exemplo pode ser *true*:

console.log(anos['2'] != anos['02']);

De forma similar, propriedades de objeto que sejam palavras reservadas(!) só podem ser acessadas como strings em notação de colchetes:

var promessa = {

'var': 'texto',

'array': [1, 2, 3, 4]

};

console.log(promessa['var']);

[**Relação entre *length* e propriedades numéricas**](https://developer.mozilla.org/pt-BR/docs/Web/JavaScript/Reference/Global_Objects/Array#rela%C3%A7%C3%A3o_entre_length_e_propriedades_num%C3%A9ricas)

As propriedades [length](https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/Array/length) e numéricas de um array Javascript são conectadas. Varios dos métodos javascript pré-definidos (por exemplo, [join](https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/Array/join), [slice](https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/Array/slice), [indexOf](https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/Array/indexOf) etc.) levam em conta o valor da propriedade length de um array quando eles são chamados. Outros métodos (por exemplo, [push](https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/Array/push), [splice](https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/Array/splice) etc.) também resultam em uma atualização na propriedade length do array.

var frutas = [];

frutas.push('banana', 'maça', 'pêssego');

console.log(frutas.length); // 3

Quando configurar uma propriedade num array Javascript em que a propriedade é um índice valido do array e este índice está fora do atual limite do array, o array irá crescer para um tamanho grande o suficiente para acomodar o elemento neste índice, e a engine irá atualizar a propriedade *length* do array de acordo com isto:

frutas[5] = 'manga';

console.log(frutas[5]); // 'manga'

console.log(Object.keys(frutas)); // ['0', '1', '2', '5']

console.log(frutas.length); // 6

Configurar a propriedade length diretamente, também resulta em um comportamento especial:

frutas.length = 10;

console.log(Object.keys(frutas)); // ['0', '1', '2', '5']

console.log(frutas.length); // 10

Diminuir o valor de length, entretanto, apaga elementos:

frutas.length = 2;

console.log(Object.keys(frutas)); // ['0', '1']

console.log(frutas.length); // 2

[**Criando um array usando o resultado de uma comparação**](https://developer.mozilla.org/pt-BR/docs/Web/JavaScript/Reference/Global_Objects/Array#criando_um_array_usando_o_resultado_de_uma_compara%C3%A7%C3%A3o)

O resultado de uma comparação entre uma *expressão regular* e uma string pode criar um array Javascript. Este array tem propriedades e elementos que disponibilizam informações sobre a comparação. Esse array é o valor de retorno dos métodos [RegExp.exec](https://developer.mozilla.org/pt-BR/docs/Web/JavaScript/Reference/Global_Objects/RegExp/exec), [String.match](https://developer.mozilla.org/pt-BR/docs/Web/JavaScript/Reference/Global_Objects/String/match), e [String.replace](https://developer.mozilla.org/pt-BR/docs/Web/JavaScript/Reference/Global_Objects/String/replace). Para explicar melhor sobre estas propriedades e elementos, veja o seguinte exemplo e então consulte a tabela abaixo:

// Encontra um d seguido por um ou mais b's seguido por um d

// Salva os b's encontrados e o d seguinte

// Ignora caixa (maiúscula/minúscula)

var minhaRegex = /d(b+)(d)/i;

var meuArray = minhaRegex.exec('cdbBdbsbz');

As propriedades e elementos retornados desta comparação são os seguintes:

|  |  |  |
| --- | --- | --- |
| Propriedade/Elemento | Descrição | Exemplo |
| input | Uma propriedade somente-leitura que reflete a string original a qual a expressão regular foi comparada. | cdbBdbsbz |
| index | Uma propriedade somente-leitura que é o índice baseado em zero da comparação na string. | 1 |
| [0] | Um elemento somente-leitura que especifica os ultimos caracteres que foram encontrados. | dbBd |
| [1], ...[n] | Elementos somente-leitura que especificam as *substrings* de comparações entre parênteses encontradas, se incluidas na expressão regular. O número de possíveis *substrings*entre parenteses é ilimitado. | [1]: bB [2]: d |

[**Propriedades**](https://developer.mozilla.org/pt-BR/docs/Web/JavaScript/Reference/Global_Objects/Array#propriedades)

Array.length

Propriedade comprimento do construtor Array, cujo valor é 1.

[get Array[@@species] (en-US)](https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/Array/@@species)

A função de construtor que é utilizada para criar objetos derivados.

Array.prototype

Permite a adição de propriedades para todos os objetos array.

[**Métodos**](https://developer.mozilla.org/pt-BR/docs/Web/JavaScript/Reference/Global_Objects/Array#m%C3%A9todos)

[Array.from()](https://developer.mozilla.org/pt-BR/docs/Web/JavaScript/Reference/Global_Objects/Array/from)

Cria uma nova instância de Array a partir de um objeto semelhante ou iterável.

[Array.isArray()](https://developer.mozilla.org/pt-BR/docs/Web/JavaScript/Reference/Global_Objects/Array/isArray)

Retorna true se a variável é um array e false caso contrário.

[Array.of()](https://developer.mozilla.org/pt-BR/docs/Web/JavaScript/Reference/Global_Objects/Array/of)

Cria uma nova instância de Array com um número variável de argumentos, independentemente do número ou tipo dos argumentos.

[**Instâncias de Array**](https://developer.mozilla.org/pt-BR/docs/Web/JavaScript/Reference/Global_Objects/Array#inst%C3%A2ncias_de_array)

Todas as instâncias de Array herdam de [Array.prototype](https://developer.mozilla.org/en-US/docs/JavaScript/Reference/Global_Objects/Array/prototype).  O protótipo do construtor Array pode ser modificado de forma a afetar todas as instâncias de Array.

[**Propriedades**](https://developer.mozilla.org/pt-BR/docs/Web/JavaScript/Reference/Global_Objects/Array#properties_of_array_instances)

{{ page('/pt-BR/docs/JavaScript/Reference/Global\_Objects/Array/prototype', 'Properties') }}

[**Métodos**](https://developer.mozilla.org/pt-BR/docs/Web/JavaScript/Reference/Global_Objects/Array#methods_of_array_instances)

**Métodos modificadores**

{{ page('/pt-BR/docs/JavaScript/Reference/Global\_Objects/Array/prototype', 'Mutator\_methods') }}

**Métodos de acesso**

{{ page('/pt-BR/docs/JavaScript/Reference/Global\_Objects/Array/prototype', 'Accessor\_methods') }}

**Métodos de iteração**

{{ page('/pt-BR/docs/JavaScript/Reference/Global\_Objects/Array/prototype', 'Iteration\_methods') }}

[**Métodos genéricos de Array**](https://developer.mozilla.org/pt-BR/docs/Web/JavaScript/Reference/Global_Objects/Array#m%C3%A9todos_gen%C3%A9ricos_de_array)

**Métodos genéricos de arrays não seguem o padrão, são obsoletos e serão removidos em breve.**

Algumas vezes você poderá querer aplicar métodos de arrays para strings ou outros objetos parecidos com arrays (como em [argumentos](https://developer.mozilla.org/en-US/docs/JavaScript/Reference/Functions_and_function_scope/arguments) de funções). Ao fazer isto, você trata uma string como um array de caracteres (ou em outros casos onde trata-se não-arrays como um array).  Por exemplo,  para checar se cada caractere em uma varivável *str* é uma letra, você poderia escrever:

function isLetter(character) {

return (character >= "a" && character <= "z");

}

if (Array.prototype.every.call(str, isLetter))

alert("A string '" + str + "' contém somente letras!");

Esta notação é um pouco despendiosa e o JavaScript 1.6 introduziu a seguinte abreviação genérica:

if (Array.every(isLetter, str))

alert("A string '" + str + "' contém somente letras!");

[Generics](https://developer.mozilla.org/en-US/docs/JavaScript/Reference/Global_Objects/String#String_generic_methods) também estão disponíveis em [String](https://developer.mozilla.org/en-US/docs/JavaScript/Reference/Global_Objects/String).

Estes não são atualmente parte dos padrões ECMAScript (através do ES2015 [Array.from()](https://github.com/monolithed/ECMAScript-6) pode se conseguir isto). A seguir segue uma adaptação para permitir o uso em todos os navegadores:

/\*globals define\*/

// Assumes Array extras already present (one may use shims for these as well)

(function () {

'use strict';

var i,

// We could also build the array of methods with the following, but the

// getOwnPropertyNames() method is non-shimable:

// Object.getOwnPropertyNames(Array).filter(function (methodName) {return typeof Array[methodName] === 'function'});

methods = [

'join', 'reverse', 'sort', 'push', 'pop', 'shift', 'unshift',

'splice', 'concat', 'slice', 'indexOf', 'lastIndexOf',

'forEach', 'map', 'reduce', 'reduceRight', 'filter',

'some', 'every', 'isArray'

],

methodCount = methods.length,

assignArrayGeneric = function (methodName) {

var method = Array.prototype[methodName];

Array[methodName] = function (arg1) {

return method.apply(arg1, Array.prototype.slice.call(arguments, 1));

};

};

for (i = 0; i < methodCount; i++) {

assignArrayGeneric(methods[i]);

}

}());

[**Exemplos**](https://developer.mozilla.org/pt-BR/docs/Web/JavaScript/Reference/Global_Objects/Array#exemplos)

[**Exemplo: Criando um array**](https://developer.mozilla.org/pt-BR/docs/Web/JavaScript/Reference/Global_Objects/Array#exemplo_criando_um_array)

O exemplo a seguir cria um array, msgArray, com *length* 0, então atribui valores para msgArray[0] e msgArray[99], trocando o *length* do array para 100.

var msgArray = new Array();

msgArray[0] = "Hello";

msgArray[99] = "world";

if (msgArray.length == 100)

print("O length é 100.");

[**Exemplo: Criando um array bi-dimensional**](https://developer.mozilla.org/pt-BR/docs/Web/JavaScript/Reference/Global_Objects/Array#exemplo_criando_um_array_bi-dimensional)

O exemplo a seguir cria um tabuleiro de xadrez usando dois arrays bi-dimensionais de string. A primeira jogada é feita copiando o 'p' em 6,4 para 4,4.  A posição antiga de 6,4 é colocada em branco.

var board =

[ ['R','N','B','Q','K','B','N','R'],

['P','P','P','P','P','P','P','P'],

[' ',' ',' ',' ',' ',' ',' ',' '],

[' ',' ',' ',' ',' ',' ',' ',' '],

[' ',' ',' ',' ',' ',' ',' ',' '],

[' ',' ',' ',' ',' ',' ',' ',' '],

['p','p','p','p','p','p','p','p'],

['r','n','b','q','k','b','n','r']];

print(board.join('\n') + '\n\n');

// Fazendo o King's Pawn avançar 2

board[4][4] = board[6][4];

board[6][4] = ' ';

print(board.join('\n'));

Saída:

R,N,B,Q,K,B,N,R

P,P,P,P,P,P,P,P

, , , , , , ,

, , , , , , ,

, , , , , , ,

, , , , , , ,

p,p,p,p,p,p,p,p

r,n,b,q,k,b,n,r

R,N,B,Q,K,B,N,R

P,P,P,P,P,P,P,P

, , , , , , ,

, , , , , , ,

, , , ,p, , ,

, , , , , , ,

p,p,p,p, ,p,p,p

r,n,b,q,k,b,n,r

[**Utilizando um array para tabular um conjunto de valores**](https://developer.mozilla.org/pt-BR/docs/Web/JavaScript/Reference/Global_Objects/Array#browser_compatibility)

values = [];

for (var x = 0; x < 10; x++){

values.push([

2 \*\* x,

2 \* x \*\* 2

])

};

console.table(values)

Saída:

0 1 0

1 2 2

2 4 8

3 8 18

4 16 32

5 32 50

6 64 72

7 128 98

8 256 128

9 512 162

# Array.prototype.findIndex()

O método **findIndex()** retorna o **índice** no array do primeiro elemento que satisfizer a função de teste provida. Caso contrário, retorna -1, indicando que nenhum elemento passou no teste.

Veja também o método [find()](https://developer.mozilla.org/pt-BR/docs/Web/JavaScript/Reference/Global_Objects/Array/find), que retorna o **valor** de um elemento encontrado no array em vez de seu índice.

## [Sintaxe](https://developer.mozilla.org/pt-BR/docs/Web/JavaScript/Reference/Global_Objects/Array/findIndex#sintaxe)

arr.findIndex(callback[, thisArg])

### [Parâmetros](https://developer.mozilla.org/pt-BR/docs/Web/JavaScript/Reference/Global_Objects/Array/findIndex#par%C3%A2metros)

callback

Função para executar em cada valor no array, tomando três argumentos:

element

O elemento atual sendo processado no array.

index

O índice do elemento atual sendo processado no array.

array

O array sobre o qual findIndex foi chamado.

thisArg

Opcional. Objeto para usar como this na execução do callback.

## [Descrição](https://developer.mozilla.org/pt-BR/docs/Web/JavaScript/Reference/Global_Objects/Array/findIndex#descri%C3%A7%C3%A3o)

O método findIndex executa a função callback uma vez para cada elemento presente no array até encontrar um onde o callback retorna um valor verdadeiro. Se tal elemento for encontrado, findIndex imediatamente retorna o índice deste elemento. Caso contrário, findIndex retorna -1. callback é invocado apenas para índices no array que têm valores atribuídos; nunca é invocado para índices que foram deletados ou que nunca tiveram valores atribuídos.

callback é invocado com três argumentos: o valor do elemento, o índice do elemento e o objeto Array sendo percorrido.

Se um parâmetro thisArg for fornecido para findIndex, ele será usado como o this para cada invocação do callback. Se não for fornecido, então [undefined](https://developer.mozilla.org/pt-BR/docs/Web/JavaScript/Reference/Global_Objects/undefined) é usado.

findIndex não modifica o array sobre o qual é chamado.

A série de elementos processados por findIndex é definida antes da primeira invocação do callback. Elementos que são adicionados ao array depois que a chamada a findIndex começa não serão visitados pelo callback. Se um elemento existente não visitado do array for modificado pelo callback, seu valor passado ao callback será o valor no momento em que findIndex visitar o índice deste elemento; elementos que forem deletados não são visitados.

## [Exemplos](https://developer.mozilla.org/pt-BR/docs/Web/JavaScript/Reference/Global_Objects/Array/findIndex#exemplos)

### [Encontrar o índice de um número primo em um array](https://developer.mozilla.org/pt-BR/docs/Web/JavaScript/Reference/Global_Objects/Array/findIndex#encontrar_o_%C3%ADndice_de_um_n%C3%BAmero_primo_em_um_array)

O seguinte exemplo encontra o índice de um elemento no array que é um número primo (ou retorna -1 se não houver número primo).

function isPrime(element, index, array) {

var start = 2;

while (start <= Math.sqrt(element)) {

if (element % start++ < 1) {

return false;

}

}

return element > 1;

}

console.log([4, 6, 8, 12].findIndex(isPrime)); // -1, não encontrado

console.log([4, 6, 7, 12].findIndex(isPrime)); // 2

## [Polyfill](https://developer.mozilla.org/pt-BR/docs/Web/JavaScript/Reference/Global_Objects/Array/findIndex#polyfill)

Esse método foi adicionado à especificação do ECMAScript 6 e pode não estar disponível em todas as implementações de JavaScript ainda. Contudo, você pode fazer o polyfill de Array.prototype.findIndex com o seguinte trecho de código:

if (!Array.prototype.findIndex) {

Array.prototype.findIndex = function(predicate) {

if (this === null) {

throw new TypeError('Array.prototype.findIndex called on null or undefined');

}

if (typeof predicate !== 'function') {

throw new TypeError('predicate must be a function');

}

var list = Object(this);

var length = list.length >>> 0;

var thisArg = arguments[1];

var value;

for (var i = 0; i < length; i++) {

value = list[i];

if (predicate.call(thisArg, value, i, list)) {

return i;

}

}

return -1;

};

}

# Array.prototype.join()

O método **join()** junta todos os elementos de um array (ou um [array-like object](https://developer.mozilla.org/en-US/docs/Web/JavaScript/Guide/Indexed_collections#Working_with_array-like_objects)) em uma string e retorna esta string.

## [Sintaxe](https://developer.mozilla.org/pt-BR/docs/Web/JavaScript/Reference/Global_Objects/Array/join#syntax)

arr.join([separador = ','])

### [Parâmetros](https://developer.mozilla.org/pt-BR/docs/Web/JavaScript/Reference/Global_Objects/Array/join#parameters)

separador Optional

Específica uma string para separar cada elemento adjacente do array. O separador é convertido em uma string se necessário. Se omitido, os elementos do array são separados com uma vírgula (","). Se o separador for uma string vazia, todos os elementos são juntados sem nenhum caracter entre eles.

### [Valor de retorno](https://developer.mozilla.org/pt-BR/docs/Web/JavaScript/Reference/Global_Objects/Array/join#parameters_2)

Uma string com todos os elementos do array juntos. Se arr.length é 0, uma string vazia é retornada.

## [Descrição](https://developer.mozilla.org/pt-BR/docs/Web/JavaScript/Reference/Global_Objects/Array/join#description)

As conversões em string de todos os elementos de um array são juntados em apenas uma string.

Obs: Se um elemento é undefined ou null, ele é convertido em uma string vazia.

## [Exemplos](https://developer.mozilla.org/pt-BR/docs/Web/JavaScript/Reference/Global_Objects/Array/join#examples)

### [Juntando um array de quatro formas diferentes](https://developer.mozilla.org/pt-BR/docs/Web/JavaScript/Reference/Global_Objects/Array/join#example_joining_an_array_three_different_ways)

O exemplo interativo a seguir cria um array, a, com três elementos, e o junta três vezes: a primeira com virgulas, a segunda so junta os elementos e a terceira com um sinal de menos.

## [Experimente](https://developer.mozilla.org/pt-BR/docs/Web/JavaScript/Reference/Global_Objects/Array/join#experimente)

### [Juntando um array-like object (objeto estilo-array)](https://developer.mozilla.org/pt-BR/docs/Web/JavaScript/Reference/Global_Objects/Array/join#juntando_um_array-like_object_objeto_estilo-array)

O exemplo abaixo junta um array-like object (ex: o objeto [arguments](https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Functions/arguments)), chamando [Function.prototype.call](https://developer.mozilla.org/pt-BR/docs/Web/JavaScript/Reference/Global_Objects/Function/call) no Array.prototype.join.

function f(a, b, c) {

var s = Array.prototype.join.call(arguments);

console.log(s); // '1,a,true'

}

f(1, 'a', true);

//saida esperada: "1,a,true"