 This page was translated from English by the community. Learn more and join the MDN Web Docs community.

String



Sumário

O objeto global `String` é um construtor para **strings**, ou uma sequência de caracteres.

Sintaxe

As formas literais de declaração de *String* são:

```
'string text'  
"string text"  
"中文 español English हिन्दी العربية português বাংলা русский 日本語 ਪੰਜਾਬੀ 한국어"
```

Além da forma regular, de caracteres de impressão, caracteres especiais podem ser codificados usando a *escape notation* (notação com barra invertida):

Codigo	Saida
\0	o caractere NULL
\'	aspas simples
\"	aspas duplas
\\	barra invertida
\n	nova linha
\r	carriage return
\v	tab vertical
\t	tab
\b	backspace
\f	form feed
\uXXXX	unicode codepoint
\xXX	the Latin-1 character

Ou, usando o objeto global `String` diretamente:

```
String(thing)  
new String(thing)
```

Parâmetros

`thing`

Qualquer coisa a ser convertida para uma string.

Descrição

Strings são úteis para guardar dados que podem ser representados em forma de texto. Uma das operações mais usadas nas **strings** é checar seu `tamanho`, para construir e concatená-las usando [os operadores + e +=](#), checando pela existência ou posição de *substrings* com o método `indexOf`, ou extrair *substrings* com o método `substring`.

Acesso à caractere

Há duas maneiras de acessar um caractere individual em uma string. A primeira é o método `charAt`:

```
return 'cat'.charAt(1); // returns "a"
```



A outra maneira (introduzido no ECMAScript 5) consiste em tratar a string como um objeto Array-like, onde os caracteres individuais correspondem a um índice numérico:

```
return 'cat'[1]; // returns "a"
```



Para acesso de caracteres usando uma notação de colchetes, tentando deletar ou designar um valor a estas propriedades não haverá sucesso. As propriedades envolvidas não são nem escritas ou configuráveis. (Veja [Object.defineProperty](#) para mais informações.)

Comparando strings

Desenvolvedores de C têm a função `strcmp()` para comparar strings. No JavaScript, basta usar o operador [maior que e menor que](#):

```
var a = "a";
var b = "b";
if (a < b) // verdadeiro
    print(a + " é menor que " + b);
else if (a > b)
    print(a + " é maior que " + b);
else
    print(a + " e " + b + " são iguais.");
```



Um resultado similar pode ser alcançado usando o método [localeCompare](#) herdado pelas instâncias de `String`.

Distinção entre String primitiva e objetos String

Note que o JavaScript distingue entre objetos String e valores de string primitivas. (O mesmo é válido para [Boolean](#) e [Numbers](#).)

Strings literais (definidas por aspas duplas ou aspas simples) e strings retornadas da chamada da função `String` fora do contexto de uma função construtora (sem o uso da palavra chave [new](#)) são strings primitivas. O JavaScript converte automaticamente strings primitivas para objetos do tipo `String`, por isso é possível utilizar os métodos do objeto `String` através de strings primitivas. Em contextos onde um método é invocado de uma string primitiva ou uma propriedade é procurada, o JavaScript irá criar um objeto com a string primitiva e executar o método ou acessar a propriedade procurada.

```
var s_prim = "foo";
var s_obj = new String(s_prim);

console.log(typeof s_prim); // Loga "string"
console.log(typeof s_obj);  // Loga "object"
```



String primitivas e objetos `String` também dão resultados diferentes quando usado `eval`. Primitivas passadas para `eval` são tratadas como código fonte; Objetos `String` são tratados como todos os outros objetos são, retornando o objeto. Por exemplo:

```
s1 = "2 + 2";           // cria uma string primitiva
s2 = new String("2 + 2"); // cria um objeto de String
console.log(eval(s1));   // retorna o número 4
console.log(eval(s2));   // retorna a string "2 + 2"
```



Por estas razões, o código pode quebrar quando encontra objetos `String` quando espera na verdade uma string primitiva, apesar de que geralmente autores não precisam se preocupar com a distinção.

Um objeto `String` pode ser convertido sempre para sua contraparte primitiva com o método `valueOf`.

```
console.log(eval(s2.valueOf())); // retorna o número 4
```



Note: Para uma outra possível abordagem para strings em JavaScript, favor ler o artigo sobre [StringView – a C-like representation of strings based on typed arrays](#).

Propriedades

String.prototype (en-US)

Permite a adição de propriedades a um objeto String.

 Properties inherited from Function :
arity , caller , constructor (en-US) , length , name


Métodos

String.fromCharCode()

Retorna uma string criada usando a sequência especificada de valores Unicode.

String.fromCodePoint()

Retorna uma string criada usando a sequência especificada de posições de código.

 Methods inherited from Function :
apply , call , toSource , toString

Métodos genéricos de Strings

Métodos de instância String também estão disponíveis no Firefox a partir de JavaScript 1.6 (embora não faça parte dos padrões ECMAScript) no objeto String para aplicar métodos String a qualquer objeto:

```
var num = 15;  
alert(String.replace(num, /5/, '2'));
```



Genéricos também estão disponíveis em métodos `Array`.

O seguinte é uma implementação para fornecer suporte a navegadores sem suporte:

```
/*globals define*/  
// Assume que todos os métodos de instância String fornecidos  
// já presentes (podem ser usadas implementações para este se não disponível)  
(function () {  
    'use strict';  
  
    var i,  
        // Nós também poderíamos construir o array de métodos com os seguintes,  
        // mas o método getOwnPropertyNames() não é implementável:  
        // Object.getOwnPropertyNames(String).filter(function (methodName)  
        // {return typeof String[methodName] === 'function'});  
        methods = [  
            'quote', 'substring', 'toLowerCase', 'toUpperCase', 'charAt',  
            'charCodeAt', 'indexOf', 'lastIndexOf', 'startsWith', 'endsWith',  
            'trim', 'trimLeft', 'trimRight', 'toLocaleLowerCase',  
            'toLocaleUpperCase', 'localeCompare', 'match', 'search',  
            'replace', 'split', 'substr', 'concat', 'slice'  
        ],  
        methodCount = methods.length,  
        assignStringGeneric = function (methodName) {  
            var method = String.prototype[methodName];  
            String[methodName] = function (arg1) {  
                return method.apply(arg1, Array.prototype.slice.call(arguments, 1));  
            };  
        };  
};
```



```
for (i = 0; i < methodCount; i++) {  
    assignStringGeneric(methods[i]);  
}  
}();
```

Instâncias de `String`

Propriedades

{{page('/pt-BR/docs/Web/JavaScript/Reference/Global_Objects/String/prototype', 'Propriedades')}}}

Métodos

Métodos não relacionados ao HTML

{{page('/pt-BR/docs/Web/JavaScript/Reference/Global_Objects/String/prototype', 'Métodos_não_relacionados_ao_HTML')}}}

Métodos de envoltório HTML

{{page('/pt-BR/docs/Web/JavaScript/Reference/Global_Objects/String/prototype', 'Métodos_de_envoltório_HTML')}}}

Exemplos

Conversão de String

É possível usar `String` como uma alternativa "mais segura" `toString`, como embora normalmente ainda chama o `toString` subjacente, também funciona para `null` e `undefined`. Por exemplo:


```
var outputStrings = [];  
for (let i = 0, n = inputValues.length; i < n; ++i) {  
  outputStrings.push(String(inputValues[i]));  
}
```



Especificações

Specification	Status	Comment
ECMAScript 1st Edition.	Standard	Definições iniciais.
ECMAScript 5.1 (ECMA-262) The definition of 'String' in that specification.	Padrão	
ECMAScript 2015 (6th Edition, ECMA-262) The definition of 'String' in that specification.	Padrão	

Compatibilidade com navegadores

 **[Estamos convertendo nossos dados de compatibilidade para o formato JSON](#)** . Esta tabela de compatibilidade ainda usa o formato antigo, pois ainda não convertemos os dados que ela contém. **[Descubra como você pode ajudar! \(en-US\)](#)**.

- [Desktop](#)
- [Dispositivo móvel](#)

Característica	Chrome	Firefox (Gecko)	Internet Explorer	Opera	Safari
Basic support	0.2	(Yes)	(Yes)	(Yes)	(Yes)

Característica	Android	Chrome for Android	Firefox Mobile (Gecko)	IE Mobile	Opera Mobile	Safari Mobile
Suporte básico	(Yes)	(Yes)	(Yes)	(Yes)	(Yes)	(Yes)

Veja também

- [DOMString](#)
- [StringView](#) – a C-like representation of strings based on typed arrays
- [Binary strings](#)

Last modified: 17 de jul. de 2021, [by MDN contributors](#)

