

我的集合



Developer Network

本文档将按"原样"提供。文档中所陈述的信息和观点，包括 URL 和其他 Internet 网站参考，如有更改，恕不另行通知。本文档不向您提供有关任何 Microsoft 产品或产品名称中的任何知识产权的合法权利。您可以在内部复制并使用本文档以进行参考。您可以在内部修改本文档以进行参考。© 2016 Microsoft. 版权所有。保留所有权利。使用条款 (<https://msdn.microsoft.com/cc300389.aspx>) | 商标 (<http://www.microsoft.com/library/toolbar/3.0/trademarks/en-us.mspix>)

Table Of Contents

第 1 章

[Array 对象 \(JavaScript\)](#)
[Date 对象 \(JavaScript\)](#)
[Function 对象 \(JavaScript\)](#)
[Map 对象 \(JavaScript\)](#)
[Math 对象 \(JavaScript\)](#)
[Object 对象 \(JavaScript\)](#)
[RegExp 对象 \(JavaScript\)](#)
[Set 对象 \(JavaScript\)](#)
[String 对象 \(JavaScript\)](#)

第 1 章

Array 对象 (JavaScript)

支持创建任何数据类型的数组。

语法

```
arrayObj = new Array()  
arrayObj = new Array([size])  
arrayObj = new Array([element0[, element1[, ...[, elementN]]]])
```

参数

arrayObj

必需。 **Array** 对象分配到的变量名称。

size

可选。 数组大小。 当数组基于零时，所创建的元素将具有从零到 *size* -1 的索引。

element0,...,elementN

可选。 要置于数组中的元素。 这将创建一个具有 $n + 1$ 个元素且 **length** 为 $n + 1$ 的数组。 一旦使用此语法，你必须提供多个元素。

备注

在创建数组后，你可以通过使用 [] 表示法来访问数组的各个元素。 注意，JavaScript 中的数组是从零开始的。

JavaScript

```
var my_array = new Array();  
for (i = 0; i < 10; i++) {  
    my_array[i] = i;  
}  
x = my_array[4];  
document.write(x);
```

```
// Output: 4
```

你可以将无符号 32 位整数传递到 **Array** 构造函数以指定数组的大小。如果该值为负或不是整数，则会发生运行时错误。如果你运行以下代码，则应会看到控制台中出现此错误。

JavaScript

```
var arr = new Array(10);
document.write(arr.length);

// Output: 10

// Don't do this
var arr = new Array(-1);
arr = new Array(1.50);
```

如果将单个值传递到 **Array** 构造函数，并且该值不是数字，则将 **length** 属性设置为 1，并且唯一元素的值将成为单个传入的参数。

npl

```
var arr = new Array("one");
document.write(arr.length);
document.write("<br/>");
document.write(arr[0]);

// Output:
1
one
```

JavaScript 数组是稀疏数组，即，并非数组内的所有元素都可能包含数据。在 JavaScript 中，数组中仅存在实际包含数据的元素。这将减少数组所使用的内存量。

要求

在以下文档模式中受支持：Quirks、Internet Explorer 6 标准模式、Internet Explorer 7 标准模式、Internet Explorer 8 标准模式、Internet Explorer 9 标准模式、Internet Explorer 10 标准模式和 Internet Explorer 11 标准模式。此外，也在应用商店应用（Windows 8 和 Windows Phone 8.1）中受支持。请参阅[版本信息](#)。

更高版本中已引入以下列表中的某些成员。有关详细信息，请参阅 [JavaScript 版本信息](#)或单个成员对应的文档。

属性

下表列出了 **Array** 对象的属性。

属性	描述
constructor 属性	指定创建数组的函数。

length 属性（数组）	返回一个整数值，此整数比数组中所定义的最高位元素大 1。
prototype 属性	返回对数组的原型的引用。

函数

下表介绍了 **Array** 对象的函数。

函数	描述
Array.from 函数	从类似数组的对象或可迭代的对象返回一个数组。
Array.isArray 函数	返回一个布尔值，该值指示对象是否为数组。
Array.of 函数	从传入的参数返回一个数组。

方法

下表列出了 **Array** 对象的方法。

方法	描述
concat 方法（数组）	返回由两个数组组合而成的新数组。
entries 方法	返回包含数组的键/值对的迭代器。
every 方法	检查定义的回调函数是否为数组中的所有元素返回 true 。
fill 方法	使用指定值填充数组。
filter 方法	对数组的每个元素调用定义的回调函数，并返回回调函数为其返回 true 的值的数组。
findIndex 方法	返回满足回调函数中指定的测试条件的第一个数组元素的索引值。
forEach 方法	为数组中的每个元素调用定义的回调函数。
hasOwnProperty 方法	返回一个布尔值，该值指示某个对象是否具有指定名称的属性。
indexOf 方法（数组）	返回某个值在数组中的第一个匹配项的索引。
isPrototypeOf 方法	返回一个布尔值，该值指示某个对象是否存在于另一个对象的原型链中。
join 方法	返回由一个数组的所有元素串联而成的 String 对象。

keys 方法	返回包含数组的索引值的迭代器。
lastIndexOf 方法（数组）	返回指定值在数组中的最后一个匹配项的索引。
map 方法	对数组的每个元素调用定义的回调函数并返回包含结果的数组。
pop 方法	从数组中移除最后一个元素并将该元素返回。
propertyIsEnumerable 方法	返回一个布尔值，该值指示指定属性是否为对象的一部分且是否可枚举。
push 方法	将新元素追加到一个数组中，并返回数组的新长度。
reduce 方法	通过对数组中的所有元素调用定义的回调函数来累积单个结果。回调函数的返回值是累积的结果，并且作为对回调函数的下一个调用中的参数提供。
reduceRight 方法	通过对数组中的所有元素调用定义的回调函数来按降序顺序累积单个结果。回调函数的返回值是累积的结果，并且作为对回调函数的下一个调用中的参数提供。
reverse 方法	将元素顺序被反转的 Array 对象返回。
shift 方法	从数组中移除第一个元素并将返回该元素。
slice 方法（数组）	返回一个数组中的一部分。
some 方法	检查定义的回调函数是否为数组的任何元素返回 true 。
sort 方法	返回一个元素已经进行了排序的 Array 对象。
splice 方法	从一个数组中移除元素，如有必要，在所移除元素的位置上插入新元素，并返回所移除的元素。
toLocaleString 方法	返回使用当前区域设置的字符串。
toString 方法	返回数组的字符串表示形式。
unshift 方法	在数组的开头插入新元素。
valueOf 方法	获取对数组的引用。
values 方法	返回包含数组的值的迭代器。

另请参阅

[滚动、平移和缩放示例应用](#)

Date 对象 (JavaScript)

启用日期和时间的基本存储和检索。

语法

```
dateObj = new Date()  
dateObj = new Date(dateVal)  
dateObj = new Date(year, month, date[, hours[, minutes[,  
seconds[,ms]]]])
```

参数

dateObj

必需。 **Date** 对象分配到的变量名称。

dateVal

必需。 如果是数值，*dateVal* 表示指定日期与 1970 年 1 月 1 日午夜之间相差的协调世界时的毫秒数。 如果是字符串，则根据 [日期和时间字符串 \(JavaScript\)](#) 中的规则分析 *dateVal*。 *dateVal* 参数也可以是从一些 ActiveX 对象返回的 VT_DATE 值。

year

必需。 年份全称，如 1976（而不是 76）。

month

必需。 月份，用从 0 到 11 的整数表示（1 月至 12 月）。

date

必需。 日期，用从 1 到 31 的整数表示。

hours

可选。 如果提供了 *minutes* 参数，那么必须提供此参数。 一个指定小时的，从 0 到 23 的整数（午夜到 11pm）。

分钟

可选。 如果提供了 *seconds* 参数，那么必须提供此参数。 一个指定分钟的，从 0 到 59 的整数。

seconds

可选。 如果提供了 *milliseconds* 参数，那么必须提供此参数。 一个指定秒的，从 0 到 59 的整数。

ms

可选。 一个指定毫秒的，从 0 到 999 的整数。

备注

Date 对象包含表示具体时间（可以精确到毫秒）的数字。 如果参数值大于其取值范围或者为负，则会相应地修改存储的其他值。 例如，如果指定 150 秒，则 JavaScript 将该数字重新定义为 2 分 30 秒。

如果数字是 **NaN**，则对象不表示特定的时间时刻。 如果没有将任何参数传递到 **Date** 对象，该函数将初始化为当前时间 (UTC)。 必须先为对象赋值，然后才能使用它。

可在 **Date** 对象中表示的日期范围大约为 1970 年 1 月 1 日左右的 285,616 年。

有关如何使用 **Date** 对象和相关方法的更多信息，请参见[计算日期和时间 \(JavaScript\)](#)。

示例

下面的示例阐释了 **Date** 对象的用法。

JavaScript

```
var dateString = "Today's date is: ";

var newDate = new Date();

// Get the month, day, and year.
dateString += (newDate.getMonth() + 1) + "/";
dateString += newDate.getDate() + "/";
dateString += newDate.getFullYear();

document.write(dateString);

// Output: Today's date is: <date>
```

要求

Internet Explorer 6 之前的 Internet Explorer 中已引入 **Date object**。 更高版本中已引入以下列表中的某些成员。 有关更多信息，请参见 [JavaScript 版本信息](#)或与单个成员对应的文档。

属性

下表列出了 **Date Object** 的属性。

Property	描述
constructor 属性	指定创建一个对象的函数。
prototype 属性	为对象的类返回原型的引用。

函数

下表列出了 **Date Object** 的函数。

函数	描述
----	----

Date.now 函数	返回 1970 年 1 月 1 日与当前日期和时间之间的毫秒数。
Date.parse 函数	分析一个包含日期的字符串，并返回该日期与 1970 年 1 月 1 日午夜之间相差的毫秒数。
Date.UTC 函数	返回协调通用时间 (UTC) (或 GMT) 1970 年 1 月 1 日午夜与所提供的日期之间相差的毫秒数。

方法

下表列出了 **Date Object** 的方法。

方法	描述
getDate 方法	使用当地时间返回一个月某天的值。
getDay 方法	使用当地时间返回一个星期某天的值。
getFullYear 方法	使用当地时间返回年份值。
getHours 方法	使用当地时间返回小时值。
getMilliseconds 方法	使用当地时间返回毫秒值。
getMinutes 方法	使用当地时间返回分钟值。
getMonth 方法	使用当地时间返回月份值。
getSeconds 方法	使用当地时间返回秒值。
getTime 方法	将 Date 对象中的时间值返回为自 1970 年 1 月 1 日午夜起经过的毫秒数。
getTimezoneOffset 方法	返回主机的时间与协调通用时间 (UTC) 之间的分钟差值。
getUTCDate 方法	使用 UTC 返回一个月某天的值。
getUTCDay 方法	使用 UTC 返回一个星期某天的值。
getUTCFullYear 方法	使用 UTC 返回年份值。
getUTCHours 方法	使用 UTC 返回小时值。
getUTCMilliseconds 方法	使用 UTC 返回毫秒值。
getUTCMinutes 方法	使用 UTC 返回分钟值。

getUTCMonth 方法	使用 UTC 返回月份值。
getUTCSeconds 方法	使用 UTC 返回秒值。
getVarDate 方法	将 Date 对象中的 VT_DATE 值返回。
getFullYear 方法	返回年份值。
hasOwnProperty 方法	返回一个布尔值，该值指示一个对象是否具有指定名称的属性。
isPrototypeOf 方法	返回一个布尔值，该值指示对象是否存在于另一个对象的原型链中。
propertyIsEnumerable 方法	返回一个布尔值，该值指示指定属性是否为对象的一部分以及该属性是否是可枚举的。
setDate 方法	使用当地时间设置一个月中某一日的数值。
setFullYear 方法	使用当地时间设置年份值。
setHours 方法	使用当地时间设置小时值。
setMilliseconds 方法	使用当地时间设置毫秒值。
setMinutes 方法	使用当地时间设置分钟值。
setMonth 方法	使用当地时间设置月份值。
setSeconds 方法	使用当地时间设置秒值。
setTime 方法	设置 Date 对象中的日期和时间值。
setUTCDate 方法	使用 UTC 设置一个月中某一日的数值。
setUTCFullYear 方法	使用 UTC 设置年份值。
setUTCHours 方法	使用 UTC 设置小时值。
setUTCMilliseconds 方法	使用 UTC 设置毫秒值。
setUTCMinutes 方法	使用 UTC 设置分钟值。
setUTCMonth 方法	使用 UTC 设置月份值。
setUTCSeconds 方法	使用 UTC 设置秒值。
setYear 方法	使用当地时间设置年份值。
toDateString 方法	以字符串值的形式返回一个日期。
toGMTString 方法	返回使用格林尼治标准时间 (GMT) 转换为字符串的日期。
toISOString 方法	以字符串值的形式返回采用 ISO 格式的日期。

toJSON 方法	用于在 JSON 序列化之前转换目标类型的数据。
toLocaleDateString 方法	将一个日期以字符串值的形式返回，该字符串应适合于宿主环境的当前区域设置。
toLocaleString 方法	返回使用当前区域设置转换为字符串的对象。
toLocaleTimeString 方法	以字符串值的形式返回一个时间，此字符串值应与宿主环境的当前区域设置相适应。
toString 方法	返回表示对象的字符串。
getTimeString 方法	以字符串值形式返回时间。
toUTCString 方法	返回使用 UTC 转换为字符串的日期。
valueOf 方法	返回指定对象的原始值。

另请参阅

- [计算日期和时间 \(JavaScript\)](#)
- [日期和时间字符串 \(JavaScript\)](#)

© 2016 Microsoft

Function 对象 (JavaScript)

创建新函数。

语法

```
function functionName([argname1 [, ...[, argnameN]]])
{
    body
}
```

语法

```
functionName = new Function( [argname1, [... argnameN,]] body );
```

参数

- functionName*
必需。 新创建的函数的名称。
- argname1...argnameN*
可选。 该函数接受的参数的列表。
- body*
可选。 包含在调用函数时要执行的 JavaScript 代码块的字符串。

备注

该函数是 JavaScript 中的基本数据类型。 语法 1 创建了一个函数值，JavaScript 可在需要时将其转换为 **Function** 对象。 在调用该函数时，JavaScript 将根据语法 2 创建的 **Function** 对象转换为函数值。

语法 1 是在 JavaScript 中创建新函数的标准方式。 语法 2 是用于显式创建函数对象的替代形式。

例如，若要声明一个可将两个传递给它的参数相加的函数，可通过以下两种方法之一实现此目的：

示例 1

JavaScript

```
function add(x, y)
{
    return(x + y);
}
```


示例 2

```
var add = function(x, y) {
    return(x+y);
}
```

在任一示例中，您使用类似于以下内容的代码行调用该函数：

JavaScript

```
add(2, 3);
```

 说明

在调用函数时，请确保始终包括圆括号和任何必选参数。 调用不带括号的函数会导致返回函数本身，而不是函数的返回值。

属性

[0... n 属性](#) | [arguments 属性](#) | [callee 属性](#) | [caller 属性](#) | [constructor 属性](#) | [length 属性（函数）](#) | [prototype 属性](#)

方法

[apply 方法](#) | [bind 方法](#) | [call 方法](#) | [toString 方法](#) | [valueOf 方法](#)

要求

在以下文档模式中受支持：Quirks、Internet Explorer 6 标准模式、Internet Explorer 7 标准模式、Internet Explorer 8 标准模式、Internet Explorer 9 标准模式、Internet Explorer 10 标准模式和 Internet Explorer 11 标准模式。此外，也在应用商店应用（Windows 8 和 Windows Phone 8.1）中受支持。请参阅[版本信息](#)。

另请参阅

[function 语句 \(JavaScript\)](#)
[new 运算符 \(JavaScript\)](#)
[var 语句 \(JavaScript\)](#)

© 2016 Microsoft

Map 对象 (JavaScript)

键/值对的集合。

语法

JavaScript

```
mapObj = new Map()
```

备注

集合中的键和值可以是任何类型。如果使用现有密钥向集合添加值，则新值会替换旧值。

属性

下表列出了 **Map** 对象的属性。

Property	描述
构造函数	指定创建映射的函数。
Prototype — 原型	为映射返回对原型的引用。
size	返回映射中的元素数。

方法

下表列出了 **Map** 对象的方法。

方法	描述
clear	从映射中移除所有元素。
删除	从映射中移除指定的元素。
forEach	对映射中的每个元素执行指定操作。
get	返回映射中的指定元素。
有	如果映射包含指定元素，则返回 true 。
set	添加一个新建元素到映射。
toString	返回映射的字符串表示形式。
valueOf	返回指定对象的原始值。

下面的示例演示如何将成员添加到 **Map**，然后检索它们。

JavaScript

```
var m = new Map();
m.set(1, "black");
m.set(2, "red");
m.set("colors", 2);
m.set({x:1}, 3);

m.forEach(function (item, key, mapObj) {
    document.write(item.toString() + "<br />");
});

document.write("<br />");
document.write(m.get(2));
```

```
// Output:  
// black  
// red  
// 2  
// 3  
//  
// red
```

要求

在 Internet Explorer 11 标准文档模式下支持此项。此外，也在应用商店应用（Windows 8.1 和 Windows Phone 8.1）中受支持。请参阅[版本信息](#)。

在以下文档模式中不受支持：Quirks、Internet Explorer 6 标准模式、Internet Explorer 7 标准模式、Internet Explorer 8 标准模式、Internet Explorer 9 标准模式和 Internet Explorer 10 标准模式。在 Windows 8 中不受支持。

© 2016 Microsoft

Math 对象 (JavaScript)

提供基本数学功能和常量的内部对象。

语法

```
Math.[{property | method}]
```

参数

属性

必需。 **Math** 的某个属性的名称。的名称。

方法

必需。 **Math** 的某个方法的名称。的名称。

备注

无法使用 **new** 运算符创建 **Math** 对象，如果你尝试执行此操作，将收到错误。加载脚本引擎时，该引擎将创建它。其所有方法和属性始终对你的脚本可用。

要求

Internet Explorer 6 之前的 Internet Explorer 中已引入 **Math** 对象。

常量

下表列出了 **Math** 对象的常量。

返回的常量	描述
Math.E 常量	数学常数 e。这是欧拉数，自然对数的底。
Math.LN2 常量	2 的自然对数。
Math.LN10 常量	10 的自然对数。
Math.LOG2E 常量	以 2 为底 e 的对数。
Math.LOG10E 常量	以 10 为底 e 的对数。
Math.PI 常量	Pi。这是圆的周长与直径的比值。
Math.SQRT1_2 常量	0.5 的平方根，或相当于 1 除以 2 的平方根。
Math.SQRT2 常量	2 的平方根。

函数

下表列出了 **Math** 对象的函数。

函数	描述
Math.abs 函数	返回数字的绝对值。
Math.acos 函数	返回数字的反余弦值。
Math.acosh 函数	返回数字的双曲反余弦值（或反双曲余弦值）。
Math.asin 函数	返回数字的正弦值。
Math.asinh 函数	返回数字的反双曲正弦。
Math.atan 函数	返回数字的正切值。
Math.atan2 函数	将与 x 轴的角度（以弧度为单位）返回到由 y 和 x 坐标表示的点。
Math.atanh 函数	返回数字的反双曲正切。

Math.ceil 函数	返回大于或等于提供的数值表达式的最小整数。
Math.cos 函数	返回数字的余弦值。
Math.cosh 函数	返回数字的双曲余弦。
Math.exp 函数	返回 e（自然对数的底）的乘幂数。
Math.expm1 函数	返回 e（自然对数的底）的乘幂数减去 1 的结果。
Math.floor 函数	返回小于或等于提供的数值表达式的最大整数。
Math.hypot 函数	返回参数平方和的平方根。
Math.imul 函数	返回被视为 32 位带符号整数的两个数字的积。
Math.log 函数	返回数字的自然对数。
Math.log1p 函数	返回 1 加上一个数字的自然对数。
Math.log10 函数	返回数字以 10 为底的对数。
Math.log2 函数	返回数字以 2 为底的对数。
Math.max 函数	返回提供的两个数值表达式中的较大值。
Math.min 函数	返回提供的两个数字中的较小值。
Math.pow 函数	返回基表达式的指定乘幂数的值。
Math.random 函数	返回介于 0 和 1 之间的伪随机数。
Math.round 函数	返回舍入到最近整数的指定数值表达式。
Math.sign 函数	返回数字符号，它指示数字为正数、负数还是 0。
Math.sin 函数	返回数字的正弦值。
Math.sinh 函数	返回数字的反双曲正弦。
Math.sqrt 函数	返回数字的平方根。
Math.tan 函数	返回数字的正切值。
Math.tanh 函数	返回数字的双曲正切。
Math.trunc 函数	返回数字的整数部分，删除任何小数数字。

请参阅

Object 对象 (JavaScript)

提供对所有 JavaScript 对象通用的功能。

语法

```
obj
= new Object([value])
```

参数

- obj*
必需。 **Object** 对象分配到的变量名称。
- 值*
可选。 任一 JavaScript 基元数据类型（数字、布尔值或字符串）。 如果值是一个对象，则返回的对象是未修改的。 如果 *值* 是 **null**、“未定义”或“未提供”，则创建无内容的对象。

备注

Object 对象包含在所有其他 JavaScript 对象中；它的所有方法和属性均可用于所有其他对象。 方法可在用户定义的对象中进行重新定义，并由 JavaScript 在适当时间调用。 **toString**方法是频繁重新定义的 **Object** 方法的一个示例。

在此语言参考中，每个 **Object** 方法的说明均包括内部 JavaScript 对象的默认和对象特定的实现信息。

要求

已在 Internet Explorer 6 之前的 Internet Explorer 中引入 **Object Object**。 更高版本中已引入以下列表中的某些成员。

属性

下表列出了 **Object Object** 的属性。

属性	描述
----	----

__proto__ 属性	指定对象的原型。
constructor 属性	指定用于创建对象的函数。
prototype 属性	为对象的类返回原型的引用。

函数

下表列出了 **Object** 的函数。

函数	描述
Object.assign 函数	将来自一个或多个源对象中的值复制到一个目标对象。
Object.create 函数	创建具有指定原型并可选择包含指定属性的对象。
Object.defineProperties 函数	将一个或多个属性添加到对象，和/或修改现有属性的特性。
Object.defineProperty 函数	将属性添加到对象，或修改现有属性的特性。
Object.freeze 函数	防止修改现有属性的特性和值，并防止添加新属性。
Object.getOwnPropertyDescriptor 函数	返回数据属性或访问器属性的定义。
Object.getOwnPropertyNames 函数	返回对象属性及方法的名称。
Object.getOwnPropertySymbols 函数	返回对象的符号属性。
Object.getPrototypeOf 函数	返回对象的原型。
Object.is 函数	返回一个值，该值指示两个值是否相同。
Object.isExtensible 函数	返回指示是否可将新属性添加到对象的值。
Object.isFrozen 函数	如果无法在对象中修改现有属性的特性和值，并且无法将新属性添加到对象，则返回 true 。
Object.isSealed 函数	如果无法在对象中修改现有属性特性，并且无法将新属性添加到对象，则返回 true 。
Object.keys 函数	返回对象的可枚举属性和方法的名称。
Object.preventExtensions 函数	防止向对象添加新属性。
Object.seal 函数	防止修改现有属性的特性，并防止添加新属性。

Object.setPrototypeOf 函数	设置对象的原型。
--------------------------	----------

方法

下表列出了 **Object** 的方法。

方法	描述
hasOwnProperty 方法	返回一个布尔值，该值指示某个对象是否具有指定名称的属性。
isPrototypeOf 方法	返回一个布尔值，该值指示某个对象是否存在于另一个对象的原型层次结构中。
propertyIsEnumerable 方法	返回一个布尔值，该值指示指定属性是否为对象的一部分且是否可枚举。
toLocaleString 方法	返回基于当前区域设置转换为字符串的对象。
toString 方法	返回对象的字符串表示形式。
valueOf 方法	返回指定对象的基元值。

另请参阅

[JavaScript 对象](#)

RegExp 对象 (JavaScript)

内部全局对象，它存储有关正则表达式模式匹配结果的信息。

语法

<code>RegExp.property</code>

备注

必需的 *property* 参数可以是任何一个 **RegExp** 对象属性。

RegExp 对象不能直接创建，但它始终可用。在完成成功的正则表达式搜索之前，**RegExp** 对象的各种属性具有如下初始值：

属性	简写	初始值
index		-1
input	\$ _	空字符串。
lastIndex		-1
lastMatch	\$&	空字符串。
lastParen	\$+	空字符串。
leftContext	\$`	空字符串。
rightContext	\$'	空字符串。
\$1 - \$9	\$1 - \$9	空字符串。

在成功完成正则表达式搜索之前，其属性将 undefined 作为其值。

不应将全局 **RegExp** 对象与 **Regular Expression** 对象混淆。即使它们看起来相同，但它们是完全不同且相互分离的。全局 **RegExp** 对象的属性包含有关所发生的每一匹配的不断更新的信息，而 **Regular Expression** 对象的属性只包含有关与 **Regular Expression** 实例发生的匹配的信息。

下面的示例执行正则表达式搜索。它显示了来自全局 **RegExp** 对象和由 **exec** 方法返回的数组的匹配项和子匹配项。

JavaScript

```
var newLine = "<br />";

var re = /(\w+)@(\w+)\.(\w+)/g
var src = "Please send mail to george@contoso.com and someone@example.com. Thanks!"

var result;
var s = "";

// Get the first match.
result = re.exec(src);

while (result != null) {
    // Show the entire match.
    s += newLine;

    // Show the match and submatches from the RegExp global object.
    s += "RegExp.lastMatch: " + RegExp.lastMatch + newLine;
    s += "RegExp.$1: " + RegExp.$1 + newLine;
    s += "RegExp.$2: " + RegExp.$2 + newLine;
    s += "RegExp.$3: " + RegExp.$3 + newLine;
```

```
// Show the match and submatches from the array that is returned
// by the exec method.
for (var index = 0; index < result.length; index++) {
    s += index + ":";
    s += result[index];
    s += newLine;
}

// Get the next match.
result = re.exec(src);
}
document.write(s);

// Output:
// RegExp.lastMatch: george@contoso.com
// RegExp.$1: george
// RegExp.$2: contoso
// RegExp.$3: com
// 0: george@contoso.com
// 1: george
// 2: contoso
// 3: com

// RegExp.lastMatch: someone@example.com
// RegExp.$1: someone
// RegExp.$2: example
// RegExp.$3: com
// 0: someone@example.com
// 1: someone
// 2: example
// 3: com
```

属性

[\\$1...\\$9 属性](#) | [index 属性](#) | [input 属性](#) | [lastIndex 属性](#) | [lastMatch 属性](#) | [lastParen 属性](#) | [leftContext 属性](#) | [rightContext 属性](#)

方法

RegExp 对象没有方法。

要求

在以下文档模式中受支持：Quirks、Internet Explorer 6 标准模式、Internet Explorer 7 标准模式、Internet Explorer 8 标准模式、Internet Explorer 9 标准模式、Internet Explorer 10 标准模式和 Internet Explorer 11 标准模式。此外，也在应用商店应用（Windows 8 和 Windows Phone 8.1）中受支持。请参阅[版本信息](#)。

请参阅

[正则表达式对象 \(JavaScript\)](#)
[Regular Expression Syntax \(JavaScript\)](#)
[String 对象 \(JavaScript\)](#)

© 2016 Microsoft

Set 对象 (JavaScript)

可以是任何类型的单个值的集合。

语法

```
setObj = new Set()
```

备注

如果尝试将非唯一值添加到 **Set**，则新值不会添加到集合中。

属性

下表列出了 **Set** 对象的属性。

Property	描述
构造函数	指定创建集的函数。
Prototype — 原型	为集返回对原型的引用。
size	返回集中的元素数。

方法

下表列出了 **Set** 对象的方法。

方法	描述
添加	添加元素至集。

clear	从集内移除所有元素。
删除	从集中移除指定的元素。
forEach	对集合中的每个元素执行指定操作。
有	如果集包含指定元素，则返回 true 。
toString	返回集的字符串表示形式。
valueOf	返回指定对象的原始值。

下面的示例演示如何将成员添加到一个设置，然后检索它们。

JavaScript

```
var s = new Set();
s.add("Thomas Jefferson");
s.add(1776);
s.add("founding father");

s.forEach(function (item) {
    document.write(item.toString() + ",");
});

// Output:
// Thomas Jefferson, 1776, founding father,
```

要求

在 Internet Explorer 11 标准文档模式下支持此项。此外，也在应用商店应用（Windows 8.1 和 Windows Phone 8.1）中受支持。请参阅[版本信息](#)。

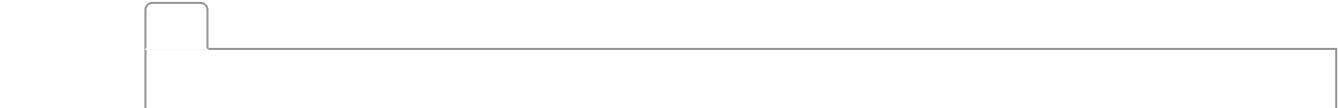
在以下文档模式中不受支持：Quirks、Internet Explorer 6 标准模式、Internet Explorer 7 标准模式、Internet Explorer 8 标准模式、Internet Explorer 9 标准模式和 Internet Explorer 10 标准模式。在 Windows 8 中不受支持。

© 2016 Microsoft

String 对象 (JavaScript)

允许操作和格式化文本字符串以及确定和定位字符串中的子字符串。

语法




```
newString = new String(["stringLiteral"])
```

参数

newString

必需。 **String** 对象分配到的变量名称。

stringLiteral

可选。 任何 Unicode 字符组。

备注

JavaScript 提供了可以包含在字符串中的转义序列，以创建不能直接键入的字符。 例如，`\t` 指定制表符。 有关详细信息，请参阅[特殊字符 \(JavaScript\)](#)。

字符串

字符串文本是以单引号或双引号括起零个或多个字符。 字符串文本具有 **string** 的主（基元）数据类型。 *String* 对象通过使用 [new 运算符](#) 创建，其数据类型为 **Object**。

以下示例显示字符串文本的数据类型与 **String** 对象的数据类型不同。

JavaScript

```
var strLit = "This is a string literal.";
var strObj = new String("This is a string object.");

document.write(typeof strLit);
document.write("<br/>");
document.write(typeof strObj);
// Output:
// string
// object
```

用于字符串文本的方法

在字符串文本上调用方法时，该方法将临时转换为字符串包装器对象。 字符串文本将被视为好像是使用 **new** 运算符创建的。

以下示例将 **toUpperCase** 方法应用于字符串文本。

JavaScript

```
var strLit = "This is a string literal.";

var result1 = strLit.toUpperCase();

var result2 = (new String(strLit)).toUpperCase();
```

```
document.write(result1);
document.write("<br/>");
document.write(result2);
// Output:
// THIS IS A STRING LITERAL.
// THIS IS A STRING LITERAL.
```

访问单个字符

可以将字符串的单个字符作为只读数组索引属性进行访问。 Internet Explorer 9 标准模式、Internet Explorer 10 标准模式、Internet Explorer 11 标准模式和 Windows 应用商店应用 中已引入此功能。 以下示例访问单个字符串字符。

JavaScript

```
var str = "abcd";
var result = str[2];
document.write (result);
// Output: c

var result = "the"[0];
document.write(result);
// Output: t
```

要求

已在 Internet Explorer 6 之前的 Internet Explorer 中引入 **String Object**。 更高版本中已引入以下列表中的某些成员。

属性

下表列出了 **String** 对象的属性。

属性	描述
constructor 属性	指定用于创建对象的函数。
length 属性（字符串）	返回 String 对象的长度。
prototype 属性	为对象的类返回原型的引用。

函数

下表列出了 **String** 对象的函数。

函数	描述
String.fromCharCode 函数	从若干 Unicode 字符值中返回一个字符串。
String.fromCodePoint 函数	返回与 Unicode UTF-16 码位关联的字符串。
String.raw 函数	返回模板字符串的原始字符串形式。

方法

下表列出了 **String** 对象的方法。

方法	描述
anchor 方法	将具有 NAME 特性的 HTML 定位点放置在文本两侧。
big 方法	将 HTML <BIG> 标记放置在文本两侧。
blink 方法	将 HTML <BLINK> 标记放置在文本两侧。
bold 方法	将 HTML 标记放置在文本两侧。
charAt 方法	返回指定索引处的字符。
charCodeAt 方法	返回指定字符的 Unicode 编码。
codePointAt 方法	返回一个 Unicode UTF-16 字符的码位。
concat 方法（字符串）	返回由提供的两个字符串串联而成的字符串。
EndsWith 方法	返回一个布尔值，该值指示字符串或子字符串是否以传入字符串结尾。
includes 方法	返回一个布尔值，该值指示传入字符串是否包含在字符串对象中。
fixed 方法	将 HTML <TT> 标记放置在文本两侧。
fontcolor 方法	将具有 COLOR 特性的 HTML 标记放置在文本两侧。
fontsize 方法	将具有 SIZE 特性的 HTML 标记放置在文本两侧。
hasOwnProperty 方法	返回一个布尔值，该值指示某个对象是否具有指定名称的属性。
indexOf 方法（字符串）	返回字符串内第一次出现子字符串的字符位置。
isPrototypeOf 方法	返回一个布尔值，该值指示某个对象是否存在于另一个对象的原型链中。
italics 方法	将 HTML <I> 标记放置在文本两侧。

lastIndexOf 方法 (字符串)	返回字符串内子字符串的最后一个匹配项。
link 方法	将具有 HREF 特性的 HTML 定位点放置在文本两侧。
localeCompare 方法	返回一个值，该值指示两个字符串在当前区域设置中是否相等。
match 方法	通过使用提供的正则表达式对象来搜索字符串并以数组形式返回结果。
normalize 方法	返回指定字符串的 Unicode 范式。
propertyIsEnumerable 方法	返回一个布尔值，该值指示指定属性是否为对象的一部分且是否可枚举。
repeat 方法	返回一个新的字符串对象，它的值等于重复了指定次数的原始字符串。
replace 方法	使用正则表达式替换字符串中的文本并返回结果。
search 方法	返回正则表达式搜索中第一个子字符串匹配项的位置。
slice 方法 (字符串)	返回字符串的片段。
small 方法	将 HTML <SMALL> 标记放置在文本两侧。
split 方法	返回一个字符串拆分为若干子字符串时所产生的字符串数组。
StartsWith 方法	返回一个布尔值，该值指示字符串或子字符串是否以传入字符串开头。
strike 方法	将 HTML <STRIKE> 标记放置在文本两侧。
sub 方法	将 HTML <SUB> 标记放置在文本两侧。
substr 方法	返回一个从指定位置开始且具有指定长度的子字符串。
substring 方法	返回 String 对象中指定位置处的子字符串。
sup 方法	将 HTML <SUP> 标记放置在文本两侧。
toLocaleLowerCase 方法	返回一个字符串，其中所有字母字符都转换为小写形式，并将考虑主机环境的当前区域设置。
toLocaleString 方法	返回使用当前区域设置转换为字符串的对象。
toLocaleUpperCase 方法	返回一个字符串，其中所有字母字符都转换为大写形式，并将考虑主机环境的当前区域设置。
toLowerCase 方法	返回一个字符串，其中所有字母字符都转换为小写形式。
toString 方法	返回字符串。
toUpperCase 方法	返回一个字符串，其中所有字母字符都转换为大写形式。
trim 方法	返回已移除前导空格、尾随空格和行终止符的字符串。

valueOf 方法	返回字符串。
------------	--------

另请参阅

[new 运算符 \(JavaScript\)](#)
[滚动、平移和缩放示例应用](#)

© 2016 Microsoft