W3School JavaScript 参考手册

来源：www.w3school.com.cn

整理：飞龙

日期：2014.10.3

目录

HTML DOM 对象

JavaScript Array 对象参考手册

Array 对象

Array 对象用于在单个的变量中存储多个值。

创建 Array 对象的语法：

```
new Array();
new Array(size);
new Array(element0, element1, ..., elementn);
```

参数

参数 *size* 是期望的数组元素个数。返回的数组，length 字段将被设为 *size* 的值。
参数 *element* ..., *elementn* 是参数列表。当使用这些参数来调用构造函数 Array() 时，新创建的数组的元素就会被初始化为这些值。它的 length 字段也会被设置为参数的个数。

返回值

返回新创建并被初始化了的数组。
如果调用构造函数 Array() 时没有使用参数，那么返回的数组为空，length 字段为 0。
当调用构造函数时只传递给它一个数字参数，该构造函数将返回具有指定个数、元素为 undefined 的数组。
当其他参数调用 Array() 时，该构造函数将用参数指定的值初始化数组。
当把构造函数作为函数调用，不使用 new 运算符时，它的行为与使用 new 运算符调用它时的行为完全一样。

Array 对象属性

FF: Firefox, IE: Internet Explorer

属性	描述	FF	IE
constructor	返回对创建此对象的数组函数的引用。	1	4
index		1	4
input		1	4
length	设置或返回数组中元素的数目。	1	4
prototype	使您有能力向对象添加属性和方法。	1	4

Array 对象方法

FF: Firefox, IE: Internet Explorer

方法	描述	FF	IE
concat()	连接两个或更多的数组，并返回结果。	1	4
join()	把数组的所有元素放入一个字符串。元素通过指定的分隔符进行分隔。	1	4
pop()	删除并返回数组的最后一个元素	1	5.5
push()	向数组的末尾添加一个或更多元素，并返回新的长度。	1	5.5
reverse()	颠倒数组中元素的顺序。	1	4
shift()	删除并返回数组的第一个元素	1	5.5
slice()	从某个已有的数组返回选定的元素	1	4
sort()	对数组的元素进行排序	1	4
splice()	删除元素，并向数组添加新元素。	1	5.5
toSource()	返回该对象的源代码。	1	-
toString()	把数组转换为字符串，并返回结果。	1	4
toLocaleString()	把数组转换为本地数组，并返回结果。	1	4
unshift()	向数组的开头添加一个或更多元素，并返回新的长度。	1	6
valueOf()	返回数组对象的原始值	1	4

JavaScript constructor 属性

定义和用法

constructor 属性返回对创建此对象的数组函数的引用。

语法

```
object.constructor
```

实例

例子 1

在本例中，我们将展示如何使用 constructor 属性：

```
<script type="text/javascript">

var test=new Array();

if (test.constructor==Array)
{
document.write("This is an Array");
}
if (test.constructor==Boolean)
{
document.write("This is a Boolean");
}
if (test.constructor==Date)
{
document.write("This is a Date");
}
if (test.constructor==String)
{
document.write("This is a String");
}

</script>
```

输出：

```
This is an Array
```

例子 2

在本例中，我们将展示如何使用 constructor 属性：

```
<script type="text/javascript">

function employee(name,job,born)
{
this.name=name;
this.job=job;
this.born=born;
}

var bill=new employee("Bill Gates","Engineer",1985);

document.write(bill.constructor);
```

```
</script>
```

输出：

```
function employee(name, jobtitle, born)
{this.name = name; this.jobtitle = job; this.born = born;}
```

JavaScript length 属性

定义和用法

length 属性可设置或返回数组中元素的数目。

语法

```
arrayObject.length
```

说明

数组的 length 属性总是比数组中定义的最后一个元素的下标大 1。对于那些具有连续元素，而且以元素 0 开始的常规数组而言，属性 length 声明了数组中的元素的个数。
数组的 length 属性在用构造函数 Array() 创建数组时被初始化。给数组添加新元素时，如果必要，将更新 length 的值。
设置 length 属性可改变数组的大小。如果设置的值比其当前值小，数组将被截断，其尾部的元素将丢失。如果设置的值比它的当前值大，数组将增大，新的元素被添加到数组的尾部，它们的值为 undefined。

实例

在本例中，我们将展示如何使用 length 属性返回并设置数组的长度：

```
<script type="text/javascript">

var arr = new Array(3)
arr[0] = "John"
arr[1] = "Andy"
arr[2] = "Wendy"
```

```
document.write("Original length: " + arr.length)
document.write("<br />")

arr.length=5
document.write("New length: " + arr.length)

</script>
```

输出：

```
Original length: 3
New length: 5
```

JavaScript prototype 属性

定义和用法

prototype 属性使您有能力向对象添加属性和方法。

语法

```
object.prototype.name=value
```

实例

在本例中，我们将展示如何使用 prototype 属性来向对象添加属性：

```
<script type="text/javascript">

function employee(name,job,born)
{
this.name=name;
this.job=job;
this.born=born;
}

var bill=new employee("Bill Gates","Engineer",1985);

employee.prototype.salary=null;
```

```
bill.salary=20000;

document.write(bill.salary);

</script>
```

输出：

```
20000
```

JavaScript concat() 方法

定义和用法

concat() 方法用于连接两个或多个数组。
该方法不会改变现有的数组，而仅仅会返回被连接数组的一个副本。

语法

```
arrayObject.concat(arrayX,arrayX,.....,arrayX)
```

参数	描述
arrayX	必需。该参数可以是具体的值，也可以是数组对象。可以是任意多个。

返回值

返回一个新的数组。该数组是通过把所有 arrayX 参数添加到 arrayObject 中生成的。如果要进行 concat() 操作的参数是数组，那么添加的是数组中的元素，而不是数组。

实例

例子 1

在本例中，我们将把 concat() 中的参数连接到数组 a 中：

```
<script type="text/javascript">
```

```
var a = [1,2,3];
document.write(a.concat(4,5));

</script>
```

输出：

```
1,2,3,4,5
```

例子 2

在本例中，我们创建了两个数组，然后使用 concat() 把它们连接起来：

```
<script type="text/javascript">

var arr = new Array(3)
arr[0] = "George"
arr[1] = "John"
arr[2] = "Thomas"

var arr2 = new Array(3)
arr2[0] = "James"
arr2[1] = "Adrew"
arr2[2] = "Martin"

document.write(arr.concat(arr2))

</script>
```

输出：

```
George,John,Thomas,James,Adrew,Martin
```

例子 3

在本例中，我们创建了三个数组，然后使用 concat() 把它们连接起来：

```
<script type="text/javascript">

var arr = new Array(3)
arr[0] = "George"
arr[1] = "John"
arr[2] = "Thomas"
```

```
var arr2 = new Array(3)
arr2[0] = "James"
arr2[1] = "Adrew"
arr2[2] = "Martin"

var arr3 = new Array(2)
arr3[0] = "William"
arr3[1] = "Franklin"

document.write(arr.concat(arr2,arr3))

</script>
```

输出：

```
George,John,Thomas,James,Adrew,Martin,William,Franklin
```

JavaScript join() 方法

定义和用法

join() 方法用于把数组中的所有元素放入一个字符串。
元素是通过指定的分隔符进行分隔的。

语法

```
arrayObject.join(separator)
```

参数	描述
separator	可选。指定要使用的分隔符。如果省略该参数，则使用逗号作为分隔符。

返回值

返回一个字符串。该字符串是通过把 arrayObject 的每个元素转换为字符串，然后把这些字符串连接起来，在两个元素之间插入 *separator* 字符串而生成的。

实例

例子 1

在本例中，我们将创建一个数组，然后把它的所有元素放入一个字符串：

```
<script type="text/javascript">

var arr = new Array(3)
arr[0] = "George"
arr[1] = "John"
arr[2] = "Thomas"

document.write(arr.join())

</script>
```

输出：

```
George,John,Thomas
```

例子 2

在本例中，我们将使用分隔符来分隔数组中的元素：

```
<script type="text/javascript">

var arr = new Array(3)
arr[0] = "George"
arr[1] = "John"
arr[2] = "Thomas"

document.write(arr.join("."))

</script>
```

输出：

```
George.John.Thomas
```

JavaScript pop() 方法

定义和用法

pop() 方法用于删除并返回数组的最后一个元素。

语法

```
arrayObject.pop()
```

返回值

arrayObject 的最后一个元素。

说明

pop() 方法将删除 arrayObject 的最后一个元素，把数组长度减 1，并且返回它删除的元素的值。如果数组已经为空，则 pop() 不改变数组，并返回 undefined 值。

实例

在本例中，我们将创建一个数组，然后删除数组的最后一个元素。请注意，这也会改变数组的程度：

```
<script type="text/javascript">

var arr = new Array(3)
arr[0] = "George"
arr[1] = "John"
arr[2] = "Thomas"

document.write(arr)

document.write("<br />")

document.write(arr.pop())

document.write("<br />")
```

```
document.write(arr)

</script>
```

输出：

```
George,John,Thomas
Thomas
George,John
```

JavaScript push() 方法

定义和用法

push() 方法可向数组的末尾添加一个或多个元素，并返回新的长度。

语法

```
arrayObject.push(newelement1,newelement2,....,newelementX)
```

参数	描述
newelement1	必需。要添加到数组的第一个元素。
newelement2	可选。要添加到数组的第二个元素。
newelementX	可选。可添加多个元素。

返回值

把指定的值添加到数组后的新长度。

说明

push() 方法可把它的参数顺序添加到 arrayObject 的尾部。它直接修改 arrayObject，而不是创建一个新的数组。push() 方法和 pop() 方法使用数组提供的先进后出栈的功能。

提示和注释

注释：该方法会改变数组的长度。
提示：要想数组的开头添加一个或多个元素，请使用 unshift() 方法。

实例

在本例中，我们将创建一个数组，并通过添加一个元素来改变其长度：

```
<script type="text/javascript">

var arr = new Array(3)
arr[0] = "George"
arr[1] = "John"
arr[2] = "Thomas"

document.write(arr + "<br />")
document.write(arr.push("James") + "<br />")
document.write(arr)

</script>
```

输出：

```
George,John,Thomas
4
George,John,Thomas,James
```

JavaScript reverse() 方法

定义和用法

reverse() 方法用于颠倒数组中元素的顺序。

语法

```
arrayObject.reverse()
```

提示和注释

注释：该方法会改变原来的数组，而不会创建新的数组。

实例

在本例中，我们将创建一个数组，然后颠倒其元素的顺序：

```
<script type="text/javascript">

var arr = new Array(3)
arr[0] = "George"
arr[1] = "John"
arr[2] = "Thomas"

document.write(arr + "<br />")
document.write(arr.reverse())

</script>
```

输出：

```
George,John,Thomas
Thomas,John,George
```

JavaScript shift() 方法

定义和用法

shift() 方法用于把数组的第一个元素从其中删除，并返回第一个元素的值。

语法

```
arrayObject.shift()
```

返回值

数组原来的第一个元素的值。

说明

如果数组是空的，那么 shift() 方法将不进行任何操作，返回 undefined 值。请注意，该方法不创建新数组，而是直接修改原有的 arrayObject。

提示和注释

注释：该方法会改变数组的长度。
提示：要删除并返回数组的最后一个元素，请使用 pop() 方法。

实例

在本例中，我们将创建一个数组，并删除数组的第一个元素。请注意，这也将改变数组的长度：

```
<script type="text/javascript">

var arr = new Array(3)
arr[0] = "George"
arr[1] = "John"
arr[2] = "Thomas"

document.write(arr + "<br />")
document.write(arr.shift() + "<br />")
document.write(arr)

</script>
```

输出：

```
George,John,Thomas
George
John,Thomas
```

JavaScript slice() 方法

定义和用法

slice() 方法可从已有的数组中返回选定的元素。

语法

```
arrayObject.slice(start,end)
```

参数	描述
start	必需。规定从何处开始选取。如果是负数，那么它规定从数组尾部开始算起的位置。也就是说，-1 指最后一个元素，-2 指倒数第二个元素，以此类推。
end	可选。规定从何处结束选取。该参数是数组片断结束处的数组下标。如果没有指定该参数，那么切分的数组包含从 start 到数组结束的所有元素。如果这个参数是负数，那么它规定的是从数组尾部开始算起的元素。

返回值

返回一个新的数组，包含从 start 到 end （不包括该元素）的 arrayObject 中的元素。

说明

请注意，该方法并不会修改数组，而是返回一个子数组。如果想删除数组中的一段元素，应该使用方法 Array.splice()。

提示和注释

注释：您可使用负值从数组的尾部选取元素。
注释：如果 end 未被规定，那么 slice() 方法会选取从 start 到数组结尾的所有元素。

实例

例子 1

在本例中，我们将创建一个新数组，然后显示从其中选取的元素：

```
<script type="text/javascript">

var arr = new Array(3)
arr[0] = "George"
arr[1] = "John"
arr[2] = "Thomas"

document.write(arr + "<br />")
```

```
document.write(arr.slice(1) + "<br />")
document.write(arr)

</script>
```

输出：

```
George,John,Thomas
John,Thomas
George,John,Thomas
```

例子 2

在本例中，我们将创建一个新数组，然后显示从其中选取的元素：

```
<script type="text/javascript">

var arr = new Array(6)
arr[0] = "George"
arr[1] = "John"
arr[2] = "Thomas"
arr[3] = "James"
arr[4] = "Adrew"
arr[5] = "Martin"

document.write(arr + "<br />")
document.write(arr.slice(2,4) + "<br />")
document.write(arr)

</script>
```

输出：

```
George,John,Thomas,James,Adrew,Martin
Thomas,James
George,John,Thomas,James,Adrew,Martin
```

JavaScript sort() 方法

定义和用法

sort() 方法用于对数组的元素进行排序。

语法

```
arrayObject.sort(sortby)
```

参数	描述
sortby	可选。规定排序顺序。必须是函数。

返回值

对数组的引用。请注意，数组在原数组上进行排序，不生成副本。

说明

如果调用该方法时没有使用参数，将按字母顺序对数组中的元素进行排序，说得更精确点，是按照字符编码的顺序进行排序。要实现这一点，首先应把数组的元素都转换成字符串（如有必要），以便进行比较。
如果想按照其他标准进行排序，就需要提供比较函数，该函数要比较两个值，然后返回一个用于说明这两个值的相对顺序的数字。比较函数应该具有两个参数 a 和 b，其返回值如下：

- 若 a 小于 b，在排序后的数组中 a 应该出现在 b 之前，则返回一个小于 0 的值。
- 若 a 等于 b，则返回 0。
- 若 a 大于 b，则返回一个大于 0 的值。

实例

例子 1

在本例中，我们将创建一个数组，并按字母顺序进行排序：

```
<script type="text/javascript">

var arr = new Array(6)
```

```
arr[0] = "George"
arr[1] = "John"
arr[2] = "Thomas"
arr[3] = "James"
arr[4] = "Adrew"
arr[5] = "Martin"

document.write(arr + "<br />")
document.write(arr.sort())

</script>
```

输出：

```
George,John,Thomas,James,Adrew,Martin
Adrew,George,James,John,Martin,Thomas
```

例子 2

在本例中，我们将创建一个数组，并按字母顺序进行排序：

```
<script type="text/javascript">

var arr = new Array(6)
arr[0] = "10"
arr[1] = "5"
arr[2] = "40"
arr[3] = "25"
arr[4] = "1000"
arr[5] = "1"

document.write(arr + "<br />")
document.write(arr.sort())

</script>
```

输出：

```
10,5,40,25,1000,1
1,10,1000,25,40,5
```

请注意，上面的代码没有按照数值的大小对数字进行排序，要实现这一点，就必须使用一个排序函数：

```
<script type="text/javascript">

function sortNumber(a,b)
{
return a - b
}

var arr = new Array(6)
arr[0] = "10"
arr[1] = "5"
arr[2] = "40"
arr[3] = "25"
arr[4] = "1000"
arr[5] = "1"

document.write(arr + "<br />")
document.write(arr.sort(sortNumber))

</script>
```

输出：

```
10,5,40,25,1000,1
1,5,10,25,40,1000
```

JavaScript splice() 方法

定义和用法

splice() 方法向/从数组中添加/删除项目，然后返回被删除的项目。
注释：该方法会改变原始数组。

语法

```
arrayObject.splice(index,howmany,item1,.....,itemX)
```

参数	描述
index	必需。整数，规定添加/删除项目的位置，使用负数可从数组结尾处规定位置。

howmany	必需。要删除的项目数量。如果设置为 0，则不会删除项目。
item1, ..., itemX	可选。向数组添加的新项目。

返回值

类型	描述
Array	包含被删除项目的新数组，如果有的话。

说明

splice() 方法可删除从 index 处开始的零个或多个元素，并且用参数列表中声明的一个或多个值来替换那些被删除的元素。
如果从 arrayObject 中删除了元素，则返回的是含有被删除的元素的数组。

技术细节

JavaScript 版本：	1.2

浏览器支持

所有主流浏览器都支持 splice() 方法。

提示和注释

注释：请注意，splice() 方法与 slice() 方法的作用是不同的，splice() 方法会直接对数组进行修改。

实例

例子 1

在本例中，我们将创建一个新数组，并向其添加一个元素：

```
<script type="text/javascript">

var arr = new Array(6)
arr[0] = "George"
```

```
arr[1] = "John"
arr[2] = "Thomas"
arr[3] = "James"
arr[4] = "Adrew"
arr[5] = "Martin"

document.write(arr + "<br />")
arr.splice(2,0,"William")
document.write(arr + "<br />")

</script>
```

输出：

```
George,John,Thomas,James,Adrew,Martin
George,John,William,Thomas,James,Adrew,Martin
```

例子 2

在本例中我们将删除位于 index 2 的元素，并添加一个新元素来替代被删除的元素：

```
<script type="text/javascript">

var arr = new Array(6)
arr[0] = "George"
arr[1] = "John"
arr[2] = "Thomas"
arr[3] = "James"
arr[4] = "Adrew"
arr[5] = "Martin"

document.write(arr + "<br />")
arr.splice(2,1,"William")
document.write(arr)

</script>
```

输出：

```
George,John,Thomas,James,Adrew,Martin
George,John,William,James,Adrew,Martin
```

例子 3

在本例中我们将删除从 index 2 ("Thomas") 开始的三个元素，并添加一个新元素 ("William") 来替代被删除的元素：

```
<script type="text/javascript">

var arr = new Array(6)
arr[0] = "George"
arr[1] = "John"
arr[2] = "Thomas"
arr[3] = "James"
arr[4] = "Adrew"
arr[5] = "Martin"

document.write(arr + "<br />")
arr.splice(2,3,"William")
document.write(arr)

</script>
```

输出：

```
George,John,Thomas,James,Adrew,Martin
George,John,William,Martin
```

JavaScript toSource() 方法

定义和用法

toSource() 方法表示对象的源代码。
该原始值由 Array 对象派生的所有对象继承。
toSource() 方法通常由 JavaScript 在后台自动调用，并不显式地出现在代码中。

语法

```
object.toSource()
```

浏览器支持

只有 Gecko 核心的浏览器（比如 Firefox）支持该方法，也就是说 IE、Safari、Chrome、Opera

等浏览器均不支持该方法。

实例

下面的例子向您展示 toSource() 方法的用法：

```
<script type="text/javascript">

function employee(name,job,born)
{
this.name=name;
this.job=job;
this.born=born;
}

var bill=new employee("Bill Gates","Engineer",1985);

document.write(bill.toSource());

</script>
```

输出：

```
({name:"Bill Gates", job:"Engineer", born:1985})
```

JavaScript toString() 方法

定义和用法

toString() 方法可把数组转换为字符串，并返回结果。

语法

```
arrayObject.toString()
```

返回值

arrayObject 的字符串表示。返回值与没有参数的 join() 方法返回的字符串相同。

说明

当数组用于字符串环境时，JavaScript 会调用这一方法将数组自动转换成字符串。但是在某些情况下，需要显式地调用该方法。

提示和注释

注释：数组中的元素之间用逗号分隔。

实例

```
<script type="text/javascript">

var arr = new Array(3)
arr[0] = "George"
arr[1] = "John"
arr[2] = "Thomas"

document.write(arr.toString())

</script>
```

输出：

```
George,John,Thomas
```

JavaScript toLocaleString() 方法

定义和用法

把数组转换为本地字符串。

语法

```
arrayObject.toLocaleString()
```

返回值

arrayObject 的本地字符串表示。

说明

首先调用每个数组元素的 toLocaleString() 方法，然后使用地区特定的分隔符把生成的字符串连接起来，形成一个字符串。

实例

```
<script type="text/javascript">

var arr = new Array(3)
arr[0] = "George"
arr[1] = "John"
arr[2] = "Thomas"

document.write(arr.toLocaleString())

</script>
```

输出：

```
George, John, Thomas
```

JavaScript unshift() 方法

定义和用法

unshift() 方法可向数组的开头添加一个或更多元素，并返回新的长度。

语法

```
arrayObject.unshift(newelement1,newelement2,....,newelementX)
```

参数	描述

newelement1	必需。向数组添加的第一个元素。
newelement2	可选。向数组添加的第二个元素。
newelementX	可选。可添加若干个元素。

返回值

arrayObject 的新长度。

说明

unshift() 方法将把它的参数插入 arrayObject 的头部，并将已经存在的元素顺次地移到较高的下标处，以便留出空间。该方法的第一个参数将成为数组的新元素 0，如果还有第二个参数，它将成为新的元素 1，以此类推。
请注意，unshift() 方法不创建新的创建，而是直接修改原有的数组。

提示和注释

注释：该方法会改变数组的长度。
注释：unshift() 方法无法在 Internet Explorer 中正确地工作！
提示：要把一个或多个元素添加到数组的尾部，请使用 push() 方法。

实例

在本例中，我们将创建一个数组，并把一个元素添加到数组的开头，并返回数组的新长度：

```
<script type="text/javascript">

var arr = new Array()
arr[0] = "George"
arr[1] = "John"
arr[2] = "Thomas"

document.write(arr + "<br />")
document.write(arr.unshift("William") + "<br />")
document.write(arr)

</script>
```

输出：

```
George,John,Thomas
4
William,George,John,Thomas
```

JavaScript valueOf() 方法

定义和用法

valueOf() 方法返回 Array 对象的原始值。
该原始值由 Array 对象派生的所有对象继承。
valueOf() 方法通常由 JavaScript 在后台自动调用，并不显式地出现在代码中。

语法

```
arrayObject.valueOf()
```

JavaScript Boolean 对象参考手册

Boolean 对象

Boolean 对象表示两个值："true" 或 "false"。

创建 Boolean 对象的语法：

```
new Boolean(value);  //构造函数
Boolean(value);      //转换函数
```

参数

参数 *value* 由布尔对象存放的值或者要转换成布尔值的值。

返回值

当作为一个构造函数（带有运算符 new）调用时，Boolean() 将把它的参数转换成一个布尔值，并且返回一个包含该值的 Boolean 对象。

如果作为一个函数（不带有运算符 new）调用时，Boolean() 只将把它的参数转换成一个原始的布尔值，并且返回这个值。
注释：如果省略 value 参数，或者设置为 0、-0、null、""、false、undefined 或 NaN，则该对象设置为 false。否则设置为 true（即使 value 参数是字符串 "false"）。

Boolean 对象属性

FF: Firefox, IE: Internet Explorer

属性	描述	FF	IE
constructor	返回对创建此对象的 Boolean 函数的引用	1	4
prototype	使您有能力向对象添加属性和方法。	1	4

Boolean 对象方法

FF: Firefox, IE: Internet Explorer

方法	描述	FF	IE
toSource()	返回该对象的源代码。	1	-
toString()	把逻辑值转换为字符串，并返回结果。	1	4
valueOf()	返回 Boolean 对象的原始值。	1	4

Boolean 对象描述

在 JavaScript 中，布尔值是一种基本的数据类型。Boolean 对象是一个将布尔值打包的布尔对象。Boolean 对象主要用于提供将布尔值转换成字符串的 toString() 方法。
当调用 toString() 方法将布尔值转换成字符串时（通常是由 JavaScript 隐式地调用），JavaScript 会内在地将这个布尔值转换成一个临时的 Boolean 对象，然后调用这个对象的 toString() 方法。

课外书

如需更多信息，请阅读 JavaScript 高级教程中的相关内容：

ECMAScript 引用类型
引用类型通常叫做类（class）或对象。本节讲解 ECMAScript 的预定义引用类型。

JavaScript constructor 属性

定义和用法

constructor 属性返回对创建此对象的 Boolean 函数的引用。

语法

```
object.constructor
```

实例

在本例中，我们将展示如何使用 constructor 属性：

```
<script type="text/javascript">

var test=new Boolean();

if (test.constructor==Array)
{
document.write("This is an Array");
}
if (test.constructor==Boolean)
{
document.write("This is a Boolean");
}
if (test.constructor==Date)
{
document.write("This is a Date");
}
if (test.constructor==String)
{
document.write("This is a String");
}

</script>
```

输出：

```
This is a Boolean
```

JavaScript prototype 属性

定义和用法

prototype 属性使您有能力向对象添加属性和方法。

语法

```
object.prototype.name=value
```

实例

在本例中，我们将展示如何使用 prototype 属性来向对象添加属性：

```
<script type="text/javascript">

function employee(name,job,born)
{
this.name=name;
this.job=job;
this.born=born;
}

var bill=new employee("Bill Gates","Engineer",1985);

employee.prototype.salary=null;
bill.salary=20000;

document.write(bill.salary);

</script>
```

输出：

```
20000
```

JavaScript toSource() 方法

定义和用法

toSource() 方法返回表示对象源代码的字符串。

语法

```
object.toSource()
```

提示和注释

注释：该方法在 Internet Explorer 中无效。

实例

下面的例子向您展示 toSource() 方法的用法：

```
<script type="text/javascript">

function employee(name,job,born)
{
this.name=name;
this.job=job;
this.born=born;
}

var bill=new employee("Bill Gates","Engineer",1985);

document.write(bill.toSource());

</script>
```

输出：

```
({name:"Bill Gates", job:"Engineer", born:1985})
```

JavaScript toString() 方法

定义和用法

toString() 方法可把一个逻辑值转换为字符串，并返回结果。

语法

```
booleanObject.toString()
```

返回值

根据原始布尔值或者 booleanObject 对象的值返回字符串 "true" 或 "false"。

抛出

如果调用该方法的对象不是 Boolean，则抛出异常 TypeError。

提示和注释

注释：在 Boolean 对象被用于字符串环境中时，此方法会被自动调用。

实例

在本例中，我们将创建一个 Boolean 对象，并把它转换成字符串：

```
<script type="text/javascript">

 var boo = new Boolean(true)
document.write(boo.toString())

</script>
```

输出：

```
true
```

JavaScript valueOf() 方法

定义和用法

valueOf() 方法可返回 Boolean 对象的原始值。

语法

```
booleanObject.valueOf()
```

返回值

booleanObject 的原始布尔值。

抛出

如果调用该方法的对象不是 Boolean，则抛出异常 TypeError。

实例

在本例中，我们将创建一个 Boolean 对象，并使用 valueOf() 来取得此对象的原始值：

```
<script type="text/javascript">

var boo = new Boolean(false)
document.write(boo.valueOf())

</script>
```

输出：

```
false
```

JavaScript Date 对象参考手册

Date 对象

Date 对象用于处理日期和时间。

创建 Date 对象的语法：

```
var myDate=new Date()
```

注释：Date 对象会自动把当前日期和时间保存为其初始值。

Date 对象属性

FF: Firefox, IE: Internet Explorer

属性	描述	FF	IE
constructor	返回对创建此对象的 Date 函数的引用。	1	4
prototype	使您有能力向对象添加属性和方法。	1	4

Date 对象方法

FF: Firefox, IE: Internet Explorer

方法	描述	FF	IE
Date()	返回当日的日期和时间。	1	3
getDate()	从 Date 对象返回一个月中的某一天 (1 ~ 31)。	1	3
getDay()	从 Date 对象返回一周中的某一天 (0 ~ 6)。	1	3
getMonth()	从 Date 对象返回月份 (0 ~ 11)。	1	3
getFullYear()	从 Date 对象以四位数字返回年份。	1	4
getYear()	请使用 getFullYear() 方法代替。	1	3
getHours()	返回 Date 对象的小时 (0 ~ 23)。	1	3
getMinutes()	返回 Date 对象的分钟 (0 ~ 59)。	1	3

getSeconds()	返回 Date 对象的秒数 (0 ~ 59)。	1	3
getMilliseconds()	返回 Date 对象的毫秒(0 ~ 999)。	1	4
getTime()	返回 1970 年 1 月 1 日至今的毫秒数。	1	3
getTimezoneOffset()	返回本地时间与格林威治标准时间 (GMT) 的分钟差。	1	3
getUTCDate()	根据世界时从 Date 对象返回月中的一天 (1 ~ 31)。	1	4
getUTCDay()	根据世界时从 Date 对象返回周中的一天 (0 ~ 6)。	1	4
getUTCMonth()	根据世界时从 Date 对象返回月份 (0 ~ 11)。	1	4
getUTCFullYear()	根据世界时从 Date 对象返回四位数的年份。	1	4
getUTCHours()	根据世界时返回 Date 对象的小时 (0 ~ 23)。	1	4
getUTCMinutes()	根据世界时返回 Date 对象的分钟 (0 ~ 59)。	1	4
getUTCSeconds()	根据世界时返回 Date 对象的秒钟 (0 ~ 59)。	1	4
getUTCMilliseconds()	根据世界时返回 Date 对象的毫秒(0 ~ 999)。	1	4
parse()	返回 1970 年 1 月 1 日午夜到指定日期（字符串）的毫秒数。	1	3
setDate()	设置 Date 对象中月的某一天 (1 ~ 31)。	1	3
setMonth()	设置 Date 对象中月份 (0 ~ 11)。	1	3
setFullYear()	设置 Date 对象中的年份（四位数字）。	1	4
setYear()	请使用 setFullYear() 方法代替。	1	3
setHours()	设置 Date 对象中的小时 (0 ~ 23)。	1	3
setMinutes()	设置 Date 对象中的分钟 (0 ~ 59)。	1	3
setSeconds()	设置 Date 对象中的秒钟 (0 ~ 59)。	1	3
setMilliseconds()	设置 Date 对象中的毫秒 (0 ~ 999)。	1	4
setTime()	以毫秒设置 Date 对象。	1	3
setUTCDate()	根据世界时设置 Date 对象中月份的一天 (1 ~ 31)。	1	4
setUTCMonth()	根据世界时设置 Date 对象中的月份 (0 ~ 11)。	1	4
setUTCFullYear()	根据世界时设置 Date 对象中的年份（四位数字）。	1	4
setUTCHours()	根据世界时设置 Date 对象中的小时 (0 ~ 23)。	1	4
setUTCMinutes()	根据世界时设置 Date 对象中的分钟 (0 ~ 59)。	1	4
setUTCSeconds()	根据世界时设置 Date 对象中的秒钟 (0 ~ 59)。	1	4

setUTCMilliseconds()	根据世界时设置 Date 对象中的毫秒 (0 ~ 999)。	1	4
toSource()	返回该对象的源代码。	1	-
toString()	把 Date 对象转换为字符串。	1	4
toTimeString()	把 Date 对象的时间部分转换为字符串。	1	4
toDateString()	把 Date 对象的日期部分转换为字符串。	1	4
toGMTString()	请使用 toUTCString() 方法代替。	1	3
toUTCString()	根据世界时，把 Date 对象转换为字符串。	1	4
toLocaleString()	根据本地时间格式，把 Date 对象转换为字符串。	1	3
toLocaleTimeString()	根据本地时间格式，把 Date 对象的时间部分转换为字符串。	1	3
toLocaleDateString()	根据本地时间格式，把 Date 对象的日期部分转换为字符串。	1	3
UTC()	根据世界时返回 1970 年 1 月 1 日 到指定日期的毫秒数。	1	3
valueOf()	返回 Date 对象的原始值。	1	4

JavaScript constructor 属性

定义和用法

constructor 属性返回对创建此对象的 Date 函数的引用。

语法

```
object.constructor
```

实例

在本例中，我们将展示如何使用 constructor 属性：

```
<script type="text/javascript">

var test=new Date();
```

```
if (test.constructor==Array)
{
document.write("This is an Array");
}
if (test.constructor==Boolean)
{
document.write("This is a Boolean");
}
if (test.constructor==Date)
{
document.write("This is a Date");
}
if (test.constructor==String)
{
document.write("This is a String");
}

</script>
```

输出：

```
This is a Date
```

JavaScript prototype 属性

定义和用法

prototype 属性使您有能力向对象添加属性和方法。

语法

```
object.prototype.name=value
```

实例

在本例中，我们将展示如何使用 prototype 属性来向对象添加属性：

```
<script type="text/javascript">
```

```
function employee(name,job,born)
{
this.name=name;
this.job=job;
this.born=born;
}

var bill=new employee("Bill Gates","Engineer",1985);

employee.prototype.salary=null;
bill.salary=20000;

document.write(bill.salary);

</script>
```

输出：

```
20000
```

JavaScript Date() 方法

定义和用法

Date() 方法可返回当天的日期和时间。

语法

```
Date()
```

实例

在本例中，我们将输出今天的日期和时间：

```
<script type="text/javascript">

document.write(Date())
```

```
</script>
```

输出：

```
Thu Oct 10 2013 19:40:14 GMT+0800 (中国标准时间)
```

JavaScript Date() 方法

定义和用法

getDate() 方法可返回月份的某一天。

语法

```
dateObject.getDate()
```

返回值

dateObject 所指的月份中的某一天，使用本地时间。返回值是 1 ~ 31 之间的一个整数。

提示和注释：

注释：该方法总是结合一个 Date 对象来使用。

实例

例子 1

在本例中，我们将输出当前月份的日期：

```
<script type="text/javascript">

var d = new Date()
document.write(d.getDate())

</script>
```

输出：

```
10
```

例子 2

在本例中，我们定义了一个带有具体日期的变量，并输出变量中月份的天：

```
<script type="text/javascript">

var birthday = new Date("July 21, 1983 01:15:00")
document.write(birthday.getDate())

</script>
```

输出：

```
21
```

JavaScript getDay() 方法

定义和用法

getDay() 方法可返回表示星期的某一天的数字。

语法

```
dateObject.getDay()
```

返回值

dateObject 所指的星期中的某一天，使用本地时间。返回值是 0（周日） 到 6（周六） 之间的一个整数。

提示和注释：

注释：该方法总是结合一个 Date 对象来使用。

实例

例子 1

在本例中，我们将以数字取得星期的当前一天：

```
<script type="text/javascript">

var d=new Date()
document.write(d.getDay())

</script>
```

输出：

```
4
```

例子 2

现在，我们将创建一个数组，这样就可以使上面的例子输出星期的名称，而不是数字：

```
<script type="text/javascript">

var d=new Date()

var weekday=new Array(7)
weekday[0]="Sunday"
weekday[1]="Monday"
weekday[2]="Tuesday"
weekday[3]="Wednesday"
weekday[4]="Thursday"
weekday[5]="Friday"
weekday[6]="Saturday"

document.write("Today it is " + weekday[d.getDay()])

</script>
```

输出：

```
Today it is Thursday
```

JavaScript getMonth() 方法

定义和用法

getMonth() 方法可返回表示月份的数字。

语法

```
dateObject.getMonth()
```

返回值

dateObject 的月份字段，使用本地时间。返回值是 0（一月） 到 11（十二月） 之间的一个整数。

提示和注释：

注释：该方法总是结合一个 Date 对象来使用。

实例

例子 1

在本例中，我们将取得当前的日期，并把它输出：

```
<script type="text/javascript">

var d=new Date()

document.write(d.getMonth())

</script>
```

输出：

```
9
```

例子 2

现在，我们将创建一个数组，以输出月份的名称，而不是一个数字：

```
<script type="text/javascript">

var d=new Date()

var month=new Array(12)
month[0]="January"
month[1]="February"
month[2]="March"
month[3]="April"
month[4]="May"
month[5]="June"
month[6]="July"
month[7]="August"
month[8]="September"
month[9]="October"
month[10]="November"
month[11]="December"

document.write("The month is " + month[d.getMonth()])

</script>
```

输出：

```
The month is October
```

JavaScript getFullYear() 方法

定义和用法

getFullYear() 方法可返回一个表示年份的 4 位数字。

语法

```
dateObject.getFullYear()
```

返回值

当 dateObject 用本地时间表示时返回的年份。返回值是一个四位数，表示包括世纪值在内的完整年份，而不是两位数的缩写形式。

提示和注释：

注释：该方法总是结合一个 Date 对象来使用。

实例

例子 1

在本例中，我们将取得当前的年份，并输出它：

```
<script type="text/javascript">

var d = new Date()
document.write(d.getFullYear())

</script>
```

输出：

```
2013
```

例子 2

在本例中，我们将从具体到日期提取年份：

```
<script type="text/javascript">

var born = new Date("July 21, 1983 01:15:00")
document.write("I was born in " + born.getFullYear())

</script>
```

输出：

```
I was born in 1983
```

JavaScript getYear() 方法

定义和用法

getYear() 方法可返回表示年份的两位或四位的数字。

语法

```
dateObject.getYear()
```

返回值

返回 Date 对象的年份字段。

提示和注释：

注释：由 getYear() 返回的值不总是 4 位的数字！对于介于 1900 与 1999 之间的年份，getYear() 方法仅返回两位数字。对于 1900 之前或 1999 之后的年份，则返回 4 位数字！

注释：该方法总是结合一个 Date 对象来使用。

重要事项：从 ECMAScript v3 开始，JavaScript 的实现就不再使用该方法，而使用 getFullYear() 方法取而代之！

实例

例子 1

在本例中，我们将取得当前的年份，并输出它：

```
<script type="text/javascript">

var d = new Date()
document.write(d.getYear())

</script>
```

输出：

```
113
```

例子 2

在本例中，我们将从具体的日期中提取年份：

```
<script type="text/javascript">

var born = new Date("July 21, 1983 01:15:00")
document.write("I was born in " + born.getYear())

</script>
```

输出：

```
I was born in 83
```

JavaScript getHours() 方法

定义和用法

getHours() 方法可返回时间的小时字段。

语法

```
dateObject.getHours()
```

返回值

dateObject 的小时字段，以本地时间显示。返回值是 0 （午夜） 到 23 （晚上 11 点）之间的一个整数。

提示和注释：

注释：由 getHours() 返回的值是一个两位的数字。不过返回值不总是两位的，如果该值小于 10，则仅返回一位数字。
注释：该方法总是结合一个 Date 对象来使用。

实例

例子 1

在本例中，我们可取得当前时间的小时：

```
<script type="text/javascript">

var d = new Date()
document.write(d.getHours())

</script>
```

输出：

```
19
```

例子 2

在本例中，我们将从具体的日期和时间中提取出小时字段：

```
<script type="text/javascript">

var born = new Date("July 21, 1983 01:15:00")
document.write(born.getHours())

</script>
```

输出：

```
1
```

JavaScript getMinutes() 方法

定义和用法

getMinutes() 方法可返回时间的分钟字段。

语法

```
dateObject.getMinutes()
```

返回值

dateObject 的分钟字段，以本地时间显示。返回值是 0 ~ 59 之间的一个整数。

提示和注释：

注释：由 getMinutes() 返回的值是一个两位的数字。不过返回值不总是两位的，如果该值小于 10，则仅返回一位数字。
注释：该方法总是结合一个 Date 对象来使用。

实例

例子 1

在本例中，我们可取得当前时间的分钟：

```
<script type="text/javascript">

var d = new Date()
document.write(d.getMinutes())

</script>
```

输出：

```
50
```

例子 2

在本例中，我们将从具体的日期和时间中提取出分钟字段：

```
<script type="text/javascript">

var born = new Date("July 21, 1983 01:15:00")
document.write(born.getMinutes())

</script>
```

输出：

```
15
```

JavaScript getSeconds() 方法

定义和用法

getSeconds() 方法可返回时间的秒。

语法

```
dateObject.getSeconds()
```

返回值

dateObject 的分钟字段，以本地时间显示。返回值是 0 ~ 59 之间的一个整数。

提示和注释：

注释：由 getSeconds() 返回的值是一个两位的数字。不过返回值不总是两位的，如果该值小于 10，则仅返回一位数字。
注释：该方法总是结合一个 Date 对象来使用。

实例

例子 1

在本例中，我们可取得当前时间的秒：

```
<script type="text/javascript">

var d = new Date()
document.write(d.getSeconds())

</script>
```

输出：

```
8
```

例子 2

在本例中，我们将从具体的日期和时间中提取出秒字段：

```
<script type="text/javascript">

var Birthday = new Date("July 21, 1983 01:15:00")
document.write(Birthday.getSeconds())

</script>
```

输出：

```
0
```

JavaScript getMilliseconds() 方法

定义和用法

getMilliseconds() 方法可返回时间的毫秒。

语法

```
dateObject.getMilliseconds()
```

返回值

dateObject 的毫秒字段，以本地时间显示。返回值是 0 ~ 999 之间的一个整数。

提示和注释：

注释：由 getMilliseconds() 返回的值是一个三位的数字。不过返回值不总是三位的，如果该值小于 100，则仅返回两位数字，如果该值小于 10，则仅返回一位数字。
注释：该方法总是结合一个 Date 对象来使用。

实例

例子 1

在本例中，我们可取得当前时间的毫秒：

```
<script type="text/javascript">

var d = new Date()
document.write(d.getMilliseconds())

</script>
```

输出：

```
120
```

例子 2

在本例中，我们将从具体的日期和时间中提取出毫秒字段：

```
<script type="text/javascript">

var born = new Date("July 21, 1983 01:15:00")
document.write(born.getMilliseconds())

</script>
```

输出：

```
0
```

JavaScript getTime() 方法

定义和用法

getTime() 方法可返回距 1970 年 1 月 1 日之间的毫秒数。

语法

```
dateObject.getTime()
```

返回值

dateObject 指定的日期和时间距 1970 年 1 月 1 日午夜（GMT 时间）之间的毫秒数。

提示和注释：

注释：该方法总是结合一个 Date 对象来使用。

实例

例子 1

在本例中，我们将取得从 1970/01/01 至今的毫秒数，并输出它：

```
<script type="text/javascript">

var d = new Date()
document.write(d.getTime() + " milliseconds since 1970/01/01")

</script>
```

输出：

```
1381406194654 milliseconds since 1970/01/01
```

例子 2

在下面的例子中，我们将计算出从 1970/01/01 至今有多少年：

```
<script type="text/javascript">

var minutes = 1000*60
var hours = minutes*60
var days = hours*24
var years = days*365
var d = new Date()
var t = d.getTime()
var y = t/years
document.write("It's been: " + y + " years since 1970/01/01!")

</script>
```

输出：

```
It's been: 43.804103077562154 years since 1970/01/01!
```

JavaScript getTimezoneOffset() 方法

定义和用法

getTimezoneOffset() 方法可返回格林威治时间和本地时间之间的时差，以分钟为单位。

语法

```
dateObject.getTimezoneOffset()
```

返回值

本地时间与 GMT 时间之间的时间差，以分钟为单位。

说明

getTimezoneOffset() 方法返回的是本地时间与 GMT 时间或 UTC 时间之间相差的分钟数。
实际上，该函数告诉我们运行 JavaScript 代码的时区，以及指定的时间是否是夏令时。
返回之所以以分钟计，而不是以小时计，原因是某些国家所占有的时区甚至不到一个小时的间隔。

提示和注释：

注释：由于使用夏令时的惯例，该方法的返回值不是一个常量。
注释：该方法总是结合一个 Date 对象来使用。

实例

例子 1

在下面的例子中，我们将取得 GMT 时间与本地时间以分钟计的时间差：

```
<script type="text/javascript">
```

```
var d = new Date()
document.write(d.getTimezoneOffset())

</script>
```

输出：

```
-480
```

例子 2

现在，我们将把上面的例子转换为 GMT +/- 小时：

```
<script type="text/javascript">

var d = new Date()
var gmtHours = d.getTimezoneOffset()/60
document.write("The local time zone is: GMT " + gmtHours)

</script>
```

输出：

```
The local time zone is: GMT -8
```

JavaScript getUTCDate() 方法

定义和用法

getUTCDate() 方法可根据世界时返回一个月 (UTC) 中的某一天。

语法

```
dateObject.getUTCDate()
```

返回值

dateObject 用世界时表示时，返回该月中的某一天 （是 1 ~ 31 中的一个值）。

提示和注释：

注释：该方法总是结合一个 Date 对象来使用。
提示：有关通用协调时间 (UTC) 的更多资料，请参阅百度百科。

实例

例子 1

在本例中，我们将根据 UTC 来输出此约的当前一天：

```
<script type="text/javascript">

var d = new Date()
document.write(d.getUTCDate())

</script>
```

输出：

```
10
```

例子 2

我们在此处定义了一个带有具体日期的变量，然后根据 UTC 输出变量中的月中的那一天：

```
<script type="text/javascript">

var birthday = new Date("July 21, 1983 01:15:00")
document.write(birthday.getUTCDate())

</script>
```

输出：

```
21
```

JavaScript getUTCDay() 方法

定义和用法

getUTCDay() 方法根据世界时返回表示星期的一天的一个数字。

语法

```
dateObject.getUTCDay()
```

返回值

dateObject 用世界时表示时，返回该星期中的某一天 ，该值是 0（星期天） ~ 6（星期六）中的一个值。

提示和注释：

注释：该方法总是结合一个 Date 对象来使用。
提示：有关通用协调时间 (UTC) 的更多资料，请参阅百度百科。

实例

例子 1

在本例中，我们将取得本周的当前 UTC 天（以一个数字）：

```
<script type="text/javascript">

var d=new Date()
document.write(d.getUTCDay())

</script>
```

输出：

```
4
```

例子 2

现在，我将创建一个数组，使得上面的例子能够输出星期的名称，而不仅仅是一个数字：

```
<script type="text/javascript">

var d=new Date()

var weekday=new Array(7)
weekday[0]="Sunday"
weekday[1]="Monday"
weekday[2]="Tuesday"
weekday[3]="Wednesday"
weekday[4]="Thursday"
weekday[5]="Friday"
weekday[6]="Saturday"

document.write("Today it is " + weekday[d.getUTCDay()])

</script>
```

输出：

```
Today it is Thursday
```

JavaScript getUTCMonth() 方法

定义和用法

getUTCMonth() 方法可返回一个表示月份的数字（按照世界时 UTC）。

语法

```
dateObject.getUTCMonth()
```

返回值

返回 dateObject 用世界时表示时的月份，该值是 0（一月） ~ 11（十二月） 之间中的一个整数。
需要注意的是，Date 对象使用 1 来表示月的第一天，而不是像月份字段那样使用 0 来代

表一年的第一个月。

提示和注释：

注释：该方法总是结合一个 Date 对象来使用。
提示：有关通用协调时间 (UTC) 的更多资料，请参阅百度百科。

实例

例子 1

在本例中，我们将取得当前月份，然后输出它：

```
<script type="text/javascript">

var d=new Date()

document.write(d.getUTCMonth())

</script>
```

输出：

```
9
```

例子 2

现在，我们将创建一个数组，使得我们上面的例子能够输出月份的名称，而不仅仅是一个数字：

```
<script type="text/javascript">

var d=new Date()

var month=new Array(12)
month[0]="January"
month[1]="February"
month[2]="March"
month[3]="April"
month[4]="May"
month[5]="June"
month[6]="July"
month[7]="August"
```

```
month[8]="September"
month[9]="October"
month[10]="November"
month[11]="December"

document.write("The month is " + month[d.getUTCMonth()])

</script>
```

输出：

```
The month is October
```

JavaScript getUTCFullYear() 方法

定义和用法

getUTCFullYear() 方法可返回根据世界时 (UTC) 表示的年份的四位数字。

语法

```
dateObject.getUTCFullYear()
```

返回值

返回 dateObject 用世界时表示时的年份，该值是一个四位的整数，而不是两位数的缩写。

提示和注释：

注释：该方法总是结合一个 Date 对象来使用。
提示：有关通用协调时间 (UTC) 的更多资料，请参阅百度百科。

实例

例子 1

In this example we get the current year and print it:

```
<script type="text/javascript">

var d = new Date()
document.write(d.getUTCFullYear())

</script>
```

输出：

```
2013
```

例子 2

Here we will extract the year out of the specific date:

```
<script type="text/javascript">

var born = new Date("July 21, 1983 01:15:00")
document.write("I was born in " + born.getUTCFullYear())

</script>
```

输出：

```
I was born in 1983
```

JavaScript getUTCHours() 方法

定义和用法

getUTCHours() 方法可根据世界时 (UTC) 返回时间的小时。

语法

```
dateObject.getUTCHours()
```

返回值

返回 dateObject 用世界时表示时的小时字段，该值是一个 0（午夜） ~ 23（晚上 11 点）

之间的整数。

提示和注释：

注释：由 getUTCHours() 返回的值是一个两位的数字。不过返回值不总是两位的，如果该值小于 10，则仅返回一位数字。
注释：该方法总是结合一个 Date 对象来使用。
提示：有关通用协调时间 (UTC) 的更多资料，请参阅百度百科。

实例

例子 1

在本例中，我们将取得当前时间的 UTC 小时：

```
<script type="text/javascript">

var d = new Date()
document.write(d.getUTCHours())

</script>
```

输出：

```
12
```

例子 2

在这里，我们将从具体的日期和时间中提取 UTC 小时：

```
<script type="text/javascript">

var born = new Date("July 21, 1983 01:15:00")
document.write(born.getUTCHours())

</script>
```

输出：

```
17
```

JavaScript getUTCMinutes() 方法

定义和用法

getUTCMinutes() 方法可根据世界时 (UTC) 返回时间的分钟字段。

语法

```
dateObject.getUTCMinutes()
```

返回值

返回 dateObject 用世界时表示时的分钟字段，该值是一个 0 ~ 59 之间的整数。

提示和注释：

注释：由 getUTCMinutes() 返回的值是一个两位的数字。不过返回值不总是两位的，如果该值小于 10，则仅返回一位数字。
注释：该方法总是结合一个 Date 对象来使用。
提示：有关通用协调时间 (UTC) 的更多资料，请参阅百度百科。

实例

例子 1

在本例中，我们将取得当前时间的 UTC 分钟：

```
<script type="text/javascript">

var d = new Date()
document.write(d.getUTCMinutes())

</script>
```

输出：

```
3
```

例子 2

在此处，我们将从具体的日期和时间中提取 UTC 分钟：

```
<script type="text/javascript">

var born = new Date("July 21, 1983 01:15:00")
document.write(born.getUTCMinutes())

</script>
```

输出：

```
15
```

JavaScript getUTCSeconds() 方法

定义和用法

getUTCSeconds() 方法可根据世界时返回时间的秒。

语法

```
dateObject.getUTCSeconds()
```

返回值

当 dateObject 用世界时表示时，返回它的秒字段，该值是一个 0 ~ 59 之间的整数。

提示和注释：

注释：由 getUTCSeconds() 返回的值是一个两位的数字。不过返回值不总是两位的，如果该值小于 10，则仅返回一位数字。
注释：该方法总是结合一个 Date 对象来使用。
提示：有关通用协调时间 (UTC) 的更多资料，请参阅百度百科。

实例

例子 1

在本例中，我们将取得当前时间的 UTC 秒：

```
<script type="text/javascript">

var d = new Date()
document.write(d.getUTCSeconds())

</script>
```

输出：

```
24
```

例子 2

在此处，我们将从具体的日期和时间中提取 UTC 秒：

```
<script type="text/javascript">

var Birthday = new Date("July 21, 1983 01:15:00")
document.write(Birthday.getUTCSeconds())

</script>
```

输出：

```
0
```

JavaScript getUTCMilliseconds() 方法

定义和用法

getUTCMilliseconds() 方法可根据世界时 (UTC) 返回时间的毫秒。

语法

```
dateObject.getUTCMilliseconds()
```

返回值

当 dateObject 用世界时表示时，返回它的毫秒字段，该值是一个 0 ~ 999 之间的整数。

提示和注释：

注释：由 getUTCMilliseconds() 返回的值是一个三位的数字。不过返回值不总是三位的，如果该值小于 100，则仅返回两位数字，如果该值小于 10，则仅返回一位数字。
注释：该方法总是结合一个 Date 对象来使用。
提示：有关通用协调时间 (UTC) 的更多资料，请参阅百度百科。

实例

例子 1

在本例中，我们将取得当前时间的 UTC 毫秒：

```
<script type="text/javascript">

var d = new Date()
document.write(d.getUTCMilliseconds())

</script>
```

输出：

```
546
```

例子 2

在此处，我们将从具体的日期和时间中提取 UTC 毫秒：

```
<script type="text/javascript">

var born = new Date("July 21, 1983 01:15:00")
document.write(born.getUTCMilliseconds())
```

```
</script>
```

输出：

```
0
```

注释：上面的代码将把毫秒设置为 1，这是由于在日期中没有定义毫秒。

JavaScript parse() 方法

定义和用法

parse() 方法可解析一个日期时间字符串，并返回 1970/1/1 午夜距离该日期时间的毫秒数。

语法

```
Date.parse(datestring)
```

参数	描述
datestring	必需。表示日期和时间的字符串。

返回值

指定的日期和时间据 1970/1/1 午夜（GMT 时间）之间的毫秒数。

说明

该方法是 Date 对象的静态方法。一般采用 Date.parse() 的形式来调用，而不是通过 dateobject.parse() 调用该方法。

提示和注释：

注释：Date.parse() 是 Date 对象的静态方法。

实例

例子 1

在本例中，我们将取得从 1970/01/01 到 2005/07/08 的毫秒数：

```
<script type="text/javascript">

var d = Date.parse("Jul 8, 2005")
document.write(d)

</script>
```

输出：

```
1120752000000
```

例子 2

现在，我们将把上面的例子所输出的结果转换成年：

```
<script type="text/javascript">

var minutes = 1000 * 60
var hours = minutes * 60
var days = hours * 24
var years = days * 365
var t = Date.parse("Jul 8, 2005")
var y = t/years
document.write("It's been: " + y + " years from 1970/01/01")
document.write(" to 2005/07/08!")

</script>
```

输出：

```
It's been: 35.538812785388124 years from 1970/01/01 to 2005/07/08!
```

JavaScript setDate() 方法

定义和用法

setDate() 方法用于设置一个月的某一天。

语法

```
dateObject.setDate(day)
```

参数	描述
day	必需。表示一个月中的一天的一个数值（1 ~ 31）。

返回值

调整过的日期的毫秒表示。在 ECMAScript 标准化之前，该方法什么都不返回。

提示和注释：

注释：该方法总是结合一个 Date 对象来使用。

实例

在本例中，我们将通过 setDate() 方法把当前月的天设置为 15：

```
<script type="text/javascript">

var d = new Date()
d.setDate(15)
document.write(d)

</script>
```

输出：

```
Tue Oct 15 2013 20:06:43 GMT+0800 (中国标准时间)
```

JavaScript setMonth() 方法

定义和用法

setMonth() 方法用于设置月份。

语法

```
dateObject.setMonth(month,day)
```

参数	描述
month	必需。一个表示月份的数值，该值介于 0（一月） ~ 11（十二月） 之间。
day	可选。一个表示月的某一天的数值，该值介于 1 ~ 31 之间（以本地时间计）。在 EMCAScript 标准化之前，不支持该参数。

返回值

调整过的日期的毫秒表示。在 ECMAScript 标准化之前，该方法什么都不返回。

提示和注释：

注释：该方法总是结合一个 Date 对象来使用。

实例

例子 1

在本例中，我们将通过 setMonth() 方法把对象 d 的月字段设置为 0 （一月）：

```
<script type="text/javascript">

var d=new Date()
d.setMonth(0)
document.write(d)

</script>
```

输出：

```
Thu Jan 10 2013 20:08:07 GMT+0800 (中国标准时间)
```

例子 2

在本例中，我们将通过 setMonth() 方法把对象 d 的月字段设置为 0 （一月），把天字段设置为 20：

```
<script type="text/javascript">

var d=new Date()
d.setMonth(0,20)
document.write(d)

</script>
```

输出：

```
Sun Jan 20 2013 20:08:07 GMT+0800 (中国标准时间)
```

JavaScript setFullYear() 方法

定义和用法

setFullYear() 方法用于设置年份。

语法

```
dateObject.setFullYear(year,month,day)
```

参数	描述
year	必需。表示年份的四位整数。用本地时间表示。
month	可选。表示月份的数值，介于 0 ~ 11 之间。用本地时间表示。
day	可选。表示月中某一天的数值，介于 1 ~ 31 之间。用本地时间表示。

返回值

返回调整过的日期的毫秒表示。

提示和注释：

注释：该方法总是结合一个 Date 对象来使用。

实例

例子 1

在本例中，我们将通过 setFullYear() 把年份设置为 1992：

```
<script type="text/javascript">

var d = new Date()
d.setFullYear(1992)
document.write(d)



</script>
```

输出：

```
Sat Oct 10 1992 20:09:00 GMT+0800 (中国标准时间)
```

例子 2

在本例中，我们将通过 setFullYear() 把日期设置为 November 3, 1992：

```
<script type="text/javascript">

var d = new Date()
d.setFullYear(1992,10,3)
document.write(d)

</script>
```

输出：

```
Tue Nov 03 1992 20:09:00 GMT+0800 (中国标准时间)
```

JavaScript setYear() 方法

定义和用法

setYear() 方法用于设置年份。

语法

```
dateObject.setYear(year)
```

参数	描述
year	必需。表示年份的两位或四位数字。

返回值

调整过的日期的毫秒表示。在 ECMAScript 标准化之前，该方法什么都不返回。

提示和注释：

注释：如果 year 参数是两位的数字，比如 setYear(91)，则该方法会理解为 1991。如果要规定 1990 年之前或 1999 年之后的年份，请使用四位数字。
注释：该方法总是结合一个 Date 对象来使用。
重要事项：从 ECMAScript v3 起，JavaScript 实现不再要求使用该函数，而使用 setFullYear() 函数代替它。

实例

在本例中，我们将通过 setYear() 方法把年份设置为 1891：

```
<script type="text/javascript">

var d = new Date()
d.setYear(1891)
document.write(d)
```

```
</script>
```

输出：

```
Sat Oct 10 1891 20:10:34 GMT+0800 (中国标准时间)
```

JavaScript setHours() 方法

定义和用法

setHours() 方法用于设置指定的时间的小时字段。

语法

```
dateObject.setHours(hour,min,sec,millisec)
```

参数	描述
hour	必需。表示小时的数值，介于 0（午夜） ~ 23（晚上 11 点） 之间，以本地时间计（下同）。
min	可选。表示分钟的数值，介于 0 ~ 59 之间。在 EMCAScript 标准化之前，不支持该参数。
sec	可选。表示秒的数值，介于 0 ~ 59 之间。在 EMCAScript 标准化之前，不支持该参数。
millisec	可选。表示毫秒的数值，介于 0 ~ 999 之间。在 EMCAScript 标准化之前，不支持该参数。

返回值

调整过的日期的毫秒表示。在 ECMAScript 标准化之前，该方法什么都不返回。

提示和注释：

注释：如果上面的参数之一使用一位的数字来规定，那么 JavaScript 会在结果中加一或两个前置 0。
注释：该方法总是结合一个 Date 对象来使用。

实例

例子 1

在本例中，我们将通过 setHours() 方法把当前时间的小时字段设置为 15：

```
<script type="text/javascript">

var d = new Date()
d.setHours(15)
document.write(d)

</script>
```

输出：

```
Thu Oct 10 2013 15:11:16 GMT+0800 (中国标准时间)
```

例子 2

在本例中，我们将通过 setHours() 方法把时间设置为 15:35:01：

```
<script type="text/javascript">

var d = new Date()
d.setHours(15,35,1)
document.write(d)

</script>
```

输出：

```
Thu Oct 10 2013 15:35:01 GMT+0800 (中国标准时间)
```

JavaScript setMinutes() 方法

定义和用法

setMinutes() 方法用于设置指定时间的分钟字段。

语法

```
dateObject.setMinutes(min,sec,millisec)
```

参数	描述
min	必需。表示分钟的数值，介于 0 ~ 59 之间，以本地时间计（下同）。
sec	可选。表示秒的数值，介于 0 ~ 59 之间。在 EMCAScript 标准化之前，不支持该参数。
millisec	可选。表示毫秒的数值，介于 0 ~ 999 之间。在 EMCAScript 标准化之前，不支持该参数。

返回值

调整过的日期的毫秒表示。在 ECMAScript 标准化之前，该方法什么都不返回。

提示和注释：

注释：如果上面的参数之一使用一位的数字来规定，那么 JavaScript 会在结果中加一或两个前置 0。
注释：该方法总是结合一个 Date 对象来使用。

实例

在本例中，我们将通过 setMinutes() 方法把当前时间的分钟字段设置为 01：

```
<script type="text/javascript">

var d = new Date()
d.setMinutes(1)
document.write(d)

</script>
```

输出：

```
Thu Oct 10 2013 20:01:05 GMT+0800 (中国标准时间)
```

JavaScript setSeconds() 方法

定义和用法

The setSeconds() method is used to set the seconds of a specified time.

语法

```
dateObject.setSeconds(sec,millisec)
```

参数	描述
sec	必需。表示秒的数值，该值是介于 0 ~ 59 之间的整数。
millisec	可选。表示毫秒的数值，介于 0 ~ 999 之间。在 EMCAScript 标准化之前，不支持该参数。

返回值

调整过的日期的毫秒表示。在 ECMAScript 标准化之前，该方法什么都不返回。

提示和注释：

注释：如果上面的参数之一使用一位的数字来规定，那么 JavaScript 会在结果中加一或两个前置 0。
注释：该方法总是结合一个 Date 对象来使用。

实例

在本例中，我们将通过 setSeconds() 方法把当前时间的秒字段设置为 01：

```
<script type="text/javascript">

var d = new Date()
d.setSeconds(1)
document.write(d)

</script>
```

输出：

```
Thu Oct 10 2013 20:15:01 GMT+0800 (中国标准时间)
```

JavaScript setMilliseconds() 方法

定义和用法

setMilliseconds() 方法用于设置指定时间的毫秒字段。

语法

```
dateObject.setMilliseconds(millisec)
```

参数	描述
millisec	必需。用于设置 dateObject 毫秒字段，该参数是介于 0 ~ 999 之间的整数。

返回值

调整过的日期的毫秒表示。

提示和注释：

注释：如果上面的参数之一使用一位的数字来规定，那么 JavaScript 会在结果中加一或两个前置 0。
注释：该方法总是结合一个 Date 对象来使用。

实例

在本例中，我们将通过 setMilliseconds() 方法把当前时间的毫秒字段设置为 001：

```
<script type="text/javascript">

var d = new Date()
d.setMilliseconds(1)
document.write(d)
```

```
</script>
```

输出：

```
Thu Oct 10 2013 20:16:10 GMT+0800 (中国标准时间)
```

JavaScript setTime() 方法

定义和用法

setTime() 方法以毫秒设置 Date 对象。

语法

```
dateObject.setTime(millisec)
```

参数	描述
millisec	必需。要设置的日期和时间据 GMT 时间 1970 年 1 月 1 日午夜之间的毫秒数。这种类型的毫秒值可以传递给 Date() 构造函数，可以通过调用 Date.UTC() 和 Date.parse() 方法获得该值。以毫秒形式表示日期可以使它独立于时区。

返回值

返回参数 *millisec*。在 ECMAScript 标准化之前，该方法不返回值。

提示和注释：

注释：该方法总是结合一个 Date 对象来使用。

实例

例子 1

在本例中，我们将向 1970/01/01 添加 77771564221 毫秒，并显示新的日期和时间：

```
<script type="text/javascript">
```

```
var d = new Date()
d.setTime(77771564221)
document.write(d)

</script>
```

输出：

```
Mon Jun 19 1972 11:12:44 GMT+0800 (中国标准时间)
```

例子 2

在本例中，我们将从 1970/01/01 减去 77771564221 毫秒，并显示新的日期和时间：

```
<script type="text/javascript">

var d = new Date()
d.setTime(-77771564221)
document.write(d)

</script>
```

输出：

```
Sun Jul 16 1967 04:47:15 GMT+0800 (中国标准时间)
```

JavaScript setUTCDate() 方法

定义和用法

setUTCDate() 方法用于根据世界时 (UTC) 设置一个月中的某一天。

语法

```
dateObject.setUTCDate(day)
```

参数	描述
day	必需。要给 dateObject 设置的一个月中的某一天，用世界时表示。该参数是 1 ~ 31

	之间的整数。

返回值

调整过的日期的毫秒表示。

提示和注释：

注释：该方法总是结合一个 Date 对象来使用。
提示：有关通用协调时间 (UTC) 的更多资料，请参阅百度百科。

实例

在本例中，我们将通过 setUTCDate() 方法把当前月的天字段设置为 15：

```
<script type="text/javascript">

var d = new Date()
d.setUTCDate(15)
document.write(d)

</script>
```

输出：

```
Tue Oct 15 2013 20:18:29 GMT+0800 (中国标准时间)
```

JavaScript setUTCMonth() 方法

定义和用法

setUTCMonth() 方法用于根据世界时 (UTC) 来设置月份。

语法

```
dateObject.setUTCMonth(month,day)
```

参数	描述
month	必需。要给 dateObject 设置的月份字段的值，用世界时表示。 该参数是 0（一月） ~ 11（十二月） 之间的整数。
day	可选。在 1 ~ 31 之间的整数，用作 dateObject 的天字段，用世界时表示。

返回值

调整过的日期的毫秒表示。

提示和注释：

注释：该方法总是结合一个 Date 对象来使用。
提示：有关通用协调时间 (UTC) 的更多资料，请参阅百度百科。

实例

例子 1

在本例中，我们将通过 setUTCMonth() 方法把月字段设置为 0 （一月）：

```
<script type="text/javascript">

var d=new Date()
d.setUTCMonth(0)
document.write(d)

</script>
```

输出：

```
Thu Jan 10 2013 20:19:20 GMT+0800 (中国标准时间)
```

例子 2

在本例中，我们将通过 setUTCMonth() 把月份设置为 0 （一月），把天字段设置为 20：

```
<script type="text/javascript">

var d=new Date()
d.setUTCMonth(0,20)
```

```
document.write(d)

</script>
```

输出：

```
Sun Jan 20 2013 20:19:20 GMT+0800 (中国标准时间)
```

JavaScript setUTCFullYear() 方法

定义和用法

setUTCFullYear() 方法用于根据世界时 (UTC) 设置年份。

语法

```
dateObject.setUTCFullYear(year,month,day)
```

参数	描述
year	必需。要给 dateObject 设置的年份字段的值。 该参数应该是含有世纪值的完整年份，如 1999，而不只是缩写的年份值，比如 99。
month	可选。要给 dateObject 设置的月份字段的值。使用世界时表示。 该参数是 0 ~ 11 之间的整数。
day	可选。要给 dateObject 设置的天字段的值。使用世界时表示。 该参数是 1 ~ 31 之间的整数。

返回值

调整过的日期的毫秒表示。

提示和注释：

注释：该方法总是结合一个 Date 对象来使用。
提示：有关通用协调时间 (UTC) 的更多资料，请参阅百度百科。

实例

例子 1

在本例中，我们将通过 setUTCFullYear() 方法把年份设置为 1992：

```
<script type="text/javascript">

var d = new Date()
d.setUTCFullYear(1992)
document.write(d)

</script>
```

输出：

```
Sat Oct 10 1992 20:20:28 GMT+0800 (中国标准时间)
```

例子 2

在本例中，我将通过 setUTCFullYear() 方法把日期设置为 November 3, 1992：

```
<script type="text/javascript">

var d = new Date()
d.setUTCFullYear(1992,10,3)
document.write(d)

</script>
```

输出：

```
Tue Nov 03 1992 20:20:28 GMT+0800 (中国标准时间)
```

JavaScript setUTCHours() 方法

定义和用法

setUTCHours() 方法用于根据世界时 (UTC) 设置小时（0 - 23）。

语法

```
dateObject.setUTCHours(hour,min,sec,millisec)
```

参数	描述
hour	必需。要给 dateObject 设置的小时字段的值。 该参数是 0 ~ 23 之间的整数。
min	可选。要给 dateObject 设置的分钟字段的值。 该参数是 0 ~ 59 之间的整数。
sec	可选。要给 dateObject 设置的秒字段的值。 该参数是 0 ~ 59 之间的整数。
millisec	可选。要给 dateObject 设置的毫秒字段的值。 该参数是 1 ~ 999 之间的整数。

说明

如果没有规定 *min*, *sec* 以及 *millisec* 参数，则使用从 getUTCMinutes, getUTCSeconds 以及 getUTCMilliseconds 方法返回的值。
如果您规定的参数在指定范围之外，则 setUTCHours 尝试据此来更新 Date 对象中的日期信息。例如，如果 *sec* 参数的值是 100，则增加 1 分钟 (*min* + 1)，而秒数为 40。

返回值

调整过的日期的毫秒表示。

提示和注释：

注释：该方法总是结合一个 Date 对象来使用。
提示：有关通用协调时间 (UTC) 的更多资料，请参阅百度百科。

实例

例子 1

在本例中，我们将通过 setUTCHours() 方法将 UTC 小时设置为 23：

```
<script type="text/javascript">
```

```
var d = new Date();
d.setUTCHours(23);
document.write(d);

</script>
```

输出：

```
Fri Oct 11 2013 07:21:22 GMT+0800 (中国标准时间)
```

例子 2

在本例中，我们将通过 setUTCHours() 方法将 UTC 小时设置为 23:15:06 ：

```
<script type="text/javascript">

var d = new Date();
d.setUTCHours(23,15,6);
document.write(d);

</script>
```

输出：

```
Fri Oct 11 2013 07:15:06 GMT+0800 (中国标准时间)
```

JavaScript setUTCMinutes() 方法

定义和用法

setUTCMinutes() 方法用于根据世界时 (UTC) 来设置指定时间的分钟。

语法

```
dateObject.setUTCMinutes(min,sec,millisec)
```

参数	描述
min	必需。要给 dateObject 设置的分钟字段的值，用世界时表示。

	该参数应该是 0 ~ 59 之间的整数。
sec	可选。要给 dateObject 设置的秒字段的值。使用世界时表示。 该参数是 0 ~ 59 之间的整数。
millisec	可选。要给 dateObject 设置的毫秒字段的值。使用世界时表示。 该参数是 0 ~ 999 之间的整数。

返回值

调整过的日期的毫秒表示。

提示和注释：

注释：如果上面的参数之一使用一位的数字来规定，那么 JavaScript 会在结果中加一或两个前置 0。
注释：该方法总是结合一个 Date 对象来使用。
提示：有关通用协调时间 (UTC) 的更多资料，请参阅百度百科。

实例

在本例中，我们将通过 setUTCMinutes() 方法把当前时间的分钟字段设置为 01：

```
<script type="text/javascript">

var d = new Date()
d.setUTCMinutes(1)
document.write(d)

</script>
```

输出：

```
Thu Oct 10 2013 20:01:54 GMT+0800 (中国标准时间)
```

JavaScript setUTCSeconds() 方法

定义和用法

setUTCSeconds() 方法用于根据世界时 (UTC) 设置指定时间的秒。

语法

```
dateObject.setUTCSeconds(sec,millisec)
```

参数	描述
sec	必需。要给 dateObject 设置的秒字段的值。使用世界时表示。 该参数是 0 ~ 59 之间的整数。
millisec	可选。要给 dateObject 设置的毫秒字段的值。使用世界时表示。 该参数是 0 ~ 999 之间的整数。

返回值

调整过的日期的毫秒表示。

提示和注释：

注释：如果上面的参数之一使用一位的数字来规定，那么 JavaScript 会在结果中加一或两个前置 0。
注释：该方法总是结合一个 Date 对象来使用。
提示：有关通用协调时间 (UTC) 的更多资料，请参阅百度百科。

实例

在本例中，我们将通过 setUTCSeconds() 把当前时间的秒字段设置为 01：

```
<script type="text/javascript">

var d = new Date()
d.setUTCSeconds(1)
document.write(d)
```

```
</script>
```

输出：

```
Thu Oct 10 2013 20:23:01 GMT+0800 (中国标准时间)
```

JavaScript setUTCMilliseconds() 方法

定义和用法

setUTCMilliseconds() 方法用于根据世界时 (UTC) 设置指定时间的毫秒。

语法

```
dateObject.setUTCMilliseconds(millisec)
```

参数	描述
millisec	必需。要给 dateObject 设置的毫秒字段的值。使用世界时表示。 该参数是 0 ~ 999 之间的整数。

返回值

调整过的日期的毫秒表示。

提示和注释：

注释：如果上面的参数之一使用一位的数字来规定，那么 JavaScript 会在结果中加一或两个前置 0。
注释：该方法总是结合一个 Date 对象来使用。
提示：有关通用协调时间 (UTC) 的更多资料，请参阅百度百科。

实例

在本例中，我们将通过 setUTCMilliseconds() 方法把当前时间的毫秒字段设置为 001：

```
<script type="text/javascript">
```

```
var d = new Date()
d.setUTCMilliseconds(1)
document.write(d)

</script>
```

输出：

```
Thu Oct 10 2013 20:24:44 GMT+0800 (中国标准时间)
```

JavaScript toSource() 方法

定义和用法

toSource() 方法返回表示对象源代码的字符串。

语法

```
object.toSource()
```

提示和注释

注释：该方法在 Internet Explorer 中无效。

实例

下面的例子向您展示 toSource() 方法的用法：

```
<script type="text/javascript">

function employee(name,job,born)
{
this.name=name;
this.job=job;
this.born=born;
}

var bill=new employee("Bill Gates","Engineer",1985);
```

```
document.write(bill.toSource());

</script>
```

输出：

```
({name:"Bill Gates", job:"Engineer", born:1985})
```

JavaScript toString() 方法

定义和用法

toString() 方法可把 Date 对象转换为字符串，并返回结果。

语法

```
dateObject.toString()
```

返回值

dateObject 的字符串表示，使用本地时间表示。

实例

在本例中，我们将把今天的日期转换为字符串：

```
<script type="text/javascript">

var d = new Date()
document.write (d.toString())

</script>
```

输出：

```
Thu Oct 10 2013 20:26:41 GMT+0800 (中国标准时间)
```

JavaScript toTimeString() 方法

定义和用法

toTimeString() 方法可把 Date 对象的时间部分转换为字符串，并返回结果。

语法

```
dateObject.toTimeString()
```

返回值

dateObject 的时间部分的字符串表示，由实现决定，使用本地时间表示。

JavaScript toDateString() 方法

定义和用法

toDateString() 方法可把 Date 对象的日期部分转换为字符串，并返回结果。

语法

```
dateObject.toDateString()
```

返回值

dateObject 的日期部分的字符串表示，由实现决定，使用本地时间表示。

JavaScript toGMTString() 方法

定义和用法

toGMTString() 方法可根据格林威治时间 (GMT) 把 Date 对象转换为字符串，并返回结果。

语法

```
dateObject.toGMTString()
```

返回值

dateObject 的字符串表示。此日期会在转换为字符串之前由本地时区转换为 GMT 时区。

提示和注释

不赞成使用此方法。请使用 toUTCString() 取而代之!!

实例

例子 1

在本例中，我们将把今天的日期转换为（根据 GMT）字符串：

```
<script type="text/javascript">

var d = new Date()
document.write (d.toGMTString())

</script>
```

输出：

```
Thu, 10 Oct 2013 12:28:37 GMT
```

例子 2

在下面的例子中，我们将把具体的日期转换为（根据 GMT）字符串：

```
<script type="text/javascript">

var born = new Date("July 21, 1983 01:15:00")
document.write(born.toGMTString())

</script>
```

输出：

```
Wed, 20 Jul 1983 17:15:00 GMT
```

JavaScript toUTCString() 方法

定义和用法

toUTCString() 方法可根据世界时 (UTC) 把 Date 对象转换为字符串，并返回结果。

语法

```
dateObject.toUTCString()
```

返回值

dateObject 的字符串表示，用世界时表示。

实例

例子 1

在下面的例子中，我们将使用 toUTCString() 来把今天的日期转换为（根据 UTC）字符串：

```
<script type="text/javascript">

var d = new Date()
document.write (d.toUTCString())

</script>
```

输出：

```
Thu, 10 Oct 2013 12:29:20 GMT
```

例子 2

在下面的例子中，我们将把具体的日期转换为（根据 UTC）字符串：

```
<script type="text/javascript">

var born = new Date("July 21, 1983 01:15:00")
document.write(born.toUTCString())

</script>
```

输出：

```
Wed, 20 Jul 1983 17:15:00 GMT
```

JavaScript toLocaleString() 方法

定义和用法

toLocaleString() 方法可根据本地时间把 Date 对象转换为字符串，并返回结果。

语法

```
dateObject.toLocaleString()
```

返回值

dateObject 的字符串表示，以本地时间区表示，并根据本地规则格式化。

实例

例子 1

在本例中，我们将根据本地时间把今天的日期转换为字符串：

```
<script type="text/javascript">

var d = new Date()
document.write(d.toLocaleString())

</script>
```

输出：

```
2013 年 10 月 10 日 下午 8:29:53
```

例子 2

在本例中，我们将根据本地时间把具体的日期转换为字符串：

```
<script type="text/javascript">

var born = new Date("July 21, 1983 01:15:00")
document.write(born.toLocaleString())

</script>
```

输出：

```
1983 年 7 月 21 日 上午 1:15:00
```

JavaScript toLocaleTimeString() 方法

定义和用法

toLocaleTimeString() 方法可根据本地时间把 Date 对象的时间部分转换为字符串，并返回结果。

语法

```
dateObject.toLocaleTimeString()
```

返回值

dateObject 的时间部分的字符串表示，以本地时间区表示，并根据本地规则格式化。

JavaScript toLocaleDateString() 方法

定义和用法

toLocaleDateString() 方法可根据本地时间把 Date 对象的日期部分转换为字符串，并返回结果。

语法

```
dateObject.toLocaleDateString()
```

返回值

dateObject 的日期部分的字符串表示，以本地时间区表示，并根据本地规则格式化。

JavaScript UTC() 方法

定义和用法

UTC() 方法可根据世界时返回 1970 年 1 月 1 日 到指定日期的毫秒数。

语法

```
Date.UTC(year,month,day,hours,minutes,seconds,ms)
```

参数	描述
year	必需。表示年份的四位数字。
month	必需。表示月份的整数，介于 0 ~ 11。
day	必需。表示日期的整数，介于 1 ~ 31。

hours	可选。表示小时的整数，介于 0 ~ 23。
minutes	可选。表示分钟的整数，介于 0 ~ 59。
seconds	可选。表示秒的整数，介于 0 ~ 59。
ms	可选。表示毫秒的整数，介于 0 ~ 999。

返回值

返回指定的时间距 GMT 时间 1970 年 1 月 1 日午夜的毫秒数。

说明

Date.UTC() 是一种静态方法，因为需要使用构造函数 Date() 来调用它，而不是通过某个 Date 对象调用。
Date.UTC() 方法的参数指定日期和时间，它们都是 UTC 时间，处于 GMT 时区。指定的 UTC 时间将转换成毫秒的形式，这样构造函数 Date() 和方法 Date.setTime() 就可以使用它了。

实例

例子 1

在本例中，我们将根据世界时取得从 1970/01/01 到 2005/07/08 的毫秒数：

```
<script type="text/javascript">

var d = Date.UTC(2005,7,8)
document.write(d)

</script>
```

输出：

```
1123459200000
```

例子 2

现在，我们将改造上面的例子，使得输出转换为年：

```
<script type="text/javascript">

var minutes = 1000 * 60
```

```
var hours = minutes * 60
var days = hours * 24
var years = days * 365
var t = Date.UTC(2005,7,8)
var y = t/years
document.write("It's been: " + y + " years from 1970/01/01")
document.write(" to 2005/07/08!")

</script>
```

输出：

```
It's been: 35.62465753424657 years from 1970/01/01 to 2005/07/08!
```

JavaScript valueOf() 方法

定义和用法

valueOf() 方法返回 Date 对象的原始值。
该原始值由 Date 对象派生的所有对象继承。
valueOf() 方法通常由 JavaScript 在后台自动调用，并不显式地出现在代码中。

返回值

date 的毫秒表示。返回值和方法 Date.getTime 返回的值相等。
JavaScript Math 对象的参考手册

Math 对象

Math 对象用于执行数学任务。

使用 Math 的属性和方法的语法：

```
var pi_value=Math.PI;
var sqrt_value=Math.sqrt(15);
```

注释：Math 对象并不像 Date 和 String 那样是对象的类，因此没有构造函数 Math()，像

Math.sin() 这样的函数只是函数，不是某个对象的方法。您无需创建它，通过把 Math 作为对象使用就可以调用其所有属性和方法。

Math 对象属性

FF: Firefox, IE: Internet Explorer

属性	描述	FF	IE
E	返回算术常量 e，即自然对数的底数（约等于 2.718）。	1	3
LN2	返回 2 的自然对数（约等于 0.693）。	1	3
LN10	返回 10 的自然对数（约等于 2.302）。	1	3
LOG2E	返回以 2 为底的 e 的对数（约等于 1.414）。	1	3
LOG10E	返回以 10 为底的 e 的对数（约等于 0.434）。	1	3
PI	返回圆周率（约等于 3.14159）。	1	3
SQRT1_2	返回返回 2 的平方根的倒数（约等于 0.707）。	1	3
SQRT2	返回 2 的平方根（约等于 1.414）。	1	3

Math 对象方法

FF: Firefox, IE: Internet Explorer

方法	描述	FF	IE
abs(x)	返回数的绝对值。	1	3
acos(x)	返回数的反余弦值。	1	3
asin(x)	返回数的反正弦值。	1	3
atan(x)	以介于 -PI/2 与 PI/2 弧度之间的数值来返回 x 的反正切值。	1	3
atan2(y,x)	返回从 x 轴到点 (x,y) 的角度（介于 -PI/2 与 PI/2 弧度之间）。	1	3
ceil(x)	对数进行上舍入。	1	3
cos(x)	返回数的余弦。	1	3
exp(x)	返回 e 的指数。	1	3
floor(x)	对数进行下舍入。	1	3
log(x)	返回数的自然对数（底为 e）。	1	3

max(x,y)	返回 x 和 y 中的最高值。	1	3
min(x,y)	返回 x 和 y 中的最低值。	1	3
pow(x,y)	返回 x 的 y 次幂。	1	3
random()	返回 0 ~ 1 之间的随机数。	1	3
round(x)	把数四舍五入为最接近的整数。	1	3
sin(x)	返回数的正弦。	1	3
sqrt(x)	返回数的平方根。	1	3
tan(x)	返回角的正切。	1	3
toSource()	返回该对象的源代码。	1	-
valueOf()	返回 Math 对象的原始值。	1	4

JavaScript E 属性

定义和用法

Math.E 属性代表算术常量 e，即自然对数的底数，其值近似于 2.71828。

语法

```
Math.E
```

实例

返回 Euler 数：

```
<script type="text/javascript">

document.write("Euler's number: " + Math.E);

</script>
```

输出：

```
Euler's number: 2.718281828459045
```

JavaScript LN2 属性

定义和用法

LN2 属性就是 $\log_e 2$，即 2 的自然对数，其值近似于 0.6931471805599453。

语法

```
Math.LN2
```

实例

返回 2 的自然对数：

```
<script type="text/javascript">

document.write("LN2: " + Math.LN2);

</script>
```

输出：

```
LN2: 0.6931471805599453
```

JavaScript LN10 属性

定义和用法

LN10 属性就是 $\log_e 10$，即 10 的自然对数，其值近似于 2.3025850929940459011。

语法

```
Math.LN10
```

实例

返回 10 的自然对数：

```
<script type="text/javascript">

document.write("LN10: " + Math.LN10);

</script>
```

输出：

```
LN10: 2.302585092994046
```

JavaScript LOG2E 属性

定义和用法

LOG2E 属性就是 $\log_2 e$，即以 2 为底 e 的对数，其值近似于 1.4426950408889633。

语法

```
Math.LOG2E
```

实例

返回以 2 为底 e 的对数：

```
<script type="text/javascript">

document.write("LOG2E: " + Math.LOG2E);

</script>
```

输出：

```
LOG2E: 1.4426950408889634
```

JavaScript LOG10E 属性

定义和用法

LOG10E 属性就是 $\log_{10}^{e}$，即以 10 为底 e 的对数，其值近似于 0.43429448190325181667。

语法

```
Math.LOG10E
```

实例

返回以 10 为底 e 的对数：

```
<script type="text/javascript">

document.write("LOG10E: " + Math.LOG10E);

</script>
```

输出：

```
LOG10E: 0.4342944819032518
```

JavaScript PI 属性

定义和用法

PI 属性就是 π，即圆的周长和它的直径之比。这个值近似为 3.141592653589793。

语法

```
Math.PI
```

实例

返回 PI：

```
<script type="text/javascript">

document.write("PI: " + Math.PI);

</script>
```

输出：

```
PI: 3.141592653589793
```

JavaScript SQRT1_2 属性

定义和用法

SQRT1_2 属性返回 2 的平方根的倒数。这个值近似为 0.7071067811865476。

语法

```
Math.SQRT1_2
```

实例

返回 1/2 的平方根：

```
<script type="text/javascript">

document.write("SQRT1_2: " + Math.SQRT1_2);

</script>
```

输出：

```
SQRT1_2: 0.7071067811865476
```

JavaScript SQRT2 属性

定义和用法

SQRT2 属性返回 2 的平方根。这个值近似为 1.4142135623730951。

语法

```
Math.SQRT2
```

实例

返回 2 的平方根：

```
<script type="text/javascript">

document.write("SQRT2: " + Math.SQRT2);

</script>
```

输出：

```
SQRT2: 1.4142135623730951
```

JavaScript abs() 方法

定义和用法

abs() 方法可返回数的绝对值。

语法

```
Math.abs(x)
```

参数	描述

x	必需。必须是一个数值。

返回值

x 的绝对值。

实例

在本例中，我将取得正数和负数的绝对值：

```
<script type="text/javascript">

document.write(Math.abs(7.25) + "<br />")
document.write(Math.abs(-7.25) + "<br />")
document.write(Math.abs(7.25-10))

</script>
```

输出：

```
7.25
7.25
2.75
```

JavaScript acos() 方法

定义和用法

acos() 方法可返回一个数的反余弦。

语法

```
Math.acos(x)
```

参数	描述
x	必需。必须是 -1.0 ~ 1.0 之间的数。

返回值

x 的反余弦值。返回的值是 0 到 PI 之间的弧度值。

提示和注释

注释：如果参数 x 超过了 -1.0 ~ 1.0 的范围，那么浏览器将返回 NaN。
注释：如果参数 x 取值 -1，那么将返回 PI。

实例

在本例中，我们将取得不同数的反余弦值：

```
<script type="text/javascript">

document.write(Math.acos(0.64) + "<br />")
document.write(Math.acos(0) + "<br />")
document.write(Math.acos(-1) + "<br />")
document.write(Math.acos(1) + "<br />")
document.write(Math.acos(2))

</script>
```

输出：

```
0.8762980611683406
1.5707963267948965
3.141592653589793
0
NaN
```

JavaScript asin() 方法

定义和用法

asin() 方法可返回一个数的反正弦值。

语法

```
Math.asin(x)
```

参数	描述
x	必需。必须是一个数值，该值介于 -1.0 ~ 1.0 之间。

返回值

x 的反正弦值。返回的值是 -PI/2 到 PI/2 之间的弧度值。

提示和注释

注释：如果参数 x 超过了 -1.0 ~ 1.0 的范围，那么浏览器将返回 NaN。
注释：如果参数 x 取值 1，那么将返回 PI/2。

实例

在本例中，我们将取得不同数字的反正弦值：

```
<script type="text/javascript">

document.write(Math.asin(0.64) + "<br />")
document.write(Math.asin(0) + "<br />")
document.write(Math.asin(-1) + "<br />")
document.write(Math.asin(1) + "<br />")
document.write(Math.asin(2))

</script>
```

输出：

```
0.6944982656265559
0
-1.5707963267948965
1.5707963267948965
NaN
```

JavaScript atan() 方法

定义和用法

atan() 方法可返回数字的反正切值。

语法

Math.atan(x)

参数	描述
x	必需。必须是一个数值。

返回值

x 的反正切值。返回的值是 -PI/2 到 PI/2 之间的弧度值。

实例

下面这个例子可通过 atan() 方法返回不同数字的反正切值：

```
<script type="text/javascript">

document.write(Math.atan(0.50) + "<br />")
document.write(Math.atan(-0.50) + "<br />")
document.write(Math.atan(5) + "<br />")
document.write(Math.atan(10) + "<br />")
document.write(Math.atan(-5) + "<br />")
document.write(Math.atan(-10))

</script>
```

输出：

```
0.4636476090008061
-0.4636476090008061
1.373400766945016
1.4711276743037347
```

```
-1.373400766945016
-1.4711276743037347
```

JavaScript atan2() 方法

定义和用法

atan2() 方法可返回从 x 轴到点 (x,y) 之间的角度。

语法

```
Math.atan2(y,x)
```

参数	描述
x	必需。指定点的 X 坐标。
y	必需。指定点的 Y 坐标。

返回值

-PI 到 PI 之间的值，是从 X 轴正向逆时针旋转到点 (x,y) 时经过的角度。

提示和注释

注释：请注意这个函数的参数顺序，Y 坐标在 X 坐标之前传递。

实例

下面这个例子可通过 atan2() 方法返回不同 (x,y) 点的角度：

```
<script type="text/javascript">

document.write(Math.atan2(0.50,0.50) + "<br />")
document.write(Math.atan2(-0.50,-0.50) + "<br />")
document.write(Math.atan2(5,5) + "<br />")
document.write(Math.atan2(10,20) + "<br />")
document.write(Math.atan2(-5,-5) + "<br />")
```

```
document.write(Math.atan2(-10,10))

</script>
```

输出：

```
0.7853981633974483
-2.356194490192345
0.7853981633974483
0.4636476090008061
-2.356194490192345
-0.7853981633974483
```

JavaScript ceil() 方法

定义和用法

ceil() 方法可对一个数进行上舍入。

语法

```
Math.ceil(x)
```

参数	描述
x	必需。必须是一个数值。

返回值

大于等于 x，并且与它最接近的整数。

说明

ceil() 方法执行的是向上取整计算，它返回的是大于或等于函数参数，并且与之最接近的整数。

实例

在本例中，我们将把 ceil() 方法运用到不同的数字上：

```
<script type="text/javascript">

document.write(Math.ceil(0.60) + "<br />")
document.write(Math.ceil(0.40) + "<br />")
document.write(Math.ceil(5) + "<br />")
document.write(Math.ceil(5.1) + "<br />")
document.write(Math.ceil(-5.1) + "<br />")
document.write(Math.ceil(-5.9))

</script>
```

输出：

```
1
1
5
6
-5
-5
```

JavaScript cos() 方法

定义和用法

cos() 方法可返回一个数字的余弦值。

语法

```
Math.cos(x)
```

参数	描述
x	必需。必须是一个数值。

返回值

x 的余弦值。返回的是 -1.0 到 1.0 之间的数。

实例

在本例中，我们将返回不同数值的余弦值：

```
<script type="text/javascript">

document.write(Math.cos(3) + "<br />")
document.write(Math.cos(-3) + "<br />")
document.write(Math.cos(0) + "<br />")
document.write(Math.cos(Math.PI) + "<br />")
document.write(Math.cos(2*Math.PI))

</script>
```

输出：

```
-0.9899924966004454
-0.9899924966004454
1
-1
1
```

JavaScript exp() 方法

定义和用法

exp() 方法可返回 e 的 x 次幂的值。

语法

```
Math.exp(x)
```

参数	描述

x	必需。任意数值或表达式。被用作指数。

返回值

返回 e 的 x 次幂。e 代表自然对数的底数，其值近似为 2.71828。

实例

在下面的例子中，我们将把 exp() 运用到不同的数值上：

```
<script type="text/javascript">

document.write(Math.exp(1) + "<br />")
document.write(Math.exp(-1) + "<br />")
document.write(Math.exp(5) + "<br />")
document.write(Math.exp(10) + "<br />")

</script>
```

输出：

```
2.718281828459045
0.36787944117144233
148.4131591025766
22026.465794806718
```

JavaScript floor() 方法

定义和用法

floor() 方法可对一个数进行下舍入。

语法

```
Math.floor(x)
```

参数	描述

x	必需。任意数值或表达式。

返回值

小于等于 x，且与 x 最接近的整数。

说明

floor() 方法执行的是向下取整计算，它返回的是小于或等于函数参数，并且与之最接近的整数。

实例

In this example we will use the floor() method on different numbers:

```
<script type="text/javascript">

document.write(Math.floor(0.60) + "<br />")
document.write(Math.floor(0.40) + "<br />")
document.write(Math.floor(5) + "<br />")
document.write(Math.floor(5.1) + "<br />")
document.write(Math.floor(-5.1) + "<br />")
document.write(Math.floor(-5.9))

</script>
```

输出：

```
0
0
5
5
-6
-6
```

JavaScript log() 方法

定义和用法

log() 方法可返回一个数的自然对数。

语法

```
Math.log(x)
```

参数	描述
x	必需。任意数值或表达式。

返回值

x 的自然对数。

说明

参数 x 必须大于 0。

实例

在本例中，我们将对不同的数运用 log() ：

```
<script type="text/javascript">

document.write(Math.log(2.7183) + "<br />")
document.write(Math.log(2) + "<br />")
document.write(Math.log(1) + "<br />")
document.write(Math.log(0) + "<br />")
document.write(Math.log(-1))

</script>
```

输出：

```
1.0000066849139877
0.6931471805599453
0
-Infinity
NaN
```

JavaScript max() 方法

定义和用法

max() 方法可返回两个指定的数中带有较大的值的那个数。

语法

```
Math.max(x...)
```

参数	描述
x	0 或多个值。在 ECMASCript v3 之前，该方法只有两个参数。

返回值

参数中最大的值。如果没有参数，则返回 -Infinity。如果有某个参数为 NaN，或是不能转换成数字的非数字值，则返回 NaN。

实例

在本例中，我们将展示如何使用 max() 来返回指定数字中带有最高值的数字：

```
<script type="text/javascript">

document.write(Math.max(5,7) + "<br />")
document.write(Math.max(-3,5) + "<br />")
document.write(Math.max(-3,-5) + "<br />")
document.write(Math.max(7.25,7.30))

</script>
```

输出：

```
7
5
-3
7.3
```

JavaScript min() 方法

定义和用法

min() 方法可返回指定的数字中带有最低值的数字。

语法

```
Math.min(x,y)
```

参数	描述
x	0 或多个值。在 ECMASCript v3 之前，该方法只有两个参数。

返回值

参数中最小的值。如果没有参数，则返回 Infinity。如果有某个参数为 NaN，或是不能转换成数字的非数字值，则返回 NaN。

实例

在本例中，我们将展示如何使用 min() 来返回指定数字中带有最低值的数字：

```
<script type="text/javascript">

document.write(Math.min(5,7) + "<br />")
document.write(Math.min(-3,5) + "<br />")
document.write(Math.min(-3,-5) + "<br />")
document.write(Math.min(7.25,7.30))

</script>
```

输出：

```
5
-3
-5
7.25
```

JavaScript pow() 方法

定义和用法

pow() 方法可返回 x 的 y 次幂的值。

语法

```
Math.pow(x,y)
```

参数	描述
x	必需。底数。必须是数字。
y	必需。幂数。必须是数字。

返回值

x 的 y 次幂。

说明

如果结果是虚数或负数，则该方法将返回 NaN。如果由于指数过大而引起浮点溢出，则该方法将返回 Infinity。

实例

在下面的例子中，我们将把 pow() 运用到不同的数字组合上：

```
<script type="text/javascript">
```

```
document.write(Math.pow(0,0) + "<br />")
document.write(Math.pow(0,1) + "<br />")
document.write(Math.pow(1,1) + "<br />")
document.write(Math.pow(1,10) + "<br />")
document.write(Math.pow(2,3) + "<br />")
document.write(Math.pow(-2,3) + "<br />")
document.write(Math.pow(2,4) + "<br />")
document.write(Math.pow(-2,4) + "<br />")

</script>
```

输出：

```
1
0
1
1
8
-8
16
16
```

JavaScript random() 方法

定义和用法

random() 方法可返回介于 0 ~ 1 之间的一个随机数。

语法

```
Math.random()
```

返回值

0.0 ~ 1.0 之间的一个伪随机数。

实例

在本例中，我们将取得介于 0 到 1 之间的一个随机数：

```
<script type="text/javascript">

document.write(Math.random())

</script>
```

输出：

```
0.21242664568126202
```

JavaScript round() 方法

定义和用法

round() 方法可把一个数字舍入为最接近的整数。

语法

```
Math.round(x)
```

参数	描述
x	必需。必须是数字。

返回值

与 x 最接近的整数。

说明

对于 0.5，该方法将进行上舍入。
例如，3.5 将舍入为 4，而 -3.5 将舍入为 -3。

实例

把不同的数舍入为最接近的整数：

```
<script type="text/javascript">

document.write(Math.round(0.60) + "<br />")
document.write(Math.round(0.50) + "<br />")
document.write(Math.round(0.49) + "<br />")
document.write(Math.round(-4.40) + "<br />")
document.write(Math.round(-4.60))

</script>
```

输出：

```
1
1
0
-4
-5
```

JavaScript sin() 方法

定义和用法

sin() 方法可返回一个数字的正弦。

语法

```
Math.sin(x)
```

参数	描述
x	必需。一个以弧度表示的角。将角度乘以 0.017453293 （2PI/360）即可转换为弧度。

返回值

参数 x 的正弦值。返回值在 -1.0 到 1.0 之间。

实例

在本例中，我们将返回不同数字的正弦：

```
<script type="text/javascript">

document.write(Math.sin(3) + "<br />")
document.write(Math.sin(-3) + "<br />")
document.write(Math.sin(0) + "<br />")
document.write(Math.sin(Math.PI) + "<br />")
document.write(Math.sin(Math.PI/2)

</script>
```

输出：

```
0.1411200080598672
-0.1411200080598672
0
1.2246063538223772e-16
1
```

JavaScript sqrt() 方法

定义和用法

sqrt() 方法可返回一个数的平方根。

语法

```
Math.sqrt(x)
```

参数	描述

x	必需。必须是大于等于 0 的数。

返回值

参数 *x* 的平方根。如果 *x* 小于 0，则返回 NaN。

提示和注释

提示：Math.pow() 方法可以计算一个数的任意次根。

实例

在本例中，我们将返回不同数字的平方根：

```
var a=Math.sqrt(0);
var b=Math.sqrt(1);
var c=Math.sqrt(9);
var d=Math.sqrt(0.64);
var e=Math.sqrt(-9);
```

输出：

```
0
1
3
0.8
NaN
```

JavaScript tan() 方法

定义和用法

tan() 方法可返回一个表示某个角的正切的数字。

语法

```
Math.tan(x)
```

参数	描述
x	必需。一个以弧度表示的角。将角度乘以 0.017453293 （2PI/360）即可转换为弧度。

返回值

参数 x 的正切值。

实例

在本例中，我们将计算不同数字的正切值：

```
<script type="text/javascript">

document.write(Math.tan(0.50) + "<br />")
document.write(Math.tan(-0.50) + "<br />")
document.write(Math.tan(5) + "<br />")
document.write(Math.tan(10) + "<br />")
document.write(Math.tan(-5) + "<br />")
document.write(Math.tan(-10))

</script>
```

输出：

```
0.5463024898437905
-0.5463024898437905
-3.380515006246586
0.6483608274590866
3.380515006246586
-0.6483608274590866
```

JavaScript toSource() 方法

定义和用法

toSource() 方法返回表示对象源代码的字符串。

语法

object.toSource()

提示和注释

注释：该方法在 Internet Explorer 中无效。

实例

下面的例子向您展示 toSource() 方法的用法：

```
<script type="text/javascript">

function employee(name,job,born)
{
this.name=name;
this.job=job;
this.born=born;
}

var bill=new employee("Bill Gates","Engineer",1985);

document.write(bill.toSource());

</script>
```

输出：

```
({name:"Bill Gates", job:"Engineer", born:1985})
```

JavaScript valueOf() 方法

定义和用法

valueOf() 方法返回 Math 对象的原始值。
该原始值由 Math 对象派生的所有对象继承。
valueOf() 方法通常由 JavaScript 在后台自动调用，并不显式地出现在代码中。

语法

```
mathObject.valueOf()
```

Number 对象

Number 对象是原始数值的包装对象。

创建 Number 对象的语法：

```
var myNum=new Number(value);
var myNum=Number(value);
```

参数

参数 *value* 是要创建的 Number 对象的数值，或是要转换成数字的值。

返回值

当 Number() 和运算符 new 一起作为构造函数使用时，它返回一个新创建的 Number 对象。如果不用 new 运算符，把 Number() 作为一个函数来调用，它将把自己的参数转换成一个原始的数值，并且返回这个值（如果转换失败，则返回 NaN）。

Number 对象属性

FF: Firefox, IE: Internet Explorer

属性	描述	FF	IE
constructor	返回对创建此对象的 Number 函数的引用。	1.0	4.0
MAX_VALUE	可表示的最大的数。	1.0	4.0
MIN_VALUE	可表示的最小的数。	1.0	4.0
NaN	非数字值。	1.0	4.0
NEGATIVE_INFINITY	负无穷大，溢出时返回该值。	1.0	4.0
POSITIVE_INFINITY	正无穷大，溢出时返回该值。	1.0	4.0

prototype	使您有能力向对象添加属性和方法。	1.0	4.0

Number 对象方法

FF: Firefox, IE: Internet Explorer

方法	描述	FF	IE
toString	把数字转换为字符串，使用指定的基数。	1.0	4.0
toLocaleString	把数字转换为字符串，使用本地数字格式顺序。	1.0	4.0
toFixed	把数字转换为字符串，结果的小数点后有指定位数的数字。	1.0	5.5
toExponential	把对象的值转换为指数计数法。	1.0	5.5
toPrecision	把数字格式化为指定的长度。	1.0	5.5
valueOf	返回一个 Number 对象的基本数字值。	1.0	4.0

Number 对象描述

在 JavaScript 中，数字是一种基本的数据类型。JavaScript 还支持 Number 对象，该对象是原始数值的包装对象。在必要时，JavaScript 会自动地在原始数据和对象之间转换。在 JavaScript 1.1 中，可以用构造函数 Number() 明确地创建一个 Number 对象，尽管这样做并没有什么必要。
构造函数 Number() 可以不与运算符 new 一起使用，而直接作为转化函数来使用。以这种方式调用 Number() 时，它会把自己的参数转化成一个数字，然后返回转换后的原始数值（或 NaN）。
构造函数通常还用作 5 个有用的数字常量的占位符，这 5 个有用的数字常量分别是可表示的最大数、可表示的最小数、正无穷大、负无穷大和特殊的 NaN 值。 注意，这些值是构造函数 Number() 自身的属性，而不是单独的某个 Number 对象的属性。
比如这样使用属性 MAX_VALUE 是正确的：

```
var big = Number.MAX_VALUE
```

但是这样是错误的：

```
var n= new Number(2);
var big = n.MAX_VALUE
```

作为比较，我们看一下 toString() 和 Number 对象的其他方法，它们是每个 Number 对象的方法，而不是 Number() 构造函数的方法。前面提到过，在必要时，JavaScript 会自动地

把原始数值转化成 Number 对象，调用 Number 方法的既可以是 Number 对象，也可以是原始数字值。

```
var n = 123;
var binary_value = n.toString(2);
```

课外书

如需更多信息，请阅读 JavaScript 高级教程中的相关内容：

ECMAScript 引用类型
引用类型通常叫做类（class）或对象。本节讲解 ECMAScript 的预定义引用类型。

JavaScript constructor 属性

定义和用法

constructor 属性返回对创建此对象的 Boolean 函数的引用。

语法

```
object.constructor
```

实例

在本例中，我们将展示如何使用 constructor 属性：

```
<script type="text/javascript">

var test=new Boolean();

if (test.constructor==Array)
{
document.write("This is an Array");
}
if (test.constructor==Boolean)
{
document.write("This is a Boolean");
}
```

```
if (test.constructor==Date)
{
document.write("This is a Date");
}
if (test.constructor==String)
{
document.write("This is a String");
}

</script>
```

输出：

```
This is a Boolean
```

JavaScript MAX_VALUE 属性

定义和用法

MAX_VALUE 属性是 JavaScript 中可表示的最大的数。它的近似值为 1.7976931348623157 x 10^{308}。

语法

```
Number.MAX_VALUE
```

实例

返回 JavaScript 中可能的最大值：

```
<script type="text/javascript">

document.write(Number.MAX_VALUE);

</script>
```

输出：

```
1.7976931348623157e+308
```

JavaScript MIN_VALUE 属性

定义和用法

MIN_VALUE 属性是 JavaScript 中可表示的最小的数（接近 0 ，但不是负数）。它的近似值为 5×10^{-324}。

语法

```
Number.MIN_VALUE
```

实例

返回 JavaScript 中可能的最小值：

```
<script type="text/javascript">

document.write(Number.MIN_VALUE);

</script>
```

输出：

```
5e-324
```

JavaScript NaN 属性

定义和用法

NaN 属性是代表非数字值的特殊值。该属性用于指示某个值不是数字。可以把 Number 对象设置为该值，来指示其不是数字值。

提示：请使用 isNaN() 全局函数来判断一个值是否是 NaN 值。

语法

```
Number.NaN
```

说明

Number.NaN 是一个特殊值，说明某些算术运算（如求负数的平方根）的结果不是数字。方法 parseInt() 和 parseFloat() 在不能解析指定的字符串时就返回这个值。对于一些常规情况下返回有效数字的函数，也可以采用这种方法，用 Number.NaN 说明它的错误情况。
JavaScript 以 NaN 的形式输出 Number.NaN。请注意，NaN 与其他数值进行比较的结果总是不相等的，包括它自身在内。因此，不能与 Number.NaN 比较来检测一个值是不是数字，而只能调用 isNaN() 来比较。
在 ECMAScript v1 和其后的版本中，还可以用预定义的全局属性 NaN 代替 Number.NaN。

提示和注释

提示：请使用 isNaN() 来判断一个值是否是数字。原因是 NaN 与所有值都不相等，包括它自己。

实例

用 NaN 指示某个值是否是数字：

```
<script type="text/javascript">

var Month=30;

if (Month < 1 || Month > 12)
{
Month = Number.NaN;
}

document.write(Month);

</script>
```

输出：

```
Nan
```

JavaScript NEGATIVE_INFINITY 属性

定义和用法

NEGATIVE_INFINITY 属性表示小于 Number.MIN_VALUE 的值。
该值代表负无穷大。

语法

```
Number.NEGATIVE_INFINITY
```

说明

Number.NEGATIVE_INFINITY 是一个特殊值，它在算术运算或函数生成一个比 JavaScript 能表示的最小负数还小的数（也就是比 -Number.MAX_VALUE 还小的数）时返回。
JavaScript 显示 NEGATIVE_INFINITY 时使用的是 -Infinity。这个值的算术行为和无穷大非常相似。例如，任何数乘无穷大结果仍为无穷大，任何数被无穷大除的结果为 0。
在 ECMAScript v1 和其后的版本中，还可以用 -Infinity 代替 Number.NEGATIVE_INFINITY。

实例

在本例中，我们将使用 NEGATIVE_INFINITY：

```
<script type="text/javascript">

var x=(-Number.MAX_VALUE)*2
if (x==Number.NEGATIVE_INFINITY)
  {
  document.write("Value of x: " + x);
  }

</script>
```

输出：

```
Value of x: -Infinity
```

JavaScript POSITIVE_INFINITY 属性

定义和用法

POSITIVE_INFINITY 属性表示大于 Number.MAX_VALUE 的值。
该值代表正无穷大。

语法

```
Number.POSITIVE_INFINITY
```

说明

Number.POSITIVE_INFINITY 是一个特殊值，它在算术运算或函数生成一个比 JavaScript 能表示的最大的数还大的数（也就是比 Number.MAX_VALUE 还大的数）时返回。
JavaScript 显示 POSITIVE_INFINITY 时使用的是 Infinity。这个值的算术行为和无穷大非常相似。例如，任何数乘无穷大结果仍为无穷大，任何数被无穷大除的结果为 0。
在 ECMAScript v1 和其后的版本中，还可以用 Infinity 代替 Number.POSITIVE_INFINITY。

实例

在本例中，我们将使用 POSITIVE_INFINITY：

```
<script type="text/javascript">

var x=(Number.MAX_VALUE)*2
if (x==Number.POSITIVE_INFINITY)
  {
  document.write("Value of x: " + x);
  }

</script>
```

输出：

```
Value of x: Infinity
```

JavaScript toString() 方法

定义和用法

toString() 方法可把一个 Number 对象转换为一个字符串，并返回结果。

语法

```
NumberObject.toString(radix)
```

参数	描述
radix	可选。规定表示数字的基数，使 2 ~ 36 之间的整数。若省略该参数，则使用基数 10。但是要注意，如果该参数是 10 以外的其他值，则 ECMAScript 标准允许实现返回任意值。

返回值

数字的字符串表示。例如，当 radix 为 2 时，NumberObject 会被转换为二进制值表示的字符串。

抛出

当调用该方法的对象不是 Number 时抛出 TypeError 异常。

实例

在本例中，我们将把一个数字转换为字符串：

```
<script type="text/javascript">

var number = new Number(1337);
document.write (number.toString())

</script>
```

输出：

```
1337
```

JavaScript toLocaleString() 方法

定义和用法

toLocaleString() 方法可把一个 Number 对象转换为本地格式的字符串。

语法

```
NumberObject.toLocaleString()
```

返回值

数字的字符串表示，由实现决定，根据本地规范进行格式化，可能影响到小数点或千分位分隔符采用的标点符号。

抛出

当调用该方法的对象不是 Number 时抛出 TypeError 异常。

JavaScript toFixed() 方法

定义和用法

toFixed() 方法可把 Number 四舍五入为指定小数位数的数字。

语法

```
NumberObject.toFixed(num)
```

参数	描述
num	必需。规定小数的位数，是 0 ~ 20 之间的值，包括 0 和 20，有些实现可以支持更大的数值范围。如果省略了该参数，将用 0 代替。

返回值

返回 NumberObject 的字符串表示，不采用指数计数法，小数点后有固定的 num 位数字。如果必要，该数字会被舍入，也可以用 0 补足，以便它达到指定的长度。如果 num 大于 le+21，则该方法只调用 NumberObject.toString()，返回采用指数计数法表示的字符串。

抛出

当 num 太小或太大时抛出异常 RangeError。0 ~ 20 之间的值不会引发该异常。有些实现支持更大范围或更小范围内的值。
当调用该方法的对象不是 Number 时抛出 TypeError 异常。

实例

在本例中，我们将把数字舍入为仅有一位小数的数字：

```
Show the number 13.37 with one decimal:
<script type="text/javascript">
var num = new Number(13.37);
document.write (num.toFixed(1))
</script>
```

输出：

```
Show the number 13.37 with one decimal:
13.4
```

JavaScript toExponential() 方法

定义和用法

toExponential() 方法可把对象的值转换成指数计数法。

语法

```
NumberObject.toExponential(num)
```

参数	描述
num	必需。规定指数计数法中的小数位数，是 0 ~ 20 之间的值，包括 0 和 20，有些实现可以支持更大的数值范围。如果省略了该参数，将使用尽可能多的数字。

返回值

返回 NumberObject 的字符串表示，采用指数计数法，即小数点之前有一位数字，小数点之后有 num 位数字。该数字的小数部分将被舍入，必要时用 0 补足，以便它达到指定的长度。

抛出

当 num 太小或太大时抛出异常 RangeError。0 ~ 20 之间的值不会引发该异常。有些实现支持更大范围或更小范围内的值。
当调用该方法的对象不是 Number 时抛出 TypeError 异常。

实例

在本例中，我们将把一个数字转换为指数计数法：

```
Show 10,000 as an exponential notation:
<script type="text/javascript">
var num = new Number(10000);
document.write (num.toExponential(1))
</script>
```

输出：

```
Show 10,000 as an exponential notation:
1.0e+4
```

JavaScript toPrecision() 方法

定义和用法

toPrecision() 方法可在对象的值超出指定位数时将其转换为指数计数法。

语法

```
NumberObject.toPrecision(num)
```

参数	描述
num	必需。规定必须被转换为指数计数法的最小位数。该参数是 1 ~ 21 之间（且包括 1 和 21）的值。有效实现允许有选择地支持更大或更小的 num。如果省略了该参数，则调用方法 toString()，而不是把数字转换成十进制的值。

返回值

返回 NumberObject 的字符串表示，包含 num 个有效数字。如果 num 足够大，能够包括 NumberObject 整数部分的所有数字，那么返回的字符串将采用定点计数法。否则，采用指数计数法，即小数点前有一位数字，小数点后有 num-1 位数字。必要时，该数字会被舍入或用 0 补足。

抛出

当 num 太小或太大时抛出异常 RangeError。1 ~ 21 之间的值不会引发该异常。有些实现支持更大范围或更小范围内的值。
当调用该方法的对象不是 Number 时抛出 TypeError 异常。

实例

在本例中，我们将把一个数字转换为指数计数法：

```
Show 10,000 as an exponential notation:
<script type="text/javascript">
var num = new Number(10000);
document.write (num.toPrecision(4))
</script>
```

输出：

```
Show 10,000 as an exponential notation:
1.000e+4
```

JavaScript valueOf() 方法

定义和用法

valueOf() 方法可以字符串返回数字。
字符串的输出通常等于该数字。
valueOf() 方法通常由 JavaScript 在后台自动进行调用，而不是显式地处于代码中。

语法

```
NumberObject.valueOf()
```

返回值

NumberObject 的原始数值。

抛出

当调用该方法的对象不是 Number 时抛出 TypeError 异常。

JavaScript String 对象参考手册

String 对象

String 对象用于处理文本（字符串）。

创建 String 对象的语法：

```
new String(s);
String(s);
```

参数

参数 *s* 是要存储在 String 对象中或转换成原始字符串的值。

返回值

当 String() 和运算符 new 一起作为构造函数使用时，它返回一个新创建的 String 对象，存放的是字符串 *s* 或 *s* 的字符串表示。
当不用 new 运算符调用 String() 时，它只把 *s* 转换成原始的字符串，并返回转换后的值。

String 对象属性

FF: Firefox, IE: Internet Explorer

属性	描述	FF	IE
constructor	对创建该对象的函数的引用	1	4
length	字符串的长度	1	3
prototype	允许您向对象添加属性和方法	1	4

String 对象方法

FF: Firefox, IE: Internet Explorer

方法	描述	FF	IE
anchor()	创建 HTML 锚。	1	3
big()	用大号字体显示字符串。	1	3
blink()	显示闪动字符串。	1	
bold()	使用粗体显示字符串。	1	3
charAt()	返回在指定位置的字符。	1	3
charCodeAt()	返回在指定的位置的字符的 Unicode 编码。	1	4
concat()	连接字符串。	1	4
fixed()	以打字机文本显示字符串。	1	3
fontcolor()	使用指定的颜色来显示字符串。	1	3
fontsize()	使用指定的尺寸来显示字符串。	1	3
fromCharCode()	从字符编码创建一个字符串。	1	4
indexOf()	检索字符串。	1	3

italics()	使用斜体显示字符串。	1	3
lastIndexOf()	从后向前搜索字符串。	1	3
link()	将字符串显示为链接。	1	3
localeCompare()	用本地特定的顺序来比较两个字符串。	1	4
match()	找到一个或多个正则表达式的匹配。	1	4
replace()	替换与正则表达式匹配的子串。	1	4
search()	检索与正则表达式相匹配的值。	1	4
slice()	提取字符串的片断，并在新的字符串中返回被提取的部分。	1	4
small()	使用小字号来显示字符串。	1	3
split()	把字符串分割为字符串数组。	1	4
strike()	使用删除线来显示字符串。	1	3
sub()	把字符串显示为下标。	1	3
substr()	从起始索引号提取字符串中指定数目的字符。	1	4
substring()	提取字符串中两个指定的索引号之间的字符。	1	3
sup()	把字符串显示为上标。	1	3
toLocaleLowerCase()	把字符串转换为小写。	-	-
toLocaleUpperCase()	把字符串转换为大写。	-	-
toLowerCase()	把字符串转换为小写。	1	3
toUpperCase()	把字符串转换为大写。	1	3
toSource()	代表对象的源代码。	1	-
toString()	返回字符串。	-	-
valueOf()	返回某个字符串对象的原始值。	1	4

String 对象描述

字符串是 JavaScript 的一种基本的数据类型。
String 对象的 length 属性声明了该字符串中的字符数。
String 类定义了大量操作字符串的方法，例如从字符串中提取字符或子串，或者检索字符或子串。
需要注意的是，JavaScript 的字符串是不可变的（immutable），String 类定义的方法都不能改变字符串的内容。像 String.toUpperCase() 这样的方法，返回的是全新的字符串，而不是修改原始字符串。

在较早的 Netscape 代码基的 JavaScript 实现中（例如 Firefox 实现中），字符串的行为就像只读的字符数组。例如，从字符串 s 中提取第三个字符，可以用 s[2] 代替更加标准的 s.charAt(2)。此外，对字符串应用 for/in 循环时，它将枚举字符串中每个字符的数组下标（但要注意，ECMAScript 标准规定，不能枚举 length 属性）。因为字符串的数组行为不标准，所以应该避免使用它。

课外书

如需更多信息，请阅读 JavaScript 高级教程中的相关内容：

ECMAScript 引用类型
引用类型通常叫做类（class）或对象。本节讲解 ECMAScript 的预定义引用类型。

JavaScript length 属性

定义和用法

length 属性可返回字符串中的字符数目。

语法

```
stringObject.length
```

实例

在本例中，我们将展示如何使用 length 属性来返回字符串中的字符数：

```
<script type="text/javascript">

var txt="Hello World!"
document.write(txt.length)

</script>
```

输出：

```
12
```

JavaScript anchor() 方法

定义和用法

anchor() 方法用于创建 HTML 锚。

stringObject.anchor(anchorname)

参数	描述
anchorname	必需。为锚定义名称。

实例

在本例中，我们会为文本添加一个锚：

```
<script type="text/javascript">

var txt="Hello world!"
document.write(txt.anchor("myanchor"))

</script>
```

上面的代码将输出为纯粹的 HTML：

```
<a name="myanchor">Hello world!</a>
```

JavaScript big() 方法

定义和用法

big() 方法用于把字符串显示为大号字体。

```
stringObject.big()
```

实例

在本例中，"Hello world!" 将被显示为大号字体：

```
<script type="text/javascript">

var str="Hello world!"
document.write(str.big())

</script>
```

JavaScript blink() 方法

定义和用法

blink() 方法用于显示闪动的字符串。

语法

```
stringObject.blink()
```

提示和注释

注释：此方法无法工作于 Internet Explorer 中。

实例

在本例中，"Hello world!" 将被显示为闪动的文本：

```
<script type="text/javascript">

var str="Hello world!"
document.write(str.blink())

</script>
```

JavaScript bold() 方法

定义和用法

bold() 方法用于把字符串显示为粗体。

语法

```
stringObject.bold()
```

实例

在本例中，"Hello world!" 将被显示为粗体：

```
<script type="text/javascript">

var str="Hello world!"
document.write(str.bold())

</script>
```

JavaScript charAt() 方法

定义和用法

charAt() 方法可返回指定位置的字符。
请注意，JavaScript 并没有一种有别于字符串类型的字符数据类型，所以返回的字符是长度为 1 的字符串。

语法

```
stringObject.charAt(index)
```

参数	描述

index	必需。表示字符串中某个位置的数字，即字符在字符串中的下标。

提示和注释

注释：字符串中第一个字符的下标是 0。如果参数 index 不在 0 与 string.length 之间，该方法将返回一个空字符串。

实例

在字符串 "Hello world!" 中，我们将返回位置 1 的字符：

```
<script type="text/javascript">

var str="Hello world!"
document.write(str.charAt(1))

</script>
```

以上代码的输出是：

```
e
```

JavaScript charCodeAt() 方法

定义和用法

charCodeAt() 方法可返回指定位置的字符的 Unicode 编码。这个返回值是 0 - 65535 之间的整数。
方法 charCodeAt() 与 charAt() 方法执行的操作相似，只不过前者返回的是位于指定位置的字符的编码，而后者返回的是字符子串。

语法

```
stringObject.charCodeAt(index)
```

参数	描述
index	必需。表示字符串中某个位置的数字，即字符在字符串中的下标。

提示和注释

注释：字符串中第一个字符的下标是 0。如果 index 是负数，或大于等于字符串的长度，则 charCodeAt() 返回 NaN。

实例

在字符串 "Hello world!" 中，我们将返回位置 1 的字符的 Unicode 编码：

```
<script type="text/javascript">

var str="Hello world!"
document.write(str.charCodeAt(1))

</script>
```

以上代码的输出是：

```
101
```

JavaScript concat() 方法

定义和用法

concat() 方法用于连接两个或多个字符串。

语法

```
stringObject.concat(stringX,stringX,...,stringX)
```

参数	描述
stringX	必需。将被连接为一个字符串的一个或多个字符串对象。

concat() 方法将把它的所有参数转换成字符串，然后按顺序连接到字符串 stringObject 的尾部，并返回连接后的字符串。请注意，stringObject 本身并没有被更改。
stringObject.concat() 与 Array.concat() 很相似。

提示和注释

提示：请注意，使用 "+" 运算符来进行字符串的连接运算通常会更简便一些。

实例

在本例中，我们将创建两个字符串，然后使用 concat() 把它们显示为一个字符串：

```
<script type="text/javascript">

var str1="Hello "
var str2="world!"
document.write(str1.concat(str2))

</script>
```

以上代码的输出是：

```
Hello world!
```

JavaScript fixed() 方法

定义和用法

fixed() 方法用于把字符串显示为打字机字体。

语法

```
stringObject.fixed()
```

实例

在本例中，"Hello world!" 将被显示为打字机文本：

```
<script type="text/javascript">

var str="Hello world!"
document.write(str.fixed())
```

```
</script>
```

JavaScript fontcolor() 方法

定义和用法

fontcolor() 方法用于按照指定的颜色来显示字符串。

语法

```
stringObject.fontcolor(color)
```

参数	描述
color	必需。为字符串规定 font-color。该值必须是颜色名(red)、RGB 值(rgb(255,0,0))或者十六进制数(#FF0000)。

实例

在本例中，"Hello world!" 将显示为红色：

```
<script type="text/javascript">

var str="Hello world!"
document.write(str.fontcolor("Red"))

</script>
```

JavaScript fontsize() 方法

定义和用法

fontsize() 方法用于按照指定的尺寸来显示字符串。

语法

```
stringObject.fontsize(size)
```

提示和注释

注释：size 参数必须是从 1 至 7 的数字。
实例
在本例中，"Hello world!" 将显示为大号字：

```
<script type="text/javascript">

var str="Hello world!"
document.write(str.fontsize(7))

</script>
```

JavaScript fromCharCode() 方法

定义和用法

fromCharCode() 可接受一个指定的 Unicode 值，然后返回一个字符串。

语法

```
String.fromCharCode(numX,numX,...,numX)
```

参数	描述
numX	必需。一个或多个 Unicode 值，即要创建的字符串中的字符的 Unicode 编码。

提示和注释

注释：该方法是 String 的静态方法，字符串中的每个字符都由单独的数字 Unicode 编码指定。
它不能作为您已创建的 String 对象的方法来使用。因此它的语法应该是 String.fromCharCode()，而不是 myStringObject.fromCharCode()。

实例

在本例中，我们将根据 Unicode 来输出 "HELLO" 和 "ABC"：

```
<script type="text/javascript">

document.write(String.fromCharCode(72,69,76,76,79))
document.write("<br />")
document.write(String.fromCharCode(65,66,67))

</script>
```

以上代码的输出：

```
HELLO
ABC
```

JavaScript indexOf() 方法

定义和用法

indexOf() 方法可返回某个指定的字符串值在字符串中首次出现的位置。

语法

```
stringObject.indexOf(searchvalue,fromindex)
```

参数	描述
searchvalue	必需。规定需检索的字符串值。
fromindex	可选的整数参数。规定在字符串中开始检索的位置。它的合法取值是 0 到 stringObject.length - 1。如省略该参数，则将从字符串的首字符开始检索。

说明

该方法将从头到尾地检索字符串 stringObject，看它是否含有子串 searchvalue。开始检索的位置在字符串的 fromindex 处或字符串的开头（没有指定 fromindex 时）。如果找到一个 searchvalue，则返回 searchvalue 的第一次出现的位置。stringObject 中的字符位置是从 0

开始的。

提示和注释

注释：indexOf() 方法对大小写敏感！
注释：如果要检索的字符串值没有出现，则该方法返回 -1。

实例

在本例中，我们将在 "Hello world!" 字符串内进行不同的检索：

```
<script type="text/javascript">

var str="Hello world!"
document.write(str.indexOf("Hello") + "<br />")
document.write(str.indexOf("World") + "<br />")
document.write(str.indexOf("world"))

</script>
```

以上代码的输出：

```
0
-1
6
```

JavaScript italics() 方法

定义和用法

italics() 方法用于把字符串显示为斜体。

语法

```
stringObject.italics()
```

实例

在本例中，"Hello world!" 将被显示为斜体：

```
<script type="text/javascript">

var str="Hello world!"
document.write(str.italics())

</script>
```

JavaScript lastIndexOf() 方法

定义和用法

lastIndexOf() 方法可返回一个指定的字符串值最后出现的位置，在一个字符串中的指定位置从后向前搜索。

语法

```
stringObject.lastIndexOf(searchvalue,fromindex)
```

参数	描述
searchvalue	必需。规定需检索的字符串值。
fromindex	可选的整数参数。规定在字符串中开始检索的位置。它的合法取值是 0 到 *stringObject*.length - 1。如省略该参数，则将从字符串的最后一个字符处开始检索。

返回值

如果在 *stringObject* 中的 *fromindex* 位置之前存在 *searchvalue*，则返回的是出现的最后一个 *searchvalue* 的位置。

说明

该方法将从尾到头地检索字符串 *stringObject*，看它是否含有子串 *searchvalue*。开始检索的

位置在字符串的 *fromindex* 处或字符串的结尾（没有指定 *fromindex* 时）。如果找到一个 *searchvalue*，则返回 *searchvalue* 的第一个字符在 *stringObject* 中的位置。*stringObject* 中的字符位置是从 0 开始的。

提示和注释

注释：lastIndexOf() 方法对大小写敏感！
注释：如果要检索的字符串值没有出现，则该方法返回 -1。

实例

在本例中，我们将在 "Hello world!" 字符串内进行不同的检索：

```
<script type="text/javascript">

var str="Hello world!"
document.write(str.lastIndexOf("Hello") + "<br />")
document.write(str.lastIndexOf("World") + "<br />")
document.write(str.lastIndexOf("world"))

</script>
```

以上代码的输出：

```
0
-1
6
```

JavaScript link() 方法

定义和用法

link() 方法用于把字符串显示为超链接。

语法

```
stringObject.link(url)
```

参数	描述
url	必需。规定要链接的 URL。

实例

在本例中，"Free Web Tutorials!" 将被显示为超链接：

```
<script type="text/javascript">

var str="Free Web Tutorials!"
document.write(str.link("http://www.w3school.com.cn"))

</script>
```

JavaScript localeCompare() 方法

定义和用法

用本地特定的顺序来比较两个字符串。

语法

```
stringObject.localeCompare(target)
```

参数	描述
target	要以本地特定的顺序与 stringObject 进行比较的字符串。

返回值

说明比较结果的数字。如果 stringObject 小于 target，则 localeCompare() 返回小于 0 的数。如果 stringObject 大于 target，则该方法返回大于 0 的数。如果两个字符串相等，或根据本地排序规则没有区别，该方法返回 0。

说明

把 < 和 > 运算符应用到字符串时，它们只用字符的 Unicode 编码比较字符串，而不考虑

当地的排序规则。以这种方法生成的顺序不一定是正确的。例如，在西班牙语中，其中字符“ch”通常作为出现在字母“c”和“d”之间的字符来排序。
localeCompare() 方法提供的比较字符串的方法，考虑了默认的本地排序规则。ECMAscript 标准并没有规定如何进行本地特定的比较操作，它只规定该函数采用底层操作系统提供的排序规则。

实例

在本例中，我们将用本地特定排序规则对字符串数组进行排序：

```
var str;
str.sort (function(a,b){return a.localeCompare(b)})
```

JavaScript match() 方法

定义和用法

match() 方法可在字符串内检索指定的值，或找到一个或多个正则表达式的匹配。
该方法类似 indexOf() 和 lastIndexOf()，但是它返回指定的值，而不是字符串的位置。

语法

```
stringObject.match(searchvalue)
stringObject.match(regexp)
```

参数	描述
searchvalue	必需。规定要检索的字符串值。
regexp	必需。规定要匹配的模式的 RegExp 对象。如果该参数不是 RegExp 对象，则需要首先把它传递给 RegExp 构造函数，将其转换为 RegExp 对象。

返回值

存放匹配结果的数组。该数组的内容依赖于 regexp 是否具有全局标志 g。

说明

match() 方法将检索字符串 stringObject，以找到一个或多个与 regexp 匹配的文本。这个方法的行为在很大程度上有赖于 regexp 是否具有标志 g。

如果 regexp 没有标志 g，那么 match() 方法就只能在 stringObject 中执行一次匹配。如果没有找到任何匹配的文本， match() 将返回 null。否则，它将返回一个数组，其中存放了与它找到的匹配文本有关的信息。该数组的第 0 个元素存放的是匹配文本，而其余的元素存放的是与正则表达式的子表达式匹配的文本。除了这些常规的数组元素之外，返回的数组还含有两个对象属性。index 属性声明的是匹配文本的起始字符在 stringObject 中的位置，input 属性声明的是对 stringObject 的引用。

如果 regexp 具有标志 g，则 match() 方法将执行全局检索，找到 stringObject 中的所有匹配子字符串。若没有找到任何匹配的子串，则返回 null。如果找到了一个或多个匹配子串，则返回一个数组。不过全局匹配返回的数组的内容与前者大不相同，它的数组元素中存放的是 stringObject 中所有的匹配子串，而且也没有 index 属性或 input 属性。

注意：在全局检索模式下，match() 即不提供与子表达式匹配的文本的信息，也不声明每个匹配子串的位置。如果您需要这些全局检索的信息，可以使用 RegExp.exec()。

实例

例子 1

在本例中，我们将在 "Hello world!" 中进行不同的检索：

```
<script type="text/javascript">

var str="Hello world!"
document.write(str.match("world") + "<br />")
document.write(str.match("World") + "<br />")
document.write(str.match("worlld") + "<br />")
document.write(str.match("world!"))

</script>
```

输出：

```
world
null
null
world!
```

例子 2

在本例中，我们将使用全局匹配的正则表达式来检索字符串中的所有数字：

```
<script type="text/javascript">

var str="1 plus 2 equal 3"
document.write(str.match(/\d+/g))

</script>
```

输出：

```
1,2,3
```

JavaScript replace() 方法

定义和用法

replace() 方法用于在字符串中用一些字符替换另一些字符，或替换一个与正则表达式匹配的子串。

语法

```
stringObject.replace(regexp/substr,replacement)
```

参数	描述
regexp/substr	必需。规定子字符串或要替换的模式的 RegExp 对象。 请注意，如果该值是一个字符串，则将它作为要检索的直接量文本模式，而不是首先被转换为 RegExp 对象。
replacement	必需。一个字符串值。规定了替换文本或生成替换文本的函数。

返回值

一个新的字符串，是用 *replacement* 替换了 regexp 的第一次匹配或所有匹配之后得到的。

说明

字符串 stringObject 的 replace() 方法执行的是查找并替换的操作。它将在 stringObject 中

查找与 regexp 相匹配的子字符串，然后用 *replacement* 来替换这些子串。如果 regexp 具有全局标志 g，那么 replace() 方法将替换所有匹配的子串。否则，它只替换第一个匹配子串。*replacement* 可以是字符串，也可以是函数。如果它是字符串，那么每个匹配都将由字符串替换。但是 replacement 中的 $ 字符具有特定的含义。如下表所示，它说明从模式匹配得到的字符串将用于替换。

字符	替换文本
$1、$2、...、$99	与 regexp 中的第 1 到第 99 个子表达式相匹配的文本。
$&	与 regexp 相匹配的子串。
$`	位于匹配子串左侧的文本。
$'	位于匹配子串右侧的文本。
$$	直接量符号。

注意：ECMAScript v3 规定，replace() 方法的参数 replacement 可以是函数而不是字符串。在这种情况下，每个匹配都调用该函数，它返回的字符串将作为替换文本使用。该函数的第一个参数是匹配模式的字符串。接下来的参数是与模式中的子表达式匹配的字符串，可以有 0 个或多个这样的参数。接下来的参数是一个整数，声明了匹配在 stringObject 中出现的位置。最后一个参数是 stringObject 本身。

实例

例子 1

在本例中，我们将使用 "W3School" 替换字符串中的 "Microsoft"：

```
<script type="text/javascript">

var str="Visit Microsoft!"
document.write(str.replace(/Microsoft/, "W3School"))

</script>
```

输出：

```
Visit W3School!
```

例子 2

在本例中，我们将执行一次全局替换，每当 "Microsoft" 被找到，它就被替换为 "W3School"：

```
<script type="text/javascript">

var str="Welcome to Microsoft! "
str=str + "We are proud to announce that Microsoft has "
str=str + "one of the largest Web Developers sites in the world."

document.write(str.replace(/Microsoft/g, "W3School"))

</script>
```

输出：

```
Welcome to W3School! We are proud to announce that W3School
has one of the largest Web Developers sites in the world.
```

例子 3

您可以使用本例提供的代码来确保匹配字符串大写字符的正确：

```
text = "javascript Tutorial";
text.replace(/javascript/i, "JavaScript");
```

例子 4

在本例中，我们将把 "Doe, John" 转换为 "John Doe" 的形式：

```
name = "Doe, John";
name.replace(/(\w+)\s*, \s*(\w+)/, "$2 $1");
```

例子 5

在本例中，我们将把所有的花引号替换为直引号：

```
name = '"a", "b"';
name.replace(/"([^"]*)"/g, "'$1'");
```

例子 6

在本例中，我们将把字符串中所有单词的首字母都转换为大写：

```
name = 'aaa bbb ccc';
uw=name.replace(/\b\w+\b/g, function(word){
  return word.substring(0,1).toUpperCase()+word.substring(1);}
```

```
  );
```

JavaScript search() 方法

定义和用法

search() 方法用于检索字符串中指定的子字符串，或检索与正则表达式相匹配的子字符串。

语法

```
stringObject.search(regexp)
```

参数	描述
regexp	该参数可以是需要在 stringObject 中检索的子串，也可以是需要检索的 RegExp 对象。 注释：要执行忽略大小写的检索，请追加标志 i。

返回值

stringObject 中第一个与 regexp 相匹配的子串的起始位置。
注释：如果没有找到任何匹配的子串，则返回 -1。

说明

search() 方法不执行全局匹配，它将忽略标志 g。它同时忽略 regexp 的 lastIndex 属性，并且总是从字符串的开始进行检索，这意味着它总是返回 stringObject 的第一个匹配的位置。

实例

例子 1

在本例中，我们将检索 "W3School"：

```
<script type="text/javascript">

var str="Visit W3School!"
```

```
document.write(str.search(/W3School/))

</script>
```

输出：

```
6
```

在下面的例子中，无法检索到 w3school（因为 search() 对大小写敏感）。

```
<script type="text/javascript">

var str="Visit W3School!"
document.write(str.search(/w3school/))

</script>
```

输出：

```
-1
```

例子 2

在本例中，我们将执行一次忽略大小写的检索：

```
<script type="text/javascript">

var str="Visit W3School!"
document.write(str.search(/w3school/i))

</script>
```

输出：

```
6
```

JavaScript slice() 方法

定义和用法

slice() 方法可提取字符串的某个部分，并以新的字符串返回被提取的部分。

语法

```
stringObject.slice(start,end)
```

参数	描述
start	要抽取的片断的起始下标。如果是负数，则该参数规定的是从字符串的尾部开始算起的位置。也就是说，-1 指字符串的最后一个字符，-2 指倒数第二个字符，以此类推。
end	紧接着要抽取的片段的结尾的下标。若未指定此参数，则要提取的子串包括 start 到原字符串结尾的字符串。如果该参数是负数，那么它规定的是从字符串的尾部开始算起的位置。

返回值

一个新的字符串。包括字符串 stringObject 从 start 开始（包括 start）到 end 结束（不包括 end）为止的所有字符。

说明

String 对象的方法 slice()、substring() 和 substr() （不建议使用）都可返回字符串的指定部分。slice() 比 substring() 要灵活一些，因为它允许使用负数作为参数。slice() 与 substr() 有所不同，因为它用两个字符的位置来指定子串，而 substr() 则用字符位置和长度来指定子串。
还要注意的是，String.slice() 与 Array.slice() 相似。

实例

例子 1

在本例中，我们将提取从位置 6 开始的所有字符：

```
<script type="text/javascript">

var str="Hello happy world!"
document.write(str.slice(6))

</script>
```

输出：

```
happy world!
```

例子 2

在本例中，我们将提取从位置 6 到位置 11 的所有字符：

```
<script type="text/javascript">

var str="Hello happy world!"
document.write(str.slice(6,11))

</script>
```

输出：

```
happy
```

JavaScript small() 方法

定义和用法

small() 方法用于把字符串显示为小号字。

语法

```
stringObject.small()
```

实例

在本例中，"Hello world!" 将显示为小号字：

```
<script type="text/javascript">

var str="Hello world!"
document.write(str.small())

</script>
```

JavaScript split() 方法

定义和用法

split() 方法用于把一个字符串分割成字符串数组。

语法

```
stringObject.split(separator,howmany)
```

参数	描述
separator	必需。字符串或正则表达式，从该参数指定的地方分割 stringObject。
howmany	可选。该参数可指定返回的数组的最大长度。如果设置了该参数，返回的子串不会多于这个参数指定的数组。如果没有设置该参数，整个字符串都会被分割，不考虑它的长度。

返回值

一个字符串数组。该数组是通过在 *separator* 指定的边界处将字符串 stringObject 分割成子串创建的。返回的数组中的字串不包括 *separator* 自身。
但是，如果 *separator* 是包含子表达式的正则表达式，那么返回的数组中包括与这些子表达式匹配的字串（但不包括与整个正则表达式匹配的文本）。

提示和注释

注释：如果把空字符串 ("") 用作 *separator*，那么 stringObject 中的每个字符之间都会被分割。
注释：String.split() 执行的操作与 Array.join 执行的操作是相反的。

实例

例子 1

在本例中，我们将按照不同的方式来分割字符串：

```
<script type="text/javascript">

var str="How are you doing today?"

document.write(str.split(" ") + "<br />")
document.write(str.split("") + "<br />")
document.write(str.split(" ",3))

</script>
```

输出：

```
How,are,you,doing,today?
H,o,w, ,a,r,e, ,y,o,u, ,d,o,i,n,g, ,t,o,d,a,y,?
How,are,you
```

例子 2

在本例中，我们将分割结构更为复杂的字符串：

```
"2:3:4:5".split(":")   //将返回["2", "3", "4", "5"]
"|a|b|c".split("|") //将返回["", "a", "b", "c"]
```

例子 3

使用下面的代码，可以把句子分割成单词：

```
var words = sentence.split(' ')
```

或者使用正则表达式作为 separator：

```
var words = sentence.split(/\s+/)
```

例子 4

如果您希望把单词分割为字母，或者把字符串分割为字符，可使用下面的代码：

```
"hello".split("")   //可返回 ["h", "e", "l", "l", "o"]
```

若只需要返回一部分字符，请使用 howmany 参数：

```
"hello".split("", 3)    //可返回 ["h", "e", "l"]
```

JavaScript strike() 方法

定义和用法

strike() 方法用于显示加删除线的字符串。

语法

```
stringObject.strike()
```

实例

在本例中，"Hello world!" 将被加上一条删除线：

```
<script type="text/javascript">

var str="Hello world!"
document.write(str.strike())

</script>
```

JavaScript sub() 方法

定义和用法

sub() 方法用于把字符串显示为下标。

语法

```
stringObject.sub()
```

实例

在本例中，"Hello world!" 将被显示为下标：

```
<script type="text/javascript">

var str="Hello world!"
document.write(str.sub())

</script>
```

JavaScript substr() 方法

定义和用法

substr() 方法可在字符串中抽取从 *start* 下标开始的指定数目的字符。

语法

```
stringObject.substr(start,length)
```

参数	描述
start	必需。要抽取的子串的起始下标。必须是数值。如果是负数，那么该参数声明从字符串的尾部开始算起的位置。也就是说，-1 指字符串中最后一个字符，-2 指倒数第二个字符，以此类推。
length	可选。子串中的字符数。必须是数值。如果省略了该参数，那么返回从 *stringObject* 的开始位置到结尾的字串。

返回值

一个新的字符串，包含从 *stringObject* 的 *start*（包括 start 所指的字符） 处开始的 *length* 个字符。如果没有指定 *length*，那么返回的字符串包含从 *start* 到 *stringObject* 的结

尾的字符。

提示和注释

注释：substr() 的参数指定的是子串的开始位置和长度，因此它可以替代 substring() 和 slice() 来使用。
重要事项：ECMAscript 没有对该方法进行标准化，因此反对使用它。
重要事项：在 IE 4 中，参数 *start* 的值无效。在这个 BUG 中，*start* 规定的是第 0 个字符的位置。在之后的版本中，此 BUG 已被修正。

实例

例子 1

在本例中，我们将使用 substr() 从字符串中提取一些字符：

```
<script type="text/javascript">

var str="Hello world!"
document.write(str.substr(3))

</script>
```

输出：

```
lo world!
```

例子 2

在本例中，我们将使用 substr() 从字符串中提取一些字符：

```
<script type="text/javascript">

var str="Hello world!"
document.write(str.substr(3,7))

</script>
```

输出：

```
lo worl
```

JavaScript substring() 方法

定义和用法

substring() 方法用于提取字符串中介于两个指定下标之间的字符。

语法

```
stringObject.substring(start,stop)
```

参数	描述
start	必需。一个非负的整数，规定要提取的子串的第一个字符在 stringObject 中的位置。
stop	可选。一个非负的整数，比要提取的子串的最后一个字符在 stringObject 中的位置多 1。 如果省略该参数，那么返回的子串会一直到字符串的结尾。

返回值

一个新的字符串，该字符串值包含 *stringObject* 的一个子字符串，其内容是从 *start* 处到 *stop*-1 处的所有字符，其长度为 *stop* 减 *start*。

说明

substring() 方法返回的子串包括 *start* 处的字符，但不包括 *stop* 处的字符。
如果参数 *start* 与 *stop* 相等，那么该方法返回的就是一个空串（即长度为 0 的字符串）。如果 *start* 比 *stop* 大，那么该方法在提取子串之前会先交换这两个参数。

提示和注释

重要事项：与 slice() 和 substr() 方法不同的是，substring() 不接受负的参数。

实例

例子 1

在本例中，我们将使用 substring() 从字符串中提取一些字符：

```
<script type="text/javascript">

var str="Hello world!"
document.write(str.substring(3))

</script>
```

输出：

```
lo world!
```

例子 2

在本例中，我们将使用 substring() 从字符串中提取一些字符：

```
<script type="text/javascript">

var str="Hello world!"
document.write(str.substring(3,7))

</script>
```

输出：

```
lo w
```

JavaScript sup() 方法

定义和用法

sup() 方法用于把字符串显示为上标。

语法

```
stringObject.sup()
```

实例

在本例中，"Hello world!" 将被显示为上标：

```
<script type="text/javascript">

var str="Hello world!"
document.write(str.sup())

</script>
```

JavaScript toLocaleLowerCase() 方法

定义和用法

toLocaleLowerCase() 方法用于把字符串转换为小写。

语法

```
stringObject.toLocaleLowerCase()
```

返回值

一个新的字符串，在其中 stringObject 的所有大写字符全部被转换为了小写字符。

说明

与 toLowerCase() 不同的是，toLocaleLowerCase() 方法按照本地方式把字符串转换为小写。只有几种语言（如土耳其语）具有地方特有的大小写映射，所有该方法的返回值通常与 toLowerCase() 一样。

实例

在本例中，"Hello world!" 将以小写字母来显示：

```
<script type="text/javascript">
```

```
var str="Hello World!"
document.write(str.toLocaleLowerCase())

</script>
```

JavaScript toLocaleUpperCase() 方法

定义和用法

toLocaleUpperCase() 方法用于把字符串转换为大写。

语法

```
stringObject.toLocaleUpperCase()
```

返回值

一个新的字符串，在其中 stringObject 的所有小写字符全部被转换为了大写字符。

说明

与 toUpperCase() 不同的是，toLocaleUpperCase() 方法按照本地方式把字符串转换为大写。只有几种语言（如土耳其语）具有地方特有的大小写映射，所有该方法的返回值通常与 toUpperCase() 一样。

实例

在本例中，"Hello world!" 将以大写字母来显示：

```
<script type="text/javascript">

var str="Hello World!"
document.write(str.toLocaleUpperCase())

</script>
```

JavaScript toLowerCase() 方法

定义和用法

toLowerCase() 方法用于把字符串转换为小写。

语法

```
stringObject.toLowerCase()
```

返回值

一个新的字符串，在其中 stringObject 的所有大写字符全部被转换为了小写字符。

实例

在本例中，"Hello world!" 将以小写字母来显示：

```
<script type="text/javascript">

var str="Hello World!"
document.write(str.toLowerCase())

</script>
```

JavaScript toUpperCase() 方法

定义和用法

toUpperCase() 方法用于把字符串转换为大写。

语法

```
stringObject.toUpperCase()
```

返回值

一个新的字符串，在其中 stringObject 的所有小写字符全部被转换为了大写字符。

实例

在本例中，"Hello world!" 将以大写字母来显示：

```
<script type="text/javascript">

var str="Hello World!"
document.write(str.toUpperCase())

</script>
```

JavaScript toString() 方法

定义和用法

toString() 方法返回字符串。

语法

```
stringObject.toString()
```

返回值

stringObject 的原始字符串值。一般不会调用该方法。

抛出

当调用该方法的对象不是 String 时抛出 TypeError 异常。

参阅

array.toString() 、 boolean.toString() 、 date.toString() 、 Number.toString()
array.toLocaleString() 、 date.toLocaleString() 、 Number.toLocaleString()

stringObject.valueOf()

课外书

如需更多有关类型转换的知识，请阅读 JavaScript 高级教程中的相关内容：

ECMAScript 类型转换
本节讲解了 ECMAScript 提供的类型转换方法，以及如何进行强制类型转换。

JavaScript valueOf() 方法

定义和用法

valueOf() 方法可返回 String 对象的原始值。
原始值是由从 String 对象下来的所有对象继承的。
valueOf() 方法通常由 JavaScript 在后台自动进行调用，而不是显式地处于代码中。

语法

```
stringObject.valueOf()
```

抛出

当调用该方法的对象不是 String 时抛出 TypeError 异常。

JavaScript RegExp 对象参考手册

RegExp 对象

RegExp 对象表示正则表达式，它是对字符串执行模式匹配的强大工具。

直接量语法

```
/pattern/attributes
```

创建 RegExp 对象的语法：

```
new RegExp(pattern, attributes);
```

参数

参数 *pattern* 是一个字符串，指定了正则表达式的模式或其他正则表达式。
参数 *attributes* 是一个可选的字符串，包含属性 "g"、"i" 和 "m"，分别用于指定全局匹配、区分大小写的匹配和多行匹配。ECMAScript 标准化之前，不支持 m 属性。如果 *pattern* 是正则表达式，而不是字符串，则必须省略该参数。

返回值

一个新的 RegExp 对象，具有指定的模式和标志。如果参数 *pattern* 是正则表达式而不是字符串，那么 RegExp() 构造函数将用与指定的 RegExp 相同的模式和标志创建一个新的 RegExp 对象。
如果不用 new 运算符，而将 RegExp() 作为函数调用，那么它的行为与用 new 运算符调用时一样，只是当 *pattern* 是正则表达式时，它只返回 *pattern*，而不再创建一个新的 RegExp 对象。

抛出

SyntaxError - 如果 *pattern* 不是合法的正则表达式，或 *attributes* 含有 "g"、"i" 和 "m" 之外的字符，抛出该异常。
TypeError - 如果 *pattern* 是 RegExp 对象，但没有省略 *attributes* 参数，抛出该异常。

修饰符

修饰符	描述
i	执行对大小写不敏感的匹配。
g	执行全局匹配（查找所有匹配而非在找到第一个匹配后停止）。
m	执行多行匹配。

方括号

方括号用于查找某个范围内的字符：

表达式	描述
[abc]	查找方括号之间的任何字符。
[^abc]	查找任何不在方括号之间的字符。
[0-9]	查找任何从 0 至 9 的数字。
[a-z]	查找任何从小写 a 到小写 z 的字符。
[A-Z]	查找任何从大写 A 到大写 Z 的字符。
[A-z]	查找任何从大写 A 到小写 z 的字符。
[adgk]	查找给定集合内的任何字符。
[^adgk]	查找给定集合外的任何字符。
(red|blue|green)	查找任何指定的选项。

元字符

元字符（Metacharacter）是拥有特殊含义的字符：

元字符	描述
.	查找单个字符，除了换行和行结束符。
\w	查找单词字符。
\W	查找非单词字符。
\d	查找数字。
\D	查找非数字字符。
\s	查找空白字符。
\S	查找非空白字符。
\b	匹配单词边界。
\B	匹配非单词边界。
\0	查找 NUL 字符。
\n	查找换行符。
\f	查找换页符。
\r	查找回车符。
\t	查找制表符。

\v	查找垂直制表符。
\xxx	查找以八进制数 xxx 规定的字符。
\xdd	查找以十六进制数 dd 规定的字符。
\uxxxx	查找以十六进制数 xxxx 规定的 Unicode 字符。

量词

量词	描述
n+	匹配任何包含至少一个 n 的字符串。
n*	匹配任何包含零个或多个 n 的字符串。
n?	匹配任何包含零个或一个 n 的字符串。
n{X}	匹配包含 X 个 n 的序列的字符串。
n{X,Y}	匹配包含 X 或 Y 个 n 的序列的字符串。
n{X,}	匹配包含至少 X 个 n 的序列的字符串。
n$	匹配任何结尾为 n 的字符串。
^n	匹配任何开头为 n 的字符串。
?=n	匹配任何其后紧接指定字符串 n 的字符串。
?!n	匹配任何其后没有紧接指定字符串 n 的字符串。

RegExp 对象属性

FF: Firefox, IE: Internet Explorer

属性	描述	FF	IE
global	RegExp 对象是否具有标志 g。	1	4
ignoreCase	RegExp 对象是否具有标志 i。	1	4
lastIndex	一个整数，标示开始下一次匹配的字符位置。	1	4
multiline	RegExp 对象是否具有标志 m。	1	4
source	正则表达式的源文本。	1	4

RegExp 对象方法

FF: Firefox, IE: Internet Explorer

方法	描述	FF	IE
compile	编译正则表达式。	1	4
exec	检索字符串中指定的值。返回找到的值，并确定其位置。	1	4
test	检索字符串中指定的值。返回 true 或 false。	1	4

支持正则表达式的 String 对象的方法

FF: Firefox, IE: Internet Explorer

方法	描述	FF	IE
search	检索与正则表达式相匹配的值。	1	4
match	找到一个或多个正则表达式的匹配。	1	4
replace	替换与正则表达式匹配的子串。	1	4
split	把字符串分割为字符串数组。	1	4

JavaScript RegExp i 修饰符

定义和用法

i 修饰符用于执行对大小写不敏感的匹配。

语法

```
new RegExp("regexp","i")
```

直接量语法：

```
/regexp/i
```

浏览器支持

所有主流浏览器都支持 i 修饰符。

实例

对字符串中的 "w3school" 进行不分大小写的搜索：

```
var str="Visit W3School";
var patt1=/w3school/i;
```

下面被标记的文本显示了表达式获得匹配的位置：

```
Visit W3School
```

JavaScript RegExp g 修饰符

定义和用法

g 修饰符用于执行全局匹配（查找所有匹配而非在找到第一个匹配后停止）。

语法

```
new RegExp("regexp","g")
```

直接量语法：

```
/regexp/g
```

浏览器支持

所有主流浏览器都支持 g 修饰符。

实例

例子 1

对 "is" 进行全局搜索：

```
var str="Is this all there is?";
var patt1=/is/g;
```

下面被标记的文本显示了表达式获得匹配的位置：

```
Is this all there is?
```

例子 2

对 "is" 进行全局且大小写不敏感的搜索：

```
var str="Is this all there is?";
var patt1=/is/gi;
```

下面被标记的文本显示了表达式获得匹配的位置：

```
Is this all there is?
```

JavaScript RegExp [abc] 表达式

定义和用法

[abc] 表达式用于查找方括号之间的任何字符。
方括号内的字符可以是任何字符或字符范围。

语法

```
new RegExp("[abc]")
```

直接量语法：

```
/[abc]/
```

浏览器支持

所有主流浏览器都支持 [abc] 表达式。

实例

在字符串中对字符范围 [a-h] 进行全局搜索：

```
var str="Is this all there is?";
var patt1=/[a-h]/g;
```

下面被标记的文本显示了表达式获得匹配的位置：

```
Is this all there is?
```

JavaScript RegExp [^abc] 表达式

定义和用法

[^abc] 表达式用于查找任何不在方括号之间的字符。
方括号内的字符可以是任何字符或字符范围。

语法

```
new RegExp("[^xyz]")
```

直接量语法：

```
/[^xyz]/
```

浏览器支持

所有主流浏览器都支持 [^abc] 表达式。

实例

对不在字符范围 [a-h] 内的字符进行全局搜索：

```
var str="Is this all there is?";
var patt1=/[^a-h]/g;
```

下面被标记的文本显示了表达式获得匹配的位置：

```
Is this all there is?
```

JavaScript RegExp . 元字符

定义和用法

. 元字符用于查找单个字符，除了换行和行结束符。

语法

```
new RegExp("regexp.")
```

直接量语法：

```
/regexp./
```

浏览器支持

所有主流浏览器都支持 . 元字符。

实例

对字符串中的 "h.t" 进行全局搜索：

```
var str="That's hot!";
var patt1=/h.t/g;
```

下面被标记的文本显示了表达式获得匹配的位置：

```
That's hot!
```

JavaScript RegExp \w 元字符

定义和用法

\w 元字符用于查找单词字符。

语法

```
new RegExp("\w")
```

直接量语法：

```
/\w/
```

浏览器支持

所有主流浏览器都支持 \w 元字符。

实例

对字符串中的单词字符进行全局搜索：

```
var str="Give 100%!";
var patt1=/\w/g;
```

下面被标记的文本显示了表达式获得匹配的位置：

```
Give 100%!
```

JavaScript RegExp \W 元字符

定义和用法

\W 元字符用于查找非单词字符。
单词字符包括：a-z、A-Z、0-9，以及下划线。

语法

```
new RegExp("\W")
```

直接量语法：

```
/\W/
```

浏览器支持

所有主流浏览器都支持 \W 元字符。

实例

对字符串中的非单词字符进行全局搜索：

```
var str="Give 100%!";
var patt1=/\W/g;
```

下面被标记的文本显示了表达式获得匹配的位置：

```
Give 100%!
```

JavaScript RegExp \d 元字符

定义和用法

\d 元字符用于查找数字字符。

语法

```
new RegExp("\d")
```

直接量语法：

```
/\d/
```

浏览器支持

所有主流浏览器都支持 \d 元字符。

实例

对数字进行全局搜索：

```
var str="Give 100%!";
var patt1=/\d/g;
```

下面被标记的文本显示了表达式获得匹配的位置：

```
Give 100%!
```

JavaScript RegExp \D 元字符

定义和用法

\D 元字符用于查找非数字字符。

语法

```
new RegExp("\D")
```

直接量语法：

```
/\D/
```

浏览器支持

所有主流浏览器都支持 \D 元字符。

实例

对字符串中的非数字字符进行全局搜索：

```
var str="Give 100%!";
var patt1=/\D/g;
```

下面被标记的文本显示了表达式获得匹配的位置：

```
Give 100%!
```

JavaScript RegExp \s 元字符

定义和用法

\s 元字符用于查找空白字符。
空白字符可以是：

- 空格符 (space character)
- 制表符 (tab character)
- 回车符 (carriage return character)
- 换行符 (new line character)
- 垂直换行符 (vertical tab character)
- 换页符 (form feed character)

语法

```
new RegExp("\s")
```

直接量语法：

```
/\s/
```

浏览器支持

所有主流浏览器都支持 \s 元字符。

实例

对字符串中的空白字符进行全局搜索：

```
var str="Is this all there is?";
var patt1=/\s/g;
```

下面被标记的文本显示了表达式获得匹配的位置：

```
Is this all there is?
```

JavaScript RegExp \S 元字符

定义和用法

\S 元字符用于查找非空白字符。
空白字符可以是：

- 空格符 (space character)
- 制表符 (tab character)
- 回车符 (carriage return character)
- 换行符 (new line character)
- 垂直换行符 (vertical tab character)
- 换页符 (form feed character)

语法

```
new RegExp("\S")
```

直接量语法：

```
/\S/
```

浏览器支持

所有主流浏览器都支持 \S 元字符。

实例

对字符串中的非空白字符进行全局搜索：

```
var str="Is this all there is?";
var patt1=/\S/g;
```

下面被标记的文本显示了表达式获得匹配的位置：

```
Is this all there is?
```

JavaScript RegExp \b 元字符

定义和用法

\b 元字符匹配单词边界。
在单词边界匹配的位置，单词字符后面或前面不与另一个单词字符直接相邻。请注意，匹配的单词边界并不包含在匹配中。换句话说，匹配的单词边界的长度为零。（不要与 [\b] 混淆。）
如果未找到匹配，则返回 null。
提示：\b 元字符通常用于查找位于单词的开头或结尾的匹配。

例子：

/\bm/ 匹配 "moon" 中的 'm'；
/oo\b/ 不匹配 "moon" 中的 'oo'，因为 'oo' 后面的 'n' 是一个单词字符；
/oon\b/ 匹配 "moon" 中的 'oon'，因为 'oon' 位于字符串的末端，后面没有单词字符；
/\w\b\w/ 不匹配任何字符，因为单词字符之后绝不会同时紧跟着非单词字符和单词字符。
亲自试一试

语法

```
new RegExp("\bregexp")
```

直接量语法：

```
/\bregexp/
```

浏览器支持

所有主流浏览器都支持 \b 元字符。

实例

对字符串中的单词的开头或结尾进行 "W3" 的全局搜索：

```
var str="Visit W3School";
```

```
var patt1=/\bW3/g;
```

下面被标记的文本显示了表达式获得匹配的位置：

```
Visit W3School
```

JavaScript RegExp \B 元字符

定义和用法

\B 元字符匹配非单词边界。匹配位置的上一个和下一个字符的类型是相同的：即必须同时是单词，或必须同时是非单词字符。字符串的开头和结尾处被视为非单词字符。
如果未找到匹配，则返回 null。
提示：\B 元字符通常用于查找不处在单词的开头或结尾的匹配。
例如：/\B../ 匹配 "noonday" 中的 'oo'，而 /y\B./ 匹配 "possibly yesterday." 中的 'ye'。

语法

```
new RegExp("\Bregexp")
```

直接量语法：

```
/\Bregexp/
```

浏览器支持

所有主流浏览器都支持 \B 元字符。

实例

对字符串中不位于单词开头或结尾的 "School" 进行全局搜索：

```
var str="Visit W3School";
var patt1=/\BSchool/g;
```

下面被标记的文本显示了表达式获得匹配的位置：

```
Visit W3School
```

JavaScript RegExp \n 元字符

定义和用法

\n 元字符用于查找换行符。
\n 返回换行符被找到的位置。如果未找到匹配，则返回 -1。

语法

```
new RegExp("\n")
```

直接量语法：

```
/\n/
```

浏览器支持

所有主流浏览器都支持 \n 元字符。

实例

搜索字符串中的换行字符：

```
var str="Visit W3School.\nLearn Javascript.";
var patt1=/\n/;
```

下面被标记的文本显示了表达式获得匹配的位置：

```
Visit W3School.\nLearn Javascript.
```

JavaScript RegExp \xxx 元字符

定义和用法

\xxx 元字符用于查找以八进制数 *xxx* 规定的字符。

如果未找到匹配，则返回 null。

语法

```
new RegExp("\xxx")
```

直接量语法：

```
/\xxx/
```

浏览器支持

所有主流浏览器都支持 \xxx 元字符。

实例

对字符串中的八进制 127 (W) 进行全局搜索：

```
var str="Visit W3School. Hello World!";
var patt1=/\127/g;
```

下面被标记的文本显示了表达式获得匹配的位置：

```
Visit W3School. Hello World!
```

JavaScript RegExp \xdd 元字符

定义和用法

\xdd 元字符查找以十六进制数 *dd* 规定的字符。
如果未找到匹配，则返回 null。

语法

```
new RegExp("\xdd")
```

直接量语法：

```
/\xdd/
```

浏览器支持

所有主流浏览器都支持 \xdd 元字符。

实例

对字符串中的十六进制 57 (W) 进行全局搜索：

```
var str="Visit W3School. Hello World!";
var patt1=/\x57/g;
```

下面被标记的文本显示了表达式获得匹配的位置：

```
Visit W3School. Hello World!
```

JavaScript RegExp \uxxxx 元字符

定义和用法

\uxxxx 元字符用于查找以十六进制数 *xxxx* 规定的 Unicode 字符。
如果未找到匹配，则返回 null。

语法

```
new RegExp("\uxxxx")
```

直接量语法：

```
/\uxxxx/
```

浏览器支持

所有主流浏览器都支持 \uxxxx 元字符。

实例

对字符串中的十六进制 0057 (W) 进行全局搜索：

```
var str="Visit W3School. Hello World!";
var patt1=/\u0057/g;
```

下面被标记的文本显示了表达式获得匹配的位置：

```
Visit W3School. Hello World!
```

JavaScript RegExp + 量词

定义和用法

n+ 量词匹配包含至少一个 *n* 的任何字符串。

语法

```
new RegExp("n+")
```

直接量语法：

```
/n+/
```

浏览器支持

所有主流浏览器都支持 + 量词。

实例

例子 1

对至少一个 "o" 进行全局搜索：

```
var str="Hellooo World! Hello W3School!";
var patt1=/o+/g;
```

下面被标记的文本显示了表达式获得匹配的位置：

```
Hellooo World! Hello W3School!
```

例子 2

对至少一个单词字符进行全局搜索：

```
var str="Hello World! Hello W3School!";
var patt1=/\w+/g;
```

下面被标记的文本显示了表达式获得匹配的位置：

```
Hello World! Hello W3School!
```

JavaScript RegExp * 量词

定义和用法

n+ 量词匹配包含零个或多个 *n* 的任何字符串。

语法

```
new RegExp("n*")
```

直接量语法：

```
/n*/
```

浏览器支持

所有主流浏览器都支持 * 量词。

实例

例子 1

对 "l" 进行全局搜索，包括其后紧跟的一个或多个 "o"：

```
var str="Hellooo World! Hello W3School!";
var patt1=/lo*/g;
```

下面被标记的文本显示了表达式获得匹配的位置：

```
Hellooo World! Hello W3School!
```

例子 2

对 "1" 进行全局搜索，包括其后紧跟的一个或多个 "0"：

```
var str="1, 100 or 1000?";
var patt1=/10*/g;
```

下面被标记的文本显示了表达式获得匹配的位置：

```
1, 100 or 1000?
```

JavaScript RegExp ？ 量词

定义和用法

n？ 量词匹配任何包含零个或一个 *n* 的字符串。

语法

```
new RegExp("n?")
```

直接量语法：

```
/n?/
```

浏览器支持

所有主流浏览器都支持 ？ 量词。

实例

对 "1" 进行全局搜索，包括其后紧跟的零个或一个 "0"：

```
var str="1, 100 or 1000?";
var patt1=/10?/g;
```

下面被标记的文本显示了表达式获得匹配的位置：

```
1, 100 or 1000?
```

JavaScript RegExp {X} 量词

定义和用法

n{*X*,} 量词匹配包含 X 个 *n* 的序列的字符串。
X 必须是数字。

语法

```
new RegExp("n{X}")
```

直接量语法：

```
/n{X}/
```

浏览器支持

所有主流浏览器都支持 {X} 量词。

实例

对包含四位数字序列的子串进行全局搜索：

```
var str="100, 1000 or 10000?";
var patt1=/\d{4}/g;
```

下面被标记的文本显示了表达式获得匹配的位置：

```
100, 1000 or 10000?
```

JavaScript RegExp {X,Y} 量词

定义和用法

n{*X*,*Y*} 量词匹配包含 X 或 Y 个 *n* 的序列的字符串。
X 和 *Y* 必须是数字。

语法

```
new RegExp("n{X,Y}")
```

直接量语法：

```
/n{X,Y}/
```

浏览器支持

所有主流浏览器都支持 {X,Y} 量词。

实例

对包含三位或四位数字序列的子串进行全局搜索：

```
var str="100, 1000 or 10000?";
var patt1=/\d{3,4}/g;
```

下面被标记的文本显示了表达式获得匹配的位置：

```
100, 1000 or 10000?
```

JavaScript RegExp {X,} 量词

定义和用法

n{*X*,} 量词匹配包含至少 X 个 *n* 的序列的字符串。
X 必须是数字。

语法

```
new RegExp("n{X,}")
```

直接量语法：

```
/n{X,}/
```

浏览器支持

所有主流浏览器都支持 {X,} 量词。

实例

对包含至少三位数字序列的子串进行全局搜索：

```
var str="100, 1000 or 10000?";
var patt1=/\d{3,}/g;
```

下面被标记的文本显示了表达式获得匹配的位置：

```
100, 1000 or 10000?
```

JavaScript RegExp $ 量词

定义和用法

n$ 量词匹配任何结尾为 *n* 的字符串。

语法

```
new RegExp("n$")
```

直接量语法：

```
/n$/
```

浏览器支持

所有主流浏览器都支持 $ 量词。

实例

对字符串结尾的 "is" 进行全局搜索：

```
var str="Is this his";
var patt1=/is$/g;
```

下面被标记的文本显示了表达式获得匹配的位置：

```
Is this his
```

JavaScript RegExp ^ 量词

定义和用法

n^ 量词匹配任何开头为 *n* 的字符串。

语法

```
new RegExp("^n")
```

直接量语法：

```
/^n/
```

浏览器支持

所有主流浏览器都支持 ^ 量词。

实例

对字符串开头的 "is" 进行全局搜索：

```
var str="Is this his";
var patt1=/^Is/g;
```

下面被标记的文本显示了表达式获得匹配的位置：

```
Is this his
```

JavaScript RegExp ?= 量词

定义和用法

?=*n* 量词匹配任何其后紧接指定字符串 *n* 的字符串。

语法

```
new RegExp("regexp(?=n)")
```

直接量语法：

```
/regexp(?=n)/
```

浏览器支持

所有主流浏览器都支持 ?= 量词。

实例

对其后紧跟 "all" 的 "is" 进行全局搜索：

```
var str="Is this all there is";
var patt1=/is(?= all)/g;
```

下面被标记的文本显示了表达式获得匹配的位置：

```
Is this all there is
```

JavaScript RegExp ?! 量词

定义和用法

?!*n* 量词匹配其后没有紧接指定字符串 *n* 的任何字符串。

语法

```
new RegExp("regexp(?!n)")
```

直接量语法：

```
/regexp(?!n)/
```

浏览器支持

所有主流浏览器都支持 ?! 量词。

实例

对其后没有紧跟 "all" 的 "is" 进行全局搜索：

```
var str="Is this all there is";
var patt1=/is(?! all)/gi;
```

下面被标记的文本显示了表达式获得匹配的位置：

```
Is this all there is
```

JavaScript global 属性

定义和用法

global 属性用于返回正则表达式是否具有标志 "g"。
它声明了给定的正则表达式是否执行全局匹配。
如果 g 标志被设置，则该属性为 true，否则为 false。

语法

```
RegExpObject.global
```

实例

在下面的例子中，我们将查看正则表达式是否具有标志 g：

```
<script type="text/javascript">
var str = "Visit W3School.com.cn";
var patt1 = new RegExp("W3");

if(patt1.global)
  {
  alert("Global property is set");
  }
else
  {
  alert("Global property is NOT set.");
  }
</script>
```

JavaScript ignoreCase 属性

定义和用法

ignoreCase 属性规定是否设置 "i" 标志。
如果设置了 "i" 标志，则返回 true，否则返回 false。

语法

```
RegExpObject.ignoreCase
```

实例

在下面的例子中，我们将查看正则表达式是否具有标志 i：

```
<script type="text/javascript">
var str = "Visit W3School";
var patt1 = new RegExp("W3");

if(patt1.ignoreCase)
  {
  alert("ignoreCase property is set");
  }
else
  {
  alert("ignoreCase property is NOT set.");
  }
</script>
```

JavaScript lastIndex 属性

定义和用法

lastIndex 属性用于规定下次匹配的起始位置。

语法

```
RegExpObject.lastIndex
```

说明

该属性存放一个整数，它声明的是上一次匹配文本之后的第一个字符的位置。
上次匹配的结果是由方法 RegExp.exec() 和 RegExp.test() 找到的，它们都以 lastIndex 属性所指的位置作为下次检索的起始点。这样，就可以通过反复调用这两个方法来遍历一个字符串中的所有匹配文本。

该属性是可读可写的。只要目标字符串的下一次搜索开始，就可以对它进行设置。当方法 exec() 或 test() 再也找不到可以匹配的文本时，它们会自动把 lastIndex 属性重置为 0。

提示和注释

重要事项：不具有标志 g 和不表示全局模式的 RegExp 对象不能使用 lastIndex 属性。
提示：如果在成功地匹配了某个字符串之后就开始检索另一个新的字符串，需要手动地把这个属性设置为 0。

实例

在下面的例子中，我们将输出匹配完成之后的 lastIndex 属性：

```
<script type="text/javascript">
var str = "The rain in Spain stays mainly in the plain";
var patt1 = new RegExp("ain", "g");

patt1.test(str)
document.write("Match found. index now at: " + patt1.lastIndex);
</script>
```

JavaScript multiline 属性

定义和用法

multiline 属性用于返回正则表达式是否具有标志 m。
它声明了给定的正则表达式是否以多行模式执行模式匹配。
在这种模式中，如果要检索的字符串中含有换行符，^ 和 $ 锚除了匹配字符串的开头和结尾外还匹配每行的开头和结尾。
例如，模式 /W3School$/im 匹配 "w3school" 和 "W3School\nisgreat"。（\n 是换行符 \u000A）。
如果 m 标志被设置，则该属性为 true，否则为 false。

语法

```
RegExpObject.multiline
```

实例

在下面的例子中，我们将查看正则表达式是否具有标志 m：

```
<script type="text/javascript">
var str = "Visit W3School.com.cn";
var patt1 = new RegExp("W3","m");

if(patt1.multiline)
  {
  alert("Multiline property is set");
  }
else
  {
  alert("Multiline property is NOT set.");
  }
</script>
```

JavaScript source 属性

定义和用法

source 属性用于返回模式匹配所用的文本。
该文本不包括正则表达式直接量使用的定界符，也不包括标志 g、i、m。

语法

```
RegExpObject.source
```

实例

在下面的例子中，我们将获得用于模式匹配的文本：

```
<script type="text/javascript">
var str = "Visit W3School.com.cn";
var patt1 = new RegExp("W3S","g");

document.write("The regular expression is: " + patt1.source);
```

```
</script>
```

JavaScript compile() 方法

定义和用法

compile() 方法用于在脚本执行过程中编译正则表达式。
compile() 方法也可用于改变和重新编译正则表达式。

语法

```
RegExpObject.compile(regexp,modifier)
```

参数	描述
regexp	正则表达式。
modifier	规定匹配的类型。"g" 用于全局匹配，"i" 用于区分大小写，"gi" 用于全局区分大小写的匹配。

实例

在字符串中全局搜索 "man"，并用 "person" 替换。然后通过 compile() 方法，改变正则表达式，用 "person" 替换 "man" 或 "woman"，：

```
<script type="text/javascript">

var str="Every man in the world! Every woman on earth!";

patt=/man/g;
str2=str.replace(patt,"person");
document.write(str2+"<br />");

patt=/(wo)?man/g;
patt.compile(patt);
str2=str.replace(patt,"person");
document.write(str2);

</script>
```

输出：

```
Every person in the world! Every woperson on earth!
Every person in the world! Every person on earth!
```

JavaScript exec() 方法

定义和用法

exec() 方法用于检索字符串中的正则表达式的匹配。

语法

```
RegExpObject.exec(string)
```

参数	描述
string	必需。要检索的字符串。

返回值

返回一个数组，其中存放匹配的结果。如果未找到匹配，则返回值为 null。

说明

exec() 方法的功能非常强大，它是一个通用的方法，而且使用起来也比 test() 方法以及支持正则表达式的 String 对象的方法更为复杂。
如果 exec() 找到了匹配的文本，则返回一个结果数组。否则，返回 null。此数组的第 0 个元素是与正则表达式相匹配的文本，第 1 个元素是与 RegExpObject 的第 1 个子表达式相匹配的文本（如果有的话），第 2 个元素是与 RegExpObject 的第 2 个子表达式相匹配的文本（如果有的话），以此类推。除了数组元素和 length 属性之外，exec() 方法还返回两个属性。index 属性声明的是匹配文本的第一个字符的位置。input 属性则存放的是被检索的字符串 string。我们可以看得出，在调用非全局的 RegExp 对象的 exec() 方法时，返回的数组与调用方法 String.match() 返回的数组是相同的。
但是，当 RegExpObject 是一个全局正则表达式时，exec() 的行为就稍微复杂一些。它会在 RegExpObject 的 lastIndex 属性指定的字符处开始检索字符串 string。当 exec() 找到了与表达式相匹配的文本时，在匹配后，它将把 RegExpObject 的 lastIndex 属性设置为匹配文本的最后一个字符的下一个位置。这就是说，您可以通过反复调用 exec() 方法来遍历字符串中的所有匹配文本。当 exec() 再也找不到匹配的文本时，它将返回 null，并把 lastIndex 属

性重置为 0。

提示和注释

重要事项：如果在一个字符串中完成了一次模式匹配之后要开始检索新的字符串，就必须手动地把 lastIndex 属性重置为 0。
提示：请注意，无论 RegExpObject 是否是全局模式，exec() 都会把完整的细节添加到它返回的数组中。这就是 exec() 与 String.match() 的不同之处，后者在全局模式下返回的信息要少得多。因此我们可以这么说，在循环中反复地调用 exec() 方法是唯一一种获得全局模式的完整模式匹配信息的方法。

实例

在本例中，我们将全局检索字符串中的 W3School：

```
<script type="text/javascript">

var str = "Visit W3School";
var patt = new RegExp("W3School","g");
var result;

while ((result = patt.exec(str)) != null)  {
  document.write(result);
  document.write("<br />");
  document.write(patt.lastIndex);
 }
</script>
```

输出：

```
W3School
14
```

JavaScript test() 方法

定义和用法

test() 方法用于检测一个字符串是否匹配某个模式.

语法

```
RegExpObject.test(string)
```

参数	描述
string	必需。要检测的字符串。

返回值

如果字符串 string 中含有与 RegExpObject 匹配的文本，则返回 true，否则返回 false。

说明

调用 RegExp 对象 r 的 test() 方法，并为它传递字符串 s，与这个表示式是等价的：(r.exec(s) != null)。

实例

在下面的例子中，我们将检索 "W3School"：

```
<script type="text/javascript">
var str = "Visit W3School";
var patt1 = new RegExp("W3School");

var result = patt1.test(str);

document.write("Result: " + result);
</script>
```

输出：

```
Result: true
```

JavaScript 全局对象参考手册

全局属性和函数可用于所有内建的 JavaScript 对象。

顶层函数（全局函数）

FF: Firefox, IE: Internet Explorer

函数	描述	FF	IE
decodeURI()	解码某个编码的 URI。	1	5.5
decodeURIComponent()	解码一个编码的 URI 组件。	1	5.5
encodeURI()	把字符串编码为 URI。	1	5.5
encodeURIComponent()	把字符串编码为 URI 组件。	1	5.5
escape()	对字符串进行编码。	1	3
eval()	计算 JavaScript 字符串，并把它作为脚本代码来执行。	1	3
getClass()	返回一个 JavaObject 的 JavaClass。		
isFinite()	检查某个值是否为有穷大的数。	1	4
isNaN()	检查某个值是否是数字。	1	3
Number()	把对象的值转换为数字。	1	
parseFloat()	解析一个字符串并返回一个浮点数。	1	3
parseInt()	解析一个字符串并返回一个整数。	1	3
String()	把对象的值转换为字符串。	1	
unescape()	对由 escape() 编码的字符串进行解码。	1	3

顶层属性（全局属性）

FF: Firefox, IE: Internet Explorer

方法	描述	FF	IE
Infinity	代表正的无穷大的数值。	1	4
java	代表 java.* 包层级的一个 JavaPackage。		
NaN	指示某个值是不是数字值。	1	4
Packages	根 JavaPackage 对象。		
undefined	指示未定义的值。	1	5.5

全局对象描述

全局对象是预定义的对象，作为 JavaScript 的全局函数和全局属性的占位符。通过使用全局对象，可以访问所有其他所有预定义的对象、函数和属性。全局对象不是任何对象的属性，所以它没有名称。
在顶层 JavaScript 代码中，可以用关键字 this 引用全局对象。但通常不必用这种方式引用全局对象，因为全局对象是作用域链的头，这意味着所有非限定性的变量和函数名都会作为该对象的属性来查询。例如，当 JavaScript 代码引用 parseInt() 函数时，它引用的是全局对象的 parseInt 属性。全局对象是作用域链的头，还意味着在顶层 JavaScript 代码中声明的所有变量都将成为全局对象的属性。
全局对象只是一个对象，而不是类。既没有构造函数，也无法实例化一个新的全局对象。
在 JavaScript 代码嵌入一个特殊环境中时，全局对象通常具有环境特定的属性。实际上，ECMAScript 标准没有规定全局对象的类型，JavaScript 的实现或嵌入的 JavaScript 都可以把任意类型的对象作为全局对象，只要该对象定义了这里列出的基本属性和函数。例如，在允许通过 LiveConnect 或相关的技术来脚本化 Java 的 JavaScript 实现中，全局对象被赋予了这里列出的 java 和 Packages 属性以及 getClass() 方法。而在客户端 JavaScript 中，全局对象就是 Window 对象，表示允许 JavaScript 代码的 Web 浏览器窗口。

例子

在 JavaScript 核心语言中，全局对象的预定义属性都是不可枚举的，所有可以用 for/in 循环列出所有隐式或显式声明的全局变量，如下所示：

```
var variables = "";

for (var name in this)
{
variables += name + "<br />";
}

document.write(variables);
```

JavaScript decodeURI() 函数

定义和用法

decodeURI() 函数可对 encodeURI() 函数编码过的 URI 进行解码。

语法

```
decodeURI(URIstring)
```

参数	描述
URIstring	必需。一个字符串，含有要解码的 URI 或其他要解码的文本。

返回值

URIstring 的副本，其中的十六进制转义序列将被它们表示的字符替换。

实例

在本例中，我们将使用 decodeURI() 对一个编码后的 URI 进行解码：

```
<script type="text/javascript">

var test1="http://www.w3school.com.cn/My first/"

document.write(encodeURI(test1)+ "<br />")
document.write(decodeURI(test1))

</script>
```

输出：

```
http://www.w3school.com.cn/My%20first/
http://www.w3school.com.cn/My first/
```

JavaScript decodeURIComponent() 函数

定义和用法

decodeURIComponent() 函数可对 encodeURIComponent() 函数编码的 URI 进行解码。

语法

```
decodeURIComponent(URIstring)
```

参数	描述
URIstring	必需。一个字符串，含有编码 URI 组件或其他要解码的文本。

返回值

URIstring 的副本，其中的十六进制转义序列将被它们表示的字符替换。

实例

在本例中，我们将使用 decodeURIComponent() 对编码后的 URI 进行解码：

```
<script type="text/javascript">

var test1="http://www.w3school.com.cn/My first/"

document.write(encodeURIComponent(test1)+ "<br />")
document.write(decodeURIComponent(test1))

</script>
```

输出：

```
http%3A%2F%2Fwww.w3school.com.cn%2FMy%20first%2F
http://www.w3school.com.cn/My first/
```

JavaScript encodeURI() 函数

定义和用法

encodeURI() 函数可把字符串作为 URI 进行编码。

语法

```
encodeURI(URIstring)
```

参数	描述
URIstring	必需。一个字符串，含有 URI 或其他要编码的文本。

返回值

URIstring 的副本，其中的某些字符将被十六进制的转义序列进行替换。

说明

该方法不会对 ASCII 字母和数字进行编码，也不会对这些 ASCII 标点符号进行编码： - _ . ! ~ * ' ( ) 。
该方法的目的是对 URI 进行完整的编码，因此对以下在 URI 中具有特殊含义的 ASCII 标点符号，encodeURI() 函数是不会进行转义的：;/?:@&=+$,#

提示和注释

提示：如果 URI 组件中含有分隔符，比如 ? 和 #，则应当使用 encodeURIComponent() 方法分别对各组件进行编码。

实例

在本例中，我们将使用 encodeURI() 对 URI 进行编码：

```
<script type="text/javascript">

document.write(encodeURI("http://www.w3school.com.cn")+ "<br />")
document.write(encodeURI("http://www.w3school.com.cn/My first/"))
document.write(encodeURI(",/?:@&=+$#"))

</script>
```

输出：

```
http://www.w3school.com.cn
http://www.w3school.com.cn/My%20first/
```

```
,/?:@&=+$#
```

JavaScript encodeURIComponent() 函数

定义和用法

encodeURIComponent() 函数可把字符串作为 URI 组件进行编码。

语法

```
encodeURIComponent(URIstring)
```

参数	描述
URIstring	必需。一个字符串，含有 URI 组件或其他要编码的文本。

返回值

URIstring 的副本，其中的某些字符将被十六进制的转义序列进行替换。

说明

该方法不会对 ASCII 字母和数字进行编码，也不会对这些 ASCII 标点符号进行编码： - _ . ! ~ * ' ( ) 。
其他字符（比如 ：;/?:@&=+$,# 这些用于分隔 URI 组件的标点符号），都是由一个或多个十六进制的转义序列替换的。

提示和注释

提示：请注意 encodeURIComponent() 函数 与 encodeURI() 函数的区别之处，前者假定它的参数是 URI 的一部分（比如协议、主机名、路径或查询字符串）。因此 encodeURIComponent() 函数将转义用于分隔 URI 各个部分的标点符号。

实例

在本例中，我们将使用 encodeURIComponent() 对 URI 进行编码：

```
<script type="text/javascript">

document.write(encodeURIComponent("http://www.w3school.com.cn"))
document.write("<br />")
document.write(encodeURIComponent("http://www.w3school.com.cn/p
1/"))
document.write("<br />")
document.write(encodeURIComponent(",/?:@&=+$#"))

</script>
```

输出：

```
http%3A%2F%2Fwww.w3school.com.cn
http%3A%2F%2Fwww.w3school.com.cn%2Fp%201%2F
%2C%2F%3F%3A%40%26%3D%2B%24%23
```

JavaScript escape() 函数

定义和用法

escape() 函数可对字符串进行编码，这样就可以在所有的计算机上读取该字符串。

语法

```
escape(string)
```

参数	描述
string	必需。要被转义或编码的字符串。

返回值

已编码的 *string* 的副本。其中某些字符被替换成了十六进制的转义序列。

说明

该方法不会对 ASCII 字母和数字进行编码，也不会对下面这些 ASCII 标点符号进行编码： * @ - _ + . / 。其他所有的字符都会被转义序列替换。

提示和注释

提示：可以使用 unescape() 对 escape() 编码的字符串进行解码。
注释：ECMAScript v3 反对使用该方法，应用使用 decodeURI() 和 decodeURIComponent() 替代它。

实例

在本例中，我们将使用 escape() 来编码字符串：

```
<script type="text/javascript">

document.write(escape("Visit W3School!") + "<br />")
document.write(escape("?!=()#%&"))

</script>
```

输出：

```
Visit%20W3School%21
%3F%21%3D%28%29%23%25%26
```

JavaScript eval() 函数

定义和用法

eval() 函数可计算某个字符串，并执行其中的的 JavaScript 代码。

语法

```
eval(string)
```

参数	描述
string	必需。要计算的字符串，其中含有要计算的 JavaScript 表达式或要执行的语句。

返回值

通过计算 string 得到的值（如果有的话）。

说明

该方法只接受原始字符串作为参数，如果 string 参数不是原始字符串，那么该方法将不作任何改变地返回。因此请不要为 eval() 函数传递 String 对象来作为参数。
如果试图覆盖 eval 属性或把 eval() 方法赋予另一个属性，并通过该属性调用它，则 ECMAScript 实现允许抛出一个 EvalError 异常。

抛出

如果参数中没有合法的表达式和语句，则抛出 SyntaxError 异常。
如果非法调用 eval()，则抛出 EvalError 异常。
如果传递给 eval() 的 Javascript 代码生成了一个异常，eval() 将把该异常传递给调用者。

提示和注释

提示：虽然 eval() 的功能非常强大，但在实际使用中用到它的情况并不多。

实例

例子 1

在本例中，我们将在几个字符串上运用 eval()，并看看返回的结果：

```
<script type="text/javascript">

eval("x=10;y=20;document.write(x*y)")

document.write(eval("2+2"))

var x=10
document.write(eval(x+17))

</script>
```

输出：

```
200
4
27
```

例子 2

看一下在其他情况中，eval() 返回的结果：

```
eval("2+3")  // 返回 5
var myeval = eval;  // 可能会抛出 EvalError 异常
myeval("2+3");  // 可能会抛出 EvalError 异常
```

可以使用下面这段代码来检测 eval() 的参数是否合法：

```
try {
    alert("Result:" + eval(prompt("Enter an expression:","")));
    }

catch(exception) {
    alert(exception);
    }
```

JavaScript getClass() 函数

定义和用法

getClass() 函数可返回一个 JavaObject 的 JavaClass。

语法

```
getClass(javaobj)
```

参数	描述
javaobj	一个 JavaObject 对象。

返回值

javaobj 的 JavaClass 对象。

说明

该函数可接受一个 JavaObject 对象作为其参数，并返回该对象的 JavaClass，即返回 JavaClass 对象。该 JavaClass 对象表示 Java 对象的 Java 类，而这个 Java 对象所表示的 Java 类是由 JavaObject 指定的。

习惯用法

请不要把 JavaScript 的 getClass() 函数与所有 Java 对象的 getClass 方法混淆在一起。也不要把 JavaScript 的 JavaClass 对象与 Java java.lang.Class 类混淆了。
请看下面这行代码，它可创建一个 Java Rectangle 对象：

```
var obj = new java.awt.Rectangle();
```

obj 是一个保存了 JavaObject 的 JavaScript 变量。我们可以调用 JavaScript 函数 getClass() 返回一个 JavaClass 对象，该 JavaClass 对象表示 java.awt.Rectangle 类：

```
var cls = getClass(obj);
```

而调用 Java getClass() 的方式有所不同，且执行完全不同的功能：

```
cls = obj.getClass();
```

在执行了上面这行代码后，cls 是表示 java.lang.class 对象的一个 Java Object。这个 java.lang.class 对象是一个 Java 对象，它是 java.awt.Rectangle 类的一个 Java 表示。
最后，对于任何的 JavaObject obj，您 do 会看到下面的表示式始终为 true：

```
(getClass(obj.getClass()) == java.lang.Class)
```

JavaScript isFinite() 函数

定义和用法

isFinite() 函数用于检查其参数是否是无穷大。

语法

```
isFinite(number)
```

参数	描述
number	必需。要检测的数字。

返回值

如果 number 是有限数字（或可转换为有限数字），那么返回 true。否则，如果 number 是 NaN（非数字），或者是正、负无穷大的数，则返回 false。

实例

在本例中，我们将使用 isFinite() 在检测无穷数：

```
<script type="text/javascript">

document.write(isFinite(123)+ "<br />")
document.write(isFinite(-1.23)+ "<br />")
document.write(isFinite(5-2)+ "<br />")
document.write(isFinite(0)+ "<br />")
document.write(isFinite("Hello")+ "<br />")
document.write(isFinite("2005/12/12")+ "<br />")

</script>
```

输出：

```
true
true
true
true
false
false
```

JavaScript isNaN() 函数

定义和用法

isNaN() 函数用于检查其参数是否是非数字值。

语法

```
isNaN(x)
```

参数	描述
x	必需。要检测的值。

返回值

如果 x 是特殊的非数字值 NaN（或者能被转换为这样的值），返回的值就是 true。如果 x 是其他值，则返回 false。

说明

isNaN() 函数可用于判断其参数是否是 NaN，该值表示一个非法的数字（比如被 0 除后得到的结果）。
如果把 NaN 与任何值（包括其自身）相比得到的结果均是 false，所以要判断某个值是否是 NaN，不能使用 == 或 === 运算符。正因为如此，isNaN() 函数是必需的。

提示和注释

提示：isNaN() 函数通常用于检测 parseFloat() 和 parseInt() 的结果，以判断它们表示的是否是合法的数字。当然也可以用 isNaN() 函数来检测算数错误，比如用 0 作除数的情况。

实例

检查数字是否非法：

```
<script>

document.write(isNaN(123));
```

```
document.write(isNaN(-1.23));
document.write(isNaN(5-2));
document.write(isNaN(0));
document.write(isNaN("Hello"));
document.write(isNaN("2005/12/12"));

</script>
```

输出：

```
false
false
false
false
true
true
```

JavaScript Number() 函数

定义和用法

Number() 函数把对象的值转换为数字。

语法

```
Number(object)
```

参数	描述
object	必需。JavaScript 对象。

返回值

如果参数是 Date 对象，Number() 返回从 1970 年 1 月 1 日至今的毫秒数。
如果对象的值无法转换为数字，那么 Number() 函数返回 NaN。

实例

在本例中，我们将尝试把不同的对象转换为数字：

```
<script type="text/javascript">

var test1= new Boolean(true);
var test2= new Boolean(false);
var test3= new Date();
var test4= new String("999");
var test5= new String("999 888");

document.write(Number(test1)+ "<br />");
document.write(Number(test2)+ "<br />");
document.write(Number(test3)+ "<br />");
document.write(Number(test4)+ "<br />");
document.write(Number(test5)+ "<br />");

</script>
```

输出：

```
1
0
1256657776588
999
NaN
```

JavaScript parseFloat() 函数

定义和用法

parseFloat() 函数可解析一个字符串，并返回一个浮点数。
该函数指定字符串中的首个字符是否是数字。如果是，则对字符串进行解析，直到到达数字的末端为止，然后以数字返回该数字，而不是作为字符串。

语法

```
parseFloat(string)
```

参数	描述
string	必需。要被解析的字符串。

详细说明

parseFloat 是全局函数，不属于任何对象。
parseFloat 将它的字符串参数解析成为浮点数并返回。如果在解析过程中遇到了正负号（+ 或 -）、数字 (0-9)、小数点，或者科学记数法中的指数（e 或 E）以外的字符，则它会忽略该字符以及之后的所有字符，返回当前已经解析到的浮点数。同时参数字符串首位的空白符会被忽略。
如果参数字符串的第一个字符不能被解析成为数字，则 parseFloat 返回 NaN。
提示：您可以通过调用 isNaN 函数来判断 parseFloat 的返回结果是否是 NaN。如果让 NaN 作为了任意数学运算的操作数，则运算结果必定也是 NaN。

返回值

返回解析后的数字。

提示和注释

注释：开头和结尾的空格是允许的。
提示：如果字符串的第一个字符不能被转换为数字，那么 parseFloat() 会返回 NaN。
提示：如果只想解析数字的整数部分，请使用 parseInt() 方法。

实例

例子 1

在本例中，我们将使用 parseFloat() 来解析不同的字符串：

```
<script type="text/javascript">

document.write(parseFloat("10"))
document.write(parseFloat("10.00"))
document.write(parseFloat("10.33"))
document.write(parseFloat("34 45 66"))
document.write(parseFloat(" 60 "))
document.write(parseFloat("40 years"))
document.write(parseFloat("He was 40"))
```

```
</script>
```

输出：

```
10
10
10.33
34
60
40
NaN
```

例子 2

下面的例子都返回 3.14：

```
<script type="text/javascript">

document.write(parseFloat("3.14"))
document.write(parseFloat("314e-2"))
document.write(parseFloat("0.0314E+2"))
document.write(parseFloat("3.14more non-digit characters"))

</script>
```

输出：

```
3.14
```

例子 3

下面的例子将返回 NaN：

```
<script type="text/javascript">

document.write(parseFloat("FF2"))

</script>
```

输出：

```
NaN
```

JavaScript parseInt() 函数

定义和用法

parseInt() 函数可解析一个字符串，并返回一个整数。

语法

```
parseInt(string, radix)
```

参数	描述
string	必需。要被解析的字符串。
radix	可选。表示要解析的数字的基数。该值介于 2 ~ 36 之间。 如果省略该参数或其值为 0，则数字将以 10 为基础来解析。如果它以 “0x” 或 “0X” 开头，将以 16 为基数。 如果该参数小于 2 或者大于 36，则 parseInt() 将返回 NaN。

返回值

返回解析后的数字。

说明

当参数 *radix* 的值为 0，或没有设置该参数时，parseInt() 会根据 *string* 来判断数字的基数。举例，如果 *string* 以 "0x" 开头，parseInt() 会把 *string* 的其余部分解析为十六进制的整数。如果 *string* 以 0 开头，那么 ECMAScript v3 允许 parseInt() 的一个实现把其后的字符解析为八进制或十六进制的数字。如果 *string* 以 1 ~ 9 的数字开头，parseInt() 将把它解析为十进制的整数。

提示和注释

注释：只有字符串中的第一个数字会被返回。
注释：开头和结尾的空格是允许的。
提示：如果字符串的第一个字符不能被转换为数字，那么 parseFloat() 会返回 NaN。

实例

在本例中，我们将使用 parseInt() 来解析不同的字符串：

```
parseInt("10");          //返回 10
parseInt("19",10);       //返回 19 (10+9)
parseInt("11",2);        //返回 3 (2+1)
parseInt("17",8);        //返回 15 (8+7)
parseInt("1f",16);       //返回 31 (16+15)
parseInt("010");     //未定：返回 10 或 8
```

JavaScript String() 函数

定义和用法

String() 函数把对象的值转换为字符串。

语法

```
String(object)
```

参数	描述
object	必需。JavaScript 对象。

实例

在本例中，我们将尝试把不同的对象转换为字符串：

```
<script type="text/javascript">

var test1= new Boolean(1);
var test2= new Boolean(0);
var test3= new Boolean(true);
var test4= new Boolean(false);
var test5= new Date();
var test6= new String("999 888");
var test7=12345;
```

```
document.write(String(test1)+ "<br />");
document.write(String(test2)+ "<br />");
document.write(String(test3)+ "<br />");
document.write(String(test4)+ "<br />");
document.write(String(test5)+ "<br />");
document.write(String(test6)+ "<br />");
document.write(String(test7)+ "<br />");

</script>
```

输出：

```
true
false
true
false
Wed Oct 28 00:17:40 UTC+0800 2009
999 888
12345
```

JavaScript unescape() 函数

定义和用法

unescape() 函数可对通过 escape() 编码的字符串进行解码。

语法

```
unescape(string)
```

参数	描述
string	必需。要解码或反转义的字符串。

返回值

string 被解码后的一个副本。

说明

该函数的工作原理是这样的：通过找到形式为 %xx 和 %uxxxx 的字符序列（x 表示十六进制的数字），用 Unicode 字符 \u00xx 和 \uxxxx 替换这样的字符序列进行解码。

提示和注释

注释：ECMAScript v3 已从标准中删除了 unescape() 函数，并反对使用它，因此应该用 decodeURI() 和 decodeURIComponent() 取而代之。

实例

在本例中，我们将使用 escape() 来编码字符串，然后使用 unescape() 对其解码：

```
<script type="text/javascript">

var test1="Visit W3School!"

test1=escape(test1)
document.write (test1 + "<br />")

test1=unescape(test1)
document.write(test1 + "<br />")

</script>
```

输出：

```
Visit%20W3School%21
Visit W3School!
```

JavaScript Infinity 属性

定义和用法

Infinity 属性用于存放表示正无穷大的数值。

语法

```
Infinity
```

说明

无法使用 for/in 循环来枚举 Infinity 属性，也不能用 delete 运算符来删除它。
Infinity 不是常量，可以把它设置为其他值。

实例

在本例中，我们将展示当一个数超出了 infinity 的限制发生的情况：

```
<script type="text/javascript">

var t1=1.7976931348623157E+10308
document.write(t1)

document.write("<br />")

var t2=-1.7976931348623157E+10308
document.write(t2)

</script>
```

输出：

```
Infinity
-Infinity
```

JavaScript java 属性

定义和用法

表示 java.* 包层级的 JavaPackage。

语法

```
java
```

说明

在包含了 LiveConnect 或其他用于脚本化 Java 的技术的 JavaScript 实现中，全局 java 属性就是一个 JavaPackage 对象，它表示 java.* 包层级。这个属性的存在意味着像 java.util 这样的一个 JavaScript 表示式引用的是 java.util 包。对于不符合 java.* 层级的 Java 包，请参阅全局 Packages 属性。

JavaScript NaN 属性

定义和用法

NaN 属性用于引用特殊的非数字值。

语法

```
NaN
```

说明

无法使用 for/in 循环来枚举 NaN 属性，也不能用 delete 运算符来删除它。
NaN 不是常量，可以把它设置为其他值。

提示和注释

提示：请使用 isNaN() 来判断一个值是否是数字。原因是 NaN 与所有值都不相等，包括它自己。

实例

在本例中，我们将展示当一个数超出了 infinity 的限制发生的情况：

```
<script type="text/javascript">

var test1="300"
```

```
var test2="Hello World!"

document.write(Number(test1)+ "<br />")
document.write(Number(test2)+ "<br />")

document.write(isNaN(test1)+ "<br />")
document.write(isNaN(test2))

</script>
```

输出：

```
300
NaN
false
true
```

JavaScript Packages 属性

定义和用法

根 JavaPackage。

语法

```
Packages
```

说明

在包含了 LiveConnect 或其他用于脚本化 Java 的技术的 JavaScript 实现中，全局 Packages 属性就是一个 JavaPackage 对象，其属性是 Java 解释器所知道的所有包的根。例如，Packages.javax.swing 引用的是 Java 包 javax.swing。全局属性 java 是 Packages.java 的简写。

JavaScript undefined 属性

定义和用法

undefined 属性用于存放 JavaScript 的 undefined 值。

语法

```
undefined
```

说明

无法使用 for/in 循环来枚举 undefined 属性，也不能用 delete 运算符来删除它。
undefined 不是常量，可以把它设置为其他值。
当尝试读取不存在的对象属性时也会返回 undefined。

提示和注释

提示：只能用 === 运算来测试某个值是否是未定义的，因为 == 运算符认为 undefined 值等价于 null。
注释：null 表示无值，而 undefined 表示一个未声明的变量，或已声明但没有赋值的变量，或一个并不存在的对象属性。

实例

在本例中，我们将检测两个变量中未定义的一个：

```
<script type="text/javascript">

var t1=""
var t2

if (t1===undefined) {document.write("t1 is undefined")}
if (t2===undefined) {document.write("t2 is undefined")}

</script>
```

输出：

```
t2 is undefined
```

Window 对象

Window 对象

Window 对象表示浏览器中打开的窗口。
如果文档包含框架（frame 或 iframe 标签），浏览器会为 HTML 文档创建一个 window 对象，并为每个框架创建一个额外的 window 对象。
注释：没有应用于 window 对象的公开标准，不过所有浏览器都支持该对象。
IE: Internet Explorer, **F:** Firefox, **O:** Opera.

Window 对象集合

集合	描述	IE	F	O
frames[]	返回窗口中所有命名的框架。 该集合是 Window 对象的数组，每个 Window 对象在窗口中含有一个框架或 <iframe>。属性 frames.length 存放数组 frames[] 中含有的元素个数。注意，frames[] 数组中引用的框架可能还包括框架，它们自己也具有 frames[] 数组。	4	1	9

Window 对象属性

属性	描述	IE	F	O
closed	返回窗口是否已被关闭。	4	1	9
defaultStatus	设置或返回窗口状态栏中的默认文本。	4	No	9
document	对 Document 对象的只读引用。请参阅 Document 对象。	4	1	9
history	对 History 对象的只读引用。请参数 History 对象。	4	1	9
innerheight	返回窗口的文档显示区的高度。	No	No	No
innerwidth	返回窗口的文档显示区的宽度。	No	No	No
length	设置或返回窗口中的框架数量。	4	1	9

location	用于窗口或框架的 Location 对象。请参阅 Location 对象。	4	1	9
name	设置或返回窗口的名称。	4	1	9
Navigator	对 Navigator 对象的只读引用。请参数 Navigator 对象。	4	1	9
opener	返回对创建此窗口的窗口的引用。	4	1	9
outerheight	返回窗口的外部高度。	No	No	No
outerwidth	返回窗口的外部宽度。	No	No	No
pageXOffset	设置或返回当前页面相对于窗口显示区左上角的 X 位置。	No	No	No
pageYOffset	设置或返回当前页面相对于窗口显示区左上角的 Y 位置。	No	No	No
parent	返回父窗口。	4	1	9
Screen	对 Screen 对象的只读引用。请参数 Screen 对象。	4	1	9
self	返回对当前窗口的引用。等价于 Window 属性。	4	1	9
status	设置窗口状态栏的文本。	4	No	9
top	返回最顶层的先辈窗口。	4	1	9
window	window 属性等价于 self 属性，它包含了对窗口自身的引用。	4	1	9
• screenLeft • screenTop • screenX • screenY	只读整数。声明了窗口的左上角在屏幕上的的 x 坐标和 y 坐标。IE、Safari 和 Opera 支持 screenLeft 和 screenTop，而 Firefox 和 Safari 支持 screenX 和 screenY。	4	1	9

Window 对象方法

方法	描述	IE	F	O
alert()	显示带有一段消息和一个确认按钮的警告框。	4	1	9
blur()	把键盘焦点从顶层窗口移开。	4	1	9
clearInterval()	取消由 setInterval() 设置的 timeout。	4	1	9
clearTimeout()	取消由 setTimeout() 方法设置的 timeout。	4	1	9

close()	关闭浏览器窗口。	4	1	9
confirm()	显示带有一段消息以及确认按钮和取消按钮的对话框。	4	1	9
createPopup()	创建一个 pop-up 窗口。	4	No	No
focus()	把键盘焦点给予一个窗口。	4	1	9
moveBy()	可相对窗口的当前坐标把它移动指定的像素。	4	1	9
moveTo()	把窗口的左上角移动到一个指定的坐标。	4	1	9
open()	打开一个新的浏览器窗口或查找一个已命名的窗口。	4	1	9
print()	打印当前窗口的内容。	5	1	9
prompt()	显示可提示用户输入的对话框。	4	1	9
resizeBy()	按照指定的像素调整窗口的大小。	4	1	9
resizeTo()	把窗口的大小调整到指定的宽度和高度。	4	1.5	9
scrollBy()	按照指定的像素值来滚动内容。	4	1	9
scrollTo()	把内容滚动到指定的坐标。	4	1	9
setInterval()	按照指定的周期（以毫秒计）来调用函数或计算表达式。	4	1	9
setTimeout()	在指定的毫秒数后调用函数或计算表达式。	4	1	9

Window 对象描述

Window 对象表示一个浏览器窗口或一个框架。在客户端 JavaScript 中，Window 对象是全局对象，所有的表达式都在当前的环境中计算。也就是说，要引用当前窗口根本不需要特殊的语法，可以把那个窗口的属性作为全局变量来使用。例如，可以只写 document，而不必写 window.document。

同样，可以把当前窗口对象的方法当作函数来使用，如只写 alert()，而不必写 Window.alert()。

除了上面列出的属性和方法，Window 对象还实现了核心 JavaScript 所定义的所有全局属性和方法。

Window 对象的 window 属性和 self 属性引用的都是它自己。当你想明确地引用当前窗口，而不仅仅是隐式地引用它时，可以使用这两个属性。除了这两个属性之外，parent 属性、top 属性以及 frame[] 数组都引用了与当前 Window 对象相关的其他 Window 对象。

要引用窗口中的一个框架，可以使用如下语法：

```
frame[i]      //当前窗口的框架
self.frame[i] //当前窗口的框架
w.frame[i]    //窗口 w 的框架
```

要引用一个框架的父窗口（或父框架），可以使用下面的语法：

```
parent    //当前窗口的父窗口
self.parent   //当前窗口的父窗口
w.parent      //窗口 w 的父窗口
```

要从顶层窗口含有的任何一个框架中引用它，可以使用如下语法：

```
top    //当前框架的顶层窗口
self.top      //当前框架的顶层窗口
f.top      //框架 f 的顶层窗口
```

新的顶层浏览器窗口由方法 Window.open() 创建。当调用该方法时，应把 open() 调用的返回值存储在一个变量中，然后使用那个变量来引用新窗口。新窗口的 opener 属性反过来引用了打开它的那个窗口。
一般来说，Window 对象的方法都是对浏览器窗口或框架进行某种操作。而 alert() 方法、confirm() 方法和 prompt 方法则不同，它们通过简单的对话框与用户进行交互。

HTML DOM closed 属性

定义和用法

closed 属性可返回一个布尔值，该值声明了窗口是否已经关闭。该属性为只读。
当浏览器窗口关闭时，表示该窗口的 Windows 对象并不会消失，它将继续存在，不过它的 closed 属性将设置为 true。

语法

```
window.closed
```

实例

下面的例子可检测新窗口是否已被关闭：

```
<html>
<head>
<script type="text/javascript">
function ifClosed()
  {
```

```
  document.write("'myWindow' has been closed!")
  }

function ifNotClosed()
  {
  document.write("'myWindow' has not been closed!")
  }

function checkWin()
  {
  if (myWindow.closed)
    ifClosed()
  else
    ifNotClosed()
  }
</script>
</head>
<body>

<script type="text/javascript">
myWindow=window.open('','','width=200,height=100')
myWindow.document.write("This is 'myWindow'")
</script>

<input type="button" value="Has 'myWindow' been closed?"
onclick="checkWin()">

</body>
</html>
```

HTML DOM defaultStatus 属性

定义和用法

defaultStatus 属性可设置或返回窗口状态栏中的默认文本。该属性可读可写。
该文本会在页面加载时被显示。

语法

```
window.defaultStatus=sometext
```

实例

下面的例子讲在状态栏设置文本：

```
<html>
<body>

<script type="text/javascript">
window.defaultStatus="This is the default text in the status bar!!";
</script>

<p>Look at the text in the statusbar.</p>

</body>
</html>
```

HTML DOM innerheight、innerwidth 属性

定义和用法

只读属性，声明了窗口的文档显示区的高度和宽度，以像素计。
这里的宽度和高度不包括菜单栏、工具栏以及滚动条等的高度。
IE 不支持这些属性。它用 document.documentElement 或 document.body （与 IE 的版本相关）的 clientWidth 和 clientHeight 属性作为替代。

HTML DOM name 属性

定义和用法
name 属性可设置或返回存放窗口的名称的一个字符串。
该名称是在 open() 方法创建窗口时指定的或者使用一个 <frame> 标记的 name 属性指定的。
窗口的名称可以用作一个 <a> 或者 <form> 标记的 target 属性的值。以这种方式使用 target 属性声明了超链接文档或表单提交结果应该显示于指定的窗口或框架中。

语法

```
window.name=name
```

实例

下面的例子可返回新窗口的名称：

```
<html>
<head>
<script type="text/javascript">
function checkWin()
  {
  document.write(myWindow.name)
  }
</script>
</head>
<body>

<script type="text/javascript">
myWindow=window.open('','MyName','width=200,height=100')
myWindow.document.write("This is 'myWindow'")
</script>

<input type="button" value="What's the name of 'myWindow'?"
onclick="checkWin()">

</body>
</html>
```

HTML DOM opener 属性

定义和用法

opener 属性是一个可读可写的属性，可返回对创建该窗口的 Window 对象的引用。
opener 属性非常有用，创建的窗口可以引用创建它的窗口所定义的属性和函数。

语法

```
window.opener
```

提示和注释

注释：只有表示顶层窗口的 Window 对象的 operner 属性才有效，表示框架的 Window 对象的 operner 属性无效。

实例

下面的例子可向 opener 窗口写文本（父窗口）：

```
<html>
<body>

<script type="text/javascript">
myWindow=window.open('','MyName','width=200,height=100')
myWindow.document.write("This is 'myWindow'")
myWindow.focus()
myWindow.opener.document.write("This is the parent window")
</script>

</body>
</html>
```

HTML DOM outerheight 属性

定义和用法

outerheight 属性是一个只读的整数，声明了整个窗口的高度。

语法

```
window.outerheight=pixels
```

提示和注释

提示：IE 不支持此属性，且没有提供替代的属性。

实例

下面的例子可把新窗口设置为 100x100 像素：

```
<html>
<body>

<script type="text/javascript">
myWindow=window.open('','')
myWindow.outerheight="100"
myWindow.outerwidth="100"
myWindow.document.write("This is 'myWindow'")
myWindow.focus()
</script>

</body>
</html>
```

HTML DOM outerwidth 属性

定义和用法

outerwidth 属性是一个只读的整数，声明了整个窗口的宽度。

语法

```
window.outerwidth=pixels
```

提示和注释

提示：IE 不支持此属性，且没有提供替代的属性。

实例

下面的例子可把新窗口设置为 100x100 像素：

```
<html>
<body>

<script type="text/javascript">
myWindow=window.open('','')
myWindow.outerheight="100"
myWindow.outerwidth="100"
myWindow.document.write("This is 'myWindow'")
myWindow.focus()
</script>

</body>
</html>
```

HTML DOM self 属性

定义和用法

self 属性可返回对窗口自身的只读引用。等价于 Window 属性。

语法

```
window.self
```

实例

下面的例子窗口是否在一个框架中，如果是，则跳出框架：

```
<html>
<head>
<script type="text/javascript">
function breakout()
  {
  if (window.top!=window.self)
    {
```

```
    window.top.location="tryjs_breakout.htm"
    }
  }
</script>
</head>
<body>

<input type="button" onclick="breakout()"
value="Break out of frame!">

</body>
</html>
```

HTML DOM status 属性

定义和用法

status 属性可设置或返回窗口状态栏中的文本。

语法

```
window.status=sometext
```

说明

status 属性是一个可读可写的字符串，声明了要在窗口状态栏中显示的一条消息。通常显示这条消息的时间是有限的，直到其他的消息将它覆盖，或者用户把鼠标移动到窗口的其他区域为止。当擦除了 status 声明的消息时，状态栏要么恢复为它默认的空白状态，要么是再次显示出属性 defaultStatus 声明的默认消息。
据我们所知，不少浏览器已经关闭了脚本化它们的状态栏的功能。这是一项安全措施，防止隐藏了超链接真正目的的钓鱼攻击。

实例

下面的例子讲在状态栏设置文本：

```
<html>
<body>
```

```
<script type="text/javascript">
window.status="Some text in the status bar!!"
</script>

</body>
</html>
```

HTML DOM top 属性

定义和用法

top 属性返回最顶层的先辈窗口。
该属性返回对一个顶级窗口的只读引用。如果窗口本身就是一个顶级窗口，top 属性存放对窗口自身的引用。如果窗口是一个框架，那么 top 属性引用包含框架的顶层窗口。

语法

```
window.top
```

实例

下面的例子窗口是否在一个框架中，如果是，则跳出框架：

```
<html>
<head>
<script type="text/javascript">
function breakout()
{
if (window.top!=window.self)
{
window.top.location="tryjs_breakout.htm"
}
}
</script>
</head>

<body>
<form>
Click the button to break out of the frame:
```

```
<input type="button" onclick="breakout()" value="Break out!">
</form>
</body>

</html>
```

HTML DOM alert() 方法

定义和用法

alert() 方法用于显示带有一条指定消息和一个 OK 按钮的警告框。

语法

```
alert(message)
```

参数	描述
message	要在 window 上弹出的对话框中显示的纯文本（而非 HTML 文本）

实例

```
<html>
<head>
<script type="text/javascript">
function display_alert()
  {
  alert("I am an alert box!!")
  }
</script>
</head>
<body>

<input type="button" onclick="display_alert()"
value="Display alert box" />

</body>
</html>
```

HTML DOM blur() 方法

定义和用法

blur() 方法可把键盘焦点从顶层窗口移开。

语法

```
window.blur()
```

说明

方法 blur() 可把键盘焦点从顶层浏览器窗口移走，整个窗口由 Window 对象指定。哪个窗口最终获得键盘焦点并没有指定。

提示和注释

注释：在某些浏览器上，该方法可能无效。

实例

```
<html>
<body>

<script type="text/javascript">
myWindow=window.open('','','width=200,height=100')
myWindow.document.write("This is 'myWindow'")
myWindow.blur()
</script>

</body>
</html>
```

HTML DOM clearInterval() 方法

定义和用法

clearInterval() 方法可取消由 setInterval() 设置的 timeout。
clearInterval() 方法的参数必须是由 setInterval() 返回的 ID 值。

语法

```
clearInterval(id_of_setinterval)
```

参数	描述
id_of_setinterval	由 setInterval() 返回的 ID 值。

实例

下面这个例子将每隔 50 毫秒调用 clock() 函数。您也可以使用一个按钮来停止这个 clock：

```
<html>
<body>

<input type="text" id="clock" size="35" />
<script language=javascript>
var int=self.setInterval("clock()",50)
function clock()
  {
  var t=new Date()
  document.getElementById("clock").value=t
  }
</script>
</form>
<button onclick="int=window.clearInterval(int)">
Stop interval</button>

</body>
</html>
```

HTML DOM clearTimeout() 方法

定义和用法

clearTimeout() 方法可取消由 setTimeout() 方法设置的 timeout。

语法

```
clearTimeout(id_of_settimeout)
```

参数	描述
id_of_settimeout	由 setTimeout() 返回的 ID 值。该值标识要取消的延迟执行代码块。

实例

下面的例子每秒调用一次 timedCount() 函数。您也可以使用一个按钮来终止这个定时消息：

```
<html>
<head>
<script type="text/javascript">
var c=0
var t
function timedCount()
  {
  document.getElementById('txt').value=c
  c=c+1
  t=setTimeout("timedCount()",1000)
  }
function stopCount()
  {
  clearTimeout(t)
  }
</script>
</head>
<body>

<form>
<input type="button" value="Start count!" onClick="timedCount()">
<input type="text" id="txt">
```

```
<input type="button" value="Stop count!" onClick="stopCount()">
</form>

</body>
</html>
```

HTML DOM close() 方法

定义和用法

close() 方法用于关闭浏览器窗口。

语法

```
window.close()
```

说明

方法 close() 将关闭有 window 指定的顶层浏览器窗口。某个窗口可以通过调用 self.close() 或只调用 close() 来关闭其自身。
只有通过 JavaScript 代码打开的窗口才能够由 JavaScript 代码关闭。这阻止了恶意的脚本终止用户的浏览器。

实例

下面的例子可关闭新的浏览器：

```
<html>
<head>
<script type="text/javascript">
function closeWin()
  {
  myWindow.close()
  }
</script>
</head>
<body>

<script type="text/javascript">
```

```
myWindow=window.open('','','width=200,height=100')
myWindow.document.write("This is 'myWindow'")
</script>

<input type="button" value="Close 'myWindow'"
onclick="closeWin()" />

</body>
</html>
```

HTML DOM confirm() 方法

定义和用法

confirm() 方法用于显示一个带有指定消息和 OK 及取消按钮的对话框。

语法

```
confirm(message)
```

参数	描述
message	要在 window 上弹出的对话框中显示的纯文本（而非 HTML 文本）

说明

如果用户点击确定按钮，则 confirm() 返回 true。如果点击取消按钮，则 confirm() 返回 false。
在用户点击确定按钮或取消按钮把对话框关闭之前，它将阻止用户对浏览器的所有输入。在调用 confirm() 时，将暂停对 JavaScript 代码的执行，在用户作出响应之前，不会执行下一条语句。

提示和注释

提示：对话框按钮的文字是不可改变的，因此请小心地编写问题或消息，使它适合用确认和取消来回答。

实例

```
<html>
<head>
<script type="text/javascript">
function disp_confirm()
  {
  var r=confirm("Press a button")
  if (r==true)
    {
    document.write("You pressed OK!")
    }
  else
    {
    document.write("You pressed Cancel!")
    }
  }
</script>
</head>
<body>

<input type="button" onclick="disp_confirm()"
value="Display a confirm box" />

</body>
</html>
```

HTML DOM createPopup() 方法

定义和用法

createPopup() 方法用于创建一个 pop-up 窗口。

语法

```
window.createPopup()
```

实例

```
<html>
<head>
<script type="text/javascript">
function show_popup()
{
var p=window.createPopup()
var pbody=p.document.body
pbody.style.backgroundColor="lime"
pbody.style.border="solid black 1px"
pbody.innerHTML="This is a pop-up! Click outside to close."
p.show(150,150,200,50,document.body)
}
</script>
</head>

<body>
<button onclick="show_popup()">Create pop-up!</button>
</body>

</html>
```

HTML DOM focus() 方法

定义和用法

focus() 方法可把键盘焦点给予一个窗口。

语法

```
window.focus()
```

实例

下面的例子可确保新的窗口得到焦点：

```
<html>
<body>
```

```
<script type="text/javascript">
myWindow=window.open('','','width=200,height=100')
myWindow.document.write("This is 'myWindow'")
myWindow.focus()
</script>

</body>
</html>
```

HTML DOM moveBy() 方法

定义和用法

moveBy() 方法可相对窗口的当前坐标把它移动指定的像素。

语法

```
window.moveBy(x,y)
```

参数	描述
x	要把窗口右移的像素数
y	要把窗口下移的像素数

实例

下面的例子将把窗口相对其当前位置移动 50 个像素：

```
<html>
<head>
<script type="text/javascript">
function moveWin()
  {
  myWindow.moveBy(50,50)
  }
</script>
</head>
<body>
```

```
<script type="text/javascript">
myWindow=window.open('','','width=200,height=100')
myWindow.document.write("This is 'myWindow'")
</script>

<input type="button" value="Move 'myWindow'"
onclick="moveWin()" />

</body>
</html>
```

HTML DOM moveTo() 方法

定义和用法

moveTo() 方法可把窗口的左上角移动到一个指定的坐标。

语法

```
window.moveTo(x,y)
```

参数	描述
x	窗口新位置的 x 坐标
y	窗口新位置的 y 坐标

提示和注释

注释：出于安全方面的原因，浏览器限制此方法使其不能把窗口移出屏幕。

实例

下面的例子讲把新窗口移动到指定的坐标：

```
<html>
<head>
<script type="text/javascript">
```

```
function moveWin()
  {
  myWindow.moveTo(50,50)
  }
</script>
</head>
<body>

<script type="text/javascript">
myWindow=window.open('','','width=200,height=100')
myWindow.document.write("This is 'myWindow'")
</script>

<input type="button" value="Move 'myWindow'"
onclick="moveWin()" />

</body>
</html>
```

HTML DOM open() 方法

定义和用法

open() 方法用于打开一个新的浏览器窗口或查找一个已命名的窗口。

语法

```
window.open(URL,name,features,replace)
```

参数	描述
URL	一个可选的字符串，声明了要在新窗口中显示的文档的 URL。如果省略了这个参数，或者它的值是空字符串，那么新窗口就不会显示任何文档。
name	一个可选的字符串，该字符串是一个由逗号分隔的特征列表，其中包括数字、字母和下划线，该字符声明了新窗口的名称。这个名称可以用作标记 <a> 和 <form> 的属性 target 的值。如果该参数指定了一个已经存在的窗口，那么 open() 方法就不再创建一个新窗口，而只是返回对指定窗口的引用。在这种情况下，features 将被忽略。
features	一个可选的字符串，声明了新窗口要显示的标准浏览器的特征。如果省略该参数，

	新窗口将具有所有标准特征。在窗口特征这个表格中，我们对该字符串的格式进行了详细的说明。
replace	一个可选的布尔值。规定了装载到窗口的 URL 是在窗口的浏览历史中创建一个新条目，还是替换浏览历史中的当前条目。支持下面的值： • true - URL 替换浏览历史中的当前条目。 • false - URL 在浏览历史中创建新的条目。

提示和注释

重要事项：请不要混淆方法 Window.open() 与方法 Document.open()，这两者的功能完全不同。为了使您的代码清楚明白，请使用 Window.open()，而不要使用 open()。

实例 1

下面的例子讲在新浏览器窗口中打开 www.w3school.com.cn ：

```
<html>
<head>
<script type="text/javascript">
function open_win()
{
window.open("http://www.w3school.com.cn")
}
</script>
</head>
<body>

<input type=button value="Open Window" onclick="open_win()" />

</body>
</html>
```

实例 2

下面的例子讲在新浏览器窗口中打开 about:blank 页：

```
<html>
<body>

<script type="text/javascript">
myWindow=window.open('','','width=200,height=100')
```

```
myWindow.document.write("This is 'myWindow'")
myWindow.focus()
</script>

</body>
</html>
```

窗口特征（Window Features）

channelmode=yes|no|1|0	是否使用剧院模式显示窗口。默认为 no。
directories=yes|no|1|0	是否添加目录按钮。默认为 yes。
fullscreen=yes|no|1|0	是否使用全屏模式显示浏览器。默认是 no。处于全屏模式的窗口必须同时处于剧院模式。
height=pixels	窗口文档显示区的高度。以像素计。
left=pixels	窗口的 x 坐标。以像素计。
location=yes|no|1|0	是否显示地址字段。默认是 yes。
menubar=yes|no|1|0	是否显示菜单栏。默认是 yes。
resizable=yes|no|1|0	窗口是否可调节尺寸。默认是 yes。
scrollbars=yes|no|1|0	是否显示滚动条。默认是 yes。
status=yes|no|1|0	是否添加状态栏。默认是 yes。
titlebar=yes|no|1|0	是否显示标题栏。默认是 yes。
toolbar=yes|no|1|0	是否显示浏览器的工具栏。默认是 yes。
top=pixels	窗口的 y 坐标。
width=pixels	窗口的文档显示区的宽度。以像素计。

HTML DOM print() 方法

定义和用法

print() 方法用于打印当前窗口的内容。
调用 print() 方法所引发的行为就像用户单击浏览器的打印按钮。通常，这会产生一个对话框，让用户可以取消或定制打印请求。

语法

```
window.print()
```

实例 1

```
<html>
<head>
<script type="text/javascript">
function printpage()
  {
  window.print()
  }
</script>
</head>
<body>

<input type="button" value="Print this page"
onclick="printpage()" />

</body>
</html>
```

HTML DOM prompt() 方法

定义和用法

prompt() 方法用于显示可提示用户进行输入的对话框。

语法

```
prompt(text,defaultText)
```

参数	描述
text	可选。要在对话框中显示的纯文本（而不是 HTML 格式的文本）。
defaultText	可选。默认的输入文本。

说明

如果用户单击提示框的取消按钮，则返回 null。如果用户单击确认按钮，则返回输入字段当前显示的文本。
在用户点击确定按钮或取消按钮把对话框关闭之前，它将阻止用户对浏览器的所有输入。在调用 prompt() 时，将暂停对 JavaScript 代码的执行，在用户作出响应之前，不会执行下一条语句。

实例 1

```
<html>
<head>
<script type="text/javascript">
function disp_prompt()
  {
  var name=prompt("Please enter your name","")
  if (name!=null && name!="")
    {
    document.write("Hello " + name + "!")
    }
  }
</script>
</head>
<body>

<input type="button" onclick="disp_prompt()"
value="Display a prompt box" />

</body>
</html>
```

HTML DOM resizeBy() 方法

定义和用法

resizeBy() 方法用于根据指定的像素来调整窗口的大小。

语法

```
resizeBy(width,height)
```

参数	描述
width	必需。要使窗口宽度增加的像素数。可以是正、负数值。
height	可选。要使窗口高度增加的像素数。可以是正、负数值。

实例 1

```
<html>
<head>
<script type="text/javascript">
function resizeWindow()
  {
  window.resizeBy(-100,-100)
  }
</script>
</head>
<body>

<input type="button" onclick="resizeWindow()"
value="Resize window">

</body>
</html>
```

HTML DOM resizeTo() 方法

定义和用法

resizeTo() 方法用于把窗口大小调整为指定的宽度和高度。

语法

```
resizeTo(width,height)
```

参数	描述
width	必需。想要调整到的窗口的宽度。以像素计。
height	可选。想要调整到的窗口的高度。以像素计。

实例

```
<html>
<head>
<script type="text/javascript">
function resizeWindow()
  {
  window.resizeTo(500,300)
  }
</script>
</head>
<body>

<input type="button" onclick="resizeWindow()"
value="Resize window">

</body>
</html>
```

HTML DOM scrollBy() 方法

定义和用法

scrollBy() 方法可把内容滚动指定的像素数。

语法

```
scrollBy(xnum,ynum)
```

参数	描述
xnum	必需。把文档向右滚动的像素数。
ynum	必需。把文档向下滚动的像素数。

实例

下面的例子可把内容滚动 100 像素：

```
<html>
<head>
<script type="text/javascript">
function scrollWindow()
  {
  window.scrollBy(100,100)
  }
</script>
</head>
<body>

<input type="button" onclick="scrollWindow()" value="Scroll" />
<p>SCROLL SCROLL SCROLL SCROLL SCROLL SCROLL SCROLL SCROLL</p>
<br />
<br />
<br />
<br />
<br />
<br />
<br />
<br />
<p>SCROLL SCROLL SCROLL SCROLL SCROLL SCROLL SCROLL SCROLL</p>
<br />
<br />
<br />
<br />
<br />
<br />
<br />
<br />
<p>SCROLL SCROLL SCROLL SCROLL SCROLL SCROLL SCROLL SCROLL</p>

</body>
</html>
```

HTML DOM scrollTo() 方法

定义和用法

scrollTo() 方法可把内容滚动到指定的坐标。

语法

```
scrollTo(xpos,ypos)
```

参数	描述
xpos	必需。要在窗口文档显示区左上角显示的文档的 x 坐标。
ypos	必需。要在窗口文档显示区左上角显示的文档的 y 坐标。

实例

下面的例子可把内容滚动到位置 100,500：

```
<html>
<head>
<script type="text/javascript">
function scrollWindow()
  {
  window.scrollTo(100,500)
  }
</script>
</head>
<body>

<input type="button" onclick="scrollWindow()" value="Scroll" />
<p>SCROLL SCROLL SCROLL SCROLL SCROLL SCROLL SCROLL SCROLL</p>
<br />
<br />
<br />
<br />
<br />
<br />
<br />
```

```
<br />
<p>SCROLL SCROLL SCROLL SCROLL SCROLL SCROLL SCROLL SCROLL</p>
<br />
<br />
<br />
<br />
<br />
<br />
<br />
<br />
<p>SCROLL SCROLL SCROLL SCROLL SCROLL SCROLL SCROLL SCROLL</p>

</body>
</html>
```

HTML DOM setInterval() 方法

定义和用法

setInterval() 方法可按照指定的周期（以毫秒计）来调用函数或计算表达式。
setInterval() 方法会不停地调用函数，直到 clearInterval() 被调用或窗口被关闭。由 setInterval() 返回的 ID 值可用作 clearInterval() 方法的参数。

语法

```
setInterval(code,millisec[,"lang"])
```

参数	描述
code	必需。要调用的函数或要执行的代码串。
millisec	必须。周期性执行或调用 code 之间的时间间隔，以毫秒计。

返回值

一个可以传递给 Window.clearInterval() 从而取消对 code 的周期性执行的值。

实例

```
<html>
<body>

<input type="text" id="clock" size="35" />
<script language=javascript>
var int=self.setInterval("clock()",50)
function clock()
  {
  var t=new Date()
  document.getElementById("clock").value=t
  }
</script>
</form>
<button onclick="int=window.clearInterval(int)">
Stop interval</button>

</body>
</html>
```

HTML DOM setTimeout() 方法

定义和用法

setTimeout() 方法用于在指定的毫秒数后调用函数或计算表达式。

语法

```
setTimeout(code,millisec)
```

参数	描述
code	必需。要调用的函数后要执行的 JavaScript 代码串。
millisec	必需。在执行代码前需等待的毫秒数。

提示和注释

提示：setTimeout() 只执行 code 一次。如果要多次调用，请使用 setInterval() 或者让 code 自身再次调用 setTimeout()。

实例

```
<html>
<head>
<script type="text/javascript">
function timedMsg()
{
var t=setTimeout("alert('5 seconds!')",5000)
}
</script>
</head>

<body>
<form>
<input type="button" value="Display timed alertbox!"
onClick="timedMsg()">
</form>
<p>Click on the button above. An alert box will be
displayed after 5 seconds.</p>
</body>

</html>
```

Navigator 对象

Navigator 对象

Navigator 对象包含有关浏览器的信息。

注释：没有应用于 navigator 对象的公开标准，不过所有浏览器都支持该对象。

IE: Internet Explorer, **F:** Firefox, **O:** Opera.

Navigator 对象集合

集合	描述	IE	F	O
plugins[]	返回对文档中所有嵌入式对象的引用。 该集合是一个 Plugin 对象的数组，其中的元素代表浏览器已经安装的插件。Plug-in 对象提供的是有关插件的信息，其中包括它所支持的 MIME 类型的列表。 虽然 plugins[] 数组是由 IE 4 定义的，但是在 IE 4 中它却总是空的，因为 IE 4 不支持插件和 Plugin 对象。	4	1	9

Navigator 对象属性

属性	描述	IE	F	O
appCodeName	返回浏览器的代码名。	4	1	9
appMinorVersion	返回浏览器的次级版本。	4	No	No
appName	返回浏览器的名称。	4	1	9
appVersion	返回浏览器的平台和版本信息。	4	1	9
browserLanguage	返回当前浏览器的语言。	4	No	9
cookieEnabled	返回指明浏览器中是否启用 cookie 的布尔值。	4	1	9
cpuClass	返回浏览器系统的 CPU 等级。	4	No	No
onLine	返回指明系统是否处于脱机模式的布尔值。	4	No	No
platform	返回运行浏览器的操作系统平台。	4	1	9
systemLanguage	返回 OS 使用的默认语言。	4	No	No
userAgent	返回由客户机发送服务器的 user-agent 头部的值。	4	1	9
userLanguage	返回 OS 的自然语言设置。	4	No	9

Navigator 对象方法

方法	描述	IE	F	O
javaEnabled()	规定浏览器是否启用 Java。	4	1	9
taintEnabled()	规定浏览器是否启用数据污点 (data tainting)。	4	1	9

Navigator 对象描述

Navigator 对象包含的属性描述了正在使用的浏览器。可以使用这些属性进行平台专用的配置。
虽然这个对象的名称显而易见的是 Netscape 的 Navigator 浏览器，但其他实现了 JavaScript 的浏览器也支持这个对象。
Navigator 对象的实例是唯一的，可以用 Window 对象的 navigator 属性来引用它。

HTML DOM appCodeName 属性

定义和用法

appCodeName 属性是一个只读字符串，声明了浏览器的代码名。
在所有以 Netscape 代码为基础的浏览器中，它的值是 "Mozilla"。为了兼容起见，在 Microsoft 的浏览器中，它的值也是 "Mozilla"。

语法

```
navigator.appCodeName
```

实例

```
<html>

<body>
<script type="text/javascript">
document.write("<p>CodeName: ")
document.write(navigator.appCodeName + "</p>")
</script>
</body>

</html>
```

HTML DOM appMinorVersion 属性

定义和用法

appMinorVersion 属性可返回浏览器的次要版本。

语法

```
navigator.appMinorVersion
```

实例

```
<html>

<body>
<script type="text/javascript">
document.write("<p>MinorVersion: ")
document.write(navigator.appMinorVersion + "</p>")
</script>
</body>

</html>
```

HTML DOM appName 属性

定义和用法

appName 属性可返回浏览器的名称。
appName 属性是一个只读到字符串，声明了浏览器的名称。在基于 Netscape 的浏览器中，这个属性的值是 "Netscape"。在 IE 中，这个属性的值是 "Microsoft Internet Explorer"。其他浏览器可以正确地表示自己或者伪装成其他的浏览器以达到兼容性。

语法

```
navigator.appMinorVersion
```

实例

```
<html>

<body>
<script type="text/javascript">
document.write("<p>Name: ")
document.write(navigator.appName + "</p>")
</script>
</body>

</html>
```

HTML DOM appVersion 属性

定义和用法

appVersion 属性可返回浏览器的平台和版本信息。该属性是一个只读的字符串。
该字符串的第一部分是版本号。把该字符串传递给 parseInt() 只能获取主版本号。该属性的其余部分提供了有关浏览器版本的其他细节，包括运行它的操作系统的信息。

语法

```
navigator.appVersion
```

提示和注释

注释：不同浏览器提供的信息的格式是不同的。

实例

```
<html>

<body>
<script type="text/javascript">
document.write("<p>Version: ")
document.write(navigator.appVersion + "</p>")
</script>
```

```
</body>

</html>
```

HTML DOM browserLanguage 属性

定义和用法

browserLanguage 属性可返回当前浏览器的语言。

语法

```
navigator.browserLanguage
```

实例

```
<html>

<body>
<script type="text/javascript">
document.write("<p>BrowserLanguage: ")
document.write(navigator.browserLanguage + "</p>")
</script>
</body>

</html>
```

HTML DOM cookieEnabled 属性

定义和用法

cookieEnabled 属性可返回一个布尔值，如果浏览器启用了 cookie，该属性值为 true。如果禁用了 cookie，则值为 false。

语法

```
navigator.cookieEnabled
```

实例

```
<html>

<body>
<script type="text/javascript">
document.write("<p>CookieEnabled: ")
document.write(navigator.cookieEnabled + "</p>")
</script>
</body>

</html>
```

HTML DOM cpuClass 属性

定义和用法

cpuClass 属性可返回浏览器系统的 CPU 等级。

语法

```
navigator.cpuClassd
```

实例

```
<html>

<body>
<script type="text/javascript">
document.write("<p>CPUClass: ")
document.write(navigator.cpuClass + "</p>")
</script>
</body>
```

```
</html>
```

HTML DOM onLine 属性

定义和用法

onLine 属性是一个只读的布尔值，声明了系统是否处于脱机模式。
注释：在 IE 4+ 中，用户可以在浏览器中选择脱机工作的状态。当脱机工作被选后，系统就进入了脱机状态，内容将从缓存进行读取。

语法

```
navigator.onLine
```

实例

```
<html>

<body>
<script type="text/javascript">
document.write("<p>OnLine: ")
document.write(navigator.onLine + "</p>")
</script>
</body>

</html>
```

HTML DOM platform 属性

定义和用法

platform 属性是一个只读的字符串，声明了运行浏览器的操作系统和（或）硬件平台。
虽然该属性没有标准的值集合，但它有些常用值，比如 "Win32"、"MacPPC" 以及 "Linuxi586"，等等。

语法

```
navigator.platform
```

实例

```
<html>

<body>
<script type="text/javascript">
document.write("<p>Platform: ")
document.write(navigator.platform + "</p>")
</script>
</body>

</html>
```

HTML DOM systemLanguage 属性

定义和用法

systemLanguage 属性可返回操作系统使用的默认语言。

语法

```
navigator.systemLanguage
```

实例

```
<html>

<body>
<script type="text/javascript">
document.write("<p>SystemLanguage: ")
document.write(navigator.systemLanguage + "</p>")
</script>
</body>
```

```
</html>
```

HTML DOM userAgent 属性

定义和用法

userAgent 属性是一个只读的字符串，声明了浏览器用于 HTTP 请求的用户代理头的值。
一般来讲，它是在 navigator.appCodeName 的值之后加上斜线和 navigator.appVersion 的值构成的。
例如：Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.2; SV1; .NET CLR 1.1.4322)。
注：用户代理头：user-agent header。

语法

```
navigator.userAgent
```

实例

```
<html>

<body>
<script type="text/javascript">
document.write("<p>UserAgent: ")
document.write(navigator.userAgent + "</p>")
</script>
</body>

</html>
```

HTML DOM userLanguage 属性

定义和用法

userLanguage 属性可返回操作系统的自然语言设置。

语法

```
navigator.userLanguage
```

实例

```
<html>

<body>
<script type="text/javascript">
document.write("<p>UserLanguage: ")
document.write(navigator.userLanguage + "</p>")
</script>
</body>

</html>
```

HTML DOM javaEnabled() 方法

定义和用法

javaEnabled() 方法可返回一个布尔值，该值指示浏览器是否支持并启用了 Java。如果是，则返回 true，否则返回 false。

语法

```
navigator.javaEnabled()
```

提示和注释

提示：可以使用 navigator.javaEnabled() 来检测当前浏览器是否支持 Java，从而知道浏览器是否能显示 Java 小程序。

实例

下面的例子讲返回当前浏览器是否已启用 Java 的布尔值：

```
<html>
<body>

<script type="text/javascript">
alert(navigator.javaEnabled())
</script>

</body>
</html>
```

输出：

```
true
```

HTML DOM taintEnabled() 方法

定义和用法

taintEnabled() 方法可返回一个布尔值，该值声明了当前浏览器是否启用了 data tainting。

语法

```
navigator.taintEnabled()
```

实例

下面的例子讲返回当前浏览器是否已启用 data tainting 的布尔值：

```
<html>
<body>

<script type="text/javascript">
alert(navigator.taintEnabled())
</script>

</body>
</html>
```

输出：

```
false
```

Screen 对象

Screen 对象

Screen 对象包含有关客户端显示屏幕的信息。
注释：没有应用于 screen 对象的公开标准，不过所有浏览器都支持该对象。
IE: Internet Explorer, **F:** Firefox, **O:** Opera.

Screen 对象属性

属性	描述	IE	F	O
availHeight	返回显示屏幕的高度 (除 Windows 任务栏之外)。	4	1	9
availWidth	返回显示屏幕的宽度 (除 Windows 任务栏之外)。	4	1	9
bufferDepth	设置或返回调色板的比特深度。	4	No	No
colorDepth	返回目标设备或缓冲器上的调色板的比特深度。	4	1	9
deviceXDPI	返回显示屏幕的每英寸水平点数。	6	No	No
deviceYDPI	返回显示屏幕的每英寸垂直点数。	6	No	No
fontSmoothingEnabled	返回用户是否在显示控制面板中启用了字体平滑。	4	No	No
height	返回显示屏幕的高度。	4	1	9
logicalXDPI	返回显示屏幕每英寸的水平方向的常规点数。	6	No	No
logicalYDPI	返回显示屏幕每英寸的垂直方向的常规点数。	6	No	No
pixelDepth	返回显示屏幕的颜色分辨率（比特每像素）。	No	1	9
updateInterval	设置或返回屏幕的刷新率。	4	No	No
width	返回显示器屏幕的宽度。	4	1	9

Screen 对象描述

每个 Window 对象的 screen 属性都引用一个 Screen 对象。Screen 对象中存放着有关显示浏览器屏幕的信息。JavaScript 程序将利用这些信息来优化它们的输出，以达到用户的显示要求。例如，一个程序可以根据显示器的尺寸选择使用大图像还是使用小图像，它还可以根据显示器的颜色深度选择使用 16 位色还是使用 8 位色的图形。另外，JavaScript 程序还能根据有关屏幕尺寸的信息将新的浏览器窗口定位在屏幕中间。

HTML DOM availHeight 属性

定义和用法

availHeight 属性声明了显示浏览器的屏幕的可用高度，以像素计。在 Windows 这样的操作系统中，这个可用高度不包括分配给半永久特性（如屏幕底部的任务栏）的垂直空间。

语法

```
screen.availHeight
```

实例

```
<html>

<body>
<script type="text/javascript">
document.write("<p>Available Height: ")
document.write(screen.availHeight + "</p>")
</script>
</body>

</html>
```

HTML DOM availWidth 属性

定义和用法

availWidth 属性声明了显示浏览器的屏幕的可用宽度，以像素计。在 Windows 这样的操作系统中，这个可用高度不包括分配给半永久特性（如屏幕底部的任务栏）的垂直空间。

语法

```
screen.availWidth
```

实例

```
<html>

<body>
<script type="text/javascript">
document.write("<p>Available Width: ")
document.write(screen.availWidth + "</p>")
</script>
</body>

</html>
```

HTML DOM bufferDepth 属性

定义和用法

bufferDepth 属性设置或返回在 off-screen bitmap buffer 中调色板的比特深度。

语法

```
screen.bufferDepth=number
```

实例

```
<html>

<body>
<script type="text/javascript">
document.write("<p>Buffer Depth: ")
document.write(screen.bufferDepth + "</p>")
</script>
</body>

</html>
```

HTML DOM colorDepth 属性

定义和用法

colorDepth 属性返回目标设备或缓冲器上的调色板的比特深度。

语法

```
screen.colorDepth
```

实例

```
<html>

<body>
<script type="text/javascript">
document.write("<p>Color Depth: ")
document.write(screen.colorDepth + "</p>")
</script>
</body>

</html>
```

HTML DOM deviceXDPI 属性

定义和用法

deviceXDPI 属性返回显示屏幕的每英寸水平点数。

语法

```
screen.deviceXDPI
```

实例

```
<html>

<body>
<script type="text/javascript">
document.write("<p>Device XDPI: ")
document.write(screen.deviceXDPI + "</p>")
</script>
</body>

</html>
```

HTML DOM deviceYDPI 属性

定义和用法

deviceYDPI 属性返回显示屏幕的每英寸垂直点数。

语法

```
screen.deviceYDPI
```

实例

```
<html>

<body>
<script type="text/javascript">
document.write("<p>Device YDPI: ")
document.write(screen.deviceYDPI + "</p>")
</script>
</body>

</html>
```

HTML DOM fontSmoothingEnabled 属性

定义和用法

fontSmoothingEnabled 属性返回用户是否在显示控制面板中启用了字体平滑。

语法

```
sscreen.fontSmoothingEnabled
```

实例

```
<html>

<body>
<script type="text/javascript">
document.write("<p>FontSmoothingEnabled: ")
document.write(screen.fontSmoothingEnabled + "</p>")
</script>
</body>

</html>
```

HTML DOM height 属性

定义和用法

height 属性声明了显示浏览器的屏幕的高度，以像素计。

语法

```
screen.height
```

实例

```
<html>

<body>
<script type="text/javascript">
document.write("<p>Height: ")
document.write(screen.height + "</p>")
</script>
</body>

</html>
```

HTML DOM logicalXDPI 属性

定义和用法

logicalXDPI 属性可返回显示屏幕每英寸的水平方向的常规点数。

语法

```
screen.logicalXDPI
```

实例

```
<html>

<body>
<script type="text/javascript">
document.write("<p>Logical XDPI: ")
document.write(screen.logicalXDPI + "</p>")
</script>
</body>

</html>
```

HTML DOM logicalYDPI 属性

定义和用法

logicalYDPI 属性可返回显示屏幕每英寸的垂直方向的常规点数。

语法

```
screen.logicalYDPI
```

实例

```
<html>

<body>
<script type="text/javascript">
document.write("<p>Logical YDPI: ")
document.write(screen.logicalYDPI + "</p>")
</script>
</body>

</html>
```

HTML DOM pixelDepth 属性

定义和用法

pixelDepth 属性返回显示屏幕的颜色分辨率（比特每像素）。

语法

```
screen.pixelDepth
```

实例

```
<html>

<body>
<script type="text/javascript">
document.write("<p>Pixel Depth: ")
document.write(screen.pixelDepth + "</p>")
</script>
</body>

</html>
```

HTML DOM updateInterval 属性

定义和用法

updateInterval 属性可设置或返回屏幕的刷新率。

语法

```
screen.updateInterval=number
```

实例

```
<html>

<body>
<script type="text/javascript">
document.write("<p>Update Interval: ")
document.write(screen.updateInterval + "</p>")
</script>
</body>

</html>
```

HTML DOM width 属性

定义和用法

width 属性声明了显示浏览器的屏幕的宽度，以像素计。

语法

```
screen.width
```

实例

```
<html>

<body>
<script type="text/javascript">
document.write("<p>Width: ")
document.write(screen.width + "</p>")
</script>
</body>

</html>
```

History 对象

History 对象

History 对象包含用户（在浏览器窗口中）访问过的 URL。
History 对象是 window 对象的一部分，可通过 window.history 属性对其进行访问。
注释：没有应用于 History 对象的公开标准，不过所有浏览器都支持该对象。
IE: Internet Explorer, **F:** Firefox, **O:** Opera.

History 对象属性

属性	描述	IE	F	O
length	返回浏览器历史列表中的 URL 数量。	4	1	9

History 对象方法

方法	描述	IE	F	O
back()	加载 history 列表中的前一个 URL。	4	1	9
forward()	加载 history 列表中的下一个 URL。	4	1	9
go()	加载 history 列表中的某个具体页面。	4	1	9

History 对象描述

History 对象最初设计来表示窗口的浏览历史。但出于隐私方面的原因，History 对象不再允许脚本访问已经访问过的实际 URL。唯一保持使用的功能只有 back()、forward() 和 go() 方法。

例子

下面一行代码执行的操作与单击后退按钮执行的操作一样：

```
history.back()
```

下面一行代码执行的操作与单击两次后退按钮执行的操作一样：

```
history.go(-2)
```

HTML DOM length 属性

定义和用法

length 属性声明了浏览器历史列表中的元素数量。
注释：IE 6 和 Opera 9 以 0 开始，而 Firefox 1.5 以 1 开始。

语法

```
history.length
```

实例

```
<html>
<body>

<script type="text/javascript">
document.write(history.length);
</script>

</body>
</html>
```

输出类似：

```
3
```

HTML DOM back() 方法

定义和用法

back() 方法可加载历史列表中的前一个 URL（如果存在）。
调用该方法的效果等价于点击后退按钮或调用 history.go(-1)。

语法

```
history.back()
```

实例

下面的例子可在页面上创建一个后退按钮：

```
<html>
<head>
<script type="text/javascript">
function goBack()
  {
  window.history.back()
  }
</script>
</head>
<body>

<input type="button" value="Back" onclick="goBack()" />

</body>
</html>
```

HTML DOM forward() 方法

定义和用法

forward() 方法可加载历史列表中的下一个 URL。
调用该方法的效果等价于点击前进按钮或调用 history.go(1)。

语法

```
history.forward()
```

实例

下面的例子可以在页面上创建一个前进按钮：

```
<html>
<head>
<script type="text/javascript">
function goForward()
  {
  window.history.forward()
  }
</script>
</head>
<body>

<input type="button" value="Forward" onclick="goForward()" />

</body>
</html>
```

HTML DOM go() 方法

定义和用法

go() 方法可加载历史列表中的某个具体的页面。

语法

```
history.go(number|URL)
```

说明

URL 参数使用的是要访问的 URL，或 URL 的子串。而 number 参数使用的是要访问的 URL 在 History 的 URL 列表中的相对位置。

实例

下面例子会加载历史列表中的前一个页面：

```
<html>
<head>
<script type="text/javascript">
```

```
function goBack()
  {
  window.history.go(-1)
  }
</script>
</head>
<body>

<input type="button" value="Back" onclick="goBack()" />

</body>
</html>
```

Location 对象

Location 对象

Location 对象包含有关当前 URL 的信息。
Location 对象是 Window 对象的一个部分，可通过 window.location 属性来访问。

例子

把用户带到一个新的地址

```
<html>
<head>
<script type="text/javascript">
function currLocation()
{
alert(window.location)
}
function newLocation()
{
window.location="/index.html"
}
</script>
</head>

<body>
<input type="button" onclick="currLocation()" value="显示当前的 URL">
```

```
<input type="button" onclick="newLocation()" value="改变 URL">
</body>

</html>
```

IE: Internet Explorer, **F:** Firefox, **O:** Opera.

Location 对象属性

属性	描述	IE	F	O
hash	设置或返回从井号 (#) 开始的 URL（锚）。	4	1	9
host	设置或返回主机名和当前 URL 的端口号。	4	1	9
hostname	设置或返回当前 URL 的主机名。	4	1	9
href	设置或返回完整的 URL。	4	1	9
pathname	设置或返回当前 URL 的路径部分。	4	1	9
port	设置或返回当前 URL 的端口号。	4	1	9
protocol	设置或返回当前 URL 的协议。	4	1	9
search	设置或返回从问号 (?) 开始的 URL（查询部分）。	4	1	9

Location 对象方法

属性	描述	IE	F	O
assign()	加载新的文档。	4	1	9
reload()	重新加载当前文档。	4	1	9
replace()	用新的文档替换当前文档。	4	1	9

Location 对象描述

Location 对象存储在 Window 对象的 Location 属性中，表示那个窗口中当前显示的文档的 Web 地址。它的 href 属性存放的是文档的完整 URL，其他属性则分别描述了 URL 的各个部分。这些属性与 Anchor 对象（或 Area 对象）的 URL 属性非常相似。当一个 Location 对象被转换成字符串，href 属性的值被返回。这意味着你可以使用表达式 location 来替代 location.href。

不过 Anchor 对象表示的是文档中的超链接，Location 对象表示的却是浏览器当前显示的文档的 URL（或位置）。但是 Location 对象所能做的远远不止这些，它还能控制浏览器显示的

文档的位置。如果把一个含有 URL 的字符串赋予 Location 对象或它的 href 属性，浏览器就会把新的 URL 所指的文档装载进来，并显示出来。
除了设置 location 或 location.href 用完整的 URL 替换当前的 URL 之外，还可以修改部分 URL，只需要给 Location 对象的其他属性赋值即可。这样做就会创建新的 URL，其中的一部分与原来的 URL 不同，浏览器会将它装载并显示出来。例如，假设设置了 Location 对象的 hash 属性，那么浏览器就会转移到当前文档中的一个指定的位置。同样，如果设置了 search 属性，那么浏览器就会重新装载附加了新的查询字符串的 URL。
除了 URL 属性外，Location 对象的 reload() 方法可以重新装载当前文档，replace() 可以装载一个新文档而无须为它创建一个新的历史记录，也就是说，在浏览器的历史列表中，新文档将替换当前文档。

HTML DOM hash 属性

定义和用法

hash 属性是一个可读可写的字符串，该字符串是 URL 的锚部分（从 # 号开始的部分）。

语法

```
location.hash=anchorname
```

实例

假设当前的 URL 是: http://example.com:1234/test.htm#part2：

```
<html>
<body>

<script type="text/javascript">
document.write(location.hash);
</script>

</body>
</html>
```

输出：

```
#part2
```

HTML DOM host 属性

定义和用法

host 属性是一个可读可写的字符串，可设置或返回当前 URL 的主机名称和端口号。

语法

```
location.host
```

实例

假设当前的 URL 是: http://example.com:1234/test.htm#part2：

```
<html>
<body>

<script type="text/javascript">
document.write(location.host);
</script>

</body>
</html>
```

输出：

```
example.com:1234
```

HTML DOM hostname 属性

定义和用法

hostname 属性是一个可读可写的字符串，可设置或返回当前 URL 的主机名。

语法

```
location.hostname
```

实例

假设当前的 URL 是: http://example.com:1234/test.htm#part2：

```
<html>
<body>

<script type="text/javascript">
document.write(location.hostname);
</script>

</body>
</html>
```

输出：

```
example.com
```

HTML DOM href 属性

定义和用法

href 属性是一个可读可写的字符串，可设置或返回当前显示的文档的完整 URL。
因此，我们可以通过为该属性设置新的 URL，使浏览器读取并显示新的 URL 的内容。

语法

```
location.href=URL
```

实例

假设当前的 URL 是: http://example.com:1234/test.htm#part2：

```
<html>
<body>

<script type="text/javascript">
document.write(location.href);
</script>

</body>
</html>
```

输出：

```
http://example.com:1234/test.htm#part2
```

HTML DOM pathname 属性

定义和用法

pathname 属性是一个可读可写的字符串，可设置或返回当前 URL 的路径部分。

语法

```
location.pathname=path
```

实例

假设当前的 URL 是: http://example.com:1234/test/test.htm#part2：

```
<html>
<body>

<script type="text/javascript">
document.write(location.pathname);
</script>

</body>
</html>
```

输出：

```
/test/test.htm
```

HTML DOM port 属性

定义和用法

port 属性是一个可读可写的字符串，可设置或返回当前 URL 的端口部分。

语法

```
location.port=portnumber
```

实例

假设当前的 URL 是: http://example.com:1234/test.htm#part2：

```
<html>
<body>

<script type="text/javascript">
document.write(location.port);
</script>

</body>
</html>
```

输出：

```
1234
```

HTML DOM protocol 属性

定义和用法

protocol 属性是一个可读可写的字符串，可设置或返回当前 URL 的协议。

语法

```
location.protocol=path
```

实例

假设当前的 URL 是: http://example.com:1234/test.htm#part2：

```
<html>
<body>

<script type="text/javascript">
document.write(location.protocol);
</script>

</body>
</html>
```

输出：

```
http:
```

HTML DOM search 属性

定义和用法

search 属性是一个可读可写的字符串，可设置或返回当前 URL 的查询部分（问号 ? 之后的部分）。

语法

```
location.search=path_from_questionmark
```

实例

假设当前的 URL 是: http://www.w3school.com.cn/tiy/t.asp?f=hdom_loc_search

```
<html>
<body>

<script type="text/javascript">
document.write(location.search);
</script>

</body>
</html>
```

输出：

```
?f=hdom_loc_search
```

HTML DOM assign() 方法

定义和用法

assign() 方法可加载一个新的文档。

语法

```
location.assign(URL)
```

实例

下面的例子将使用 assign() 来加载一个新的文档：

```
<html>
<head>
<script type="text/javascript">
function newDoc()
  {
  window.location.assign("http://www.w3school.com.cn")
  }
</script>
</head>
<body>
```

```
<input type="button" value="Load new document" onclick="newDoc()" />

</body>
</html>
```

HTML DOM reload() 方法

定义和用法

reload() 方法用于重新加载当前文档。

语法

```
location.reload(force)
```

说明

如果该方法没有规定参数，或者参数是 false，它就会用 HTTP 头 If-Modified-Since 来检测服务器上的文档是否已改变。如果文档已改变，reload() 会再次下载该文档。如果文档未改变，则该方法将从缓存中装载文档。这与用户单击浏览器的刷新按钮的效果是完全一样的。如果把该方法的参数设置为 true，那么无论文档的最后修改日期是什么，它都会绕过缓存，从服务器上重新下载该文档。这与用户在单击浏览器的刷新按钮时按住 Shift 健的效果是完全一样。

实例

```
<html>
<head>
<script type="text/javascript">
function reloadPage()
  {
  window.location.reload()
  }
</script>
</head>

<body>
<input type="button" value="Reload page"
onclick="reloadPage()" />
```

```
</body>

</html>
```

HTML DOM replace() 方法

定义和用法

replace() 方法可用一个新文档取代当前文档。

语法

```
location.replace(newURL)
```

说明

replace() 方法不会在 History 对象中生成一个新的记录。当使用该方法时，新的 URL 将覆盖 History 对象中的当前记录。

实例

下面的例子将使用 replace() 方法来替换当前文档：

```
<html>
<head>
<script type="text/javascript">
function replaceDoc()
  {
  window.location.replace("http://www.w3school.com.cn")
  }
</script>
</head>
<body>

<input type="button" value="Replace document" onclick="replaceDoc()"
/>

</body>
```

```
</html>
```

HTML DOM Document 对象

Document 对象

每个载入浏览器的 HTML 文档都会成为 Document 对象。
Document 对象使我们可以从脚本中对 HTML 页面中的所有元素进行访问。
提示：Document 对象是 Window 对象的一部分，可通过 window.document 属性对其进行访问。
IE: Internet Explorer, **F:** Firefox, **O:** Opera, **W3C:** W3C 标准.

Document 对象集合

集合	描述	IE	F	O	W3C
all[]	提供对文档中所有 HTML 元素的访问。	4	1	9	No
anchors[]	返回对文档中所有 Anchor 对象的引用。	4	1	9	Yes
applets	返回对文档中所有 Applet 对象的引用。	-	-	-	-
forms[]	返回对文档中所有 Form 对象引用。	4	1	9	Yes
images[]	返回对文档中所有 Image 对象引用。	4	1	9	Yes
links[]	返回对文档中所有 Area 和 Link 对象引用。	4	1	9	Yes

Document 对象属性

属性	描述	IE	F	O	W3C
body	提供对 <body> 元素的直接访问。 对于定义了框架集的文档，该属性引用最外层的 <frameset>。				
cookie	设置或返回与当前文档有关的所有 cookie。	4	1	9	Yes
domain	返回当前文档的域名。	4	1	9	Yes
lastModified	返回文档被最后修改的日期和时间。	4	1	No	No
referrer	返回载入当前文档的文档的 URL。	4	1	9	Yes

title	返回当前文档的标题。	4	1	9	Yes
URL	返回当前文档的 URL。	4	1	9	Yes

Document 对象方法

方法	描述	IE	F	O	W3C
close()	关闭用 document.open() 方法打开的输出流，并显示选定的数据。	4	1	9	Yes
getElementById()	返回对拥有指定 id 的第一个对象的引用。	5	1	9	Yes
getElementsByName()	返回带有指定名称的对象集合。	5	1	9	Yes
getElementsByTagName()	返回带有指定标签名的对象集合。	5	1	9	Yes
open()	打开一个流，以收集来自任何 document.write() 或 document.writeln() 方法的输出。	4	1	9	Yes
write()	向文档写 HTML 表达式 或 JavaScript 代码。	4	1	9	Yes
writeln()	等同于 write() 方法，不同的是在每个表达式之后写一个换行符。	4	1	9	Yes

Document 对象描述

HTMLDocument 接口对 DOM Document 接口进行了扩展，定义 HTML 专用的属性和方法。很多属性和方法都是 HTMLCollection 对象（实际上是可以用数组或名称索引的只读数组），其中保存了对锚、表单、链接以及其他可脚本元素的引用。

这些集合属性都源自于 0 级 DOM。它们已经被 Document.getElementsByTagName() 所取代，但是仍然常常使用，因为他们很方便。

write() 方法值得注意，在文档载入和解析的时候，它允许一个脚本向文档中插入动态生成的内容。

注意，在 1 级 DOM 中，HTMLDocument 定义了一个名为 getElementById() 的非常有用的方法。在 2 级 DOM 中，该方法已经被转移到了 Document 接口，它现在由 HTMLDocument 继承而不是由它定义了。

HTML DOM all 集合

定义和用法

all 集合返回对文档中所有 HTML 元素的引用。

语法

```
document.all[i]
document.all[name]
document.all.tags[tagname]
```

说明

all[] 是一个多功能的类似数组的对象，它提供了对文档中所有 HTML 元素的访问。all[] 数组源自 IE 4 并且已经被很多其他的浏览器所采用。
all[] 已经被 Document 接口的标准的 getElementById() 方法和 getElementsByTagName() 方法以及 Document 对象的 getElementsByName() 方法所取代。尽管如此，这个 all[] 数组在已有的代码中仍然使用。
all[] 包含的元素保持了最初的顺序，如果你知道它们在数组中的确切数字化位置，可以直接从数组中提取它们。然而，更为常见的是使用 all[] 数组，根据它们的 HTML 属性 name 或 id 来访问元素。如果多个元素拥有指定的 name，将得到共享同一名称的元素的一个数组。

HTML DOM anchors 集合

定义和用法

anchors 集合可返回对文档中所有 Anchor 对象的引用。

语法

```
document.anchors[]
```

实例

```
<html>

<body>
<a name="first">First anchor</a><br />
<a name="second">Second anchor</a><br />
<a name="third">Third anchor</a><br />
<br />

Number of anchors in this document:
<script type="text/javascript">
document.write(document.anchors.length)
</script>
</body>

</html>
```

HTML DOM forms 集合

定义和用法

forms 集合可返回对文档中所有 Form 对象的引用。

语法

```
document.forms[]
```

实例

```
<html>
<body>

<form name="Form1"></form>
<form name="Form2"></form>
<form name="Form3"></form>

<script type="text/javascript">
document.write("This document contains: ")
```

```
document.write(document.forms.length + " forms.")
</script>

</body>
</html>
```

HTML DOM images 集合

定义和用法

images 集合可返回对文档中所有 Image 对象的引用。

语法

```
document.images[]
```

提示和注释

注释：为了与 0 级 DOM 兼容，该集合不包括由 <object> 标记定义的图像。

实例

```
<html>

<body>
<img border="0" src="hackanm.gif" width="48" height="48">
<br />
<img border="0" src="compman.gif" width="107" height="98">
<br /><br />

<script type="text/javascript">
document.write("This document contains: ")
document.write(document.images.length + " images.")
</script>
</body>

</html>
```

HTML DOM links 集合

定义和用法

links 集合可返回对文档中所有 Area 和 Link 对象的引用。

语法

```
document.links[]
```

实例

```
<html>
<body>

<img src="planets.gif"
width="145" height="126"
usemap="#planetmap" />

<map name="planetmap">
<area id="venus" shape="circle"
coords="124,58,8"
alt="Venus"
href="venus.htm" />
</map>
<br />

Number of links in this document:
<script type="text/javascript">
document.write(document.links.length)
</script>

</body>
</html>
```

HTML DOM cookie 属性

定义和用法

cookie 属性可设置或查询与当前文档相关的所有 cookie。

语法

```
document.cookie
```

说明

该属性是一个可读可写的字符串，可使用该属性对当前文档的 cookie 进行读取、创建、修改和删除操作。

提示和注释

提示：该属性的行为与普通的读/写属性是不同的。

实例

```
<html>
<body>

The cookies associated with this document is:
<script type="text/javascript">
document.write(document.cookie)
</script>

</body>
</html>
```

HTML DOM domain 属性

定义和用法

domain 属性可返回下载当前文档的服务器域名。

语法

```
document.domain
```

说明

该属性是一个只读的字符串，包含了载入当前文档的 web 服务器的主机名。

提示和注释

提示：domain 属性可以解决因同源安全策略带来的不同文档的属性共享问题。
点击这里，了解同源安全策略的详细信息。

实例

```
<html>
<body>

The domain name for this document is:
<script type="text/javascript">
document.write(document.domain)
</script>

</body>
</html>
```

HTML DOM lastModified 属性

定义和用法

lastModified 属性可返回文档最后被修改的日期和时间。

语法

```
document.lastModified
```

说明

该值来自于 Last-Modified HTTP 头部，它是由 Web 服务器发送的可选项。

实例

```
<html>
<body>

This document was last modified on:
<script type="text/javascript">
document.write(document.lastModified)
</script>

</body>
</html>
```

HTML DOM referrer 属性

定义和用法

referrer 属性可返回载入当前文档的文档的 URL。

语法

```
document.referrer
```

说明

如果当前文档不是通过超级链接访问的，则为 null。这个属性允许客户端 JavaScript 访问 HTTP 引用头部。

实例

```
<html>
<body>

The referrer of this document is:
<script type="text/javascript">
document.write(document.referrer)
</script>

</body>
</html>
```

HTML DOM title 属性

定义和用法

title 属性可返回当前文档的标题（ HTML title 元素中的文本）。

语法

```
document.title
```

实例

```
<html>
<head>
<title>My title</title>
```

```
</head>

<body>
The title of the document is:
<script type="text/javascript">
document.write(document.title)
</script>
</body>

</html>
```

HTML DOM URL 属性

定义和用法

URL 属性可返回当前文档的 URL。

语法

```
document.URL
```

说明

一般情况下，该属性的值与包含文档的 Window 的 location.href 属性相同。不过，在 URL 重定向发生的时候，这个 URL 属性保存了文档的实际 URL，而 location.href 保存了请求的 URL。

实例

```
<html>

<body>
The URL of this document is:
<script type="text/javascript">
document.write(document.URL)
</script>
</body>
```

```
</html>
```

HTML DOM close() 方法

定义和用法

close() 方法可关闭一个由 document.open 方法打开的输出流，并显示选定的数据。

语法

```
document.close()
```

说明

该方法将关闭 open() 方法打开的文档流，并强制地显示出所有缓存的输出内容。如果您使用 write() 方法动态地输出一个文档，必须记住当你这么做的时候要调用 close() 方法，以确保所有文档内容都能显示。
一旦调用了 close()，就不应该再次调用 write()，因为这会隐式地调用 open() 来擦除当前文档并开始一个新的文档。

实例

```
<html>
<head>
<script type="text/javascript">
function createNewDoc()
  {
  var newDoc=document.open("text/html","replace");
  var txt="<html><body>Learning about the DOM is FUN!</body></html>";
  newDoc.write(txt);
  newDoc.close();
  }
</script>
</head>
<body>

<input type="button" value="Write to a new document"
onclick="createNewDoc()">
```

```
</body>
</html>
```

HTML DOM getElementById() 方法

定义和用法

getElementById() 方法可返回对拥有指定 ID 的第一个对象的引用。

语法

```
document.getElementById(id)
```

说明

HTML DOM 定义了多种查找元素的方法，除了 getElementById() 之外，还有 getElementsByName() 和 getElementsByTagName()。
不过，如果您需要查找文档中的一个特定的元素，最有效的方法是 getElementById()。
在操作文档的一个特定的元素时，最好给该元素一个 id 属性，为它指定一个（在文档中）唯一的名称，然后就可以用该 ID 查找想要的元素。

实例

例子 1

```
<html>
<head>
<script type="text/javascript">
function getValue()
  {
  var x=document.getElementById("myHeader")
  alert(x.innerHTML)
  }
</script>
</head>
<body>

<h1 id="myHeader" onclick="getValue()">This is a header</h1>
<p>Click on the header to alert its value</p>
```

```
</body>
</html>
```

例子 2

getElementById() 是一个重要的方法，在 DOM 程序设计中，它的使用非常常见。我们为您定义了一个工具函数，这样您就可以通过一个较短的名字来使用 getElementById() 方法了：

```
function id(x) {
  if (typeof x == "string") return document.getElementById(x);
  return x;
  }
```

上面这个函数接受元素 ID 作为它们的参数。对于每个这样的参数，您只要在使用前编写 x = id(x) 就可以了。

HTML DOM getElementsByName() 方法

定义和用法

getElementsByName() 方法可返回带有指定名称的对象的集合。

语法

```
document.getElementsByName(name)
```

该方法与 getElementById() 方法相似，但是它查询元素的 name 属性，而不是 id 属性。另外，因为一个文档中的 name 属性可能不唯一（如 HTML 表单中的单选按钮通常具有相同的 name 属性），所有 getElementsByName() 方法返回的是元素的数组，而不是一个元素。

实例

```
<html>
<head>
<script type="text/javascript">
```

```
function getElements()
  {
  var x=document.getElementsByName("myInput");
  alert(x.length);
  }
</script>
</head>
<body>

<input name="myInput" type="text" size="20" /><br />
<input name="myInput" type="text" size="20" /><br />
<input name="myInput" type="text" size="20" /><br />
<br />
<input type="button" onclick="getElements()"
value="How many elements named 'myInput'?" />

</body>
</html>
```

HTML DOM getElementsByTagName() 方法

定义和用法

getElementsByTagName() 方法可返回带有指定标签名的对象的集合。

语法

```
document.getElementsByTagName(tagname)
```

说明

getElementsByTagName() 方法返回元素的顺序是它们在文档中的顺序。
如果把特殊字符串 "*" 传递给 getElementsByTagName() 方法，它将返回文档中所有元素的列表，元素排列的顺序就是它们在文档中的顺序。

提示和注释

注释：传递给 getElementsByTagName() 方法的字符串可以不区分大小写。

实例

例子 1

```
<html>
<head>
<script type="text/javascript">
function getElements()
  {
  var x=document.getElementsByTagName("input");
  alert(x.length);
  }
</script>
</head>
<body>

<input name="myInput" type="text" size="20" /><br />
<input name="myInput" type="text" size="20" /><br />
<input name="myInput" type="text" size="20" /><br />
<br />
<input type="button" onclick="getElements()"
value="How many input elements?" />

</body>
</html>
```

例子 2

可以用 getElementsByTagName() 方法获取任何类型的 HTML 元素的列表。例如，下面的代码可获取文档中所有的表：

```
var tables = document.getElementsByTagName("table");
alert ("This document contains " + tables.length + " tables");
```

例子 3

如果您非常了解文档的结构，也可以使用 getElementsByTagName() 方法获取文档中的一个特定的元素。例如，下面的代码可以获得文档中的第四个段落：

```
var myParagragh = document.getElementsByTagName("p")[3];
```

不过，我们还是认为，如果您需要操作某个特定的元素，使用 getElementById() 方法将更为有效。

HTML DOM open() 方法

定义和用法

open() 方法可打开一个新文档，并擦除当前文档的内容。

语法

```
document.open(mimetype,replace)
```

参数	描述
mimetype	可选。规定正在写的文档的类型。默认值是 "text/html"。
replace	可选。当此参数设置后，可引起新文档从父文档继承历史条目。

说明

该方法将擦除当前 HTML 文档的内容，开始一个新的文档，新文档用 write() 方法或 writeln() 方法编写。

提示和注释

重要事项：调用 open() 方法打开一个新文档并且用 write() 方法设置文档内容后，必须记住用 close 方法关闭文档，并迫使其内容显示出来。
注释：属于被覆盖的文档的一部分的脚本或事件句柄不能调用该方法，因为脚本或事件句柄自身也会被覆盖。

实例

```
<html>
<head>
```

```
<script type="text/javascript">
function createNewDoc()
  {
  var newDoc=document.open("text/html","replace");
  var txt="<html><body>Learning about the DOM is FUN!</body></html>";
  newDoc.write(txt);
  newDoc.close();
  }
</script>
</head>
<body>

<input type="button" value="Write to a new document"
onclick="createNewDoc()">

</body>
</html>
```

HTML DOM write() 方法

定义和用法

write() 方法可向文档写入 HTML 表达式或 JavaScript 代码。
可列出多个参数(exp1,exp2,exp3,...) ，它们将按顺序被追加到文档中。

语法

```
document.write(exp1,exp2,exp3,....)
```

说明

虽然根据 DOM 标准，该方法只接受单个字符串作为参数。不过根据经验，write() 可接受任何多个参数。
我们通常按照两种的方式使用 write() 方法：一是在使用该方在文档中输出 HTML，另一种是在调用该方法的的窗口之外的窗口、框架中产生新文档。在第二种情况中，请务必使用 close() 方法来关闭文档。

实例

```
<html>
<body>

<script type="text/javascript">
document.write("Hello World!");
</script>

</body>
</html>
```

HTML DOM writeln() 方法

定义和用法

writeln() 方法与 write() 方法作用相同，外加可在每个表达式后写一个换行符。

语法

```
document.writeln(exp1,exp2,exp3,....)
```

提示和注释

提示：在编写 <pre> 标记的内容时，该方法很有用。

实例

```
<html>
<body>

<script type="text/javascript">
document.writeln("Hello World!");
</script>

</body>
</html>
```

HTML DOM Anchor 对象

Anchor 对象

Anchor 对象表示 HTML 超链接。
在 HTML 文档中 <a> 标签每出现一次，就会创建 Anchor 对象。
锚可用于创建指向另一个文档的链接（通过 href 属性），或者创建文档内的书签（通过 name 属性）。
您可以通过搜索 Document 对象中的 anchors[] 数组来访问锚，或者使用 document.getElementById()。
IE: Internet Explorer, **F:** Firefox, **O:** Opera, **W3C:** W3C 标准.

Anchor 对象的属性

属性	描述	IE	F	O	W3C
accessKey	设置或返回访问一个链接的快捷键。	5	1	No	Yes
charset	设置或返回被链接资源的字符集。	6	1	9	Yes
coords	设置或返回逗号分隔列表，包含了图像映射中链接的坐标。	6	1	9	Yes
href	设置或返回被链接资源的 URL。	5	1	9	Yes
hreflang	设置或返回被链接资源的语言代码。	6	1	9	Yes
id	设置或返回一个链接的 id。	4	1	9	Yes
innerHTML	设置或返回一个链接的内容。	4	1	9	No
name	设置或返回一个链接的名称。	4	1	9	Yes
rel	设置或返回当前文档与目标 URL 之间的关系。	5	1	No	Yes
rev	设置或返回目标 URL 与之间当前文档的关系。	5	1	No	Yes
shape	设置或返回图像映射中某个链接的形状。	6	1	9	Yes
tabIndex	设置或返回某个链接的 Tab 键控制次序。	6	1	9	Yes
target	设置或返回在何处打开链接。	5	1	9	Yes
type	设置或返回被链接资源的 MIME 类型。	6	1	9	Yes

标准属性

属性	描述	IE	F	O	W3C
className	设置或返回元素的 class 属性。	5	1	9	Yes
dir	设置或返回文本的方向。	5	1	9	Yes
lang	设置或返回元素的语言代码。	5	1	9	Yes
title	设置或返回元素的 title 属性。	5	1	9	Yes

Anchor 对象的方法

方法	描述	IE	F	O	W3C
blur()	把焦点从链接上移开。	5	1	9	Yes
focus()	给链接应用焦点。	5	1	9	Yes

HTML DOM Area 对象

Area 对象

Area 对象代表图像映射的一个区域（图像映射指的是带有可点击区域的图像）
在 HTML 文档中 <area> 标签每出现一次，就会创建一个 Area 对象。
IE: Internet Explorer, **F:** Firefox, **O:** Opera, **W3C:** W3C 标准.

Area 对象的属性

属性	描述	IE	F	O	W3C
accessKey	设置或返回访问某个区域的快捷键。	5	1	No	Yes
alt	设置或返回当浏览器无法显示某个区域时的替换文字。	5	1	9	Yes
coords	设置或返回图像映射中可点击区域的坐标。	5	1	9	Yes
hash	设置或返回某个区域中 URL 的锚部分。	4	1	No	No
host	设置或返回某个区域中 URL 的主机名和端口。	4	1	No	No
href	设置或返回图像映射中链接的 URL。	4	1	9	Yes

id	设置或返回某个区域的 id。	4	1	9	Yes
noHref	设置或返回某个区域是否应是活动的还是非活动的。	5	1	9	Yes
pathname	设置或返回某个区域中的 URL 的路径名。	4	1	9	No
protocol	设置或返回某个区域中的 URL 的协议。	4	1	9	No
search	设置或返回某个区域中 URL 的查询字符串部分。	4	1	9	No
shape	设置或返回图像映射中某个区域的形状。	5	1	9	Yes
tabIndex	设置或返回某个区域的 tab 键控制次序。	5	1	9	Yes
target	设置或返回在何处打开区域中的 link-URL。	4	1	9	Yes

标准属性

属性	描述	IE	F	O	W3C
className	设置或返回元素的 class 属性。	5	1	9	Yes
dir	设置或返回文本的方向。	5	1	9	Yes
lang	设置或返回元素的语言代码。	5	1	9	Yes
title	设置或返回元素的 title。	5	1	9	Yes

HTML DOM Base 对象

Base 对象

Base 对象代表 HTML 的 base 元素。
在 HTML 文档中 <base> 每出现一次，Base 对象就会被创建。
IE: Internet Explorer, **F:** Firefox, **O:** Opera, **W3C:** W3C 标准.

Base 对象属性

属性	描述	IE	F	O	W3C
href	设置或返回针对页面中所有链接的基准 URL。	5	1	9	Yes
id	设置或返回 <base> 元素的 id。	4	1	9	Yes
target	设置或返回针对页面中所有链接的默认目标框架。	5	1	9	Yes

HTML DOM Body 对象

Body 对象

Body 对象代表文档的主体 (HTML body) 。
IE: Internet Explorer, **F:** Firefox, **O:** Opera, **W3C:** W3C 标准.

Body 对象的属性

属性	描述	IE	F	O	W3C
className	设置或返回元素的 class 属性。	5	1	9	Yes
dir	设置或返回文本的方向。	5	1	9	Yes
id	设置或返回 body 的 id。	5	1	9	Yes
lang	设置或返回元素的语言代码。	5	1	9	Yes
title	设置或返回元素的咨询性的标题。	5	1	9	Yes

HTML DOM Button 对象

Button 对象

Button 对象代表一个按钮。
在 HTML 文档中 <button> 标签每出现一次，Button 对象就会被创建。
IE: Internet Explorer, **F:** Firefox, **O:** Opera, **W3C:** W3C 标准.

Button 对象的属性

属性	描述	IE	F	O	W3C
accessKey	设置或返回访问某个按钮的快捷键。	6	1	9	Yes
disabled	设置或返回是否禁用按钮。	6	1	9	Yes
form	返回对包含按钮的表单的引用。	6	1	9	Yes
id	设置或返回按钮的 id。	6	1	9	Yes

name	设置或返回按钮的名称。	6	1	9	Yes
tabIndex	设置或返回按钮的 Tab 键控制次序。	6	1	9	Yes
type	返回按钮的表单类型。	6	1	9	Yes
value	设置或返回显示在按钮上的文本。	6	1	9	Yes

标准属性

属性	描述	IE	F	O	W3C
className	设置或返回元素的 class 属性。	5	1	9	Yes
dir	设置或返回文本的方向。	5	1	9	Yes
lang	设置或返回元素的语言代码。	5	1	9	Yes
title	设置或返回元素的 title 属性。	5	1	9	Yes

HTML DOM Canvas 对象

Canvas 对象

Canvas 对象表示一个 HTML 画布元素 - <canvas>。它没有自己的行为，但是定义了一个 API 支持脚本化客户端绘图操作。
你可以直接在该对象上指定宽度和高度，但是，其大多数功能都可以通过 CanvasRenderingContext2D 对象获得。 这是通过 Canvas 对象的 getContext() 方法并且把直接量字符串 "2d" 作为唯一的参数传递给它而获得的。
<canvas> 标记在 Safari 1.3 中引入，在制作此参考页时，它在 Firefox 1.5 和 Opera 9 中也得到了支持。在 IE 中，<canvas> 标记及其 API 可以使用位于 excanvas.sourceforge.net 的 ExplorerCanvas 开源项目来模拟。
提示：如果希望学习如何使用 <canvas> 来绘制图形，可以访问 Mozilla 提供的 Canvas 教程（英文）以及相应的 中文 Canvas 教程。

Canvas 对象的属性

height 属性
画布的高度。和一幅图像一样，这个属性可以指定为一个整数像素值或者是窗口高度的百分比。当这个值改变的时候，在该画布上已经完成的任何绘图都会擦除掉。默认值是 300。

width 属性

画布的宽度。和一幅图像一样，这个属性可以指定为一个整数像素值或者是窗口宽度的百分比。当这个值改变的时候，在该画布上已经完成的任何绘图都会擦除掉。默认值是 300。

Canvas 对象的方法

方法	描述
getContext()	返回一个用于在画布上绘图的环境。

HTML DOM getContext() 方法

定义和用法

getContext() 方法返回一个用于在画布上绘图的环境。

语法

```
Canvas.getContext(contextID)
```

参数

参数 *contextID* 指定了您想要在画布上绘制的类型。当前唯一的合法值是 "2d"，它指定了二维绘图，并且导致这个方法返回一个环境对象，该对象导出一个二维绘图 API。
提示：在未来，如果 <canvas> 标签扩展到支持 3D 绘图，getContext() 方法可能允许传递一个 "3d" 字符串参数。

返回值

一个 CanvasRenderingContext2D 对象，使用它可以绘制到 Canvas 元素中。

描述

返回一个表示用来绘制的环境类型的环境。其本意是要为不同的绘制类型（2 维、3 维）提供不同的环境。当前，唯一支持的是 "2d"，它返回一个 CanvasRenderingContext2D 对象，该对象实现了一个画布所使用的大多数方法。

HTML DOM Form 对象

Form 对象

Form 对象代表一个 HTML 表单。
在 HTML 文档中 <form> 每出现一次，Form 对象就会被创建。
IE: Internet Explorer, **F:** Firefox, **O:** Opera, **W3C:** W3C 标准.

Form 对象集合

集合	描述	IE	F	O	W3C
elements[]	包含表单中所有元素的数组。	5	1	9	Yes

Form 对象属性

属性	描述	IE	F	O	W3C
acceptCharset	服务器可接受的字符集。	No	No	No	Yes
action	设置或返回表单的 action 属性。	5	1	9	Yes
enctype	设置或返回表单用来编码内容的 MIME 类型。	6	1	9	Yes
id	设置或返回表单的 id。	5	1	9	Yes
length	返回表单中的元素数目。	5	1	9	Yes
method	设置或返回将数据发送到服务器的 HTTP 方法。	5	1	9	Yes
name	设置或返回表单的名称。	5	1	9	Yes
target	设置或返回表单提交结果的 Frame 或 Window 名。	5	1	9	Yes

标准属性

属性	描述	IE	F	O	W3C
className	设置或返回元素的 class 属性。	5	1	9	Yes
dir	设置或返回文本的方向。	5	1	9	Yes

lang	设置或返回元素的语言代码。	5	1	9	Yes
title	设置或返回元素的 title 属性。	5	1	9	Yes

Form 对象方法

方法	描述	IE	F	O	W3C
reset()	把表单的所有输入元素重置为它们的默认值。	5	1	9	Yes
submit()	提交表单。	5	1	9	Yes

Form 对象事件句柄

事件句柄	描述	IE	F	O	W3C
onreset	在重置表单元素之前调用。	5	1	9	Yes
onsubmit	在提交表单之前调用。	5	1	9	Yes

HTML DOM Frame 对象

Frame 对象

Frame 对象代表一个 HTML 框架。
在 HTML 文档中 <frame> 每出现一次，就会创建一个 Frame 对象。
IE: Internet Explorer, **F:** Firefox, **O:** Opera, **W3C:** W3C 标准.

Frame 对象的属性

属性	描述	IE	F	O	W3C
contentDocument	容纳框架的内容的文档。	No	1	9	Yes
frameBorder	设置或返回是否显示框架周围的边框。	5	1	9	Yes
id	设置或返回框架的 id。	4	1	9	Yes
longDesc	设置或返回指向包含框架内容描述文档的 URL。	6	1	9	Yes

marginHeight	设置或返回框架的顶部和底部页空白。	5	1	9	Yes
marginWidth	设置或返回框架的左边缘和右边缘的空白。	5	1	9	Yes
name	设置或返回框架的名称。	5	1	9	Yes
noResize	设置或返回框架是否可调整大小。	5	1	9	Yes
scrolling	设置或返回框架是否可拥有滚动条。	No	1	No	Yes
src	设置或返回应被加载到框架中的文档的 URL。	5	1	9	Yes

标准属性

属性	描述	IE	F	O	W3C
className	设置或返回元素的 class 属性。	5	1	9	Yes
dir	设置或返回文本的方向。	5	1	9	Yes
lang	设置或返回元素的语言代码。	5	1	9	Yes
title	设置或返回元素的 title 属性。	5	1	9	Yes

HTML DOM Frameset 对象

Frameset 对象

Frameset 对象代表 HTML 框架集。
IE: Internet Explorer, **F:** Firefox, **O:** Opera, **W3C:** W3C 标准.

Frameset 对象的属性

属性	描述	IE	F	O	W3C
cols	设置或返回框架集中列的数目。	5	1	9	Yes
id	设置或返回框架集的 id。	4	1	9	Yes
rows	设置或返回框架集中行的数目。	5	1	9	Yes

标准属性

属性	描述	IE	F	O	W3C
className	设置或返回元素的 class 属性。	5	1	9	Yes
dir	设置或返回文本的方向。	5	1	9	Yes
lang	设置或返回元素的语言代码。	5	1	9	Yes
title	设置或返回元素的 title 属性。	5	1	9	Yes

HTML DOM IFrame 对象

IFrame 对象

IFrame 对象代表一个 HTML 的内联框架。
在 HTML 文档中 <iframe> 每出现一次，一个 IFrame 对象就会被创建。
IE: Internet Explorer, **F:** Firefox, **O:** Opera, **W3C:** W3C 标准.

IFrame 对象的属性

属性	描述	IE	F	O	W3C
align	根据周围的文字排列 iframe。	6	1	9	Yes
contentDocument	容纳框架的内容的文档。	No	1	9	Yes
frameBorder	设置或返回是否显示 iframe 周围的边框。	No	1	9	Yes
height	设置或返回 iframe 的高度。	5	1	9	Yes
id	设置或返回 iframe 的 id。	4	1	9	Yes
longDesc	设置或返回描述 iframe 内容的文档的 URL。	6	1	9	Yes
marginHeight	设置或返回 iframe 的顶部和底部的页空白。	5	1	9	Yes
marginWidth	设置或返回 iframe 的左侧和右侧的页空白。	5	1	9	Yes
name	设置或返回 iframe 的名称。	5	1	9	Yes
scrolling	设置或返回 iframe 是否可拥有滚动条。	No	1	No	Yes
src	设置或返回应载入 iframe 中的文档的 URL。	5	1	9	Yes

width	设置或返回 iframe 的宽度。	5	1	9	Yes

标准属性

属性	描述	IE	F	O	W3C
className	设置或返回元素的 class 属性。	5	1	9	Yes
dir	设置或返回文本的方向。	5	1	9	Yes
lang	设置或返回元素的语言代码。	5	1	9	Yes
title	设置或返回元素的 title 属性。	5	1	9	Yes

HTML DOM Image 对象

Image 对象

Image 对象代表嵌入的图像。
<img> 标签每出现一次，一个 Image 对象就会被创建。
IE: Internet Explorer, **F:** Firefox, **O:** Opera, **W3C:** W3C 标准.

Image 对象的属性

属性	描述	IE	F	O	W3C
align	设置或返回与内联内容的对齐方式。	5	1	9	Yes
alt	设置或返回无法显示图像时的替代文本。	5	1	9	Yes
border	设置或返回图像周围的边框。	4	1	9	Yes
complete	返回浏览器是否已完成对图像的加载。	4	1	9	No
height	设置或返回图像的高度。	4	1	9	Yes
hspace	设置或返回图像左侧和右侧的空白。	4	1	9	Yes
id	设置或返回图像的 id。	4	1	9	Yes
isMap	返回图像是否是服务器端的图像映射。	5	1	9	Yes
longDesc	设置或返回指向包含图像描述的文档的 URL。	6	1	9	Yes
lowsrc	设置或返回指向图像的低分辨率版本的 URL。	4	1	9	No

name	设置或返回图像的名称。	4	1	9	Yes
src	设置或返回图像的 URL。	4	1	9	Yes
useMap	设置或返回客户端图像映射的 usemap 属性的值。	5	1	9	Yes
vspace	设置或返回图像的顶部和底部的空白。	4	1	9	Yes
width	设置或返回图像的宽度。	4	1	9	Yes

标准属性

属性	描述	IE	F	O	W3C
className	设置或返回元素的 class 属性。	5	1	9	Yes
title	设置或返回元素的 title。	5	1	9	Yes

Image 对象的事件句柄

事件句柄	描述	IE	F	O	W3C
onabort	当用户放弃图像的装载时调用的事件句柄。	5	1	9	Yes
onerror	在装载图像的过程中发生错误时调用的事件句柄。	5	1	9	Yes
onload	当图像装载完毕时调用的事件句柄。	5	1	9	Yes

HTML DOM Button 对象

Button 对象

Button 对象代表 HTML 文档中的一个按钮。
该元素没有默认的行为，但是必须有一个 onclick 事件句柄以便使用。
在 HTML 文档中 <input type="button"> 标签每出现一次，一个 Button 对象 就会被创建。
您可以通过遍历表单的 elements[] 数组来访问某个按钮，或者通过使用 document.getElementById()。
IE: Internet Explorer, **F:** Firefox, **O:** Opera, **W3C:** W3C 标准.

Button 对象的属性

属性	描述	IE	F	O	W3C
accessKey	设置或返回访问按钮的快捷键。	5	1	9	Yes
alt	设置或返回当浏览器无法显示按钮时供显示的替代文本。	5	1	9	Yes
disabled	设置或返回是否禁用按钮。	5	1	9	Yes
form	返回对包含该按钮的表单对象的引用。	4	1	9	Yes
id	设置或返回按钮的 id。	4	1	9	Yes
name	设置或返回按钮的名称。	4	1	9	Yes
tabIndex	设置或返回按钮的 tab 键控制次序。	5	1	9	Yes
type	返回按钮的表单元素类型。	4	1	9	Yes
value	设置或返回在按钮上显示的文本。	4	1	9	Yes

标准属性

属性	描述	IE	F	O	W3C
className	设置或返回元素的 class 属性。	5	1	9	Yes
dir	设置或返回文本的方向。	5	1	9	Yes
lang	设置或返回元素的语言代码。	5	1	9	Yes
title	设置或返回元素的 title 属性。	5	1	9	Yes

Button 对象的方法

方法	描述	IE	F	O	W3C
blur()	把焦点从元素上移开。	4	1	9	Yes
click()	在某个按钮上模拟一次鼠标单击。	4	1	9	Yes
focus()	为某个按钮赋予焦点。	4	1	9	Yes

HTML DOM Checkbox 对象

Checkbox 对象

Checkbox 对象代表一个 HTML 表单中的 一个选择框。
在 HTML 文档中 <input type="checkbox"> 每出现一次，Checkbox 对象就会被创建。
您可以通过遍历表单的 elements[] 数组来访问某个选择框，或者通过使用 document.getElementById() 。
IE: Internet Explorer, **F:** Firefox, **O:** Opera, **W3C:** W3C 标准.

Checkbox 对象的属性

属性	描述	IE	F	O	W3C
accessKey	设置或返回访问 checkbox 的快捷键。	4	1	9	Yes
alt	设置或返回不支持 checkbox 时显示的替代文本。	5	1	9	Yes
checked	设置或返回 checkbox 是否应被选中。	4	1	9	Yes
defaultChecked	返回 checked 属性的默认值。	4	1	9	Yes
disabled	设置或返回 checkbox 是否应被禁用。	4	1	9	Yes
form	返回对包含 checkbox 的表单的引用。	4	1	9	Yes
id	设置或返回 checkbox 的 id。	4	1	9	Yes
name	设置或返回 checkbox 的名称。	4	1	9	Yes
tabIndex	设置或返回 checkbox 的 tab 键控制次序。	4	1	9	Yes
type	返回 checkbox 的表单元素类型。	4	1	9	Yes
value	设置或返回 checkbox 的 value 属性的值	4	1	9	Yes

标准属性

属性	描述	IE	F	O	W3C
className	设置或返回元素的 class 属性。	5	1	9	Yes
dir	设置或返回文本的方向。	5	1	9	Yes
lang	设置或返回元素的语言代码。	5	1	9	Yes

title	设置或返回元素的 title 属性。	5	1	9	Yes

Checkbox 对象的方法

方法	描述	IE	F	O	W3C
blur()	从 checkbox 上移开焦点。	4	1	9	Yes
click()	模拟在 checkbox 中的一次鼠标点击。	4	1	9	Yes
focus()	为 checkbox 赋予焦点。	4	1	9	Yes

HTML DOM FileUpload 对象

FileUpload 对象

在 HTML 文档中 <input type="file"> 标签每出现一次，一个 FileUpload 对象就会被创建。
该元素包含一个文本输入字段，用来输入文件名，还有一个按钮，用来打开文件选择对话框以便图形化选择文件。
该元素的 value 属性保存了用户指定的文件的名称，但是当包含一个 file-upload 元素的表单被提交的时候，浏览器会向服务器发送选中的文件的内容而不仅仅是发送文件名。
为安全起见，file-upload 元素不允许 HTML 作者或 JavaScript 程序员指定一个默认的文件名。HTML value 属性被忽略，并且对于此类元素来说，value 属性是只读的，这意味着只有用户可以输入一个文件名。当用户选择或编辑一个文件名，file-upload 元素触发 onchange 事件句柄。
您可以通过遍历表单的 elements[] 数组，或者通过使用 document.getElementById()来访问 FileUpload 对象。
IE: Internet Explorer, **F:** Firefox, **O:** Opera, **W3C:** W3C 标准.

FileUpload 对象的属性

属性	描述	IE	F	O	W3C
accept	设置或返回指示文件传输的 MIME 类型的列表（逗号分隔）。				Yes
accessKey	设置或返回访问 FileUpload 对象的快捷键。	4			Yes
alt	设置或返回不支持 <input type="file"> 时显示的替代文字。				Yes

defaultValue	设置或返回 FileUpload 对象的初始值。	4	1		Yes
disabled	设置或返回是否禁用 FileUpload 对象。	4			Yes
form	返回对包含 FileUpload 对象的表单的引用。	4	1		Yes
id	设置或返回 FileUpload 对象的 id。	4	1		Yes
name	设置或返回 FileUpload 对象的名称。	4	1		Yes
tabIndex	设置或返回定义 FileUpload 对象的 tab 键控制次序的索引号。	4			Yes
type	返回表单元素的类型。对于 FileUpload ，则是 "file" 。	4	1		Yes
value	返回由用户输入设置的文本后，FileUpload 对象的文件名。	4	1		Yes

标准属性

属性	描述	IE	F	O	W3C
className	设置或返回元素的 class 属性。	5	1	9	Yes
dir	设置或返回文本的方向。	5	1	9	Yes
lang	设置或返回元素的语言代码。	5	1	9	Yes
title	设置或返回元素的 title 属性。	5	1	9	Yes

FileUpload 对象的方法

方法	描述	IE	F	O	W3C
blur()	从 FileUpload 对象上移开焦点。	4	1		Yes
focus()	为 FileUpload 对象赋予焦点。	4	1		Yes
select()	选取 FileUpload 对象。	4			Yes

HTML DOM Hidden 对象

Hidden 对象

Hidden 对象代表一个 HTML 表单中的某个隐藏输入域。

这种类型的输入元素实际上是隐藏的。这个不可见的表单元素的 value 属性保存了一个要提交给 Web 服务器的任意字符串。如果想要提交并非用户直接输入的数据的话，就是用这种类型的元素。

在 HTML 表单中 <input type="hidden"> 标签每出现一次，一个 Hidden 对象就会被创建。您可通过遍历表单的 elements[] 数组来访问某个隐藏输入域，或者通过使用 document.getElementById()。

IE: Internet Explorer, **F:** Firefox, **O:** Opera, **W3C:** W3C 标准.

Hidden 对象的属性

属性	描述	IE	F	O	W3C
alt	设置或返回当不支持隐藏输入域时显示的替代文本。	5	1	9	Yes
form	返回一个对包含隐藏域的表单的引用。	4	1	9	Yes
id	设置或返回隐藏域的 id。	4	1	9	Yes
name	设置或返回隐藏域的名称。	4	1	9	Yes
type	返回隐藏输入域的表单类型。	4	1	9	Yes
value	设置或返回隐藏域的 value 属性的值。	4	1	9	Yes

标准属性

属性	描述	IE	F	O	W3C
className	设置或返回元素的 class 属性。	5	1	9	Yes
dir	设置或返回文本的方向。	5	1	9	Yes
lang	设置或返回元素的语言代码。	5	1	9	Yes
title	设置或返回元素的 title 属性。	5	1	9	Yes

HTML DOM Password 对象

Password 对象

Password 对象代表 HTML 表单中的密码字段。

HTML 的 <input type="password"> 标签在表单上每出现一次，一个 Password 对象就会被创建。

该文本输入字段供用户输入某些敏感的数据，比如密码等。当用户输入的时候，他的输入是被掩盖的（例如使用星号*），以防止旁边的人从他背后看到输入的内容。不过需要注意的是，当表单提交时，输入是用明文发送的。
与类型为 "text" 的元素类似，当用户改变显示值时，它会触发 onchange 事件句柄。
您可以通过遍历表单的 elements[] array 来访问密码字段，或者通过使用 document.getElementById() 。
IE: Internet Explorer, **F:** Firefox, **O:** Opera, **W3C:** W3C 标准.

Password 对象属性

属性	描述	IE	F	O	W3C
accessKey	设置或返回访问密码字段的快捷键。	4	1	9	Yes
alt	设置或返回当不支持密码字段时显示的替代文字。	5	1	9	Yes
defaultValue	设置或返回密码字段的默认值。	4	1	9	Yes
disabled	设置或返回是否应被禁用密码字段。	5	1	9	Yes
form	返回对包含此密码字段的表单的引用。	4	1	9	Yes
id	设置或返回密码字段的 id。	4	1	9	Yes
maxLength	设置或返回密码字段中字符的最大数目。	4	1	9	Yes
name	设置或返回密码字段的名称。	4	1	9	Yes
readOnly	设置或返回密码字段是否应当是只读的。	4	1	9	Yes
size	设置或返回密码字段的长度。	4	1	9	Yes
tabIndex	设置或返回密码字段的 tab 键控制次序。	4	1	9	Yes
type	返回密码字段的表单元素类型。	4	1	9	Yes
value	设置或返回密码字段的 value 属性的值。	4	1	9	Yes

标准属性

属性	描述	IE	F	O	W3C
className	设置或返回元素的 class 属性。	5	1	9	Yes
dir	设置或返回文本的方向。	5	1	9	Yes
lang	设置或返回元素的语言代码。	5	1	9	Yes
title	设置或返回元素的 title 属性。	5	1	9	Yes

Password 对象方法

属性	描述	IE	F	O	W3C
blur()	从密码字段移开焦点。	4	1	9	Yes
focus()	为密码字段赋予焦点。	4	1	9	Yes
select()	选取密码字段中的文本。	4	1	9	Yes

HTML DOM Radio 对象

Radio 对象

Radio 对象代表 HTML 表单中的单选按钮。
在 HTML 表单中 <input type="radio"> 每出现一次，一个 Radio 对象就会被创建。
单选按钮是表示一组互斥选项按钮中的一个。当一个按钮被选中，之前选中的按钮就变为非选中的。
当单选按钮被选中或不选中时，该按钮就会触发 onclick 事件句柄。
您可通过遍历表单的 elements[] 数组来访问 Radio 对象，或者通过使用 document.getElementById()。
IE: Internet Explorer, **F:** Firefox, **O:** Opera, **W3C:** W3C 标准.

Radio 对象属性

属性	描述	IE	F	O	W3C
accessKey	设置或返回访问单选按钮的快捷键。	4	1	9	Yes
alt	设置或返回在不支持单选按钮时显示的替代文本。	5	1	9	Yes
checked	设置或返回单选按钮的状态。	4	1	9	Yes
defaultChecked	返回单选按钮的默认状态。	4	1	9	Yes
disabled	设置或返回是否禁用单选按钮。	5	1	9	Yes
form	返回一个对包含此单选按钮的表单的引用。	4	1	9	Yes
id	设置或返回单选按钮的 id。	4	1	9	Yes
name	设置或返回单选按钮的名称。	4	1	9	Yes
tabIndex	设置或返回单选按钮的 tab 键控制次序。	4	1	9	Yes

type	返回单选按钮的表单类型。	4	1	9	Yes
value	设置或返回单选按钮的 value 属性的值。	4	1	9	Yes

标准属性

属性	描述	IE	F	O	W3C
className	设置或返回元素的 class 属性。	5	1	9	Yes
dir	设置或返回文本的方向。	5	1	9	Yes
lang	设置或返回元素的语言代码。	5	1	9	Yes
title	设置或返回元素的 title 属性。	5	1	9	Yes

Radio 对象方法

方法	描述	IE	F	O	W3C
blur()	从单选按钮移开焦点。	No	1	9	Yes
click()	在单选按钮上模拟一次鼠标点击。	No	2	9	Yes
focus()	为单选按钮赋予焦点。	No	1	9	Yes

HTML DOM Reset 对象

Reset 对象

Reset 对象代表 HTML 表单中的一个重置按钮。
在 HTML 表单中 <input type="reset"> 标签每出现一次，一个 Reset 对象就会被创建。
当重置按钮被点击，包含它的表单中所有输入元素的值都重置为它们的默认值。默认值由 HTML value 属性或 JavaScript 的 defaultValue 属性指定。
重置按钮在重置表单之前触发 onclick 句柄，并且这个句柄可以通过返回 fasle 来取消。
参阅 Form.reset() 方法 和 Form.onreset 事件句柄。
您可以通过遍历表单的 elements[] 数组来访问某个重置按钮，或者通过使用 document.getElementById()。
IE: Internet Explorer, **F:** Firefox, **O:** Opera, **W3C:** W3C 标准.

Reset 对象属性

属性	描述	IE	F	O	W3C
accesskey	设置或返回访问重置按钮的快捷键。	4	1	9	Yes
alt	设置或返回当浏览器不支持重置按钮时供显示的替代文本。	5	1	9	Yes
disabled	设置或返回重置按钮是否应被禁用。	5	1	9	Yes
form	返回一个对包含此重置按钮的表单对象的引用。	4	1	9	Yes
id	设置或返回重置按钮的 id。	4	1	9	Yes
name	设置或返回重置按钮的名称。	4	1	9	Yes
tabIndex	设置或返回重置按钮的 tab 键控制次序。	4	1	9	Yes
type	返回重置按钮的表单元素类型。	4	1	9	Yes
value	设置或返回重置按钮上显示的文本。	4	1	9	Yes

标准属性

属性	描述	IE	F	O	W3C
className	设置或返回元素的 class 属性。	5	1	9	Yes
dir	设置或返回文本的方向。	5	1	9	Yes
lang	设置或返回元素的语言代码。	5	1	9	Yes
title	设置或返回元素的 title 属性。	5	1	9	Yes

Reset 对象方法

方法	描述	IE	F	O	W3C
blur()	从重置按钮上移开焦点。	4	1	9	Yes
click()	在重置按钮上模拟一次鼠标点击。	4	1	9	Yes
focus()	为重置按钮赋予焦点。	4	1	9	Yes

HTML DOM Submit 对象

Submit 对象

Submit 对象代表 HTML 表单中的一个提交按钮 (submit button)。
在 HTML 表单中 <input type="submit"> 标签每出现一次，一个 Submit 对象就会被创建。
在表单提交之前，触发 onclick 事件句柄，并且一个句柄可以通过返回 fasle 来取消表单提交。
参阅 Form.submit() 方法 和 Form.onsubmit 事件句柄。
实例：表单验证
您可以通过遍历表单的 elements[] 数组来访问某个提交按钮，或者通过使用 document.getElementById()。
IE: Internet Explorer, **F:** Firefox, **O:** Opera, **W3C:** W3C 标准.

Submit 对象属性

属性	描述	IE	F	O	W3C
accessKey	设置或返回访问提交按钮的快捷键。	4	1	9	Yes
alt	设置或返回当浏览器不支持提交按钮时供显示的替代文本。	5	1	9	Yes
disabled	设置或返回提交按钮是否应被禁用。	5	1	9	Yes
form	返回一个对包含此提交按钮的表单的引用。	4	1	9	Yes
id	设置或返回提交按钮的 id。	4	1	9	Yes
name	设置或返回提交按钮的名称。	4	1	9	Yes
tabIndex	设置或返回提交按钮的 tab 键控制次序。	4	1	9	Yes
type	返回提交按钮的表单元素类型。	4	1	9	Yes
value	设置或返回在提交按钮上显示的文本。	4	1	9	Yes

标准属性

属性	描述	IE	F	O	W3C
className	设置或返回元素的 class 属性。	5	1	9	Yes
dir	设置或返回文本的方向。	5	1	9	Yes

lang	设置或返回元素的语言代码。	5	1	9	Yes
title	设置或返回元素的 title 属性。	5	1	9	Yes

Submit 对象方法

方法	描述	IE	F	O	W3C
blur()	从提交按钮上移开焦点。	4	1	9	Yes
click()	在提交按钮上模拟一次鼠标点击。	4	1	9	Yes
focus()	为提交按钮赋予焦点。	4	1	9	Yes

HTML DOM Text 对象

Text 对象

Text 对象代表 HTML 表单中的文本输入域。
在 HTML 表单中 <input type="text"> 每出现一次，Text 对象就会被创建。
该元素可创建一个单行的文本输入字段。当用户编辑显示的文本并随后把输入焦点转移到其他元素的时候，会触发 onchange 事件句柄。
您可以使用 HTML <textarea> 标记来创建多行文本输入。参阅 Textarea 对象。
对于掩码文本输入，把 <input type="text"> 中的 type 设置为 "password"。参阅 Input Password。
您可以通过表单的 elements[] 数组来访问文本输入域，或者通过使用 document.getElementById()。
IE: Internet Explorer, **F:** Firefox, **O:** Opera, **W3C:** W3C 标准.

Text 对象属性

属性	描述	IE	F	O	W3C
accessKey	设置或返回访问文本域的快捷键。	4	1	9	Yes
alt	设置或返回当浏览器不支持文本域时供显示的替代文本。	5	1	9	Yes
defaultValue	设置或返回文本域的默认值。	4	1	9	Yes
disabled	设置或返回文本域是否应被禁用。	5	1	9	Yes

form	返回一个对包含文本域的表单对象的引用。	4	1	9	Yes
id	设置或返回文本域的 id。	4	1	9	Yes
maxLength	设置或返回文本域中的最大字符数。	4	1	9	Yes
name	设置或返回文本域的名称。	4	1	9	Yes
readOnly	设置或返回文本域是否应是只读的。	4	1	9	Yes
size	设置或返回文本域的尺寸。	4	1	9	Yes
tabIndex	设置或返回文本域的 tab 键控制次序。	4	1	9	Yes
type	返回文本域的表单元素类型。	4	1	9	Yes
value	设置或返回文本域的 value 属性的值。	4	1	9	Yes

标准属性

属性	描述	IE	F	O	W3C
className	设置或返回元素的 class 属性。	5	1	9	Yes
dir	设置或返回文本的方向。	5	1	9	Yes
lang	设置或返回元素的语言代码。	5	1	9	Yes
title	设置或返回元素的 title 属性。	5	1	9	Yes

Text 对象方法

方法	描述	IE	F	O	W3C
blur()	从文本域上移开焦点。	4	1	9	Yes
focus()	在文本域上设置焦点。	4	1	9	Yes
select()	选取文本域中的内容。	4	1	9	Yes

HTML DOM Link 对象

Link 对象

Link 对象代表某个 HTML 的 <link> 元素。<link> 元素可定义两个链接文档之间的关系。

<link> 元素被定义于 HTML 文档的 head 部分。
IE: Internet Explorer, **F:** Firefox, **O:** Opera, **W3C:** W3C 标准.

Link 对象属性

属性	描述	IE	F	O	W3C
charset	设置或返回目标 URL 的字符编码。	4	1	9	Yes
disabled	设置或返回目标 URL 是否当被禁用。	4	1	9	Yes
href	设置或返回被链接资源的 URL。	4	1	9	Yes
hreflang	设置或返回目标 URL 的基准语言。	4	1	9	Yes
id	设置或返回某个 <link> 元素的 id。	4	1	9	Yes
media	设置或返回文档显示的设备类型。	6	1	9	Yes
name	设置或返回 <link> 元素的名称。	4	No	No	Yes
rel	设置或返回当前文档与目标 URL 之间的关系。	4	1	9	Yes
rev	设置或返回目标 URL 与当前文档之间的关系。	4	1	9	Yes
type	设置或返回目标 URL 的 MIME 类型。	4	1	9	Yes

标准属性

属性	描述	IE	F	O	W3C
dir	设置或返回文本的方向。	5	1	9	Yes
lang	设置或返回元素的语言代码。	5	1	9	Yes

HTML DOM Meta 对象

Meta 对象

Meta 对象代表 HTML 的 一个 <meta> 元素。
<meta> 元素可提供有关某个 HTML 元素的元信息 (meta-information)，比如描述、针对搜索引擎的关键词以及刷新频率。
IE: Internet Explorer, **F:** Firefox, **O:** Opera, **W3C:** W3C 标准.

Meta 对象属性

属性	描述	IE	F	O	W3C
content	设置或返回 <meta> 元素的 content 属性的值。	5	1	9	Yes
httpEquiv	把 content 属性连接到一个 HTTP 头部。	5	1	9	Yes
name	把 content 属性连接到某个名称。	5	1	9	Yes
scheme	设置或返回用于解释 content 属性的值的格式。	6	1	9	Yes

HTML DOM content 属性

定义和用法

content 属性设置或返回 <meta> 元素的 content 属性的值。

语法

```
metaObject.content=text
```

实例

本例的 <meta> 标签为搜索引擎定义了一些关键词。以下代码返回 content 属性的值：

```
<html>
<head>
<meta name="keywords" content="HTML,DHTML,CSS,JavaScript" />
</head>
<body>

<script type="text/javascript">
x=document.getElementsByTagName("meta")[0];
document.write("Meta content: " + x.content);
</script>

</body>
</html>
```

HTML DOM httpEquiv 属性

定义和用法

httpEquiv 属性把 content 属性连接到 HTTP 头部。

语法

```
metaObject.httpEquiv=content-type|expires|refresh|set-cookie
```

实例

本例每 5 秒刷新页面：

```
<html>
<head>
<meta http-equiv="refresh" content="5" />
</head>
<body>

<script type="text/javascript">
x=document.getElementsByTagName("meta")[0];
document.write("Http equiv: " + x.httpEquiv);
</script>

</body>
</html>
```

HTML DOM scheme 属性

定义和用法

scheme 属性设置或返回用于解释 content 属性的值的格式。

语法

```
metaObject.scheme=format
```

实例

本例返回用于解释 content 属性的值的格式：

```
<html>
<head>
<meta name="revised" content="2006-11-03" scheme="YYYY-MM-DD" />
</head>
```

```
<body>

<script type="text/javascript">
x=document.getElementsByTagName("meta")[0];
document.write("Content format: " + x.scheme);
</script>

</body>

</html>
```

HTML DOM Object 对象

Object 对象

Object 对象代表 HTML 的 <object> 元素。
<object> 元素用于在网页中包含对象，比如：图像、音频、视频、Java applet、ActiveX、PDF、Flash 等。
W3C: W3C 标准.

Object 对象属性

属性	描述	W3C
align	设置或返回对象相对于周围文本的对齐方式。	Yes
archive	设置或返回一个字符串，用于实现对象的存档功能。	Yes
border	设置或返回围绕对象的边框。	Yes
code	设置或返回文件的 URL，该文件包含已编译的 Java 类。	Yes
codeBase	设置或返回组件的 URL。	Yes
codeType		Yes
data		Yes
declare		Yes
form	返回对对象的父表单的引用。	Yes
height	设置或返回对象的高度。	Yes

hspace	设置或返回对象的水平外边距。	Yes
name	设置或返回对象的名称。	Yes
standby	设置或返回在加载对象时返回的消息。	Yes
type	设置或返回通过 data 属性下载的数据的内容类型。	Yes
useMap		Yes
vspace	设置或返回对象的垂直外边距。	Yes
width	设置或返回对象的宽度。	Yes

标准属性

属性	描述	W3C
className	设置或返回元素的 class 属性。	Yes
dir	设置或返回文本的方向。	Yes
lang	设置或返回元素的语言代码。	Yes
title	设置或返回元素的 title 属性。	Yes

HTML DOM Option 对象

Option 对象

Option 对象代表 HTML 表单中下拉列表中的一个选项。
在 HTML 表单中 <option> 标签每出现一次，一个 Option 对象就会被创建。
您可通过表单的 elements[] 数组访问一个 Option 对象，或者通过使用 document.getElementById()。
IE: Internet Explorer, **F:** Firefox, **O:** Opera, **W3C:** W3C 标准.

Option 对象的属性

属性	描述	IE	F	O	W3C
defaultSelected	返回 selected 属性的默认值。	4	1	9	Yes
disabled	设置或返回选项是否应被禁用。	4	1	9	Yes

form	返回对包含该元素的 <form> 元素的引用。	4	1	9	Yes
id	设置或返回选项的 id。	4	1	9	Yes
index	返回下拉列表中某个选项的索引位置。	4	1	9	Yes
label	设置或返回选项的标记 （仅用于选项组）。	6			Yes
selected	设置或返回 selected 属性的值。	4	1	9	Yes
text	设置或返回某个选项的纯文本值。	4	1	9	Yes
value	设置或返回被送往服务器的值。	4	1	9	Yes

标准属性

属性	描述	IE	F	O	W3C
className	设置或返回元素的 class 属性。	5	1	9	Yes
dir	设置或返回文本的方向。	5	1	9	Yes
lang	设置或返回元素的语言代码。	5	1	9	Yes
title	设置或返回元素的 title 属性。	5	1	9	Yes

HTML DOM Select 对象

Select 对象

Select 对象代表 HTML 表单中的一个下拉列表。
在 HTML 表单中，<select> 标签每出现一次，一个 Select 对象就会被创建。
您可通过遍历表单的 elements[] 数组来访问某个 Select 对象，或者使用 document.getElementById()。
IE: Internet Explorer, **F:** Firefox, **O:** Opera, **W3C:** W3C 标准.

Select 对象集合

集合	描述	IE	F	O	W3C
options[]	返回包含下拉列表中的所有选项的一个数组。	4	1	9	Yes

Select 对象属性

属性	描述	IE	F	O	W3C
disabled	设置或返回是否应禁用下拉列表。	5	1	9	Yes
form	返回对包含下拉列表的表单的引用。	4	1	9	Yes
id	设置或返回下拉列表的 id。	4	1	9	Yes
length	返回下拉列表中的选项数目。	4	1	9	Yes
multiple	设置或返回是否选择多个项目。	4	1	9	Yes
name	设置或返回下拉列表的名称。	4	1	9	Yes
selectedIndex	设置或返回下拉列表中被选项目的索引号。	4	1	9	Yes
size	设置或返回下拉列表中的可见行数。	4	1	9	Yes
tabIndex	设置或返回下拉列表的 tab 键控制次序。	5	1	9	Yes
type	返回下拉列表的表单类型。	4	1	9	Yes

标准属性

属性	描述	IE	F	O	W3C
className	设置或返回元素的 class 属性。	5	1	9	Yes
dir	设置或返回文本的方向。	5	1	9	Yes
lang	设置或返回元素的语言代码。	5	1	9	Yes
title	设置或返回元素的 title 属性。	5	1	9	Yes

Select 对象方法

方法	描述	IE	F	O	W3C
add()	向下拉列表添加一个选项。	4	1	9	Yes
blur()	从下拉列表移开焦点。	4	1	9	Yes
focus()	在下拉列表上设置焦点。	4	1	9	Yes
remove()	从下拉列表中删除一个选项。	4	1	9	Yes

Select 对象事件句柄

事件句柄	描述	IE	F	O	W3C
onchange	当改变选择时调用的事件句柄。	4	1	9	Yes

HTML DOM Table 对象

Table 对象

Table 对象代表一个 HTML 表格。
在 HTML 文档中 <table> 标签每出现一次，一个 Table 对象就会被创建。
IE: Internet Explorer, **F:** Firefox, **O:** Opera, **W3C:** W3C 标准.

Table 对象集合

集合	描述	IE	F	O	W3C
cells[]	返回包含表格中所有单元格的一个数组。	5	1	1	No
rows[]	返回包含表格中所有行的一个数组。	4	1	9	Yes
tBodies[]	返回包含表格中所有 tbody 的一个数组。	4			Yes

HTML DOM cells 集合

定义和用法

cells 集合返回表格中所有单元格的一个数组。

语法

```
tableObject.cells[]
```

实例

下面的例子使用 cells 来显示出第一个单元格的内容：

```
<html>
```

```
<head>
<script type="text/javascript">
function cell()
  {
  var x=document.getElementById('myTable').rows[0].cells;
  alert(x[0].innerHTML);
  }
</script>
</head>
<body>

<table id="myTable" border="1">
<tr>
<td>cell 1</td>
<td>cell 2</td>
</tr>
<tr>
<td>cell 3</td>
<td>cell 4</td>
</tr>
</table>
<br />
<input type="button" onclick="cell()" value="Alert first cell">

</body>
</html>
```

HTML DOM rows 集合

定义和用法

rows 集合返回表格中所有行（TableRow 对象）的一个数组，即一个 HTMLCollection。
该集合包括 <thead>、<tfoot> 和 <tbody> 中定义的所有行。

语法

```
tableObject.rows[]
```

实例

下面的例子使用了 rows 集合来显示第一行中的内容：

```
<html>
```

```
<head>
<script type="text/javascript">
function showRow()
  {
  alert(document.getElementById('myTable').rows[0].innerHTML)
  }
</script>
</head>
<body>

<table id="myTable" border="1">
<tr>
<td>Row1 cell1</td>
<td>Row1 cell2</td>
</tr>
<tr>
<td>Row2 cell1</td>
<td>Row2 cell2</td>
</tr>
</table>
<br />
<input type="button" onclick="showRow()" value="Show innerHTML">

</body>
</html>
```

Table 对象属性

属性	描述	IE	F	O	W3C
align	表在文档中的水平对齐方式。（已废弃）	-	-	-	-
bgColor	表的背景颜色。（已废弃）	-	-	-	-
border	设置或返回表格边框的宽度。	4	1	9	Yes
caption	对表格的 <caption> 元素的引用。	4	1	9	Yes
cellPadding	设置或返回单元格内容和单元格边框之间的空白量。	4	1	9	Yes
cellSpacing	设置或返回在表格中的单元格之间的空白量。	4	1	9	Yes
frame	设置或返回表格的外部边框。	4	1	9	Yes
id	设置或返回表格的 id。	4	1	9	Yes
rules	设置或返回表格的内部边框（行线）。	4	1	9	Yes

summary	设置或返回对表格的描述（概述）。	6	1	9	Yes
tFoot	返回表格的 TFoot 对象。如果不存在该元素，则为 null。	4	1	9	Yes
tHead	返回表格的 THead 对象。如果不存在该元素，则为 null。	4	1	9	Yes
width	设置或返回表格的宽度。	4	1	9	Yes

HTML DOM caption 属性

定义和用法

caption 属性设置或返回表格的 caption 元素。
<caption> 元素定义了一个表格标题。<caption> 标签必须紧跟在 <table> 标签之后，每个表格仅能规定一个 caption。通常，caption 会位于表格之上居中的位置。

语法

```
tableObject.caption=captionObject
```

实例

下面的例子可返回表格的 <caption> 元素的文本：

```
<html>
<head>
<script type="text/javascript">
function alertCaption()
  {
  alert(document.getElementById("table1").caption.innerHTML);
  }
</script>
</head>
<body>

<table id="table1" border="1">
<caption>This is a table caption</caption>
<tr>
<th>Firstname</th>
<th>Lastname</th>
</tr>
<tr>
<td>Peter</td>
```

```
<td>Griffin</td>
</tr>
</table>
<br />
<input type="button" onclick=alertCaption()
value="Alert table caption" />

</body>
</html>
```

HTML DOM cellPadding 属性

定义和用法

cellPadding 属性可设置或返回单元格边框与单元格内容之间的空白量（以像素为单位）。

语法

```
tableObject.cellPadding=pixels
```

实例

The following example changes the cell padding of a table:

```
<html>
<head>
<script type="text/javascript">
function padding()
  {
  document.getElementById('myTable').cellPadding="15"
  }
</script>
</head>
<body>

<table id="myTable" border="1">
<tr>
<td>100</td>
<td>200</td>
</tr>
<tr>
<td>300</td>
<td>400</td>
```

```
</tr>
</table>
<br />
<input type="button" onclick="padding()" value="Change Cellpadding">

</body>
</html>
```

HTML DOM cellSpacing 属性

定义和用法

cellSpacing 属性可设置或返回在表格中的单元格之间的空白量（以像素为单位）。

语法

tableObject.cellSpacing=pixels

实例

下面的例子更改了表格的 cellSpacing：

```
<html>
<head>
<script type="text/javascript">
function spacing()
  {
  document.getElementById('myTable').cellSpacing="15"
  }
</script>
</head>
<body>

<table id="myTable" border="1">
<tr>
<td>100</td>
<td>200</td>
</tr>
<tr>
<td>300</td>
<td>400</td>
</tr>
</table>
```

```
<br />
<input type="button" onclick="spacing()" value="Change Cellspacing">

</body>
</html>
```

HTML DOM frame 属性

定义和用法

frame 属性可设置或返回表格的外部边框。

语法

```
tableObject.frame=void|above|below|hsides|vsides|lhs|rhs|box|bord
er
```

实例

下面的例子设置了表格的两种不同的边框：

```
<html>
<head>
<script type="text/javascript">
function aboveFrames()
  {
  document.getElementById('myTable').frame="above"
  }
function belowFrames()
  {
  document.getElementById('myTable').frame="below"
  }
</script>
</head>
<body>

<table id="myTable">
<tr>
<td>100</td>
<td>200</td>
</tr>
<tr>
<td>300</td>
```

```
<td>400</td>
</tr>
</table>
<br />
<input type="button" onclick="aboveFrames()"
value="Show above frames">
<input type="button" onclick="belowFrames()"
value="Show below frames">

</body>
</html>
```

HTML DOM rules 属性

定义和用法

rules 属性可设置或返回表格的内部边线。

语法

tableObject.rules=none|groups|rows|cols|all

实例

下面的例子设置了表格的两种不同的内部边线：

```
<html>
<head>
<script type="text/javascript">
function rowRules()
  {
  document.getElementById('myTable').rules="rows"
  }
function colRules()
  {
  document.getElementById('myTable').rules="cols"
  }
</script>
</head>
<body>

<table id="myTable" border="1">
<tr>
```

```
<td>Row1 cell1</td>
<td>Row1 cell2</td>
</tr>
<tr>
<td>Row2 cell1</td>
<td>Row2 cell2</td>
</tr>
<tr>
<td>Row3 cell1</td>
<td>Row3 cell2</td>
</tr>
</table>
<br />
<input type="button" onclick="rowRules()"
value="Show only row borders">
<input type="button" onclick="colRules()"
value="Show only col borders">

</body>
</html>
```

HTML DOM summary 属性

定义和用法

summary 属性可设置或返回表格的概述。
summary 属性为语音合成或非可视的浏览器规定了表格的概述。

语法

```
tableObject.summary=text
```

实例

下面的例子可返回表格的概述：

```
<html>
<head>
<script type="text/javascript">
function alertSummary()
  {
  alert(document.getElementById("table1").summary);
  }
```

```
</script>
</head>
<body>

<table id="table1" border="1"
summary="Example table of employees">
<tr>
<th>Firstname</th>
<th>Lastname</th>
</tr>
<tr>
<td>Peter</td>
<td>Griffin</td>
</tr>
</table>
<br />
<input type="button" onclick=alertSummary()
value="Alert table summary" />

</body>
</html>
```

标准属性

属性	描述	IE	F	O	W3C
className	设置或返回元素的 class 属性。	5	1	9	Yes
dir	设置或返回文本的方向。	5	1	9	Yes
lang	设置或返回元素的语言代码。	5	1	9	Yes
title	设置或返回元素的 title 属性。	5	1	9	Yes

Table 对象方法

方法	描述	IE	F	O	W3C
createCaption()	为表格创建一个 caption 元素。	4	1	9	Yes
createTFoot()	在表格中创建一个空的 tFoot 元素。	4	1	9	Yes
createTHead()	在表格中创建一个空的 tHead 元素。	4	1	9	Yes
deleteCaption()	从表格删除 caption 元素以及其内容。	4	1	9	Yes

deleteRow()	从表格删除一行。	4	1	9	Yes
deleteTFoot()	从表格删除 tFoot 元素及其内容。	4	1	9	Yes
deleteTHead()	从表格删除 tHead 元素及其内容。	4	1	9	Yes
insertRow()	在表格中插入一个新行。	4	1	9	Yes

HTML DOM createCaption() 方法

定义和用法

createCaption() 方法用于在表格中获取或创建 <caption> 元素。

语法

```
tableObject.createCaption()
```

返回值

返回一个 HTMLElement 对象，表示该表的 <caption> 元素。如果该表格已经有了标题，则返回它。如果该表没有 <caption> 元素，则创建一个新的空 <caption> 元素，把它插入表格，并返回它。

实例

下面的例子为表格创建了一个标题：

```
<html>
<head>
<script type="text/javascript">
function createCaption()
{
var x=document.getElementById('myTable').createCaption()
x.innerHTML="My table caption"
}
</script>
</head>
<body>

<table id="myTable" border="1">
<tr>
<td>Row1 cell1</td>
<td>Row1 cell2</td>
```

```
</tr>
<tr>
<td>Row2 cell1</td>
<td>Row2 cell2</td>
</tr>
</table>
<br />
<input type="button" onclick="createCaption()"
value="Create caption">

</body>
</html>
```

HTML DOM createTFoot() 方法

定义和用法

createTFoot() 方法用于在表格中获取或创建 <tfoot> 元素。

语法

```
tableObject.createTFoot()
```

返回值

返回一个 TableSection，表示该表的 <tfoot> 元素。如果该表格已经有了脚注，则返回它。如果该表没有脚注，则创建一个新的空 <tfoot> 元素，把它插入表格，并返回它。

实例

下面的例子为表格创建了一个脚注：

```
<html>
<head>
<script type="text/javascript">
function createCaption()
{
var x=document.getElementById('myTable').createTFoot()
x.innerHTML="My table foot"
}
</script>
</head>
<body>
```

```
<table id="myTable" border="1">
<tr>
<td>Row1 cell1</td>
<td>Row1 cell2</td>
</tr>
<tr>
<td>Row2 cell1</td>
<td>Row2 cell2</td>
</tr>
</table>
<br />
<input type="button" onclick="createCaption()"
value="Create caption">

</body>
</html>
```

HTML DOM createTHead() 方法

定义和用法

createTHead() 方法用于在表格中获取或创建 <thead> 元素。

语法

```
tableObject.createTHead()
```

返回值

返回一个 TableSection，表示该表的 <thead> 元素。如果该表格已经有了表头，则返回它。如果该表没有表头，则创建一个新的空 <thead> 元素，把它插入表格，并返回它。

实例

下面的例子为表格创建了一个脚注：

```
<html>
<head>
<script type="text/javascript">
function createCaption()
{
var x=document.getElementById('myTable').createTHead()
```

```
x.innerHTML="My table head"
}
</script>
</head>
<body>

<table id="myTable" border="1">
<tr>
<td>Row1 cell1</td>
<td>Row1 cell2</td>
</tr>
<tr>
<td>Row2 cell1</td>
<td>Row2 cell2</td>
</tr>
</table>
<br />
<input type="button" onclick="createCaption()"
value="Create caption">

</body>
</html>
```

HTML DOM deleteCaption() 方法

定义和用法

deleteCaption() 方法用于删除表格的 caption 元素及其内容。

语法

```
tableObject.deleteCaption()
```

说明

如果该表有 <caption> 元素，则从文档树种删除它。否则，什么也不做。

实例

下面的例子删除表格的标题：

```
<html>
<head>
```

```
<script type="text/javascript">
function deleteCaption()
  {
  document.getElementById('myTable').deleteCaption()
  }
</script>
</head>
<body>

<table id="myTable" border="1">
<caption>My table caption</caption>
<tr>
<th>Firstname</th>
<th>Lastname</th>
</tr>
<tr>
<td>Peter</td>
<td>Griffin</td>
</tr>
</table>
<br />
<input type="button" onclick="deleteCaption()"
value="Delete caption" />

</body>
</html>
```

HTML DOM deleteRow() 方法

定义和用法

deleteRow() 方法用于从表格删除指定位置的行。

语法

```
tableObject.deleteRow(index)
```

说明

参数 index 指定了要删除的行在表中的位置。行的编码顺序就是他们在文档源代码中出现的顺序。<thead> 和 <tfoot> 中的行与表中其它行一起编码。

实例

The following example deletes the first row of the table:

```
<html>
<head>
<script type="text/javascript">
function delRow()
  {
  document.getElementById('myTable').deleteRow(0)
  }
</script>
</head>
<body>

<table id="myTable" border="1">
<tr>
<td>Row1 cell1</td>
<td>Row1 cell2</td>
</tr>
<tr>
<td>Row2 cell1</td>
<td>Row2 cell2</td>
</tr>
</table>
<br />
<input type="button" onclick="delRow()"
value="Delete first row">

</body>
</html>
```

HTML DOM deleteTFoot() 方法

定义和用法

deleteTFoot() 方法用于从表格删除 <tfoot> 元素。

语法

```
tableObject.deleteTFoot()
```

说明

如果该表有 <tfoot> 元素，则将它从文档树种删除，否则什么也不做。

实例

下面的例子删除表的脚注：

```
<html>
<head>
<script type="text/javascript">
function delRow()
  {
  document.getElementById('myTable').deleteTFoot()
  }
</script>
</head>
<body>

<table id="myTable" border="1">
<tr>
<td>Row1 cell1</td>
<td>Row1 cell2</td>
</tr>
<tr>
<td>Row2 cell1</td>
<td>Row2 cell2</td>
</tr>
<tfoot><td>my table foot</td></tfoot>
</table>
<br />
<input type="button" onclick="delRow()"
value="Delete table foot">

</body>
</html>
```

HTML DOM deleteTHead() 方法

定义和用法

deleteTHead() 方法用于从表格删除 <thead> 元素。

语法

```
tableObject.deleteTHead()
```

说明

如果该表有 <thead> 元素，则将它从文档树种删除，否则什么也不做。

实例

下面的例子删除表的脚注：

```
<html>
<head>
<script type="text/javascript">
function delRow()
  {
  document.getElementById('myTable').deleteTHead()
  }
</script>
</head>
<body>

<table id="myTable" border="1">
<thead><td>my table head</td></thead>
<tr>
<td>Row1 cell1</td>
<td>Row1 cell2</td>
</tr>
<tr>
<td>Row2 cell1</td>
<td>Row2 cell2</td>
</tr>
</table>
<br />
<input type="button" onclick="delRow()"
value="Delete table foot">

</body>
</html>
```

HTML DOM insertRow() 方法

定义和用法

insertRow() 方法用于在表格中的指定位置插入一个新行。

语法

```
tableObject.insertRow(index)
```

返回值

返回一个 TableRow，表示新插入的行。

说明

该方法创建一个新的 TableRow 对象，表示一个新的 <tr> 标记，并把它插入表中的指定位置。
新行将被插入 index 所在行之前。若 index 等于表中的行数，则新行将被附加到表的末尾。如果表是空的，则新行将被插入到一个新的 <tbody> 段，该段自身会被插入表中。

抛出

若参数 index 小于 0 或大于等于表中的行数，该方法将抛出代码为 INDEX_SIZE_ERR 的 DOMException 异常。

提示和注释

提示：可以用 TableRow.insertCell() 方法给新创建的行添加内容。

实例

下面的例子在表格的开头插入一个新行：

```
<html>
<head>
<script type="text/javascript">
function insRow()
  {
  document.getElementById('myTable').insertRow(0)
  }
</script>
</head>

<body>
<table id="myTable" border="1">
<tr>
<td>Row1 cell1</td>
```

```
<td>Row1 cell2</td>
</tr>
<tr>
<td>Row2 cell1</td>
<td>Row2 cell2</td>
</tr>
</table>
<br />
<input type="button" onclick="insRow()"
value="Insert new row">

</body>
</html>
```

HTML DOM TableCell 对象

TableCell 对象

TableCell 对象代表一个 HTML 表格单元格。
在一个 HTML 文档中 <td> 标签每出现一次，一个 TableCell 对象就会被创建。
IE: Internet Explorer, **F:** Firefox, **O:** Opera, **W3C:** W3C 标准.

TableCell 对象属性

属性	描述	IE	F	O	W3C
abbr	设置或返回单元格中内容的缩写版本。	6	1	9	Yes
align	设置或返回单元格内部数据的水平排列方式。	4	1	9	Yes
axis	设置或返回相关单元格的一个逗号分隔的列表。	6	1	9	Yes
cellIndex	返回单元格在某行的单元格集合中的位置。	4	1	9	Yes
ch	设置或返回单元格的对齐字符。				Yes
chOff	设置或返回单元格的对齐字符的偏移量。				Yes
colSpan	单元格横跨的列数。	4	1	9	Yes
headers	设置或返回 header-cell 的 id 值。				Yes
id	设置或返回单元格的 id。	4	1	9	Yes

innerHTML	设置或返回单元格的开始标签和结束标签之间的 HTML。	4	1	9	No
rowSpan	设置或返回单元格可横跨的行数。	4	1	9	Yes
scope	设置或返回此单元格是否可提供标签信息。				Yes
vAlign	设置或返回表格单元格内数据的垂直排列方式。	4	1	9	Yes
width	设置或返回单元格的宽度。	4	1	9	Yes

HTML DOM abbr 属性

定义和用法

abbr 属性可设置或返回表元中内容的缩写版本（针对非可视的媒介，比如语音和盲文）。

语法

```
tabledataObject.abbr=text
```

实例

下面的例子可提示某个单元格的 abbr：

```
<html>
<head>
<script type="text/javascript">
function alertAbbr()
  {
  alert(document.getElementById('td1').abbr);
  }
</script>
</head>
<body>

<table border="1">
<tr>
<th>Firstname</th>
<th>Lastname</th>
</tr>
<tr>
<td id="td1" abbr="Some text">Peter</td>
<td id="td2">Griffin</td>
</tr>
```

```
</table>
<br />
<input type="button" onclick="alertAbbr()"
value="Alert abbr" />

</body>
</html>
```

HTML DOM axis 属性

定义和用法

axis 属性可设置或返回一个由逗号分隔的列表，其中包含了一组类型（有关联的单元格）的名称。
axis 属性可为某个单元格定义类型名称。

语法

```
tabledataObject.axis=text
```

实例

下面的例子可返回单元格的 axis：

```
<html>
<head>
<script type="text/javascript">
function alertAxis()
  {
  alert(document.getElementById('td3').axis);
  }
</script>
</head>
<body>

<table border="1">
<tr>
<th>Firstname</th>
<th>Lastname</th>
<th>City</th>
</tr>
<tr>
<td id="td1">Harry</td>
```

```
<td id="td2">Potter</td>
<td id="td3" axis="location">NY</td>
</tr>
<tr>
<td id="td4">Peter</td>
<td id="td5">Griffin</td>
<td id="td6" axis="location">Las Vegas</td>
</tr>
</table>
<br />
<input type="button" onclick="alertAxis()"
value="Alert axis" />

</body>
</html>
```

HTML DOM cellIndex 属性

定义和用法

cellIndex 属性可返回一行的单元格集合中单元格的位置。

语法

```
tabledataObject.cellIndex
```

实例

下面的例子可返回两个单元格的位置（cell index）：

```
<html>
<head>
<script type="text/javascript">
function alertTd1()
  {
  alert(document.getElementById('td1').cellIndex);
  }
function alertTd2()
  {
  alert(document.getElementById('td2').cellIndex);
  }
</script>
</head>
```

```
<body>

<table border="1">
<tr>
<th>Firstname</th>
<th>Lastname</th>
</tr>
<tr>
<td id="td1">Peter</td>
<td id="td2">Griffin</td>
</tr>
</table>
<br />
<input type="button" onclick="alertTd1()"
value="Alert cell index of 'Peter'" />
<br />
<input type="button" onclick="alertTd2()"
value="Alert cell index of 'Griffin'" />

</body>
</html>
```

HTML DOM colSpan 属性

定义和用法

colSpan 属性可设置或返回表元横跨的列数。

语法

```
tabledataObject.colSpan=number_of_columns
```

实例

下面的例子更改了表元横跨的列数：

```
<html>
<head>
<script type="text/javascript">
function changeColSpan()
  {
  document.getElementById("td1").colSpan="2";
  }
```

```
</script>
</head>
<body>

<table border="1">
<tr>
<th>Firstname</th>
<th>Lastname</th>
</tr>
<tr>
<td id="td1">Peter</td>
<td id="td2">Griffin</td>
</tr>
</table>
<br />
<input type="button" onclick=changeColSpan()
value="Change colspan" />

</body>
</html>
```

HTML DOM rowSpan 属性

定义和用法

rowSpan 属性可设置或返回表元横跨的行数。

语法

```
tabledataObject.rowSpan=number_of_rows
```

实例

下面的例子更改了表元横跨的行数：

```
<html>
<head>
<script type="text/javascript">
function changeRowSpan()
  {
  document.getElementById("td2").rowSpan="2";
  }
</script>
```

```
</head>
<body>

<table border="1">
<tr>
<th>Firstname</th>
<th>Lastname</th>
</tr>
<tr>
<td id="td1">Peter</td>
<td id="td2">Griffin</td>
</tr>
<tr>
<td id="td3">Harry</td>
<td id="td4">Potter</td>
</tr>
</table>
<br />
<input type="button" onclick=changeRowSpan()
value="Change rowspan" />

</body>
</html>
```

HTML DOM vAlign 属性

定义和用法

vAlign 属性设置或返回单元格内数据的垂直排列方式。

语法

```
tabledataObject.vAlign=top|middle|bottom|baseline
```

实例

下面的例子更改了单元格内数据的垂直排列方式：

```
<html>
<head>
<script type="text/javascript">
function topAlign()
  {
```

```
  document.getElementById('td1').vAlign="top";
  document.getElementById('td2').vAlign="top";
  document.getElementById('td3').vAlign="top";
  }
</script>
</head>
<body>

<table width="50%" border="1">
<tr>
<th>Firstname</th>
<th>Lastname</th>
<th>Text</th>
</tr>
<tr>
<td id="td1">Peter</td>
<td id="td2">Griffin</td>
<td id="td3">Hello my name is Peter Griffin.
I need a long text for this example.
I need a long text for this example.</td>
</tr>
</table>
<br />
<input type="button" onclick="topAlign()"
value="Top-align table cells" />

</body>
</html>
```

标准属性

属性	描述	IE	F	O	W3C
className	设置或返回元素的 class 属性。	5	1	9	Yes
dir	设置或返回文本的方向。	5	1	9	Yes
lang	设置或返回元素的语言代码。	5	1	9	Yes
title	设置或返回元素的 title 属性。	5	1	9	Yes

HTML DOM TableRow 对象

TableRow 对象

TableRow 对象代表一个 HTML 表格行。
在 HTML 文档中 <tr> 标签每出现一次，一个 TableRow 对象就会被创建。
IE: Internet Explorer, **F:** Firefox, **O:** Opera, **W3C:** W3C 标准.

TableRow 对象集合

集合	描述	IE	F	O	W3C
cells[]	返回包含行中所有单元格的一个数组。	4	1	9	Yes

TableRow 对象属性

属性	描述	IE	F	O	W3C
align	设置或返回在行中数据的水平排列。	4	1	9	Yes
ch	设置或返回在行中单元格的对齐字符。				Yes
chOff	设置或返回在行中单元格的对齐字符的偏移量。				Yes
id	设置或返回行的 id。	4	1	9	Yes
innerHTML	设置或返回行的开始标签和结束标签之间的 HTML。	5	1	9	No
rowIndex	返回该行在表中的位置。	4	1	9	Yes
sectionRowIndex	返回在 tBody 、tHead 或 tFoot 中，行的位置。				Yes
vAlign	设置或返回在行中的数据的垂直排列方式。	4	1	9	Yes

HTML DOM rowIndex 属性

定义和用法

rowIndex 属性返回某一行在表格的行集合中的位置（row index）。

语法

```
tablerowObject.rowIndex
```

实例

下面的例子返回了某一行在表格中的位置：

```
<html>
<head>
<script type="text/javascript">
function alertRowIndex()
  {
  alert(document.getElementById("tr1").rowIndex);
  }
</script>
</head>
<body>

<table border="1">
<tr id="tr1">
<th>Firstname</th>
<th>Lastname</th>
</tr>
<tr id="tr2">
<td>Peter</td>
<td>Griffin</td>
</tr>
</table>
<br />
<input type="button" onclick="alertRowIndex()"
value="Alert row index" />

</body>
</html>
```

HTML DOM vAlign 属性

定义和用法

vAlign 属性设置或返回数据在行中的垂直对齐方式。

语法

```
tablerowObject.vAlign=top|bottom|middle|baseline
```

实例

下面的例子更改了数据在行中的垂直排列方式：

```
<html>
<head>
<script type="text/javascript">
function topAlign()
  {
  document.getElementById('tr2').vAlign="top";
  }
</script>
</head>
<body>

<table width="50%" border="1">
<tr id="tr1">
<th>Firstname</th>
<th>Lastname</th>
<th>Text</th>
</tr>
<tr id="tr2">
<td>Peter</td>
<td>Griffin</td>
<td>Hello my name is Peter Griffin.
I need a long text for this example.
I need a long text for this example.</td>
</tr>
</table>
<br />
<input type="button" onclick="topAlign()"
value="Top-align table row" />

</body>
</html>
```

TableRow 对象方法

方法	描述	IE	F	O	W3C
deleteCell()	删除行中的指定的单元格。	4	1	9	Yes

insertCell()	在一行中的指定位置插入一个空的 <td> 元素。	4	1	9	Yes

HTML DOM deleteCell() 方法

定义和用法

deleteCell() 方法用于删除表格行中的单元格（<td> 元素）。

语法

```
tablerowObject.deleteCell(index)
```

说明

参数 index 是要删除的表元在行中的位置。
该方法将删除表行中指定位置的表元。

抛出

若参数 index 小于 0 或大于等于行中的的表元数，该方法将抛出代码为 INDEX_SIZE_ERR 的 DOMException 异常。

实例

下面的例子从行中删除了一个单元格：

```
<html>
<head>
<script type="text/javascript">
function delCell()
  {
  document.getElementById('tr2').deleteCell(0)
  }
</script>
</head>
<body>

<table border="1">
<tr id="tr1">
<th>Firstname</th>
<th>Lastname</th>
</tr>
<tr id="tr2">
```

```
<td>Peter</td>
<td>Griffin</td>
</tr>
</table>
<br />
<input type="button" onclick="delCell()" value="Delete cell">

</body>
</html>
```

HTML DOM insertCell() 方法

定义和用法

insertCell() 方法用于在 HTML 表的一行的指定位置插入一个空的 <td> 元素。

语法

```
tablerowObject.insertCell(index)
```

返回值

一个 TableCell 对象，表示新创建并被插入的 <td> 元素。

说明

该方法将创建一个新的 <td> 元素，把它插入行中指定的位置。新单元格将被插入当前位于 index 指定位置的表元之前。如果 index 等于行中的单元格数，则新单元格被附加在行的末尾。
请注意，该方法只能插入 <td> 数据表元。若需要给行添加头表元，必须用 Document.createElement() 方法和 Node.insertBefore() 方法（或相关的方法）创建并插入一个 <th> 元素。

抛出

若参数 index 小于 0 或大于等于行中的的表元数，该方法将抛出代码为 INDEX_SIZE_ERR 的 DOMException 异常。

实例

下面的例子在表格行中插入了一个单元格：

```
<html>
```

```
<head>
<script type="text/javascript">
function insCell()
  {
  var x=document.getElementById('tr2').insertCell(0)
  x.innerHTML="John"
  }
</script>
</head>
<body>

<table border="1">
<tr id="tr1">
<th>Firstname</th>
<th>Lastname</th>
</tr>
<tr id="tr2">
<td>Peter</td>
<td>Griffin</td>
</tr>
</table>
<br />
<input type="button" onclick="insCell()" value="Insert cell">

</body>
</html>
```

HTML DOM Textarea 对象

Textarea 对象

Textarea 对象代表 HTML 表单中的一个文本区 (text-area)。在表单中 <textarea> 标签每出现一次，一个 Textarea 对象就会被创建。
您可以通过索引相应表单的元素数组来访问某个 Textarea 对象，或者使用 getElementById()。
IE: Internet Explorer, **F:** Firefox, **O:** Opera, **W3C:** W3C 标准.

Textarea Object Properties

属性	描述	IE	F	O	W3C

accessKey	设置或返回访问 textarea 的键盘快捷键。	4	1	9	Yes
cols	设置或返回 textarea 的宽度。	4	1	9	Yes
defaultValue	设置或返回文本框中的初始内容。	4	1	9	Yes
disabled	设置或返回 textarea 是否应当被禁用。	5	1	9	Yes
form	返回对包含该 textarea 的表单对象的引用。	4	1	9	Yes
id	设置或返回某个 textarea 的 id。	4	1	9	Yes
name	设置或返回 textarea 的名称。	4	1	9	Yes
readOnly	设置或返回 textarea 是否应当是只读的。	4	1	9	Yes
rows	设置或返回 textarea 的高度。	4	1	9	Yes
tabIndex	设置或返回 textarea 的 tab 键控制次序。	4	1	9	Yes
type	返回该文本框的表单类型。	4	1	9	Yes
value	设置或返回在 textarea 中的文本。	4	1	9	Yes

标准属性

属性	描述	IE	F	O	W3C
className	设置或返回元素的 class 属性。	5	1	9	Yes
dir	设置或返回文本的方向。	5	1	9	Yes
lang	设置或返回元素的语言代码。	5	1	9	Yes
title	设置或返回元素的 title 属性。	5	1	9	Yes

Textarea 对象方法

方法	描述	IE	F	O	W3C
blur()	从 textarea 移开焦点。	4	1	9	Yes
focus()	在 textarea 上设置焦点。	4	1	9	Yes
select()	选择 textarea 中的文本。	4	1	9	Yes

Textarea 对象事件句柄

事件句柄	描述	IE	F	O	W3C
onchange	当输入值改变时调用的事件句柄	4	1	9	Ye

HTML DOM Style 对象

Style 对象

Style 对象代表一个单独的样式声明。可从应用样式的文档或元素访问 Style 对象。

使用 Style 对象属性的语法：

```
document.getElementById("id").style.property="值"
```

Style 对象的属性：

- 背景
- 边框和边距
- 布局
- 列表
- 杂项
- 定位
- 打印
- 滚动条
- 表格
- 文本
- 规范

IE: Internet Explorer, **M:** 仅适用于 Mac IE, **W:** 仅适用于 Windows IE, **F:** Firefox, **O:** Opera
W3C: 万维网联盟 World Wide Web Consortium (Internet 标准).

Background 属性

属性	描述	IE	F	O	W3C
background	在一行中设置所有的背景属性	4	1	9	Yes
backgroundAttachment	设置背景图像是否固定或随页面滚动	4	1	9	Yes

backgroundColor	设置元素的背景颜色	4	1	9	Yes
backgroundImage	设置元素的背景图像	4	1	9	Yes
backgroundPosition	设置背景图像的起始位置	4	No	No	Yes
backgroundPositionX	设置 backgroundPosition 属性的 X 坐标	4	No	No	No
backgroundPositionY	设置 backgroundPosition 属性的 Y 坐标	4	No	No	No
backgroundRepeat	设置是否及如何重复背景图像	4	1	9	Yes

Border 和 Margin 属性

属性	描述	IE	F	O	W3C
border	在一行设置四个边框的所有属性	4	1	9	Yes
borderBottom	在一行设置底边框的所有属性	4	1	9	Yes
borderBottomColor	设置底边框的颜色	4	1	9	Yes
borderBottomStyle	设置底边框的样式	4	1	9	Yes
borderBottomWidth	设置底边框的宽度	4	1	9	Yes
borderColor	设置所有四个边框的颜色 (可设置四种颜色)	4	1	9	Yes
borderLeft	在一行设置左边框的所有属性	4	1	9	Yes
borderLeftColor	设置左边框的颜色	4	1	9	Yes
borderLeftStyle	设置左边框的样式	4	1	9	Yes
borderLeftWidth	设置左边框的宽度	4	1	9	Yes
borderRight	在一行设置右边框的所有属性	4	1	9	Yes
borderRightColor	设置右边框的颜色	4	1	9	Yes
borderRightStyle	设置右边框的样式	4	1	9	Yes
borderRightWidth	设置右边框的宽度	4	1	9	Yes
borderStyle	设置所有四个边框的样式 (可设置四种样式)	4	1	9	Yes
borderTop	在一行设置顶边框的所有属性	4	1	9	Yes
borderTopColor	设置顶边框的颜色	4	1	9	Yes
borderTopStyle	设置顶边框的样式	4	1	9	Yes
borderTopWidth	设置顶边框的宽度	4	1	9	Yes

borderWidth	设置所有四条边框的宽度 (可设置四种宽度)	4	1	9	Yes
margin	设置元素的边距 (可设置四个值)	4	1	9	Yes
marginBottom	设置元素的底边距	4	1	9	Yes
marginLeft	设置元素的左边距	4	1	9	Yes
marginRight	设置元素的右边据	4	1	9	Yes
marginTop	设置元素的顶边距	4	1	9	Yes
outline	在一行设置所有的 outline 属性	5M	1	9	Yes
outlineColor	设置围绕元素的轮廓颜色	5M	1	9	Yes
outlineStyle	设置围绕元素的轮廓样式	5M	1	9	Yes
outlineWidth	设置围绕元素的轮廓宽度	5M	1	9	Yes
padding	设置元素的填充 (可设置四个值)	4	1	9	Yes
paddingBottom	设置元素的下填充	4	1	9	Yes
paddingLeft	设置元素的左填充	4	1	9	Yes
paddingRight	设置元素的右填充	4	1	9	Yes
paddingTop	设置元素的顶填充	4	1	9	Yes

Layout 属性

属性	描述	IE	F	O	W3C
clear	设置在元素的哪边不允许其他的浮动元素	4	1	9	Yes
clip	设置元素的形状	4	1	9	Yes
content	设置元信息	5M	1		Yes
counterIncrement	设置其后是正数的计数器名称的列表。其中整数指示每当元素出现时计数器的增量。默认是 1。	5M	1		Yes
counterReset	设置其后是正数的计数器名称的列表。其中整数指示每当元素出现时计数器被设置的值。默认是 0。	5M	1		Yes
cssFloat	设置图像或文本将出现（浮动）在另一元素中的何处。	5M	1	9	Yes
cursor	设置显示的指针类型	4	1	9	Yes

direction	设置元素的文本方向	5	1	9	Yes
display	设置元素如何被显示	4	1	9	Yes
height	设置元素的高度	4	1	9	Yes
markerOffset	设置 marker box 的 principal box 距离其最近的边框边缘的距离	5M	1		Yes
marks	设置是否 cross marks 或 crop marks 应仅仅被呈现于 page box 边缘之外	5M	1		Yes
maxHeight	设置元素的最大高度	5M	1	9	Yes
maxWidth	设置元素的最大宽度	5M	1	9	Yes
minHeight	设置元素的最小高度	5M	1	9	Yes
minWidth	设置元素的最小宽度	5M	1	9	Yes
overflow	规定如何处理不适合元素盒的内容	4	1	9	Yes
verticalAlign	设置对元素中的内容进行垂直排列	4	1	No	Yes
visibility	设置元素是否可见	4	1	9	Yes
width	设置元素的宽度	4	1	9	Yes

List 属性

属性	描述	IE	F	O	W3C
listStyle	在一行设置列表的所有属性	4	1	9	Yes
listStyleImage	把图像设置为列表项标记	4	1	No	Yes
listStylePosition	改变列表项标记的位置	4	1	9	Yes
listStyleType	设置列表项标记的类型	4	1	9	Yes

Positioning 属性

属性	描述	IE	F	O	W3C
bottom	设置元素的底边缘距离父元素底边缘的之上或之下的距离	5	1	9	Yes
left	置元素的左边缘距离父元素左边缘的左边或右边的距离	4	1	9	Yes

position	把元素放置在 static, relative, absolute 或 fixed 的位置	4	1	9	Yes
right	置元素的右边缘距离父元素右边缘的左边或右边的距离	5	1	9	Yes
top	设置元素的顶边缘距离父元素顶边缘的之上或之下的距离	4	1	9	Yes
zIndex	设置元素的堆叠次序	4	1	9	Yes

Printing 属性

属性	描述	IE	F	O	W3C
orphans	设置段落留到页面底部的最小行数	5M	1	9	Yes
page	设置显示某元素时使用的页面类型	5M	1	9	Yes
pageBreakAfter	设置某元素之后的分页行为	4	1	9	Yes
pageBreakBefore	设置某元素之前的分页行为	4	1	9	Yes
pageBreakInside	设置某元素内部的分页行为	5M	1	9	Yes
size	设置页面的方向和尺寸		1	9	Yes
widows	设置段落必须留到页面顶部的最小行数	5M	1	9	Yes

Scrollbar 属性 (IE-only)

属性	描述	IE	F	O	W3C
scrollbar3dLightColor	设置箭头和滚动条左侧和顶边的颜色	5W	No	No	No
scrollbarArrowColor	设置滚动条上的箭头颜色	5W	No	No	No
scrollbarBaseColor	设置滚动条的底色	5W	No	No	No
scrollbarDarkShadowColor	设置箭头和滚动条右侧和底边的颜色	5W	No	No	No
scrollbarFaceColor	设置滚动条的表色	5W	No	No	No
scrollbarHighlightColor	设置箭头和滚动条左侧和顶边的颜色，以及滚动条的背景	5W	No	No	No
scrollbarShadowColor	设置箭头和滚动条右侧和底边的颜	5W	No	No	No

	色				
scrollbarTrackColor	设置滚动条的背景色	5W	No	No	No

Table 属性

属性	描述	IE	F	O	W3C
borderCollapse	设置表格边框是否合并为单边框，或者像在标准的 HTML 中那样分离。	5	1	9	Yes
borderSpacing	设置分隔单元格边框的距离	5M	1	9	Yes
captionSide	设置表格标题的位置	5M	No	No	Yes
emptyCells	设置是否显示表格中的空单元格	5M	1	9	Yes
tableLayout	设置用来显示表格单元格、行以及列的算法	5	No	No	Yes

Text 属性

属性	描述	IE	F	O	W3C
color	设置文本的颜色	4	1	9	Yes
font	在一行设置所有的字体属性	4	1	9	Yes
fontFamily	设置元素的字体系列。	4	1	9	Yes
fontSize	设置元素的字体大小。	4	1	9	Yes
fontSizeAdjust	设置/调整文本的尺寸	5M	1	No	Yes
fontStretch	设置如何紧缩或伸展字体	5M	No	No	Yes
fontStyle	设置元素的字体样式	4	1	9	Yes
fontVariant	用小型大写字母字体来显示文本	4	1	9	Yes
fontWeight	设置字体的粗细	4	1	9	Yes
letterSpacing	设置字符间距	4	1	9	Yes
lineHeight	设置行间距	4	1	9	Yes
quotes	设置在文本中使用哪种引号	5M	1		Yes
textAlign	排列文本	4	1	9	Yes
textDecoration	设置文本的修饰	4	1	9	Yes

textIndent	缩紧首行的文本	4	1	9	Yes
textShadow	设置文本的阴影效果	5M	1		Yes
textTransform	对文本设置大写效果	4	1	9	Yes
unicodeBidi		5	1		Yes
whiteSpace	设置如何设置文本中的折行和空白符	4	1	9	Yes
wordSpacing	设置文本中的词间距	6	1	9	Yes

标准属性

属性	描述	IE	F	O	W3C
dir	设置或返回文本的方向	5	1	9	Yes
lang	设置或返回元素的语言代码	5	1	9	Yes
title	设置或返回元素的咨询性的标题	5	1	9	Yes

cssText 属性

它是一组样式属性及其值的文本表示。这个文本格式化为一个 CSS 样式表，去掉了包围属性和值的元素选择器的花括号。
将这一属性设置为非法的值将会抛出一个代码为 SYNTAX_ERR 的 DOMException 异常。当 CSS2Properties 对象是只读的时候，试图设置这一属性将会抛出一个代码为 NO_MODIFICATION_ALLOWED_ERR 的 DOMException 异常。

关于 CSS2Properties 对象

CSS2Properties 对象表示一组 CSS 样式属性及其值。它为 CSS 规范定义的每一个 CSS 属性都定义一个 JavaScript 属性。
一个 HTMLElement 的 style 属性是一个可读可写的 CSS2Properties 对象，就好像 CSSRule 对象的 style 属性一样。不过，Window.getComputedStyle() 的返回值是一个 CSS2Properties 对象，其属性是只读的。

HTML DOM background 属性

定义和用法

background 属性在一个声明中设置所有的背景属性。

语法：

```
Object.style.background=background-color background-image
background-repeat background-attachment background-position
```

参数	描述	值
background-color	设置元素的背景色。	• color-name • color-rgb • color-hex • transparent
background-image	设置背景图像。	• url(URL) • none
background-repeat	设置背景图像是否及如何重复。	• repeat • repeat-x • repeat-y • no-repeat
background-attachment	背景图像是否固定或者随着页面的其余部分滚动。	• scroll • fixed
background-position	设置背景图像的起始位置。	• top left • top center • top right • center left • center center • center right • bottom left • bottom center • bottom right • x% y% • xpos ypos

实例

下面的例子改变了文档背景的样式：

```
<html>
<head>
<script type="text/javascript">
function setStyle()
{
document.body.style.background="#FFCC80 url(bgdesert.jpg)
repeat-y";
}
</script>
</head>
<body>

<input type="button" onclick="setStyle()"
value="Set background style" />

</body>
</html>
```

TIY

改变文档背景的样式

```
<html>
<head>
<script type="text/javascript">
function setStyle()
{
document.body.style.background="#FFCC80 url(/i/eg_bg_desert.jpg)
repeat-y";
}
</script>
</head>
<body>

<input type="button" onclick="setStyle()" value="Set background
style" />

</body>
```

```
</html>
```

HTML DOM backgroundAttachment 属性

定义和用法

backgroundAttachment 属性设置背景图像是否固定或者随着页面的其余部分滚动。

语法：

```
Object.style.backgroundAttachment=scroll|fixed
```

提示和注释

提示：请设置一种可用的背景颜色，这样的话，假如背景图像不可用，页面也可获得良好的视觉效果。

实例

本例设置在点击按钮后把背景图像设置为固定（不滚动）：

```
<html>
<head>
<style type="text/css">
body
{
background-color="#FFCC80";
background-image:url(bgdesert.jpg);
}
p
{
color:white;
}
</style>
<script type="text/javascript">
function changeAttachment()
```

```
{
document.body.style.backgroundAttachment="fixed";
}
</script>

</head>
<body>

<input type="button" onclick="changeAttachment()"
value="Set background-image to be fixed" />
<p>This is a paragraph</p>
<p>This is a paragraph</p>
<p>This is a paragraph</p>
<p>This is a paragraph</p>
<p>This is a paragraph</p>
<p>This is a paragraph</p>
<p>This is a paragraph</p>
<p>This is a paragraph</p>
<p>This is a paragraph</p>
<p>This is a paragraph</p>
<p>This is a paragraph</p>
<p>This is a paragraph</p>
<p>This is a paragraph</p>
<p>This is a paragraph</p>

</body>
</html>
```

TIY

把背景图像设置为固定

```
<html>
<head>
<style type="text/css">
body
{
background-color="#FFCC80";
background-image:url(/i/eg_bg_desert.jpg);
}
p
{
color:white;
```

```
}
</style>
<script type="text/javascript">
function changeAttachment()
{
document.body.style.backgroundAttachment="fixed";
}
</script>

</head>
<body>

<input type="button" onclick="changeAttachment()" value="Set
background-image to be fixed" />
<p>This is a paragraph</p>
<p>This is a paragraph</p>
<p>This is a paragraph</p>
<p>This is a paragraph</p>
<p>This is a paragraph</p>
<p>This is a paragraph</p>
<p>This is a paragraph</p>
<p>This is a paragraph</p>
<p>This is a paragraph</p>
<p>This is a paragraph</p>
<p>This is a paragraph</p>
<p>This is a paragraph</p>
<p>This is a paragraph</p>
<p>This is a paragraph</p>

</body>
</html>
```

HTML DOM backgroundColor 属性

定义和用法

backgroundColor 属性设置元素的背景颜色。

语法：

```
Object.style.backgroundColor=color-name|color-rgb
|color-hex|transparent
```

实例

本例设置 body 的背景色：

```
<html>
<head>
<style type="text/css">
body
{
background-color:#B8BFD8;
}
</style>
<script type="text/javascript">
function changeStyle()
{
document.body.style.backgroundColor="#FFCC80";
}
</script>
</head>
<body>

<input type="button" onclick="changeStyle()"
value="Change background color" />

</body>
</html>
```

TIY

改变背景色（十六进制）

```
<html>
<head>
<style type="text/css">
body
{
background-color:#B8BFD8;
```

```
}
</style>
<script type="text/javascript">
function changeStyle()
{
document.body.style.backgroundColor="#FFCC80";
}
</script>
</head>
<body>

<input type="button" onclick="changeStyle()" value="Change
background color" />

</body>
</html>
```

改变背景色（颜色名）

```
<html>
<head>
<style type="text/css">
body
{
background-color:#B8BFD8;
}
</style>
<script type="text/javascript">
function changeStyle()
{
document.body.style.backgroundColor="red";
}
</script>
</head>
<body>

<input type="button" onclick="changeStyle()" value="Change
background color" />

</body>
</html>
```

HTML DOM backgroundImage 属性

定义和用法

backgroundImage 属性设置元素的背景图像。

语法：

```
Object.style.backgroundImage=url(URL)|none
```

参数	描述
url(*URL*)	图像的路径。
none	无背景图像。

提示和注释

提示：请设置一种可用的背景颜色，这样的话，假如背景图像不可用，页面也可获得良好的视觉效果。

实例

本例设置了背景图像：

```
<html>
<head>
<script type="text/javascript">
function changeStyle()
{
document.body.style.backgroundColor="#FFCC80";
document.body.style.backgroundImage="url(bgdesert.jpg)";
}
</script>
</head>
<body>

<input type="button" onclick="changeStyle()"
```

```
value="Set background image" />

</body>
</html>
```

TIY

设置背景图像

```
<html>
<head>
<script type="text/javascript">
function changeStyle()
{
document.body.style.backgroundColor="#FFCC80";
document.body.style.backgroundImage="url(/i/eg_bg_desert.jpg)";
}
</script>
</head>
<body>

<input type="button" onclick="changeStyle()" value="Set
background-image" />

</body>
</html>
```

HTML DOM backgroundPosition 属性

定义和用法

backgroundPosition 属性设置背景图像的位置。

语法：

```
Object.style.backgroundPosition=position
```

参数	描述

• top left • top center • top right • center left • center center • center right • bottom left • bottom center • bottom right	如果您仅规定了一个关键词，那么第二个值将是"center"。 默认值：0% 0%。
x% y%	第一个值是水平位置，第二个值是垂直位置。 左上角是 0% 0%。右下角是 100% 100%。 如果您仅规定了一个值，另一个值将是 50%。
xpos ypos	第一个值是水平位置，第二个值是垂直位置。 左上角是 0 0。单位是像素 (0px 0px) 或任何其他的 CSS 单位。 如果您仅规定了一个值，另一个值将是 50%。 您可以混合使用 % 和 position 值。

实例

本例改变背景图像的位置：

```
<html>
<head>
<style type="text/css">
body
{
background-color:#FFCC80;
background-image:url(bgdesert.jpg);
background-repeat:no-repeat
}
</style>
<script type="text/javascript">
function changePosition()
{
document.body.style.backgroundPosition="bottom center";
}
</script>
</head>
<body>

<input type="button" onclick="changePosition()"
value="Change background-image position" />
```

```
</body>
</html>
```

TIY

改变背景图像的位置

```
<html>
<head>
<style type="text/css">
body
{
background-color:#FFCC80;
background-image:url(/i/eg_bg_desert.jpg);
background-repeat:no-repeat;
background-attachment:fixed;
}
</style>
<script type="text/javascript">
function changePosition()
{
document.body.style.backgroundPosition="bottom center";
}
</script>
</head>
<body>

<input type="button" onclick="changePosition()" value="Change
background-image position" />
<p><b>Note:</b> For this to work in Mozilla, the background-attachment
property must be set to "fixed".</p>
</body>
</html>
```

HTML DOM backgroundPositionX 属性

定义和用法

backgroundPositionX 属性设置背景图像的水平位置。

语法：

```
Object.style.backgroundPositionX=position
```

参数	描述
• left • center • right	水平位置。
x%	水平位置。左是 0%。右是 100%。
xpos	水平位置。左是 0。单位是像素 (0px) 或其他 CSS 单位。

实例

本例改变了背景图像的水平位置：

```
<html>
<head>
<style type="text/css">
body
{
background-color:#FFCC80;
background-image:url(bgdesert.jpg);
background-repeat:no-repeat
}
</style>
<script type="text/javascript">
function changePosition()
{
```

```
document.body.style.backgroundPositionX="right";
}
</script>
</head>
<body>

<input type="button" onclick="changePosition()"
value="Change background-image's x-position" />

</body>
</html>
```

TIY

改变背景图像的水平位置

```
<html>
<head>
<style type="text/css">
body
{
background-color:#FFCC80;
background-image:url(/i/eg_bg_desert.jpg);
background-repeat:no-repeat;
background-attachment:fixed;
}
</style>
<script type="text/javascript">
function changePosition()
{
document.body.style.backgroundPositionX="right";
}
</script>
</head>
<body>

<input type="button" onclick="changePosition()" value="Change
background-image's x-position" />
<p><b>Note:</b> For this to work in Mozilla, the background-attachment
property must be set to "fixed".</p>
</body>
</html>
```

HTML DOM backgroundPositionY 属性

定义和用法

backgroundPositionY 属性设置 background-image 的垂直位置。

语法：

```
Object.style.backgroundPositionY=position
```

参数	描述
• top • center • bottom	垂直位置。
y%	垂直位置。Top 是 0%。Bottom 是 100%
ypos	垂直位置。Top 是 0。单位是像素 (0px) 或其他 CSS 单位。

实例

下面的例子改变 background-image 的垂直位置：

```
<html>
<head>
<style type="text/css">
body
{
background-color:#FFCC80;
background-image:url(bgdesert.jpg);
background-repeat:no-repeat
}
</style>
<script type="text/javascript">
function changePosition()
{
```

```
document.body.style.backgroundPositionY="bottom";
}
</script>
</head>
<body>

<input type="button" onclick="changePosition()"
value="Change background-image's y-position" />

</body>
</html>
```

TIY

改变 background-image 的垂直位置

```
<html>
<head>
<style type="text/css">
body
{
background-color:#FFCC80;
background-image:url(/i/eg_bg_desert.jpg);
background-repeat:no-repeat;
background-attachment:fixed;
}
</style>
<script type="text/javascript">
function changePosition()
{
document.body.style.backgroundPositionY="bottom";
}
</script>
</head>
<body>

<input type="button" onclick="changePosition()" value="Change
background-image's y-position" />
<p><b>Note:</b> For this to work in Mozilla, the background-attachment
property must be set to "fixed".</p>
</body>
</html>
```

HTML DOM backgroundRepeat 属性

定义和用法

backgroundRepeat 属性设置背景图像是否及如何重复。

语法：

```
Object.style.backgroundRepeat=repeat_value
```

参数	描述
repeat	默认。背景图像将在垂直方向和水平方向重复。
repeat-x	背景图像将在水平方向重复。
repeat-y	背景图像将在垂直方向重复。
no-repeat	背景图像将仅显示一次。

实例

本例仅在 y 轴重复背景图像：

```
<html>
<head>
<style type="text/css">
body
{
background-color:#FFCC80;
background-image:url(bgdesert.jpg);
}
</style>
<script type="text/javascript">
function changeRepeat()
{
document.body.style.backgroundRepeat="repeat-y";
}
</script>
</head>
<body>
```

```
<input type="button" onclick="changeRepeat()"
value="Repeat background-image only on the y-axis" />

</body>
</html>
```

TIY

改变 background-image 的 repeat 属性

```
<html>
<head>
<style type="text/css">
body
{
background-color:#FFCC80;
background-image:url(/i/eg_bg_desert.jpg);
}
</style>
<script type="text/javascript">
function changeRepeat()
{
document.body.style.backgroundRepeat="repeat-y";
}
</script>
</head>

<body>

<input type="button" onclick="changeRepeat()" value="Repeat
background-image only on the y-axis" />

</body>
</html>
```

HTML DOM border 属性

定义和用法

border 属性在一个声明中设置所有边框属性。

语法：

```
Object.style.border=borderWidth borderStyle borderColor
```

值	描述	值
borderWidth	设置边框的宽度。	• thin • medium • thick • length
borderStyle	设置边框的样式。	• none • hidden • dotted • dashed • solid • double • groove • ridge • inset • outset
borderColor	设置边框的颜色。	• color-name • color-rgb • color-hex • transparent

实例

本例改变元素的边框：

```
<html>
<head>
<style type="text/css">
```

```
p
{
border: thin dotted #FF0000
}
</style>
<script type="text/javascript">
function changeBorder()
{
document.getElementById("p1").style.border="thick solid #0000FF";
}
</script>
</head>
<body>

<input type="button" onclick="changeBorder()"
value="Change border" />

<p id="p1">This is a paragraph</p>

</body>
</html>
```

TIY

border - 改变元素的边框

```
<html>
<head>
<style type="text/css">
p
{
border: thin dotted #FF0000;
}
</style>
<script type="text/javascript">
function changeBorder()
{
document.getElementById("p1").style.border="thick solid #0000FF";
}
</script>
</head>
<body>
```

```
<input type="button" onclick="changeBorder()" value="Change border"
/>

<p id="p1">This is a paragraph</p>

</body>
</html>
```

HTML DOM borderBottom 属性

定义和用法

borderBottom 属性在一个声明中为下边框设置所有的属性。

语法：

```
Object.style.borderBottom=borderWidth borderStyle borderColor
```

值	描述	值
borderWidth	设置边框的宽度。	• thin • medium • thick • length
borderStyle	设置边框的样式。	• none • hidden • dotted • dashed • solid • double • groove • ridge • inset • outset
borderColor	设置边框的颜色。	• color-name • color-rgb • color-hex • transparent

实例

下面的例子设置下边框：

```
<html>
<head>
<style type="text/css">
p {border: thin dotted #FF0000}
</style>
<script type="text/javascript">
function changeBorder()
{
document.getElementById("p1").style.borderBottom="thick solid
blue";
}
</script>
</head>
<body>

<input type="button" onclick="changeBorder()"
value="Change bottom paragraph border" />
<p id="p1">This is a paragraph with a border</p>

</body>
</html>
```

TIY

borderBottom - 改变元素的下边框

```
<html>
<head>
<style type="text/css">
p
{
border: thin dotted #FF0000;
}
</style>
<script type="text/javascript">
function changeBorder()
{
document.getElementById("p1").style.borderBottom="thick solid
blue";
```

```
}
</script>
</head>
<body>

<input type="button" onclick="changeBorder()" value="Change bottom
border" />
<p id="p1">This is a paragraph with a border</p>

</body>
</html>
```

HTML DOM borderBottomColor 属性

定义和用法

borderBottomColor 属性设置元素的下边框的颜色。

语法：

```
Object.style.borderBottomColor=color-name|color-rgb|color-hex
```

实例

本例改变下边框的颜色：

```
<html>
<head>
<style type="text/css">
p {border: thick solid #FF0000}
</style>
<script type="text/javascript">
function changeBorderColor()
{
document.getElementById("p1").style.borderBottomColor="blue";
}
</script>
</head>
<body>
```

```
<input type="button" onclick="changeBorderColor()"
value="Change border color" />
<p id="p1">This is a paragraph</p>

</body>
</html>
```

TIY

borderBottomColor - 改变下边框的颜色

```
<html>
<head>
<style type="text/css">
p
{
border: thick solid #FF0000;
}
</style>
<script type="text/javascript">
function changeBorderColor()
{
document.getElementById("p1").style.borderBottomColor="blue";
}
</script>
</head>
<body>

<input type="button" onclick="changeBorderColor()" value="Change
border color" />
<p id="p1">This is a paragraph</p>

</body>
</html>
```

HTML DOM borderBottomStyle 属性

定义和用法

borderBottomStyle 属性设置下边框的样式。

语法：

```
Object.style.borderBottomStyle=style
```

值	描述
none	定义无边框。
hidden	与 "none" 相同。不过应用于表时除外，对于表，hidden 用于解决边框冲突。
dotted	定义点状边框。在大多数浏览器中呈现为实线。
dashed	定义虚线。在大多数浏览器中呈现为实线。
solid	定义实线。
double	定义双线。双线的宽度等于 border-width 的值。
groove	定义 3D 凹槽边框。其效果取决于 border-color 的值。
ridge	定义 3D 垄状边框。其效果取决于 border-color 的值。
inset	定义 3D inset 边框。其效果取决于 border-color 的值。
outset	定义 3D outset 边框。其效果取决于 border-color 的值。

实例

本例改变下边框的样式：

```
<html>
<head>
<style type="text/css">
p
{
border: thick solid #FF0000
}
</style>
```

```
<script type="text/javascript">
function changeBorder()
{
document.getElementById("p1").style.borderBottomStyle="dotted";
}
</script>
</head>
<body>

<input type="button" onclick="changeBorder()"
value="Change bottom border" />

<p id="p1">This is a paragraph</p>

</body>
</html>
```

TIY

borderBottomStyle - 改变下边框的样式

```
<html>
<head>
<style type="text/css">
p
{
border: thick solid #FF0000;
}
</style>
<script type="text/javascript">
function changeBorder()
{
document.getElementById("p1").style.borderBottomStyle="dotted";
}
</script>
</head>
<body>

<input type="button" onclick="changeBorder()" value="Change bottom
border" />

<p id="p1">This is a paragraph</p>
```

```
</body>
</html>
```

HTML DOM borderBottomWidth 属性

定义和用法

borderBottomWidth 属性设置下边框的宽度。

语法：

```
Object.style.borderBottomWidth=thin|medium|thick|length
```

值	描述
thin	定义细的下边框。
medium	默认。定义中等的下边框。
thick	定义粗的下边框。
length	允许您自定义下边框的宽度。

实例

本例改变下边框的宽度：

```
<html>
<head>
<style type="text/css">
p
{
border: thin solid #FF0000
}
</style>
<script type="text/javascript">
function changeBorderWidth()
{
```

```
document.getElementById("p1").style.borderBottomWidth="thick";
}
</script>
</head>
<body>

<input type="button" onclick="changeBorderWidth()"
value="Change bottom border" />

<p id="p1">This is a paragraph</p>

</body>
</html>
```

TIY

borderBottomWidth - 改变下边框的宽度

```
<html>
<head>
<style type="text/css">
p
{
border: thin solid #FF0000;
}
</style>
<script type="text/javascript">
function changeBorderWidth()
{
document.getElementById("p1").style.borderBottomWidth="thick";
}
</script>
</head>
<body>

<input type="button" onclick="changeBorderWidth()" value="Change
bottom border" />

<p id="p1">This is a paragraph</p>

</body>
</html>
```

HTML DOM borderColor 属性

定义和用法

borderColor 属性设置元素周围边框的颜色。
本属性可使用 1 到 4 种颜色：

- 如果规定一种颜色，比如 table {border-color: red} - 所有的边框都是红色。
- 如果规定了两种颜色，比如 table {border-color: red green} - 上边框和下边框是红色，而左边框和右边框是绿色。
- 如果规定了三种颜色，比如 table {border-color: red green blue}- 上边框是红色，左边框和右边框是绿色，下边框是蓝色。
- 如果规定了四种颜色，比如 table {border-color: red green blue yellow} - 上边框是红色，右边框是绿色，下边框是蓝色，左边框是黄色。

语法：

```
Object.style.borderColor=color-name|color-rgb|color-hex|transpare
nt
```

实例

本例改变边框的颜色：

```
<html>
<head>
<style type="text/css">
p
{
border: thick solid #FF0000
}
</style>
<script type="text/javascript">
function changeBorderColor()
{
document.getElementById("p1").style.borderColor="#0000FF #00FF00";
}
</script>
</head>
<body>
```

```
<input type="button" onclick="changeBorderColor()"
value="Change border color" />

<p id="p1">This is a paragraph</p>

</body>
</html>
```

TIY

borderColor - 改变边框的颜色

```
<html>
<head>
<style type="text/css">
p
{
border: thick solid #FF0000;
}
</style>
<script type="text/javascript">
function changeBorderColor()
{
document.getElementById("p1").style.borderColor="#0000FF #00FF00";
}
</script>
</head>
<body>

<input type="button" onclick="changeBorderColor()" value="Change
border color" />

<p id="p1">This is a paragraph</p>

</body>
</html>
```

HTML DOM borderLeft 属性

定义和用法

borderLeft 属性在一个声明中为左边框设置所有属性。

语法：

```
Object.style.borderLeft=borderWidth borderStyle borderColor
```

可能的值：

值	描述	值
borderWidth	设置边框的宽度。	• thin • medium • thick • length
borderStyle	设置边框的样式。	• none • hidden • dotted • dashed • solid • double • groove • ridge • inset • outset
borderColor	设置边框的颜色。	• color-name • color-rgb • color-hex • transparent

实例

本例改变左边框的宽度、样式和颜色：

```
<html>
```

```
<head>
<style type="text/css">
p
{
border: thin dotted #FF0000
}
</style>
<script type="text/javascript">
function changeBorder()
{
document.getElementById("p1").style.borderLeft="thick solid
#0000FF";
}
</script>
</head>
<body>

<input type="button" onclick="changeBorder()"
value="Change left border" />

<p id="p1">This is a paragraph</p>

</body>
</html>
```

TIY

borderLeft - 改变左边框的宽度、样式和颜色

```
<html>
<head>
<style type="text/css">
p
{
border: thin dotted #FF0000;
}
</style>
<script type="text/javascript">
function changeBorder()
{
document.getElementById("p1").style.borderLeft="thick solid
#0000FF";
}
```

```
</script>
</head>
<body>

<input type="button" onclick="changeBorder()" value="Change left
border" />

<p id="p1">This is a paragraph</p>

</body>
</html>
```

HTML DOM borderLeftColor 属性

定义和用法

borderLeftColor 属性设置元素的左边框的颜色。

语法：

```
Object.style.borderLeftColor=color-name|color-rgb|color-hex|trans
parent
```

实例

本例改变左边框的颜色：

```
<html>
<head>
<style type="text/css">
p
{
border: thick solid #FF0000
}
</style>
<script type="text/javascript">
function changeBorderColor()
{
document.getElementById("p1").style.borderLeftColor="#0000FF";
```

```
}
</script>
</head>
<body>

<input type="button" onclick="changeBorderColor()"
value="Change left border" />

<p id="p1">This is a paragraph</p>

</body>
</html>
```

TIY

borderleftColor - 改变左边框的颜色

```
<html>
<head>
<style type="text/css">
p
{
border: thick solid #FF0000;
}
</style>
<script type="text/javascript">
function changeBorderColor()
{
document.getElementById("p1").style.borderLeftColor="#0000FF";
}
</script>
</head>
<body>

<input type="button" onclick="changeBorderColor()" value="Change
left border" />

<p id="p1">This is a paragraph</p>

</body>
</html>
```

HTML DOM borderLeftStyle 属性

定义和用法

borderLeftStyle 属性设置左边框的样式。

语法：

```
Object.style.borderLeftStyle=style
```

可能的值：

值	描述
none	定义无边框。
hidden	与 "none" 相同。不过应用于表时除外，对于表，hidden 用于解决边框冲突。
dotted	定义点状边框。在大多数浏览器中呈现为实线。
dashed	定义虚线。在大多数浏览器中呈现为实线。
solid	定义实线。
double	定义双线。双线的宽度等于 border-width 的值。
groove	定义 3D 凹槽边框。其效果取决于 border-color 的值。
ridge	定义 3D 垄状边框。其效果取决于 border-color 的值。
inset	定义 3D inset 边框。其效果取决于 border-color 的值。
outset	定义 3D outset 边框。其效果取决于 border-color 的值。

实例

本例设置左边框的样式：

```
<html>
<head>
<style type="text/css">
p
{
```

```
border: thick solid #FF0000;
}
</style>
<script type="text/javascript">
function changeBorder()
{
document.getElementById("p1").style.borderLeftStyle="dotted";
}
</script>
</head>
<body>

<input type="button" onclick="changeBorder()"
value="Change left border" />

<p id="p1">This is a paragraph</p>

</body>
</html>
```

TIY

borderLeftStyle - 改变左边框的样式

```
<html>
<head>
<style type="text/css">
p
{
border: thick solid #FF0000;
}
</style>
<script type="text/javascript">
function changeBorder()
{
document.getElementById("p1").style.borderLeftStyle="dotted";
}
</script>
</head>
<body>

<input type="button" onclick="changeBorder()" value="Change left
border" />
```

```
<p id="p1">This is a paragraph</p>

</body>
</html>
```

HTML DOM borderLeftWidth 属性

定义和用法

borderLeftWidth 属性设置左边框的宽度。

语法：

```
Object.style.borderLeftWidth=thin|medium|thick|length
```

可能的值：

值	描述
thin	定义细的下边框。
medium	默认。定义中等的下边框。
thick	定义粗的下边框。
length	允许您自定义下边框的宽度。

实例

本例改变左边框的宽度：

```
<html>
<head>
<style type="text/css">
p
{
border: thin solid #FF0000;
}
</style>
```

```
<script type="text/javascript">
function changeBorderWidth()
{
document.getElementById("p1").style.borderLeftWidth="thick";
}
</script>
</head>
<body>

<input type="button" onclick="changeBorderWidth()"
value="Change left border" />

<p id="p1">This is a paragraph</p>

</body>
</html>
```

HTML DOM borderRight 属性

定义和用法

borderRight 属性在一个声明中为右边框设置所有属性。

语法：

```
Object.style.borderRight=borderWidth borderStyle borderColor
```

可能的值：

值	描述	值
borderWidth	设置边框的宽度。	• thin • medium • thick • length
borderStyle	设置边框的样式。	• none • hidden • dotted • dashed

		• solid • double • groove • ridge • inset • outset
borderColor	设置边框的颜色。	• color-name • color-rgb • color-hex • transparent

实例

本例改变右边框的宽度、样式和颜色：

```
<html>
<head>
<style type="text/css">
p
{
border: thin dotted #FF0000
}
</style>
<script type="text/javascript">
function changeBorder()
{
document.getElementById("p1").style.borderRight="thick solid
#0000FF";
}
</script>
</head>
<body>

<input type="button" onclick="changeBorder()"
value="Change right border" />

<p id="p1">This is a paragraph</p>

</body>
</html>
```

TIY

borderRight - 改变右边框的宽度、样式和颜色

```
<html>
<head>
<style type="text/css">
p
{
border: thin dotted #FF0000;
}
</style>
<script type="text/javascript">
function changeBorder()
{
document.getElementById("p1").style.borderRight="thick solid
#0000FF";
}
</script>
</head>
<body>

<input type="button" onclick="changeBorder()" value="Change right
border" />

<p id="p1">This is a paragraph</p>

</body>
</html>
```

HTML DOM borderRightColor 属性

定义和用法

borderRightColor 属性设置右边框的颜色。

语法：

```
Object.style.borderRightColor=color-name|color-rgb|color-hex|tran
```

```
sparent
```

实例

本例改变右边框的颜色：

```
<html>
<head>
<style type="text/css">
p
{
border: thick solid #FF0000;
}
</style>
<script type="text/javascript">
function changeBorderColor()
{
document.getElementById("p1").style.borderRightColor="#0000FF";
}
</script>
</head>
<body>

<input type="button" onclick="changeBorderColor()"
value="Change right border" />

<p id="p1">This is a paragraph</p>

</body>
</html>
```

TIY

borderRightColor - 改变右边框的颜色

```
<html>
<head>
<style type="text/css">
p
{
border: thick solid #FF0000;
}
</style>
```

```
<script type="text/javascript">
function changeBorderColor()
{
document.getElementById("p1").style.borderRightColor="#0000FF";
}
</script>
</head>
<body>

<input type="button" onclick="changeBorderColor()" value="Change
right border" />

<p id="p1">This is a paragraph</p>

</body>
</html>
```

HTML DOM borderRightStyle 属性

定义和用法

borderRightStyle 属性设置右边框的样式。

语法：

```
Object.style.borderRightStyle=style
```

可能的值：

值	描述
none	定义无边框。
hidden	与 "none" 相同。不过应用于表时除外，对于表，hidden 用于解决边框冲突。
dotted	定义点状边框。在大多数浏览器中呈现为实线。
dashed	定义虚线。在大多数浏览器中呈现为实线。
solid	定义实线。
double	定义双线。双线的宽度等于 border-width 的值。

groove	定义 3D 凹槽边框。其效果取决于 border-color 的值。
ridge	定义 3D 垄状边框。其效果取决于 border-color 的值。
inset	定义 3D inset 边框。其效果取决于 border-color 的值。
outset	定义 3D outset 边框。其效果取决于 border-color 的值。

实例

本例改变右边框的样式：

```
<html>
<head>
<style type="text/css">
p
{
border: thick solid #FF0000;
}
</style>
<script type="text/javascript">
function changeBorder()
{
document.getElementById("p1").style.borderRightStyle="dotted";
}
</script>
</head>
<body>

<input type="button" onclick="changeBorder()"
value="Change right border" />

<p id="p1">This is a paragraph</p>

</body>
</html>
```

TIY

borderRightStyle - 改变右边框的样式

```
<html>
<head>
<style type="text/css">
```

```
p
{
border: thick solid #FF0000;
}
</style>
<script type="text/javascript">
function changeBorder()
{
document.getElementById("p1").style.borderRightStyle="dotted";
}
</script>
</head>
<body>

<input type="button" onclick="changeBorder()" value="Change right
border" />

<p id="p1">This is a paragraph</p>

</body>
</html>
```

HTML DOM borderRightWidth 属性

定义和用法

borderRightWidth 属性设置元素的右边框的宽度。

语法：

```
Object.style.borderRightWidth=thin|medium|thick|length
```

Possible Values

值	描述
thin	定义细的下边框。
medium	默认。定义中等的下边框。

thick	定义粗的下边框。
length	允许您自定义下边框的宽度。

实例

本例改变右边框的宽度：

```
<html>
<head>
<style type="text/css">
p
{
border: thin solid #FF0000
}
</style>
<script type="text/javascript">
function changeBorderWidth()
{
document.getElementById("p1").style.borderRightWidth="thick";
}
</script>
</head>
<body>

<input type="button" onclick="changeBorderWidth()"
value="Change right border width" />

<p id="p1">This is a paragraph</p>

</body>
</html>
```

TIY

borderRightWidth - 改变右边框的宽度

```
<html>
<head>
<style type="text/css">
p
{
border: thin solid #FF0000
```

```
}
</style>
<script type="text/javascript">
function changeBorderWidth()
{
document.getElementById("p1").style.borderRightWidth="thick";
}
</script>
</head>
<body>

<input type="button" onclick="changeBorderWidth()" value="Change
right border width" />

<p id="p1">This is a paragraph</p>

</body>
</html>
```

HTML DOM borderStyle 属性

定义和用法

borderStyle 属性在一行声明中为所有四个设置或返回边框样式。
该属性可使用 1 到 4 种样式。

语法：

```
Object.style.borderStyle=style
```

可能的值：

值	描述
none	定义无边框。
hidden	与 "none" 相同。不过应用于表时除外，对于表，hidden 用于解决边框冲突。
dotted	定义点状边框。在大多数浏览器中呈现为实线。
dashed	定义虚线。在大多数浏览器中呈现为实线。

solid	定义实线。
double	定义双线。双线的宽度等于 border-width 的值。
groove	定义 3D 凹槽边框。其效果取决于 border-color 的值。
ridge	定义 3D 垄状边框。其效果取决于 border-color 的值。
inset	定义 3D inset 边框。其效果取决于 border-color 的值。
outset	定义 3D outset 边框。其效果取决于 border-color 的值。

实例

本例改变边框的样式：

```
<html>
<head>
<style>
p
{
border: thick solid #FF0000
}
</style>

<script type="text/javascript">
function changeBorder()
{
document.getElementById("p1").style.borderStyle="dotted double";
}
</script>

</head>
<body>

<form>
<input type="button" onclick="changeBorder()"
value="Change paragraph border style" />
</form>

<p id="p1">This is a paragraph</p>

</body>
</html>
```

TIY

改变边框的样式

```
<html>
<head>
<style>
p
{
border: thick solid #FF0000
}
</style>

<script type="text/javascript">
function changeBorder()
{
document.getElementById("p1").style.borderStyle="dotted double";
}
</script>
</head>
<body>

<form>
<input type="button" onclick="changeBorder()" value="Change
paragraph border style" />
</form>

<p id="p1">This is a paragraph</p>

</body>
</html>
```

HTML DOM borderTop 属性

定义和用法

borderTop 属性在一行声明中为上边框设置所有的属性。

语法：

```
Object.style.borderTop=borderWidth borderStyle borderColor
```

可能的值：

值	描述	值
borderWidth	设置边框的宽度。	• thin • medium • thick • length
borderStyle	设置边框的样式。	• none • hidden • dotted • dashed • solid • double • groove • ridge • inset • outset
borderColor	设置边框的颜色。	• color-name • color-rgb • color-hex • transparent

实例

本例改变上边框的宽度、样式和颜色：

```
<html>
<head>
<style type="text/css">
p
{
border: thin dotted #FF0000
}
</style>
<script type="text/javascript">
function changeBorder()
```

```
{
document.getElementById("p1").style.borderTop="thick solid
#0000FF";
}
</script>
</head>
<body>

<input type="button" onclick="changeBorder()"
value="Change top border" />

<p id="p1">This is a paragraph</p>

</body>
</html>
```

TIY

borderTop - 改变上边框的宽度、样式和颜色

```
<html>
<head>
<style type="text/css">
p
{
border: thin dotted #FF0000
}
</style>
<script type="text/javascript">
function changeBorder()
{
document.getElementById("p1").style.borderTop="thick solid
#0000FF";
}
</script>
</head>
<body>

<input type="button" onclick="changeBorder()" value="Change top
border" />

<p id="p1">This is a paragraph</p>
```

```
</body>
</html>
```

HTML DOM borderTopColor 属性

定义和用法

borderTopColor 属性设置元素上边框的颜色。

语法：

```
Object.style.borderTopColor=color-name|color-RGB|color-hex
```

实例

本例改变段落的上边框的颜色：

```
<html>
<head>
<style type="text/css">
p
{
border: thick solid #FF0000
}
</style>
<script type="text/javascript">
function changeBorderColor()
{
document.getElementById("p1").style.borderTopColor="#0000FF";
}
</script>
</head>
<body>

<input type="button" onclick="changeBorderColor()"
value="Change paragraph border color" />

<p id="p1">This is a paragraph</p>
```

```
</body>
</html>
```

TIY

borderTopColor - 改变段落的上边框的颜色

```
<html>
<head>
<style type="text/css">
p
{
border: thick solid #FF0000
}
</style>
<script type="text/javascript">
function changeBorderColor()
{
document.getElementById("p1").style.borderTopColor="#0000FF";
}
</script>
</head>
<body>

<input type="button" onclick="changeBorderColor()" value="Change top
border color" />

<p id="p1">This is a paragraph</p>

</body>
</html>
```

HTML DOM borderTopStyle 属性

定义和用法

borderTopStyle 属性设置元素的上边框的样式。

语法：

```
Object.style.borderTopStyle=style
```

可能的值：

值	描述
none	定义无边框。
hidden	与 "none" 相同。不过应用于表时除外，对于表，hidden 用于解决边框冲突。
dotted	定义点状边框。在大多数浏览器中呈现为实线。
dashed	定义虚线。在大多数浏览器中呈现为实线。
solid	定义实线。
double	定义双线。双线的宽度等于 border-width 的值。
groove	定义 3D 凹槽边框。其效果取决于 border-color 的值。
ridge	定义 3D 垄状边框。其效果取决于 border-color 的值。
inset	定义 3D inset 边框。其效果取决于 border-color 的值。
outset	定义 3D outset 边框。其效果取决于 border-color 的值。

实例

本例更改段落的上边框的样式：

```
<html>
<head>
<style type="text/css">
p
{
border: thick solid #FF0000
}
</style>
<script type="text/javascript">
function changeBorder()
{
document.getElementById("p1").style.borderTopStyle="dotted";
}
</script>
```

```
</head>
<body>

<input type="button" onclick="changeBorder()"
value="Change border style" />

<p id="p1">This is a paragraph</p>

</body>
</html>
```

TIY

borderTopStyle - 改变上边框的样式

```
<html>
<head>
<style type="text/css">
p
{
border: thick solid #FF0000
}
</style>
<script type="text/javascript">
function changeBorder()
{
document.getElementById("p1").style.borderTopStyle="dotted";
}
</script>
</head>
<body>

<input type="button" onclick="changeBorder()" value="Change top
border style" />

<p id="p1">This is a paragraph</p>

</body>
</html>
```

HTML DOM borderTopWidth 属性

定义和用法

borderTopWidth 属性设置元素的上边框的宽度。

语法：

```
Object.style.borderTopWidth=thin|medium|thick|length
```

可能的值：

值	描述
thin	定义细的下边框。
medium	默认。定义中等的下边框。
thick	定义粗的下边框。
length	允许您自定义下边框的宽度。

实例

下面的例子改变了上边框的宽度：

```
<html>
<head>
<style type="text/css">
p
{
border: thin solid #FF0000
}
</style>
<script type="text/javascript">
function changeBorderWidth()
{
document.getElementById("p1").style.borderTopWidth="thick";
}
</script>
```

```
</head>
<body>

<input type="button" onclick="changeBorderWidth()"
value="Change border width" />

<p id="p1">This is a paragraph</p>

</body>
</html>
```

TIY

borderTopWidth - 改变上边框的宽度

```
<html>
<head>
<style type="text/css">
p
{
border: thin solid #FF0000
}
</style>
<script type="text/javascript">
function changeBorderWidth()
{
document.getElementById("p1").style.borderTopWidth="thick";
}
</script>
</head>
<body>

<input type="button" onclick="changeBorderWidth()" value="Change top
border width" />

<p id="p1">This is a paragraph</p>

</body>
</html>
```

HTML DOM borderWidth 属性

定义和用法

borderWidth 属性设置所有四个边框的宽度。
该属性可使用 1 到 4 个值：

- 如果规定一个值，比如 div {border-width: thick} - 所有四个边框都是粗线。
- 如果规定两个值，比如 div {border-width: thick thin} - 上下边框是粗线，而左右边框是细线。
- 如果规定三个值，比如 div {border-width: thick thin medium}- 上边框是粗线，左右边框是细线，下边框是中等 (medium) 的宽度。
- 如果规定四个值，比如 div {border-width: thick thin medium 10px} - 上边框是粗线，右边框是细线，下边框是中等的宽度，而左边框是 10px 的宽度。

语法：

```
Object.style.borderWidth=thin|medium|thick|length
```

可能的值：

值	描述
thin	定义细的下边框。
medium	默认。定义中等的下边框。
thick	定义粗的下边框。
length	允许您自定义下边框的宽度。

实例

本例改变边框的宽度：

```
<html>
<head>
<style type="text/css">
p
{
border: thin solid #FF0000
```

```
}
</style>
<script type="text/javascript">
function changeBorderWidth()
{
document.getElementById("p1").style.borderWidth="thick thin";
}
</script>
</head>
<body>

<input type="button" onclick="changeBorderWidth()"
value="Change border widths" />

<p id="p1">This is a paragraph</p>

</body>
</html>
```

TIY

borderWidth - 改变所有四个边框的宽度

```
<html>
<head>
<style type="text/css">
p
{
border: thin solid #FF0000
}
</style>
<script type="text/javascript">
function changeBorderWidth()
{
document.getElementById("p1").style.borderWidth="thick thin";
}
</script>
</head>
<body>

<input type="button" onclick="changeBorderWidth()" value="Change
border widths" />
```

```
<p id="p1">This is a paragraph</p>

</body>
</html>
```

HTML DOM margin 属性

定义和用法

margin 属性设置元素的外边距。
该属性可使用 1 到 4 个值：

- 如果规定一个值，比如 div {margin: 50px} - 所有的外边距都是 50 px
- 如果规定两个值，比如 div {margin: 50px 10px} - 上下外边距是 50px，左右外边距是 10 px。
- 如果规定三个值，比如 div {margin: 50px 10px 20px}- 上外边距是 50 px，而左右外边距是 10 px，下外边距是 20 px。
- 如果规定四个值，比如 div {margin: 50px 10px 20px 30px} - 上外边距是 50 px，右外边距是 10 px，下外边距是 20 px，左外边距是 30 px。

语法：

```
Object.style.margin=margin
```

可能的值：

值	描述
margin	设置外边距。 值可以是： • 百分比（基于父对象总高度或宽度的百分比） • 长度值（定义一个固定的边距） • auto（浏览器设定的值）。

实例

下面的例子改变了段落的外边距：

```
<html>
<head>
```

```
<script type="text/javascript">
function changeMargin()
{
document.getElementById("p1").style.margin="100px";
}
</script>
</head>
<body>

<input type="button" onclick="changeMargin()"
value="Change margins of a paragraph" />

<p id="p1">This is a paragraph</p>

</body>
</html>
```

TIY

margin - 改变段落的外边距

```
<html>
<head>
<script type="text/javascript">
function changeMargin()
{
document.getElementById("p1").style.margin="100px";
}
</script>
</head>
<body>

<input type="button" onclick="changeMargin()" value="Change margins
of a paragraph" />

<p id="p1">This is a paragraph</p>

</body>
</html>
```

HTML DOM marginBottom 属性

定义和用法

marginBottom 属性设置元素的下外边距。

语法：

```
Object.style.marginBottom=auto|length|%
```

可能的值：

值	描述
auto	浏览器设置的一个下外边距。
length	定义一个固定的下外边距。默认值是 0。
%	定义基于父对象总高度的百分比下外边距。

实例

本例改变段落的下外边距：

```
<html>
<head>
<script type="text/javascript">
function changeMargin()
{
document.getElementById("p1").style.marginBottom="32px";
}
</script>
</head>
<body>

<input type="button" onclick="changeMargin()"
value="Change the bottom margin of a paragraph" />

<p>This is a paragraph</p>
<p id="p1">This is a paragraph</p>
```

```
<p>This is a paragraph</p>

</body>
</html>
```

TIY

marginBottom - 改变元素的下外边距

```
<html>
<head>
<script type="text/javascript">
function changeMargin()
{
document.getElementById("p1").style.marginBottom="32px";
}
</script>
</head>
<body>

<input type="button" onclick="changeMargin()" value="Change the
bottom margin of a paragraph" />

<p>This is a paragraph</p>
<p id="p1">This is a paragraph</p>
<p>This is a paragraph</p>

</body>
</html>
```

HTML DOM marginLeft 属性

定义和用法

marginLeft 属性设置元素的左外边距。

语法：

```
Object.style.marginLeft=auto|length|%
```

可能的值：

值	描述
auto	浏览器设置的左外边距。
length	定义固定的左外边距。默认值是 0。
%	定义基于父对象总高度的百分比左外边距。

实例

本例更改了段落的左外边距：

```
<html>
<head>
<script type="text/javascript">
function changeMargin()
{
document.getElementById("p1").style.marginLeft="32px";
}
</script>
</head>
<body>

<input type="button" onclick="changeMargin()"
value="Change the left margin of a paragraph" />

<p>This is a paragraph</p>
<p id="p1">This is a paragraph</p>
<p>This is a paragraph</p>

</body>
</html>
```

TIY

marginLeft - 改变元素的左外边距

```
<html>
<head>
<script type="text/javascript">
function changeMargin()
```

```
{
document.getElementById("p1").style.marginLeft="32px";
}
</script>
</head>
<body>

<input type="button" onclick="changeMargin()" value="Change the left
margin of a paragraph" />

<p>This is a paragraph</p>
<p id="p1">This is a paragraph</p>
<p>This is a paragraph</p>

</body>
</html>
```

HTML DOM marginRight 属性

定义和用法

marginRight 属性设置元素的右外边距。

语法：

```
Object.style.marginRight=auto|length|%
```

可能的值

值	描述
auto	浏览器设置的外右边距。
length	定义固定的外右边距。默认值是 0。
%	定义基于父对象总高度的百分比外右边距。

实例

下面的例子改变了段落的右外边距：

```
<html>
<head>
<style type="text/css">
p
{
text-align: right
}
</style>
<script type="text/javascript">
function changeMargin()
{
document.getElementById("p1").style.marginRight="32px";
}
</script>
</head>
<body>

<input type="button" onclick="changeMargin()"
value="Change the right margin of a paragraph" />

<p>This is a paragraph</p>
<p id="p1">This is a paragraph</p>
<p>This is a paragraph</p>

</body>
</html>
```

TIY

marginRight - 改变段落的右外边距

```
<html>
<head>
<style type="text/css">
p
{
text-align: right;
}
</style>
<script type="text/javascript">
function changeMargin()
{
```

```
document.getElementById("p1").style.marginRight="32px";
}
</script>
</head>
<body>

<input type="button" onclick="changeMargin()" value="Change the
right margin of a paragraph" />

<p>This is a paragraph</p>
<p id="p1">This is a paragraph</p>
<p>This is a paragraph</p>

</body>
</html>
```

HTML DOM marginTop 属性

定义和用法

marginTop 属性设置元素的上外边距。

语法：

```
Object.style.marginTop=auto|length|%
```

可能的值

值	描述
auto	浏览器设置的上外边距。
length	定义固定的上外边距。默认值是 0。
%	定义基于父对象总高度的百分比上外边距。

实例

本例改变段落的上外边距：

```
<html>
<head>
<script type="text/javascript">
function changeMargin()
{
document.getElementById("p1").style.marginTop="32px";
}
</script>
</head>
<body>

<input type="button" onclick="changeMargin()"
value="Change the top margin of a paragraph" />

<p>This is a paragraph</p>
<p id="p1">This is a paragraph</p>
<p>This is a paragraph</p>

</body>
</html>
```

TIY

marginTop - 改变段落的上外边距

```
<html>
<head>
<script type="text/javascript">
function changeMargin()
{
document.getElementById("p1").style.marginTop="32px";
}
</script>
</head>
<body>

<input type="button" onclick="changeMargin()" value="Change the top
margin of a paragraph" />

<p>This is a paragraph</p>
<p id="p1">This is a paragraph</p>
<p>This is a paragraph</p>
```

```
</body>
</html>
```

HTML DOM outline 属性

定义和用法

outline 属性在一个声明中设置所有轮廓属性。

语法：

```
Object.style.outline = outlineWidth outlineStyle outlineColor
```

可能的值

值	描述	值
outlineWidth	设置轮廓的宽度。	• thin • medium • thick • length
outlineStyle	设置轮廓的样式。	• none • hidden • dotted • dashed • solid • double • groove • ridge • inset • outset
outlineColor	设置轮廓的颜色。	• color-name • color-rgb • color-hex • transparent

实例

下面的例子改变段落周围的轮廓：

```
<html>
<head>
<style type="text/css">
p
{
border: thin solid #00FF00;
outline: thick solid #FF0000;
}
</style>
<script type="text/javascript">
function changeOutline()
{
document.getElementById("p1").style.outline="thin dotted #0000FF";
}
</script>
</head>
<body>

<input type="button" onclick="changeOutline()"
value="Change outline" />

<p id="p1">This is a paragraph</p>

</body>
</html>
```

TIY

outline - 改变元素周围的轮廓（请在非 IE 浏览器中浏览）

```
<html>
<head>
<style type="text/css">
p
{
border: thin solid #00FF00;
outline: thick solid #FF0000;
}
</style>
```

```
<script type="text/javascript">
function changeOutline()
{
document.getElementById("p1").style.outline="thin dotted #0000FF";
}
</script>
</head>
<body>

<input type="button" onclick="changeOutline()" value="Change
outline" />

<p id="p1">This is a paragraph</p>

</body>
</html>
```

HTML DOM outlineColor 属性

定义和用法

outlineColor 属性设置元素周围的轮廓的颜色。

语法：

```
Object.style.outlineColor=color-name|color-rgb|color-hex
```

实例

本例改变轮廓的颜色：

```
<html>
<head>
<style type="text/css">
p
{
border: thin solid #00FF00;
outline: thick solid #FF0000;
}
```

```
</style>
<script type="text/javascript">
function changeOutlineColor()
{
document.getElementById("p1").style.outlineColor="#00FF00";
}
</script>
</head>
<body>

<input type="button" onclick="changeOutlineColor()"
value="Change outline color" />

<p id="p1">This is a paragraph</p>

</body>
</html>
```

TIY

outlineColor - 改变围绕元素的轮廓的颜色（请在非 IE 浏览器中浏览）

```
<html>
<head>
<style type="text/css">
p
{
border: thin solid #00FF00;
outline: thick solid #FF0000;
}
</style>
<script type="text/javascript">
function changeOutlineColor()
{
document.getElementById("p1").style.outlineColor="#00FF00";
}
</script>
</head>
<body>

<input type="button" onclick="changeOutlineColor()" value="Change
outline color" />
```

```
<p id="p1">This is a paragraph</p>

</body>
</html>
```

HTML DOM outlineStyle 属性

定义和用法

outlineStyle 属性设置围绕元素的轮廓的样式。

语法：

```
Object.style.outlineStyle=style
```

可能的值

值	描述
none	默认。定义无轮廓。
dotted	定义一个点状的轮廓。
dashed	定义一个虚线轮廓。
solid	定义一个实线轮廓。
double	定义一个双线轮廓。双线的宽度等同于 outline-width 的值。
groove	定义一个 3D 凹槽轮廓。此效果取决于 outline-color 值。
ridge	定义一个 3D 凸槽轮廓。此效果取决于 outline-color 值。
inset	定义一个 3D 凹边轮廓。此效果取决于 outline-color 值。
outset	定义一个 3D 凸边轮廓。此效果取决于 outline-color 值。

实例

下面的例子改变轮廓的样式：

```
<html>
```

```
<head>
<style type="text/css">
p
{
border: thin solid #00FF00;
outline: thick solid #FF0000;
}
</style>
<script type="text/javascript">
function changeOutline()
{
document.getElementById("p1").style.outlineStyle="dotted";
}
</script>
</head>
<body>

<input type="button" onclick="changeOutline()"
value="Change outline style" />

<p id="p1">This is a paragraph</p>

</body>
</html>
```

TIY

outlineStyle - 改变围绕元素的轮廓的样式（请在非 IE 浏览器中浏览）

```
<html>
<head>
<style type="text/css">
p
{
border: thin solid #00FF00;
outline: thick solid #FF0000;
}
</style>
<script type="text/javascript">
function changeOutline()
{
document.getElementById("p1").style.outlineStyle="dotted";
}
```

```
</script>
</head>
<body>

<input type="button" onclick="changeOutline()" value="Change outline
style" />

<p id="p1">This is a paragraph</p>

</body>
</html>
```

HTML DOM outlineWidth 属性

定义和用法

outlineWidth 属性设置围绕元素的轮廓的宽度。

语法：

```
Object.style.outlineWidth=thin|medium|thick|length
```

可能的值

值	描述
thin	定义细轮廓。
medium	默认。定义中等的轮廓。
thick	定义粗的轮廓。
length	允许您定义轮廓粗细的值。

实例

本例改变轮廓的宽度：

```
<html>
<head>
```

```
<style type="text/css">
p
{
border: thin solid #00FF00;
outline: thick solid #FF0000;
}
</style>
<script type="text/javascript">
function changeOutlineWidth()
{
document.getElementById("p1").style.outlineWidth="thin";
}
</script>
</head>
<body>

<input type="button" onclick="changeOutlineWidth()"
value="Change outline width" />

<p id="p1">This is a paragraph</p>

</body>
</html>
```

TIY

outlineWidth - 改变围绕元素的轮廓的宽度（请在非 IE 浏览器中浏览）

```
<html>
<head>
<style type="text/css">
p
{
border: thin solid #00FF00;
outline: thick solid #FF0000;
}
</style>
<script type="text/javascript">
function changeOutlineWidth()
{
document.getElementById("p1").style.outlineWidth="thin";
}
</script>
```

```
</head>
<body>

<input type="button" onclick="changeOutlineWidth()" value="Change
outline width" />

<p id="p1">This is a paragraph</p>

</body>
</html>
```

HTML DOM padding 属性

定义和用法

padding 属性设置元素的内边距。
padding 属性定义元素边框与元素内容之间的空间。
该属性可采取 4 个值：

- 如果规定一个值，比如 div {padding: 50px} - 所有四个边的 padding 都是 50 px。
- 如果规定两个值，比如 div {padding: 50px 10px} - 上下内边距是 50 px，左右内边距是 10 px。
- 如果规定三个值，比如 div {padding: 50px 10px 20px} - 上内边距是 50 px，左右内边距是 10 px，下内边距是 20 px。
- 如果规定四个值，比如 div {padding: 50px 10px 20px 30px} - 上内边距是 50 px，右内边距是 10 px，下内边距是 20 px，左内边距是 30 px。

语法：

```
Object.style.padding=padding
```

Possible Values

值	描述
padding	设置内边距。 值可以是： • 百分比（基于父对象总高度或宽度的百分比） • 长度值（定义一个固定的边距） • auto（浏览器设定的值）。

实例

本例改变元素的内边距：

```
<html>
<head>
<style type="text/css">
p
{
border: thin dotted #FF0000;
}
</style>
<script type="text/javascript">
function changePadding()
{
document.getElementById("p1").style.padding="2cm";
}
</script>
</head>
<body>

<input type="button" onclick="changePadding()"
value="Change padding" />

<p id="p1">This is a paragraph</p>

</body>
</html>
```

TIY

padding - 改变元素的内边距

```
<html>
<head>
<style type="text/css">
p
{
border: thin dotted #FF0000;
}
</style>
<script type="text/javascript">
function changePadding()
```

```
{
document.getElementById("p1").style.padding="2cm";
}
</script>
</head>
<body>

<input type="button" onclick="changePadding()" value="Change
padding" />

<p id="p1">This is a paragraph</p>

</body>
</html>
```

HTML DOM paddingBottom 属性

定义和用法

paddingBottom 属性设置元素的下内边距。
padding 属性定义元素边框与元素内容之间的空间。

语法：

```
Object.style.paddingBottom=length|%
```

可能的值

值	描述
length	定义固定的下内边距值。默认值是 0。
%	定义基于父元素高度的百分比下内边距。此值不会如预期的那样工作于所有的浏览器中。

实例

下面的例子改变元素的下内边距：

```
<html>
<head>
<style type="text/css">
p
{
border: thin dotted #FF0000;
}
</style>
<script type="text/javascript">
function changePadding()
{
document.getElementById("p1").style.paddingBottom="2cm";
}
</script>
</head>
<body>

<input type="button" onclick="changePadding()"
value="Change bottom padding" />

<p id="p1">This is a paragraph</p>

</body>
</html>
```

TIY

paddingBottom - 改变元素的下内边距

```
<html>
<head>
<style type="text/css">
p
{
border: thin dotted #FF0000;
}
</style>
<script type="text/javascript">
function changePadding()
{
document.getElementById("p1").style.paddingBottom="2cm";
}
</script>
```

```
</head>
<body>

<input type="button" onclick="changePadding()" value="Change bottom
padding" />

<p id="p1">This is a paragraph</p>

</body>
</html>
```

HTML DOM paddingLeft 属性

定义和用法

paddingLeft 属性设置元素的左内边距。
padding 属性定义元素边框与元素内容之间的空间。

语法：

```
Object.style.paddingLeft=length|%
```

可能的值

值	描述
length	定义固定的下内边距值。默认值是 0。
%	定义基于父元素高度的百分比下内边距。此值不会如预期的那样工作于所有的浏览器中。

实例

本例改变元素的左内边距：

```
<html>
<head>
<style type="text/css">
p
```

```
{
border: thin dotted #FF0000;
}
</style>
<script type="text/javascript">
function changePadding()
{
document.getElementById("p1").style.paddingLeft="2cm";
}
</script>
</head>
<body>

<input type="button" onclick="changePadding()"
value="Change left padding" />

<p id="p1">This is a paragraph</p>

</body>
</html>
```

TIY

paddingLeft - 改变元素的左内边距

```
<html>
<head>
<style type="text/css">
p
{
border: thin dotted #FF0000;
}
</style>
<script type="text/javascript">
function changePadding()
{
document.getElementById("p1").style.paddingLeft="2cm";
}
</script>
</head>
<body>

<input type="button" onclick="changePadding()" value="Change left
```

```
padding" />

<p id="p1">This is a paragraph</p>

</body>
</html>
```

HTML DOM paddingRight 属性

定义和用法

paddingRight 属性设置元素的右内边距。
padding 属性定义元素边框与元素内容之间的空间。

语法：

```
Object.style.paddingRight=auto|length|%
```

可能的值

值	描述
length	定义固定的下内边距值。默认值是 0。
%	定义基于父元素高度的百分比下内边距。此值不会如预期的那样工作于所有的浏览器中。

实例

本例改变元素的右内边距：

```
<html>
<head>
<style type="text/css">
p
{
border: thin dotted #FF0000;
}
</style>
```

```
<script type="text/javascript">
function changePadding()
{
document.getElementById("p1").style.paddingRight="2cm";
}
</script>
</head>
<body>

<input type="button" onclick="changePadding()"
value="Change right padding" />

<p id="p1">This is a paragraph. This is a paragraph.
This is a paragraph. This is a paragraph.
This is a paragraph. This is a paragraph.</p>

</body>
</html>
```

TIY

paddingRight - 改变元素的右内边距

```
<html>
<head>
<style type="text/css">
p
{
border: thin dotted #FF0000;
}
</style>
<script type="text/javascript">
function changePadding()
{
document.getElementById("p1").style.paddingRight="2cm";
}
</script>
</head>
<body>

<input type="button" onclick="changePadding()" value="Change right
padding" />
```

```
<p id="p1">This is a paragraph. This is a paragraph.
This is a paragraph. This is a paragraph.
This is a paragraph. This is a paragraph.</p>

</body>
</html>
```

HTML DOM paddingTop 属性

定义和用法

padding 属性设置元素的上内边距。
padding 属性定义元素边框与元素内容之间的空间。

语法：

```
Object.style.paddingTop=length|%
```

可能的值

值	描述
length	定义固定的下内边距值。默认值是 0。
%	定义基于父元素高度的百分比下内边距。此值不会如预期的那样工作于所有的浏览器中。

实例

本例改变元素的上内边距：

```
<html>
<head>
<style type="text/css">
p
{
border: thin dotted #FF0000;
}
</style>
```

```
<script type="text/javascript">
function changePadding()
{
document.getElementById("p1").style.paddingTop="2cm";
}
</script>
</head>
<body>

<input type="button" onclick="changePadding()"
value="Change top padding" />

<p id="p1">This is a paragraph</p>

</body>
</html>
```

TIY

paddingTop - 改变元素的上内边距

```
<html>
<head>
<style type="text/css">
p
{
border: thin dotted #FF0000;
}
</style>
<script type="text/javascript">
function changePadding()
{
document.getElementById("p1").style.paddingTop="2cm";
}
</script>
</head>
<body>

<input type="button" onclick="changePadding()" value="Change top
padding" />

<p id="p1">This is a paragraph</p>
```

```
</body>
</html>
```

HTML DOM clear 属性

定义和用法

clear 属性设置一个元素的侧面是否允许其他的浮动元素。

语法：

```
Object.style.clear=left|right|both|none
```

可能的值

值	描述
left	在左侧不允许浮动元素
right	在右侧不允许浮动元素
both	在左右两侧均不允许浮动元素
none	默认。允许浮动元素出现在两侧。

实例

本例不允许文本左边的浮动元素：

```
<html>
<head>
<style type="text/css">
img
{
float:left;
}
</style>
<script type="text/javascript">
function clearLeft()
{
```

```
document.getElementById("p1").style.clear="left";
}
</script>
</head>
<body>

<img src="logocss.gif" width="95" height="84" />

<p id="p1">This is some text. This is some text. This is some text.
This is some text. This is some text. This is some text.
This is some text. This is some text. This is some text.</p>

<input type="button" onclick="clearLeft()"
value="Clear left side of text" />

</body>
</html>
```

TIY

清除文本左边的浮动元素

```
<html>
<head>
<style type="text/css">
img
{
float:left;
}
</style>
<script type="text/javascript">
function clearLeft()
{
document.getElementById("p1").style.clear="left";
}
</script>
</head>
<body>

<img src="/i/ct_beatles.jpg" />

<p id="p1">This is some text. This is some text. This is some text.
This is some text. This is some text. This is some text.
```

```
This is some text. This is some text. This is some text.</p>

<input type="button" onclick="clearLeft()" value="Clear left side of
text" />

</body>
</html>
```

HTML DOM clip 属性

定义和用法

clip 属性设置元素的形状。
当图像大于它所在的元素时会发生什么？"clip" 属性允许你规定元素的可见尺寸，以及元素被裁剪和显示的形状。

语法：

```
Object.style.clip=rect(top,right,bottom,left)|auto
```

可能的值

值	描述
rect(*top,right,bottom,left*)	设置元素的形状。
auto	浏览器设置元素的形状。

提示和注释

注释：该属性不能用于 "overflow" 设置为 "visible" 的元素。

实例

本例把图像裁剪为指定的形状：

```
<html>
<head>
```

```
<style type="text/css">
img
{
position:absolute;
top:100px;
}
</style>
<script type="text/javascript">
function clipImage()
{
document.getElementById("img1").style.clip="rect(0px,50px,50px,0p
x)";
}
</script>
</head>
<body>

<img id="img1" border="0" src="logocss.gif" width="95" height="84"
/>

<input type="button" onclick=clipImage() value="Clip image" />

</body>
</html>
```

TIY

把图像裁剪为指定的形状

```
<html>
<head>
<style type="text/css">
img
{
position:absolute;
top:100px;
}
</style>
<script type="text/javascript">
function clipImage()
{
document.getElementById("img1").style.clip="rect(0px,50px,50px,0p
x)";
```

```
}
</script>
</head>
<body>

<img id="img1" border="0" src="/i/eg_sun.gif" />

<input type="button" onclick=clipImage() value="Clip image" />

</body>
</html>
```

HTML DOM content 属性

定义和用法

content 属性设置文本或图像出现（浮动）在另一个元素中的什么地方。

语法：

```
Object.style.content=value
```

可能的值

值	描述
string	定义文本内容。
url	定义 url。
• counter(*name*) • counter(*name, list-style-type*) • counters(*name, string*) • counters(*name, string, list-style-type*)	
attr(X)	定义显示在该选择器之前或之后的选择器的属性。
open-quote	
close-quote	

no-open-quote	
no-close-quote	

提示和注释

注释：如果在一行中对于浮动元素而言空间太少，则这个元素会跳到下一行，这个过程会持续到有足够空间的一行为止。
注释：行内元素的内容、背景和边框应该出现在浮动元素之前。块级元素的背景和边框应当出现在浮动元素之后，但是块级元素的内容应当在浮动元素的前面。

实例

本例设置图像浮动于左边：

```
<html>
<head>
<script type="text/javascript">
function setFloatLeft()
{
document.getElementById("img1").style.cssFloat="left";
}
</script>

</head>
<body>

<img id="img1" src="logocss.gif" width="95" height="84" />

<p>This is some text. This is some text. This is some text.
This is some text. This is some text. This is some text.
This is some text. This is some text. This is some text.</p>

<form>
<input type="button" onclick="setFloatLeft()" value="Set image to
float left" />
</form>

</body>
</html>
```

TIY

设置图像浮动于文字的左边

```
<html>
<head>
<script type="text/javascript">
function setFloat()
{
document.getElementById("img1").style.cssFloat="left";
}
</script>
</head>
<body>

<img id="img1" src="/i/ct_beatles.jpg" />

<p>This is some text. This is some text. This is some text.
This is some text. This is some text. This is some text.
This is some text. This is some text. This is some text.</p>

<input type="button" onclick="setFloat()" value="Set image to float
to the left" />

</body>
</html>
```

HTML DOM cssFloat 属性

定义和用法

设置文本或图像出现（浮动）在另一个元素中的什么地方。

语法：

```
Object.style.cssFloat=left|right|none
```

可能的值

值	描述
left	图像或文本浮动在父元素的左边。
right	图像或文本浮动在父元素的右边。
none	图像或文本浮动显示在它在父元素中出现的位置。

提示和注释

注释：如果在一行中对于浮动元素而言空间太少，则这个元素会跳到下一行，这个过程会持续到有足够空间的一行为止。

实例

本例设置图像浮动于左边：

```
<html>
<head>
<script type="text/javascript">
function setFloat()
{
document.getElementById("img1").style.cssFloat="left";
}
</script>
</head>
<body>

<img id="img1" src="logocss.gif" />

<p>This is some text. This is some text. This is some text.
This is some text. This is some text. This is some text.
This is some text. This is some text. This is some text.</p>

<input type="button" onclick="setFloat()"
value="Set image to float to the left" />

</body>
</html>
```

TIY

cssFloat - 设置图像浮动于文字的左边（请在非 IE 浏览器中查看）

```
<html>
<head>
<script type="text/javascript">
function setFloat()
{
document.getElementById("img1").style.cssFloat="left";
}
</script>
</head>
<body>

<img id="img1" src="/i/ct_beatles.jpg" />

<p>This is some text. This is some text. This is some text.
This is some text. This is some text. This is some text.
This is some text. This is some text. This is some text.</p>

<input type="button" onclick="setFloat()" value="Set image to float
to the left" />

</body>
</html>
```

HTML DOM cursor 属性

定义和用法

cursor 属性规定所显示的指针（光标）的类型。

语法：

```
Object.style.cursor=cursortype
```

可能的值

值	描述
url	需被使用的自定义光标的 URL 注释：请在此列表的末端始终定义一种普通的光标，以防没有由 URL 定义的可用光标。
default	默认光标（通常是一个箭头）
auto	默认。浏览器设置的光标。
crosshair	光标呈现为十字线。
pointer	光标呈现为指示链接的指针（一只手）
move	此光标指示某对象可被移动。
e-resize	此光标指示矩形框的边缘可被向右（东）移动。
ne-resize	此光标指示矩形框的边缘可被向上及向右移动（北/东）。
nw-resize	此光标指示矩形框的边缘可被向上及向左移动（北/西）。
n-resize	此光标指示矩形框的边缘可被向上（北）移动。
se-resize	此光标指示矩形框的边缘可被向下及向右移动（南/东）。
sw-resize	此光标指示矩形框的边缘可被向下及向左移动（南/西）。
s-resize	此光标指示矩形框的边缘可被向下移动（北/西）。
w-resize	此光标指示矩形框的边缘可被向左移动（西）。
text	此光标指示文本。
wait	此光标指示程序正忙（通常是一只表或沙漏）。
help	此光标指示可用的帮助（通常是一个问号或一个气球）。

实例

本例改变指针：

```
<html>
<head>
<script type="text/javascript">
function changeCursor()
{
document.body.style.cursor="crosshair";
```

```
document.getElementById("p1").style.cursor="text";
}
</script>
</head>
<body>

<p id="p1">This is some text. This is some text.
This is some text. This is some text. This is some text.
This is some text. This is some text. This is some text.
This is some text.</p>

<input type="button" onclick="changeCursor()"
value="Change cursor" />

</body>
</html>
```

TIY

cursor - 改变指针

```
<html>
<head>
<script type="text/javascript">
function changeCursor()
{
document.body.style.cursor="crosshair";
document.getElementById("p1").style.cursor="text";
}
</script>
</head>
<body>

<p id="p1">This is some text. This is some text. This is some text.
This is some text. This is some text. This is some text.
This is some text. This is some text. This is some text.</p>

<input type="button" onclick="changeCursor()" value="Change cursor"
/>

</body>
</html>
```

HTML DOM direction 属性

定义和用法

direction 属性设置元素的文本方向。

语法：

```
Object.style.direction=ltr|rtl|inherit
```

Possible Values

值	描述
ltr	默认。内容从左向右流动。
rtl	内容从右向左流动。
inherit	内容流动从父元素继承。

实例

本例把文本设置为从右向左流动：

```
<html>
<head>
<script type="text/javascript">
function changeTextDirection()
{
document.getElementById("p1").style.direction="rtl";
}
</script>
</head>
<body>

<p id="p1">This is some text. This is some text. This is some text.
This is some text. This is some text. This is some text.
This is some text. This is some text. This is some text.</p>

<input type="button" onclick="changeTextDirection()"
```

```
value="Set text to flow right to left" />

</body>
</html>
```

TIY

direction - 把文本设置为从右向左流动

```
<html>
<head>
<script type="text/javascript">
function changeTextDirection()
{
document.getElementById("p1").style.direction="rtl";
}
</script>
</head>
<body>

<p id="p1">This is some text. This is some text. This is some text.
This is some text. This is some text. This is some text.
This is some text. This is some text. This is some text.</p>

<input type="button" onclick="changeTextDirection()" value="Set text
to flow from right to left" />

</body>
</html>
```

HTML DOM display 属性

定义和用法

display 属性设置元素如何显示。

语法：

```
Object.style.display=value
```

Possible Values

值	描述
none	此元素不会被显示。
block	此元素将显示为块级元素，此元素前后会带有换行符。
inline	默认。此元素会被显示为内联元素，元素前后没有换行符。
list-item	此元素会作为列表显示。
run-in	此元素会根据上下文作为块级元素或内联元素显示。
compact	此元素会根据上下文作为块级元素或内联元素显示。
marker	
table	此元素会作为块级表格来显示（类似 <table>），表格前后带有换行符。
inline-table	此元素会作为内联表格来显示（类似 <table>），表格前后没有换行符。
table-row-group	此元素会作为一个或多个行的分组来显示（类似 <tbody>）。
table-header-group	此元素会作为一个或多个行的分组来显示（类似 <thead>）。
table-footer-group	此元素会作为一个或多个行的分组来显示（类似 <tfoot>）。
table-row	此元素会作为一个表格行显示（类似 <tr>）。
table-column-group	此元素会作为一个或多个列的分组来显示（类似 <colgroup>）。
table-column	此元素会作为一个单元格列显示（类似 <col>）
table-cell	此元素会作为一个表格单元格显示（类似 <td> 和 <th>）
table-caption	此元素会作为一个表格标题显示（类似 <caption>）

实例

本例设置不显示元素：

```
<html>
<head>
<script type="text/javascript">
function removeElement()
{
document.getElementById("p1").style.display="none";
}
</script>
```

```
</head>
<body>

<h1>Hello</h1>

<p id="p1">This is some text. This is some text. This is some text.
This is some text. This is some text. This is some text.
This is some text. This is some text. This is some text.</p>

<input type="button" onclick="removeElement()"
value="Do not display paragraph" />

</body>
</html>
```

TIY

display - 设置不显示元素

```
<html>
<head>
<script type="text/javascript">
function removeElement()
{
document.getElementById("p1").style.display="none";
}
</script>
</head>
<body>

<h1>This is a header</h1>

<p id="p1">This is some text. This is some text. This is some text.
This is some text. This is some text. This is some text.
This is some text. This is some text. This is some text.</p>

<input type="button" onclick="removeElement()" value="Do not display
paragraph" />

</body>
</html>
```

HTML DOM height 属性

定义和用法

height 属性设置元素的高度。

语法：

```
Object.style.height=auto|length|%
```

可能的值

值	描述
auto	默认。浏览器会计算出实际的高度。
length	使用 px、cm 等单位定义高度。
%	基于其包含块的百分比高度。

实例

本例设置按钮的高度：

```
<html>
<head>
<script type="text/javascript">
function setHeight()
{
document.getElementById("b1").style.height="50px";
}
</script>
</head>
<body>

<input type="button" id="b1" onclick="setHeight()"
value="Change height of button to 50 px" />

</body>
</html>
```

TIY

height - 设置元素的高度

```
<html>
<head>
<script type="text/javascript">
function setHeight()
{
document.getElementById("b1").style.height="50px";
}
</script>
</head>
<body>

<input type="button" id="b1" onclick="setHeight()" value="Change
height of button to 50 px" />

</body>
</html>
```

HTML DOM maxHeight 属性

定义和用法

maxHeight 属性设置元素的最大高度。

语法：

```
Object.style.maxHeight=length|%
```

可能的值

值	描述
length	定义元素的最大高度值。
%	定义基于其包含块的百分比最大高度。

实例

本例设置元素框的最大高度：

```
<html>
<head>
<script type="text/javascript">
function setMaxHeight()
{
document.getElementById("p1").style.maxHeight="10px";
}
</script>
</head>
<body>

<p id="p1">This is some text. This is some text. This is some text.
This is some text. This is some text. This is some text.
This is some text. This is some text. This is some text.</p>

<input type="button" onclick="setMaxHeight()"
value="Set max height" />

</body>
</html>
```

TIY

maxHeight - 设置元素框的最大高度

```
<html>
<head>
<script type="text/javascript">
function setMaxHeight()
{
document.getElementById("p1").style.maxHeight="10px";
}
</script>
</head>
<body>

<p id="p1">This is some text. This is some text. This is some text.
This is some text. This is some text. This is some text.
This is some text. This is some text. This is some text.</p>
```

```
<input type="button" onclick="setMaxHeight()" value="Set max height"
/>

</body>
</html>
```

HTML DOM maxWidth 属性

定义和用法

maxWidth 属性设置元素的最大宽度。

语法：

```
Object.style.maxWidth=length|%
```

可能的值

值	描述
length	定义元素的最大宽度值。
%	定义基于其包含块的百分比最大宽度。

实例

本例设置元素框的最大宽度：

```
<html>
<head>
<script type="text/javascript">
function setMaxWidth()
{
document.getElementById("p1").style.maxWidth="10px";
}
</script>
</head>
<body>
```

```
<p id="p1">This is some text. This is some text. This is some text.
This is some text. This is some text. This is some text.
This is some text. This is some text. This is some text.</p>

<input type="button" onclick="setMaxWidth()"
value="Set max width" />

</body>
</html>
```

TIY

maxWidth - 设置元素框的最大宽度

```
<html>
<head>
<script type="text/javascript">
function setMaxWidth()
{
document.getElementById("p1").style.maxWidth="10px";
}
</script>
</head>
<body>

<p id="p1">This is some text. This is some text. This is some text.
This is some text. This is some text. This is some text.
This is some text. This is some text. This is some text.</p>

<input type="button" onclick="setMaxWidth()" value="Set max width"
/>

</body>
</html>
```

HTML DOM minHeight 属性

定义和用法

minHeight 属性设置元素的最小高度。

语法：

```
Object.style.minHeight=length|%
```

可能的值

值	描述
length	定义元素的最小高度值。
%	定义基于其包含块的百分比最小高度。

实例

本例设置元素框的最小高度：

```
<html>
<head>
<script type="text/javascript">
function setMinHeight()
{
document.getElementById("p1").style.minHeight="200px";
}
</script>
</head>
<body>

<p id="p1">This is some text. This is some text. This is some text.
This is some text. This is some text. This is some text.
This is some text. This is some text. This is some text.</p>

<input type="button" onclick="setMinHeight()"
value="Set min height" />
```

```
</body>
</html>
```

TIY

minHeight - 设置元素框的最小高度

```
<html>
<head>
<script type="text/javascript">
function setMinHeight()
{
document.getElementById("p1").style.minHeight="200px";
}
</script>
</head>
<body>

<p id="p1">This is some text. This is some text. This is some text.
This is some text. This is some text. This is some text.
This is some text. This is some text. This is some text.</p>

<input type="button" onclick="setMinHeight()" value="Set min height"
/>

</body>
</html>
```

HTML DOM minWidth 属性

定义和用法

minWidth 属性设置元素框的最小宽度。

语法：

```
Object.style.minWidth=length|%
```

可能的值

值	描述
length	定义元素的最小宽度值。
%	定义基于其包含块的百分比最小宽度。

实例

本例设置元素框的最小宽度：

```
<html>
<head>
<script type="text/javascript">
function setMinWidth()
{
document.getElementById("p1").style.minWidth="1000px";
}
</script>
</head>
<body>

<p id="p1">This is some text. This is some text. This is some text.
This is some text. This is some text. This is some text.
This is some text. This is some text. This is some text.</p>

<input type="button" onclick="setMinWidth()" value="Set min width"
/>

</body>
</html>
```

TIY

minWidth - 设置元素框的最小宽度

```
<html>
<head>
<script type="text/javascript">
function setMinWidth()
{
```

```
document.getElementById("p1").style.minWidth="1000px";
}
</script>
</head>
<body>

<p id="p1">This is some text. This is some text. This is some text.
This is some text. This is some text. This is some text.
This is some text. This is some text. This is some text.</p>

<input type="button" onclick="setMinWidth()" value="Set min width"
/>

</body>
</html>
```

HTML DOM overflow 属性

定义和用法

overflow 属性规定如何处理如何处理不符合元素框的内容。

语法：

```
Object.style.overflow=visible|hidden|scroll|auto
```

可能的值

值	描述
visible	内容不会被修剪，会呈现在元素框之外。
hidden	内容会被修剪，但是浏览器不会显示供查看内容的滚动条。
scroll	内容会被修剪，但是浏览器会显示滚动条以便查看其余的内容。
auto	由浏览器决定如何显示。如果需要，则显示滚动条。

实例

本例使用 overflow 来显示溢出元素框的内容：

```
<html>
<head>
<style type="text/css">
div
{
border:thin solid green;
width:100px;
height:100px;
}
</style>
<script type="text/javascript">
function hideOverflow()
{
document.getElementById("div1").style.overflow="hidden";
}
</script>
</head>
<body>

<div id="div1">
This is some text. This is some text. This is some text.
This is some text. This is some text. This is some text.
This is some text. This is some text. This is some text.
</div>
<br />
<input type="button" onclick="hideOverflow()"
value="Hide overflow" />

</body>
</html>
```

TIY

overflow - 隐藏内容溢出

```
<html>
<head>
<style type="text/css">
div
```

```
{
border:thin solid green;
width:100px;
height:100px;
}
</style>
<script type="text/javascript">
function hideOverflow()
{
document.getElementById("div1").style.overflow="hidden";
}
</script>
</head>
<body>

<div id="div1">
This is some text. This is some text. This is some text.
This is some text. This is some text. This is some text.
This is some text. This is some text. This is some text.
</div>
<br />
<input type="button" onclick="hideOverflow()" value="Hide overflow"
/>

</body>
</html>
```

HTML DOM verticalAlign 属性

定义和用法

verticalAlign 属性设置内容在元素框中的垂直对齐方式。

语法：

```
Object.style.verticalAlign=value
```

可能的值

值	描述
baseline	默认。元素放置在父元素的基线上。
sub	垂直对齐文本的下标。
super	垂直对齐文本的上标
top	把元素的顶端与行中最高元素的顶端对齐
text-top	把元素的顶端与父元素字体的顶端对齐
middle	把此元素放置在父元素的中部。
bottom	把元素的顶端与行中最低的元素的顶端对齐。
text-bottom	把元素的底端与父元素字体的底端对齐。
length	
%	使用 "line-height" 属性的百分比值来排列此元素。允许使用负值。

实例

本例设置表格中文本的垂直对齐方式：

```
<html>
<head>
<script type="text/javascript">
function alignText()
{
document.getElementById("td1").style.verticalAlign="bottom";
}
</script>
</head>
<body>

<table border="1" height="100px">
  <tr>
    <td id="td1">
    Some example text
    </td>
  </tr>
</table>
<br />
```

```
<input type="button" onclick="alignText()"
value="Align text" />

</body>
</html>
```

TIY

verticalAlign - 垂直对齐表格中的文本

```
<html>
<head>
<script type="text/javascript">
function alignText()
{
document.getElementById("td1").style.verticalAlign="bottom";
}
</script>
</head>
<body>

<table border="1" height="100px">
  <tr>
    <td id="td1">
    Some example text
    </td>
  </tr>
</table>
<br />
<input type="button" onclick="alignText()" value="Align text" />

</body>
</html>
```

HTML DOM visibility 属性

定义和用法

visibility 属性设置元素是否可见。

语法：

```
Object.style.visibility=visible|hidden|collapse
```

可能的值

值	描述
visible	默认。元素框是可见的。
hidden	元素框不可见，但仍然影响布局。
collapse	当在表格元素中使用时，此值可删除一行或一列，但是它不会影响表格的布局。被行或列占据的空间会留给其他内容使用。如果此值被用在其他的元素上，会呈现为 "hidden"。

实例

下面的例子隐藏一段文本：

```
<html>
<head>
<script type="text/javascript">
function changeVisibility()
{
document.getElementById("p1").style.visibility="hidden";
}
</script>

</head>
<body>

<p id="p1">This is some text. This is some text. This is some text.
This is some text. This is some text. This is some text.
This is some text. This is some text. This is some text.</p>

<form>
<input type="button" onclick="changeVisibility()"
value="Hide paragraph" />
</form>

</body>
</html>
```

TIY

visibility - 隐藏一段文本

```
<html>
<head>
<script type="text/javascript">
function changeVisibility()
{
document.getElementById("p1").style.visibility="hidden";
}
</script>
</head>
<body>

<p id="p1">This is some text. This is some text. This is some text.
This is some text. This is some text. This is some text.
This is some text. This is some text. This is some text.</p>

<input type="button" onclick="changeVisibility()" value="Hide text"
/>

</body>
</html>
```

HTML DOM width 属性

定义和用法

width 属性设置元素的宽度。

语法：

```
Object.style.width=auto|length|%
```

可能的值

值	描述

auto	默认。浏览器可计算出实际的宽度。
%	定义基于其包含块的百分比宽度。
length	使用 px、cm 等单位定义宽度。

实例

本例设置按钮的宽度：

```
<html>
<head>
<script type="text/javascript">
function setWidth()
{
document.getElementById("b1").style.width="300px";
}
</script>
</head>
<body>

<input type="button" id="b1" onclick="setWidth()"
value="Change width to 300 px" />

</body>
</html>
```

TIY

width - 设置元素的宽度

```
<html>
<head>
<script type="text/javascript">
function setWidth()
{
document.getElementById("b1").style.width="300px";
}
</script>
</head>
<body>

<input type="button" id="b1" onclick="setWidth()" value="Change
```

```
width to 300 px" />

</body>
</html>
```

HTML DOM listStyle 属性

定义和用法

listStyle 属性在一个声明中设置所有列表属性。

语法：

```
Object.style.listStyle=listStyleType listStylePosition
listStyleImage
```

Possible Values

值	描述	值
listStyleType	设置列表项标志的类型。 注释：并非右列中的所有值在所有浏览器中都有效。	• none • disc • circle • square • decimal • decimal-leading-zero • lower-roman • upper-roman • lower-alpha • upper-alpha • lower-greek • lower-latin • upper-latin • hebrew • armenian • georgian • cjk-ideographic • hiragana • katakana

		• hiragana-iroha • katakana-iroha
listStylePosition	对列表项标志进行定位。	• outside • inside
listStyleImage	把图像设置为列表项标志。	• none • url(url)

实例

本例改变列表的类型：

```
<html>
<head>
<script type="text/javascript">
function changeList()
  {
  document.getElementById("ul1").style.listStyle="decimal inside";
  }
</script>
</head>
<body>

<ul id="ul1">
  <li>Coffee</li>
  <li>Tea</li>
  <li>Water</li>
  <li>Soda</li>
</ul>

<input type="button" onclick="changeList()"
value="Change list style" />

</body>
</html>
```

TIY

listStyle - 改变列表的类型

```
<html>
<head>
```

```
<script type="text/javascript">
function changeList()
{
document.getElementById("ul1").style.listStyle="decimal inside";
}
</script>
</head>
<body>

<ul id="ul1">
  <li>Coffee</li>
  <li>Tea</li>
  <li>Water</li>
  <li>Soda</li>
</ul>

<input type="button" onclick="changeList()" value="Change list
style" />

</body>
</html>
```

HTML DOM listStyleImage 属性

定义和用法

listStyleImage 属性把图像设置为列表项标志。

语法：

```
Object.style.listStyleImage=url|none
```

可能的值

值	描述
url	图形的路径。
none	不会显示图像。

实例

下面的例子改变列表的列表项标志：

```
<html>
<head>
<script type="text/javascript">
function changeList()
{
document.getElementById("ul1").style.listStyleImage="url('blueb.g
if')";
}
</script>
</head>
<body>

<ul id="ul1">
  <li>Coffee</li>
  <li>Tea</li>
  <li>Water</li>
  <li>Soda</li>
</ul>

<input type="button" onclick="changeList()"
value="Change list-item marker" />

</body>
</html>
```

TIY

listStyleImage - 改变列表的列表项标志

```
<html>
<head>
<script type="text/javascript">
function changeList()
{
document.getElementById("ul1").style.listStyleImage="url('/i/ct_b
lueball.gif')";
}
</script>
</head>
```

```
<body>

<ul id="ul1">
  <li>Coffee</li>
  <li>Tea</li>
  <li>Water</li>
  <li>Soda</li>
</ul>

<input type="button" onclick="changeList()" value="Change list-item
marker" />

</body>
</html>
```

HTML DOM listStylePosition 属性

定义和用法

listStylePosition 属性对列表中的列表项标记进行定位。

语法：

```
Object.style.listStylePosition=inside|outside
```

可能的值

值	描述
inside	列表项目标记放置在文本以内，且环绕文本根据标记对齐。
outside	默认。保持标记位于文本的左侧。列表项目标记放置在文本以外，且环绕文本不根据标记对齐。

实例

本例改变列表项标志的位置：

```
<html>
```

```
<head>
<script type="text/javascript">
function changeList()
{
document.getElementById("ul1").style.listStylePosition="inside";
}
</script>
</head>
<body>

<ul id="ul1">
  <li>Coffee</li>
  <li>Tea</li>
  <li>Water</li>
  <li>Soda</li>
</ul>

<input type="button" onclick="changeList()"
value="Change list-item marker position" />

</body>
</html>
```

TIY

listStylePosition - 改变列表项标志的位置

```
<html>
<head>
<script type="text/javascript">
function changeList()
{
document.getElementById("ul1").style.listStylePosition="inside";
}
</script>
</head>
<body>

<ul id="ul1">
  <li>Coffee</li>
  <li>Tea</li>
  <li>Water</li>
  <li>Soda</li>
```

```
</ul>

<input type="button" onclick="changeList()" value="Change list-item
marker position" />

</body>
</html>
```

HTML DOM listStyleType 属性

定义和用法

listStyleType 属性设置列表项标志的类型。

语法：

```
Object.style.listStyleType=value
```

可能的值

CSS2 的值：

值	描述
none	无标记。
disc	默认。标记是实心圆。
circle	标记是空心圆。
square	标记是实心方块。
decimal	标记是数字。
decimal-leading-zero	0 开头的数字标记。(01, 02, 03, 等。)
lower-roman	小写罗马数字(i, ii, iii, iv, v, 等。)
upper-roman	大写罗马数字(I, II, III, IV, V, 等。)
lower-alpha	小写英文字母 The marker is lower-alpha (a, b, c, d, e, 等。)
upper-alpha	大写英文字母 The marker is upper-alpha (A, B, C, D, E, 等。)

lower-greek	小写希腊字母(alpha, beta, gamma, 等。)
lower-latin	小写拉丁字母(a, b, c, d, e, 等。)
upper-latin	大写拉丁字母(A, B, C, D, E, 等。)
hebrew	传统的希伯来编号方式
armenian	传统的亚美尼亚编号方式
georgian	传统的乔治亚编号方式(an, ban, gan, 等。)
cjk-ideographic	简单的表意数字
hiragana	标记是：a, i, u, e, o, ka, ki, 等。（日文片假名）
katakana	标记是：A, I, U, E, O, KA, KI, 等。（日文片假名）
hiragana-iroha	标记是：i, ro, ha, ni, ho, he, to, 等。（日文片假名）
katakana-iroha	标记是：I, RO, HA, NI, HO, HE, TO, 等。（日文片假名）

CSS2.1 的值：

```
disc | circle | square | decimal | decimal-leading-zero |
lower-roman | upper-roman | lower-greek | lower-latin | upper-latin
| armenian | georgian | none | inherit
```

实例

本例改变列表项的类型：

```
<html>
<head>
<script type="text/javascript">
function changeList()
{
document.getElementById("ul1").style.listStyleType="square";
}
</script>
</head>
<body>

<ul id="ul1">
  <li>Coffee</li>
  <li>Tea</li>
  <li>Water</li>
  <li>Soda</li>
```

```
</ul>

<input type="button" onclick="changeList()"
value="Change list-item marker type" />

</body>
</html>
```

TIY

listStyleType - 改变列表项的类型

```
<html>
<head>
<script type="text/javascript">
function changeList()
{
document.getElementById("ul1").style.listStyleType="square";
}
</script>
</head>
<body>

<ul id="ul1">
  <li>Coffee</li>
  <li>Tea</li>
  <li>Water</li>
  <li>Soda</li>
</ul>

<input type="button" onclick="changeList()" value="Change list-item
marker type" />

</body>
</html>
```

HTML DOM bottom 属性

定义和用法

bottom 属性设置定位元素下外边距边界与其包含块下边界之间的偏移。

语法：

```
Object.style.bottom=auto|%|length
```

可能的值

值	描述
auto	默认。通过浏览器计算底部的位置。
%	设置元素的底边到最近一个具有定位设置父元素的底部边缘的百分比位置。
length	使用 px、cm 等单位设置元素的底边到最近一个具有定位设置父元素的底部边缘的位置。可使用负值。

提示和注释

Note:如果 "position" 属性的值为 "static"，那么设置 "bottom" 属性不会产生任何效果。

实例

本例对元素的下边进行设置：

```
<html>
<head>
<style type="text/css">
input
{
position:absolute;
}
</style>
<script type="text/javascript">
function setBottomEdge()
```

```
{
document.getElementById("b1").style.bottom="100px";
}
</script>
</head>
<body>

<input type="button" id="b1" onclick="setBottomEdge()"
value="Set bottom edge to 100 px" />

</body>
</html>
```

TIY

bottom - 对元素的下边进行设置

```
<html>
<head>
<style type="text/css">
input
{
position:absolute;
}
</style>
<script type="text/javascript">
function setBottomEdge()
{
document.getElementById("b1").style.bottom="100px";
}
</script>

</head>
<body>

<input type="button" id="b1" onclick="setBottomEdge()" value="Set
bottom edge to 100 px" />

</body>
</html>
```

HTML DOM left 属性

定义和用法

left 属性设置定位元素左外边距边界与其包含块左边界之间的偏移。

语法：

```
Object.style.left=auto|%|length
```

可能的值

值	描述
auto	默认。通过浏览器来计算左侧的位置。
%	设置元素的左边到最近一个具有定位设置父元素的左边缘的百分比位置。
length	使用 px、cm 等单位设置元素的左边到最近一个具有定位设置父元素的左边缘的位置。可使用负值。

提示和注释

注释：如果 "position" 属性的值为 "static"，那么设置 "left" 属性不会产生任何效果。

实例

下面的例子对按钮的左边进行设置：

```
<html>
<head>
<style type="text/css">
input
{
position:absolute;
}
</style>
<script type="text/javascript">
function setLeftEdge()
```

```
{
document.getElementById("b1").style.left="100px";
}
</script>
</head>
<body>

<input type="button" id="b1" onclick="setLeftEdge()"
value="Set left edge to 100 px" />

</body>
</html>
```

TIY

left - 对元素的左边进行设置

```
<html>
<head>
<style type="text/css">
input
{
position:absolute;
}
</style>
<script type="text/javascript">
function setLeftEdge()
{
document.getElementById("b1").style.left="100px";
}
</script>
</head>
<body>

<input type="button" id="b1" onclick="setLeftEdge()" value="Set left
edge to 100 px" />

</body>
</html>
```

HTML DOM position 属性

定义和用法

position 属性把元素放置到一个静态的、相对的、绝对的、或固定的位置中。

语法：

```
Object.style.position=static|relative|absolute|fixed
```

可能的值

值	描述
static	默认。位置设置为 static 的元素，它始终会处于页面流给予的位置（static 元素会忽略任何 top、bottom、left 或 right 声明）。
relative	位置被设置为 relative 的元素，可将其移至相对于其正常位置的地方，因此 "left:20" 会将元素移至元素正常位置左边 20 个像素的位置。
absolute	位置设置为 absolute 的元素，可定位于相对于包含它的元素的指定坐标。此元素的位置可通过 "left"、"top"、"right" 以及 "bottom" 属性来规定。
fixed	位置被设置为 fixed 的元素，可定位于相对于浏览器窗口的指定坐标。此元素的位置可通过 "left"、"top"、"right" 以及"bottom" 属性来规定。不论窗口滚动与否，元素都会留在那个位置。工作于 IE7（strict 模式）。

实例

本例把元素位置由相对改为绝对：

```
<html>
<head>
<style type="text/css">
input
{
position:relative;
}
</style>
<script type="text/javascript">
```

```
function setPositionAbsolute()
{
document.getElementById("b1").style.position="absolute";
document.getElementById("b1").style.top="10px";
}
</script>
</head>
<body>

<p>This is an example paragraph</p>
<p>This is an example paragraph</p>

<input type="button" id="b1" onclick="setPositionAbsolute()"
value="Set button position to be absolute" />

</body>
</html>
```

TIY

position - 改变元素的位置

```
<html>
<head>
<style type="text/css">
input
{
position:relative;
}
</style>
<script type="text/javascript">
function setPositionAbsolute()
{
document.getElementById("b1").style.position="absolute";
document.getElementById("b1").style.top="10px";
}
</script>
</head>
<body>

<p>This is an example paragraph</p>
<p>This is an example paragraph</p>
<p>This is an example paragraph</p>
```

```
<p>This is an example paragraph</p>
<p>This is an example paragraph</p>
<p>This is an example paragraph</p>
<p>This is an example paragraph</p>
<p>This is an example paragraph</p>
<p>This is an example paragraph</p>
<p>This is an example paragraph</p>
<p>This is an example paragraph</p>
<p>This is an example paragraph</p>
<p>This is an example paragraph</p>
<p>This is an example paragraph</p>

<input type="button" id="b1" onclick="setPositionAbsolute()"
value="Set button position to be absolute" />

</body>
</html>
```

HTML DOM right 属性

定义和用法

right 属性设置定位元素右外边距边界与其包含块右边界之间的偏移。

语法：

```
Object.style.right=auto|%|length
```

可能的值

值	描述
auto	默认。通过浏览器来计算右侧的位置。
%	设置元素的右边到最近一个具有定位设置父元素的右边缘的百分比位置。
length	使用 px、cm 等单位设置元素的右边到最近一个具有定位设置父元素的右边缘的位置。可使用负值。

提示和注释

注释：如果 "position" 属性的值为 "static"，那么设置 "right" 属性不会产生任何效果。

实例

本例对按钮的左边进行设置：

```
<html>
<head>
<style type="text/css">
input
{
position:absolute;
}
</style>
<script type="text/javascript">
function setRightEdge()
{
document.getElementById("b1").style.right="100px";
}
</script>
</head>
<body>

<input type="button" id="b1" onclick="setRightEdge()"
value="Set right edge to 100 px" />

</body>
</html>
```

TIY

right - 设置元素的右边界

```
<html>
<head>
<style type="text/css">
input
{
position:absolute;
}
```

```
</style>
<script type="text/javascript">
function setRightEdge()
{
document.getElementById("b1").style.right="100px";
}
</script>
</head>
<body>

<input type="button" id="b1" onclick="setRightEdge()" value="Set
right edge to 100 px" />

</body>
</html>
```

HTML DOM top 属性

定义和用法

top 属性设置一个定位元素的上外边距边界与其包含块上边界之间的偏移。

语法：

```
Object.style.top=auto|%|length
```

可能的值

值	描述
auto	默认。通过浏览器来计算顶部的位置。
%	设置元素的顶部到最近一个具有定位设置父元素的上边缘的百分比位置。
length	使用 px、cm 等单位设置元素的顶部到最近一个具有定位设置上边缘的顶部的位置。可使用负值。

提示和注释

注释：如果 "position" 属性的值为 "static"，那么设置 "top" 属性不会产生任何效果。

实例

本例设置元素的上边界：

```
<html>
<head>
<style type="text/css">
input
{
position:absolute;
}
</style>
<script type="text/javascript">
function setTopEdge()
{
document.getElementById("b1").style.top="100px";
}
</script>
</head>
<body>

<input type="button" id="b1" onclick="setTopEdge()"
value="Set top edge to 100 px" />

</body>
</html>
```

TIY

top - 设置元素的上边界

```
<html>
<head>
<style type="text/css">
input
{
position:absolute;
}
</style>
<script type="text/javascript">
function setTopEdge()
{
document.getElementById("b1").style.top="100px";
```

```
}
</script>
</head>
<body>

<input type="button" id="b1" onclick="setTopEdge()" value="Set top edge to 100 px" />

</body>
</html>
```

HTML DOM zIndex 属性

定义和用法

zIndex 属性设置元素的堆叠顺序。
该属性设置一个定位元素沿 z 轴的位置，z 轴定义为垂直延伸到显示区的轴。如果为正数，则离用户更近，为负数则表示离用户更远。

语法：

```
Object.style.zIndex=auto|number
```

可能的值

值	描述
auto	默认。堆叠顺序与父元素相等。
number	设置元素的堆叠顺序。

提示和注释

注释：元素可拥有负的 z-index 属性值。
注释：Z-index 仅能在定位元素上奏效（例如 position:absolute;）!

实例

本例改变元素的堆叠顺序：

```
<html>
<head>
<style type="text/css">
#img1
{
position:absolute;
left:0px;
top:0px;
z-index:-1
}
</style>
<script type="text/javascript">
function changeStackOrder()
{
document.getElementById("img1").style.zIndex="1";
}
</script>
</head>
<body>

<h1>This is a Heading</h1>

<img id="img1" src="bulbon.gif" width="100" height="180">

<p>Default z-index is 0. Z-index -1 has lower priority.</p>

<input type="button" onclick="changeStackOrder()"
value="Change stack order" />

</body>
</html>
```

TIY

zIndex - 改变元素的堆叠顺序

```
<html>
<head>
<style type="text/css">
```

```
#img1
{
position:absolute;
left:0px;
top:0px;
z-index:-1
}
</style>
<script type="text/javascript">
function changeStackOrder()
{
document.getElementById("img1").style.zIndex="1";
}
</script>
</head>
<body>

<h1>This is a Heading</h1>

<img id="img1" src="/i/eg_bulbon.gif">

<p>Default z-index is 0. Z-index -1 has lower priority.</p>

<input type="button" onclick="changeStackOrder()" value="Change
stack order" />

</body>
</html>
```

HTML DOM pageBreakAfter 属性

定义和用法

pageBreakAfter 属性设置元素后否应当放置分页符。

语法：

```
Object.style.pageBreakAfter=auto|always|avoid|left|right
```

可能的值

值	描述
auto	默认。如果必要则在元素后插入分页符。
always	在元素后插入分页符。
avoid	避免在元素后插入分页符。
left	在元素之后足够的分页符，一直到一张空白的左页为止。
right	在元素之后足够的分页符，一直到一张空白的右页为止。

实例

本例在 id 为 p1 的段落之后设置了一个 page-break：

```
<html>
<head>
<script type="text/javascript">
function setPageBreak()
{
document.getElementById("p1").style.pageBreakAfter="always";
}
</script>
</head>
<body>

<p id="p1">This is a test paragraph.</p>

<input type="button" onclick="setPageBreak()"
value="Set page-break" />

<p>This is also a test paragraph.</p>

</body>
</html>
```

HTML DOM pageBreakBefore 属性

定义和用法

pageBreakBefore 属性声明一个元素前是否应当放置分页符。

语法：

```
Object.style.pageBreakBefore=auto|always|avoid|left|right
```

可能的值

值	描述
auto	默认。如果必要则在元素前插入分页符。
always	在元素前插入分页符。
avoid	避免在元素前插入分页符。
left	在元素之前足够的分页符，一直到一张空白的左页为止。
right	在元素之前足够的分页符，一直到一张空白的右页为止。

实例

本例在 id 为 p2 的段落之前设置一个 page-break：

```
<html>
<head>

<script type="text/javascript">
function setPageBreak()
{
document.getElementById("p2").style.pageBreakBefore="always";
}
</script>
</head>
<body>

<p>This is a test paragraph.</p>
```

```
<input type="button" onclick="setPageBreak()"
value="Set page-break" />

<p id="p2">This is also a test paragraph.</p>

</body>
</html>
```

HTML DOM pageBreakInside 属性

定义和用法

pageBreakInside 声明一个元素内部是否应当放置分页符。

语法：

```
Object.style.pageBreakInside=auto|avoid
```

可能的值

值	描述
auto	默认。如果必要则在元素内部插入分页符。
avoid	避免在元素内部插入分页符。

实例

本例避免在 id 为 p2 的段落中的 page-break：

```
<html>
<head>
<script type="text/javascript">
function setPageBreak()
{
document.getElementById("p2").style.pageBreakInside="avoid";
}
</script>
```

```
</head>
<body>

<p>This is a test paragraph.</p>

<input type="button" onclick="setPageBreak()"
value="Set page to not break inside the second paragraph" />

<p id="p2">This is also a test paragraph. We need some
more text here. This is just filler text.</p>

</body>
</html>
```

HTML DOM scrollbar3dLightColor 属性

定义和用法

scrollbar3dLightColor 属性设置箭头和滚动条的左边和上边的颜色。

语法：

```
Object.style.scrollbar3dLightColor=color
```

可能的值

值	描述
color	颜色值可以是颜色名 (red)，rgb 值 (rgb(255,0,0))，或十六进制 (#ff0000)。

实例

本例设置箭头和滚动条的左边和上边的颜色：

```
<html>
<head>
```

```
<script type="text/javascript">
function setScrollbar3dLightColor()
{
document.body.style.scrollbar3dLightColor="blue";
}
</script>
</head>
<body>

<input type="button" onclick="setScrollbar3dLightColor()"
value="Set Scrollbar3dLightColor" />

</body>
</html>
```

TIY

scrollbar3dLightColor

```
<html>
<head>
<script type="text/javascript">
function setScrollbar3dLightColor()
{
document.body.style.scrollbar3dLightColor="blue";
}
</script>
</head>
<body>

<input type="button" onclick="setScrollbar3dLightColor()"
value="Set Scrollbar3dLightColor" />

<p>An example paragraph</p>
<p>An example paragraph</p>
<p>An example paragraph</p>
<p>An example paragraph</p>
<p>An example paragraph</p>
<p>An example paragraph</p>
<p>An example paragraph</p>
<p>An example paragraph</p>
<p>An example paragraph</p>
<p>An example paragraph</p>
```

```
<p>An example paragraph</p>

</body>
</html>
```

HTML DOM scrollbarArrowColor 属性

定义和用法

scrollbarArrowColor 属性设置滚动条中箭头的颜色。

语法：

```
Object.style.scrollbarArrowColor=color
```

可能的值

值	描述
color	颜色值可以是颜色名 (red)，rgb 值 (rgb(255,0,0))，或十六进制 (#ff0000)。

实例

本例设置滚动条中箭头的颜色：

```
<html>
<head>
<script type="text/javascript">
function setScrollbarArrowColor()
{
document.body.style.scrollbarArrowColor="green";
}
</script>
</head>
<body>

<input type="button" onclick="setScrollbarArrowColor()"
value="Set ScrollbarArrowColor" />
```

```
</body>
</html>
```

TIY

scrollbarArrowColor

```
<html>
<head>
<script type="text/javascript">
function setScrollbarArrowColor()
{
document.body.style.scrollbarArrowColor="green";
}
</script>
</head>
<body>

<input type="button" onclick="setScrollbarArrowColor()" value="Set
ScrollbarArrowColor" />

<p>An example paragraph</p>
<p>An example paragraph</p>
<p>An example paragraph</p>
<p>An example paragraph</p>
<p>An example paragraph</p>
<p>An example paragraph</p>
<p>An example paragraph</p>
<p>An example paragraph</p>
<p>An example paragraph</p>
<p>An example paragraph</p>
<p>An example paragraph</p>

</body>
</html>
```

HTML DOM scrollbarBaseColor 属性

定义和用法

scrollbarBaseColor 属性为整个滚动条设置底色。

语法：

```
Object.style.scrollbarBaseColor=color
```

可能的值

值	描述
color	颜色值可以是颜色名 (red)，rgb 值 (rgb(255,0,0))，或十六进制 (#ff0000)。

实例

本例改变滚动条的颜色：

```
<html>
<head>
<script type="text/javascript">
function setScrollbarColor()
{
document.body.style.scrollbarBaseColor="green";
}
</script>
</head>
<body>

<input type="button" onclick="setScrollbarColor()"
value="Set ScrollbarColor" />

</body>
</html>
```

TIY

scrollbarColor

```
<html>
<head>
<script type="text/javascript">
function setScrollbarColor()
{
document.body.style.scrollbarBaseColor="green";
}
</script>
</head>
<body>

<input type="button" onclick="setScrollbarColor()" value="Set
ScrollbarColor" />

<p>An example paragraph</p>
<p>An example paragraph</p>
<p>An example paragraph</p>
<p>An example paragraph</p>
<p>An example paragraph</p>
<p>An example paragraph</p>
<p>An example paragraph</p>
<p>An example paragraph</p>
<p>An example paragraph</p>
<p>An example paragraph</p>
<p>An example paragraph</p>

</body>
</html>
```

HTML DOM scrollbarDarkShadowColor 属性

定义和用法

scrollbarDarkShadowColor 属性设置箭头和滚动条的右边和下边的颜色。
提示：可以与 scrollbarShadowColor 一同使用，设置滚动条的阴影效果。

语法：

```
Object.style.scrollbarDarkShadowColor=color
```

可能的值

值	描述
color	颜色值可以是颜色名 (red)，rgb 值 (rgb(255,0,0))，或十六进制 (#ff0000)。

实例

本例设置箭头和滚动条的右边和下边的颜色：

```
<html>
<head>
<script type="text/javascript">
function setScrollbarDarkShadowColor()
{
document.body.style.scrollbarDarkShadowColor="darkred";
}
</script>
</head>
<body>

<input type="button" onclick="setScrollbarDarkShadowColor()"
value="Set ScrollbarDarkShadowColor" />

</body>
```

```
</html>
```

TIY

scrollbarDarkShadowColor

```
<html>
<head>
<script type="text/javascript">
function setScrollbarDarkShadowColor()
{
document.body.style.scrollbarDarkShadowColor="darkred";
}
</script>
</head>
<body>

<input type="button" onclick="setScrollbarDarkShadowColor()"
value="Set ScrollbarDarkShadowColor" />

<p>An example paragraph</p>
<p>An example paragraph</p>
<p>An example paragraph</p>
<p>An example paragraph</p>
<p>An example paragraph</p>
<p>An example paragraph</p>
<p>An example paragraph</p>
<p>An example paragraph</p>
<p>An example paragraph</p>
<p>An example paragraph</p>
<p>An example paragraph</p>

</body>
</html>
```

HTML DOM scrollbarFaceColor 属性

定义和用法

scrollbarFaceColor 属性设置滚动条的前景色。

语法：

```
Object.style.scrollbarFaceColor=color
```

可能的值

值	描述
color	颜色值可以是颜色名 (red)，rgb 值 (rgb(255,0,0))，或十六进制 (#ff0000)。

实例

本例改变滚动条的前景色：

```
<html>
<head>
<script type="text/javascript">
function setScrollbarFaceColor()
{
document.body.style.scrollbarFaceColor="green";
}
</script>
</head>
<body>

<input type="button" onclick="setScrollbarFaceColor()"
value="Set ScrollbarFaceColor" />

</body>
</html>
```

TIY

scrollbarFaceColor

```
<html>
<head>
<script type="text/javascript">
function setScrollbarFaceColor()
{
document.body.style.scrollbarFaceColor="green";
```

```
}
</script>
</head>
<body>

<input type="button" onclick="setScrollbarFaceColor()" value="Set
ScrollbarFaceColor" />

<p>An example paragraph</p>
<p>An example paragraph</p>
<p>An example paragraph</p>
<p>An example paragraph</p>
<p>An example paragraph</p>
<p>An example paragraph</p>
<p>An example paragraph</p>
<p>An example paragraph</p>
<p>An example paragraph</p>
<p>An example paragraph</p>
<p>An example paragraph</p>

</body>
</html>
```

HTML DOM scrollbarHighlightColor 属性

定义和用法

scrollbarHighlightColor 属性设置箭头和滚动条的左边和上边的颜色，以及滚动条的背景。

语法：

```
Object.style.scrollbarHighlightColor=color
```

可能的值

值	描述

color	颜色值可以是颜色名 (red)，rgb 值 (rgb(255,0,0))，或十六进制 (#ff0000)。

实例

本例设置箭头和滚动条的左边和上边的颜色，以及滚动条的背景：

```
<html>
<head>
<script type="text/javascript">
function setScrollbarHighlightColor()
{
document.body.style.scrollbarHighlightColor="purple";
}
</script>
</head>
<body>

<input type="button" onclick="setScrollbarHighlightColor()"
value="Set ScrollbarHighlightColor" />

</body>
</html>
```

TIY

scrollbarHighlightColor

```
<html>
<head>
<script type="text/javascript">
function setScrollbarHighlightColor()
{
document.body.style.scrollbarHighlightColor="purple";
}
</script>
</head>
<body>

<input type="button" onclick="setScrollbarHighlightColor()"
value="Set ScrollbarHighlightColor" />

<p>An example paragraph</p>
```

```
<p>An example paragraph</p>
<p>An example paragraph</p>
<p>An example paragraph</p>
<p>An example paragraph</p>
<p>An example paragraph</p>
<p>An example paragraph</p>
<p>An example paragraph</p>
<p>An example paragraph</p>
<p>An example paragraph</p>
<p>An example paragraph</p>

</body>
</html>
```

HTML DOM scrollbarShadowColor 属性

定义和用法

scrollbarShadowColor 属性设置箭头和滚动条的右边和下边的颜色。
提示：该属性可与 scrollbarDarkShadowColor 一同使用，在滚动条上设置阴影效果。

语法：

```
Object.style.scrollbarShadowColor=color
```

可能的值

值	描述
Color	颜色值可以是颜色名 (red)，rgb 值 (rgb(255,0,0))，或十六进制 (#ff0000)。

实例

本例设置箭头和滚动条的右边和下边的颜色：

```
<html>
```

```
<head>
<script type="text/javascript">
function setScrollbarShadowColor()
{
document.body.style.scrollbarShadowColor="pink";
}
</script>
</head>
<body>

<input type="button" onclick="setScrollbarShadowColor()"
value="Set ScrollbarShadowColor" />

</body>
</html>
```

TIY

scrollbarShadowColor

```
<html>
<head>
<script type="text/javascript">
function setScrollbarShadowColor()
{
document.body.style.scrollbarShadowColor="pink";
}
</script>
</head>
<body>

<input type="button" onclick="setScrollbarShadowColor()" value="Set
ScrollbarShadowColor" />

<p>An example paragraph</p>
<p>An example paragraph</p>
<p>An example paragraph</p>
<p>An example paragraph</p>
<p>An example paragraph</p>
<p>An example paragraph</p>
<p>An example paragraph</p>
<p>An example paragraph</p>
<p>An example paragraph</p>
```

```
<p>An example paragraph</p>
<p>An example paragraph</p>

</body>
</html>
```

HTML DOM scrollbarTrackColor 属性

定义和用法

scrollbarTrackColor 属性设置滚动条的背景色。

语法：

```
Object.style.scrollbarTrackColor=color
```

可能的值

值	描述
color	颜色值可以是颜色名 (red)，rgb 值 (rgb(255,0,0))，或十六进制 (#ff0000)。

实例

本例设置滚动条的背景色：

```
<html>
<head>
<script type="text/javascript">
function setScrollbarTrackColor()
{
document.body.style.scrollbarTrackColor="pink";
}
</script>
</head>
<body>

<input type="button" onclick="setScrollbarTrackColor()"
value="Set ScrollbarTrackColor" />
```

```
</body>
</html>
```

TIY

scrollbarTrackColor

```
<html>
<head>
<script type="text/javascript">
function setScrollbarTrackColor()
{
document.body.style.scrollbarTrackColor="pink";
}
</script>
</head>
<body>

<input type="button" onclick="setScrollbarTrackColor()" value="Set
ScrollbarTrackColor" />

<p>An example paragraph</p>
<p>An example paragraph</p>
<p>An example paragraph</p>
<p>An example paragraph</p>
<p>An example paragraph</p>
<p>An example paragraph</p>
<p>An example paragraph</p>
<p>An example paragraph</p>
<p>An example paragraph</p>
<p>An example paragraph</p>
<p>An example paragraph</p>

</body>
</html>
```

HTML DOM borderCollapse 属性

定义和用法

borderCollapse 属性设置表格的边框是否被合并为一个单一的边框，还是象在标准的 HTML 中那样分开显示。

语法：

```
Object.style.borderCollapse=collapse|separate
```

可能的值

值	描述
separate	边框会被分开。
collapse	默认。如果可能，边框会被合并为一个单一的边框。

实例

下面的例子合并表格边框：

```
<html>
<head>
<script type="text/javascript">
function setBorderCollapse()
{
document.getElementById('myTable').style.borderCollapse="collapse
"
}
</script>
</head>
<body>

<table border="1" id="myTable">
  <tr>
    <td>100</td>
    <td>200</td>
```

```
  </tr>
  <tr>
    <td>300</td>
    <td>400</td>
  </tr>
</table>
<br />
<input type="button" onclick="setBorderCollapse()"
value="Collapse border">

</body>
</html>
```

TIY

borderCollapse - 把表格边框合并为单一边框

```
<html>
<head>
<script type="text/javascript">
function setBorderCollapse()
{
document.getElementById('myTable').style.borderCollapse="collapse
";
}
</script>
</head>
<body>

<table border="1" id="myTable">
  <tr>
    <td>100</td>
    <td>200</td>
  </tr>
  <tr>
    <td>300</td>
    <td>400</td>
  </tr>
</table>
<br />
<input type="button" onclick="setBorderCollapse()" value="Collapse
border">
```

```
</body>
</html>
```

HTML DOM borderSpacing 属性

定义和用法

borderSpacing 属性相邻单元格的边框之间的距离（仅用于“边框分离”模式）。

语法：

```
Object.style.borderSpacing=length,length
```

可能的值

值	描述
length length	使用 px、cm 等单位定义距离。如果定义一个 length 参数，那么定义的是水平和垂直间距。如果定义了两个 length 参数，那么第一个设置水平间距，而第二个设置垂直间距。length 不能为负值。

实例

本例设置相邻单元格的边框之间的距离：

```
<html>
<head>
<script type="text/javascript">
function changeBorderSpacing()
{
document.getElementById('myTable').style.borderSpacing="10px"
}
</script>
</head>
<body>

<table border="1" id="myTable">
  <tr>
    <td>100</td>
```

```
    <td>200</td>
  </tr>
  <tr>
    <td>300</td>
    <td>400</td>
  </tr>
</table>

<input type="button" onclick="changeBorderSpacing()"
value="Change border spacing">

</body>
</html>
```

TIY

borderSpacing - 改变相邻单元格的边框之间的距离（请在非 IE 浏览器中查看）

```
<html>
<head>
<script type="text/javascript">
function changeBorderSpacing()
{
document.getElementById('myTable').style.borderSpacing="10px";
}
</script>
</head>
<body>

<table border="1" id="myTable">
  <tr>
    <td>100</td>
    <td>200</td>
  </tr>
  <tr>
    <td>300</td>
    <td>400</td>
  </tr>
</table>
<br />
<input type="button" onclick="changeBorderSpacing()" value="Change
border spacing">
```

```
</body>
</html>
```

HTML DOM captionSide 属性

定义和用法

captionSide 属性设置表格标题的位置。

语法：

```
Object.style.captionSide=top|bottom|left|right
```

可能的值

值	描述
top	默认。把表格标题定位在表格之上。
bottom	把表格标题定位在表格之下。
left	把表格标题定位在表格的左边。
right	把表格标题定位在表格的右边。

实例

本例移动表格标题：

```
<html>
<head>
<style type="text/css">
caption
{
caption-side:bottom;
}
</style>
<script type="text/javascript">
function moveCaption()
{
```

```
document.getElementById('myTable').style.captionSide="right"
}
</script>
</head>
<body>

<table border="1" id="myTable">
  <caption>This is a caption</caption>
  <tr>
    <td>100</td>
    <td>200</td>
  </tr>
  <tr>
    <td>300</td>
    <td>400</td>
  </tr>
</table>
<br />
<input type="button" onclick="moveCaption()"
value="Move table caption">

</body>
</html>
```

TIY

captionSide - 对表格标题进行定位（请在非 IE 浏览器中查看）

```
<html>
<head>
<style type="text/css">
caption
{
caption-side:bottom;
}
</style>
<script type="text/javascript">
function moveCaption()
{
document.getElementById('myTable').style.captionSide="right";
}
</script>
</head>
```

```
<body>

<table border="1" id="myTable">
  <caption>This is a caption</caption>
  <tr>
    <td>100</td>
    <td>200</td>
  </tr>
  <tr>
    <td>300</td>
    <td>400</td>
  </tr>
</table>
<br />
<input type="button" onclick="moveCaption()" value="Move table
caption">

</body>
</html>
```

HTML DOM emptyCells 属性

定义和用法

emptyCells 属性设置是否显示表格中的空单元格（仅用于“分离边框”模式）。

语法：

```
Object.style.emptyCells=hide|show
```

可能的值

值	描述
hide	默认。不在空单元格周围绘制边框。
show	在空单元格周围绘制边框。

实例

本例改变空单元格的显示方式：

```
<html>
<head>
<script type="text/javascript">
function showEmptyCells()
{
document.getElementById('myTable').style.emptyCells="show"
}
</script>
</head>
<body>

<table border="1" id="myTable">
<tr>
<td>100</td>
<td>200</td>
</tr>
<tr>
<td>300</td>
<td></td>
</tr>
</table>

<input type="button" onclick="showEmptyCells()"
value="Show empty cells">

</body>
</html>
```

TIY

改变空单元格的显示方式

```
<html>
<head>
<script type="text/javascript">
function showEmptyCells()
{
document.getElementById('myTable').style.emptyCells="show";
}
```

```
</script>
</head>
<body>

<table border="1" id="myTable">
<tr>
<td>100</td>
<td>200</td>
</tr>
<tr>
<td>300</td>
<td></td>
</tr>
</table>

<input type="button" onclick="showEmptyCells()" value="Show empty
cells">

</body>
</html>
```

HTML DOM tableLayout 属性

定义和用法

tableLayout 属性用来显示表格单元格、行、列的算法规则。

固定表格布局：

固定表格布局与自动表格布局相比，允许浏览器更快地对表格进行布局。
在固定表格布局中，水平布局仅取决于表格宽度、列宽度、表格边框宽度、单元格间距，而与单元格的内容无关。
通过使用固定表格布局，用户代理在接收到第一行后就可以显示表格。

自动表格布局：

在自动表格布局中，列的宽度是由列单元格中没有折行的最宽的内容设定的。
此算法有时会较慢，这是由于它需要在确定最终的布局之前访问表格中所有的内容。

语法：

```
Object.style.tableLayout=automatic|fixed
```

可能的值

值	描述
automatic	默认。列宽度由单元格内容设定。
fixed	列宽由表格宽度和列宽度设定。

实例

本例设置固定表格布局：

```
<html>
<head>
<script type="text/javascript">
function setFixedTableLayout()
{
document.getElementById('myTable').style.tableLayout="fixed";
}
</script>
</head>
<body>

<table id="myTable" border="1" width="100%">
<col width="20%"><col width="40%"><col width="40%">
<tr>
<td>1000000000000000000000000000</td>
<td>10000000</td>
<td>100</td>
</tr>
</table>

<input type="button" onclick="setFixedTableLayout()"
value="Set fixed table layout">

</body>
</html>
```

TIY

改变表格的布局 (tableLayout)

```
<html>
<head>
<script type="text/javascript">
function setFixedTableLayout()
{
document.getElementById('myTable').style.tableLayout="fixed";
}
</script>
</head>
<body>

<table id="myTable" border="1" width="100%">
<col width="20%"><col width="40%"><col width="40%">
<tr>
<td>1000000000000000000000000000</td>
<td>10000000</td>
<td>100</td>
</tr>
</table>

<input type="button" onclick="setFixedTableLayout()" value="Set
fixed table layout">

</body>
</html>
```

HTML DOM color 属性

定义和用法

color 属性设置文本的颜色（元素的前景色）。

语法：

```
Object.style.color=color
```

可能的值

值	描述
color	颜色值可以是颜色名称、rgb 值或者十六进制数。 默认值：取决于浏览器。

实例

本例改变文本的颜色：

```
<html>
<head>
<script type="text/javascript">
function setColor()
{
document.getElementById("p1").style.color="#ff0000";
document.getElementById("p2").style.color="magenta";
}
</script>
</head>
<body>

<p id="p1">This is an example paragraph</p>
<p id="p2">This is also an example paragraph</p>

<input type="button" onclick="setColor()"
value="Change color of text" />

</body>
</html>
```

TIY

改变文本的颜色

```
<html>
<head>
<script type="text/javascript">
function setColor()
{
document.getElementById("p1").style.color="#ff0000";
```

```
document.getElementById("p2").style.color="magenta";
}
</script>
</head>
<body>

<p id="p1">This is an example paragraph.</p>
<p id="p2">This is also an example paragraph.</p>

<input type="button" onclick="setColor()" value="Change color of
text" />

</body>
</html>
```

HTML DOM font 属性

定义和用法

font 属性在一个声明中设置所有字体属性。

语法：

```
Object.style.font=value
```

可能的值

值	描述
• fontStyle • fontVariant • fontWeight • fontSize/lineHeight • fontFamily	设置字体的属性。
caption	为控件定义字体（比如按钮、下拉列表等）。
icon	定义用于标注图标的字体。
menu	定义菜单中使用的字体。

message-box	定义对话框中使用的字体。
small-caption	定义用于标注小型控件的字体。
status-bar	定义在窗口状态栏中使用的字体。

实例

本例改变文本的字体：

```
<html>
<head>
<script type="text/javascript">
function setFont()
{
document.getElementById("p1").style.font="italic bold 12px
arial,serif";
}
</script>
</head>
<body>

<p id="p1">This is an example paragraph.</p>

<input type="button" onclick="setFont()" value="Change paragraph
style" />

</body>
</html>
```

TIY

改变文本的字体

```
<html>
<head>
<script type="text/javascript">
function setFont()
{
document.getElementById("p1").style.font="italic bold 12px
arial,serif";
}
</script>
```

```
</head>
<body>

<p id="p1">This is an example paragraph.</p>

<input type="button" onclick="setFont()" value="Change font" />

</body>
</html>
```

HTML DOM fontFamily 属性

定义和用法

fontFamily 属性定义用于元素文本显示的字体系列。

语法：

```
Object.style.fontFamily=font1,font2,....
```

可能的值

Value	Description
font1, font2,....	字体族名称或类组名称的列表，由逗号分割。

提示和注释

提示：使用逗号分割每个值，并始终提供一个类族名称作为最后的选择。
注释：如果字体族名称中含有空格，请为其加上引号。

实例

本例改变文本的字体系列：

```
<html>
<head>
<script type="text/javascript">
```

```
function setFont()
{
document.getElementById("p1").style.fontFamily="arial,sans-serif"
;
}
</script>
</head>
<body>

<p id="p1">This is an example paragraph.</p>

<input type="button" onclick="setFont()" value="Change font" />

</body>
</html>
```

TIY

fontFamily - 改变文本的字体系列

```
<html>
<head>
<script type="text/javascript">
function setFont()
{
document.getElementById("p1").style.fontFamily="arial,sans-serif"
;
}
</script>
</head>
<body>

<p id="p1">This is an example paragraph.</p>

<input type="button" onclick="setFont()" value="Change font" />

</body>
</html>
```

HTML DOM fontSize 属性

定义和用法

fontSize 属性设置元素的字体大小。

语法：

```
Object.style.fontSize=value
```

可能的值

值	描述
• xx-small • x-small • small • medium • large • x-large • xx-large	把字体的尺寸设置为不同的尺寸，从 xx-small 到 xx-large。 默认值：medium。
smaller	把 font-size 设置为比父元素更小的尺寸。
larger	把 font-size 设置为比父元素更大的尺寸。
length	把 font-size 设置为一个固定的值。
%	把 font-size 设置为基于父元素的一个百分比。

实例

本例改变文本的字体大小：

```
<html>
<head>
<script type="text/javascript">
function setFontSize()
{
document.getElementById("p1").style.fontSize="larger";
}
```

```
</script>
</head>
<body>

<p id="p1">This is an example paragraph.</p>

<input type="button" onclick="setFontSize()"
value="Change font-size" />

</body>
</html>
```

TIY

fontSize - 改变文本的字体大小

```
<html>
<head>
<script type="text/javascript">
function setFontSize()
{
document.getElementById("p1").style.fontSize="larger";
}
</script>
</head>
<body>

<p id="p1">This is an example paragraph.</p>

<input type="button" onclick="setFontSize()" value="Change
font-size" />

</body>
</html>
```

HTML DOM fontSizeAdjust 属性

定义和用法

fontSizeAdjust 属性设置文本的尺寸。

字体的小写字母 "x" 的高度与 "font-size" 高度之间的比率被称为一个字体的 **aspect 值**。当字体拥有高的 **aspect 值**时，那么当此字体被设置为很小的尺寸时会更易阅读。举例：Verdana 的 **aspect 值**是 0.58（意味着当字体尺寸为 100px 时，它的 x-height 是 58px ）。Times New Roman 的 **aspect 值**是 0.46。这就意味着 Verdana 在小尺寸时比 Times New Roman 更易阅读。
该可为某个元素规定一个 **aspect 值**，这样就可以保持首选字体的 x-height。

语法：

```
Object.style.fontSizeAdjust=none|number
```

可能的值

值	描述
none	默认。如果此字体不可用，则不保持此字体的 x-height。
number	定义字体的 aspect 值比率。 **可使用的公式：** 首选字体的字体尺寸 * （font-size-adjust 值 / 可用字体的 aspect 值）=可应用到可用字体的字体尺寸 **举例：** 如果 14px 的 Verdana（aspect 值是 0.58）不可用，但是某个可用的字体的 aspect 值是 0.46，那么替代字体的尺寸将是 14 * (0.58/0.46) = 17.65px。

实例

本例调节字体的大小：

```
<html>
<head>
<script type="text/javascript">
function adjustFontSize()
{
document.getElementById("p1").style.fontSizeAdjust="0.70";
}
</script>
</head>
<body>

<p id="p1">This is an example paragraph</p>
```

```
<input type="button" onclick="adjustFontSize()"
value="Adjust font-size" />

</body>
</html>
```

TIY

fontSizeAdjust - 调节字体的大小

```
<html>
<head>
<script type="text/javascript">
function adjustFontSize()
{
document.getElementById("p1").style.fontSizeAdjust="0.70";
}
</script>
</head>
<body>

<p id="p1">This is an example paragraph</p>

<input type="button" onclick="adjustFontSize()" value="Adjust
font-size" />

</body>
</html>
```

HTML DOM fontStretch 属性

定义和用法

fontStretch 属性用于对当前的 font-family 进行伸缩变形。

语法：

```
Object.style.fontStretch=value
```

可能的值

值	描述
normal	默认值。把缩放比例设置为标准。
wider	把伸展比例设置为更进一步的伸展值
narrower	把收缩比例设置为更进一步的收缩值
• ultra-condensed • extra-condensed • condensed • semi-condensed • semi-expanded • expanded • extra-expanded • ultra-expanded	设置 font-family 的缩放比例。 "ultra-condensed" 是最宽的值，而 "ultra-expanded" 是最窄的值。

实例

本例对字体进行伸展：

```
<html>
<head>
<script type="text/javascript">
function setFontStretch()
{
document.getElementById("p1").style.fontStretch="ultra-expanded";
}
</script>
</head>
<body>

<p id="p1">This is an example paragraph.</p>

<input type="button" onclick="setFontStretch()"
value="Stretch font" />

</body>
</html>
```

TIY

fontStretch - 伸展字体

```
<html>
<head>
<script type="text/javascript">
function setFontStretch()
{
document.getElementById("p1").style.fontStretch="ultra-expanded";
}
</script>
</head>
<body>

<p id="p1">This is an example paragraph.</p>

<input type="button" onclick="setFontStretch()" value="Stretch font"
/>

</body>
</html>
```

HTML DOM fontStyle 属性

定义和用法

fontStyle 属性设置字体的风格。

语法：

```
Object.style.fontStyle=normal|italic|oblique
```

可能的值

值	描述
normal	默认。浏览器显示一个标准的字体。

italic	浏览器会显示一个斜体的字体。
oblique	浏览器会显示一个倾斜的字体。

实例

本例向文本添加斜体样式：

```
<html>
<head>
<script type="text/javascript">
function setFontStyle()
{
document.getElementById("p1").style.fontStyle="italic";
}
</script>
</head>
<body>

<p id="p1">This is an example paragraph.</p>

<input type="button" onclick="setFontStyle()"
value="Change font-style" />

</body>
</html>
```

TIY

fontStyle - 改变文本的风格

```
<html>
<head>
<script type="text/javascript">
function setFontStyle()
{
document.getElementById("p1").style.fontStyle="italic";
}
</script>
</head>
<body>

<p id="p1">This is an example paragraph.</p>
```

```
<input type="button" onclick="setFontStyle()" value="Change
font-style" />

</body>
</html>
```

HTML DOM fontVariant 属性

定义和用法

fontVariant 属性用于设置小型大写字母的字体显示文本

语法：

```
Object.style.fontVariant=normal|small-caps
```

可能的值

值	描述
normal	默认。浏览器会显示一个标准的字体。
small-caps	浏览器会显示小型大写字母的字体。

实例

本例把第一段设置为小型大写字母：

```
<html>
<head>
<script type="text/javascript">
function setSmallCaps()
{
document.getElementById("p1").style.fontVariant="small-caps";
}
</script>
</head>
<body>
```

```
<p id="p1">This is an example paragraph.</p>
<p>This is another example paragraph.</p>

<input type="button" onclick="setSmallCaps()"
value="Display small-caps font" />

</body>
</html>
```

TIY

fontVariant - 把文本设置为小型大写字母

```
<html>
<head>
<script type="text/javascript">
function setSmallCaps()
{
document.getElementById("p1").style.fontVariant="small-caps";
}
</script>
</head>
<body>

<p id="p1">This is an example paragraph.</p>
<p>This is another example paragraph.</p>

<input type="button" onclick="setSmallCaps()" value="Display
small-caps font" />

</body>
</html>
```

HTML DOM fontWeight 属性

定义和用法

fontWeight 属性设置字符的粗细。

语法：

```
Object.style.fontWeight=value
```

可能的值

值	描述
normal	默认。定义标准的字符。
bold	定义粗体字符。
bolder	定义更粗的字符。
lighter	定义更细的字符。
• 100 • 200 • 300 • 400 • 500 • 600 • 700 • 800 • 900	定义由粗到细的字符。400 等同于 normal，而 700 等同于 bold。

实例

本例把第一段设置为粗体：

```
<html>
<head>
<script type="text/javascript">
function setFontWeight()
{
document.getElementById("p1").style.fontWeight="900";
}
</script>
</head>
<body>

<p id="p1">This is an example paragraph.</p>
<p>This is another example paragraph.</p>
```

```
<input type="button" onclick="setFontWeight()"
value="Display bold text" />

</body>
</html>
```

TIY

fontWeight - 把文本设置为粗体

```
<html>
<head>
<script type="text/javascript">
function setFontWeight()
{
document.getElementById("p1").style.fontWeight="900";
}
</script>
</head>
<body>

<p id="p1">This is an example paragraph.</p>
<p>This is another example paragraph.</p>

<input type="button" onclick="setFontWeight()" value="Display bold
text" />

</body>
</html>
```

HTML DOM letterSpacing 属性

定义和用法

letterSpacing 属性设置字符之间的空白。

语法：

```
Object.style.letterSpacing=normal|length
```

可能的值

值	描述
normal	默认。定义字符间的标准空间。
length	定义字符间的固定空间。

实例

本例改变字符之间的空白：

```
<html>
<head>
<script type="text/javascript">
function changeLetterSpacing()
{
document.getElementById("p1").style.letterSpacing="3";
document.getElementById("p2").style.letterSpacing="-1";
}
</script>
</head>
<body>

<p id="p1">This is an example paragraph.</p>
<p id="p2">This is another example paragraph.</p>

<input type="button" onclick="changeLetterSpacing()"
value="Change letter-spacing" />

</body>
</html>
```

TIY

letterSpacing - 改变字符之间的空白

```
<html>
<head>
<script type="text/javascript">
function changeLetterSpacing()
{
```

```
document.getElementById("p1").style.letterSpacing="3";
document.getElementById("p2").style.letterSpacing="-1";
}
</script>
</head>
<body>

<p id="p1">This is an example paragraph.</p>
<p id="p2">This is another example paragraph.</p>

<input type="button" onclick="changeLetterSpacing()" value="Change
letter-spacing" />

</body>
</html>
```

HTML DOM lineHeight 属性

定义和用法

lineHeight 属性设置行之间的距离（行高）。

语法：

```
Object.style.lineHeight=normal|number|length|%
```

可能的值

值	描述
normal	默认。设置合理的行间距。
number	设置数字，此数字会与当前的字体尺寸相乘来设置行间距。
length	设置固定的行间距。
%	基于当前字体尺寸的百分比行间距。

实例

本例改变行高：

```
<html>
<head>
<script type="text/javascript">
function changeLineHeight()
{
document.getElementById("div1").style.lineHeight="2";
}
</script>
</head>
<body>

<div id="div1">
This is some text. This is some text. This is some text.
This is some text. This is some text. This is some text.
This is some text. This is some text. This is some text.
This is some text. This is some text. This is some text.
</div>
<br />
<input type="button" onclick="changeLineHeight()"
value="Change line-height" />

</body>
</html>
```

TIY

lineHeight - 改变行高

```
<html>
<head>
<script type="text/javascript">
function changeLineHeight()
{
document.getElementById("div1").style.lineHeight="2";
}
</script>
</head>
<body>
```

```
<div id="div1">
This is some text. This is some text. This is some text.
This is some text. This is some text. This is some text.
This is some text. This is some text. This is some text.
This is some text. This is some text. This is some text.
</div>
<br />
<input type="button" onclick="changeLineHeight()" value="Change
line-height" />

</body>
</html>
```

HTML DOM quotes 属性

定义和用法

quotes 属性设置嵌入引用的引号类型。

语法：

```
Object.style.quotes=none|string string string string
```

可能的值

值	描述
none	规定 "content" 属性的 "open-quote" 和 "close-quote" 的值不会产生任何引号。
string string string string	定义要使用的引号。前两个值规定第一级引用嵌套，后两个值规定下一级引号嵌套。

HTML DOM textAlign 属性

定义和用法

textAlign 属性对齐元素中的文本。

语法：

```
Object.style.textAlign=left|right|center|justify
```

可能的值

值	描述
left	把文本排列到左边。默认值：由浏览器决定。
right	把文本排列到右边。
center	把文本排列到中间。
justify	根据 textJustify 属性对齐文本。

实例

本例对齐一段文本：

```
<html>
<head>
<script type="text/javascript">
function setTextAlign()
{
document.getElementById("p1").style.textAlign="right";
}
</script>
</head>
<body>

<p id="p1">This is an example paragraph.</p>

<input type="button" onclick="setTextAlign()"
```

```
value="Align text" />

</body>
</html>
```

TIY

textAlign - 对齐一段文本

```
<html>
<head>
<script type="text/javascript">
function setTextAlign()
{
document.getElementById("p1").style.textAlign="right";
}
</script>
</head>
<body>

<p id="p1">This is an example paragraph.</p>

<input type="button" onclick="setTextAlign()" value="Align text" />

</body>
</html>
```

HTML DOM textDecoration 属性

定义和用法

textDecoration 属性对文本进行修饰。

语法：

```
Object.style.textDecoration=none|underline|overline|line-through|
blink
```

可能的值

值	描述
none	默认。定义标准的文本。
underline	定义文本下的一条线。
overline	定义文本上的一条线。
line-through	定义穿过文本下的一条线。
blink	定义闪烁的文本（无法运行在 IE 和 Opera 中）。

实例

本例给文本设置下划线：

```
<html>
<head>
<script type="text/javascript">
function setTextDecoration()
{
document.getElementById("p1").style.textDecoration="overline";
}
</script>
</head>
<body>

<p id="p1">This is an example paragraph.</p>

<input type="button" onclick="setTextDecoration()"
value="Set text-decoration" />

</body>
</html>
```

TIY

textDecoration - 装饰文本

```
<html>
<head>
<script type="text/javascript">
```

```
function setTextDecoration()
{
document.getElementById("p1").style.textDecoration="overline";
}
</script>
</head>
<body>

<p id="p1">This is an example paragraph.</p>

<input type="button" onclick="setTextDecoration()" value="Set
text-decoration" />

</body>
</html>
```

HTML DOM textIndent 属性

定义和用法

textIndent 属性缩进元素中的首行文本。

语法：

```
Object.style.textIndent=length|%
```

可能的值

值	描述
length	定义固定的缩进。默认值：0。
%	定义基于父元素宽度的百分比的缩进。

实例

本例缩进文本：

```
<html>
```

```
<head>
<script type="text/javascript">
function setTextIndent()
{
document.getElementById("div1").style.textIndent="50px";
}
</script>
</head>
<body>

<div id="div1">
This is some text. This is some text. This is some text.
This is some text. This is some text. This is some text.
This is some text. This is some text. This is some text.
This is some text. This is some text. This is some text.
</div>
<br />
<input type="button" onclick="setTextIndent()"
value="Indent first line of text" />

</body>
</html>
```

TIY

textIndent - 缩进文本

```
<html>
<head>
<script type="text/javascript">
function setTextIndent()
{
document.getElementById("div1").style.textIndent="50px";
}
</script>
</head>
<body>

<div id="div1">
This is some text. This is some text. This is some text.
This is some text. This is some text. This is some text.
This is some text. This is some text. This is some text.
This is some text. This is some text. This is some text.
```

```
</div>
<br />
<input type="button" onclick="setTextIndent()" value="Indent first
line of text" />

</body>
</html>
```

HTML DOM textTransform 属性

定义和用法

textTransform 属性会改变元素中的字母大小写，而不论源文档中文本的大小写。

语法：

```
Object.style.textTransform="none|capitalize|uppercase|lowercase|i
nherit"
```

可能的值

值	描述
none	默认。定义带有小写字母和大写字母的标准的文本。
capitalize	文本中的每个单词以大写字母开头。
uppercase	定义仅有大写字母。
lowercase	定义无大写字母，仅有小写字母。

实例

本例把每个单词的第一个字母转换为大写：

```
<html>
<head>
<script type="text/javascript">
function displayResult()
{
```

```
document.getElementById("p1").style.textTransform="capitalize";
}
</script>
</head>
<body>

<p id="p1">This is some text.</p>
<br />

<button type="button" onclick="displayResult()">Convert
text</button>

</body>
</html>
```

HTML DOM whiteSpace 属性

定义和用法

whiteSpace 属性设置如何处理文本中的空白符（比如空格和换行符）。

语法：

```
Object.style.whiteSpace=normal|nowrap|pre
```

可能的值

值	描述
normal	默认。空白会被浏览器忽略。
pre	空白会被浏览器保留。其行为方式类似 HTML 中的 <pre> 标签。
nowrap	文本不会换行，文本会在在同一行上继续，直到遇到 标签为止。

实例

本例不允许在文本内换行：

```
<html>
```

```
<head>
<script type="text/javascript">
function removeWrapping()
{
document.getElementById("div1").style.whiteSpace="nowrap";
}
</script>
</head>
<body>

<div id="div1">
This is some text. This is some text. This is some text.
This is some text. This is some text. This is some text.
This is some text. This is some text. This is some text.
This is some text. This is some text. This is some text.
</div>
<br />
<input type="button" onclick="removeWrapping()"
value="Do not let the text wrap" />

</body>
</html>
```

TIY

whiteSpace - 不允许在文本内换行

```
<html>
<head>
<script type="text/javascript">
function removeWrapping()
{
document.getElementById("div1").style.whiteSpace="nowrap";
}
</script>
</head>
<body>

<div id="div1">
This is some text. This is some text. This is some text.
This is some text. This is some text. This is some text.
This is some text. This is some text. This is some text.
This is some text. This is some text. This is some text.
```

```
</div>
<br />
<input type="button" onclick="removeWrapping()" value="Do not let the
text wrap" />

</body>
</html>
```

HTML DOM wordSpacing 属性

定义和用法

wordSpacing 属性设置文本中的单词间距。

语法：

```
Object.style.wordSpacing=normal|length
```

可能的值

值	描述
normal	默认。定义单词间的标准空间。
length	定义单词间的固定空间。

实例

本例改变单词间的间距：

```
<html>
<head>
<script type="text/javascript">
function changeWordSpacing()
{
document.getElementById("p1").style.wordSpacing="10px";
}
</script>
</head>
```

```
<body>

<p id="p1">This is an example paragraph</p>

<input type="button" onclick="changeWordSpacing()"
value="Change space between words" />

</body>
</html>
```

TIY

wordSpacing - 改变单词间的间距

```
<html>
<head>
<script type="text/javascript">
function changeWordSpacing()
{
document.getElementById("p1").style.wordSpacing="10px";
}
</script>
</head>
<body>

<p id="p1">This is an example paragraph</p>

<input type="button" onclick="changeWordSpacing()" value="Change
space between words" />

</body>
</html>
```

HTML DOM acceptCharset 属性

定义和用法

acceptCharset 属性可设置或返回一个逗号分隔的列表，内容是服务器可接受的字符集。

语法

```
formObject.acceptCharset=charset
```

实例

下面的例子提示出表单数据可能的字符集：

```
<html>
<head>
<script type="text/javascript">
function showCharset()
  {
  var x=document.getElementById("myForm")
  alert(x.acceptCharset)
  }
</script>
</head>
<body>

<form id="myForm" acceptCharset="ISO-8859-1">
Text: <input type="text" value="Hello World!">
<input type="button" onclick="showCharset()"
value="Show charset">
</form>

</body>
</html>
```

TIY

取得表单数据可能的字符集

```
<html>
<head>
<script type="text/javascript">
function showCharset()
{
var x=document.getElementById("myForm")
alert(x.acceptCharset)
}
</script>
```

```
</head>
<body>

<form id="myForm" acceptCharset="ISO-8859-1">
Text: <input type="text" value="Hello World!">
<input type="button" onclick="showCharset()" value="Show charset">
</form>

</body>
</html>
```

HTML DOM accessKey 属性

定义和用法

accessKey 属性可设置或返回访问一个链接的键盘按键。
注释：请使用 Alt + accessKey 为拥有指定快捷键的元素赋予焦点。

语法

```
anchorObject.accessKey=accessKey
```

实例

下面的例子将为链接设置快捷键：

```
<html>
<head>
<script type="text/javascript">
function accesskey()
  {
  document.getElementById('w3').accessKey="w"
  }
</script>
</head>

<body onload="accesskey()">

<a id="w3" href="http://www.w3school.com.cn/">W3School.com.cn</a>
```

```
</body>
</html>
```

TIY

向超链接添加快捷键

```
<html>
<head>
<script type="text/javascript">
function accesskey()
{
document.getElementById('w3').accessKey="w"
document.getElementById('w3dom').accessKey="d"
}
</script>
</head>

<body onload="accesskey()">

<p><a id="w3"
href="http://www.w3school.com.cn/">W3School.com.cn</a> （请使用 Alt
+ w 给该链接赋予焦点）</p>
<p><a id="w3dom" href="http://www.w3school.com.cn/htmldom/">HTML
DOM</a> （请使用 Alt + d 为该链接赋予焦点）</p>

</body>
</html>
```

HTML DOM action 属性

定义和用法

action 属性可设置或返回表单的 action 属性。
action 属性定义了当表单被提交时数据被送往何处。

语法

```
formObject.action=URL
```

实例

下面的例子可更改表单的 action URL：

```
<html>
<head>
<script type="text/javascript">
function changeAction()
  {
  var x=document.getElementById("myForm")
  alert("Original action: " + x.action)
  x.action="index.asp"
  alert("New action: " + x.action)
  x.submit()
  }
</script>
</head>
<body>

<form id="myForm" action="js_examples.asp">
Name: <input type="text" value="Mickey Mouse" />
<input type="button" onclick="changeAction()"
value="Change action attribute and submit form" />
</form>

</body>
</html>
```

TIY

更改表单的 action 属性

```
<html>
<head>
<script type="text/javascript">
function changeAction()
{
var x=document.getElementById("myForm")
```

```
alert("Original action: " + x.action)
x.action="/htmldom/index.asp"
alert("New action: " + x.action)
x.submit()
}
</script>
</head>
<body>

<form id="myForm" action="/i/eg_smile.gif">
名称：<input type="text" value="米老鼠" />
<input type="button" onclick="changeAction()"
value="改变 action 属性并提交表单" />
</form>

</body>
</html>
```

HTML DOM align 属性

定义和用法

align 属性可设置或返回如何根据周围的文本来排列 iframe。

语法

```
iframeObject.align=left|right|top|middle|bottom
```

实例

下面的例子将把 iframe 根据环绕文本排列到右侧：

```
<html>
<body>

<iframe src="frame_a.htm" id="frame1"></iframe>

<p>Some text. Some text. Some text. Some text.</p>
```

```
<script type="text/javascript">
document.getElementById("frame1").align="right";
</script>

</body>
</html>
```

TIY

设置 <iframe> 的排列方式

```
<html>
<body>

<img id="compman" src="/i/eg_compman.gif" alt="Computerman" />
<p>Some text. Some text. Some text. Some text. Some text. Some text.
Some text. Some text. Some text. Some text.</p>

<script type="text/javascript">
document.getElementById("compman").align="right";
</script>

</body>
</html>
```

HTML DOM alt 属性

定义和用法

alt 属性可设置或返回当浏览器无法显示某个区域时所显示的替换文本。

语法

areaObject.alt=alternate_text

实例

下面的例子可返回图像映射中 "Venus" 区域的替换文本：

```
<html>
<body>

<img src="planets.gif"
width="145" height="126"
usemap="#planetmap" />

<map name="planetmap">
<area id="venus" shape="circle"
coords="124,58,8"
alt="The planet Venus"
href="venus.htm" />
</map>

<p>Alternate text for "Venus" is:
<script type="text/javascript">
x=document.getElementById('venus');
document.write(x.alt);
</script>
</p>

</body>
</html>
```

TIY

返回图像映射中某个区域的替换文本

```
<html>
<body>

<img src="/i/eg_planets.jpg"
usemap="#planetmap" />

<map name="planetmap">
<area id="venus" shape="circle"
coords="180,139,14"
alt="The planet Venus"
href="/example/hdom/venus.html" />
</map>

<p>"Venus" 的替代文本是:
<script type="text/javascript">
```

```
x=document.getElementById('venus');
document.write(x.alt);
</script>
</p>

</body>
</html>
```

HTML DOM border 属性

定义和用法

border 属性可设置或返回图像周围的边框宽度。

语法

```
imageObject.border=pixels
```

实例

下面的例子可更改图像的边框宽度：

```
<html>
<head>
<script type="text/javascript">
function changeBorder()
  {
  document.getElementById("compman").border="3"
  }
</script>
</head>
<body>

<img id="compman" src="compman.gif" alt="Computerman" />
<br /><br />
<input type="button" onclick="changeBorder()"
value="Change border" />

</body>
```

```
</html>
```

TIY

更改图像周围的边框

```
<html>
<head>
<script type="text/javascript">
function changeBorder()
  {
  document.getElementById("compman").border="3"
  }
</script>
</head>
<body>

<img id="compman" src="/i/eg_compman.gif" alt="Computerman" />
<br /><br />
<input type="button" onclick="changeBorder()" value="Change border"
/>

</body>
</html>
```

HTML DOM charset 属性

定义和用法

charset 属性可设置或返回被链接资源的字符集。

语法

```
anchorObject.charset=charset
```

实例

下面的例子可设置被链资源的字符串：

```
<html>
<body>

<p>
<a id="myAnchor"
href="http://www.w3school.com.cn">W3School.com.cn</a>
</p>

<script type="text/javascript">
x=document.getElementById("myAnchor");
x.charset="ISO-8859-1";
</script>

</body>
</html>
```

TIY

设置被链资源的字符串

```
<html>
<body>
<p>
<a id="myAnchor"
href="http://www.w3school.com.cn">W3School.com.cn</a>
</p>
<script type="text/javascript">
x=document.getElementById("myAnchor");
x.charset="ISO-8859-1";
document.write(x.charset);
</script>
</body>
</html>
```

HTML DOM checked 属性

定义和用法

checked 属性设置或返回 checkbox 是否应被选中。

语法

```
checkboxObject.checked=true|false
```

说明

该属性保存了 checkbox 的当前状态，不管何时，这个值发生变化的时候，onclick 事件句柄就会触发（也可能触发 onchange 事件句柄）。

实例

下面的例子可设置该 checkbox 的状态：

```
<html>
<head>
<script type="text/javascript">
function check()
  {
  document.getElementById("check1").checked=true
  }
function uncheck()
  {
  document.getElementById("check1").checked=false
  }
</script>
</head>
<body>

<form>
<input type="checkbox" id="check1" />
<input type="button" onclick="check()" value="Check Checkbox" />
<input type="button" onclick="uncheck()" value="Uncheck Checkbox" />
</form>

</body>
</html>
```

TIY

选定以及不选定 checkbox

```
<html>
<head>
<script type="text/javascript">
function check()
{
document.getElementById("myCheck").checked=true
}

function uncheck()
{
document.getElementById("myCheck").checked=false
}
</script>
</head>

<body>
<form>
<input type="checkbox" id="myCheck" />
<input type="button" onclick="check()" value="选定复选框" />
<input type="button" onclick="uncheck()" value="取消选定复选框" />
</form>
</body>

</html>
```

一个表单中的若干个 checkbox

```
<html>
<head>
<script type="text/javascript">
function createOrder()
{
coffee=document.forms[0].coffee
txt=""
for (i=0;i<coffee.length;++ i)
{
if (coffee[i].checked)
{
txt=txt + coffee[i].value + " "
}
}
document.getElementById("order").value="您订购的咖啡带有： " + txt
}
</script>
```

```
</head>

<body>
<p>你喜欢怎么喝咖啡？</p>
<form>
<input type="checkbox" name="coffee" value="奶油">加奶油<br />
<input type="checkbox" name="coffee" value="糖块">加糖块<br />
<br />
<input type="button" onclick="createOrder()" value="发送订单">
<br /><br />
<input type="text" id="order" size="50">
</form>
</body>

</html>
```

Checkbox - 把文本转换为大写

```
<html>
<head>
<script type="text/javascript">
function convertToUcase()
{
document.getElementById("fname").value=document.getElementById("f
name").value.toUpperCase()
document.getElementById("lname").value=document.getElementById("l
name").value.toUpperCase()
}
</script>
</head>

<body>
<form name="form1">
名：<input type="text" id="fname" size="20" />
<br /><br />
姓：<input type="text" id="lname" size="20" />
<br /><br />
转换为大写
<input type="checkBox" onclick="if (this.checked)
{convertToUcase()}">
</form>
</body>

</html>
```

HTML DOM coords 属性

定义和用法

coords 属性设置或返回一个逗号分隔列表，其中包含了图像映射中一个链接的坐标。

语法

```
anchorObject.coords=coordinates
```

实例

下面的例子可返回图像映射中一个链接的坐标：

```
<html>
<body>

<img src="planets.gif"
width="145" height="126"
usemap="#planetmap" />

<map name="planetmap">
<area id="venus" shape="circle"
coords="124,58,8"
alt="Venus"
href="venus.htm" />
</map>

<p>Venus coordinates:
<script type="text/javascript">
x=document.getElementById("venus");
document.write(x.coords);
</script>
</p>

</body>
</html>
```

TIY

取得图像映射中一个链接的坐标

```
<html>
<body>

<img src="/i/eg_planets.jpg"
usemap="#planetmap" />

<map name="planetmap">
<area id="venus" shape="circle"
coords="180,139,14"
alt="Venus"
href="/example/hdom/venus.html" />
</map>

<p>Venus 的坐标（coords 属性）是：
<script type="text/javascript">
x=document.getElementById("venus");
document.write(x.coords);
</script>
</p>

</body>
</html>
```

HTML DOM cols 属性

定义和用法

cols 属性可设置或返回框架集中列的数目和尺寸。
由逗号分割的像素或百分比列表定义了列的数目和宽度：

语法

```
framesetObject.cols=col1,col2,col3....
```

实例

在我们的例子中，首先将创建包含带有两个列的框架集的 HTML 文档。每列设置为浏览器窗口的 50%：

```
<html>
<frameset cols="50%,50%">
  <frame src="frame_cols.htm">
  <frame src="frame_a.htm">
</frameset>
</html>
```

HTML 文档 "frame_cols.htm" 被放入第一列，而 HTML 文档 "frame_a.htm" 被放入第二列。

下面是 "frame_cols.htm" 的源代码：

```
<html>
<head>
<script type="text/javascript">
function changeCols()
  {
  parent.document.getElementById("main").cols="30%,70%"
  }

function restoreCols()
  {
  parent.document.getElementById("main").cols="50%,50%"
  }
</script>
</head>
<body>

<input type="button" onclick="changeCols()"
value="Change column size" />
<input type="button" onclick="restoreCols()"
value="Restore column size" />

</body>
</html>
```

HTML DOM complete 属性

定义和用法

complete 属性可返回浏览器是否已完成对图像的加载。
如果加载完成，则返回 true，否则返回 fasle。

语法

```
imageObject.complete
```

实例

下面的例子可检测图形是否已完成加载：

```
<html>
<head>
<script type="text/javascript">
function alertComplete()
{
alert(document.getElementById("compman").complete)
}
</script>
</head>
<body onload="alertComplete()">

<img id="compman" src="compman.gif" alt="Computerman"
width="107" height="98" />

</body>
</html>
```

TIY

检测图形是否已完成加载

```
<html>
<head>
<script type="text/javascript">
```

```
function alertComplete()
  {
  alert("Image loaded=" +
document.getElementById("compman").complete)
  }
</script>
</head>
<body onload="alertComplete()">

<img id="compman" src="/i/eg_compman.gif" alt="Computerman" />

</body>
</html>
```

HTML DOM contentDocument 属性

定义和用法

contentDocument 属性能够以 HTML 对象来返回 iframe 中的文档。
可以通过所有标准的 DOM 方法来处理被返回的对象。

语法

```
iframeObject.contentDocument
```

实例

下面的例子可从载入 iframe 的文档的第一个 <h2> 元素中提取文本：

```
<html>
<head>
<script type="text/javascript">
function getTextNode()
{
var x=document.getElementById("frame1").contentDocument;
alert(x.getElementsByTagName("h2")[0].childNodes[0].nodeValue);
}
</script>
</head>
```

```
<body>
<iframe src="frame_a.htm" id="frame1"></iframe>
<br /><br />
<input type="button" onclick="getTextNode()" value="Get text" />

</body>
</html>
```

TIY

从 iframe 文档中提取文本节点

```
<html>
<head>
<script type="text/javascript">
function getText()
  {
  var x=document.getElementById("frame1").contentDocument;
  alert(x.getElementsByTagName("h3")[0].childNodes[0].nodeValue);
  }
</script>
</head>
<body>

<iframe src="/example/hdom/frame_a.html" id="frame1" ></iframe>
<br /><br />
<input type="button" onclick="getText()" value="Get text" />

</body>
</html>
```

HTML DOM defaultChecked 属性

定义和用法

defaultChecked 属性可返回 checked 属性的默认值。
该属性在 checkbox 默认被选定时返回 true，否则返回 false。

语法

```
checkboxObject.defaultChecked
```

实例

下面的例子可返回 checked 属性的值：

```
<html>
<body>

<form>
<input type="checkbox" id="myCheck" checked="checked" />
</form>

<script type="text/javascript">
alert(document.getElementById("myCheck").defaultChecked)
</script>

</body>
</html>
```

TIY

返回 checkbox 是否默认被选中

```
<html>
<body>

<form>
<input type="checkbox" id="myCheck" checked="checked" />
</form>

<script type="text/javascript">
alert(document.getElementById("myCheck").defaultChecked)
</script>

</body>
</html>
```

选定以及不选定 checkbox

```
<html>
<head>
<script type="text/javascript">
function check()
{
document.getElementById("myCheck").checked=true
}

function uncheck()
{
document.getElementById("myCheck").checked=false
}
</script>
</head>

<body>
<form>
<input type="checkbox" id="myCheck" />
<input type="button" onclick="check()" value="选定复选框" />
<input type="button" onclick="uncheck()" value="取消选定复选框" />
</form>
</body>

</html>
```

HTML DOM disabled 属性

定义和用法

disabled 属性可设置或返回是否禁用按钮。

语法

```
buttonObject.disabled=true|false
```

实例

下面的例子将禁用按钮：

```
<html>
<head>
<script type="text/javascript">
function disableButton()
{
document.getElementById("myButton").disabled=true
}
</script>
</head>

<body>
<form>
<button id="myButton" onClick="disableButton()">
Click Me!</button>
</form>
</body>

</html>
```

TIY

提示按钮的 id 和 类型 + 禁用按钮

```
<html>
<head>
<script type="text/javascript">
function alertId()
{
var txt="Id: " + document.getElementById("myButton").id
txt=txt + ", type: " + document.getElementById("myButton").type
alert(txt)
document.getElementById("myButton").disabled=true
}
</script>
</head>

<body>
<form>
<button id="myButton" onClick="alertId()">请点击我！</button>
```

```
</form>
</body>

</html>
```

HTML DOM enctype 属性

定义和用法

enctype 属性可设置或返回用于编码表单内容的 MIME 类型。
如果表单没有 enctype 属性，那么当提交文本时的默认值是 "application/x-www-form-urlencoded"。
当 input type 是 "file" 时，值是 "multipart/form-data"。

语法

```
formObject.encoding=encoding
```

实例

下面的例子可返回表单的编码类型：

```
<html>
<head>
<script type="text/javascript">
function showEnctype()
  {
  var x=document.getElementById("myForm")
  alert(x.enctype)
  }
</script>
</head>
<body>

<form id="myForm" enctype="application/x-www-form-urlencoded">
Name: <input type="text" value="Mickey Mouse" />
<input type="button" onclick="showEnctype()"
value="Show enctype" />
</form>
```

```
</body>
</html>
```

TIY

取得表单的编码类型。

```
<html>
<head>
<script type="text/javascript">
function showEnctype()
  {
  var x=document.getElementById("myForm")
  alert(x.enctype)
  }
</script>
</head>
<body>

<form id="myForm" enctype="application/x-www-form-urlencoded">
Name: <input type="text" value="Mickey Mouse" />
<input type="button" onclick="showEnctype()"
value="Show enctype" />
</form>

</body>
</html>
```

HTML DOM form 属性

定义和用法

form 属性可返回对包含该按钮的表单的引用。
如果包含该按钮，则返回一个 form 对象，如果未包含该按钮，则返回 null。

语法

```
buttonObject.form
```

实例

下面的例子可返回该按钮所属的表单的 id：

```
<html>
<body>

<form id="form1">
<button id="button1">Click me!</button>
</form>

<p>The form containing the button is:
<script type="text/javascript">
x=document.getElementById('button1');
document.write(x.form.id);
</script></p>

</body>
</html>
```

TIY

返回按钮所属的表单的 id

```
<!DOCTYPE html>
<html>
<body>

<form id="form1">
<button id="button1" type="button">点击这里</button>
</form>

<p>包含该按钮的表单的 id 是：
<script>
document.write(document.getElementById("button1").form.id);
</script>
</p>

</body>
</html>
```

HTML DOM hash 属性

定义和用法

hash 属性可设置或返回一个区域中 URL 的锚部分。

语法

```
areaObject.hash=anchorname
```

实例

下面的例子可把 URL 的锚部分 #top 更改为 #bottom：

```
<html>
<head>
<script type="text/javascript">
function changeLink()
  {
  document.getElementById('venus').hash="bottom"
  }
</script>
</head>
<body>

<img src="planets.gif"
width="145" height="126"
usemap="#planetmap" />

<map name="planetmap">
<area id="venus" shape="circle"
coords="124,58,8"
alt="Venus"
href="venus2.htm#top" />
</map>

<input type="button" onclick="changeLink()" value="Change link" />

</body>
```

```
</html>
```

TIY

更改一个区域中 URL 的锚部分

```
<html>
<head>
<script type="text/javascript">
function changeLink()
{
document.getElementById('venus').hash="bottom"
}
</script>
</head>
<body>

<img src="/i/eg_planets.jpg"
usemap="#planetmap" />

<map name="planetmap">
<area
id="venus"
shape="circle"
coords="180,139,14"
href ="/example/hdom/venus2.html#top"
alt="Venus" />
</map>

<input type="button" onclick="changeLink()" value="改变链接" />

</body>
</html>
```

HTML DOM height 属性

定义和用法

height 属性可设置或返回 iframe 的高度（以像素或百分比为单位）。

语法

```
iframeObject.height=pixels
```

实例

下面的例子可更改 iframe 的高度：

```
<html>
<head>
<script type="text/javascript">
function changeHeight()
  {
  document.getElementById("frame1").height="200";
  }
</script>
</head>
<body>

<iframe src="frame_a.htm" id="frame1" height="100"></iframe>
<br /><br />

<input type="button" onclick="changeHeight()"
value="Change height" />

</body>
</html>
```

TIY

更改 iframe 的高度和宽度

```
<html>
<head>
<script type="text/javascript">
function changeSize()
{
document.getElementById("frame1").height="300";
document.getElementById("frame1").width="300";
}
function restoreSize()
{
```

```
document.getElementById("frame1").height="200";
document.getElementById("frame1").width="200";
}
</script>
</head>

<body>
<iframe src="/example/hdom/frame_a.html" id="frame1" height="200"
width="200"></iframe>
<br /><br />

<input type="button" onclick="changeSize()" value="Change size" />
<input type="button" onclick="restoreSize()" value="Restore size" />

</body>
</html>
```

HTML DOM host 属性

定义和用法

host 属性可设置或返回 URL 的主机名和端口。

语法

```
areaObject.host=host
```

实例

下面的例子可返回 "Venus" 区域的主机名和端口：

```
<html>
<body>

<img src="planets.gif"
width="145" height="126"
usemap="#planetmap" />

<map name="planetmap">
```

```
<area id="venus" shape="circle"
coords="124,58,8"
alt="Venus"
href="venus.htm" />
</map>

<p>The hostname and port for the "Venus" area are:
<script type="text/javascript">
x=document.getElementById('venus');
document.write(x.host);
</script>
</p>

</body>
</html>
```

TIY

返回图像映射中某个区域的链接的主机名和端口

```
<html>
<body>

<img src="/i/eg_planets.jpg"
usemap="#planetmap" />

<map name="planetmap">
<area
id="venus"
shape="circle"
coords="180,139,14"
href ="/example/html/venus.html"
alt="Venus" />
</map>

<p>"Venus" 区域的 hostname 和 port 是:
<script type="text/javascript">
x=document.getElementById('venus');
document.write(x.host);
</script>
</p>

</body>
```

```
</html>
```

HTML DOM href 属性

定义和用法

href 属性可设置或返回被链资源的 URL。

语法

```
anchorObject.href=URL
```

实例

下面的例子将更改一个链接的文本、URL 以及 target：

```
<html>
<head>
<script type="text/javascript">
function changeLink()
{
document.getElementById('myAnchor').innerHTML="W3School";
document.getElementById('myAnchor').href="http://www.w3school.com.
cn";
document.getElementById('myAnchor').target="_blank";
}
</script>
</head>

<body>
<a id="myAnchor" href="http://www.microsoft.com">Microsoft</a>
<input type="button" onclick="changeLink()" value="Change link">
</body>

</html>
```

TIY

更改一个链接的文本、URL 以及 target

```
<html>
<head>
<script type="text/javascript">
function changeLink()
{
document.getElementById('myAnchor').innerHTML="访问 W3School"
document.getElementById('myAnchor').href="http://www.w3school.com.
cn"
document.getElementById('myAnchor').target="_blank"
}
</script>
</head>

<body>
<a id="myAnchor" href="http://www.microsoft.com">访问 Microsoft</a>
<input type="button" onclick="changeLink()" value="改变链接">
<p>在本例中，我们改变超链接的文本和 URL。我们也改变 target 属性。target 属性的默认设置是 "_self"，这意味着会在相同的窗口中打开链接。通过把 target 属性设置为 "_blank"，链接将在新窗口中打开。</p>
</body>

</html>
```

HTML DOM hreflang 属性

定义和用法

hreflang 属性可设置或返回被链资源的语言代码。

语法

```
anchorObject.hreflang=languagecode
```

实例

下面的例子将返回被链资源的语言代码：

```
<html>
```

```
<body>

<p><a id="myAnchor" hreflang="zh-cn"
href="http://www.w3school.com.cn">W3School.com.cn</a></p>

<script type="text/javascript">
x=document.getElementById("myAnchor");
document.write(x.hreflang);
</script>

</body>
</html>
```

TIY

取得被链资源的语言代码

```
<html>
<body>

<p><a id="myAnchor" hreflang="zh-cn"
href="http://www.w3school.com.cn">W3School.com.cn</a></p>

<script type="text/javascript">
x=document.getElementById("myAnchor");
document.write(x.hreflang);
</script>

</body>
</html>
```

HTML DOM id 属性

定义和用法

id 属性可返回一个链接的 id。

语法

```
anchorObject.id=ide
```

实例

下面的例子可输出一个链接的 id：

```
<html>
<body>

<p><a id="myAnchor"
href="http://www.w3school.com.cn">Visit W3School.com.cn</a></p>

<script type="text/javascript">
x=document.getElementById("myAnchor");
document.write(x.id);
</script>

</body>
</html>
```

TIY

取得链接的 id

```
<html>
<body>

<p><a id="myAnchor" href="http://www.w3school.com.cn">Visit
W3School.com.cn</a></p>

<script type="text/javascript">
x=document.getElementById("myAnchor");
document.write(x.id);
</script>

</body>
</html>
```

HTML DOM isMap 属性

定义和用法

isMap 属性返回图像是否是服务器端图像映射。

语法

```
imageObject.isMap
```

实例

下面的例子可检测图像是否是服务器端图像映射：

```
<html>
<body>

<img id="planets" src="planets.gif"
width="145" height="126"
usemap="#planetmap" />

<map name="planetmap">
<area id="venus" shape="circle"
coords="124,58,8"
alt="The planet Venus"
href="venus.htm" />
</map>

<p>Is the image a server-side image map?
<script type="text/javascript">
x=document.getElementById('planets');
document.write(x.isMap);
</script>
</p>

</body>
</html>
```

TIY

检测图像是否是服务器端图像映射

```
<html>
<body>

<img id="planets"
src="/i/eg_planets.jpg"
usemap="#planetmap" />

<map name="planetmap">
<area shape="circle"
coords="180,139,14"
href ="/example/html/venus.html"
target ="_blank"
alt="Venus" />
</map>

<p>该图像是服务器端图像映射吗?
<script type="text/javascript">
x=document.getElementById('planets');
document.write(x.isMap);
</script>
</p>

</body>
</html>
```

HTML DOM innerHTML 属性

定义和用法

innerHTML 属性设置或返回一个链接的内容。

语法

```
anchorObject.innerHTML=text
```

实例

下面的例子将更改一个链接的文本、URL 以及 target：

```
<html>
<head>
<script type="text/javascript">
function changeLink()
{
document.getElementById('myAnchor').innerHTML="W3School";
document.getElementById('myAnchor').href="http://www.w3school.com.
cn";
document.getElementById('myAnchor').target="_blank";
}
</script>
</head>

<body>
<a id="myAnchor" href="http://www.microsoft.com">Microsoft</a>
<input type="button" onclick="changeLink()" value="Change link">
</body>

</html>
```

TIY

更改一个链接的文本、URL 以及 target

```
<html>
<head>
<script type="text/javascript">
function changeLink()
{
document.getElementById('myAnchor').innerHTML="访问 W3School"
document.getElementById('myAnchor').href="http://www.w3school.com.
cn"
document.getElementById('myAnchor').target="_blank"
}
</script>
</head>

<body>
<a id="myAnchor" href="http://www.microsoft.com">访问 Microsoft</a>
```

```
<input type="button" onclick="changeLink()" value="改变链接">
<p>在本例中，我们改变超链接的文本和 URL。我们也改变 target 属性。target 属性的默认设置是 "_self"，这意味着会在相同的窗口中打开链接。通过把 target 属性设置为 "_blank"，链接将在新窗口中打开。</p>
</body>

</html>
```

HTML DOM length 属性

定义和用法

length 属性可返回表单中元素的数目。

语法

```
formObject.length
```

实例

下面的例子可返回表单中元素的数目：

```
<html>
<body>

<form id="myForm" method="get">
Name: <input type="text" size="20" value="Peter Griffin" /><br />
Text: <input type="text" size="20" value="Hello World" />
</form>

<script type="text/javascript">
document.write("Number of elements in myForm: ");
document.write(document.getElementById('myForm').length);
</script>

</body>
</html>
```

TIY

取得表单中元素的数目

```
<html>
<body>

<form id="myForm" method="get">
Name: <input type="text" size="20" value="Peter Griffin" /><br />
Text: <input type="text" size="20" value="Hello World" />
</form>

<script type="text/javascript">
document.write("Number of elements in myForm: ");
document.write(document.getElementById('myForm').length);
</script>

</body>
</html>
```

HTML DOM longDesc 属性

定义和用法

longDesc 属性可设置或返回包含 iframe 内容描述的文档的 URL。
提示：请把该属性用于不支持框架的浏览器。

语法

```
iframeObject.longDesc=URL
```

实例

下面的例子创建了一个指向一个描述页面的链接：

```
<html>
<body>
<iframe src="frame_a.htm" id="frame1"
longdesc="frame_description.htm"></iframe>
```

```
<br />

<script type="text/javascript">
x=document.getElementById("frame1");
document.write("Description for frame contents: ");
document.write('<a href="' + x.longDesc + '">Description</a>');
</script>

</body>
</html>
```

TIY

使用 <iframe> 元素中的 longdesc 属性

```
<html>
<body>

<iframe src="/example/hdom/frame_a.html" id="frame1"
longdesc="/example/hdom/frame_description.html"></iframe>
<br />

<script type="text/javascript">
x=document.getElementById("frame1");
document.write("Description for frame contents: ");
document.write('<a href="' + x.longDesc + '">Description</a>');
</script>

</body>
</html>
```

HTML DOM lowsrc 属性

定义和用法

lowsrc 属性可设置或返回图像的低分辨率版本的 URL。

语法

```
imageObject.lowsrc=URL
```

实例

下面的例子可创建一个指向图像的低分辨率版本的链接：

```
<html>
<body>
<img id="compman" src="compman.gif"
lowsrc="compman_lowres.gif" alt="Computerman" />
<br />

<script type="text/javascript">
var x=document.getElementById("compman");
document.write('<a href="' + x.lowsrc + '">Low resolution</a>');
</script>

</body>
</html>
```

TIY

创建一个指向图像的低分辨率版本的链接

```
<html>
<body>

<img id="compman" src="/i/eg_compman.gif"
lowsrc="/i/eg_compman_lowres.gif" alt="Computerman" />
<br />

<script type="text/javascript">
var x=document.getElementById("compman");
document.write('<a href="' + x.lowsrc + '">Low resolution
version</a>');
</script>

</body>
</html>
```

HTML DOM marginHeight 属性

定义和用法

marginHeight 属性可设置或返回框架顶部和底部的页面空白（以像素计）。

语法

```
iframeObject.marginHeight=pixels
```

实例

下面的例子可返回 iframe 顶部和底部的页面空白：

```
<html>
<body>
<iframe src="frame_a.htm" id="frame1" marginheight="50"></iframe>
<br />

<script type="text/javascript">
x=document.getElementById("frame1");
document.write("Top and bottom margins of the iframe are: ");
document.write(x.marginHeight);
</script>

</body>
</html>
```

TIY

返回 iframe 的顶部和底部页空白

```
<html>
<body>
<iframe src="/example/hdom/frame_a.html" id="frame1"
marginheight="50"></iframe>
<br /><br />

<script type="text/javascript">
```

```
x=document.getElementById("frame1");
document.write("Top and bottom margins of the iframe are: ");
document.write(x.marginHeight);
</script>

</body>
</html>
```

HTML DOM marginWidth 属性

定义和用法

marginWidth 属性可设置或返回框架的左侧和右侧边缘的页面空白（以像素为单位）。

语法

```
iframeObject.marginWidth=pixels
```

实例

下面的例子将返回 iframe 的左边缘和右边缘的页空白：

```
<html>
<body>
<iframe src="frame_a.htm" id="frame1" marginwidth="50"></iframe>
<br /><br />

<script type="text/javascript">
x=document.getElementById("frame1");
document.write("Left and right margins of the iframe are: ");
document.write(x.marginWidth);
</script>

</body>
</html>
```

TIY

返回 iframe 的左边缘和右边缘的页空白

```
<html>
<body>
<iframe src="/example/hdom/frame_a.html" id="frame1"
marginwidth="50"></iframe>
<br /><br />

<script type="text/javascript">
x=document.getElementById("frame1");
document.write("Left and right margins of the iframe are: ");
document.write(x.marginWidth);
</script>

</body>
</html>
```

HTML DOM maxLength 属性

定义和用法

maxLength 属性可设置或返回密码域中所允许的最大字符数。

语法

```
passwordObject.maxLength=number_of_chars
```

实例

下面的例子可显示指定的密码域中所允许的最大字符数：

```
<html>
<head>
<script type="text/javascript">
function alertLength()
  {
  alert(document.getElementById("password1").maxLength)
  }
</script>
</head>
```

```
<body>

<form>
<input type="password" id="password1" maxLength="8" />
<input type="button" id="button1" onclick="alertLength()"
value="Alert maxLength" />
</form>

</body>
</html>
```

TIY

获得密码域中所允许的最大字符数

```
<html>
<head>
<script type="text/javascript">
function alertLength()
  {
  alert(document.getElementById("password1").maxLength)
  }
</script>
</head>
<body>

<form>
<input type="password" id="password1" maxLength="8" />
<input type="button" id="button1" onclick="alertLength()"
value="Alert maxLength" />
</form>

</body>
</html>
```

HTML DOM media 属性

定义和用法

media 属性设置或返回显示文档的设备。

对于样式信息而言，目标媒介非常重要。移动设备和桌面计算机的样式可能是不同的。

语法

```
linkObject.media=device
```

设备可以是下列值之一：

值	描述
screen	针对不分页的计算机屏幕。
tty	针对使用等宽字符网格的媒介（比如打字机、终端或其他具有有限显示能力的设备）。
tv	针对 TV 类型的设备。
projection	针对投影机。
handheld	针对手持设备。
print	针对打印预览模式的页面和文档。
braille	针对盲人用点字法触觉反馈装置。
aural	针对语音合成。
all	针对所有设备或装置。

实例

下面的例子获得使用 link 元素的媒介类型：

```
<html>
<head>
<link rel="stylesheet" type="text/css" media="screen"
id="style1" href="try_dom_link.css" />
</head>
<body>

<script type="text/javascript">
x=document.getElementById("style1");
document.write("Intended media type=" + x.media);
</script>

</body>
```

```
</html>
```

TIY

获得使用 link 元素的媒介类型

```
<html>
<head>
<link rel="stylesheet" type="text/css" media="screen"
id="style1" href="/example/hdom/try_dom_link.css" />
</head>
<body>

<script type="text/javascript">
x=document.getElementById("style1");
document.write("Intended media type=" + x.media);
</script>

</body>
</html>
```

HTML DOM method 属性

定义和用法

method 属性可设置或返回用于表单提交的 HTTP 方法。

语法

```
formObject.method=get|post
```

实例

下面的例子弹出的对话框中显示了发送表单数据所使用的 HTTP 方法：

```
<html>
<head>
<script type="text/javascript">
function showMethod()
```

```
  {
  var x=document.getElementById("myForm")
  alert(x.method)
  }
</script>
</head>
<body>

<form id="myForm" method="post">
Name: <input type="text" size="20" value="Mickey Mouse" />
<input type="button" onclick="showMethod()" value="Show method" />
</form>

</body>
</html>
```

TIY

返回向服务器发送数据的 HTTP 方法

```
<html>
<head>
<script type="text/javascript">
function showMethod()
  {
  var x=document.getElementById("myForm")
  alert(x.method)
  }
</script>
</head>
<body>

<form id="myForm" method="post">
名称：<input type="text" value="米老鼠" />
<input type="button" onclick="showMethod()" value="显示 method" />
</form>

</body>
</html>
```

HTML DOM multiple 属性

定义和用法

multiple 属性可设置或返回是否可有多个选项被选中。
注释：Opera 9 无法在脚本中设置该属性，仅能返回它。

语法

```
selectObject.multiple=true|false
```

实例

The following example returns the id of the dropdown list:

```
<html>
<head>
<script type="text/javascript">
function selectMultiple()
  {
  document.getElementById("mySelect").multiple=true
  }
</script>
</head>
<body>

<form>
<select id="mySelect" size="4">
  <option>Apple</option>
  <option>Pear</option>
  <option>Banana</option>
  <option>Orange</option>
</select>
<input type="button" onclick="selectMultiple()"
value="Select multiple">
</form>

</body>
</html>
```

TIY

选择下拉列表中的多个选项

```
<html>
<head>
<script type="text/javascript">
function selectMultiple()
  {
  document.getElementById("mySelect").multiple=true
  }
</script>
</head>
<body>

<form>
<select id="mySelect" size="4">
  <option>苹果</option>
  <option>桃子</option>
  <option>香蕉</option>
  <option>桔子</option>
</select>
<input type="button" onclick="selectMultiple()" value="选择多个">
</form>
<p>在您点击“选择多个”按钮之前，请尝试同时选取多个选项。在点击“选择多个”按钮之后，请再试一次。</p>

</body>
</html>
```

HTML DOM name 属性

定义和用法

name 属性可设置或返回链接的名称。

语法

```
anchorObject.name=name
```

实例

下面的例子可返回链接的名称：

```
<html>
<body>

<p><a id="myAnchor" name="myAnchor"
href="http://www.w3school.com.cn">Visit W3School.com.cn</a></p>

<script type="text/javascript">
x=document.getElementById("myAnchor");
document.write(x.name);
</script>

</body>
</html>
```

TIY

取得链接的名称

```
<html>
<body>

<p><a id="myAnchor" name="myAnchor"
href="http://www.w3school.com.cn">Visit W3School.com.cn</a></p>

<script type="text/javascript">
x=document.getElementById("myAnchor");
document.write(x.name);
</script>

</body>
</html>
```

HTML DOM noHref 属性

定义和用法

noHref 属性可设置或返回某个区域是活动的还是非活动的。

语法

```
areaObject.noHref=true|false
```

说明

<area> 标签的 nohref 属性在客户端图像映射中定义了一个对鼠标敏感的区域，而该区域不会在用户选取它时采取任何操作。你必须为每个 <area> 标签都包含一个 href 或 nohref 属性。

实例

下面的例子可返回图像映射中 "Venus" 区域的 "noHref" 状态：

```
<html>
<body>

<img src="planets.gif"
width="145" height="126"
usemap="#planetmap" />

<map name="planetmap">
<area id="venus" shape="circle"
coords="124,58,8"
alt="Venus"
href="venus.htm" />
</map>

<p>noHref status:
<script type="text/javascript">
x=document.getElementById('venus');
document.write(x.noHref);
</script>
```

```
</p>

</script>
</body>
</html>
```

TIY

返回图像映射中某个区域的 noHref 状态

```
<html>
<body>

<img src="/i/eg_planets.jpg"
usemap="#planetmap" />

<map name="planetmap">
<area
id="venus"
shape="circle"
coords="180,139,14"
href ="/example/html/venus.html"
alt="Venus" />
</map>

<p>noHref 状态:
<script type="text/javascript">
x=document.getElementById('venus');
document.write(x.noHref);
</script>
</p>

</script>
</body>
</html>
```

HTML DOM pathname 属性

定义和用法

pathname 属性可设置或返回某个区域中 link-URL 的路径名。

语法

```
areaObject.pathname=pathnamee
```

实例

下面的例子可返回 "Venus" 区域中 link-URL 的路径名：

```
<html>
<body>

<img src="planets.gif"
width="145" height="126"
usemap="#planetmap" />

<map name="planetmap">
<area id="venus" shape="circle"
coords="124,58,8"
alt="Venus"
href="venus.htm" />
</map>

<p>Venus' pathname:
<script type="text/javascript">
x=document.getElementById('venus');
document.write(x.pathname);
</script>
</p>

</body>
</html>
```

TIY

取得图像映射中某个区域的 link-URL 的路径名

```
<html>
<body>

<img src="/i/eg_planets.jpg"
usemap="#planetmap" />

<map name="planetmap">
<area
id="venus"
shape="circle"
coords="180,139,14"
href ="/example/html/venus.html"
alt="Venus" />
</map>

<p>Venus 的路径:
<script type="text/javascript">
x=document.getElementById('venus');
document.write(x.pathname);
</script>
</p>

</body>
</html>
```

HTML DOM protocol 属性

定义和用法

protocol 属性可设置或返回某个区域中 link-URL 的协议。

语法

```
areaObject.protocol=protocol
```

实例

下面的例子可返回 "Venus" 区域中 link-URL 的协议：

```
<html>
<body>

<img src="planets.gif"
width="145" height="126"
usemap="#planetmap" />

<map name="planetmap">
<area id="venus" shape="circle"
coords="124,58,8"
alt="Venus"
href="venus.htm" />
</map>

<p>Venus' protocol:
<script type="text/javascript">
x=document.getElementById('venus');
document.write(x.protocol);
</script>
</p>

</body>
</html>
```

TIY

取得图像地图中某个区域的 link-URL 所用的协议

```
<html>
<body>

<img src="/i/eg_planets.jpg"
usemap="#planetmap" />

<map name="planetmap">
<area
id="venus"
shape="circle"
coords="180,139,14"
```

```
href ="/example/html/venus.html"
alt="Venus" />
</map>

<p>Venus 的协议：
<script type="text/javascript">
x=document.getElementById('venus');
document.write(x.protocol);
</script>
</p>

</body>
</html>
```

HTML DOM readOnly 属性

定义和用法

readOnly 属性可设置或返回密码域是否为只读。

语法

```
passwordObject.readOnly=true|false
```

实例

下面的例子把密码域设置为只读：

```
<html>
<head>
<script type="text/javascript">
function makeReadOnly()
  {
  document.getElementById("password1").readOnly=true
  }
</script>
</head>
<body>
```

```
<form>
<input type="password" id="password1" value="thgrt456" />
<input type="button" onclick="makeReadOnly()"
value="Make read-only" />
</form>

</body>
</html>
```

TIY

把密码域设置为只读

```
<html>
<head>
<script type="text/javascript">
function makeReadOnly()
  {
  document.getElementById("password1").readOnly=true
  }
</script>
</head>
<body>

<form>
<input type="password" id="password1" value="thgrt456" />
<input type="button" onclick="makeReadOnly()" value="Make read-only"
/>
</form>

</body>
</html>
```

HTML DOM rel 属性

定义和用法

rel 属性可设置或返回到目标文档的关系。
rel 和 rev 属性可以同时在 <a> 标签中使用。
rel (relationship) 属性规定了从源文档到目标文档的关系。rev (reverse) 属性规定了从目标文

档到当前文档的反向关系。
浏览器可以使用它们改变锚内容的外观，或者自动构建文档浏览菜单。其他工具也可以使用这些属性构建特殊的链接集合、目录和索引等。

语法

```
anchorObject.rel=relationship
```

两个属性均可采用下面的值：

值	描述
appendix	链接到文档的附录页。
alternate	链接到一个备选的源。
bookmark	链接到一个书签。在 web blogs 中常用于 "permalink"。
chapter	从当前文档链接到一个章节。
contents	链接到当前文档的内容目录。
copyright	链接到当前文档的版权或隐私页面。
glossary	链接到当前文档术语表。
index	链接到当前文档的索引。
next	链接到集合中的下一个文档。
prev	链接到集合中的前一个文档。
section	链接到文档列表中的一个小节。
start	链接到当前文档的第一页。
subsection	链接到当前文档列表中的子小节。（一个小节可拥有多个子小节。）
head	链接到集合中的顶级文档。
toc	链接到集合的目录。
parent	链接到源上面的文档。
child	链接到源下面的文档。

实例

下面的例子可返回链接的关系：

```
<html>
<body>

<p><a id="myAnchor" rel="index"
href="http://www.w3school.com.cn">Visit W3School.com.cn</a></p>

<script type="text/javascript">
x=document.getElementById("myAnchor");
document.write(x.rel);
</script>

</body>
</html>
```

TIY

取得链接的关系

```
<html>
<body>

<p><a id="myAnchor" rel="index"
href="http://www.w3school.com.cn">Visit W3School.com.cn</a></p>

<script type="text/javascript">
x=document.getElementById("myAnchor");
document.write(x.rel);
</script>

</body>
</html>
```

HTML DOM rev 属性

定义和用法

rev 属性可设置或返回一个链接的反向关系。
rel 和 rev 属性可以同时在 <a> 标签中使用。
rel (relationship) 属性规定了从源文档到目标文档的关系。rev (reverse) 属性规定了从目标文档到当前文档的反向关系。

浏览器可以使用它们改变锚内容的外观，或者自动构建文档浏览菜单。其他工具也可以使用这些属性构建特殊的链接集合、目录和索引等。

语法

```
anchorObject.rev=relationship
```

两个属性均可采用下面的值：

值	描述
appendix	链接到文档的附录页。
alternate	链接到一个备选的源。
bookmark	链接到一个书签。在 web blogs 中常用于 "permalink"。
chapter	从当前文档链接到一个章节。
contents	链接到当前文档的内容目录。
copyright	链接到当前文档的版权或隐私页面。
glossary	链接到当前文档术语表。
index	链接到当前文档的索引。
next	链接到集合中的下一个文档。
prev	链接到集合中的前一个文档。
section	链接到文档列表中的一个小节。
start	链接到当前文档的第一页。
subsection	链接到当前文档列表中的子小节。（一个小节可拥有多个子小节。）
head	链接到集合中的顶级文档。
toc	链接到集合的目录。
parent	链接到源上面的文档。
child	链接到源下面的文档。

实例

下面的例子可返回链接的反向关系：

```
<html>
```

```
<body>

<p><a id="myAnchor" rev="subsection"
href="/htmldom/prop_anchor_rev.asp">
HTML DOM rev Property</a></p>

<script type="text/javascript">
x=document.getElementById("myAnchor");
document.write(x.rev);
</script>

</body>
</html>
```

TIY

取得链接的反向关系

```
<html>
<body>

<p><a id="myAnchor" rev="subsection"
href="/htmldom/prop_anchor_rev.asp">
HTML DOM rev Property</a></p>

<script type="text/javascript">
x=document.getElementById("myAnchor");
document.write(x.rev);
</script>

</body>
</html>
```

HTML DOM rows 属性

定义和用法

rows 属性可设置或返回框架中行的数目和尺寸。
由逗号分割的像素或百分比列表定义了列的数目和高度：

语法

```
framesetObject.rows=row1,row2,row3....
```

实例

在我们的例子中，首先将创建包含带有两个列的框架集的 HTML 文档。每列设置为浏览器窗口的 50%：

```
<html>
<frameset rows="50%,50%">
  <frame src="frame_rows.htm">
  <frame src="frame_a.htm">
</frameset>
</html>
```

HTML 文档 "frame_rows.htm" 被放入第一列，而 HTML 文档 "frame_a.htm" 被放入第二列。
下面是 "frame_rows.htm" 的源代码：

```
<html>
<head>
<script type="text/javascript">
function changeRows()
  {
  parent.document.getElementById("main").rows="30%,70%"
  }

function restoreRows()
  {
  parent.document.getElementById("main").rows="50%,50%"
  }
</script>
</head>
<body>

<form>
<input type="button" onclick="changeRows()"
value="Change row size" />
<input type="button" onclick="restoreRows()"
value="Restore row size" />
</form>
```

```
</body>
</html>
```

TIY

更改框架集的行尺寸

```
<html>
<frameset id="main" rows="50%,50%">
  <frame id="leftFrame" src="/example/hdom/frame_rows.html">
  <frame id="rightFrame" src="/example/hdom/frame_a.html" >
</frameset>
</html>
```

HTML DOM scrolling 属性

定义和用法

scrolling 属性可设置或返回 iframe 是否拥有滚动条（滚动策略）。

语法

```
iframeObject.scrolling=auto|yes|no
```

实例

下面的例子可添加和删除 iframe 的滚动条：

```
<html>
<head>
<script type="text/javascript">
function addScrollbars()
  {
  document.getElementById("frame1").scrolling="yes";
  }
  function removeScrollbars()
  {
  document.getElementById("frame1").scrolling="no";
```

```
  }
</script>
</head>

<body>
<iframe src="frame_a.htm" id="frame1" ></iframe><br />

<input type="button" onclick="addScrollbars()"
value="Add Scrollbars" />
<input type="button" onclick="removeScrollbars()"
value="Remove Scrollbars" />

</body>
</html>
```

TIY

添加和删除 <iframe> 元素的滚动条

```
<html>
<head>
<script type="text/javascript">
function addScrollbars()
  {
  document.getElementById("frame1").scrolling="yes";
  }
function removeScrollbars()
  {
  document.getElementById("frame1").scrolling="no";
  }
</script>
</head>

<body>
<iframe src="/example/hdom/frame_a.html" id="frame1" ></iframe>
<br />
<input type="button" onclick="addScrollbars()" value="Add
Scrollbars" />
<input type="button" onclick="removeScrollbars()" value="Remove
Scrollbars" />
</body>
</html>
```

HTML DOM selectedIndex 属性

定义和用法

selectedIndex 属性可设置或返回下拉列表中被选选项的索引号。
注释：若允许多重选择，则仅会返回第一个被选选项的索引号。

语法

```
selectObject.selectedIndex=number
```

实例

下面的例子可提示出被选选项的索引号：

```
<html>
<head>
<script type="text/javascript">
function getIndex()
  {
  var x=document.getElementById("mySelect")
  alert(x.selectedIndex)
  }
</script>
</head>
<body>

<form>
Select your favorite fruit:
<select id="mySelect">
  <option>Apple</option>
  <option>Orange</option>
  <option>Pineapple</option>
  <option>Banana</option>
</select>
<br /><br />
<input type="button" onclick="getIndex()"
value="Alert index of selected option">
</form>
```

```
</body>
</html>
```

TIY

显示被选选项的索引号

```
<html>
<head>
<script type="text/javascript">
function getIndex()
  {
  var x=document.getElementById("mySelect")
  alert(x.selectedIndex)
  }
</script>
</head>
<body>

<form>
Select your favorite fruit:
<select id="mySelect">
  <option>Apple</option>
  <option>Orange</option>
  <option>Pineapple</option>
  <option>Banana</option>
</select>
<br /><br />
<input type="button" onclick="getIndex()" value="Alert index of
selected option">
</form>

</body>
</html>
```

设置下拉列表中被选选项的文本

```
<html>
<head>
<script type="text/javascript">
function changeText()
  {
  var x=document.getElementById("mySelect")
```

```
  x.options[x.selectedIndex].text="Melon"
  }
</script>
</head>
<body>

<form>
Select your favorite fruit:
<select id="mySelect">
  <option>Apple</option>
  <option>Orange</option>
  <option>Pineapple</option>
  <option>Banana</option>
</select>
<br /><br />
<input type="button" onclick="changeText()" value="Set text of
selected option">
</form>

</body>
</html>
```

从下拉列表中删除选项

```
<html>
<head>
<script type="text/javascript">
function removeOption()
  {
  var x=document.getElementById("mySelect")
  x.remove(x.selectedIndex)
  }
</script>
</head>
<body>

<form>
<select id="mySelect">
  <option>苹果</option>
  <option>桃子</option>
  <option>香蕉</option>
  <option>桔子</option>
```

```
</select>
<input type="button" onclick="removeOption()" value="删除被选的选项">
</form>

</body>
</html>
```

HTML DOM shape 属性

定义和用法

shape 属性可设置或返回图像映射中链接的形状。

语法

```
anchorObject.shape=RECT|CIRCLE|POLY|DEFAULT
```

实例

下面的例子可返回图像映射中链接的形状：

```
<html>
<body>

<img src="planets.gif"
width="145" height="126"
usemap="#planetmap" />

<map name="planetmap">
<area id="venus" shape="circle"
coords="124,58,8"
alt="Venus"
href="venus.htm" />
</map>

<p>Shape of Venus-link:
<script type="text/javascript">
x=document.getElementById("venus");
document.write(x.shape);
```

```
</script>
</p>

</body>
</html>
```

TIY

取得图像映射中链接的形状

```
<html>
<body>

<img src="/i/eg_planets.jpg"
usemap="#planetmap" />

<map name="planetmap">
<area id="venus" shape="circle"
coords="180,139,14"
alt="Venus"
href="/example/hdom/venus.html" />
</map>

<p>Venus area 的 shape 属性是:
<script type="text/javascript">
x=document.getElementById("venus");
document.write(x.shape);
</script>
</p>

</body>
</html>
```

HTML DOM size 属性

定义和用法

size 属性可设置或返回密码域的大小（以字符数计）。

语法

```
passwordObject.size=number_of_chars
```

实例

下面的例子更改了密码域的大小：

```
<html>
<head>
<script type="text/javascript">
function changeSize()
  {
  document.getElementById("password1").size=40
  }
</script>
</head>
<body>

<form>
<input type="password" id="password1" />
<input type="button" id="button1" onclick="changeSize()"
value="Change size of the password field" />
</form>

</body>
</html>
```

TIY

更改密码域的尺寸

```
<html>
<head>
<script type="text/javascript">
function changeSize()
  {
  document.getElementById("password1").size=40
  }
</script>
</head>
<body>
```

```
<form>
<input type="password" id="password1" />
<input type="button" id="button1" onclick="changeSize()"
value="Change size of the password field" />
</form>

</body>
</html>
```

HTML DOM src 属性

定义和用法

src 属性可设置或返回被载入 iframe 中的文档的 URL。

语法

```
iframeObject.src=URL
```

实例

下面的例子可更改两个框架的源：

```
<html>
<head>
<script type="text/javascript">
function changeSource()
  {
  document.getElementById("frame1").src="frame_c.htm"
  document.getElementById("frame2").src="frame_d.htm"
  }
</script>
</head>
<body>

<iframe src="frame_a.htm" id="frame1"></iframe>
<iframe src="frame_b.htm" id="frame2"></iframe>
<br /><br />
```

```
<input type="button" onclick="changeSource()"
value="Change source of the two iframes" />

</body>
</html>
```

TIY

更改两个框架的源

```
<html>
<head>
<script type="text/javascript">
function changeSource()
{
document.getElementById("frame1").src="/example/hdom/frame_c.html
"
document.getElementById("frame2").src="/example/hdom/frame_d.html
"
}
</script>
</head>

<body>
<iframe src="/example/hdom/frame_a.html" id="frame1"></iframe>
<iframe src="/example/hdom/frame_b.html" id="frame2"></iframe>
<br /><br />
<input type="button" onclick="changeSource()" value="改变两个框架的
source">

</body>
</html>
```

HTML DOM tabIndex 属性

定义和用法

tabIndex 属性可为链接设置或返回 tab 键控制次序。

语法

```
anchorObject.tabIndex=tabIndex
```

实例

下面的例子可更改三个链接的 tab 键控制次序：

```
<html>
<head>
<script type="text/javascript">
function changeTabIndex()
  {
  document.getElementById('1').tabIndex="3"
  document.getElementById('2').tabIndex="2"
  document.getElementById('3').tabIndex="1"
  }
</script>
</head>

<body>
<p><a id="1" href="http://www.w3school.com.cn">1</a></p>
<p><a id="2" href="http://www.w3school.com.cn">2</a></p>
<p><a id="3" href="http://www.w3school.com.cn">3</a></p>

<input type="button" onclick="changeTabIndex()"
value="Change TabIndex" />

</body>
</html>
```

TIY

为链接设置 tab 键控制次序

```
<html>
<head>
<script type="text/javascript">
function changeTabIndex()
{
document.getElementById('1').tabIndex="3"
document.getElementById('2').tabIndex="2"
```

```
document.getElementById('3').tabIndex="1"
}
</script>
</head>
<body>
<p><a id="1" href="http://www.w3school.com.cn">1</a></p>
<p><a id="2" href="http://www.w3school.com.cn">2</a></p>
<p><a id="3" href="http://www.w3school.com.cn">3</a></p>

<input type="button" onclick="changeTabIndex()"
value="Change TabIndex" />

<p>Try navigating the links by using the "Tab" button on you keyboard
before and after pushing the "Change TabIndex" button.</p>

</body>
</html>
```

HTML DOM target 属性

定义和用法

target 属性可设置或返回在何处打开链接。

语法

```
anchorObject.target=_blank|_parent|_self|_top
```

四个保留的目标名称：

- _blank - 在一个新的未命名的窗口载入文档
- _self - 在相同的框架或窗口中载入目标文档
- _parent - 把文档载入父窗口或包含了超链接引用的框架的框架集
- _top - 把文档载入包含该超链接的窗口,取代任何当前正在窗口中显示的框架

实例

下面的例子将更改一个链接的文本、URL 以及 target：

```
<html>
```

```
<head>
<script type="text/javascript">
function changeLink()
{
document.getElementById('myAnchor').innerHTML="W3School";
document.getElementById('myAnchor').href="http://www.w3school.com.
cn";
document.getElementById('myAnchor').target="_blank";
}
</script>
</head>

<body>
<a id="myAnchor" href="http://www.microsoft.com">Microsoft</a>
<input type="button" onclick="changeLink()" value="Change link">
</body>

</html>
```

TIY

更改一个链接的文本、URL 以及 target

```
<html>
<head>
<script type="text/javascript">
function changeLink()
{
document.getElementById('myAnchor').innerHTML="访问 W3School"
document.getElementById('myAnchor').href="http://www.w3school.com.
cn"
document.getElementById('myAnchor').target="_blank"
}
</script>
</head>

<body>
<a id="myAnchor" href="http://www.microsoft.com">访问 Microsoft</a>
<input type="button" onclick="changeLink()" value="改变链接">
<p>在本例中，我们改变超链接的文本和 URL。我们也改变 target 属性。target 属性
的默认设置是 "_self"，这意味着会在相同的窗口中打开链接。通过把 target 属性设
置为 "_blank"，链接将在新窗口中打开。</p>
</body>
```

```
</html>
```

HTML DOM text 属性

定义和用法

text 属性可设置或返回选项的文本值。
该属性设置或返回的是包含在 <option> 元素中的纯文本。这个文本会作为选项的标签出现。

语法

```
optionObject.text=sometext
```

实例

下面的例子返回下拉列表中所有选项的文本：

```
<html>
<head>
<script type="text/javascript">
function getOptions()
  {
  var x=document.getElementById("mySelect");
  var y="";
  for (i=0;i<x.length;i++)
    {
    y+=x.options[i].text;
    y+="<br />";
    }
  document.write(y);
  }
</script>
</head>

<body>

<form>
```

```
请选择您喜欢的水果：
<select id="mySelect">
  <option>苹果</option>
  <option>桃子</option>
  <option>香蕉</option>
  <option>桔子</option>
</select>
<br /><br />
<input type="button" onclick="getOptions()" value="输出所有选项">
</form>

</body>
</html>
```

TIY

输出下拉列表中所有选项的文本

```
<html>
<head>
<script type="text/javascript">
function getOptions()
  {
  var x=document.getElementById("mySelect");
  var y="";
  for (i=0;i<x.length;i++)
    {
    y+=x.options[i].text;
    y+="<br />";
    }
  document.write(y);
  }
</script>
</head>

<body>

<form>
请选择您喜欢的水果：
<select id="mySelect">
  <option>苹果</option>
  <option>桃子</option>
  <option>香蕉</option>
```

```
  <option>桔子</option>
</select>
<br /><br />
<input type="button" onclick="getOptions()" value="输出所有选项">
</form>

</body>
</html>
```

设置下拉列表中被选选项的文本

```
<html>
<head>
<script type="text/javascript">
function changeText()
  {
  var x=document.getElementById("mySelect")
  x.options[x.selectedIndex].text="Melon"
  }
</script>
</head>
<body>

<form>
Select your favorite fruit:
<select id="mySelect">
  <option>Apple</option>
  <option>Orange</option>
  <option>Pineapple</option>
  <option>Banana</option>
</select>
<br /><br />
<input type="button" onclick="changeText()" value="Set text of
selected option">
</form>

</body>
</html>
```

HTML DOM type 属性

定义和用法

type 属性可设置或返回被链资源的 MIME 类型。

语法

```
anchorObject.type=type
```

实例

下面的例子可返回被链资源的 MIME 类型：

```
<html>
<body>

<p><a id="myAnchor" type="text/html"
href="http://www.w3school.com.cn">W3School.com.cn</a></p>

<script type="text/javascript">
x=document.getElementById("myAnchor");
document.write(x.type);
</script>

</body>
</html>
```

TIY

取得链接的 MIME 类型

```
<html>
<body>

<p><a id="myAnchor" type="text/html"
href="http://www.w3school.com.cn">W3School.com.cn</a></p>

<script type="text/javascript">
```

```
x=document.getElementById("myAnchor");
document.write(x.type);
</script>

</body>
</html>
```

HTML DOM useMap 属性

定义和用法

useMap 属性可设置或返回图像的 usemap 属性的值。

语法

```
imageObject.useMap=URL
```

实例

下面的例子可输出客户端图像映射的 usemap 属性的值：

```
<html>
<body>

<img id="planets" src="planets.gif"
width="145" height="126"
usemap="#planetmap" />

<map name="planetmap">
<area id="venus" shape="circle"
coords="124,58,8"
alt="The planet Venus"
href="venus.htm" />
</map>

<p>The value of the usemap attribute=
<script type="text/javascript">
x=document.getElementById('planets');
document.write(x.useMap);
```

```
</script>
</p>

</body>
</html>
```

TIY

返回客户端图像映射的 usemap 属性的值

```
<html>
<body>

<img id="planets"
src="/i/eg_planets.jpg"
usemap="#planetmap" />

<map name="planetmap">
<area
shape="circle"
coords="180,139,14"
href ="/example/html/venus.html"
target ="_blank"
alt="Venus" />
</map>

<p>usemap 属性的值等于 =
<script type="text/javascript">
x=document.getElementById('planets');
document.write(x.useMap);
</script>
</p>

</body>
</html>
```

HTML DOM value 属性

定义和用法

value 属性设置或返回显示在按钮上的文本。

语法

```
buttonObject.value=value
```

实例

下面的例子可返回显示在按钮上的文本：

```
<html>
<body>

<input type="button" id="button1" name="button1" value="Click Me!"
/>

<p>The text on the button is:
<script type="text/javascript">
x=document.getElementById('button1');
document.write(x.value);
</script></p>

</body>
</html>
```

TIY

获取按钮上的文本

```
<html>
<body>

<input type="button" id="button1"
name="button1" value="Click Me!" />
```

```
<p>The text on the button is:
<script type="text/javascript">
x=document.getElementById('button1');
document.write(x.value);
</script></p>

</body>
</html>
```

HTML DOM vspace 属性

定义和用法

vspace 属性设置或返回图像的顶部和底部的空白。
Thspace 和 vspace 属性可与 align 一同使用，来设置图像与周围文本的距离。

语法

```
imageObject.vspace=pixels
```

实例

下面的例子将设置图像的 hspace 和 vspace 属性：

```
<html>
<head>
<script type="text/javascript">
function setSpace()
{
document.getElementById("compman").hspace="50"
document.getElementById("compman").vspace="50"
}
</script>
</head>
<body>

<img id="compman" src="compman.gif" alt="Computerman" align="right"
/>
```

```
<p>Some text. Some text. Some text. Some text.</p>

<input type="button" onclick="setSpace()"
value="Set hspace and vspace">

</body>
</html>
```

TIY

设置图像的水平和垂直定位

```
<html>
<head>
<script type="text/javascript">
function setSpace()
  {
  document.getElementById("compman").hspace="50"
  document.getElementById("compman").vspace="50"
  }
</script>
</head>
<body>

<img id="compman" src="/i/eg_compman.gif" alt="Computerman"
align="right" />
<p>Some text. Some text. Some text. Some text.</p>
<input type="button" onclick="setSpace()" value="Set hspace and
vspace">

</body>
</html>
```

HTML DOM width 属性

定义和用法

width 属性可设置或返回图像的宽度。

语法

```
imageObject.width=pixels
```

实例

下面的例子可设置图像的高度和宽度：

```
<html>
<head>
<script type="text/javascript">
function changeSize()
{
document.getElementById("compman").height="250"
document.getElementById("compman").width="300"
}
</script>
</head>

<body>
<img id="compman" src="compman.gif" width="107" height="98" />
<br /><br />
<input type="button" onclick="changeSize()"
value="Change size of image">
</body>

</html>
```

TIY

更改图像的高度和宽度

```
<html>
<head>
<script type="text/javascript">
function changeSize()
{
document.getElementById("compman").height="270"
document.getElementById("compman").width="164"
}
</script>
</head>
```

```
<body>
<img id="compman" src="/i/eg_bulbon.gif" />
<br /><br />
<input type="button" onclick="changeSize()" value="改变图像的高度和宽度">
</body>

</html>
```

HTML DOM className 属性

定义和用法

className 属性设置或返回元素的 class 属性。

语法

```
object.className=classname
```

实例

本例展示了两种获得 <body> 元素的 class 属性的方法：

```
<html>
<body id="myid" class="mystyle">

<script type="text/javascript">
x=document.getElementsByTagName('body')[0];
document.write("Body CSS class: " + x.className);
document.write("<br />");
document.write("An alternate way: ");
document.write(document.getElementById('myid').className);
</script>

</body>
</html>
```

输出：

```
Body CSS class: mystyle
An alternate way: mystyle
```

TIY

返回 <body> 元素的 class 属性

```
<html>
<body id="myid" class="mystyle">

<script type="text/javascript">
x=document.getElementsByTagName('body')[0];
document.write("Body CSS class: " + x.className);
document.write("<br />");
document.write("An alternate way: ");
document.write(document.getElementById('myid').className);
</script>

</body>
</html>
```

HTML DOM dir 属性

定义和用法

dir 属性设置或返回元素的文字方向。

语法：

```
object.dir=text-direction
```

实例

本例展示了两种获取 <body> 元素的文本方向的方法：

```
<html>
<body id="myid" dir="rtl">
```

```
<script type="text/javascript">
x=document.getElementsByTagName('body')[0];
document.write("Text direction: " + x.dir);
document.write("<br />");
document.write("An alternate way: ");
document.write(document.getElementById('myid').dir);
</script>

</body>
</html>
```

TIY

<body> 元素的文本方向

```
<html>
<body id="myid" dir="rtl">

<script type="text/javascript">
x=document.getElementsByTagName('body')[0];
document.write("Text direction: " + x.dir);
document.write("<br />");
document.write("An alternate way: ");
document.write(document.getElementById('myid').dir);
</script>

</body>
</html>
```

HTML DOM lang 属性

定义和用法

lang 属性设置或返回元素的语言。

语法：

```
object.lang=language-code
```

实例

本例展示了两种获取 <body> 元素的语言的方法：

```
<html>
<body id="myid" lang="en-us">

<script type="text/javascript">
x=document.getElementsByTagName('body')[0];
document.write("Body language: " + x.lang);
document.write("<br />");
document.write("An alternate way: ");
document.write(document.getElementById('myid').lang);
</script>

</body>
</html>
```

TIY

<body> 元素的语言代码

```
<html>
<body id="myid" lang="en-us">

<script type="text/javascript">
x=document.getElementsByTagName('body')[0];
document.write("Body language: " + x.lang);
document.write("<br />");
document.write("An alternate way: ");
document.write(document.getElementById('myid').lang);
</script>

</body>
</html>
```

HTML DOM title 属性

定义和用法

title 属性设置或返回元素的（咨询性质的）标题。

语法：

```
object.title=title
```

实例 1

本例展示了两种获取 <body> 元素的 title 属性的方法：

```
<html>
<body id="myid" title="mytitle">

<script type="text/javascript">
x=document.getElementsByTagName('body')[0];
document.write("Body title: " + x.title);
document.write("<br />");
document.write("An alternate way: ");
document.write(document.getElementById('myid').title);
</script>

</body>
</html>
```

实例 2

本例返回图像映射中某个区域的 title：

```
<html>
<body>

<img src ="planets.gif"
width="145" height="126"
alt="Planets"
usemap ="#planetmap" />
```

```
<map name ="planetmap">
<area id="venus" shape="circle"
coords ="124,58,8"
title="Venus"
href ="venus.htm"  />
</map>

<p>Venus' advisory title (mouse over the Venus planet):
<script type="text/javascript">
x=document.getElementById('venus')
document.write(x.title)
</script>
</p>

</body>
</html>
```

TIY

返回 <body> 元素的 title 属性

```
<html>
<body id="myid" title="mytitle">

<script type="text/javascript">
x=document.getElementsByTagName('body')[0];
document.write("Body title: " + x.title);
document.write("<br />");
document.write("An alternate way: ");
document.write(document.getElementById('myid').title);
</script>

</body>
</html>
```

返回图像映射中某个区域的 title

```
<html>
<body>

<img id="planets"
src="/i/eg_planets.jpg"
```

```
usemap="#planetmap" />

<map name="planetmap">
<area id="venus" shape="circle"
coords="180,139,14"
href ="/example/html/venus.html"
target ="_blank"
title="Venus" />
</map>
</map>

<p>Venus 的 title（即把鼠标移动到火星上显示的文本）是：
<script type="text/javascript">
x=document.getElementById('venus')
document.write(x.title)
</script>
</p>

</body>
</html>
```

HTML DOM blur() 方法

定义和用法

blur() 方法用于从链接上移开焦点。

语法

```
anchorObject.blur()
```

实例

```
<html>
<head>
<style type="text/css">
a:active {color:green}
</style>
```

```
<script type="text/javascript">
function getfocus()
{document.getElementById('myAnchor').focus()}

function losefocus()
{document.getElementById('myAnchor').blur()}
</script>
</head>

<body>
<a id="myAnchor"
href="http://www.w3school.com.cn">Visit W3School.com.cn</a>
<br /><br/>
<input type="button" onclick="getfocus()" value="Get focus">
<input type="button" onclick="losefocus()" value="Lose focus">
</body>

</html>
```

TIY

使用 focus() 和 blur()

```
<html>
<head>
<style type="text/css">
a:active {color:green}
</style>

<script type="text/javascript">
function getfocus()
{document.getElementById('myAnchor').focus()}

function losefocus()
{document.getElementById('myAnchor').blur()}
</script>
</head>

<body>
<a id="myAnchor" href="http://www.w3school.com.cn">访问
W3School.com.cn</a>
<br /><br/>
<input type="button" onclick="getfocus()" value="获得焦点">
```

```
<input type="button" onclick="losefocus()" value="失去焦点">
</body>

</html>
```

HTML DOM click() 方法

定义和用法

click() 方法可模拟在按钮上的一次鼠标单击。

语法

```
buttonObject.click()
```

实例

下面的例子将在 body onload 上模拟在按钮上的一次鼠标单击：

```
<html>
<head>
<script type="text/javascript">
function clickButton()
  {
  document.getElementById('button1').click()
  }
function alertMsg()
  {
  alert("Button 1 was clicked!")
  }
</script>
</head>
<body onload="clickButton()">

<form>
<input type="button" id="button1" onclick="alertMsg()"
value="Button 1" />
</form>
```

```
</body>
</html>
```

TIY

模拟在按钮上的一次鼠标单击

```
<html>
<head>
<script type="text/javascript">
function clickButton()
  {
  document.getElementById('button1').click()
  }
function alertMsg()
  {
  alert("Button 1 was clicked!")
  }
</script>
</head>
<body onload="clickButton()">

<form>
<input type="button" id="button1" onclick="alertMsg()" value="Button
1" />
</form>

</body>
</html>
```

HTML DOM focus() 方法

定义和用法

focus() 方法用于向按钮赋予焦点。

语法

```
buttonObject.focus()
```

实例

下面的例子将在按钮上设置焦点：

```
<html>
<head>
<script type="text/javascript">
function setFocus()
  {
  document.getElementById('button1').focus()
  }
</script>
</head>
<body>

<form>
<input type="button" id="button1" value="Button1" />
<br />
<input type="button" onclick="setFocus()"
value="Set focus to Button 1" />
</form>

</body>
</html>
```

TIY

设置并从按钮上移开焦点

```
<html>
<head>
<script type="text/javascript">
function setFocus()
  {
  document.getElementById('button1').focus()
  }
function removeFocus()
  {
  document.getElementById('button1').blur()
  }
</script>
</head>
<body>
```

```
<form>
<input type="button" id="button1" value="Button1" />
<br /><br />
<input type="button" onclick="setFocus()" value="Set focus" />
<input type="button" onclick="removeFocus()" value="Remove focus" />
</form>

</body>
</html>
```

HTML DOM reset() 方法

定义和用法

reset() 方法可把表单中的元素重置为它们的默认值。

语法

```
formObject.reset()
```

说明

调用该方法的结果类似用户单击了 Reset 按钮的结果，只是表单的事件句柄 onreset 不会被调用。

实例

```
<html>
<head>
<script type="text/javascript">
function formReset()
  {
  document.getElementById("myForm").reset()
  }
</script>
</head>

<form id="myForm">
```

```
Name: <input type="text" size="20"><br />
Age: <input type="text" size="20"><br />
<br />
<input type="button" onclick="formReset()" value="Reset">
</form>
</body>

</html>
```

TIY

重置表单

```
<html>
<head>
<script type="text/javascript">
function formReset()
{
document.getElementById("myForm").reset()
}
</script>
</head>

<body>
<p>在下面的文本框中输入一些文本，然后点击重置按钮就可以重置表单。</p>

<form id="myForm">
姓名：<input type="text" size="20"><br />
年龄：<input type="text" size="20"><br />
<br />
<input type="button" onclick="formReset()" value="重置">
</form>
</body>

</html>
```

HTML DOM submit() 方法

定义和用法

submit() 方法把表单数据提交到 Web 服务器。

语法

```
formObject.submit()
```

说明

该方法提交表单的方式与用户单击 Submit 按钮一样，但是表单的 onsubmit 事件句柄不会被调用。

实例

```
<html>
<head>
<script type="text/javascript">
function formSubmit()
  {
  document.getElementById("myForm").submit()
  }
</script>
</head>

<body>

<form id="myForm" action="js_form_action.asp" method="get">
Firstname: <input type="text" name="firstname" size="20"><br />
Lastname: <input type="text" name="lastname" size="20"><br />
<br />
<input type="button" onclick="formSubmit()" value="Submit">
</form>
</body>

</html>
```

TIY

提交表单

```
<html>
<head>
<script type="text/javascript">
function formSubmit()
{
document.getElementById("myForm").submit()
}
</script>
</head>

<body>
<p>在下面的文本框中输入一些文本，然后点击提交按钮就可以提交表单。</p>

<form id="myForm" action="/i/eg_smile.gif" method="get">
名: <input type="text" name="firstname" size="20"><br />
姓: <input type="text" name="lastname" size="20"><br />
<br />
<input type="button" onclick="formSubmit()" value="提交">
</form>
</body>

</html>
```

HTML DOM Event 对象

实例

哪个鼠标按钮被点击？

```
<html>
<head>
<script type="text/javascript">
function whichButton(event)
{
var btnNum = event.button;
if (btnNum==2)
```

```
  {
  alert("您点击了鼠标右键！")
  }
else if(btnNum==0)
  {
  alert("您点击了鼠标左键！")
  }
else if(btnNum==1)
  {
  alert("您点击了鼠标中键！");
  }
else
  {
  alert("您点击了" + btnNum+ "号键，我不能确定它的名称。");
  }
}
</script>
</head>

<body onmousedown="whichButton(event)">
<p>请在文档中点击鼠标。一个消息框会提示出您点击了哪个鼠标按键。</p>
</body>

</html>
```

光标的坐标是？

```
<html>
<head>
<script type="text/javascript">
function show_coords(event)
{
x=event.clientX
y=event.clientY
alert("X 坐标: " + x + ", Y 坐标: " + y)
}
</script>
</head>

<body onmousedown="show_coords(event)">

<p>请在文档中点击。一个消息框会提示出鼠标指针的 x 和 y 坐标。</p>

</body>
```

```
</html>
```

被按的按键的 unicode 是？

```
<html>
<head>
<script type="text/javascript">
function whichButton(event)
{
alert(event.keyCode)
}

</script>
</head>

<body onkeyup="whichButton(event)">
<p><b>注释：</b>在测试这个例子时，要确保右侧的框架获得了焦点。</p>
<p>在键盘上按一个键。消息框会提示出该按键的 unicode。</p>
</body>

</html>
```

相对于屏幕，光标的坐标是？

```
<html>
<head>

<script type="text/javascript">
function coordinates(event)
{
x=event.screenX
y=event.screenY
alert("X=" + x + " Y=" + y)
}

</script>
</head>
<body onmousedown="coordinates(event)">

<p>
在文档中点击某个位置。消息框会提示出指针相对于屏幕的 x 和 y 坐标。
</p>

</body>
```

```
</html>
```

shift 键被按了吗？

```
<html>
<head>
<script type="text/javascript">
function isKeyPressed(event)
{
  if (event.shiftKey==1)
    {
    alert("The shift key was pressed!")
    }
  else
    {
    alert("The shift key was NOT pressed!")
    }
  }
</script>
</head>

<body onmousedown="isKeyPressed(event)">

<p>在文档中点击某个位置。消息框会告诉你是否按下了 shift 键。</p>

</body>
</html>
```

哪个元素被点击了？

```
<html>
<head>
<script type="text/javascript">
function whichElement(e)
{
var targ
if (!e) var e = window.event
if (e.target) targ = e.target
else if (e.srcElement) targ = e.srcElement
if (targ.nodeType == 3) // defeat Safari bug
   targ = targ.parentNode
var tname
tname=targ.tagName
alert("You clicked on a " + tname + " element.")
```

```
}
</script>
</head>

<body onmousedown="whichElement(event)">
<p>在文档中点击某个位置。消息框会提示出您所点击的标签的名称。</p>

<h3>这是标题</h3>
<p>这是段落。</p>
<img src="/i/eg_mouse2.jpg" />
</body>

</html>
```

哪个事件类型发生了？

```
<html>
<head>
<script type="text/javascript">
function getEventType(event)
  {
  alert(event.type);
  }
</script>
</head>

<body onmousedown="getEventType(event)">

<p>在文档中点击某个位置。消息框会提示出被触发的事件的类型。</p>

</body>
</html>
```

Event 对象

Event 对象代表事件的状态，比如事件在其中发生的元素、键盘按键的状态、鼠标的位置、鼠标按钮的状态。
事件通常与函数结合使用，函数不会在事件发生前被执行！
IE: Internet Explorer, **F:** Firefox, **O:** Opera, **W3C:** W3C 标准.

事件句柄 (Event Handlers)

HTML 4.0 的新特性之一是能够使 HTML 事件触发浏览器中的行为，比如当用户点击某个

HTML 元素时启动一段 JavaScript。下面是一个属性列表，可将之插入 HTML 标签以定义事件的行为。

属性	此事件发生在何时...	IE	F	O	W3C
onabort	图像的加载被中断。	4	1	9	Yes
onblur	元素失去焦点。	3	1	9	Yes
onchange	域的内容被改变。	3	1	9	Yes
onclick	当用户点击某个对象时调用的事件句柄。	3	1	9	Yes
ondblclick	当用户双击某个对象时调用的事件句柄。	4	1	9	Yes
onerror	在加载文档或图像时发生错误。	4	1	9	Yes
onfocus	元素获得焦点。	3	1	9	Yes
onkeydown	某个键盘按键被按下。	3	1	No	Yes
onkeypress	某个键盘按键被按下并松开。	3	1	9	Yes
onkeyup	某个键盘按键被松开。	3	1	9	Yes
onload	一张页面或一幅图像完成加载。	3	1	9	Yes
onmousedown	鼠标按钮被按下。	4	1	9	Yes
onmousemove	鼠标被移动。	3	1	9	Yes
onmouseout	鼠标从某元素移开。	4	1	9	Yes
onmouseover	鼠标移到某元素之上。	3	1	9	Yes
onmouseup	鼠标按键被松开。	4	1	9	Yes
onreset	重置按钮被点击。	4	1	9	Yes
onresize	窗口或框架被重新调整大小。	4	1	9	Yes
onselect	文本被选中。	3	1	9	Yes
onsubmit	确认按钮被点击。	3	1	9	Yes
onunload	用户退出页面。	3	1	9	Yes

鼠标 / 键盘属性

属性	描述	IE	F	O	W3C
altKey	返回当事件被触发时，"ALT" 是否被按下。	6	1	9	Yes

button	返回当事件被触发时，哪个鼠标按钮被点击。	6	1	9	Yes
clientX	返回当事件被触发时，鼠标指针的水平坐标。	6	1	9	Yes
clientY	返回当事件被触发时，鼠标指针的垂直坐标。	6	1	9	Yes
ctrlKey	返回当事件被触发时，"CTRL" 键是否被按下。	6	1	9	Yes
metaKey	返回当事件被触发时，"meta" 键是否被按下。	No	1	9	Yes
relatedTarget	返回与事件的目标节点相关的节点。	No	1	9	Yes
screenX	返回当某个事件被触发时，鼠标指针的水平坐标。	6	1	9	Yes
screenY	返回当某个事件被触发时，鼠标指针的垂直坐标。	6	1	9	Yes
shiftKey	返回当事件被触发时，"SHIFT" 键是否被按下。	6	1	9	Yes

IE 属性

除了上面的鼠标/事件属性，IE 浏览器还支持下面的属性：

属性	描述
cancelBubble	如果事件句柄想阻止事件传播到包容对象，必须把该属性设为 true。
fromElement	对于 mouseover 和 mouseout 事件，fromElement 引用移出鼠标的元素。
keyCode	对于 keypress 事件，该属性声明了被敲击的键生成的 Unicode 字符码。对于 keydown 和 keyup 事件，它指定了被敲击的键的虚拟键盘码。虚拟键盘码可能和使用的键盘的布局相关。
offsetX,offsetY	发生事件的地点在事件源元素的坐标系统中的 x 坐标和 y 坐标。
returnValue	如果设置了该属性，它的值比事件句柄的返回值优先级高。把这个属性设置为 fasle，可以取消发生事件的源元素的默认动作。
srcElement	对于生成事件的 Window 对象、Document 对象或 Element 对象的引用。
toElement	对于 mouseover 和 mouseout 事件，该属性引用移入鼠标的元素。
x,y	事件发生的位置的 x 坐标和 y 坐标，它们相对于用 CSS 动态定位的最内层包容元素。

标准 Event 属性

下面列出了 2 级 DOM 事件标准定义的属性。

属性	描述	IE	F	O	W3C

bubbles	返回布尔值，指示事件是否是起泡事件类型。	No	1	9	Yes
cancelable	返回布尔值，指示事件是否可拥可取消的默认动作。	No	1	9	Yes
currentTarget	返回其事件监听器触发该事件的元素。	No	1	9	Yes
eventPhase	返回事件传播的当前阶段。				Yes
target	返回触发此事件的元素（事件的目标节点）。	No	1	9	Yes
timeStamp	返回事件生成的日期和时间。	No	1	9	Yes
type	返回当前 Event 对象表示的事件的名称。	6	1	9	Yes

标准 Event 方法

下面列出了 2 级 DOM 事件标准定义的方法。IE 的事件模型不支持这些方法：

方法	描述	IE	F	O	W3C
initEvent()	初始化新创建的 Event 对象的属性。	No	1	9	Yes
preventDefault()	通知浏览器不要执行与事件关联的默认动作。	No	1	9	Yes
stopPropagation()	不再派发事件。	No	1	9	Yes

onabort 事件

定义和用法

onabort 事件会在图像加载被中断时发生。
当用户在图像完成载入之前放弃图像的装载（如单击了 stop 按钮）时，就会调用该句柄。

语法

```
onabort="SomeJavaScriptCode"
```

参数	描述
SomeJavaScriptCode	必需。规定该事件发生时执行的 JavaScript。

支持该事件的 HTML 标签：

```
<img>
```

支持该事件的 JavaScript 对象：

```
image
```

实例 1

在本例中，如果图像的加载被中断，则会显示一个对话框：

```
<img src="image_w3default.gif"
onabort="alert('Error: Loading of the image was aborted')" />
```

实例 2

在本例中，如果图像的加载中断，我们将调用一个函数：

```
<html>
<head>
<script type="text/javascript">
function abortImage()
{
alert('Error: Loading of the image was aborted')
}
</script>
</head>

<body>
<img src="image_w3default.gif" onabort="abortImage()" />
</body>

</html>
```

onblur 事件

定义和用法

onblur 事件会在对象失去焦点时发生。

语法

```
onblur="SomeJavaScriptCode"
```

参数	描述
SomeJavaScriptCode	必需。规定该事件发生时执行的 JavaScript。

支持该事件的 HTML 标签：

```
<a>, <acronym>, <address>, <area>, <b>, <bdo>, <big>, <blockquote>,
<button>, <caption>, <cite>, <dd>, <del>, <dfn>, <div>, <dl>, <dt>,
<em>, <fieldset>, <form>, <frame>, <frameset>, <h1> to <h6>, <hr>,
<i>, <iframe>, <img>, <input>, <ins>, <kbd>, <label>, <legend>, <li>,
<object>, <ol>, <p>, <pre>, <q>, <samp>, <select>, <small>, <span>,
<strong>, <sub>, <sup>, <table>, <tbody>, <td>, <textarea>, <tfoot>,
<th>, <thead>, <tr>, <tt>, <ul>, <var>
```

支持该事件的 JavaScript 对象：

```
button, checkbox, fileUpload, layer, frame, password,
radio, reset, submit, text, textarea, window
```

实例 1

在本例中，我们将在用户离开输入框时执行 JavaScript 代码：

```
<html>
<head>
<script type="text/javascript">
function upperCase()
{
```

```
var x=document.getElementById("fname").value
document.getElementById("fname").value=x.toUpperCase()
}
</script>
</head>

<body>

输入您的姓名：
<input type="text" id="fname" onblur="upperCase()" />

</body>
</html>
```

TIY

onblur

如何使用 onblur 事件在用户离开输入框时执行 JavaScript 代码

```
<html>

<head>
<script type="text/javascript">
function upperCase()
{
var x=document.getElementById("fname").value
document.getElementById("fname").value=x.toUpperCase()
}
</script>
</head>

<body>

Enter your name: <input type="text" id="fname"
onblur="upperCase()"><br />
Enter your age: <input type="text" id="age" onblur="alert(this.id)">

</body>
</html>
```

onchange 事件

定义和用法

onchange 事件会在域的内容改变时发生。

语法

```
onchange="SomeJavaScriptCode"
```

参数	描述
SomeJavaScriptCode	必需。规定该事件发生时执行的 JavaScript。

支持该事件的 HTML 标签：

```
<input type="text">, <select>, <textarea>
```

支持该事件的 JavaScript 对象：

```
fileUpload, select, text, textarea
```

实例 1

在本例中，我们将在用户改变输入域内容时执行 JavaScript 代码：

```
<html>
<head>
<script type="text/javascript">
function upperCase(x)
{
var y=document.getElementById(x).value
document.getElementById(x).value=y.toUpperCase()
}
</script>
</head>

<body>
```

```
输入您的姓名：
<input type="text" id="fname" onchange="upperCase(this.id)" />

</body>
</html>
```

TIY

onchange

如何使用 onchange 事件在用户改变输入域的内容时执行 JavaScript 代码

```
<html>

<head>
<script type="text/javascript">
function upperCase(x)
{
var y=document.getElementById(x).value
document.getElementById(x).value=y.toUpperCase()
}
</script>
</head>

<body>

Enter your name: <input type="text" id="fname"
onchange="upperCase(this.id)">

</body>
</html>
```

onclick 事件

定义和用法

onclick 事件会在对象被点击时发生。
请注意， onclick 与 onmousedown 不同。单击事件是在同一元素上发生了鼠标按下事件之后又发生了鼠标放开事件时才发生的。

语法

```
onclick="SomeJavaScriptCode"
```

参数	描述
SomeJavaScriptCode	必需。规定该事件发生时执行的 JavaScript。

支持该事件的 HTML 标签：

```
<a>, <address>, <area>, <b>, <bdo>, <big>, <blockquote>, <body>,
<button>, <caption>, <cite>, <code>, <dd>, <dfn>, <div>, <dl>, <dt>,
<em>, <fieldset>, <form>, <h1> to <h6>, <hr>, <i>, <img>, <input>,
<kbd>, <label>, <legend>, <li>, <map>, <object>, <ol>, <p>, <pre>,
<samp>, <select>, <small>, <span>, <strong>, <sub>, <sup>, <table>,
<tbody>, <td>, <textarea>, <tfoot>, <th>, <thead>, <tr>, <tt>, <ul>,
<var>
```

支持该事件的 JavaScript 对象：

```
button, document, checkbox, link, radio, reset, submit
```

实例 1

在本例中，当按钮被单击时，第一个输入框中的文本会被拷贝到第二个输入框中：

```
<html>
<body>

Field1: <input type="text" id="field1" value="Hello World!">
<br />
Field2: <input type="text" id="field2">
<br /><br />
点击下面的按钮，把 Field1 的内容拷贝到 Field2 中：
<br />
<button onclick="document.getElementById('field2').value=
document.getElementById('field1').value">Copy Text</button>

</body>
</html>
```

TIY

onclick

如何使用 onclick

```
<html>
<body>

Field1: <input type="text" id="field1" value="Hello World!">
<br />
Field2: <input type="text" id="field2">
<br /><br />
Click the button below to copy the content of Field1 to Field2.
<br />
<button onclick="document.getElementById('field2').value=
document.getElementById('field1').value">Copy Text</button>

</body>
</html>
```

ondblclick 事件

定义和用法

ondblclick 事件会在对象被双击时发生。

语法

```
ondblclick="SomeJavaScriptCode"
```

参数	描述
SomeJavaScriptCode	必需。规定该事件发生时执行的 JavaScript。

支持该事件的 HTML 标签：

```
<a>, <address>, <area>, <b>, <bdo>, <big>, <blockquote>, <body>,
```

```
<button>, <caption>, <cite>, <code>, <dd>, <dfn>, <div>, <dl>, <dt>,
<em>, <fieldset>, <form>, <h1> to <h6>, <hr>, <i>, <img>, <input>,
<kbd>, <label>, <legend>, <li>, <map>, <object>, <ol>, <p>, <pre>,
<samp>, <select>, <small>, <span>, <strong>, <sub>, <sup>, <table>,
<tbody>, <td>, <textarea>, <tfoot>, <th>, <thead>, <tr>, <tt>, <ul>,
<var>
```

支持该事件的 JavaScript 对象：

```
document, link
```

实例 1

在本例中，当您双击按钮时，第二个域中内容会根据第一个域的内容而改变：

```
<html>
<body>

Field1: <input type="text" id="field1" value="Hello World!">
<br />
Field2: <input type="text" id="field2">
<br /><br />
双击下面的按钮，把 Field1 的内容拷贝到 Field2 中：
<br />
<button ondblclick="document.getElementById('field2').value=
document.getElementById('field1').value">Copy Text</button>

</body>
</html>
```

TIY

ondblclick

如何使用 ondblclick。

```
<html>
<body>

Field1: <input type="text" id="field1" value="Hello World!">
<br />
Field2: <input type="text" id="field2">
```

```
<br /><br />
Click the button below to copy the content of Field1 to Field2.
<br />
<button ondblclick="document.getElementById('field2').value=
document.getElementById('field1').value">Copy Text</button>

</body>
</html>
```

onerror 事件

定义和用法

onerror 事件会在文档或图像加载过程中发生错误时被触发。
在装载文档或图像的过程中如果发生了错误，就会调用该事件句柄。

语法

```
onerror="SomeJavaScriptCode"
```

参数	描述
SomeJavaScriptCode	必需。规定该事件发生时执行的 JavaScript。

支持该事件的 HTML 标签：

```
<img>, <object>, <style>
```

支持该事件的 JavaScript 对象：

```
window, image
```

实例 1

在本例中，如果装载图像时发生了错误，则显示一个对话框：

```
<img src="image.gif" onerror="alert('The image could not be
```

```
loaded.')" />
```

TIY

onerror

如何使用 onerror。

```
<html>
<body>

<img src="image.gif"
onerror="alert('The image could not be loaded.')">

<p>在本例中，我们引用的图片不存在，所有弹出一个提示框。</p>

</body>
</html>
```

onfocus 事件

定义和用法

onfocus 事件在对象获得焦点时发生。

语法

```
onfocus="SomeJavaScriptCode"
```

参数	描述
SomeJavaScriptCode	必需。规定该事件发生时执行的 JavaScript。

支持该事件的 HTML 标签：

```
<a>, <acronym>, <address>, <area>, <b>, <bdo>, <big>, <blockquote>,
<button>, <caption>, <cite>, <dd>, <del>, <dfn>, <div>, <dl>, <dt>,
<em>, <fieldset>, <form>, <frame>, <frameset>, <h1> to <h6>, <hr>,
<i>, <iframe>, <img>, <input>, <ins>, <kbd>, <label>, <legend>, <li>,
```

```
<object>, <ol>, <p>, <pre>, <q>, <samp>, <select>, <small>, <span>,
<strong>, <sub>, <sup>, <table>, <tbody>, <td>, <textarea>, <tfoot>,
<th>, <thead>, <tr>, <tt>, <ul>, <var>
```

支持该事件的 JavaScript 对象：

```
button, checkbox, fileUpload, layer, frame, password, radio, reset,
select, submit, text, textarea, window
```

实例

在本例中，当输入框获得焦点时，其背景颜色将改变：

```
<html>
<head>
<script type="text/javascript">
function setStyle(x)
{
document.getElementById(x).style.background="yellow"
}
</script>
</head>

<body>

First name: <input type="text"
onfocus="setStyle(this.id)" id="fname" />
<br />
Last name: <input type="text"
onfocus="setStyle(this.id)" id="lname" />

</body>
</html>
```

TIY

onfocus

如何使用 onfocus。

```
<html>
<head>
```

```
<script type="text/javascript">
function setStyle(x)
{
document.getElementById(x).style.background="yellow"
}
</script>
</head>
<body>

First name: <input type="text" onfocus="setStyle(this.id)"
id="fname">
<br />
Last name: <input type="text" onfocus="setStyle(this.id)"
id="lname">

</body>
</html>
```

onkeydown 事件

定义和用法

onkeydown 事件会在用户按下一个键盘按键时发生。

语法

```
onkeydown="SomeJavaScriptCode"
```

参数	描述
SomeJavaScriptCode	必需。规定该事件发生时执行的 JavaScript。

支持该事件的 HTML 标签：

```
<a>, <acronym>, <address>, <area>, <b>, <bdo>, <big>, <blockquote>,
<body>, <button>, <caption>, <cite>, <code>, <dd>, <del>, <dfn>, <div>,
<dt>, <em>, <fieldset>, <form>, <h1> to <h6>, <hr>, <i>, <input>, <kbd>,
<label>, <legend>, <li>, <map>, <object>, <ol>, <p>, <pre>, <q>, <samp>,
<select>, <small>, <span>, <strong>, <sub>, <sup>, <table>, <tbody>,
```

```
<td>, <textarea>, <tfoot>, <th>, <thead>, <tr>, <tt>, <ul>, <var>
```

支持该事件的 JavaScript 对象：

```
document, image, link, textarea
```

提示和注释

浏览器差异： Internet Explorer 使用 event.keyCode 取回被按下的字符，而 Netscape/Firefox/Opera 使用 event.which。

实例

在本例中，用户无法在输入框中键入数字：

```
<html>
<body>
<script type="text/javascript">
function noNumbers(e)
{
var keynum
var keychar
var numcheck

if(window.event) // IE
  {
  keynum = e.keyCode
  }
else if(e.which) // Netscape/Firefox/Opera
  {
  keynum = e.which
  }

keychar = String.fromCharCode(keynum)
numcheck = /\d/
return !numcheck.test(keychar)
}
</script>

<form>
<input type="text" onkeydown="return noNumbers(event)" />
</form>
```

```
</html>
```

rTIY

onkeydown

如何使用 onkeydown。

```
<html>
<body>
<script type="text/javascript">
function noNumbers(e)
{
var keynum
var keychar
var numcheck

if(window.event) // IE
    {
    keynum = e.keyCode
    }
else if(e.which) // Netscape/Firefox/Opera
    {
    keynum = e.which
    }
keychar = String.fromCharCode(keynum)
numcheck = /\d/
return !numcheck.test(keychar)
}
</script>

<form>
Type some text (numbers not allowed):
<input type="text" onkeydown="return noNumbers(event)" />
</form>

</html>
```

onkeypress 事件

定义和用法

onkeypress 事件会在键盘按键被按下并释放一个键时发生。

语法

```
onkeypress="SomeJavaScriptCode"
```

参数	描述
SomeJavaScriptCode	必需。规定该事件发生时执行的 JavaScript。

支持该事件的 HTML 标签：

```
<a>, <acronym>, <address>, <area>, <b>, <bdo>, <big>, <blockquote>,
<body>, <button>, <caption>, <cite>, <code>, <dd>, <del>, <dfn>, <div>,
<dt>, <em>, <fieldset>, <form>, <h1> to <h6>, <hr>, <i>, <input>, <kbd>,
<label>, <legend>, <li>, <map>, <object>, <ol>, <p>, <pre>, <q>, <samp>,
<select>, <small>, <span>, <strong>, <sub>, <sup>, <table>, <tbody>,
<td>, <textarea>, <tfoot>, <th>, <thead>, <tr>, <tt>, <ul>, <var>
```

支持该事件的 JavaScript 对象：

```
document, image, link, textarea
```

提示和注释

浏览器差异： Internet Explorer 使用 event.keyCode 取回被按下的字符，而 Netscape/Firefox/Opera 使用 event.which。

实例

在本例中，用户无法在输入框中键入数字：

```
<html>
```

```
<body>
<script type="text/javascript">
function noNumbers(e)
{
var keynum
var keychar
var numcheck

if(window.event) // IE
  {
  keynum = e.keyCode
  }
else if(e.which) // Netscape/Firefox/Opera
  {
  keynum = e.which
  }
keychar = String.fromCharCode(keynum)
numcheck = /\d/
return !numcheck.test(keychar)
}
</script>

<form>
<input type="text" onkeypress="return noNumbers(event)" />
</form>

</html>
```

TIY

onkeypress

如何使用 onkeypress

```
<html>
<body>
<script type="text/javascript">
function noNumbers(e)
{
var keynum
var keychar
var numcheck
```

```
if(window.event) // IE
    {
    keynum = e.keyCode
    }
else if(e.which) // Netscape/Firefox/Opera
    {
    keynum = e.which
    }
keychar = String.fromCharCode(keynum)
numcheck = /\d/
return !numcheck.test(keychar)
}
</script>

<form>
Type some text (numbers not allowed):
<input type="text" onkeypress="return noNumbers(event)" />
</form>

</html>
```

onKeyUp 事件

定义和用法

onkeyup 事件会在键盘按键被松开时发生。

语法

```
onkeyup="SomeJavaScriptCode"
```

参数	描述
SomeJavaScriptCode	必需。规定该事件发生时执行的 JavaScript。

支持该事件的 HTML 标签：

```
<a>, <acronym>, <address>, <area>, <b>, <bdo>, <big>, <blockquote>,
<body>, <button>, <caption>, <cite>, <code>, <dd>, <del>, <dfn>, <div>,
```

```
<dt>, <em>, <fieldset>, <form>, <h1> to <h6>, <hr>, <i>, <input>, <kbd>,
<label>, <legend>, <li>, <map>, <object>, <ol>, <p>, <pre>, <q>, <samp>,
<select>, <small>, <span>, <strong>, <sub>, <sup>, <table>, <tbody>,
<td>, <textarea>, <tfoot>, <th>, <thead>, <tr>, <tt>, <ul>, <var>
```

支持该事件的 JavaScript 对象：

```
document, image, link, textarea
```

实例

当您在例子中的输入域中键入字符时，字符会被更改为大写（逐一地）：

```
<html>

<head>
<script type="text/javascript">
function upperCase(x)
{
var y=document.getElementById(x).value
document.getElementById(x).value=y.toUpperCase()
}
</script>
</head>

<body>

输入您的姓名： <input type="text" id="fname"
onkeyup="upperCase(this.id)" />

</body>
</html>
```

TIY

onkeyup

如何使用 onkeyup。

```
<html>

<head>
```

```
<script type="text/javascript">
function upperCase(x)
{
var y=document.getElementById(x).value
document.getElementById(x).value=y.toUpperCase()
}
</script>
</head>

<body>

Enter your name: <input type="text" id="fname"
onkeyup="upperCase(this.id)">

</body>
</html>
```

onload 事件

定义和用法

onload 事件会在页面或图像加载完成后立即发生。

语法

```
onload="SomeJavaScriptCode"
```

参数	描述
SomeJavaScriptCode	必需。规定该事件发生时执行的 JavaScript。

支持该事件的 HTML 标签：

```
<body>, <frame>, <frameset>, <iframe>, <img>, <link>, <script>
```

支持该事件的 JavaScript 对象：

```
image, layer, window
```

实例

在本例中，文本 "Page is loaded" 会被显示在状态栏中：

```
<html>
<head>
<script type="text/javascript">
function load()
{
window.status="Page is loaded"
}
</script>
</head>

<body onload="load()">
</body>

</html>
```

TIY

onload

如何使用 onload 在页面完成加载时在状态栏显示一段文本。

```
<html>
<head>
<script type="text/javascript">
function load()
{
window.status="Page is loaded"
}
</script>
</head>

<body onload="load()">
</body>

</html>
```

onmousedown 事件

定义和用法

onmousedown 事件会在鼠标按键被按下时发生。

语法

```
onmousedown="SomeJavaScriptCode"
```

参数	描述
SomeJavaScriptCode	必需。规定该事件发生时执行的 JavaScript。

支持该事件的 HTML 标签：

```
<a>, <address>, <area>, <b>, <bdo>, <big>, <blockquote>, <body>,
<button>, <caption>, <cite>, <code>, <dd>, <dfn>, <div>, <dl>, <dt>,
<em>, <fieldset>, <form>, <h1> to <h6>, <hr>, <i>, <img>, <input>,
<kbd>, <label>, <legend>, <li>, <map>, <ol>, <p>, <pre>, <samp>,
<select>, <small>, <span>, <strong>, <sub>, <sup>, <table>, <tbody>,
<td>, <textarea>, <tfoot>, <th>, <thead>, <tr>, <tt>, <ul>, <var>
```

支持该事件的 JavaScript 对象：

```
button, document, link
```

实例

在本例中，当您点击下面的图片时会弹出一个对话框：

```
<img src="/i/example_mouse2.jpg"
onmousedown="alert('You clicked the picture!')" />
```

实例 2

在本例中，对话框将显示出您所点击的元素的标签名：

```
<html>
<head>
<script type="text/javascript">
function whichElement(e)
{
var targ
if (!e) var e = window.event
if (e.target) targ = e.target
else if (e.srcElement) targ = e.srcElement
if (targ.nodeType == 3) // defeat Safari bug
targ = targ.parentNode
var tname
tname=targ.tagName
alert("You clicked on a " + tname + " element.")
}
</script>
</head>

<body onmousedown="whichElement(event)">

<h2>This is a header</h2>
<p>This is a paragraph</p>
<img border="0" src="ball16.gif" alt="Ball">

</body>
</html>
```

TIY

onmousedown

如何使用 onmousedown 在图像被点击时显示一个对话框。

```
<html>
<body>

<img src="/i/eg_mouse2.jpg" onMouseDown="alert('You clicked the
picture!')">
<p>Click the picture</p>

</body>
</html>
```

onmousedown 2

如何使用 onmousedown 来显示您所点击的元素的标签名。

```
<html>
<head>
<script type="text/javascript">
function whichElement(e)
{
var targ
if (!e) var e = window.event
if (e.target) targ = e.target
else if (e.srcElement) targ = e.srcElement
if (targ.nodeType == 3) // defeat Safari bug
   targ = targ.parentNode
var tname
tname=targ.tagName
alert("You clicked on a " + tname + " element.")
}
</script>
</head>

<body onmousedown="whichElement(event)">
<p>在文档中点击某个位置。消息框会提示出您所点击的标签的名称。</p>

<h3>这是标题</h3>
<p>这是段落。</p>
<img src="/i/eg_mouse2.jpg" />
</body>

</html>
```

onmousemove 事件

定义和用法

onmousemove 事件会在鼠标指针移动时发生。

语法

```
onmousemove="SomeJavaScriptCode"
```

参数	描述
SomeJavaScriptCode	必需。规定该事件发生时执行的 JavaScript。

支持该事件的 HTML 标签：

```
<a>, <address>, <area>, <b>, <bdo>, <big>, <blockquote>, <body>,
<button>, <caption>, <cite>, <code>, <dd>, <dfn>, <div>, <dl>, <dt>,
<em>, <fieldset>, <form>, <h1> to <h6>, <hr>, <i>, <img>, <input>,
<kbd>, <label>, <legend>, <li>, <map>, <ol>, <p>, <pre>, <samp>,
<select>, <small>, <span>, <strong>, <sub>, <sup>, <table>, <tbody>,
<td>, <textarea>, <tfoot>, <th>, <thead>, <tr>, <tt>, <ul>, <var>
```

支持该事件的 JavaScript 对象：

```
onmousemove is, by default, not an event of any object,
because mouse movement happens very frequently.
```

提示和注释

注释：每当用户把鼠标移动一个像素，就会发生一个 mousemove 事件。这会耗费系统资源去处理所有这些 mousemove 事件。因此请审慎地使用该事件。

实例

下面的例子中，当用户把鼠标移动到图像上时，将显示一个对话框：

```
<img src="/i/eg_mouse2.jpg" alt="mouse"
onmousemove="alert('您的鼠标刚才经过了图片！')" />
```

TIY

onmousemove

如何使用 onmousemove。

```
<html>
<body>

<img src="/i/eg_mouse2.jpg" name="mousetest">

<p onmousemove="document.images['mousetest']
.src='/i/eg_mouse.jpg'">When you move the mouse pointer over this
paragraph, the image changes.</p>

</body>
</html>
```

onmouseout 事件

定义和用法

onmouseout 事件会在鼠标指针移出指定的对象时发生。

语法

```
onmouseout="SomeJavaScriptCode"
```

参数	描述
SomeJavaScriptCode	必需。规定该事件发生时执行的 JavaScript。

支持该事件的 HTML 标签：

```
<a>, <address>, <area>, <b>, <bdo>, <big>, <blockquote>, <body>,
<button>, <caption>, <cite>, <code>, <dd>, <dfn>, <div>, <dl>, <dt>,
<em>, <fieldset>, <form>, <h1> to <h6>, <hr>, <i>, <img>, <input>,
<kbd>, <label>, <legend>, <li>, <map>, <ol>, <p>, <pre>, <samp>,
<select>, <small>, <span>, <strong>, <sub>, <sup>, <table>, <tbody>,
<td>, <textarea>, <tfoot>, <th>, <thead>, <tr>, <tt>, <ul>, <var>
```

支持该事件的 JavaScript 对象：

```
layer, link
```

实例 1

在下面的例子中，我们将在鼠标指针移出图像时显示一个对话框：

```
<img src="/i/example_mouse2.jpg" alt="mouse"
onmousemove="alert('您的鼠标刚才离开了图片！')" />
```

实例 2

下面的例子中，我们将在网页上添加一个用作连接按钮的图像，然后我们会添加 onMouseOver 和 onMouseOut 事件，这样就可以在运行两个 JavaScript 函数来切换两幅图像：

```
<html>
<head>
<script type="text/javascript">
function mouseOver()
{
document.getElementById('b1').src ="/i/eg_mouse.jpg"
}
function mouseOut()
{
document.getElementById('b1').src ="/i/eg_mouse2.jpg"
}
</script>
</head>

<body>
<a href="http://www.w3school.com.cn"
onmouseover="mouseOver()" onmouseout="mouseOut()">
<img alt="Visit W3School!" src="/i/example_mouse2.jpg" id="b1" />
</a>
</body>
</html>
```

TIY

onmouseout

如何使用 onmouseout。

```
<html>
<head>
<script type="text/javascript">
function mouseOver()
{
document.getElementById('b1').src ="/i/eg_mouse.jpg"
}
function mouseOut()
{
document.getElementById('b1').src ="/i/eg_mouse2.jpg"
}
</script>
</head>

<body>
<a href="http://www.w3school.com.cn" onmouseover="mouseOver()"
onmouseout="mouseOut()">
<img alt="Visit W3School!" src="/i/eg_mouse2.jpg" id="b1" />
</a>
</body>
</html>
```

onmouseover 事件

定义和用法

onmouseover 事件会在鼠标指针移动到指定的对象上时发生。

语法

```
onmouseover="SomeJavaScriptCode"
```

参数	描述

SomeJavaScriptCode	必需。规定该事件发生时执行的 JavaScript。

支持该事件的 HTML 标签：

```
<a>, <address>, <area>, <b>, <bdo>, <big>, <blockquote>, <body>,
<button>, <caption>, <cite>, <code>, <dd>, <dfn>, <div>, <dl>, <dt>,
<em>, <fieldset>, <form>, <h1> to <h6>, <hr>, <i>, <img>, <input>,
<kbd>, <label>, <legend>, <li>, <map>, <ol>, <p>, <pre>, <samp>,
<select>, <small>, <span>, <strong>, <sub>, <sup>, <table>, <tbody>,
<td>, <textarea>, <tfoot>, <th>, <thead>, <tr>, <tt>, <ul>, <var>
```

支持该事件的 JavaScript 对象：

```
layer, link
```

实例 1

在下面的例子中，我们将在用户把鼠标指针移动到图像上时显示一个对话框：

```
<img src="/i/eg_mouse2.jpg" alt="mouse"
onmouseover="alert('您的鼠标在图片上！')" />
```

实例 2

下面的例子中，我们将在网页上添加一个用作连接按钮的图像，然后我们会添加 onMouseOver 和 onMouseOut 事件，这样就可以在运行两个 JavaScript 函数来切换两幅图像：

```
<html>
<head>
<script type="text/javascript">
function mouseOver()
{
document.getElementById('b1').src ="/i/eg_mouse.jpg"
}
function mouseOut()
{
document.getElementById('b1').src ="/i/eg_mouse2.jpg"
}
</script>
```

```
</head>

<body>
<a href="http://www.w3school.com.cn"
onmouseover="mouseOver()" onmouseout="mouseOut()">
<img alt="Visit W3School!" src="/i/eg_mouse2.jpg" id="b1" />
</a>
</body>
</html>
```

TIY

onmouseover

如何使用 onmouseover。

```
<html>
<head>
<script type="text/javascript">
function mouseOver()
{
document.getElementById('b1').src ="/i/eg_mouse.jpg"
}
function mouseOut()
{
document.getElementById('b1').src ="/i/eg_mouse2.jpg"
}
</script>
</head>

<body>
<a href="http://www.w3school.com.cn" onmouseover="mouseOver()"
onmouseout="mouseOut()">
<img alt="Visit W3School!" src="/i/eg_mouse2.jpg" id="b1" />
</a>
</body>
</html>
```

onreset 事件

定义和用法

onreset 事件会在表单中的重置按钮被点击时发生。

语法

```
onreset="SomeJavaScriptCode"
```

参数	描述
SomeJavaScriptCode	必需。规定该事件发生时执行的 JavaScript。

支持该事件的 HTML 标签：

```
<form>
```

支持该事件的 JavaScript 对象：

```
form
```

实例

在本例中，当重置按钮被点击时，表单会改为默认值，并显示一个对话框：

```
<form onreset="alert('The form will be reset')">

Firstname: <input type="text" name="fname" value="John" />
<br />
Lastname: <input type="text" name="lname" />
<br /><br />
<input type="reset" value="Reset">

</form>
```

TIY

onreset

如何使用 onreset 在 reset 事件发生时显示一个对话框。

```
<html>
<body>

<form onreset="alert('The form will be reset')">
Firstname: <input type="text" name="fname" value="John" />
<br />
Lastname: <input type="text" name="lname" />
<br /><br />
<input type="reset" value="Reset">
</form>

</body>
</html>
```

onresize 事件

定义和用法

onresize 事件会在窗口或框架被调整大小时发生。

语法

```
onresize="SomeJavaScriptCode"
```

参数	描述
SomeJavaScriptCode	必需。规定该事件发生时执行的 JavaScript。

支持该事件的 HTML 标签：

```
<a>, <address>, <b>, <big>, <blockquote>, <body>, <button>, <cite>,
<code>, <dd>, <dfn>, <div>, <dl>, <dt>, <em>, <fieldset>, <form>,
```

```
<frame>, <h1> to <h6>,<hr>, <i>, <img>, <input>, <kbd>, <label>,
<legend>, <li>, <object>, <ol>, <p>, <pre>, <samp>, <select>, <small>,
<span>, <strong>, <sub>, <sup>, <table>, <textarea>, <tt>, <ul>, <var>
```

支持该事件的 JavaScript 对象：

```
window
```

实例

在本例中，当用户试图调整窗口的大小时，将显示一个对话框：

```
<body onresize="alert('You have changed the size of the window')">
</body>
```

TIY

onresize

如何使用 onresize 在窗口被调整大小时显示一个对话框。

```
<html>

<body onresize="alert('You have changed the size of the window')">
<p>Try to resize the browser window.</p>
</body>

</html>
```

onselect 事件

定义和用法

onselect 事件会在文本框中的文本被选中时发生。

语法

```
onselect="SomeJavaScriptCode"
```

参数	描述
SomeJavaScriptCode	必需。规定该事件发生时执行的 JavaScript。

支持该事件的 HTML 标签：

```
<input type="text">, <textarea>
```

支持该事件的 JavaScript 对象：

```
window
```

实例

在本例中，当用户试图选择文本框中的文本时，会显示一个对话框：

```
<form>

Select text: <input type="text" value="Hello world!"
onselect="alert('You have selected some of the text.')" />
<br /><br />
Select text: <textarea cols="20" rows="5"
onselect="alert('You have selected some of the text.')">
Hello world!</textarea>

</form>
```

TIY

onselect

如何使用 onselect。

```
<form>
Select text: <input type="text" value="Hello world!"
onselect="alert('You have selected some of the text.')">
<br /><br />
Select text: <textarea cols="20" rows="5"
onselect="alert('You have selected some of the text.')">
Hello world!
```

onsubmit 事件

定义和用法

onsubmit 事件会在表单中的确认按钮被点击时发生。

语法

```
onsubmit="SomeJavaScriptCode"
```

参数	描述
SomeJavaScriptCode	必需。规定该事件发生时执行的 JavaScript。

支持该事件的 HTML 标签：

```
<form>
```

支持该事件的 JavaScript 对象：

```
form
```

实例

在本例中，当用户点击提交按钮时，会显示一个对话框：

```
<form name="testform" action="jsref_onsubmit.asp"
onsubmit="alert('Hello ' + testform.fname.value +'!')">

What is your name?<br />
<input type="text" name="fname" />
<input type="submit" value="Submit" />

</form>
```

TIY

onsubmit

如何使用 onsubmit。

```
<html>

<body>

<h1>What is your name?</h1>
<form name="testform" action="/example/hdom/hdom_onsubmit.html"
onSubmit="alert('Hello ' + testform.inputfield.value +'!')">
<input type="text" name="inputfield" size="20">
<input type="submit" value="Submit">
</form>

</body>

</html>
```

onunload 事件

定义和用法

onunload 事件在用户退出页面时发生。

语法

```
onunload="SomeJavaScriptCode"
```

参数	描述
SomeJavaScriptCode	必需。规定该事件发生时执行的 JavaScript。

支持该事件的 HTML 标签：

```
<body>, <frameset>
```

支持该事件的 JavaScript 对象：

```
window
```

实例

在本例中，在页面关闭时会显示一个对话框：

```
<body onunload="alert('The onunload event was triggered')">
</body>
```

TIY

onunload

如何使用 onunload。

```
<html>

<body onunload="alert('The onunload event was triggered')">
<p>Close the page to trigger the onunload event.</p>
</body>

</html>
```

altKey 事件属性

定义和用法

altKey 事件属性返回一个布尔值。指示在指定的事件发生时，Alt 键是否被按下并保持住了。

语法

```
event.altKey=true|false|1|0
```

实例

下面的例子可提示当鼠标按键被点击时 "ALT" 键是否已被按住：

```
<html>
<head>
<script type="text/javascript">
function isKeyPressed(event)
{
  if (event.altKey==1)
    {
    alert("The ALT key was pressed!")
    }
  else
    {
    alert("The ALT key was NOT pressed!")
    }
  }
</script>
</head>

<body onmousedown="isKeyPressed(event)">

<p>Click somewhere in the document.
An alert box will tell you if you
pressed the ALT key or not.</p>

</body>
</html>
```

TIY

altKey

检测 ALT 键是否已被按住。

```
<html>
<head>
<script type="text/javascript">
function isKeyPressed(event)
{
  if (event.altKey==1)
    {
```

```
    alert("The ALT key was pressed!")
    }
  else
    {
    alert("The ALT key was NOT pressed!")
    }
  }
</script>
</head>

<body onmousedown="isKeyPressed(event)">

<p>Click somewhere in the document. An alert box will tell you if you pressed the ALT key or not.</p>

</body>
</html>
```

button 事件属性

定义和用法

button 事件属性可返回一个整数，指示当事件被触发时哪个鼠标按键被点击。

语法

```
event.button=0|1|2
```

参数	描述
0	规定鼠标左键。
1	规定鼠标中键。
2	规定鼠标右键。

Internet Explorer 拥有不同的参数：

参数	描述
1	规定鼠标左键。

4	规定鼠标中键。
2	规定鼠标右键。

提示和注释

注释：对于惯用左手的鼠标配置，上面的参数是颠倒的。
提示：Mozilla 的 Ctrl–Click 参数是 2 （等价于右击）。

实例

The following example alerts which mouse button was clicked:

```
<html>
<head>
<script type="text/javascript">
function whichButton(event)
  {
    if (event.button==2)
      {
      alert("You clicked the right mouse button!")
      }
    else
      {
      alert("You clicked the left mouse button!")
      }
  }
</script>
</head>

<body onmousedown="whichButton(event)">

<p>Click in the document. An alert box will
alert which mouse button you clicked.</p>

</body>
</html>
```

TIY

button

检测哪个鼠标键被点击了。

```
<html>
<head>
<script type="text/javascript">
function whichButton(event)
{
var btnNum = event.button;
if (btnNum==2)
  {
  alert("您点击了鼠标右键！")
  }
else if(btnNum==0)
  {
  alert("您点击了鼠标左键！")
  }
else if(btnNum==1)
  {
  alert("您点击了鼠标中键！");
  }
else
  {
  alert("您点击了" + btnNum+ "号键，我不能确定它的名称。");
  }
}
</script>
</head>

<body onmousedown="whichButton(event)">
<p>请在文档中点击鼠标。一个消息框会提示出您点击了哪个鼠标按键。</p>
</body>

</html>
```

clientX 事件属性

定义和用法

clientX 事件属性返回当事件被触发时鼠标指针向对于浏览器页面（或客户区）的水平坐标。客户区指的是当前窗口。

语法

```
event.clientX
```

提示和注释

注释：2 级 DOM 没有提供把窗口坐标转换为文档坐标的标准方法。在 IE 以外的浏览器，使用 window.pageXOffset 和 window.pageYOffset 即可。

实例

下面的例子可显示出事件发生时鼠标指针的坐标：

```
<html>
<head>
<script type="text/javascript">
function show_coords(event)
  {
  x=event.clientX
  y=event.clientY
  alert("X coords: " + x + ", Y coords: " + y)
  }
</script>
</head>

<body onmousedown="show_coords(event)">

   <p>Click in the document. An alert box will alert
the x and y coordinates of the mouse pointer.</p>

</body>
</html>
```

TIY

clientX

显示鼠标指针的坐标。

```
<html>
<head>
```

```
<script type="text/javascript">
function show_coords(event)
{
x=event.clientX
y=event.clientY
alert("X 坐标: " + x + ", Y 坐标: " + y)
}
</script>
</head>

<body onmousedown="show_coords(event)">

<p>请在文档中点击。一个消息框会提示出鼠标指针的 x 和 y 坐标。</p>

</body>
</html>
```

clientY 事件属性

定义和用法

clientY 事件属性返回当事件被触发时鼠标指针向对于浏览器页面（客户区）的垂直坐标。客户区指的是当前窗口。

语法

```
event.clientY
```

提示和注释

注释：注意，该坐标不考虑文档的滚动。如果事件发生在窗口的顶部，无论文档滚了多远，clientY 的值都是 0。但是，2 级 DOM 没有提供把窗口坐标转换为文档坐标的标准方法。在 IE 以外的浏览器，使用 window.pageXOffset 和 window.pageYOffset 即可。

实例

下面的例子可显示出事件发生时鼠标指针的坐标：

```
<html>
```

```
<head>
<script type="text/javascript">
function show_coords(event)
  {
  x=event.clientX
  y=event.clientY
  alert("X coords: " + x + ", Y coords: " + y)
  }
</script>
</head>

<body onmousedown="show_coords(event)">

<p>Click in the document. An alert box will alert
the x and y coordinates of the mouse pointer.</p>

</body>
</html>
```

TIY

clientY

显示鼠标指针的坐标。

```
<html>
<head>
<script type="text/javascript">
function show_coords(event)
{
x=event.clientX
y=event.clientY
alert("X 坐标: " + x + ", Y 坐标: " + y)
}
</script>
</head>

<body onmousedown="show_coords(event)">

<p>请在文档中点击。一个消息框会提示出鼠标指针的 x 和 y 坐标。</p>

</body>
</html>
```

ctrlKey 事件属性

定义和用法

ctrlKey 事件属性可返回一个布尔值，指示当事件发生时，Ctrl 键是否被按下并保持住。

语法

```
event.ctrlKey=true|false|1|0
```

实例

下面的例子可提示当鼠标按键被点击时 "CTRL" 键是否被按住：

```
<html>
<head>
<script type="text/javascript">
function isKeyPressed(event)
{
  if (event.ctrlKey==1)
    {
    alert("The CTRL key was pressed!")
    }
  else
    {
    alert("The CTRL key was NOT pressed!")
    }
  }
</script>
</head>

<body onmousedown="isKeyPressed(event)">

    <p>Click somewhere in the document.
An alert box will tell you if you
pressed the CTRL key or not.</p>

</body>
</html>
```

TIY

ctrlKey

检测 CTRL 键是否被按下了。

```
<html>
<head>
<script type="text/javascript">
function isKeyPressed(event)
{
  if (event.ctrlKey==1)
    {
    alert("The CTRL key was pressed!")
    }
  else
    {
    alert("The CTRL key was NOT pressed!")
    }
  }

</script>
</head>
<body onmousedown="isKeyPressed(event)">

<p>Click somewhere in the document. An alert box will tell you if you
pressed the CTRL key or not.</p>

</body>
</html>
```

metaKey 事件属性

定义和用法

metaKey 事件属性可返回一个布尔值，指示当事件发生时，"meta" 键是否被按下并保持住。

语法

```
event.metaKey
```

实例

下面的例子可提示当鼠标按键被点击时 "meta" 键是否被按住：

```
<html>
<head>
<script type="text/javascript">
function isKeyPressed(event)
{
  if (event.metaKey==1)
    {
    alert("The meta key was pressed!")
    }
  else
    {
    alert("The meta key was NOT pressed!")
    }
  }
</script>
</head>

<body onmousedown="isKeyPressed(event)">

<p>Click somewhere in the document.
An alert box will tell you if you
pressed the meta key or not.</p>

</body>
</html>
```

TIY

metakey

检测 meta 键是否被按住（不支持 IE）。

```
<html>
<head>
```

```
<script type="text/javascript">
function isKeyPressed(event)
{
  if (event.metaKey==1)
    {
    alert("The meta key was pressed!")
    }
  else
    {
    alert("The meta key was NOT pressed!")
    }
  }

</script>
</head>
<body onmousedown="isKeyPressed(event)">

<p>Click somewhere in the document. An alert box will tell you if you pressed the meta key or not.</p>

</body>
</html>
```

relatedTarget 事件属性

定义和用法

relatedTarget 事件属性返回与事件的目标节点相关的节点。
对于 mouseover 事件来说，该属性是鼠标指针移到目标节点上时所离开的那个节点。
对于 mouseout 事件来说，该属性是离开目标时，鼠标指针进入的节点。
对于其他类型的事件来说，这个属性没有用。

语法

```
event.relatedTarget
```

实例

下面例子可返回指针刚刚离开的元素：

```
<html>
<head>
<script type="text/javascript">
function getRelElement(event)
  {
  var txt="The cursor just exited the ";
  txt=txt + event.relatedTarget.tagName + " element.";
  alert(txt);
  }
</script>
</head>
<body>

<p onmouseover="getRelElement(event)">
Mouse over this paragraph.</p>

</body>
</html>
```

TIY

relatedtarget

返回指针刚刚离开的元素（不支持 IE）。

```
<html>
<head>
<script type="text/javascript">
function getRelatedElement(event)
  {
  alert("The cursor just exited the " + event.relatedTarget.tagName
+ " element.");
  }
</script>
</head>
<body>

<p onmouseover="getRelatedElement(event)">Mouse over this
paragraph.</p>

</body>
</html>
```

screenX 事件属性

定义和用法

screenX 事件属性可返回事件发生时鼠标指针相对于屏幕的水平坐标。

语法

```
event.screenX
```

实例

下面的例子可显示出事件发生时鼠标指针的坐标：

```
<html>
<head>
<script type="text/javascript">
function show_coords(event)
  {
  x=event.screenX
  y=event.screenY
  alert("X coords: " + x + ", Y coords: " + y)
  }
</script>
</head>
<body onmousedown="show_coords(event)">

<p>Click in the document. An alert box will alert
the x and y coordinates of the cursor.</p>

</body>
</html>
```

TIY

screenX

提示鼠标指针的坐标。

```
<html>
<head>

<script type="text/javascript">
function coordinates(event)
{
x=event.screenX
y=event.screenY
alert("X=" + x + " Y=" + y)
}

</script>
</head>
<body onmousedown="coordinates(event)">

<p>
在文档中点击某个位置。消息框会提示出指针相对于屏幕的 x 和 y 坐标。
</p>

</body>
</html>
```

screenY 事件属性

定义和用法

screenY 事件属性可返回事件发生时鼠标指针相对于屏幕的垂直坐标。

语法

```
event.screenY
```

实例

下面的例子可显示出事件发生时鼠标指针的坐标：

```
<html>
<head>
<script type="text/javascript">
```

```
function show_coords(event)
  {
  x=event.screenX
  y=event.screenY
  alert("X coords: " + x + ", Y coords: " + y)
  }
</script>
</head>
<body onmousedown="show_coords(event)">

<p>Click in the document. An alert box will alert
the x and y coordinates of the cursor.</p>

</body>
</html>
```

TIY

screenY

提示鼠标指针的坐标。

```
<html>
<head>

<script type="text/javascript">
function coordinates(event)
{
x=event.screenX
y=event.screenY
alert("X=" + x + " Y=" + y)
}

</script>
</head>
<body onmousedown="coordinates(event)">

<p>
在文档中点击某个位置。消息框会提示出指针相对于屏幕的 x 和 y 坐标。
</p>

</body>
</html>
```

shiftKey 事件属性

定义和用法

shiftKey 事件属性可返回一个布尔值，指示当事件发生时，"SHIFT" 键是否被按下并保持住。

语法

```
event.shiftKey=true|false|1|0
```

实例

下面的例子可提示当鼠标按键被点击时 "SHIFT" 键是否被按住：

```
<html>
<head>
<script type="text/javascript">
function isKeyPressed(event)
{
  if (event.shiftKey==1)
    {
    alert("The shift key was pressed!")
    }
  else
    {
    alert("The shift key was NOT pressed!")
    }
  }
</script>
</head>

<body onmousedown="isKeyPressed(event)">

<p>Click somewhere in the document.
An alert box will tell you if you
pressed the shift key or not.</p>

</body>
</html>
```

TIY

shiftKey

检测 SHIFT 键是否被按住。

```
<html>
<head>
<script type="text/javascript">
function isKeyPressed(event)
{
  if (event.shiftKey==1)
    {
    alert("The shift key was pressed!")
    }
  else
    {
    alert("The shift key was NOT pressed!")
    }
  }
</script>
</head>

<body onmousedown="isKeyPressed(event)">

<p>在文档中点击某个位置。消息框会告诉你是否按下了 shift 键。</p>

</body>
</html>
```

bubbles 事件属性

定义和用法

bubbles 事件属性返回一个布尔值，如果事件是起泡类型，则返回 true，否则返回 fasle。

语法

```
event.bubbles
```

事件传播

在 2 级 DOM 中，事件传播分为三个阶段：
第一，捕获阶段。事件从 Document 对象沿着文档树向下传递给目标节点。如果目标的任何一个先辈专门注册了捕获事件句柄，那么在事件传播过程中运行这些句柄。
第二个阶段发生在目标节点自身。直接注册砸目标上的适合的事件句柄将运行。这与 0 级事件模型提供的事件处理方法相似。
第三，起泡阶段。在此阶段，事件将从目标元素向上传播回或起泡回 Document 对象的文档层次。

实例

下面的例子可检测发生的事件是否是一个起泡事件：

```
<html>
<head>

<script type="text/javascript">
function getEventType(event)
  {
  alert(event.bubbles);
  }
</script>
</head>

<body onmousedown="getEventType(event)">

<p>Click somewhere in the document.
An alert box will tell if the event is a bubbling event.</p>

</body>
</html>
```

TIY

bubbling event

检测事件是否是起泡事件（IE 浏览器不支持）。

```
<html>
<head>
```

```
<script type="text/javascript">
function getEventType(event)
  {
  alert(event.bubbles);
  }
</script>
</head>

<body onmousedown="getEventType(event)">

<p>Click somewhere in the document. An alert box will tell if the event is a bubbling event.</p>

</body>
</html>
```

cancelable 事件属性

定义和用法

cancelable 事件返回一个布尔值。如果用 preventDefault() 方法可以取消与事件关联的默认动作，则为 true，否则为 fasle。

语法

```
event.cancelable
```

实例

下面的例子讲检测发生的事件是否是一个 cancelable 事件：

```
<html>
<head>

<script type="text/javascript">
function isEventCancelable(event)
  {
  alert(event.cancelable);
  }
```

```
</script>
</head>

<body onmousedown="isEventCancelable(event)">

<p>Click somewhere in the document.
An alert box will tell if the
event is a cancelable event.</p>

</body>
</html>
```

TIY

cancelable event

检测事件是否是 cancelable 事件（IE 浏览器不支持）。

```
<html>
<head>
<script type="text/javascript">
function isEventCancelable(event)
  {
  alert(event.cancelable);
  }
</script>
</head>

<body onmousedown="isEventCancelable(event)">

<p>Click somewhere in the document. An alert box will tell if the event
is a cancelable event.</p>

</body>
</html>
```

currentTarget 事件属性

定义和用法

currentTarget 事件属性返回其监听器触发事件的节点，即当前处理该事件的元素、文档或窗口。
在捕获和起泡阶段，该属性是非常有用的，因为在这两个节点，它不同于 target 属性。

语法

```
event.currentTarget
```

实例

下面的例子可获得哪个元素的监听器触发了事件：

```
<html>
<head>

<script type="text/javascript">
function getEventTrigger(event)
  {
  x=event.currentTarget;
  alert("The id of the triggered element: "
  + x.id);
  }
</script>
</head>

<body >

<p id="p1" onmousedown="getEventTrigger(event)">
Click on this paragraph. An alert box will
show which element triggered the event.</p>

</body>
</html>
```

TIY

currenttarget event

获得其监听器触发了事件的那个元素。

```
<html>
<head>

<script type="text/javascript">
function getEventTrigger(event)
  {
  x=event.currentTarget;
  alert("The id of the triggered element: "
   + x.id);
  }
</script>
</head>

<body >

<p id="p1" onmousedown="getEventTrigger(event)">
Click on this paragraph. An alert box will
show which element triggered the event.</p>

</body>
</html>
```

eventPhase 属性

定义和用法

eventPhase 属性返回事件传播的当前阶段。它的值是下面的三个常量之一，它们分别表示捕获阶段、正常事件派发和起泡阶段。

eventPhase 常量

常量	值

Event.CAPTURING_PHASE	1
Event.AT_TARGET	2
Event.BUBBLING_PHASE	3

target 事件属性

定义和用法

target 事件属性可返回事件的目标节点（触发该事件的节点），如生成事件的元素、文档或窗口。

语法

```
event.target
```

实例

下面的例子可获得触发事件的元素：

```
<html>
<head>

<script type="text/javascript">
function getEventTrigger(event)
  {
  x=event.target;
  alert("The id of the triggered element: "
  + x.id);
  }
</script>

</head>
<body >

<p id="p1" onmousedown="getEventTrigger(event)">
Click on this paragraph. An alert box will
show which element triggered the event.</p>
```

```
</body>
</html>
```

TIY

target event

获得触发事件的元素（IE 浏览器不支持）。

```
<html>
<head>

<script type="text/javascript">
function getEventTrigger(event)
  {
  x=event.target;
  alert("The id of the triggered element: " + x.id);
  }
</script>
</head>

<body >

<p id="p1" onmousedown="getEventTrigger(event)">
Click on this paragraph. An alert box will show which element triggered
the event.</p>

</body>
</html>
```

timeStamp 事件属性

定义和用法

timeStamp 事件属性可返回一个时间戳。指示发生事件的日期和时间（从 epoch 开始的毫秒数）。
epoch 是一个事件参考点。在这里，它是客户机启动的时间。
并非所有系统都提供该信息，因此，timeStamp 属性并非对所有系统/事件都是可用的。

语法

```
event.timeStamp
```

实例

下面的例子可获得系统启动开始的事件戳：

```
<html>
<head>
<script type="text/javascript">
function showTimestamp(event)
  {
  var minutes = 1000*60
  x=event.timeStamp;
  alert(x/minutes)
  }
</script>
</head>

<body onmousedown="showTimestamp(event)">

<p>Click in the document. An alert
box will alert the timestamp.</p>

</body>
</html>
```

TIY

timestamp event

返回系统启动至今的分钟数（IE 浏览器不支持）。

```
<html>
<head>
<script type="text/javascript">
function showTimestamp(event)
  {
  var minutes = 1000*60
  x=event.timeStamp;
  alert(x/minutes)
```

```
  }
</script>
</head>

<body onmousedown="showTimestamp(event)">

<p>Click in the document. An alert box will alert the timestamp.</p>

</body>
</html>
```

type 事件属性

定义和用法

type 事件属性返回发生的事件的类型，即当前 Event 对象表示的事件的名称。
它与注册的事件句柄同名，或者是事件句柄属性删除前缀 "on" 比如 "submit"、"load" 或 "click"。

语法

```
event.type
```

实例

下面的例子可返回被触发的事件的类型：

```
<html>
<head>
<script type="text/javascript">
function getEventType(event)
  {
  alert(event.type);
  }
</script>
</head>

<body onmousedown="getEventType(event)">
```

```
<p>Click somewhere in the document.
An alert box will tell what event
type you triggered.</p>

</body>
</html>
```

TIY

type event

返回被触发的事件的类型（IE 浏览器不支持）。

```
<html>
<head>
<script type="text/javascript">
function getEventType(event)
  {
  alert(event.type);
  }
</script>
</head>

<body onmousedown="getEventType(event)">

<p>在文档中点击某个位置。消息框会提示出被触发的事件的类型。</p>

</body>
</html>
```

initEvent() 方法

定义和用法

初始化新事件对象的属性

语法

```
event.initEvent(eventType,canBubble,cancelable)
```

参数	描述
eventType	字符串值。事件的类型。
canBubble	事件是否起泡。
cancelable	是否可以用 preventDefault() 方法取消事件。

说明

该方法将初始化 Document.createEvent() 方法创建的合成 Event 对象的 type 属性、bubbles 属性和 cancelable 属性。只有在新创建的 Event 对象被 Document 对象或 Element 对象的 dispatchEvent() 方法分派之前，才能调用 Event.initEvent() 方法。

preventDefault() 方法

定义和用法

取消事件的默认动作。

语法

```
event.preventDefault()
```

说明

该方法将通知 Web 浏览器不要执行与事件关联的默认动作（如果存在这样的动作）。例如，如果 type 属性是 "submit"，在事件传播的任意阶段可以调用任意的事件句柄，通过调用该方法，可以阻止提交表单。注意，如果 Event 对象的 cancelable 属性是 fasle，那么就没有默认动作，或者不能阻止默认动作。无论哪种情况，调用该方法都没有作用。

stopPropagation() 方法

定义和用法

不再派发事件。

终止事件在传播过程的捕获、目标处理或起泡阶段进一步传播。调用该方法后，该节点上处理该事件的处理程序将被调用，事件不再被分派到其他节点。

语法

```
event.stopPropagation()
```

说明

该方法将停止事件的传播，阻止它被分派到其他 Document 节点。在事件传播的任何阶段都可以调用它。注意，虽然该方法不能阻止同一个 Document 节点上的其他事件句柄被调用，但是它可以阻止把事件分派到其他节点。