目　录

第一部分　命令查询

JavaScript 的语法和命令，按照它们的用法和功能分为几种类型。

对象是 JavaScript 的构造块，它们被用来返回并修改表格、页面、浏览器和程序员定义的变量等的状况。理解对象的一种比较容易的方法是把它看成一个名词。猫，汽车，房屋，计算机和表格都是名词，都可以表示为一个对象（参见“Objects”）。

我们用属性来区分同一个类当中的各个对象——例如，所有的猫对象。属性是形容词性的，它是指一些特性，借助于它可将一个对象和其他对象区分开。以猫为例，这种属性可以是重量，颜色，品种，性情，和当前的活动（参见“Properties”）。

我们用方法来给对象传递消息，有时也用它来还改变对象的属性。例如，一个方法能够用来改变猫当前的活动，比如从吃食改为睡觉，而另一个方法则能用来改变它的重量，从轻改变成重（参见“Methods”）。

下面将列出 JavaScript 的构造块。

---

abs

（方法）

返回一个自变量的绝对值（无符号）。

Math.abs（*argument*）

用法

下面的例子分别返回 10 和 12.5。

```
document.writeln(Math.abs (-10));
John.age.value=12.5
document.writenln(Math.abs(John.age.value))
```

相关条目

Math 的方法。

---

acos

(方法)

返回其自变量的反余弦值（从 0 到 π 弧度）。

Math.acos (*argument*)

用法

自变量应该是-1 到 1 之间的数，若超出有效范围，则返回 0。

相关条目

Math 的方法。

参见 asin，atan，cos，sin 和 tan 方法。

---

action

(属性)

在 HTML<FORM>标记中的动作特性的反映。

document.*formName*.action

document.forms [*index*] .action

用法

action 返回一个由目标 URL 构成的字符串，用于从表格中提交数据。这个值在文件被装载和格式化之前或之后，可以被设置或改变。

在下面例子中，一个称为 outlineForm 的表格，其 action 被设置为在 outlineURL 变量中的 URL。

outlineURL = "http: //www.wossamottau.edu/cgi—bin/outline.cgi"

outlineForm.action = outlineURL

相关条目

form 的属性。

参见 encoding，method 和 target 属性。

---

alert

（方法）

显示一个 JavaScript 的警告对话框，其中有一个 OK 按钮和一段用户定义信息（参见图 QR-1）。

[window.] alert(*AlertMessage*)

图 QR-1 用户必须单击 JavaScript 警告对话框中的 OK 按钮，才能返回文档

用法

用户要继续操作，必须先按警告对话框中的 OK 按钮。

相关条目

window 的方法。

参见 confirm 和 prompt 方法。

---

alinkColor

（属性）

鼠标键被按下但未被释放时，一个链接的颜色。

document.alinkColor

用法

和 JavaScript 中的所有颜色一样，alinkColor 表示成十六进制的 RGB 三色分量或字符串文字。在 HTML 源文件处理完后，alinkColor 不能再改变。下面两个例子都是将颜色设置为 alice 蓝（一种浅蓝色）。

document.alinkColor＝" aliceblue"

document.alinkColor＝" F0F8FF"

相关条目

document 的属性。

参见 bgColor，fgColor，linkColor 和 vlinkColor 属性。

---

anchor

（方法）

创建和显示一个 HTML 超文本目标。

textString.anchor（*anchorName*）

用法

使用 write 或 writeln 方法时，anchor 在当前文档中创建一个 HTML 锚点，在这里，textString 即是用户所看见的，而 anchorName 相当于一个 HTML<ANCHOR>标记的 name 属性。

```
anchorString = " Louie's Place";
document.writeln (anchorString.anchor (" louies _place"))
```

相关条目

string 的方法。

参见 link 方法。

---

anchors

（属性）

当前文档中所有已定义锚点的一个数组。欲知更详细的描述，请参见 anchor 对象。

```
document.anchors [index]
```

用法

如果文档中一个锚点数组的长度为 5，那么这个 anchors 数组就表示为 document.anchors [0] 至 document.anchors [4]。

相关条目

document 的属性。

参见 anchor 对象。

参见 length 和 links 属性。

anchors array

（对象）

一个数组，它包含了一个文档中超文本链所有可能目标的信息。

[*windowName.*] document.anchors [*index*]

用法

anchors array 是一个只读对象，它在 HTML 中用<A>标记来设置。

<A [HREF=URL] NAME="*anchor name*"
[TARGET="*windowName*"] >
anchor text
</A>

包含 HREF 的一个值，可以使 anchor 成为一个链接，并将它加入到 links 数组中去。利用 anchor 方法，新的 anchors 可以由 JavaScript 定义。

为了确定一个文档中有多少个锚点，可以利用 length 属性。

document.anchors.length

document.anchor [*index*] 返回值为零，例如，document.anchor [0] 返回值为空，即使它是第一个用<A NAME = " bob" > All about Bob</A>创建的 anchor。

相关条目

document 的属性。

参见 link 对象和 anchor 方法。

appCodeName

（属性）

返回一个带有浏览器代码名称的只读字符串。

navigator. appCodeName

用法

为了显示当前浏览器的代码名称，使用下面一行语句：

document. write （" The code name of your browser is " + navigator. appCodeName +" ."）

对于 Netscape Navigator 2.0，这将返回：

The code name of your browser is Mozilla.

相关条目

navigator 的属性。

参见 appName、appVersion 和 userAgent 属性。

appName

（属性）

返回一个带有浏览器名称的只读字符串。

navigator. appName

用法

为了显示当前浏览器的应用程序名，使用下面一行语句：

document. write （" The name of your browser is" + naviga-

tor. appName + " .")

对于 Netscape Navigator 2.0，这将返回：

The name of your browser is Netscape.

相关条目

navigator 的属性。

参见 appCodeName，appVersion 和 userAgent 属性。

---

appVersion

(属性)

返回一个带浏览器版本信息的字符串。

navigator. appVersion

用法

appVersion 用来检查客户正在使用的是哪个浏览器版本。它以 *releaseNumber* (*platform*; *country*) 的格式返回。对于 Netscape 2.0 的 Windows 95 版本：

document. write ("The version of your browser is: " + navigator. appVersion + " .")

返回

The version of your browser is: 2.0 (Win95 ; I)

这表明是运行于 Windows95 上的 Navigator 2.0 国际版。国家代码是 U 则表明是美国版，而 I 则表明是国际版。

相关条目

navigator 的属性。

参见 appName，appCodeName 和 userAgent 属性。

asin

（方法）

返回其自变量的反正弦值。

Math.asin（*argument*）

用法

传递一个介于－1 到 1 之间的数给 asin，将返回反正弦值（介于－π/2 弧度到 π/2 弧度之间）。如果传递的数据越界，将返回一个 0。

相关条目

Math 的方法。

参见 acos，atan，cos，sin 和 tan 方法。

atan

（方法）

返回其自变量的反正切值。

Math.atan（*argument*）

用法

用弧度表示该角度大小时，atan 返回的数介于－π/2 到 π/2 弧度之间。其自变量是－1 到 1 之间的一个数，它表示相应的正切值。

相关条目

Math 的属性。

参见 acos，asin，cos，sin 和 tan 方法。

back

（方法）

从历史记录表（history list）中调用前一个 URL。

history. back（）

用法

back 的用法和 history. go（－1）相同。

相关条目

history 的方法。

参见 forward 和 go 方法。

bgColor

（属性）

文档的背景颜色。

document. bgcolor

用法

使用 bgColor 可以取代预先设置在浏览器中的背景颜色。颜色采用十六进制的 RGB 三色分量或字符文字来表示。它可以在任意时间更改。下例允许用户使用选择按钮来设置他们自己的背

景颜色。

```
function newColor (colorString) {
    document.bgColor = colorString
}
...
<FORM NAME = " colors" >
<INPUT TYPE = " radio" NAME = " color" VALUE =
" F0F8FF"
    onClink = " newColor (this.value)" >Alice Blue
<INPUT TYPE = " radio" NAME = " color" VALUE =
" FF4500"
    onClink = " newColor (this.value)" >Ochre
<INPUT TYPE = " radio" NAME = " color" VALUE =
" FFEFD5"
    onClink = " newColor (this.value)" >Papaya Whip
</FORM>
```

相关条目

document 的属性。

参见 alinkColor，fgColor，linkColor 的 vlinkColor 属性。

---

big

（方法）

将一个 string 对象转化为大字体格式。

stringName.big ( )

用法

在功能上，big 的用途相当于用 HTML<BIG>标记将文本

括起来。以下两个例子得到同样的输出：用大字体显示信息" Welcome to my home page. "。

```
var welcomMessage = " Welcome to my home page. "
document. write (welcomeMessage. big ( ))
<BIG>Welcome to my home page. </BIG>
```

相关条目

string 的方法。

参见 fontsize 和 small 方法。

---

blink

（方法）

将一个 string 对象转化为闪烁行格式。

stringname. blink ()

用法

使用 blink 相当于用 HTML<BLINK>标记将文本括起来。下面两个例子都产生一个内容为"Notice" 的闪烁行：

```
var attentionMessage="Notice"
document. write (attentionMessage. blink ())
<BLINK>Notice</BLINK>
```

相关条目

string 的方法。

参见 bold，italics 和 strike 方法。

blur

(方法)

从指定的 form 元素中除去焦点。

document. *formName*. *elementName*. blur ( )

document. forms [*index*] . elements [*index*] . blur ( )

用法

下面一行语句从 feedback 元素中除去焦点：

feedback. blur ( )

这里假设 feedback 定义如下：

<INPUT TYPE="text" NAME="feedback">

相关条目

password，select，text 和 textarea 的方法。

参见 focus 和 select 方法。

bold

(方法)

将一个 string 对象转化为粗体文本格式。

stringName. bold ( )

用法

使用 bold 相当于用 HTML<B>标记将文本括起来。

相关条目

string 的方法。

参见 blink，italics 和 strike 方法。

---

button

（对象）

表格上的一个按钮。

formName.buttonName

forms [*index*] .elements [*index*]

用法

按钮必须定义在<FORM>标记中，它能用来执行一个动作。

<INPUT TYPE="button" NAME="buttonName" VALUE="textOnButton" [onClick="eventHandler"] >

当在一个表格内部进行访问时，表格名是不言自明的。为避免混乱并产生更清晰的代码，最好用带表格元素的表格名。和 onLink 事件处理器一起使用，一个按钮将成为一个定制项，它除了最基本的提交（submit）和复位（reset）功能外，还能启动事件和活动。

下面的按钮，当它被按下时，将调用 validateForm 函数。

<INPUT TYPE="button" NAME="validateForm" VALUE = " Check for Accuracy" onClick = " validateForm (this.form)">

相关条目

form 的属性。

参见 reset 和 submit 对象。

参见 name 和 value 属性。

参见 click 方法。

参见 onClick 事件处理器。

ceil

（方法）

返回最接近的、大于等于自变量的整数。

Math.ceil (*argument*)

用法

ceil 返回一个大于等于其自变量（整数或浮点十进制数）的最小整数。例如：Math.ceil (1.01)

返回 2。

相关条目

Math 的方法。

参见 floor 方法。

charAt

（方法）

从字符串中返回一个字符。

stringName.charAt (*index*)

用法

这个方法接收一个下标作为其参数，返回字符串中处在该下标位置的字符。第一个字符在 0 位置，最后一个字符在长度减 1 的

位置。

例如：

var userName="Bobba Louie"

document.write (userName.charAt (4))

返回 "a"。

相关条目

string 的方法。

参见 indexOf 和 lastIndexOf 方法。

---

checkbox

(对象)

一个表格元素，用户通过单击 checkbox，将它设置为 on 或 off (参见图 QR-2)。

formName.checkboxName

forms [*index*]..elements [*index*]

图 QR-2 一个表格的检查框，只有 true 和 false 两个值。如果这个框没有经过检查，如图所示，其值为假，否则为真。

用法

checkbox 在<FORM>标记中定义。

<INPUT TYPE="checkbox" NAME="checkboxName" VALUE="checkboxValue" [CHECKED] [onClick="*eventHandler*"] > *textToDisplay*

checkbox 的属性和方法可以通过多种方法来使用。

利用 checkbox 的 checked 值，可以看到它当前是被选上

(ture) 还是未被选上 (false)。如果CHECKED选项被用来作为定义的一部分，defalutChecked也返回真。

相关条目

form 的属性。

参见 radio 对象。

参见 checked，defalutChecked，name 和 value 属性。

参见 click 方法。

参见 onClick 事件处理器。

---

checked

(属性)

返回一个布尔标记，它表示单个的检查框或选择按钮的状态。

formName.checkboxName.checked

formName.radioButtonName [*index*] .checked

forms [*index*] .elements [*index*] .checked

用法

checked 返回的一个布尔值 (true 或 false)，它表明一个检查框或选择按钮是否被选中。当该项被检查时，这个值立刻被更新。使用 for...in 语句时，它能检查按钮的状态：

```
function whichOneChecked ( ) {
    var checkedValue=" "
    for (var i in document. formName. radioName) {
        if (document. formName. radioName [i] . checked
        ==true)
          checkedValue = document. formName. radioName
```

```
            [i] .value
        }
    }
```

相关条目

checkbox 和 radio 的属性。

参见 defaultChecked 属性。

---

clear

（方法）

象清屏一样执行清除窗口内容的操作。

document.clear( )

用法

不管窗口是怎样填充的，clear 都将擦除窗口内容。

相关条目

document 的方法。

参见 close，open，write 和 writeln 方法。

---

clearTimeout

（方法）

取消一个超时。

[*windowName.*] clearTimeout (*argument*)

parent. [*frameName.*] clearTimeout (*argument*)

用法

clearTimeout 清除一个超时，这个超时是先前用 setTimeout 方法设置的。一个超时是用一个唯一的超时标识符来设定，也必须用此标识符来清除这个超时：

clearTimeout（waitTime）

相关条目

frame 和 window 方法。

参见 setTimeout 方法。

click

（方法）

模拟鼠标单击。

formName.elementName.click（）

forms［*index*］.elements［*index*］.click（）

用法

一个 click 的效果依赖于所引用的表格元素的类型。

表 4-1 Click 方法和它对表格元素的影响

表格元素	动作
Button、Reset 和 submit	与单击按钮同
Radio	选取选择按钮
Checkbox	标记检查框，并将值设为 on

相关条目

button，checkbox，radio，reset 和 submit 的属性。

close

（方法）

对一个 document 对象，关闭当前的输出流并停止其显示。

对一个 window 对象，则关闭当前窗口。

document. close （）

window. close （）

[*windowName.*] close （）

用法

对于文档，close 关闭当前 winsock 浏览器的动画并在状态条中显示" Document：Done"。对于窗口，和用所有窗口命令一样，window 对象是被假定的。例如：

window. close （ ）

close （ ）

self. close （ ）

均关闭当前窗口。

相关条目

document 和 window 的方法。

参见 clear，open，write 和 writeln 方法。

confirm

（方法）

显示一个 JavaScript 确认对话框（参见图 QR-3）。

window. confirm （ ）

[*windowName.*] confirm （ ）

用法

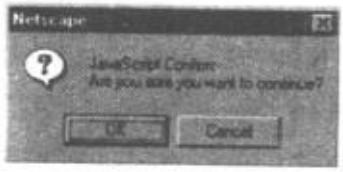


图 QR-3 JavaScript 确认对话框允许用户继续或取消一个操作

和一个附有取消按钮的 alert 一样，confirm 显示一条信息和一个按钮，以便继续操作。如果用户选择 OK，则 confirm 返回 true，如果选择 Cancel，则返回 false。若用户按 OK 键，下面的例子将载入一个新窗口：

```
if (confirm ("Are you sure you want to enter.") {
    tourWindow=window.open ("http:\www.haunted.com
    \","hauntedhouse")
    }
```

相关条目

window 方法。

参见 alert 和 prompt 方法。

---

cookie

(属性)

一小段信息的字符串值，它由 Navigator 保存在客户端的 cookies.txt 文件中。

document.cookie

用法

存储在 cookie 中的值可以通过子字符串 charAt，IndexOf 和 lastIndexOf 来找到。

cookie 是一个特殊的属性，它包含了客户机的状态/状况信息，这些信息可以被服务器访问。包含在个 state 中的是 URL 范围的细节描述，对于这些 URL，该状态是有效的。

若来自客户的 HTTP 请求落在状态对象中所描述的一系列 URL 中时，它将包含状态对象的当前值传输，该传输过程是从客户机到服务器。

这种数据存储的简单格式，允许服务器对客户机提供个别的服务。在线销售商能保存当前在电子售货篮中的商品信息，提供的服务包括发送登记信息和一些自动的功能，例如，打印一个用户 ID。

用户的优先权保存在客户机中，当站点一经接触，即由服务器来检索。对于限制使用的信息，例如采购服务，你也可以在 cookie 信息的生存周期中设置一个时间限。

为了在 HTML 脚本中发送并检查 cookie 的设置，赋一个值给这个属性。

document.cookie="*string*"

CGI 脚本同样是用来设置并检索 cookie 值。要生成 cookie，要求在格式中发送一个 HTTP 标题：

Set-Cookie: NAME=Value; [EXPIRES=date;] [PATH=pathname;] [DOMAIN=domainname;] [SECURE]

当一个针对 cookie 信息的请求发生时，cookie 信息表即被搜索，在表中找出与当前的 URL 匹配的所有的 URL。凡是匹配的，均用以下格式返回：

cookie: NAME1=*string*1; NAME2=*string*2; ...

cookie 是一个任意赋值的名称。如果需要更多的关于 cookie 和其功能的信息，可以参见 Netscape 的 cookie 规格说明，地址在 **http://home.netscape.com/newsref/std/cookie_spec.html**。

相关条目

document 的属性。

参见 hidden 对象。

cos

（方法）

返回其自变量的余弦值。
Math.cos（*argument*）

用法

角度大小必须用弧度表示，结果介于－1 与 1 之间。

相关条目

Math 方法。
参见 acos，asin，atan，sin 和 tan 方法。

Date

（对象）

提供了一组处理日期和时间的方法。
Date.*method*（*parameters*）

用法

这种内部的 Date 对象取代了通常的日期类型。尽管内部的 Date 对象没有任何属性，但它具有一系列方法来设置、修改日期和时间的值。

虽然日期值是按标准格式和语法返回的，但实际存储的值是自 1970 年 1 月 1 日午夜算起的毫秒数。采用这一约定，就禁止使用 1970 年 1 月 1 日以前的日期了。

为了创建一个新的 Date 对象，要使用下述之一的语法约定：
objectName=new Date（）//Creates object with current date and time

objectName=new Date("*month day*, *year* [*hours*: *minutes*: *seconds*]")

//Creates date object with date and time values in string variable or constant

objectName=new Date (*year*, *month*, *day* [, *hours*, *minutes*, *seconds*])

//Creates date object with integer values

在你创建一个 Date 对象时，如果略去时间部分，它的缺省值为午夜（00：00：00）。取得及设置日期和时间信息的方法分为四类：set，get，to 和 parse/UTC。

除了每月中的日以外，日期的各个分量的所有数字表示都从 0 开始计数。除月份外，这么做不会出现问题——月份是用 0（一月）到 11（十二月）来表示的。

标准的日期语法是"Thu，11，Jan 1996 06：20：00 GMT"。美国时区缩写也能被理解；但是为了通用，要规定时差。例如，"Thu，11，Jan 1996 06：20：00 GMT+0530"是在格林威治子午线以西 5 小时 30 分地方。

相关条目

参见 getDate，getDay，getHours，getMinutes，getMonth，getSeconds，getTime，getTimezoneOffset，getYear，parse，setDate，setHours，setMinutes，setMonth，setSeconds，setTime，setYear，toGMTString，toLocaleString 和 toUTC 方法。

defaultChecked

（属性）

一个布尔值（true 或 false），它指明一个检查框或选择按钮是否被缺省检查。

formName. *elementName*. defaultChecked

forms [*index*] . elements [*index*] . defaultChecked

用法

对 defaultChecked 设定一个值会取代一个表格元素的被检查属性。下面的一段代码，通过寻找并设置缺省按钮，把一组选择按钮复位为初始状态：

```
for (var i in menuForm. choice) {
  if (menuForm. choice [i] . defaultChecked) {
    menuForm. choice [i] . defaultChecked=true
}
}
```

按钮的显示不会因 defaultChecked 的变化而受影响，即使其他按钮的状态受到影响。

相关条目

chechbox 和 radio 的属性。

参见 form 对象。

参见 checked 属性。

---

defaultSelected

(属性)

一个表格选择元素中的一项的缺省状态。

formName. elementName. defaultSelected

forms [*index*] . elements [*index*] . defaultSelected

用法

defaultSelected 返回 true 或 false，取决于 CHECKED 选项是否和一个 select 表格元素一道使用。设置这一属性的值可以取代

<OPTION>标记被选的特性。defaultSelected 的语法和行为与 defaultChecked 的相同。

相关条目

options 的属性。

参见 index，selected 和 selectedIndex 属性。

---

defaultStatus

（属性）

显示在一个 Navigator 窗口底部状态条中的缺省信息（见图 QR-4）。

[*windowName.*] defaultStatus

图 QR-4 窗口状态条，它可以保留除链接 URL 以外的预先定义的文本

用法

当状态条中没有其他内容被显示时，将缺省信息放入状态条中。一个优先的或暂时的信息，如一个 anchor 的 mouseOver 事件，将会取代这个缺省信息。例如：

window. defaultStatus="Welcome to my home page"

当鼠标不在一个链上，或者 Netscape 没有执行需要提醒用户注意的动作时，就会显示这条欢迎信息。

相关条目

window 的属性。

参见 status 属性。

defaultValue

(属性)

文本类型表格元素的初始内容。

formName. *elementName*. defaultValue

forms [*index*] . elements [*index*] . defaultValue

用法

对于任何标准 HTML 文本表格域——hidden、password、text、textarea（在<TEXTAREA>标记之间）和 string——defaultValue 返回其初始内容，而不论当前值如何。对口令字元素，由于安全的原因，无论怎样设置 value，该属性都在初始时被设置为空。

相关条目

hidden，password，text 和 textarea 的属性。

参见 value 属性。

document

(对象)

在装入一个页面时，Navigator 创建的一个对象。

document. *propertyOrMethod*

document. *objectName*. *propertyOrMethod*

用法

document 是 JavaScript 的基本对象之一。它包含当前文档的

信息，诸如标题、背景颜色和各个表格。这些属性是在<BODY>标记中定义的。通过 write 和 writeln，document 也提供了向用户显示 HTML 文本的方法。

利用 document 对象的适当的数组，你可以引用一个文档的锚点、表格和链接。这些数组包含了一个文档中每一个 anchor、form 或 link 的入口。

相关条目

window 的属性。

参见 frame 对象；alinkColor，anchor，bgColor，cookie，fgColor，forms，lastModified，linkColor，links，location，referrer，title 和 vlinkColor 属性。

参见 clear，close，open，write 和 writeln 方法。

参见 onLoad 和 onUnload 事件处理器。

E

（属性）

自然对数的底。

Math.E

用法

亦称为欧拉常数，这个值近似为 2.71828。

相关条目

Math 的属性。

参见 LN2，LN10，LOG2E，LOG10E，PI，SQRT1_2 和 SQRT2 属性。

elements

（属性）

一个对象数组，它按 HTML 源程序顺序包含一系列表格元素。

formName.elements [*index*]

forms [*index*] .elements [*index*]

用法

数组下标从 0 开始，到 form 元素个数减 1 为止。关于 elements 的全面讨论，参见 elements 对象。

相关条目

form 的属性。

参见 elements 对象。

elements array

（对象）

一个按源程序顺序排列的 form 元素数组。

formName.elements [*index*]

forms [*index*] .elements [*index*]

用法

这个数组包含一个表格的所有元素。它可以通过表格名或 forms 数组进行访问。表格元素包括按钮、检查框、选择按钮、文本和文本区域对象。可以通过元素下标引用这些元素。

例如，一个表格包含 2 个文本域、3 个选择按钮和 2 个按钮。

那么，在 elements 数组中，可以用 formName. elements[0]到 formName. elements [6] 引用这些元素。注意，下标计数从 0 开始，而不是从 1 开始。

为了获得一个表格中元素的个数，可以使用 length 属性。元素数组中一个成员的值是用以创建它的完整 HTML 文本。

元素也可以用元素名来引用。例如，一个称为 newPassword 的口令字元素，是一个 HTML 页面上的第二个表格元素，它的值可用下述三种方式进行访问：

formName. elements [1] . value

formName. elements ["newPassword"] . value

formName. newPassword. value

使用只读 elements 数组，不能设定或修改这些值。

相关条目

form 的属性。

参见 length 属性。

---

encoding

(属性)

返回一个反映 MIME 编码类型的字符串。

formName. encoding

forms [*index*] . encoding

用法

MIME 编码类型是在一个 HTML<FORM>标记的 enctype 特性中设置的。HTML 中的 MIME 编码标准尚未确立，但进展情况和草案存放在加利福尼亚大学(Irvine)信息与计算机科学系，可以按下面的地址在 Web 上找到：http: //www. ics. uci. edu/pub/ietf/html/。

相关条目

form 的属性。

参见 action，method 和 target 属性。

---

escape*

（方法）

返回参数的 ASCII 码。

escape（*argument*）

用法

HTML ASCII 码基于 ISO Latin－1 字符集，形式为 %xxx，其中 xxx 是十进制 ASCII 码。这个方法不与任何其他对象关联，但它实际上是 JavaScript 语言的一个固有部分。对于字母、数字、字符（字母和数字）将返回它们自身，而对于符号将返回它们的 ASCII 码。

```
document.write (escape ("Hi!"))
```

返回

```
Hi%21
```

相关条目

参见 unescape 方法。

---

eval*

（方法）

对作为数字表达式的一个字符串进行求值。

eval（*string*）

用法

这个内部函数把一个字符串或数字表达式作为它的参数。如果是字符串，eval 将试图将它转变成数字表达式，然后求出表达式的值，并返回这个值。

```
var  x=10
var  y=20
document.write (eval ("x + y"))
```

这个方法也可以用来执行作为一部分被包含在字符串中的 JavaScript 命令。

```
var doThis="if (x==0) {alert ('Your maximum has been
reached')}"
function checkMax () {
    x++;
    eval (doThis)
}
```

当需要把一个日期从一种格式（总是一个字符串）转换为数值表达式或数字的时候，这个方法可以派上用场。

exp

（方法）

返回一个指数函数值。

Math.exp (*argument*)

用法

exp 返回 E 的自变量次幂。

相关条目

Math 的方法。

参见 log 和 pow 方法。

参见 E 属性。

fgColor

（属性）

前景文本的颜色。

document. fgColor

用法

JavaScript 中的颜色用一个十六进制的 RGB 三色分量或一个字符串文字来表示。在文档开始处理以后，fgColor 的值不能改变，尽管可以利用 fontcolor 方法改变文本各个单独部分的颜色。

fgColor 有两种形式：

document. fgColor="aliceblue"

document. fgColor="F0F8FF"

其效果同于<BODY>标记中的 TEXT 特性：

<BODY TEXT="aliceblue">

相关条目

document 的属性。

参见 alinkColor，bgColor，linkColor 和 vlinkColor 属性。

参见 fontcolor 方法。

fixed

（方法）

将调用该方法的字符串转化为定距（fixed－pitch）字体格式。

stringName. fixed ()

用法

使用 fixed 等同于把一个字符串参数放在 HTML＜TT＞标记内。

相关条目

string 的方法。

---

floor

（方法）

返回小于参数的最接近整数。

Math. floor (*argument*)

用法

把一个整数或浮点十进制数传递给该方法时，返回小于等于这个参数的整数。例如：

Math. floor (2. 99)

返回 2。

相关条目

Math 的方法。

参见 ceil 方法。

---

focus

（方法）

将焦点对准一个特定表格元素。

formName. *elementName*. focus ( )

forms [*index*] . elements [*index*] . focus ( )

用法

focus 利用表格和元素的名字给出元素的焦点。从那个焦点，可以利用 JavaScript 命令来输入一个值，或者用户可以完成该输入。

相关条目

password，select，text 和 textarea 的方法。

参见 blur 和 select 方法。

---

fontcolor

（方法）

取代一个字符串对象的缺省前景颜色。

stringName. fontcolor ( )

用法

fontcolor 将字符串对象转换为指定的颜色，该颜色作为参数，以十六进制 RGB 三色分量或一个字符串文字表示。fontcolor 的使用类似于<FONT COLOR=COLOR>的使用。

```
myDog="Brown";
document.write (myDog.fontcolor ("sienna"))
```

相关条目

string 的方法。

fontsize

（方法）

将字符串对象转换为指定的字体大小。

stringName.fontsize（*argument*）

用法

这个方法利用一个整数，通过<FONTSIZE=SIZE>标记，选用七种已定义字体大小当中的一种。如果传递给该方法的参数是字符串，那么字体大小就相对于<BASEFONT>标记中设置的值来改变。

参数代表字体大小。如果参数是整数，它就是字体的大小，并且必须是1到7的数字。如果参数是字符串，字体的大小就相对于基本字体来改变。

相关条目

string的方法。

参见big和small方法。

form（forms array）

（对象）

一个对象，代表页面上的一个表格。

document.*formName*

document.forms[*index*]

用法

form 是 document 对象的一个属性。一个文档中的每一表格都是独立的、各不相同的对象，可以把该表格的名字作为表格对象来引用它。表格对象也可以表示为一个数组，该数组是在这些表格通过 HTML 标记定义时被创建的。

如果一个文档中的第一个表格称作 orderForm，那么可以用 document. orderForm 或 document. forms [0] 来引用它。如果没有命名这个表格，则只能利用它在表格数组中的下标来引用它。用 length 属性可以得到一个页面上的表格的数目。

document. forms. length

表格的各个元素是通过它们的名字或利用 elements 数组来引用的。

document. formName. elements [index]

表格数组是一个只读对象。试图通过下面这样的语句：

document. forms [1] ="OldGuestBook"

来设置值是无效的。

表格数组中每一项的值在语法上的表示和 HTML 标记的语法类似。例如，一个名为 userInfo 的表格，其 form 对象的值是<OBJECT USERINFO>。

相关条目

document 的属性。

参见 hidden 对象，action，elements，encoding，forms，method，name 和 target 属性。

参见 submit 方法。

参见 onSubmit 事件处理器。

---

forms

(属性)

一个对象数组，它对应于按HTML源程序顺序排列的若干个已命名的表格。

document.forms

用法

forms是document对象的一个属性。它包含了文档中每个form对象的入口。详细的讨论参见form对象

相关条目

document的属性。

参见form对象。

参见length属性。

---

forward

（方法）

装入在URL历史记录表上的下一个文档。

history.forward（）

用法

这个方法与history.go（1）相同。

相关条目

history的方法。

参见back和go方法

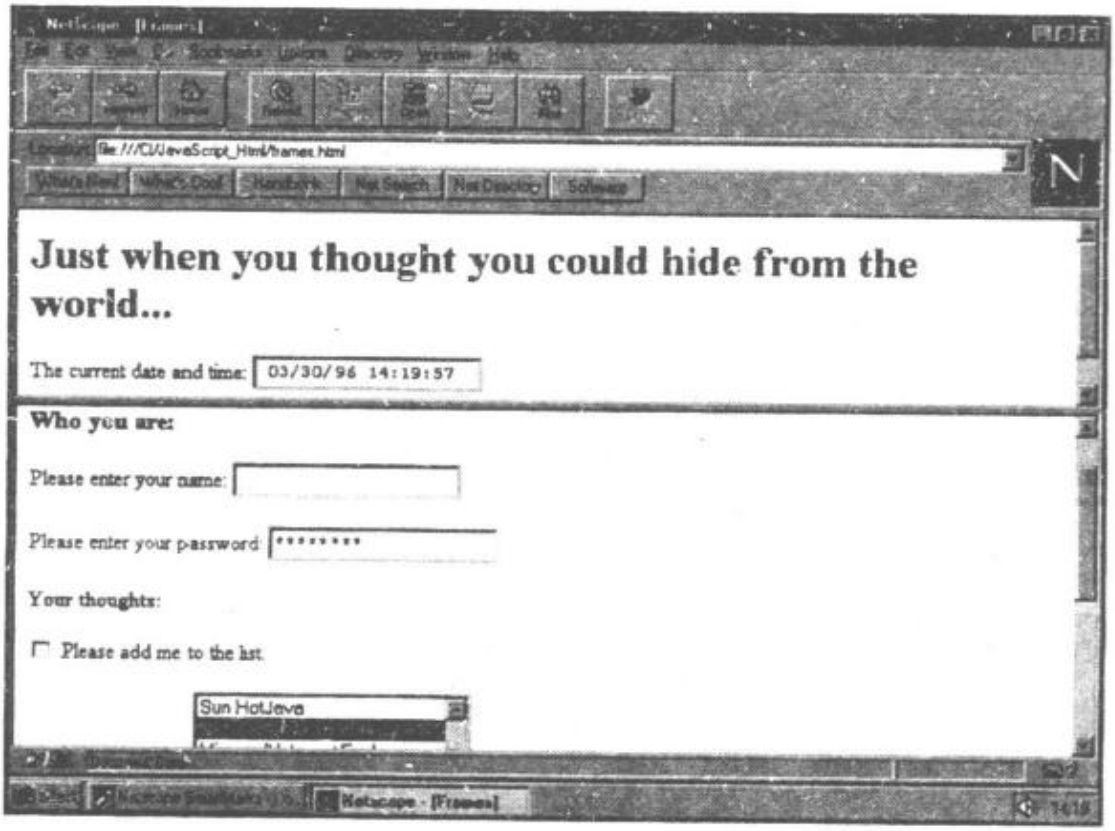
frame

（对象）

包含若干HTML子文档的一个窗口，其中的子文档可以相互独立地滚动（见图QR-5）。

[*windowName.*] [parent.] *frameName*

[*windowName.*] [parent.] frames [*index*]



图QR-5　在一个导航器窗口里的一组框架。每个框架包含它自己的HTML文档

用法

框架可以指向不同URL，并且可以是同一窗口内任何其他框架所指的目标。每个frame都是一个利用<FRAMESET>标记定义的window对象，它确定了构成页面的布局。页面由一个父

HTML 文档中定义。所有子文档都是父文档的孩子。

如果一个 frame 包含对 SRC 和 NAME 特性的定义，那么通过利用 parent 对象，由 parent.frameName 或 parent.frames[index] 的形式，可以将该 frame 和它的兄弟区分开。

相关条目

window 的属性。

参见 document 和 window 对象。

参见 defaultStatus，frames，parent，self，status，top 和 window 属性。

参见 setTimeout 和 clearTimeout 方法。

---

frames

（属性）

对应于各个子框架窗口的一个数组，其中的子框架窗口是使用<FRAMESET>标记创建的。

[*windowName.*] [parent.] *frameName*

[*windowName.*] [parent.] frames [*index*]

用法

为了得到一个窗口中子框架的数目，可以使用 length 属性。欲知关于 frames 数组的更多信息，参见 frame 对象。

相关条目

window 的属性。

参见 frame 对象。

参见 length 属性。

getDate

（方法）

返回月中的日期。

Date. getDate （）

用法

这个方法返回一个 1 到 31 之间的数字，它表示月中的日期，这是 JavaScript 中少数不从 0 开始的项之一。

```
endOfTheWorld = new Date ("January 11, 1996 06: 18: 00")
document. write (endOfTheWorld. getDate ()) //Returns 11
```

相关条目

Date 的方法。

参见 setDate 方法。

getDay

（方法）

用 0（星期日）至 6（星期六）之间的一个整数返回星期几。

Date. getDay （）

dateName. getDay （）

用法

没有与本方法对应的 setDay 命令，因为一旦日期值确定，星期几就自动计算出来了。

相关条目

Date 的方法。

getHours

（方法）

返回一天中的钟点数。
Date. getHours ()
dateName. getHours ()

用法

返回值按 24 小时格式，从 0（午夜）到 23（晚 11 时）。

相关条目

Date 的方法。
参见 setHours 方法。

getMinutes

（方法）

返回分钟数，一个 0 到 59 之间的整数。
Date. getMinutes ()
dateName. getMinutes ()

用法

象其他日期函数一样，getMinutes 直接返回时间的一个成分。

```
endOfTheWorld =new Date ("January 11, 1996 06: 18: 00")
document. write (endOfTheWorld. getMinutes ()) //
Returns 18
```

相关条目

Date 的方法。

参见 setMinutes 方法。

getMonth

（方法）

返回一年中的月份。

Date. getMonth ()

dateName. getMonth ()

用法

返回的月份是一个 0（一月）到 11（十二月）之间的整数，而不是一个字符串。这个值容易造成混乱，因为它不符合传统的月份计数方法。当把这个值显示到屏幕上，或者从用户输入那里得到一个月份的时候，一定要做转换。

```
function toReality () {//converts month to 1－12 numbering system
    this +=1
}
function toConvention () {//converts month to 0－11 numbering system
    this --=1
}
```

相关条目

Date 的方法。

参见 setMonth 方法。

getSeconds

（方法）

返回秒。

Date. getSeconds ()

dateName. getSeconds ()

用法

秒被返回时是一个 0 到 59 之间的整数。

相关条目

Date 的方法。

参见 setSeconds 方法。

---

getTime

(方法)

返回一个代表日期对象当前值的整数。

Date. getTime ()

dateName. getTime ()

用法

. 这个值是自 1970 年 1 月 1 日午夜算起的毫秒数。该值可用来比较两个日期值的时间长度。

对于涉及日期计算的函数，建立一组用毫秒定义的分、时、日的变量是很有用的。

```
var dayMillisec=1000 * 60 * 60 * 24 //1,000 milliseconds
    x 60 sec x 60 min x 24 hrs
var hourMillisec=1000 * 60 * 60 //1,000 milliseconds x 60
    sec x 60 min
var minuteMillisec=1000 * 60 //1,000 milliseconds x 60 sec
```

相关条目

Date 的属性。

参见 setTime 方法。

---

getTimezoneOffset

（方法）

以分为单位返回客户机与格林威治时间（GMT）之间的差值。

Date.getTimezoneOffset ()

dateName.getTimezoneOffset ()

用法

除了夏令时情况之外，这个值是一个常数。

相关条目

Date 的方法。

---

getYear

（方法）

返回日期对象的年份与 1900 之差。

Date.getYear ()

dateName.getYeat ()

用法

尽管可以在一个字符串中用四位数字向一个日期对象传递年份，但从 getYear 返回的年份值是一个两位的数字。例如，对于

1996年，将会返回96。

相关条目

Date的方法。

参见setYear方法。

go

（方法）

装入一个在历史记录表中指定的文档。

history.*go*（*argumentOrURL*）

用法

这个方法可以通过URL来引用历史记录表中的文档，也可以通过相对于该表中当前位置的值来引用。如果URL是不完整的，就使用最接近的匹配。查找是大小写不敏感的。

相关条目

history的方法。

参见back和forward方法。

hash

（属性）

返回一个字符串，该字符串是一个URL中从hash符（#）开始的那一部分。

document.*linkName*.hash

document.links［*index*］.hash

document.location.hash

用法

以‘#’开始的字符串表示一个 anchor 名字的片段。它可以用来设置一个 hash 属性，尽管把整个 URL 设置为 href 属性是最安全的。如果在当前位置没有发现 hash，将返回一个出错信息。

相关条目

link 和 location 的属性。

参见 anchor 对象。

参见 host，hostname，href，pathname，port，protocol 和 search 属性。

---

hidden

（对象）

一个禁止在 HTML 表格上显示的文本对象。

document. *formName. hiddenName*

document. forms[*index*]. elements[*index*]. *propertyOrMethod*

用法

hidden 对象可以用来为客户/服务器通信传递名字—值对，这是除 cookie 之外的又一种传递方式。二者的差别在于，对于 cookie 客户端在会话中是不变的，而 hidden 对象对于各个表格是特定的。

在一个函数中，通过对 hidden 对象的 value 属性赋予新的内容，可以改变该对象的初始内容。

```
<INPUT TYPE="hidden" NAME="failedTries" VALUE
="0">
... statements ...
```

```
function setRetry () {
    document. userPasswordForm. failedTries. value++
}
```

每次调用 setRetry 函数，名为 failedTries 的 hidden 对象的值就增加 1。这也是 JavaScript 灵活类型转换的一个例子。尽管初始值是一个字符串，当脚本遇到算术运算符时，便试图将该值转变为整数。

相关条目

form 的属性。

参见 cookie，defaultValue，name 和 value 属性。

---

history

（对象）

先前访问过的 URL 历史记录表，它与浏览器中的 Go 菜单相同。

document. history

用法

这个对象来自 Go 菜单，并包含有上一次访问过的页面中的 URL 信息。它的方法可用来导航至历史记录表中的任意点。

要确定历史记录表中项的个数，使用 length 属性：

document. history. length

利用 forward 和 back 方法，可以实现在历史记录表上相对移动的导航。它类似于使用菜单条中的导航按钮。

document. history. forward ()

document. history. back ()

go 这个方法可以在历史记录表中做到除简单向前和向后移

动之外的跳跃移动。这类似于从 Go 菜单中直接选择一个地址。

```
document.history.go (-2) //loads the page two links ago
```

当你指定一个窗口时，其他窗口或框架中的导航仍是可以控制的。下面的例子就是从 content 框架的 history 记录表中，装入一个新的页面。注意，框架名取代了 document 符号。

```
parent.content.history.back () //loads the previous page in
the frame
```

相关条目

document 的属性。

参见 location 对象。

参见 length 属性。

参见 back，forward 和 go 方法。

---

host

(属性)

返回一个 URL 的 hostname 和 port 属性组合成的一个字符串：

```
location.host
linkName.host
links [index] .host
```

用法

提供一个方法，来观察和改变地址类型对象的 URL 主机属性。如果没有指定一个端口，则 host 属性等同于 hostname 属性。

```
location.host="www.montana.com: 80"
```

相关条目

link 和 location 的属性。

参见 hash，hostname，href，pathname，port，protocol 和 search 属性。

hostname

（属性）

返回或改变带有 URL 的域名或 IP 地址的一个字符串。

location. hostname

linkName. hostname

links [*index*] . hostname

用法

这个属性除了不包含端口信息之外，其他与 host 属性相似。当端口属性为空时，host 属性与 hostname 属性是相同的。

虽然 hostname 可以在任何时候被改变，但建议同时修改整个 URL。如果没有找到 hostname，那么将返回出错信息。

相关条目

link 和 location 的属性。

参见 hash，host，href，pathname，port，protocol 和 search 属性。

href

（属性）

返回带有当前文档的整个 URL 的一个字符串。

location. href

linkName. href

links [*index*] . href

用法

所有其他的 location 和 link 属性都是 href 的子字符串，它能够在任何时候被修改。当前文档的 URL 是利用 document. write 来反映到屏幕上。

document. write ("You are here: " + window. location. href)

相关条目

link 和 location 的属性。

参见 hash，host，hostname，pathname，port，protocol 和 search 属性。

---

index

（属性）

返回一个选择（select）元素选项的下标。

formName. *selectName*. options [*index*] . index

forms [*index*] . elements [*index*] . options [*index*] . index

用法

在选择（select）对象中该选项的位置，其编号从 0 开始。

相关条目

select（options）array 的属性。

参见 defaultSelected，selected 和 selectedIndex 属性。

---

indexOf

（方法）

返回一个特定字符或字符串的位置。

stringName. indexOf (*character/string*, *startingPoint*)

用法

查寻从一个特定位置开始。字符串的第一个字符的位置被规定为0，而最后一个字符为字符串的长度减1。如果该字符串没有找到，此方法返回－1。

startingPoint 缺省时是 0。

```
if (navigator. appVersion. indexOf ('Unix') ! =－1)
    return true
```

相关条目

string 的方法。

参见 charAt 和 lastIndexof 方法。

---

isNaN *

(方法)

检验一个参数是否不是数字。

isNaN (*argument*)

用法

仅对 UNIX 平台，如果被检验的参数不是一个数字，那么这个独立的函数返回一个 true。在除 windows 以外的所有平台上，当被检验的参数不是一个数字时，parseFloat 和 parseInt 返回 NaN。

相关条目

参见 parseFloat 和 parseInt 方法。

italics

（方法）

将一个字符串对象转化为斜体格式。

stringName. italics ()

用法

italics 等同于将一个字符串括在 HTML<I>标记中。

相关条目

string 的方法。

参见 blink，bold 和 strike 方法。

lastIndexOf

（方法）

返回在一个 string 中从串尾开始搜索的一个字符或字符串的下标。

stringName. lastIndexOf ()

用法

返回一个下标，它是在一个 string 对象中，从字符串末尾或用户指定的下标开始，通过反向查寻的方法，得到的一个字符或字符串的下标。如果该字符串没有找到，返回一个－1。

```
if (navigator.appVersion.lastIndexOf ('Win') ! =－1)
    return true
```

相关条目

string 的方法。

参见 charAt 和 indexOf 方法。

---

lastModified

(属性)

一个包含当前文档最后一次修改日期的只读字符串。

document. lastModified

用法

这个属性是基于源文件特性的，这个字符串是按照 JavaScript 所用的标准形式来构成的。一个通常的用法是：

```
dateModified=="This page last modified on " +
document. lastModified
document. write (dateModified. small ())
```

相关条目

document 的属性。

---

length

(属性)

一个整数，它反映了一个对象与长度有关或与大小有关的属性。

formName. length

forms. length

formName. elements. length
forms [*index*] . length
[*windowName*.] *frameName*. length
frameRef. *frames*. length
history. length
radioName. length
selectName. length
selectName. options. length
stringName. length
windowName. length
anchors. length
links. length

用法

length 返回值的意义，由它所应用到的数组或对象来决定。

表 4-2 长度属性的结果

对象/数组	测出的属性
history	历史记录表的长度
string	字符串的长度值（整数）；空串长度值为 0
radio	选择按钮的个数
anchors、forms、frames	数组中元素的个数
links、options	

相关条目

anchors，elements，forms，frame，frames，history，links，options，radio，string，和 window 的属性。

link

（方法）

创建一个到其他 URL 的超文本链接。

stringName. link（argument）

用法

通过定义<HREF>属性和表示到用户的链的文本，来创建一个新的超链接。

```
linkText="Wossamatta University";
linkURL="http://www.wossamatta.edu/";
document.write("Rocky's alma mater is " + linkText.link(linkURL))
```

相关条目

string 的方法。

参见 anchor 方法。

link（links array）

（对象）

定义成一个到其他 URL 的超文本链的文本或图像。

```
document.linkName
document.links[index]
```

用法

一个 link 是一个 location 对象，因此它和一个 location 对象一样，具有相同的属性和方法。

如果一个名字被定义给这个对象，那么它也就被定义作为一个锚点，并被给予一个在 anchors 数组中的入口。

```
<A HREF='http://www.cnet.com/'>c|net's front door</A>
<A HREF='http://www.cnet.com/' NAME='cnet'>c|net's front door</A>
```

在上面的例子中，第一行只建立了一个在 links 数组中的入口。再添加 NAME 属性，另一个入口又在 anchors 数组中被建立起来。

link 对象是只读的。如果要建立一个新的超文本链接，要用 link 方法（string 的方法）。

相关条目

document 的属性。

参见 anchor 对象。

参见 hash，host，hoatname，href，length，pathname，port，protocol，search 和 target 属性。

参见 link 方法。

参见 onClick 和 onMouseOver 事件处理器。

linkColor

（属性）

在文档中显示超链接颜色。

document.linkColor

用法

颜色被表示成一个 16 进制的 RGB 三色分量或表示成一个字符串文字。超链接颜色对应于 HTML<BODY>标记中的 link 特性，它在文档被处理之后不能再改变。

```
document.write ("The current link color is " +
document.linkColor)
```

相关条目

document 的属性。

参见 alinkColor，bgColor，fgColor 和 vlinkColor 属性。

links

(属性)

表示若干 link 对象的一个数组。

document. links [*index*]

用法

在 HTML 中，links 是用<A HREF=URL>标记来定义的。这些都被反映在 links 属性中，这里的第一个 link 等同于 document. links [0]。欲知更详细的描述，请参看 link 对象。

相关条目

参见 link 对象。

参见 anchors 和 length 属性。

LN2

(属性)

表示 2 的自然对数的一个常数。

Math. LN2

用法

这个值近似为 0. 69315。

相关条目

Math 的属性。

参见 E, LN10, LOG2E, LOG10E, PI, SQRT1_2 和 SQRT2 属性。

LN10

（属性）

表示 10 的自然对数的一个常数。
Math.LN10

用法

这个值近似为 2.30259。

相关条目

Math 的属性。
参见 E,LN2,LOG2E,LOG10E,PI,SQRT1_2和SQRT2属性。

location

（对象）

当前文档的完整 URL 信息。
[*WindowName.*] [*frameName.*] *location.propertyName*
parent.[*frameName.*] *location.propertyName*

用法

location 是用来确定任何现用文档的 URL，包括在其他浏览器窗口或浏览器框架中的那些 URL。如果窗口对象被省略，那么就是指当前窗口。

location 的每个属性都包含了这个 URL 的一个不同部分。共有 URL 的六个部分反映在 location 对象中：

protocol：//*hostname*：*port*/*pathname search*#*hash*

传输协议包括地址的开始部分（http，mailto，ftp 等）直到冒号（包括冒号）为止。JavaScript 还包含有若干附加协议。

javascript 协议计算冒号后的表达式，并试图装入其结果的字符串值。如果计算没有结果或者没有定义，则保留当前页面。

javascript：parent.content.history.go（－1）

about 协议提供了三种方法来查看关于浏览器的信息。单独地，它同选择help 和 About 是一样的。另外两种方法（cache 和 plugins）反映了高速缓存的当前状态和关于所建立的插入应用程序的信息。

about：cache

about：plugins

不要把这个对象（它是 window 的一个属性）和 document 的 location 属性混淆起来。一般来说，它们反映了同样的值，但是 location 属性不能改变，而 location 对象的属性是可以改变的。

相关条目

window 的属性。

参见 history 对象。

参见 hash，host，hostname，href，location，pathname，port，protocol，search 和 target 属性。

---

location

（属性）

返回一个带有当前文档的 URL 的一个字符串：

document.location

用法

这个只读属性（document.location）不同于 location 对象的属性（window.location.propertyName）。后者可以被改变。

相关条目

document 的属性。

参见 location 对象。

log

（方法）

返回一个大于0的正数值表达式的自然对数（以 E 为底）。

Math.log（*expression*）

用法

一个超出范围的数永远返回－1.7976931348623l6e＋308。

相关条目

Math 的方法。

参见 exp 和 pow 方法。

LOG2E

（属性）

一个常数，它是 E 的以2为底的对数。

Math.LOG2E

用法

这个值近似为1.44270。

相关条目

Math 的属性。

参见 E，LN2，LN10，LOG10E，PI，SQRT1_2和 SQRT2属性。

LOG10E

（属性）

一个常数，它是E的以10为底的对数。

Math. LOG10E

用法

这个值近似为0. 43429。

相关条目

Math 的属性。

参见 E，LN2，LN10，LOG2E，SQRT1_2和 SQRT2属性。

Math

（对象）

一个提供常数和数学函数的内部对象。

Math. *property*

Math. *method* （*argument*）

用法

Math 对象分为两个部分：.包含常数的属性和实现函数的方法。例如，为了在一个等式中存取 PI 的值就用：

Math. PI

标准三角函数，对数函数和指数函数都在其中。三角函数中的所有参数都必须用弧度。另外还提供了几个比较操作，例如象 max 这样的操作用来确定两个数中较大者。

因为 Math 对象的目的是为数学运算提供一种工具，所以没有提供一个构造，来生成复合的 Math 对象。

对于需要大量使用 JavaScript 数学函数和常数的函数来说，把 Math 包含在每个等式中是很麻烦的。with 语句可以简化这种情况下的语法。注意下面两段代码的区别。它们都实现了同样的运算。

```
function Hard () {
    circleArea = Math.PI * (radius ^ 2);
    radians = (degrees/360) * Math.PI;
    result = Math.cos (radians);
}
function Easy () {
    with Math {
        circleArea = PI * (radius ^ 2);
        radians = (degrees/360) * PI;
        result = cos (radians);
    }
}
```

相关条目

参见 E，LN10，LN2，PI，SQRT1—2 和 SQRT2 属性。

参见 abs，acos，asin，atan，ceil，cos，exp，floor，log，max，min，pow，random，round，sin，sqrt 和 tan 方法。

---

max

（方法）

返回两个参数中较大的一个。

Math.max (*argument1*, *argument2*)

用法

可以接受数字文字或变量的任意组合，并返回最大的值，例如：

```
firstNum = 1
secondNum = 100
Math.max (firstNum, secondNum)
```

返回 100。

相关条目

Math 的方法。

参见 min 方法。

method

（属性）

反映 HTML<FORM>标记的方法特性。

formName.method

forms [*index*].method

用法

返回值是 get 或 post。它在任何时候都可以被设置成一个新值。

第一个函数返回表格对象的当前值，而第二个函数将 method 设置成 newMethod 的内容。

```
function getMethod (formObj) {
    return formObj.method
```

```
}
function setMethod (formObj.newMethod) {
    formObj.method = newMethod
}
```

相关条目

form 的属性。

参见 action，encoding 和 target 属性。

min

（方法）

返回两个参数中较小的一个。

Math.min (*argument1*, *argument2*)

用法

可以接受文字和变量的任意组合作为它的参数，并返回较小的数，例如：

```
firstNum = 1
secondNum = 100
Math.min (firstNum, secondNum)
```

返回 1。

相关条目

Math 的方法。

参见 max 方法。

name

(属性)

返回带有该对象 name 特性的一个字符串。

objectName. name

frameRef. name

frameRef. frames. name

radioName [*index*]. name

selectName. options. name

windowRef. name

windowRef. frames. name

用法

这个属性的特性依赖于所用的对象。它可以在任意时候修改。

对于 button，reset 和 submit 对象，这个属性指的是内部名称，而非屏幕上的标记。

例如，用以下行打开一个新窗口：

indexOutline = window.open ("http: //www.wossamatta.com/outline. html","MenuPage")

并且发出如下命令：

document. write (indexOutline. name)

然后 JavaScript 返回 MenuPage，它被规定为名字特性。

对于选择按钮来说，组中的每个按钮其名字都是相同的，而每个单个按钮是由它们的下标位置来识别的。

相关条目

button，checkbox，frame，password，radio，reset，select，submit，text，textarea 和 window 的属性。

参见 value 属性。

navigator

（对象）

包含客户当前浏览器的信息。

navigator

用法

navigator 对象返回关于浏览器版本的信息，诸如版本号，名字和用户代理的标题。一个通常用法是确定客户正在使用的平台类型。这样浏览器的特征，例如换行符和随机数，就能被正确使用。

```
function UnixMachine () {
    if (navigator.appVersion.lastIndexOf('Unix') != -1)
        return true
    else
        return false
}
```

相关条目

参见 link 和 anchors 对象。

参见 appName，appCodeName，appVersion 和 userAgent 属性。

onBlur

(事件处理器)

出现在 select，text 和 textarea 表格元素失去焦点时。

<INPUT TYPE="*elementType*" onBlur="*function*">

用法

当用户离开表格元素时，一个模糊(blur)事件能够检查输入。这是不同于 onChange 的，onChange 只能出现在域的内容被改变的情况下。

<INPUT TYPE="textarea" VALUE="" NAME="feedback" onBlur="checkSignature (this.value)">

相关条目

select，text 和 textarea 的事件处理器。

参见 focus 和 blur 方法。

参见 onChange 和 onFocus 事件处理器。

---

onChange

(事件处理器)

出现在 select，text 和 textarea 表格元素的值被改变且失去焦点时。

<INPUT TYPE="*elementType*" onChange="*function*">

用法

这个事件对确认用户表格输入是非常有效的。

<INPUT TYPE="text" VALUE="MT" NAME="state" onChange="checkAvailability (this.value)">

相关条目

select，text 和 textarea 的事件处理器。

参见 onBlur 和 onFocus 事件处理器。

---

onClick

（事件处理器）

出现在一个可选对象被鼠标选中时。

<INPUT TYPE="*elementType*" onClick="*function*">

用法

onClick 为页面上的按钮和其他对象提供了各种功能。按钮可以被用来在提交之前确认输入或者计算一个表格或等式的结果。点击其他对象（例如检查框和选择按钮），就能激活对其他信息的捕捉。

下面这个例子是将 overtime 表格的内容发送给 howRich 函数。

```
<FORM NAME="overtime">
Full days worked: <INPUT TYPE="text" VALUE="0"
NAME="days" SIZE=3>
Hours worked: <INPUT TYPE="text" VALUE="0"
NAME="hours" SIZE=30>
<INPUT TYPE="button" VALUE="compute" NAME
="computeWage"onClick="howRich (this.form)">
</FORM>
```

相关条目

button，checkbox，radio，link，reset 和 submit 的事件处理

器。

onFocus

(事件处理器)

出现在用户为了输入而选择 select，text 或 textarea 时。

<INPUT TYPE="*inputType*" onFocus="*function*">

用法

当用户使用制表键或用鼠标点击一个输入区域时，表格元素得到焦点。在一个域中进行选择时，将导致一个 select 事件。

onFocus 函数的一种用法是，当一个项被首次选中时，弹出帮助信息。

相关条目

select，text 和 textarea 的事件处理器。

参见 onBlur 和 onChange 事件处理器。

onLoad

(事件处理器)

出现在一个文档完成对一个窗口或一幅画面的装入时。

<BODY onLoad="*function*">

<FRAMESET onLoad="*function*">

用法

当浏览器完成对一个窗口，或<FRAMESET>标记中所有框架的装入时，产生一个装入事件。

相关条目

window 的事件处理器。

参见 onUnload 事件处理器。

---

onMouseOver

(事件处理器)

出现在鼠标被放在一个超链接上时。

<A HREF="*URL*" onMouseOver="*function*">*linkText*</A>

用法

onMouseOver 出现在鼠标箭头放在一个 link 对象上时，为了和 status 或 defaultStatus 属性一道运行，该事件处理器必须返回 true。

```
<A HREF="http://home.netscape.com/"
onMouseOver="window.status='Netscape Home'; return true">
Netscape</A>
```

相关条目

link 的事件处理器。

onSelect

(事件处理器)

当在表格元素中的文本是高亮时，出现 onSelect。

<INPUT TYPE="*textType*" onSelect="*function*">

用法

一个选择事件是通过在一个 text 或 textarea 域中，选择部分或全部文本来激活的。

相关条目

text 和 textarea 的事件处理器。

onSubmit

(事件处理器)

出现在用户通过提交按钮提交一个表格时。

<TAG onSubmit = "*function*">

用法

用户提交一个表格时 onSubmit 被触发。除 false 外的任意返回值，包括省略了 return 语句的情况，都提交该表格。为了让代码更清楚易懂，建议对两种选择情况都加上 return 语句。

```
<FORM onSubmit = "feedbackSubmit ( )">
... form elements...
```

```
</FORM>
function feedbackSubmit ( ) {
...statements...
if (! validData) {
    return true }
else {
    return false; }
}
```

相关条目

form 的事件处理器。
参见 submit 对象。
参见 submit 方法。

onUnload

(事件处理器)

出现在用户退出一个文档时。
<BODY onUnload = "*function*">
<FRAMESET onUnload = "*function*">

用法

当几个卸载事件都包含在一个框架关系中时，操作的顺序从孩子到父亲。

例如，一个卸载事件涉及了两个文档和父<FRAMESET>标记，该标记装载了这两个文档。当子文档改变时，它的卸载事件被激发，但框架集的卸载不受影响。当用户选择一个选项，该选项用一个新的源文件取代父文档，那么其顶端的卸载事件即被

激发。

相关条目

window 的事件处理器。

参见 onLoad 事件处理器。

---

open

(方法)

创建一个新文档或窗口实例。

document.open ([*MIMEtype*])

window.open (*"URL"*, *"windowName"* [," *windowFeatures"*])

用法

对于一个文档,open 打开一个流来收集 write 或 writeln 方法的输出。如果 MIME 类型是一个 text 或 image 形式，例如 text/html 或者是 image/gif，文档即被打开，以供对这些文本或图像进行布局。否则，该流被引导到一个 plug—in。如果已经有一个文档在一个目标窗口中存在，那么打开方法将清除它。这个流通过使用 document.close ( ) 方法来结束。

对于一个窗口，open 用与上面类似的方法打开一个新的浏览窗，以便从浏览器菜单中选择文件或 NEW WEB 浏览器。利用 URL 参数，open 方法将一个文档载入到这个新的窗口当中；否则这个窗口为空。当表格被当成事件处理器的一部分时，它必须包括这个窗口对象；否则，文档只是被假定的。

窗口的功能部件通过一个由逗号隔开的选择表定义，其中等于 1 或等于 yes 为允许，等于 0 或等于 no 为禁止。窗口的功能部件包括：toolbar (工具条)，location (定位)，dirctories (目录)，

status（状态），menubar（菜单条），scrollbars（滚动条），resizable（重置大小），copyhistory（复制历史），width（宽）和 height（高）。

相关条目

document 和 window 的方法。

参见 clear，close，write 和 writeln 方法。

---

options

（属性）

该选项的数组是一个 select 表格元素的属性。这个数组是通过在一组<SELECT>标记中，使用<OPTION>标记来创建的。

formName . *selectName*. options [*index*]

forms [*index*]. elements [*index*]. options [*index*]

用法

第一个选项的下标是 0，第二个为 1，以此类推。欲知更多详细的信息，请参见 select 对象。

相关条目

参见 select 对象。

---

parent

（属性）

指当前文档中正在执行调用的文档，框架由〈FRAMESET〉标记创建。

parent

parent. *frameName*

parent. frames [*index*]

parent. property

用法

使用 parent 就能够访问用同样的<FRAMESET>标记创建的其他框架。例如，两个被调用的框架，分别称为 index 和 contents。利用下面的语法，index 框架可以写入到 contents 框架中：

parent. contents. document. write (" Kilroy was here. ")

相关条目

frame 和 window 的属性。

---

parse

(方法)

取一个日期字符串，例如. Jan 11，1996，并返回从 1970 年 1 月 1 日午夜开始算起的毫秒数。

用法

这个函数能够用来设置基于字符串值的日期。当发送一个带时间的字符串时，它返回时间值。

由于 parse 是 Date 的一个静态函数，它总是这样使用：Date. parse ( )，而不作为一个被创建的 Date 对象的方法来使用。

Date. parse ("Jan 11，1996 ");

```
Today = new Date ( );
    Date.parse (Today.toLocaleString ( ))
```

相关条目

Date 的方法。

参见 UTC 方法。

---

parseFloat*

(方法)

```
parseFloat (string)
```

用法

parseFloat 分析一个字符串参数，如果第一个字符是一个正号、负号、十进制小数点、指数或数字，它将返回一个浮点数。

如果 parseFloat 碰到一个非有效字符，那么返回值到那个位置为止，而忽略所有后继字符。若第一个字符不是有效字符，parseFloat 基于平台返回以下两个值中的一个：

Windows	0
Non—Windows	NaN

相关条目

参见 isNaN 和 parseInt 方法。

---

parseInt*

(方法)

分析一个字符串参数，并基于指定的基数或底数之上返回一个整数。

parseInt (*string* [, *radix*])

用法

若基数为 10 将值转化为十进制，基数为 8 将值转化为八进制，基数为 16 将其转化为十六进制。当基数为 16，大于 10 的数，用字母 A—F 来替代数字。利用基数 2 进行二进制数的转化。

浮点数值被转化为整数。字符串求值的规则和 parseFloat 的规则相同。

如果基数省略或它与首字符相矛盾时，JavaScript 假定基数是基于字符串的第一个字符。

表 4-3 基于首字符的缺省基数

字符	基数
0	8（八进制）
0x	16（十六进制）
其他	10（十进制）

相关条目

参见 isNaN 和 parseFloat 方法。

password

（对象）

在<HTML>表格中的口令字元素（参见图 QR-6）。

document. *formName* . *passwordName*

document.forms [*index*].elements [*index*]

图 QR-6 由于保密性，用星号将一条口令字屏蔽

用法

一条口令字即一个文本域。为了保密起见，当用户输入时，它被星号屏蔽。

当页面载入时，作为 HTML 定义的一部分被包含的任一缺省值都被清 0。这就防止了有意或无意的保密性破坏。即使 defaultValue 属性对 password 有效，它总是返回空值。

password 对象的值可以在一个脚本中，通过程序计算得出。不过由于明显的安全原因，建议不要使用一个文字。

相关条目

form 的属性。

参见 text 对象。

参见 defaultValue，name 和 value 属性。

参见 focus，blur 和 select 方法。

---

pathname

(属性)

返回一个 URL 的路径部分。

location.pathname

link.pathname

links [*index*].pathname

用法

虽然 pathname 可以在任何时间被更改，但是利用 href 属性同时更改整个 URL，总是比较安全的。

相关条目

link 和 location 的属性。

参见 hash，host，hostname，href，port，protocol 和 search 属性。

PI

（属性）

返回 π 值。

Math. PI

用法

Math. PI 的值近似为 3.14159，这是圆周与其直径的比。

```
circumference = 2 * Math. Π * radius
area = Math. Π * Math. pow (radius, 2)
```

相关条目

Math 的属性。

参见 E，LN2，LN10，LOG2E，LOG10E，SQRT1—2 和 SQRT2 属性。

---

port

（属性）

返回一个 URL 地址的端口号。

location. port

link. port

links [*index*]. port

用法

端口号是 href 中 host 属性的一个子字符串。

相关条目

link 和 location 的属性。

参见 hash，host，hostname，href，pathname，protocol 和 search 属性。

---

pow

（方法）

返回底数的指数幂。

Math. pow（*argument* 1）

用法

很多语言是利用插入操作符（^）来计算指数幂，JavaScript 用它自己的方法来做到这一点。插入符被用在 JavaScript 中，是计算按位异或操作。

相关条目

Math 的方法。

参见 exp 和 log 方法。

prompt

（方法）

显示一个用来接受用户输入的提示对话框（参见图 QR-7）。

[*windowName.*] prompt (*message* [*inputDefault*])

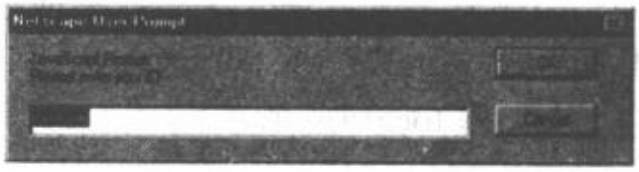


图 QR-7 提示对话框，用来取得表格外的用户输入

用法

如果没有对 inputDefault 指定一个初始值，对话框显示＜UNDEFINED＞值。产生图 QR.7 所示的提示，需要使用下面一行代码：

```
var userid = prompt ("Please enter your ID", " ")
```

相关条目

window 的方法。

参见 alert 和 confirm 方法。

protocol

（属性）

返回文件访问方法。

location. protocol

link. protocol

links [*index*].protocol

用法

用这种属性返回的字符串是URL中的开始部分，这部分直到冒号为止（包括冒号在内）。这是URL中指明访问方法（http，ftp，mailto等等）的那部分。

about	关于客户浏览器的信息
ftp	下载文件的文件传送协议地址
http	超文本传送地址，它是www（全球信息网）的基础
mailto	一个email（电子邮件）地址
news	一个usenet消息站点
file	指本地机器上的一个文件
javascript	加在一套JavaScript命令之前

相关条目

link和location的属性。

参见hash，host，hostname，href，pathname，port和search属性。

---

radio

（对象）

一组选择按钮。

formName.*radioName* [*index*]

forms [*index*].elements [*index*]

用法

radio 对象在 HTML<FORM>标记中被创建，表示选择按钮。一组选择按钮使用户能从一组选项中选择一项。

当用选择按钮名引用一个对象时，下标是由具有相同 name 属性的按钮组成。当用 elements 数组访问一个选择按钮时，每个按钮都是下标中一个独立的项。

相关条目

form 的属性。

参见 checkbox 和 select 对象。

参见 checked，defaultChecked，index，length，name 和 value 属性。

参见 click 方法。

参见 onClick 事件处理器。

---

random

（方法）

返回 0 到 1 之间的随机数（仅限于 UNIX）。

Math.random（）

用法

random 方法只在 UNIX 平台上才有效。Windows 和 Macintosh 用户需要使用另一种替代的方式来产生随机数。在书后的任务参考中有这种用法的实例。

相关条目

Math 的方法。

referrer

（属性）

文档的 URL 地址，它引导到当前文档。

document. referrer

用法

返回一个只读字符串，它包含了文档的完整的 URL，该文档调用了当前的文档。referrer 可以和一个 CGI 脚本一起使用来跟踪用户如何被链接到一个页面。

```
document. write ("Click <A HREF = \' " +document.referrer + " \' >here</A>to go back from whence you come." ")
```

相关条目

document 的属性。

reset

（对象）

将一个表格恢复到其缺省值的按钮。

formName . resetButtonName

forms [*index*]. elements [*index*]

用法

这个按钮与 HTML 中的复位按钮相互关联，它将所有 form

对象复位其缺省值。

一个 reset 对象必须在一个<FORM>标记中创建，并且不能通过 onClick 事件处理器来控制。当按钮被单击时，表格即复位。不过，事件处理器能用复位来调用其他操作。

相关条目

form 的属性。

参见 button 和 submit 对象。

参见 name 和 value 属性。

参见 click 方法。

参见 onClick 事件处理器。

---

round

（方法）

将一个数字舍入成与它最接近的一个整数。

Math.round (*argument*)

用法

返回一个浮点自变量的舍入值，如果小数部分大于等于 0.5，则舍入为大于它的最小整数；若是小于 0.5，则舍入为小于它的最大整数。

```
Math.round (2.1) //return 2
Math.round (2.9) //return3
```

相关条目

Math 的方法。

search

（属性）

返回属于 URL 的任意查询信息。

location. search

linkName. search

links [*index*]. search

用法

返回一个字符串，它包含任何属于 URL 的查询信息。查询数据前加一个问号。这些数据是包含在文档 URL 中的最后一项。字符串中的信息按以下的方法格式化：

? *elementName* = *element* + *value*

象 href 属性的所有子字符串一样，search 能在任何时候改变。

相关条目

link 和 location 的属性。

参见 hash，host，hostname，href，pathname，port 和 protocol 属性。

select

（方法）

选择一个指定表格元素的输入区域。

formName . *elementName* . select （ ）

forms [*index*]. elements [*index*]. *select* ( )

用法

和 focus 方法一起使用，JavaScript 可以将一块区域高亮显示，并且将光标定位，以便用户输入。

相关条目

password ，text 和 textarea 的方法。

参见 blur 和 focus 方法。

select（options array）

（对象）

在一个 HTML 表格上的一个选择表或滚动表（参见图 QR-8）。

formName . *selectName*

forms [*index*]. elements [*index*]

formName . *selectName* [*index*]. options [*index*]

forms [*index*]. elements [*index*]. options [*index*]

用法

一个选择表使用户能从一个表中选择任一项。一个滚动表允许从一个表中选择一项或多项，这要在输入标记中使用 MULTIPL 特性来实现。

当使用时若不带选项数组(options array)，select 对象指的是

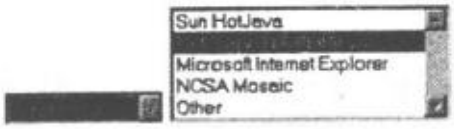


图 QR-8 左边是 HTML 表格中的一个下拉式选择表，它允许一个选项；右边是滚动表，允许多个选项

整列表，可使用诸如 length 和 name 之类 的可选属性。value 和 selectedIndex 指出在选择表中的当前被选项或在滚动表中的第一个被选项。

相关条目

form 的属性。

参见 radio 对象。

参见 length，name，option 和 selectedIndex 属性。

参见 blur 和 focus 方法。

参见 onBlur，onChange 和 onFocus 事件处理器。

对于 select 的 options 属性，参见 defaultSelected，index，selected，text 和 value 属性。

---

selected

(属性)

返回一个布尔值 (true 或 false)，它表示在一个 select 对象中一个选项的当前状态。

formName . *elementName* . [options [*index*].] selected

forms [*index*].elements [*index*]. [options [*index*].] selected

用法

被选属性在任何时间都可更改，并且显示将立即更新来表现新值。被选属性对于用 multiple 特性创建的 select 元素是很有用的。

利用 selected 属性，你能观查或修改在 options 数组中的任一元素值，而无需更改数组中的其他元素的值。

相关条目

options 的属性。

参见 defaultSelected，index 和 selectedIndex 属性。

---

selectedIndex

(属性)

返回一个整数，指出一个被选项的下标。

formName . *elementName* . [options [*index*].] selected

forms [*index*].elements [*index*].[options [*index*].] selected

用法

selectedIndex 属性对于那些 select 元素很有用，这些元素无需使用<SELECT>标记中的 MULTIPLE 特性来创建。如果 se-

lectedIndex 在 MULTIPLE 特性被设置时被求值，该属性只返回第一个选项的下标。设置该属性将清除元素中其他被选中的选项。

相关条目

select 和 options 的属性。

参见 defaultSelected，index 和 selected 属性。

---

self

（属性）

指的是当前窗口或表格。

self

用法

当处理同名的 window 和 form 属性时，self 对消除二义性很有用。

相关条目

frame 和 window 的属性。

参见 window 属性。

---

setDate

（方法）

设置月中的日。

Date.setDate (*argument*)

dateName.setDate (*argument*)

用法

setDate 用一个 1 到 31 之间的整数来设置日期对象中的日。

```
endOfTheWorld = new Date("January 11,1996 06:18:00")
endOfTheWorld.setDate (26)
document.write (endOfTheWorld.getDate ()) //Returns 26
```

相关条目

Date 的方法。

参见 getDate 方法。

---

setHours

(方法)

设置当前时间的点钟数。

Date.setHours (*argument*)

dateName.setHours (*argument*)

用法

setHours 按军队时间用一个 0 (午夜) 到 23 (晚 11 点) 之间的整数来设置一天中的点钟数。

相关条目

Date 的方法。

参见 getHours 方法。

setMinutes

（方法）

设置当前时间的分钟数。

Date. setMinutes （*argument*）

dateName. setMinutes （*argument*）

用法

用一个 0 到 59 之间的整数来设置日期对象中的分钟数。

相关条目

Date 的方法。

参见 getMinutes 方法。

setMonth

（方法）

设置一个日期对象的月份值。

Date. setMonth （*argument*）

dateName. setMonth （*argument*）

用法

它使用 0（一月）到 11（十二月）之间的一个整数。这是 Date 对象中不符合通常计数习惯的一项。当把月份值从 JavaScript 格式转变为能被用户理解的形式时，一定要作调整。

相关条目

Date 的方法。

参见 getMonth 方法。

---

setSeconds

（方法）

设置秒。

Date. setSeconds（*argument*）

dateName. setSeconds（*argument*）

用法

用一个 0 到 59 之间的整数来设置一个日期对象中的秒。尽管 Date 对象以毫秒为单位跟踪时间，但是当输入一个特定时间的时候，所允许的最小单位是秒。

相关条目

Date 的方法。

参见 getSeconds 方法。

---

setTime

（方法）

设置一个日期对象的值。

dateName. setTime（*argument*）

用法

这是一个 date 对象的基本形式。它返回自 1970 年 1 月 1 日午夜算起的毫秒数。尽管没有必要知道这一数字，但是它可作为将一个日期对象的值拷贝给另一个日期对象的一种简单方法。

```
endOfTheWorld = new Date (userGuess)
checkDate = new Date ()
checkDate.setTime (endOfTheWorld.getTime ())
```

相关条目

Date 的方法。

参见 getTime 方法。

SetTimeout

（方法）

在指定的一段时间后对一个表达式求值，时间以毫秒为单位表示。

```
[window.] setTimeout (timerID)
[windosName.] setTimeout (timerID)
```

用法

超时不是长期重复的。例如，设置一个超时为 3 秒，就会在 3 秒之后对表达式求值一次，而不是每 3 秒钟求一次值。

为了循环地调用 setTimeout，就要在本方法所调用的函数中将超时复位。在下面的例子当中，对 startclock 函数的调用就建立了一个循环，每个循环中的动作是清除超时、显示当前时间、设置超时，从而以秒为单位重复地显示时间。

```
<SCRIPT>
```

```
var timerID = null;
var timerRunning = false;

function stopclock () {
  if (timerRunning) clearTimeout (timerID);
  timerRunning=false;
}
function startclock () {
  stopclock ();
  showtime ();
}
function showtime () {
  var now = new Date ();
  document.clock.display.value = now.toLocaleString ();
  timerID = setTimeout ("startclock ()", 1000);
  timerRunning = true;
}
</SCRIPT>

<BODY onLoad="startclock ()">
<FORM NAME="clock">
<INPUT ITEM = text NAME = "display" VALUE = "
Standby for the time">
</FORM>
</BODY>
```

相关条目

window 的方法。

参见 clearTimeout 方法。

SetYear

（方法）

设置当前日期中的年份。

Date.setYear（*argument*）

dateName.setYear（*argument*）

用法

setYear 需要一个两位整数来表示年份与 1900 之差。

相关条目

Date 的方法。

参见 getYear 方法。

sin

（方法）

返回一个自变量的正弦值。

Math.sin（*argument*）

用法

自变量是用弧度表示的角的大小，返回值在－1 到 1 之间。

相关条目

Math 的方法。

参见 acos，asin，atan，cos 和 tan 方法。

small

（方法）

stringName.small ()

用法

small 方法利用 HTML<SMALL>标记将一个 string 对象转化为一个小字体格式。

相关条目

string 的方法。

参见 big 和 fontsize 方法。

sqrt

（方法）

返回一个正的数字表达式的平方根。

Math.sqrt (*argument*)

用法

如果自变量的值超出范围，返回值为 0。

SQRT1_2

(属性)

1/2 的平方根。

Math. SQTR1 _ 2

用法

1/2 的平方根也可以表示为 2 的平方根的倒数（近似为 0.707）。

相关条目

Math 的属性。

参见 E，LN2，LN10，LOG2E，LOG10E，PI 和 SQRT2 属性。

SQRT2

(属性)

2 的平方根。

Math. SQRT2

用法

这个常数的值近似为 1.414。

相关条目

Math 的属性。

参见 E，LN2，LN10，LOG2E，LOG10E，PI 和 SQRT1—2 属性。

status

(属性)

指定一条优先的或暂时的消息显示在状态条上。

window. status

[*windowName*]. status

用法

状态条位于窗口的底部。通常，一条用户状态信息是由一个anchor 的 mouseOver 事件来触发的。为了当鼠标放于一个链上时在状态条中显示一条信息，其用法是：

< A anchorDefinition onMouseOver = " window. status = 'Your message.'; return true" > link </A >

请注意嵌套引号的使用和 return true 的使用，这些是操作所要求的。

一旦状态信息被唤醒，它就会一直保持着，直到鼠标被放在另一个链上。

相关条目

window 的属性。

参见 defaultStatus 属性。

strike

(方法)

将一个字符串对象转化为带删除线的文本。

stringName.strike ()

用法

使用 strike 与使用 HTML<STRIKE>标记的作用相同。

相关条目

string 的方法。

参见 blink，bold 和 italics 方法。

---

string

(对象)

用双引号或单引号定义的一串字符。

stringName

用法

例如：

myDog = "Brittany Spaniel"

返回一个称为 myDog 的字符串对象，其值为 Brittany Spaniel。引号不是字符串值的一部分，它们只用于界定一个字符串。

利用能返回字符串的一种变体的方法，可以对字符串对象的值进行处理，例如 myDog.toUpperCase()返回 BRITTANY SPANIEL。字符串对象也包含一些能返回字符串的 HTML 变体的方法，诸如 bold 和 italics。

相关条目

参见 text 和 textarea 对象。

参见 length 属性。

参见 anchor，big，blink，bold，charAt，fixed，fontcolor，fontsize，indexOf，italics，lastIndexOf，link，small，strike，sub，substring，sup，toLowerCase 和 toUpperCase 方法。

sub

（方法）

将一个字符串对象转化成下标文本格式。

stringName.sub()

用法

使用 sub 与使用 HTML<SUB>标记的作用相同。

相关条目

string 的方法。

参见 sup 方法。

submit

（对象）

HTML 页面上的一个提交按钮。

formName.buttonName

forms [*index*].elements [*index*]

用法

单击该按钮，将把表格提交给 action 属性所指定的程序。该

按钮是在HTML<FORM>标记中创建的，并且它总是装入一个新的页面，如果没有规定一个动作，新页面和当前页面相同。

相关条目

form 的属性。
参见 button 和 reset 对象。
参见 name 和 value 属性。
参见 click 方法。
参见 onClick 事件处理器。

submit

(方法)

执行与单击提交按钮相同的动作。
formName. submit ()
forms [*index*]. submit ()

用法

提交来自表格的信息，依赖于 METHOD 特性的 get 特性或 post 特性。

相关条目

form 的方法。
参见 submit 对象。
参见 onSubmit 事件处理器。

substring

(方法)

根据两个下标，返回一个字符串对象的子集。

stringName.substring (*index*1, *index*2)

用法

如果下标相等，将返回空字符串。不论下标的顺序如何，子字符串都将是从小下标到大下标那段字符来构成。

相关条目

string 的方法。

sup

(方法)

将一个字符串对象转化为上标文本格式。

stringName.sup ()

用法

使用 sup 与使用 HTML<SUP>标记的作用相同。

相关条目

string 的方法。

参见 sub 方法。

tan

(方法)

返回自变量的正切值。

Math. tan (*argument*)

用法

自变量是用弧度表示的角的大小。

相关条目

Math 的方法。

参见 acos, asin, atan, cos, sin 方法。

target

(属性)

规定一个窗口名的字符串，这个窗口是用来在提交一个表格之后公布相应反应的。

formName. target

forms [*index*]. target

location. target

link. target

links [*index*]. target

用法

通常 target 用于查看表格提交的目的地，它也可用于查看或

改变一个链接的目的地。对于一个链接，target 返回一个指明窗口名的字符串，这个窗口显示一个被选中的超文本链接的内容。

homePage. target = "http: //www. wossamatta. edu"

你必须使用文字来设置 target 属性，使用 JavaScript 表达式和变量来设置是无效的。

相关条目

form，link 和 location 的属性。

参见 action，encoding 和 method 属性。

---

text

（对象）

HTML 表格上的单行输入域（见图 QR-9）。

formName. textName

forms [*index*]. elements [*index*]

图 QR-9 文本输入框是表格内的对象

用法

text 对象接受字符或数字输入。规定一个 text 对象的值为新的内容，可以更新该对象。

相关条目

form 的属性。

参见 password，string 和 textarea 对象。

参见 defaultValue，name 和 value 属性。

参见 focus，blur 和 select 方法。

参见 onBlur，onChange，onFocus 和 onSelect 事件处理器。

---

text

（属性）

返回在一个 select 对象中，接在<OPTION>标记后的文本的值。

formName. *selectName*. options [*index*]. text

forms [*index*]. elements [*index*]. options [index]. text

用法

text 也能用来改变选项的值，此时，一个重要的限制是：虽然值已经被改变，但它在屏幕上的显示没有改变。

相关条目

options 的属性。

---

textarea

（对象）

textarea 与一个 text 对象相似，但外加许多行（参见图 QR-

10)。

formName.textAreaName

forms [*index*].elements [*index*]

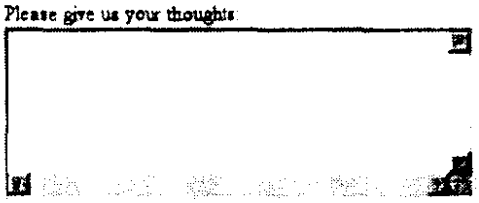


图 QR-10 textarea 对象能从用户处接受多行文本

用法

一个 textarea 对象也能通过对其值赋予新内容来加以改变。在赋予新值之后，屏幕立刻被更新。

```
<FORM>
<ITEM INPUT = textarea NAME =" sponsorMessage" VALUE="And now a brief message form our sponsor.">
</FORM>
...
<SCRIPT>
sponsorMessage.value = "Now is the time \r\n for everybody to get up \r\n and run away."
</SCRIPT>
```

注意换行符\n 的用法，一个新行的实现依赖于平台。对于 Windows，它是由\r\n 组合成的；而对于 Macintosh 和 UNIX，它是\n。

当在一个表格中定义一个文本区域时，通过把文本包含在两个<TEXTAREA>标记之间，你能输入一个缺省值。下面的例子把三行文本和一个空行装到一个文本区域元素中。

```
<BR><TEXTAREA NAME="user—comments" ROWS=4 COLS=40>
Enter your comments here.
Or just press the submit button to let us
know you liked what you saw.
```

</TEXTAREA>

相关条目

form 的属性。

参见 password，string 和 text 对象。

参见 defaultValue，name 和 value 属性。

参见 focus，blur 和 select 方法。

参见 onBlur，onChange，onFocus 和 onSelect 事件处理器。

title

（属性）

返回在 HTML<TITLE>标记中设置的只读值。

document. title

用法

如果一个文档未包含一个标题，这个值就为 null。

相关条目

document 的属性。

toGMTString

（方法）

使用国际互联网（Internet）的 GMT 约定，将一个日期对象转换为一个字符串。

Date.toGMTString()
dateName.toGMTString()

用法

这个字符串转换返回一个包含 GMT 格式的时间字符串，它会随平台而变化。

例如，如果 today 是一个日期对象：

today.toGMTString ()

那么将返回的字符串是"THU，11 JAN 1996 06：05：15"。实际的格式也许会因平台不同而变化，时间和日期基于客户机。

相关条目

Date 的方法。

参见 toLocaleString 方法。

---

toLocaleString

(方法)

使用本地约定将一个日期对象转换为一个字符串。

Date.toLocaleString ()
dateName.toLocaleString ()

用法

用这种方法返回的日期字符串依赖于定义在客户机上的优先顺序，诸如 mm/dd/yy hh：mm：ss。

相关条目

Date 的方法。

参见 toGMTString 方法。

toLowerCase

(方法)

将一个字符串中的所有字符转换成小写字符。

stringName.toLowerCase ()

用法

toLowerCase 方法的结果整个地显示成小写字符，尽管字符串的实际内容并没有改变。

相关条目

string 的方法。

参见 toUpperCase 方法。

top

(属性)

最高层浏览器窗口。

top

top.*frameName*

top.frames [*index*]

用法

也称做一个前驱(ancestor)或 Web 浏览器窗口(Web browser window)，top 属性指的是包含 frames 或嵌套 framesets 的最高优先级窗口。

相关条目

window 的属性。

---

toUpperCase

(方法)

将一个字符串中的所有字符转换成大写字符。

stringName.toUpperCase ()

用法

尽管它影响字符串的直接显示，但 toUpperCase 并不影响它的对象的值。

相关条目

string 的方法。

参见 toLowerCase 方法。

---

unescape *

(方法)

返回一个基于其 ASCII 码值的字符。

unescape(*string*)

用法

这个返回值是表示为%xxx 格式的一个字符串，这里的 xxx

可以是 0 到 255 之间的十进制数，或 0x0 到 0xFF 之间的十六进制数。

相关条目

参见 escape 方法。

---

userAgent

（属性）

作为从客户机向服务器发送的 HTTP 协议部分的首部，它用来标识客户机类型。

navigator. userAgent

用法

返回值的语法与 appVersion 情况一样，外加浏览器应用程序代码名。

codename/releaseNumber (platform; country)

相关条目

navigator 的属性。

参见 appName，appVersion 和 appCodeName 属性。

UTC

（方法）

返回一个在通用协调时间（universal coordinated time，简作

UTC）下的日期的，从1970年1月1日午夜算起的毫秒数。

Date.UTC（*year*，*month*，*day*[，*hrs*][，*min*][，*sec*])

用法

UTC始终是从同一个日期开始计算，因此永远使用Date.UTC()的方式，而不用创建一个日期对象。如果包含月份值的话，不要忘记JavaScript的月份数字是从0到11。

相关条目

Date的方法。

参见parse方法。

---

value

（属性）

返回一个对象的值。

formName.buttonName.value

formName.resetName.value

formName.submit.value

formName.checkboxName.value

formName.radioName.value

formName.hiddenName.value

formName.textName.value

formName.textareaName.value

formName.selectName.value

formName.passwordName.value

forms[*index*].elements[*index*].value

用法

一个对象的值依赖于所用对象的类型。

改变一个 text 或 textarea 对象的值，将会立刻改变屏幕上的显示。所有其他 form 对象的值被改变时，不改变屏幕显示。

表 4-4 各种表格对象的值

对象	值
button，reset，submit	显现在屏幕上的值的特性，但它不是按钮的名字
checkbox	如果该项被选中，为 On；没有被选中，为 Off
radio	值的字符串反映
hidden，text，textarea	域的内容
select	选项值的反映
password	返回一个有效的缺省值，如果由用户来修改，则是加密形式

相关条目

button，checkbox，hidden，options，password，radio，reset，submit，text 和 textarea 的属性。

对于 password，text 和 textarea，参见 defaultValue 属性。

对于 button，reset 和 submit，参见 name 属性。

对于 options，参见 defaultSelected，selected，selectedIndex 和 text 属性。

对于checkbox 和radio，参见checked 和defaultChecked 属性。

vlinkColor

(属性)

返回或设置被访问的链接的颜色。

document.vlinkColor

用法

象所有颜色一样，vlinkColor 使用十六进制 RGB 三色分量或

一个字符串文字。在文档已被格式化以后不能设置该属性。为了取代浏览器缺省颜色，颜色的设置应和<BODY>标记中的 onLoad 事件处理器一道使用：

<BODY onLoad="document.vlinkColor= 'alicebiue'">

相关条目

document 的属性。

参见 alinkColor，bgColor，fgColor 和 linkColor 属性。

---

window

（对象）

与一个打开的 Navigator 窗口相关的最高优先级对象，它可由 JavaScript 访问。

window

self

top

parent

windowName

propertyName

methodName（*parameters*）

用法

当包含用于整个窗口属性的一个页面被装入时，导航器创建一个 *window*。对每一个 *document*，*location* 和 *history* 对象来说，它是最高层的对象。因为它的存在是被假定的，所以当访问它的对象，属性和方法时，可以不必引用窗口的名字。

例如，下面的两行可以得到同样的结果（在状态行中打印一条信息）：

status="Go away from here."

window.status="Go away from here."

引用一个窗口有多种方法，这取决于它与当前位置的关系，如表 4-5 所示。

表 4-5　窗口别名

窗口名称	参考
window、*self*	包含当前文档的窗口。当这些别名被省略时，当前文档仍然是被假定的。例外情况发生在描述事件处理器时，在这里，诸如 *open* 和 *close* 方法必须和所规定的窗口或框架一道使用
top	指的是最高层窗口。对于通过多重<*FRAMESET*>标记产生的父一子一子关系是非常有用的
parent	包含<*FRAMESET*>标记的窗口，该标记创建了当前窗口
windowName	用于引用 *HTML* 标记中的窗口。当要使用一个窗口的属性和方法时，就用这个窗口变量的名字

一个新窗口是利用 *open* 方法来创建：

aNewWindow = window.open ("URL", "Window _ Name" [, *"windowFeatures"*])

变量名是用来指出窗口的属性和方法。窗口名被用于一个 form 或 anchor 标记的目标参数。窗口功能部件表（表 4-6）控制着浏览器的外部特征和功能（参见图 QR-11）。

表 4-6　窗口功能部件特性

选项	用法
toolbar	包含标准工具条，包括 forward，back，home 和 print 按钮
location	创建一个位置对象
directories	在 Netscape 中，包含用于标准链接的按钮表，如 What' s New，What' s Cool 和 Handbook 等按钮
status	在屏幕的底部创建一个状态条
menubar	包含在屏幕顶部的菜单条，它含有诸如 File，Edit 和 View 等项目
scrollbar	如果文档超出屏幕尺寸，就增加滚动条
resizable	允许用户修改窗口大小
width	按象素点来初始化窗口的宽度
height	按象素点来初始化窗口的高度

如果没有列出功能部件，那么所有的都被缺省包含。如果任何功能部件是明确定义的，那么不包含在功能部件表中的，将被缺省地排除。

相关条目

参见 document 和 frame 对象。

参见 defaultStatus，frames，parent，self，status，top 和 window 属性。

参见 alert，close，confirm，open，prompt，setTimeout 和 clearTimeout 方法。

参见 onLoad 和 onUnload 事件处理器。

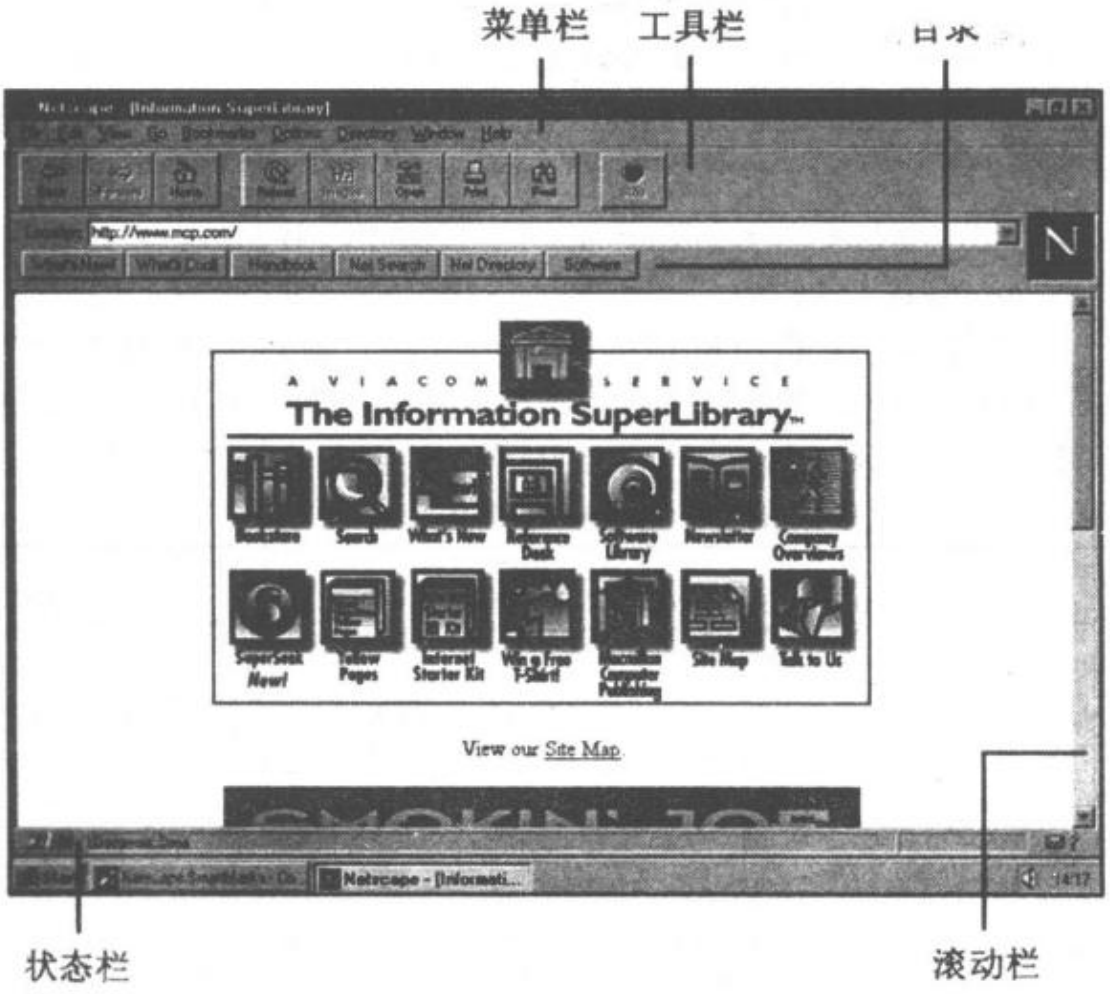


图 QR-11 一个导航器窗口的某些功能部件，它们是通过 window.open 方法来控制的

window

(属性)

是当前窗口的同义词。

frameName. window

[*windowName*.] window

用法

window 属性是用来消除同名的 window 和 form 对象之间的多义性。虽然 window 也适用于当前框架，但使用 self 属性会较少产生歧义。

相关条目

frame 和 window 的属性。

参见 self 属性。

write

(方法)

将一行或多行写入一个文档窗口。

document. write (*string*)

用法

写到一个窗口中的字符串能包含 HTML 标记和 JavaScript 表达式(包括数字，字符串和逻辑值)。write 方法不把一个换行符(
 或 /n) 加到输出的末尾。在 write 被一个事件处理器调用时，如果没有建立一个用来输出的新窗口，那么当前文档被清除。

相关条目

document 的方法。

参见 close，clear，open 和 writeln 方法。

writeln

（方法）

将一行或多行写入一个文档窗口，后面跟一个换行符。

document. writeln (*string*)

用法

writeln 和与其相关的 *write* 一样，它能包含 *HTML* 标记和 *JavaScript* 表达式（包括数字，字符串和逻辑值）。在 *writeln* 被一个事件处理器调用时，如果没有建立一个用来输出的新窗口，那么当前文档被清除。

HTML 忽略换行符，除非它在<*PRE*>标记中使用。

相关条目

document 的方法。

参见 *close*，*clear*，*open* 和 *write* 方法。

第二部分　JavaScript 语句与运算符

JavaScript 语句

在 JavaScript 中，用来控制程序流程的语句，与 Java 语言和 C 语言的语句相似。如果必要的话，一条语句可以跨数行，或者多条语句被放在同一行中。

有两个要点应当记住。第一，象函数定义这样的语句块，必须用大括号括起来。这就是 JavaScript 如何来描述代码块的。第二，分号必须放在所有语句之间。如果没有分号，那么脚本（script）动作将是无法预料的。

因为 JavaScript 的格式编排不够严格，所以你必须提供行间断和缩进格式，以确保今后代码是可读的和易于理解的。

break

结束当前的 for 或者 while 循环，并将控制权转给循环后的第一条语句。

用法

下面的例子是将一个表格上的元素进行累加，这里假设所有元素都是数字值。如果遇到“0”值，则停止累加操作。

```
function checkValues (form) {
    var total
    for (I=0; I<=form.elements.length; I++) {
```

```
            if (element [I] .value = " 0") {
                break; }
            else {                    total+=I
            document. write("The running total is " + total);
}
        }
        return total
    }
```

comment

脚本程序作者所写的注释部分，它们将被解释程序忽略掉。单行注释时，注释部分前使用“//”符号，多行注释时，开始部分用“/*”符号，结尾用“*/”符号。

用法

```
/* These comments could start here
and
end
down here. */
...statements...
// This comment is limited to this line only.
```

continue

在 while 循环中把控制交给其条件；在 for 循环中把控制交给 update 表达式。continue 与 break 最重要的区别是，它不结束循环。

用法

下面的例子是将一个表格上的元素进行累加，这里假设所有元素都是数字值。如果遇到小于 0 的值，则不包含在运行总数中。

```
function checkValues (form) {
    var total
    for (I=0; I<=form.elements.length; I++) {
        if (element [I] .value < 0) {
            continue; }
        else {
            total +=I;
            document.write ("The running total is "+to-
tal); }
    }
    return total
}
```

for

创建一个循环，它带有包含在圆括号中，并由分号隔开的三个任意表达式，其后是循环时要执行的一组语句。

```
for (initialExpression; condition; updateExpression) {
    ...statements...
}
```

initial 表达式被用来初始化计数变量，它可以是一个用 var 说明的新变量。

condition 表达式在整个循环的每一遍当中都要求值。如果条件为 true，则循环语句继续执行。如果条件被省略，那么它缺省为 true，且循环持续执行，直到出错或者遇到 break。

update 表达式用来增加计数变量的值。它也是可选的，并且能够在循环语句内由程序修改。

用法

for 创建一个循环，该循环持续执行，直到出错或是执行 break 语句。increment 这个变量在循环中每次加 2。

```
for (var increment=0; ; increment+=2) {
    ...statements...
}
```

下面的例子是一个没有更新计数器值的循环。如果在语句的执行过程中，计数器从来都没有被修改过，那么计数器的值始终为 10。

```
for (var increment=10; increment<101;) {
    ...statements...
}
```

for...in

对一个对象的全部属性都重复一个变量。对每一个属性，for...in 都执行循环内的语句段：

```
for (objectVariable) {
    ...statements...
}
```

用法

for...in 是一个对于调试非常有用的函数，由于它能在一个循环中，显示一个对象的全部属性。

```
function objectDisplay (obj) {
    var displayLine;
    for (var prop in obj) {
        displayLine = obj.name + "." + prop + " = "
+ obj [prop];
        document.write (displayLine + "<BR>")
    }
    document.write ("End of object " + obj.name)
}
```

function

定义一个带有名字和参数的 JavaScript 函数。为了返回一个值，函数必须包含有一个 return 语句。一个函数的定义，不能嵌套定义在另一个函数中。

function name ([*parameter* [..., *parameter*]]) {

...*statements*...

}

if...else

一个条件语句，如果条件为 true，则执行第一段语句，如果条件为 false，则执行 else 后面的语句。

if...else 语句能够任意层地被嵌套。如果条件语句后面是单条语句，那么是不需要大括号的。

if (*condition*) {

...*statements*...

} [*else* {

...*statements*...

}]

用法

下面的例子是在时钟显示时，将分钟转换为一个两位的数字。

```
function makeMinutes () {

    var minString = "";
    var now = new Date ();
    var min = Date.getMinutes ();
    if (min < 10) {
        minString += ": 0" + min; }
    else {
        minString += ":" + min; }
```

```
    return minString
}
```

return

规定一个被函数返回的值。

return *expression*;

用法

下面的例子是将三个字符串合并在一起，并用逗号将它们分隔开。

```
function stringAssemble (string1, string2, string3)
{
    return string1+ ", "+string2+ ", "+string3
}
```

var

说明一个变量，初始化其值是可选的。一个变量的作用范围可以是当前函数，或者是当前文档（只有当变量说明在函数外时）。

var variableName [=value] [..., variableName [=value]]

用法

globalString 变量在当前文档中可以被用在任意函数或脚本。而 localString 只能被用在 bracket 函数中。

```
var globalString
function bracket () {
    var localString = " ["+ globalString + "]";
    document. write (localString);
}
```

while

当表达式为 true 时，重复一个循环。如果条件不再是 true，则执行 while 循环后面的第一条语句。

```
while (condition) {
    ...statements...
}
```

用法

下面的例子是从一个字符串中查找一个特殊字符，当查找到或已经到字符串的末尾时停止查找。

```
var found = false
n = 0
while (n <= searchString.length || ! found) {
    if (searchString.charAt [n] == "?")
        found = true
    else
        n++;
}
```

with

为一段语句建立一个缺省的对象。任何无对象的属性引用，都假定使用该缺省的对象。

```
with (object) {
    ...statements...
}
```

用法

当对一组计算应用 Math 对象时，with 语句是非常有用的。例如：

```
with (Math) {
    var Value1 = cos (angle);
    var Value2 = sin (angle);
}
```

替代：

```
var Value1 = Math.cos (angle);
var Value2 = Math.sin (angle);
```

运算符

JavaScript 中有两种类型的运算符，一类是给一个变量赋值，另一类是。未经赋值，而产生一个值。

表达式

x = 1+1

有两部分。首先，表达式右边进行计算，结果为 2，然后，结果被赋值给左边的变量 x。另一方面

1+1

计算得 2，表达式即计算完毕。但是由于没有赋值运算符，没有做赋值操作。当表达式的剩余部分，诸如方法或函数调用完成时，这个值即作废了。

JavaScript 中，包含一个 null 值，用于那些还没有赋值的变量。在一个等式中，任何试图在一个等式中使用一个 null 变量，都会导致出错，除非用一个赋值语句初始化该变量，例如 timerID = null。

特殊运算符

```
.       //调用
[ ]     //下标
( )     //成员
```

句点用来将对象和它们的属性和方法分隔开来；

中括号用来表示数组下标，例如，form 和 elements；

圆括号有两个用途，第一，它们包括函数和方法中的参数和变量；第二，它们给出复杂等式的运算次序。虽然以下的两个等式看上去相同，但结果并不一样：

```
I = 5 * 5 + 10 // 结果 : 35
```

```
I=5 * (5+10) //结果：75
```

用法

当处理一个等式时，JavaScript 先从圆括号里的运算符开始计算。一直计算，直到所有的运算完毕。在第一个例子中，乘号的优先级比加号更高，所以 JavaScript 将 5 乘 5 然后再加 10。而第二个例子中，圆括号强行计算 5 加 10，然后将结果乘以 5。

一元运算符

```
++      //加一
--      //减一
!       //求反
-       //一元求反
```

双加和双减用来将单个变元增一和减一，也可以用来作为前缀或后缀。

用法

当一个双运算符位于一个变量的前面（前缀），则这一运算操作在赋值之前完成。当双运算符位于变量之后（后缀），则先将值赋给左边的变量，然后完成这个运算。

```
J = 1
I = J++ // I = 1, J = 2
I = ++J // I = 3, J = 3
```

求反运算符用来将布尔值取反，而变量本身未变。

```
testResult = true
document.write (testResult) // "true"
document.write (! testResult) // "false"
```

一元求反改变变量的符号，就象该数乘以－1 得到的结果一样。

二元运算符

```
+     //加法
-     //减法
*     //乘法
/     //除法
%     //取模
```

二元运算符需要两个操作数。加、减、乘、除是标准形式的二元运算符。

用法

模数运算是一种特殊的除运算，只返回除以后的余数。例如

```
I2 = 8 % 2     //返回 0
I3 = 8 % 3     //返回 2
```

按位运算符

```
~       //按位取反
<<      //左移
>>      //右移
>>>     //右移，以 0 填充
&       //与
^       //异或
|       //或
```

按位运算符对变量进行其最底层：即位（0 和 1）层操作。移位运算符将其左边的操作数转化成一个 32 位整数，它被移动的位数在操作符右边给定。按位逻辑运算符在比较两个值之前先将它们转化成 32 位整数。

用法

按位取反和常规取反运算类似，只是在位一级上操作。所有

为 1 的位都变为 0，而所有为 0 的位变为 1。

左移，即将所有的位向左移动等式右边规定的位数次，其后用 0 填满。而 0 填充右移操作朝反方向同样执行。标准的右移，在移位时传播最左位。

```
function bitShift ( ) {
        I = −1
        for ( increment = 0 ; increment<9; increment++) {document. write (I)
            I = I << 1
        }
}
```

按位逻辑运算符工作方式有细微差别。当与另一个值相比较时，它是按照表 4-7，4-8 和 4-9，基于位与位之间的比较返回值。

表 4-7　布尔运算——与（&）

位 1	位 2	结果
1	1	1
0	1	0
1	0	0
0	0	0

表 4-8　布尔运算——异或（^）

位 1	位 2	结果
1	1	0
0	1	1
1	0	1
0	0	0

表 4-9　布尔运算——或（/）

位 1	位 2	结果
1	1	1
0	1	1
1	0	1
0	0	0

例如，13 用 4 位二进制数表示为 1101，分别与 15（二进制 1111）和 0（二进制 0000）执行二元运算的结果，将返回以下值：

```
bin13 = 13;        //1101
bin15 = 15;        //1111
bin0 = 0;          //0000
document.writeln (bin13 & bin15); //结果为 13 (1101)
document.writeln (bin13 & bin0 ); //结果为 0 (0000)
document.writeln (bin13 ( bin15); //结果为 2 (0010)
document.writeln (bin13 ( bin 0); //结果为 13 (1101)
document.writeln (bin13 | bin15); //结果为 15 (1111)
document.writeln (bin13 | bin0); //结果为 13 (1101)
```

关系/相等性

```
<     //小于
>     //大于
<=    //小于等于
>=    //大于等于
= =   //等于
! =   //不等于
?:    //条件
```

把变量或文字与关系/相等性运算符一起使用时，将返回一个布尔值。

用法

在上面列出的运算符中，要特别注意等与不等两个运算符。

JavaScript 为了区别赋值和比较操作，需要用双等号表示相等。对于不等号，通常习惯是用相对的箭头（< >）表示。但在 JavaScript 中，是在等号前加上求反符实现。

条件运算符是特殊类型的比较运算，只与赋值运算符一起使用。它象 if—then 语句一样执行赋值操作。

```
underAge = ( age >=21 ) ? "no " : "yes"
```

如果括号中的表达式计算结果为真，则将第一个值赋给变量；若结果为假，则赋冒号后的值。

逻辑运算

```
&&      //与
|       //或
```

逻辑运算符用来比较两个布尔值(通常可以是其他的关系/相等性表达式)。

用法

逻辑运算符以按位“and”和“or”的相同方式执行操作，不同的只是它在变量一级执行。

在下面的例子中，如果变量 age 大于等于 21，并且变量 hasID 为布尔 true，则以下的语句块将执行。

```
if ( (age>=21) && (hasID ) ) {
    ...statements... }
```

在完成表达式的右半边之前，逻辑比较能否立即得出结果，取决于已经完成的比较。

```
false && anyExpression      // Shorts to false
true||anyExpression         // Shorts to true
```

赋值

```
=          //赋值
+=         //加并赋值、字符串并置
-=         //减并赋值
*=         //乘并赋值
/=         //除并赋值
%=         //取模并赋值
<<=        //按位左移并赋值
```

```
>>=      //按位右移并赋值
>>>=     //按位右移，以 0 填充，并赋值
&=       //按位与，并赋值
^=       //按位异或，并赋值
|=       //按位或，并赋值
```

当与一个其他的二元运算符一起使用时，赋值运算符提供了一种更新变元的方便简写形式。

用法

例如，下面两个语句完成同样的任务（将变量 shipping 加 5）：

```
shipping = shipping + 5;
shipping += 5;
```

+= 运算符也可用来连接字符串：

```
sentence = "";
subject = "The dog";
predicate = "walked home.";
sentence += subject;
sentence += predicate;
document.write (sentence); //结果为" The dog walked home."
```

运算符优先级

优先级指的是复合运算进行计算时的先后次序。同一级别的运算符具有相同的优先级。对于所有的二元运算都是从左到右进行计算，从表顶端的运算符开始，计算依次往下执行。

调用，成员	.	[ ]	( )		
求反/增减	++	--	!	~	-
乘/除	*	/	%		
加/减	+	-			
按位移	<<	>>	>>>		

关系　　< 　　> 　　<= 　　>=

相等性　　== 　　!=

按位与　　&

按位异或　　^

按位或　　|

逻辑与　　&&

逻辑或　　||

条件　　?:

赋值　　= 　　+= 　　-= 　　*= 　　/= 　　%= 　　<<= 　　>>= 　　>>>= 　　&= 　　^= 　　|=

逗号　　,

第三部分　参考表

ISO　拉丁字符集

当使用诸如 escape 和 unscape 这类方法时，返回值与 ISO 拉丁字符集有关联。在表 QR-10，你能看到集中的值和相应的字符的一览表：

表　QR-10　ISO　拉丁字符

十进制数	字符	实体参考
0	NUL	
1	SOH	
2	STX	
3	ETX	
4	EOT	
5	ENQ	
6	ACK	
7	BEL	
8	BS	
9	HT	
10	NL	
11	VT	
12	NP	
13	CR	
14	SO	
15	SI	
16	DLE	
17	DC1	
18	DC2	
19	DC3	

表 QR-10 （续）

十进制数	字符	实体参考
20	DC4	
21	NAK	
22	SYN	
23	ETB	
24	CAN	
25	EM	
26	SUB	
27	ESC	
28	FS	
29	GS	
30	RS	
31	US	
32	SP	
33	!	
34	"	&quot;
35	#	
36	$	
37	%	
38	&	&amp
39	'	
40	(	
41	)	
42	*	
43	+	
44	,	
45	-	
46	.	
47	/	
48	0	
49	1	
50	2	
51	3	
52	4	
53	5	
54	6	
55	7	

表 QR-10 （续）

十进制数	字符	实体参考
56	8	
57	9	
58	:	
59	;	
60	<	<
61	=	
62	>	>
63	?	
64	@	
65	A	
66	B	
67	C	
68	D	
69	E	
70	F	
71	G	
72	H	
73	I	
74	J	
75	K	
76	L	
77	M	
78	N	
79	O	
80	P	
81	Q	
82	R	
83	S	
84	T	
85	U	
86	V	
87	W	
88	X	
89	Y	
90	Z	
91	[	

表 QR-10 （续）

十进制数	字符	实体参考
92	\	
93	]	
94	^	
95	_	
96	`	
97	a	
98	b	
99	c	
100	d	
101	e	
102	f	
103	g	
104	h	
105	i	
106	j	
107	k	
108	l	
109	m	
110	n	
111	o	
112	p	
113	q	
114	r	
115	s	
116	t	
117	u	
118	v	
119	w	
120	x	
121	y	
122	z	
123	{	
124	|	
125	}	
126	~	
127	DEL	

表 QR-10 （续）

十进制数	字符	实体参考
128—159	--	
160		
161	¡	
162	¢	
163		
164		
165	¥	
166	|	
167	§	
168	¨	
169	©	
170	a	
171	«	
172		
173	—	
174	®	
175	¯	
176	°	
177	±	
178	2	
179	3	
180	´	
181	μ	
182		
183	·	
184	¸	
185	1	
186	º	
187	»	
188	--	
189	—	
190	—	
191	¿	
192	A	À
193	A	Á
194	Â	Â

表 QR-10 （续）

十进制数	字符	实体参考
231		ç
232		è
233		é
234		ê
235		ë
236	ì	ì
237	í	í
238		î
239		ï
240		ð
241	ñ	ñ
242	ò	ò
243	ó	ó
244	ô	ô
245	õ	õ
246	ö	ö
247	÷	
248	ø	ø
249	ù	ù
250	ú	ú
251	û	û
252	ü	ü
253	y	ý
254	—	þ
255	ÿ	ÿ

颜色值

颜色可以在不同的属性中以两种方法来引用。第一种是利用字符串文字，即颜色名称。第二种是利用 RGB 十六进制三色分量，它通过组合三种颜色值形成。例如，兰色的一种特定的暗色调用字符文字 aliceblue 或三色分量 FOF8FF 表示。浏览器上显示的实际颜色取决于客户机所支持的颜色范围—— 16 色的 VGA 监视器远不及 16M 色 SVGA 那样丰富多采。

颜色字符串	红	绿	蓝
aliceblue	F0	F8	FF
antiquewhite	FA	EB D7	
aqua	00	FF	FF
aquamarine	7F	FF	D4
azure	F0	FF	FF
beige	F5	F5	DC
bisque	FF	E4	C4
black	00	00	00
blanchedalmond	FF	EB	CD
blue	00	00	FF
blueviolet	8A	2B	E2
brown	A5	2A	2A
burlywood	DE	B8	87
cadetblue	5F	9E	A0
chartreuse	7F	FF	00
chocolate	D2	69	1E
coral	FF	7F	50
cornflowerblue	64	95	ED
cornsilk	FF	F8	DC
crimson	DC	14	3C
cyan	00	FF	FF
darkblue	00	00	8B
darkcyan	00	8B	8B
darkgoldenrod	B8	86	0B
darkgray	A9	A9	A9
darkgreen	00	64	00
darkkhaki	BD	B7	6B
darkmagenta	8B	00	8B
darkolivegreen	55	6B	2F
darkorange	FF	8C	00
darkorchid	99	32	CC
darkred	8B	00	00
darksalmon	E9	96	7A

颜色字符串	红	绿	蓝
darkseagreen	8F	BC	8F
darkslategray	2F	4F	4F
darkturquoise	00	CE	D1
darkviolet	94	00	D3
deeppink	FF	14	93
deepskyblue	00	BF	FF
dimgray	69	69	69
dodgerblue	1E	90	FF
firebrick	B2	22	22
floralwhite	FF	FA	F0
forestgreen	22	8B	22
fuchsia	FF	00	FF
gainsboro	DC	DC	DC
ghostwhite	F8	F8	FF
gold	FF	D7	00
goldenrod	DA	A5	20
gray	80	80	80
green	00	80	00
greenyellow	AD	FF	2F
honeydew	F0	FF	F0
hotpink	FF	69	B4
indianred	CD	5C	5C
indigo	4B	00	82
ivory	FF	FF	F0
khaki	F0	E6	8C
lavender	E6	E6	FA
lavenderblush	FF	F0	F5
lawngreen	7C	FC	00
lemonchiffon	FF	FA	CD
lightblue	AD	D8	E6
lightcoral	F0	80	80
lightcyan	E0	FF	FF
lightgoldenrodyellow	FA	FA	D2

颜色字符串	红	绿	蓝
lightgreen	90	EE	90
lightgrey	D3	D3	D3
lightpink	FF	B6	C1
lightsalmon	FF	A0	7A
lightseagreen	20	B2	AA
lightskyblue	87	CE	FA
lightslategray	77	88	99
lightsteelblue	B0	C4	DE
lightyellow	FF	FF	E0
lime	00	FF	00
limegreen	32	CD	32
linen	FA	F0	E6
magenta	FF	00	FF
maroon	80	00	00
mediumaquamarine	66	CD	AA
mediumblue	00	00	CD
mediumorchid	BA	55	D3
mediumpurple	93	70	DB
mediumseagreen	3C	B3	71
mediumslateblue	7B	68	EE
mediumspringgreen	00	FA	9A
mediumturquoise	48	D1	CC
mediumvioletred	C7	15	85
midnightblue	19	19	70
mintcream	F5	FF	FA
mintyrose	FF	E4	E1
moccasin	FF	E4	B5
navajowhite	FF	DE	AD
navy	00	00	80
oldlace	FD	F5	E6
olive	80	80	00
olivedrab	6B	8E	23
orange	FF A5	00	

颜色字符串	红	绿	蓝
orangered	FF 45	00	
orchid	DA	70	D6
palegoldenrod	EE	E8	AA
palegreen	98	FB	98
paleturquoise	AF	EE	EE
palevioletred	DB	70	93
pa			
payawhip	FF	EF	D5
peachpuff	FF	DA	B9
peru	CD	85	3F
pink	FF	C0	CB
plum	DD	A0	DD
powderblue	B0	E0	E6
purple	80	00	80
red	FF	00	00
rosybrown	BC	8F	8F
royalblue	41	69	E1
saddlebrown	8B	45	13
salmon	FA	80	72
sandybrown	F4	A4	60
seagreen	2E	8B	57
seashell	FF	F5	EE
sienna	A0	52	2D
silver	C0	C0	C0
skyblue	87	CE	EB
slateblue	6A	5A	CD
slategray	70	80	90
snow	FF	FA	FA
springgreen	00	FF	7F
steelblue	46	82	B4
tan	D2	B4	8C
teal	00	80	80
thistle	D8	BF	D8

颜色字符串	红	绿	蓝
tomato	FF	63	47
turquoise	40	E0	D0
violet	EE	82	EE
wheat	F5	DE	B3
white	FF	FF	FF
whitesmoke	F5	F5	F5
yellow	FF	FF	00
yellowgreen	9A	CD	32

保留字

以下的保留字不能在JavaScript编码中被用作用户对象或变量。并非所有保留字都在当前已被JavaScript使用，它们是为以后保留的。

abstract	float	public
boolean	for	return
break	function	short
byte	goto	static
case	if	super
catch	implements	switch
char	import	synchronized
class	in	this
const	instanceof	throw
continue	int	throws
default	interface	transient
do	long	true
double	native	try
else	new	var
extends	null	void
false	package	while
final	private	with
finally	protected	

附录 A　任务参考

JavaScript 提供了若干工具来执行 HTML 文档中的各种各样的任务，而无需与服务器交互。

新浏览器

JavaScript 有一种强有力的功能，对实现演示和漫游十分有用，即它能生成具有可控功能级的新式样的客户浏览器。

创建一个新浏览器的基本命令是：

windowVar = window.open ("*URL*", "*windowName*" [,"*windowFeatures*"])

打开一个带有 hotlink－only 导航的普通窗口：

// Note: Setting one feature automatically sets all non－mentioned features to false.

Window.open ("*URL*", "*windowName*", "toolbar=no")

打开一个没有目录和菜单条的窗口：

Window.open ("URL", "windowName",

"toolbar=yes, location=yes, directories=no, status=yes,

menubar=no, scrollbars=yes, resizable=yes")

欲知更多的信息，参见：

window 对象

open 方法

建立一个定制导航的 Web 站点

JavaScript 的一个强大功能就是它能够控制浏览器的功能性（见"任务参考，新浏览器"）。这对于创建引导漫游或演示程序很有用处。为了建立一个站点，需要一个前门来引发应用程序的其

他部分。

```
<FORM>
<INPUT TYPE="button" NAME="tour" VALUE="
Start Tour"
onClick = " window.open ('tourframes.html', 'tourWin-
dow', 'toolbar=no')">
</FORM>
```

tourframes.html 文件创建了包含一个启动页面和导航条的框架。

```
<FRAMESET COLS="%10,%90">
    <FRAME SRC="navbar.html">
     <FRAME SRC =" toursrart.html" NAME =" con-
tentWin">
</FRAMESET>
```

navBar 文件是若干按钮的一个简单集合，这些按钮带有定制 onClick 事件处理器用以指挥浏览器。

```
<FORM NAME = "navBar">
<INPUT TYPE="button" NAME="back" VALUE="
<-- Back"
onClick="contentWin.document.history.back ()">
<INPUT TYPE ="button"NAME="forw" VALUE="
Forward -->"
onClick="contentWin.document.history.forward ()">
<INPUT TYPE="button" NAME="home" VALUE="
Home"onClick="contentWin.document.history.go (0)">
<INPUT TYPE ="button" NAME ="quit" VALUE ="
Quit the Tour" onClick="parent.close ()">
</FORM>
```

为了避免不成熟的退出，退出按钮也可以在关闭窗口之前调用一个函数，来确认用户的选择。欲知更多的信息，参见：

button，document，form，history 和 window 对象

back，close，forward 和 go 方法

top 和 parent 属性

onClick 事件处理器

自复位状态信息

window.status 属性有一个常常另人烦恼的副作用——它的持续性。一旦设置，它就不会改变，除非遇到了另一个 window.status 赋值。

你可以用 setTimeout 方法来克服这个性质。

```
timeDelay = 1500 //1.5 seconds
function eraseStatus () {
    window.status = "" //This can also be set to a 'default' message.
};
function setStatus (statusText) {
    window.status = statusText;
    setTimeout ("eraseStatus ()", timeDelay)
}
```

使用这两个函数是一件简单的事，只需把 setStatus 函数包含在 onMouseOver 事件处理器中。

```
<A HREF=URL onMouseOver="setStatus('Your messate here.'); return true">linkText</A>
```

欲知更多的信息，参见：

window 对象

status 属性

setTimeout 方法

onMouseOver 事件处理器

特定平台的换行字符

使用哪种形式的换行字符取决于客户所使用的平台。除了其

他平台都需要的/n 之外，Windows 平台还需要一个/r。由于不可能控制访问你的页面的是哪种平台，一个简单的函数可以确保使用适当格式的换行字符。

```
function brk () {
    if (navigator.appVersion.lastIndexOf('Win') ! = -1)
        return "\r\n"
    else
        return "\n"
}
```

欲知更多的信息，参见：

navigator 对象

appVersion 属性

lastIndexOf 方法

确认表格信息

通过 CGI 脚本确认信息是一个耗时的过程。这不仅要增加客户机与服务器间的通信，而且开发实际的 CGI 脚本还需要时间和专门的知识。

用 JavaScript 将表格确认直接包含在 HTML 页面上，会加快速度，并将处理过程局限在终端用户一边。这使得终端用户很难向服务器发送不兼容的数据，这些数据可能对服务器造成破坏。

有几种方法可用来进行表格确认，但一个基本的原则是把一个 JavaScript 函数加到一个真提交按钮上。提交按钮的 HTML 定义看上去象下面这样：

```
<INPUT TYPE = "BUTTON" NAME = "SUBMIT"
VALUE="SUBMIT" onClick="checkInformation (this.form)"
>
```

checkInformation 将验证表格信息是否满足 CGI 脚本的期望。如果不满足，该函数应至少不作提交而返回文档，或将焦点返回到出错的项。如果所有的信息都通过了检验，该函数也可以

提交这个表格。

```
function checkInformation (form) {
    ... validation statements ... ;
    if (validationPassed) {
        form.submit (); }
    return;
}
```

欲知更多的信息，参见：

form 和 button 对象

focus 方法

onClick 事件处理器

创建数组

尽管 JavaScript 在表示它的若干对象（表格、元素等）时使用了数组，但它并没有提供一种直接的方法来创建用户定义的数组，而数组是进行数据处理的一种主要数据类型。下面的函数通过初始化各个元素创建一个新数组。这个函数对小的数组是很有用的，但对于大数组就显得笨拙了。

```
function arrayCreator () {
    this.length = initArray.arguments.length; //counts the number of arguments included when the function is called
    for (var I=0; I<this.length; I++) {
        this [I+1] = userArray.arguments [I] //load the new values into the array
    }
}
```

为了初始化一个新数组，要按下面的语法使用上述函数：

var *arrayName* = new userArray (argument1 [, argument2] [, argument3] [etc.])

产生一个随机数（在非 UNIX 平台上）

目前，random 方法只能在 Netscape 的 UNIX 版本上运行。还有不用这个内部方法的其他方法来产生伪随机数。这称之为计算随机数，如果在短时间内反复地访问它，就会表现出它的偏差和真实非随机的性质。

为了保证一个脚本在不同平台上的兼容性，任何依赖于随机数的脚本都不应使用 random 方法，取而代之的应该依靠象下面这样的用户自定义函数。

```
function UnixMachine () {
    if (navigator.appVersion.lastindexOf ('Unix') != -1)
        return true
    else
        return false
}
function randomNumber () {
    if (UnixMachine ()) {
        num = Math.random () }
    else {
        num = Math.abs. (Math.sin (Date.getTime ()));
    }
    return num;
}
```

这个函数产生一个 0 到 1 之间的数，对于每几秒钟需要一个随机数的应用程序来说，它工作得很好。如果需要随机数的频率更高，你就要每隔两个时间段在等式中加入更多的变化，例如不同的计算（cos，tan，log），或采用诸如此类的办法。

欲知更多的信息，参见：

Date 和 Math 对象

random 方法

function 和 return 语句

附录 B　Internet 资源

由于 JavaScript 的非常具体的平台基础（目前它只能被 Netscape Navigator 所识别），直接介绍它的“官方的”在线资源为数极少。然而，“非官方的”资源（由试验者提供，他们愿意和别人一起分享自己在这项新技术中的发现）却在迅速增长。

全球信息网（The World Wide Web）

由于 JavaScript 是针对 Web 的，因此，在 Web 上寻找有关其使用信息的最佳资源是唯一合适的。正如大多数其他基于 Internet 的资源一样，大部分 JavaScript 站点主要面向 Java，而 JavaScript 作为其中一部分包含在里面。

下面的清单决算不上全面。为了跟上 Web 上新提供的信息，最好的办法就是利用许多站点都有的其他热链接页面。

Navigator 是第一个支持 JavaScript 的浏览器，应将 Netscape 的主站点作为一个定期查看的好地方，特别是对于了解 JavaScript 语言规范的更新/新增内容，更是如此。

Netscape 也有自己的“开发伙伴计划”，向用户提供：进一步的技术和编程支持，关于即将推出的新产品、扩展、plug—in 的信息，以及对新的浏览器预 β 版、服务器和 plug—in 的访问。

也一定要查看位于 http：//www.hidaho.com/colorcenter/的 Netscape 彩色中心。对于将颜色值和它们在屏幕上的表现作比较，这是一个方便的地方。

Voodoo JavaScript Tutorial（http：//rummelplatz.uni—mannheim.de/~skoch/js/script.htm）

Voodoo JavaScript Tutorial 是由若干易于理解的章节组成的一个连续的教程，它覆盖了 JavaScript 的基本内容。其中包括建在

页面上的例子，以及描述文字和代码示例。这是一个入门的好地方。

The Unofficial JavaScript Resource Center

(http：//www. intercom. net/user/mecha/java/index. html)

The Unofficial JavaScript Resource Center 是一个制作精良的专用于 JavaScript 的新站点。当初它还很有限，但它很有发展前途，会不断地给一大批用户带来更多的示例和技术。其想法是提供一些例子和代码片段，供人们拷贝和浏览。随着内容的扩充，该站点的组织安排会使其成为一个很有用的资源。

Danny Goodman' s JavaScript Pages

(http：//www/dannyg. com：80/javascript)

Danny Goodman' s JavaScript Pages 集合了若干例子，覆盖了较多的 JavaScript 高级概念，其中包括 cookies。Danny Goodman 是 Web 上关于 JavaScript 的一个事实上的专家，它提供了一些可用于学习和改写其他应用程序的好的范例。

JavaScript Index (http：//www. c2. org/~ andreww/javascript/)

JavaScript Index 是 JavaScript 实现与试验的一个完整的提纲，其中包括一个不断增长的个人主页清单，在这些主页上可以看到各种各样的 JavaScript 技巧。在这个站点上，有一部分是 JavaScript 库，它集合了来自 Web 圈子的源代码，目前还小，但正在扩大。

Gamelan (http：//www. gamelan. com/)

EarthWeb' s Gamelan，也称为“在线 Java 索引”，它广泛收集了指向其他站点、例子、工具、实用程序以及其他有趣东西的链接。尽管它主要是针对 Java 的，但 JavaScript 部分也具有相当规模。

Sun Microsystems (http：//java. sun. com/)

在 Sun Microsystems，Java 的发源地，Sun 主持着 Java 的主站点。此外，Sun 拥有一个 Java 用户团体（Sun 用户团体中的一

个分支组），并拥有若干邮件清单与布告清单，这些将使开发者能够及时了解最新的事件。

JavaWorld（http：//www. javaworld. com）

为了支持将 Java 的开发集成到 Latte 中去的努力，关于 Java 开发的 Borland 主站点承诺不断地向 Java 开发者通报最新情况。

Symantec（http：//cafe. symantec. com/）

当 Symantec 终于为 Windows 和 Macintosh 提供了一个用于生成 Java 小应用程序的开发平台时，它处于领先地位。随着加在其流行的 C++软件包上的 Java 开发附加件的首次公开（免费、面向 Windows NT/95），Symantec 提供了第一个用于小应用程序生成的基于图形用户界面（GUI）的开发环境。

Dimension X（http：//www. dnx. com/）

Dimension X 是 Liquid Reality 的主页。Liquid Reality 是一个 Java 小应用程序的开发平台。该平台包括具有三维建模能力的软件包，并带有一个 Java 小应用程序编码器。

The Java Developer（http：//www. digtalfocus. com/faq/）

由 Digital Focus 发起，The Java Developer 充当了 The Java Developer FAQ 的主站点。当你浏览该站点时，查找和询问提交按钮的显示是一个比较有趣的框架实现。

Usenet

现在已经出现一些 Usenet 新闻组，它们为那些寻找 Java、JavaScript 和一般 Web 编程指导的开发者们提供了渠道。这些新闻组有如下。

comp. lang. javascript

comp. lang. javascript 是专门致力于 JavaScript 开发的。每天都有大量消息在这一论坛上登出，其中有很多是来自 JavaScript 方面的公认的权威。它包含了大量的信息，如果你有时间研读所有这些消息的话。

netscape. navigator（news：//secnews. netscape. com）

保持一条由 Netscape 的 JavaScript 开发者监管的直通线是

有益的，而 netscape. navigator 就是与之最接近的事物。在这个新闻组里，JavaScript 方面的话题的确是少数，但如果你去看看，这儿确实是有的。

请注意这个与众不同的 news 服务器。它的名字暗示它是安全的，但它似乎是很容易浏览和公告的。

comp. lang. java

comp. lang. java 是一个新闻组，comp. lang. javascript 就是从它派生出来的。它专门处理 Sun 的 Java 语言，但偶尔也有关于 JavaScript 的信息。

comp. infosystems. www. authoring. *

comp. infosystems. www 曾经是有关 Web 讨论的传统的新闻组集合地。随着 Web 的扩大，这些新闻组也在不断扩大，它们覆盖了方方面面，从浏览器到新开通 Web 站点的发布。

尽管在 comp. infosystems 系统中没有专门的 JavaScript 组，但是却有如下若干组讨论 Web 创作的各个方面：

* comp. infosystems. www. authoring. cgi
* comp. infosystems. www. authoring. html
* comp. infosystems. www. authoring. images
* comp. infosystems. www. authoring. misc

通信名单

对于那些乐于感受收件箱爆满刺激的人们，有专门用于 JavaScript 的通信名单。这些名单能提供与 Usenet 新闻组提供类似的信息。

然而要当心，通信名单很象一条电话合用线，它可能变得很唠叨（你必须在收件箱的大量邮件当中费力地找出能为你所用的东西）。

如果你打算大量使用通信名单，你也许要去了解一个 e—mail 程序，它能够把相同主题的消息连接在一起，以使使大量信息变得有条理。

关于通信名单还要说一句。尽管你将你的问题和评论发往名

单上的地址（为了再向名单上的其他读者广播），但是，向名单订阅和不从名单订阅是通过一个单独的 e－mail 地址完成的，特别是名单服务器（listserver）的地址。

下面各通信名单，既提到了名单地址，也提到了名单服务器地址，而向名单地址发订阅请求（从而名单上的每个人都知道你不清楚自己正在做什么）是一种打上“newbie“标记的有保证的方法。

如果你希望得到更多的关于如何与名单服务器通信的信息（或关于一个特定服务器的其他可能名单的信息），你可以向名单服务器地址发一条消息，在消息体中带上 help。

javascript@obscure.org

由 Obscure Organization（http：//www.obscure.org/）和 TeleGlogal Media，Inc.（http：//www.tgm.com/）发起，其 JavaScript Index 是唯一专门针对 JavaScript 的通信名单。

讨论的内容越来越广泛，从入门级的问题，到很多关于如何最佳处理动画、帧频调节、重装等专题的讨论。

若想订阅，就向 majordomo@obscure.org 发送一条在其消息体中带有 subscribe javascript 的消息。或者，将你的浏览器指向 http：//www.obscure.org/javascript，以获得进一步的信息。

java@borland.com

一个与 Borland JavaWorld 站点活动相伴随的、定期发布的业务通讯，它能让你了解到 Borland 为把 Java 技术集成到其开发工具中所作的工作。若想订阅，就向 listserv@borland.com 发送一条在其消息体中带有 subscribe java {your first name} {your last name} 的消息。

java-announce@java.sun.com

Sun Microsystems，Java 的发源地，有它自己的通信名单集。java-announce 名单主要用于与 Java 有关的新工具的发布。若想订阅，就向 majordomo@java.sun.com 发送一条在其消息体中带有 subscribe java-announce 的消息。

术语对照

符号

" "	（引号），用于定义字符串
( )	（圆括号），括上 for 语句
*	（星号），用在方法中
/*	用在注释中
//	（双斜杠），用在注释中
;	（分号）
	用于 for 语句
	用在语句中
\	（反斜杠），用在字符串中
{ }	（花括号），括上语句

A

B

background color	背景颜色
backslashes (\) in strings	字符串中的反斜杠 (\)
bgColor property	bgColor 属性
big method	big 方法
binary operators	二元运算符
bitwise operators	按位运算符
blink method	blink 方法
blur method	blur 方法
bold method	bold 方法
boolean literals	布尔文字
boxes	框
checkboxes	检查框
checkbox object	checkbox 对象
checked property	checked 属性
defaultChecked	
onClick event handler	onClick 事件处理器
dialog boxes	对话框
confirm method	confirm 方法
prompt method	prompt 方法
break statement	break 语句
browsers	浏览器
appCodeName property	appCodeName 属性
appName property	appName 属性
appVersion property	appVersion 属性
creating	创建
navigator object	navigator 对象
new	新浏览器
status bars	状态条
top property	top 属性
button object	button 对象
onClick event handler	onClick 事件处理器

C

clear method — clear 方法
clearTimeout method — clearTimeout 方法
click method — click 方法
clocks, see Date object, time — 时钟，参见 Date 对象和 time
close method — close 方法
color — 颜色
    backgrounds — 背景
    color values — 颜色值
    fontcolor method — fontcolor 方法
    linkColor property — linkColor 属性
    links
    Netscape ColorCenter Web site — Netscape 彩色中心 Web 站点
    text

commands, see methods, objects, properties — 命令，参见方法，对象，属性
comments — 注释
    /* symbols — /* 符号
    // (double slashes) — 双斜杠
    <! --> tags — <! -->标记
    comment statement — 注释语句

《confirm method — confirm 方法
constants — 常数
continue statement — continue 语句
cookie property — cookie 属性
cos method — cos 方法
counters (unary operands) — 计数器（一元运算操作数）

D

Date object — Date 对象
    methods — 方法
        getDate

E

F

onSubmit event handler	onSubmit 事件处理器
properties	属性
action,	
button	
checkbox	
elements	
elements array	
encoding	
hidden	
length	
methods	
password	
radio	
select	
submit	
target	
text	
textarea	
forms, validating information	表格，确认信息
properties	属性
forms property	forms 属性
forward method	forward 方法
frame object	frame 对象
clearTimeout method	clearTimeout 方法
properties	属性
length	
name	
parent	
self	
window	

G

H

back

forward

go

host property	host 属性
hostname property	hostname 属性
href property	href 属性
HTML tags	HTML 标记
<! ——>	
<A>	
<A HREF>	
links array	links 数组
links property	links 属性
onMouseOver event handler	onMouseOver 事件处理器
<ANCHOR>	
<B>	
<BASEFONT>	
<BIG>	
<BLINK>	
<BODY>	
document object	document 对象
fgColor property	fgColor 属性
linkColor property	linkColor 属性
onLoad event handler	onLoad 事件处理器
onUnload event handler	onUnload 事件处理器
vlinkColor property	vlinkColor 属性
<FONT COLOR=>	
<FONT SIZE=>	
<FORM>	
action property	action 属性
button objects	button 对象

checkboxes object — checkboxes 对象
encoding property — encoding 属性
method property — method 属性
onSubmit event handler — onSubmit 事件处理器
radio object — radio 对象
reset object — reset 对象
submit object — submit 对象
window object — window 对象

<FRAMESET>

frame objects — frame 对象
frames property — frames 属性
onLoad event handler — onLoad 事件处理器
onUnload event handler — onUnload 事件处理器
parent property
parent 属性
window object — window 对象

<HREF>

<I>

<INPUT TYPE=>

blur method — blur 方法
button object — button 对象
checkbox object — checkbox 对象
hidden object — hidden 对象
onBlur event handler — onBlur 事件处理器
onChange event handler — onChange 事件处理器
onClick event handler — onClick 事件处理器
onFocus event handler — onFocus 事件处理器
onSelect event handler — onFocus 事件处理器

<OPTION>

defaultSelected property — defaultSelected 属性
options property — options 属性
text property — text 属性

<PRE> (writeln method)	<PRE> (writeln 方法)
<SCRIPT>	
<SELECT>	
options property	options 属性
selectedIndex property	selectedIndex 属性
<SMALL>	
<STRIKE>	
<SUB>	
<SUP>	
<TEXTAREA>	
<TITLE>	
<tt> (fixed method)	<tt>fixed 方法

I—J—K

if...else statement	if...else 语句
index property	index 属性
indexOf method	indexOf 方法
instances	实例
instance hierarchy	实例层次
integers	整数
internet resources	internet 资源
mailing lists	通信名单
UseNet newsgroups	UseNet 新闻组
World Wide Web sites	全球信息网站点
isNaN method	isNaN 方法
ISO Latin character set	ISO 拉丁字符集
italics method	italics 方法
Java	Java
applets	小应用程序
Internet resources	Internet 资源
mailing lists	通信名单

linkColor property	linkColor 属性
links	链接
anchor method	anchor 方法
anchors array	anchors 数组
anchors property	anchors 属性
color	颜色
alinkColor property	alinkColor 属性
linkColor property	linkColor 属性
vlinkColor	
onMouseOver event handler	onMouseOver 事件处理器
links array (object)	links 数组（对象）
links property	links 属性
literals	文字
boolean literals	布尔文字
floating—point numbers	浮点数
integers	整数
strings	字符串
LN10 property	LN10 属性
LN2 property	LN2 属性
location object	location 对象
properties	属性
hash	
host	
hostname	
href	
pathname	
port	
protocol	
search	
target	

M

round
sin
sqrt
tan
properties 属性
E
LN10
LN2
LOG10E
LOG2E
PI
SQRT1_2
SQRT2

max method max 方法
method property method 属性
methods 方法
* (asterisks) used with methods 与方法一起使用的星号
abs
acos
alert
anchor
asin
atan
back
big
blink
blur
bold
ceil
charAt
clear

clearTimeout
click
close
confirm
cos
escape
returned values (ISO Latin characters)　返回值 (ISO 拉丁字符)
eval
exp
fixed
floor
focus
fontcolor
fontsize
forward
getDate
getDay
getHours
getMinutes
getMonth
getSeconds
getTime
getTimezoneOffset
getYear
go
indexOf
isNaN
italics
lastIndexOf
link
log

toLowerCase
toUpperCase
unescape
returned values (ISO Latin characters)
返回值 (ISO 拉丁字符)
UTC
write
writeln

min method
min 方法
mouse actions
鼠标动作
click method
click 方法
onMouseOver event handler
onMouseOver 事件处理器

N .

name property
name 属性
navigator object
navigator 对象
properties
属性
appCodeName
appName
appVersion
userAgent

Netscape
Netscape
ColorCenter Web site
彩色中心 Web 站点
Navigator
导航器
status bar massages
状态条信息
status bars
状态条
UseNet newsgroup
UseNet 新闻组

newline characters
换行符

selectedIndex
text
value
select objects select 对象

P

parent property parent 属性
parentheses ( ) enclosing for ststement 圆括号 ( ) 括上 for 语句
parse method parse 方法
parseFloat method parseFloat 方法
parseInt method parseInt 方法
password object password 对象
methods 方法
blur
focus
select
properties 属性
defaultValue
name
value
pathname property pathname 属性
PI property PI 属性
port property port 属性
pow method pow 方法
precedence of operators 运算符优先级
prompt method prompt 方法
properties 属性
action
alinkColor

anchors
appCodeName
appName
appVersion
bgColor
checked
cookie
defaultSelected
defaultStatus
defaultValue
E
elements
encoding
fgColor
forms
frames
hash
host
hostname
href
index
lastModified
length
linkColor
links
LN10
LN2
location
LOG10E
LOG2E
method
name

options
parent
pathname
PI
port
protocol
referrer
search
selected
selectedIndex
self
SQRT1_2
SQRT2
status
target
text
title
top
userAgent
value
vlinkColor
window

protocol property　　protocol 属性

Q—R

quotation marks (" ") defining strings　　定义字符串的引号 (" ")

radio object　　radio 对象
- click method　　click 方法
- onClick event handler　　onClick 事件处理器
- properties　　属性

S

properties	属性
index	
name	
options	
selectedIndex	
selected property	selected 属性
selectedIndex property	selectedIndex 属性
self property	self 属性
semicolons (;)	分号
for ststements	用于 for 语句
in statements	用在语句中
setDate method	setDate 方法
setHours method	setHours 方法
setMinutes method	setMinutes 方法
setMonth method	setMonth 方法
setSeconds method	setSeconds 方法
setTime method	setTime 方法
setTimeout method	setTimeout 方法
self--resetting status messages	自复位状态信息
setYear method	setYear 方法
sin method	sin 方法
slashes	斜杠
//in comments	用于注释
\ in strings	用于字符串
small method	small 方法
special operators	特殊运算符
sqrt method	sqrt 方法
SQRT1_2 property	SQRT_2 属性
SQRT2 property	SQRT2 属性
statements	语句

T

text	文本
big method	big 方法
blink method	blink 方法
bold method	bold 方法
fgColor property	fgColor 属性
italics method	italics 方法
onBlur event handler	onBlur 事件处理器
onChange event handler	onChange 事件处理器
onFocus event handler	onFocus 事件处理器
onSelect event handler	onSelect 事件处理器
sup method (superscript text)	sup 方法 (上标文本)
toLowerCase method	toLowerCase 方法
toUpperCase method	toUpperCase 方法
text object	text 对象
event handlers	事件处理器
onBlur	
onChange	
onFocus	
onSelect	
methods	方法
blur	
focus	
select	
properties	属性
defaultValue	
name	
value	
text property	text 属性
textarea object	textarea 对象
event handlers	事件处理器

onBlur	
onChange	
onFocus	
onSelect	
methods	方法
blur	
focus	
select	
properties	属性
defaultValue	
name	
value	
time	时间
getTime method	getTime 方法
getTimezoneOffset method	getTimezoneOffset 方法
lastModified property	lastModified 属性
see also Date object	还参见 Date 对象
timeouts	超时
clearTimeout method	clearTimeout 方法
setTimeout method	setTimeout 方法
title property	title 属性
toGMTString method	toGMTString 方法
toLocaleString method	toLocaleString 方法
toLowerCase method	toLowerCase 方法
top property	top 属性
toUpperCase method	toUpperCase 方法
type casting variables	类型转换变量

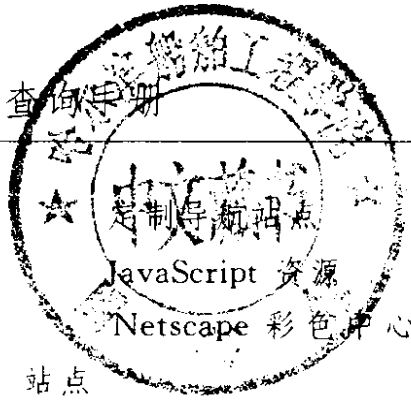
U—V

W—X—Y—Z

onLoad
onUnload
methods 方法
alert
clearTimeout
close
confirm
open
prompt
setTimeout
propeties 属性
decfaultStatus
document
frame
frames
length
location
name
parent
self
status
top
window

window property window 属性
windows
clear method 方法
frame object 对象
frames property 属性

with statement with 语句
World Wide Web 全球信息网



custom—navigation sites	定制导航站点
JavaScript resources	JavaScript 资源
Netscape ColorCenter Web site	Netscape 彩色中心 Web 站点

write method	write 方法
writeln method	writeln 方法

无忧书库书籍版权申明