Dynamic HTML: The Definitive Reference

A Comprehensive Resource for XHTML, CSS, DOM, and JavaScript

第3版
针对Ajax和Web 2.0的最新版

Dynamic HTML

权威指南

Danny Goodman 著
杨文俊 张苏 王卓 译

O'REILLY®

電子工業出版社
PUBLISHING HOUSE OF ELECTRONICS INDUSTRY
http://www.phei.com.cn

O'REILLY®

Dynamic HTML权威指南

本书涵盖了最新的Web规范和各种浏览器功能特性，如果您要使用HTML、XHTML、CSS、文档对象模型（DOM）和JavaScript进行开发，那么它正是您要寻找的一站式终极资源宝库。本书为富互联网应用程序的设计者提供了全面而翔实的参考。在本书的帮助下，您设计的应用程序能够在Internet Explorer 7、Firefox 2、Safari、Opera等现代浏览器中畅通无阻。

通过本书，您可以即时了解浏览器对各种基于标准的最新技术的支持情况，其中包括CSS Level 3、DOM Level 3、Web Forms 2.0、针对Ajax应用程序的XMLHttpRequest，以及JavaScript 1.7。

本书：

- 为HTML、XHTML、CSS、DOM，以及核心JavaScript的标签、属性、对象、方法、事件等提供即时参考。
- 可供您快速查阅某个功能特性或术语，以获知它是否适用于期望的浏览器和版本。
- 提供便捷的交叉索引，您只须查阅一个元素属性（或者对象属性、方法、事件类型）即可找到所有的相关条目，其中包括相关联的HTML标签、样式属性，以及文档对象模型方法、属性和事件等。

《Dynamic HTML: The Definitive Reference》可以加速您在Web页面中增添高端功能的进程。对所有致力于创建动态Web内容的开发者而言，这本畅销书绝对值得收藏。

Danny Goodman是一位富有经验的专业作者，同时也是一位为公司和顶尖Intranet开发团队服务的资深程序开发顾问。在实现跨浏览器自动感知及客户端脚本编程方面，他的专业知识颇受欢迎，这也使得他能够轻松地解决很多实际问题。到目前为止，他已撰写了40余本专业书籍，包括畅销的O'Reilly图书，如大家所熟知的《Dynamic HTML: The Definitive Reference》（第1版&第2版）、《JavaScript & DHTML Cookbook》等。

图书分类：Web编程/JavaScript

责任编辑：王继花

项目管理：梁晶

www.oreilly.com

Broadview®
WWW.BROADVIEW.COM.CN

www.phei.com.cn

定价：99.80元

Dynamic HTML权威指南（第3版）

Dynamic HTML：The Definitive Reference，3rd Edition

【美】 Danny Goodman 著

杨文俊 张苏 王卓 译

電子工業出版社

Publishing House of Electronics Industry

北京 • BEIJING

内 容 简 介

本书涵盖了最新的 Web 规范和各种浏览器功能特性，如果您要使用 HTML、XHTML、CSS、文档对象模型（DOM）和 JavaScript 进行开发，那么它正是您要寻找的一站式终极资源宝库。本书为富互联网应用程序的设计者提供了全面而翔实的参考。在本书的帮助下，您设计的应用程序能够在 Internet Explorer 7、Firefox 2、Safari、Opera 等现代浏览器中畅通无阻。

版权贸易合同登记号　图字：01-2007-2018

图书在版编目（CIP）数据

Dynamic HTML 权威指南：第 3 版 /（美）古德曼（Goodman,D.）著；杨文俊，张苏，王卓译.—北京：电子工业出版社，2009.6

书名原文:Dynamic HTML：The Definitive Reference，3rd Edition

ISBN 978-7-121-08775-2

I. D…　II.①古…②杨…③张…④王…　III.超文本标记语言，HTML—程序设计　IV.TP312

中国版本图书馆 CIP 数据核字（2009）第 071370 号

责任编辑：王继花
印　　刷：北京天宇星印刷厂
装　　订：三河市鑫金马印装有限公司
出版发行：电子工业出版社
　　北京市海淀区万寿路 173 信箱　邮编 100036
开　　本：860×1092　1/16　印张：54.25　字数：1680 千字
印　　次：2009 年 6 月第 1 次印刷
定　　价：99.80 元

凡所购买电子工业出版社图书有缺损问题，请向购买书店调换。若书店售缺，请与本社发行部联系，联系及邮购电话：（010）88254888。

质量投诉请发邮件至 zlts@phei.com.cn，盗版侵权举报请发邮件至 dbqq@phei.com.cn。服务热线：（010）88258888。

前言

Preface

在我的咨询和开发工作中，一直需要一部工具书来辅助工作，正是这个想法驱使我撰写了本书并在最近推出了第三版。关于动态HTML（Dynamic HTML，简称DHTML），市面上充斥着庞杂的在线参考文档和为数众多的书籍，它们有的来自于Netscape、Microsoft等公司，有的则来自于W3C（World Wide Web Consortium，万维网联盟）。伴随着这些资料，我在一个古老的浏览器上努力工作了很多天才真正弄明白DHTML的玄妙。DHTML包含数百个既相似又存在细小差异的术语，如HTML属性、样式单和可脚本化的对象模型等，我这人类的小脑瓜可无法装下它们。与此同时，任何一个浏览器生产商都无法告诉我某个功能在另一个浏览器上的兼容性。从那时开始，我决定编撰自己的参考手册。我最初的想法是将现有资源直接进行整合，并以我的项目开发经验作为补充。然而随着对现有文档检查工作的不断深入，形势变得越来越严峻。来自于浏览器生产商的开发文档往往包含着前后矛盾、不完整甚至错误的信息，即使W3C的文档也不例外。因此，从一开始我就告诫自己，不能相信在文档里读到的任何信息，必须尽可能多地亲自尝试不同种类、不同版本的浏览器。我对数百个相关的HTML特性、CSS属性、对象属性、对象方法和事件都进行了反复测试，为本书的第1版打下了基础。在彻底领悟它们之前，夜以继日地编码和撰写的那几个月都成为了值得留恋的历史。经过了那次洗礼，在编写本书第2版时就相对轻松了一些。尽管W3C推出的DOM的为对象模型带来了全新的概念。但将W3C规范中的理想世界与Mozilla浏览器的实际开发工作结合起来是一个耗时巨大的工程，它意味着需要彻底梳理浏览器源代码和不同处理级别的缺陷报告（中断开发进行修复、即将修复或日后再修复）。但是，一旦将这些工作拓展到不断成长的IE（Internet Explorer）世界之后，信息数量的增长更是达到了一个令人难以置信的地步。在IE里，包含着一万五千多个由不同document 对象支持的大量属性、方法和事件句柄。

本书的第3版的产生则来自于几个方面的驱动。首先，尽管IE浏览器已经稳定了若干年，但浏览器世界并没有停滞不前。Mozilla的浏览器已经逐渐成为很多用户的第2个不错选择。而作为Macintosh操作系统的绑定浏览器，Safari也逐渐确立了它的地位，越来越多的的Mac OS X核心用户使用它来访问网页。面对这些浏览器的挑战，Opera家族也保持着快速的开发步伐，新推出的Opera 9就包含许多有风险的新功能，如Web Forms 2.0。与此同时，一些W3C工作组也完成（或基本完成）了第3版DOM和CSS模型的主体部分，它们的一些特性也已在某些非IE浏览器中得到了应用。与以往相比，现在要跟踪每个浏览器的开发进展变得更为困难。

正是以上这些原因，使我怀着激动的心情继续编写本书第3版。有了它，我的工作室中总会有一本触手可及的DHTML参考手册。哦，要是哪天这本书被用破了，我甚至准备好了胶带来修补它。

我想，这个世界上只有我能承诺，这本书将是十全十美的。尽管自本书第1版出版以来，DHTML脚本程序设计的可预见性和可靠程度一直在显著提高，但在产品供应商、标准规范与主流浏览器所呈现的现实之间依然存在着矛盾。正因为如此，事实才更需要呈现于纸端。这使得我回想起高中时的一位物理老师，他总是一边喊着对我们说："眼见为实"，一边迅速地演示一个光学幻象。期望我在这个领域所拥有的丰富经验能够帮助我穿透幻象，将真正的事实呈现在你们眼前。

背景知识

作为一本参考书，本书假设您对动态HTML有所涉猎，精通HTML并对客户端的JavaScript脚本编程有一定了解。您不需要是一位DHTML专家，但对DHTML要有基本的认识。对那些已经能够轻松直接手写Web页面（或至少能够修改可视化编辑器自动生成的HTML文本）的读者而言，网络上的在线教程将是快速入门的好途径。

本书内容简介

本书主要划分为3个主题：

第I部分　动态HTML参考

第I部分的各个章节概略性地介绍了JavaScript的核心内容和HTML、XHTML、CSS及DOM的主要概念，如标签（tag）、属性（attribute）、对象（object）、属性（property）、方法（method）和事件（event）等。在这些章节里，我花费了全部的时间来查询一个HTML元素所拥有的特性，或检验一个特定的对象属性能否在某个我期望的浏览器中正常运行。这些结果信息也将以一种精炼的方式呈现给读者。与此同时，有关兼容性问题的讨论还涉及到Safari和Opera这两个系列的浏览器。

第II部分　混合参考

第II部分以不同的视角剖析了第I部分中所介绍的信息。有时也许你能想起曾经用过的某个特性的名称，但始终无法在脑海中搜索到具体是哪个元素提供了这个特性。如果发生了这种情况，你就能在这一部分里查找这个属性、方法或事件类型，以找到所有的相关支持信息。（译注1）

第III部分　附录

本书中的若干附录针对HTML创作与脚本编程中的各种实用值提供了快捷的查找方法。附录D则包含了一些控制命令，针对用户可编辑的内容，可以在3种浏览器中使用这些命令。在本书的最后部分，还提供了一个词汇表，DHTML中一些容易引起混淆的新词汇在此做出了解释。（译注1）

约定

斜体主要用于：

- 路径名、文件名、程序名称、email地址及网址等。
- 术语的定义。

粗体主要用于：

- 关键字。
- GUI菜单项和按钮。

译注1：原书的第II部分和第III部分内容请读者访问网址：*http://www.china-pub.com/195581/*进行在线阅读。

等宽字体主要用于：

- 任何HTML、CSS或脚本编程术语，如HTML中的标签、属性名、对象名、方法及事件句柄等。
- 所有的HTML和脚本代码。

等宽斜字体主要用于：

- 方法与函数的参数、在实际使用中将会被真实值所代替的占位符。

在线参考文档

前两个版本的读者将会发现在本版中删去了有关DHTML标准与应用程序的相关章节。别害怕！为使第3版的参考内容更为翔实，我已经更新了这些章节（主要更新了实例）并提供了下载（*http://www.oreilly.com/catalog/dhtmlref3*）。这7个在线参考如下所述：

I The State of the Art: Standards
II Cross-Platform Compromises
III Adding Cascading Style Sheets to Documents
IV Changing Page Content and Styles
V Adding Dynamic Positioning to Documents
VI Scripting Events
VII XMLHttpRequest and Ajax

如何使用代码示例

本书的目的就是帮助读者出色地完成相关工作。一般说来，您也许会在自己的脚本和文档中使用本书中给出的shell脚本。除非您大量复制这些脚本，否则您并不需要每次都联系我们以获取示例代码的使用许可。例如，如果您将O'Reilly书籍中的示例制作成光盘并销售或分发，此时您就需要获得相关的许可。而在回答问题时引用本书中的代码作为例证则不需要获得任何许可。但是，如果要将本书中的大量代码写入您的产品代码或文档中，则需要获得许可。

我们将很感激您在引用本书内容时附上出处。引注往往包含标题、作者、出版商和ISBN号，例如："*Dynamic HTML: The Definitive Reference*，第3版，Danny Goodman. Copyright 2007 O'Reilly Media, Inc., 978-0-596-52740-2."。

如果您觉得对本书代码的使用超出了上文所规定的许可范围，您可以通过电子邮件与我们取得联系。我们的电邮是：*permissions@oreilly.com*。

联系我们

我们已尽力核验本书所提供的信息，尽管如此，仍不能保证本书完全没有瑕疵，而网络世界的变化之快，也使得本书永不过时的保证成为不可能。如果读者发现本书内容上的错误，不管是赘字、错字、语意不清，甚至是技术错误，我们都竭诚虚心接受读者指教。如果您有任何问题，请按照以下的联系方式与我们联系。

奥莱理软件（北京）有限公司
北京市 西城区 西直门 南大街2号 成铭大厦C座807室
邮政编码：100080
网页：*http://www.oreilly.com.cn*
E-mail：*info@mail.oreilly.com.cn*

O'Reilly & Associates, Inc.
1005 Gravenstein Highway North
Sebastopol, CA 95472
(800) 998-9938 (in the United States or Canada)
(707) 829-0515 (international/local)
(707) 829-0104 (fax)

与本书有关的在线信息如下所示。

http://www.oreilly.com/catalog/9780596527402（原书）
http://www.oreilly.com.cn/book.php?bn=978-7-121-08775-2（中文版）

北京博文视点资讯有限公司（武汉分部）

湖北省 武汉市 洪山区 吴家湾 邮科院路特1号 湖北信息产业科技大厦1402室
邮政编码：430074
电话：(027)87690813　　传真：(027)87690813转817
读者服务网页：*http://bv.csdn.net*
E-mail：
reader@broadview.com.cn（读者信箱）
bvtougao@gmail.com（投稿信箱）

致谢

无论过去几年中我编写了多少书，你得记住我的话，我不会因为那些累加的字符不断消耗纸浆、使森林遭劫而欢欣鼓舞。只是因为DHTML所涉及的内容不断膨胀，无论我如何精简，书页也仍然快速增加。

编写这样一部厚重的书是一件非常复杂的工程，它承载了许多人的辛勤工作，正是他们将哪些虚无的电子字符编印成华丽的篇章。我要感谢Tim O'Reilly，他一直忠于其“方便作者”的信条。他在构建科技动力源泉的过程中一直以追求高质量而享有令人尊敬的声望。他的编辑和出版团队也一直在极大的截稿压力下创造奇迹。

能够帮助读者释放才能，创造伟大的解决方案是给予我的最好的回报。正是你们的鼓励激发我更好地完成本次第3版图书的编写工作。我希望这本书能够指导你们了解客户端脚本编程和Web开发的最新进展。我衷心欢迎新读者们来到DHTML的世界，让我们在这个充满魔力和希望的世界里共同探索吧！

译者序

万维网诞生20年来，小小的Web发生了翻天覆地的变化，呆板的文字信息一统天下的日子早已离我们远去，一成不变的图片、动画等页面元素逐渐无法满足人们的需要，这也正是动态HTML出现并日渐成熟的一大主因。动态HTML并不属于一种专门的技术，它既不是某种语言，也不是插件，而是一种由各种技术综合而成的实用理念。无论是古老的HTML，还是不可或缺的CSS、JavaScript，甚至方兴未艾的AJAX技术，都是动态HTML的组成要素。但是，当将这些功能各异的技术、不同的实现规范，以及各种品牌和版本的浏览器都纳入考虑范围时，我们才会发现让网页完美地“动起来”并不是一件手到擒来的简单工作。正因为如此，Danny Goodman才编写了《Dynamic HTML: The Definitive Reference》一书，并且在新版本中不断地加入新的概念和内容，使得这本书始终是最好的动态HTML参考书籍之一。

作为一个Web开发者，这本书有3个方面深深吸引着我。首先，本书内容翔实，它涵盖了动态HTML开发中常用的HTML、DOM、CSS和JavaScript等诸多内容。虽然这些信息主要来自于各个官方规范，但作者对它们进行了深入的探究，其深度和准确性远非其他基础书籍可比。更难得的是，作者还将自己十几年的Web开发经验都灌注于本书中，实际应用中会遇到哪些常见问题，各种标签、对象的使用技巧等经验之谈贯穿始终。其次，由于网页相关技术发展历史悠久，目前在互联网上存在着各种规范、不同的浏览器品牌，以及各自的不同版本，这也给动态HTML的开发带来了不小的麻烦。本书作者结合官方规范，经过大量的实际测试，给出了详细的平台兼容性说明。这是本书的一大特色，同时也为所有Web开发人员规避兼容性错误提供了有力的帮助。根据这些信息，还可以从细微之处了解整个动态HTML的发展史。第三，秉承本书前两版的风格，作者依然在内容编排上下足了功夫，各项内容安排得井井有条。按字母顺序排列的各个参考内容十分便于读者查阅。有了这本书，读者可以告别以往漫无目的的搜寻工作，创建动态页面所需的各类信息都可以从本书中信手拈来。

总体说来，在各种基础教程充斥市面的今天，这本书算一个另类。它不会手把手地教大家如何一步一步地进行Web开发，但它始终在细微处给予读者睿智的提示。面对这本厚书，通篇阅读可不是一个明智的选择，正如作者所说，它应该成为开发人员的助手，常伴身边，不时被召唤以提供帮助。本书的内容十分丰富，读者在阅读中肯定会发现许多新鲜而又不失深刻的内容，还会获得许多曾经被忽略的重要知识，这也是我们在翻译本书中的亲身体会。

本书主要由杨文俊、张苏、王卓翻译。杨文俊负责全书的校对和审定。译文虽经多次修改和校正，但难免存在疏漏、缺点及错误，我们真诚地希望同行和读者不吝赐教。不胜感激（译注1）。

译注 1：由于本书篇幅较大，我们将原书的第 II 部分（第 6~9 章）和第 III 部分（附录 A~F）放到网上供读者在线阅读，请访问地址 *http://www.china-pub.com/195581/* 阅读这部分内容；同时为了节约成本和便于读者阅读，我们将原书版式进行了压缩，原书页码已做标示，供读者对照。本书索引所列页码为原书页码。

联系博文视点

您可以通过如下方式与本书的出版方取得联系。

读者信箱：*reader@broadview.com.cn*

投稿信箱：*bvtougao@gmail.com*

北京博文视点资讯有限公司（武汉分部）

湖北省 武汉市 洪山区 吴家湾 邮科院路特1号 湖北信息产业科技大厦1402室

邮政编码：430074

电　　话：027-87690813

传　　真：027-87690595

若您希望参加博文视点的有奖读者调查，或对写作和翻译感兴趣，欢迎您访问：*http://bv.csdn.net*

关于本书的勘误、资源下载及博文视点的最新书讯，欢迎您访问博文视点官方博客：*http://blog.csdn.net/bvbook*

O'Reilly Media, Inc.介绍

为了满足读者对网络和软件技术知识的迫切需求，世界著名计算机图书出版机构 O'Reilly Media, Inc. 授权电子工业出版社，翻译出版一批该公司久负盛名的英文经典技术专著。

O'Reilly Media, Inc. 是世界上在UNIX、X、Internet和其他开放系统图书领域具有领导地位的出版公司，同时也是联机出版的先锋。

从最畅销的《The Whole Internet User's Guide & Catalog》(被纽约公共图书馆评为20世纪最重要的50本书之一）到GNN（最早的Internet 门户和商业网站），再到 WebSite（第一个桌面 PC 的Web服务器软件），O'Reilly Media, Inc. 一直处于Internet发展的最前沿。

许多书店的反馈表明，O'Reilly Media, Inc. 是最稳定的计算机图书出版商 —— 每一本书都一版再版。与大多数计算机图书出版商相比，O'Reilly Media, Inc. 具有深厚的计算机专业背景，这使得O'Reilly Media, Inc. 形成了一个非常不同于其他出版商的出版方针。O'Reilly Media, Inc. 所有的编辑人员以前都是程序员，或者是顶尖级的技术专家。O'Reilly Media, Inc. 还有许多固定的作者群体 —— 他们本身是相关领域的技术专家、咨询专家，而现在编写著作，O'Reilly Media, Inc. 依靠他们及时地推出图书。因为 O'Reilly Media, Inc. 紧密地与计算机业界联系着，所以O'Reilly Media, Inc. 知道市场上真正需要什么图书。

目录

Table of Contents

第1章

HTML与XHTML参考

HTML and XHTML Reference

本章提供了一份由HTML标签（tag）和属性（attribute）所组成的完整清单，这些标签和属性都已列入W3C的HTML4.01及XHTML1.1推荐标准之中，或已在过去和目前的主流浏览器中得以实现。这个列表还包含了许多来自Web超文本应用技术工作组（Web Hypertext Application Technology Working Group，简称WHATWG）的内容，如Web Forms 2.0。此外，每一个标签和属性都提供了相应的版本信息，读者可以很方便地检查某个标签或属性是否能够在特定的浏览器中使用。同时也可以很清晰地看到，每个条目是在Internet Explorer（简称IE）、早期的Mozilla 网景浏览器（Netscape Navigator，简称NN）、Mozilla浏览器（简称Moz）、Safari（简称Saf）、Opera（简称Op）及W3C CSS规范的哪个版本中被首次引入。由于本书主要探讨动态HTML相关的内容，因此对应的历史时间表仅仅回溯到HTML 3.2版、Netscape Navigator 2和Internet Explorer 3。有关Mozilla基金会系列浏览器（如Firefox、Netscape 6及更高版本、Camino等）的相关资料，请查阅附录F（译注1）。虽然在Opera的一些早期版本中也支持部分DHTML的功能，但由于Opera 7与目前Opera中的绘制引擎同时推出，因此本书从2003年发布的Opera 7开始讲述相关内容。因此，更早版本的Opera浏览器也可能支持被标记为“Op 7”的条目。

除Opera之外，如果一个条目在上述所有浏览器版本之前已经存在，或者在新发布的浏览器中得以使用（如Mozilla和Safari），那么就简单地将其标记为“all”。如果某条目还没有得到应用，则将其标记为“n/a”。另外，当只有一个单独版本的浏览器支持它时，则用管道符将版本号括起来进行标示，例如，|4|代表只支持版本4。对于那些不再建议使用的条目，由于现在的浏览器为了向后兼容仍然会支持它们，而遗留下来的旧代码也可能使用这些条目，因此本书也列出了这些已被废止的条目。如果已不被推荐使用某个条目，而且已将它从一个标准中删除，那么就用一个“小于”号进行标示，如“<4”。与本书上一版本不同的是，由于在Netscape 6中第一次出现的条目，均包含在Mozilla的条目中，因此NN 指示符只支持Netscape Navigator 2-4版。接下来的内容将会介绍所有HTML元素共有的特性，本章按照HTML元素（或标签）的字母顺序来组织内容；在每一个元素的描述中，也按字母顺序来排列元素的特性。

为使读者方便地查询哪些HTML元素需要结束标签、哪些特性是可选属性还是必选属性，本书对参考条目进行了精心设计。除了某些需要不同DOM风格的元素之外，如Navigator 4的 layer 元素，脚本对象引用均使用W3C的DOM标准语法风格。尽管在列表中大量使用了W3C DOM的`document.getElementById()`语法格式，但如果IE要通过DOM脚本支持该元素的话，也可以使用`document.all`这一方式来引用此元素。在条目的描述中，会详细地说明它在各个浏览器具体实现时的显著区别，但是，本参考手册并不打算因此而成为一本普通意义上的漏洞大全。

在本章中，由于绝大多数DHTML浏览器都支持W3C HTML标准的格式，而且所有的读者对此也比较熟悉，因此，所有的示例代码均按照该格式进行编写。同时，这些代码也演示了最受推崇的XHTML代码风格，如小写的标签和属性名，以及用引号引起来的所有属性值。示例代码中不符合XHTML格式特征的情况是：空元素未使用结束标签（如XHTML中的`<br />`技术），以及未对那些在HTML中不需要的属性明确赋值（如input元素中类型为checkbox时的`selected`属性）。如果您的代码必须遵守XHTML验证，也很容易修改这些代码的风格（参见在线参考I）。

译注 1：此部分内容请读者访问 *http://www.china-pub.com/195581* 进行在线阅读。

为了更深入地理解本章所提及的事件属性相关的事件类型，请参见第3章。如果想查看HTML和 XHTML DTD支持的特定元素或属性，请参考附录E（译注2）。

属性值类型

由于很多HTML元素属性共享类似的数据要求，因此为使参考列表简洁明了，不可能详细描述列表中的每一项，所以本节只对一些公共的属性值类型进行详细说明。当属性中使用了上述某个属性值类型时，请参照本节的类型描述。

长度

长度值定义了文档空间状态的线性度量，如一个`table`元素的宽度。其度量单位可以是任何能够在屏幕上标识位置或空间的适用单位。HTML 中长度属性的度量单位统一使用像素（pixel，简称px），但在其他场合中，如在层叠样式单（见第4章）中设置这一属性时，度量单位可以用英寸（inche）、派卡（pica）、em（译注3）或其他相关度量单位来表示。当它用于定义距离一个元素边框的偏移量时，一个单一的数字值就可以表示一个长度。例如，坐标点（10,20）包含两个长度值，分别表示该点距离某个元素的左边框和上边框的像素值。另外，还可以使用百分数来表示应用于水平、竖直空间的长度值，比如`width="50%"`。我们不赞成使用与长度值相关的HTML属性，而推荐使用与之对应的CSS属性，以顺应严格的HTML 4和XHTML规范。

标识符

标识符（通常指定给`name`或`id`属性）是一个符合严格语法规则的名字。最重要的是，标识符中不允许出现空格。如果您需要多个单词来描述一个项目，可以使用骆驼拼写法（内部单词首字母大写，如thisIsASample），或者在单词间使用下划线进行分隔（如，this_is_a_sample）。除所有数字和字母外，大多数标点符号不允许在标识符中使用。同时，为避免与通过标识符来引用项目的脚本语言发生冲突，不使用数字作为标识符首字符是一种良好的编程习惯。

URI 与 URL

通用资源标识符（译注4）是一个用于网络内容地址的广义术语。通用资源定位器（译注5）是URI中的一种。由于大多数浏览器只关注URL，所以对网站建设者而言，可以将它们等同视之。通常URL应用于`href`和`src`属性，使用时既可以是完整路径（包括协议、主机、域等部分），也可以是相对于当前文档URL的相对路径。在支持脚本的浏览器中，能够接受URI值的属性也可以接受JavaScript伪协议（pseudo-protocol），即利用一段脚本语句或函数来作为目标链接。这种伪协议虽然已经被广泛使用，但它并不是一个正式的标准，而且在禁用JavaScript的浏览器中也无法工作。

语言代码

目前有一个详尽的标准代码列表用以标识世界上的口语与书面语。一种语言代码通常会包含一个主要的语言代码，如“en”代表英语，“zh”代表中文。常用的两字母的主要语言代码收录于ISO-639（查看代码摘录列表，可以访问*http://www.ietf.org/rfc/rfc1766.txt*）。通常为区分某种语言在不同国家或地区的使用习惯，还可以用一个可选的子代码（通过一个连字符与主代码分开）来标识主语言的一种特定实现。例如，虽然“en”代表英语，但“en-US”代表一种特定的英语——美式英语。由于语言代码的含意关系到元素属性的具体取值，因此浏览器必须支持一种特定的语言代码。

译注 2：此部分内容请读者访问 *http://www.china-pub.com/195581* 进行在线阅读。

译注 3：12 em 为 1 pica，6 pica 为 1 英寸，1 英寸=2.54 厘米。

译注 4：Universal Resource Identifier，简称 URI。W3C 中全称为 Uniform Resource Identifier。

译注 5：Universal Resource Locator，简称 URL。W3C 中全称为 Uniform Resource Locator。

对齐常量

由于对齐属性对于不同的元素组合传达了不同的含义，因此在使用常用的对齐属性align（以及次常用的valign属性）来描述对齐属性值时容易产生误解。再加上一些浏览器实现了很多专有值，这些问题使得在选择对齐属性值时很容易产生困惑。正因为如此，已经不再推荐使用这类属性。而CSS的text-align（水平方向）和vertical-align属性则令那些通过CSS进行布局控制的作者感到欣慰。对于这部分读者，本节内容可略过。

元素对齐可分为5种类别，每一种都有其适用的元素和允许的属性值。虽然浏览器可以接受的值并不区分大小写，但如果为了让这些属性值的适用性更加广泛，开发人员应该养成使用小写属性值的习惯，以便与传统的DTD相符合。

盒外对齐

第1种对齐属性决定环绕着元素外部矩形空间的文本的对齐方式。W3C中，这类HTML元素包括applet、iframe、img、input和object。IE还支持embed、fieldset和select，但它却不支持iframe元素。以下简述受到广泛支持的元素对齐属性设置，以及它们对元素和环绕文本内容的影响作用。

absbottom

使文本（包括下行字母）的最底端与元素的最底端保持在同一水平线上。

absmiddle

使文本高度（包括上行字母与下行字母）的中部与元素高度的中部水平对齐。

baseline

文本的基线与元素最下端水平对齐。注意，文本的下行笔画位于基线以下。

bottom

W3C认可的值，等同于baseline。

left

如果元素所在行之前已经有文本存在，则该元素会移动至下一行，并显示在相邻的最外层容器的左侧。元素之后的文本将紧跟元素前的文本，使得整段文本环绕在对象或图像的周围（这种效果称为浮动）。这是一个W3C认可的值。

middle

文本的基线与元素高度的中央对齐。这是一个W3C认可的值。

right

如果元素所在行之前已经有文本存在，则该元素会移至下一行，并显示在相邻的最外层容器的右侧。元素后的文本紧跟元素前的文本，使得整段文本环绕在对象或图像的周围（这种效果称为浮动）。这是一个W3C认可的值。

texttop

元素的最上边与其前面的文本的上行笔划位于同一水平线。

top

元素的最上边与同一行中最高的元素（文本或其他元素）的上边界对齐。这是W3C认可的值。

容器盒（containing box）内文本对齐

legend元素就像是为表单中fieldset元素提供的一个标注。table元素中的caption也起到类似的作用。这类元素的对齐方式决定了其文本相对于表单的fieldset边框或table占据的矩形空间的具体位置。现在的各种浏览

器对这种元素-属性对的支持往往大有不同。这类align属性设置主要包含如下几种。

bottom

文本底部与相关元素底部对齐或位于该元素盒下边。浏览器只会针对caption元素执行这种对齐方式。这是W3C认可的值。

center

文本位于相关元素盒的顶部或上部，并水平居中。虽然此值并未被W3C所认可，现在的浏览器依然支持在caption元素使用该属性，但legend元素在Safari或Opera浏览器中则无法使用该属性。

left

虽然W3C认可此值（这个值表明文本应该位于容器元素的左侧），但对于legend元素而言，浏览器将文本对齐在盒子的左上部；而对于caption元素，只有Mozilla浏览器会将标题文本放在元素的左边。

right

虽然W3C认可此值（这个值表明文本应该位于容器元素的右侧），但对于legend元素，浏览器将文本对齐在盒子的右上部；而对于caption元素，只有Mozilla浏览器会将标题文本放在元素的右边。

top

legend元素的文本将在其容器盒的左上部对齐，而caption元素的文本将在中上部对齐。这是W3C认可的值。

块元素的水平对齐

由于浏览器对p、div、h1~h6和hr中align属性处理各不相同，从而导致在使用这个分类时可能令人摸不着头脑。这些块元素通常会占据一个透明盒空间，其宽度和相邻的最外层容器的宽度相同。对大多数元素来说，容器是body元素，其宽度差不多和整个浏览器窗口的宽度一致。因此，一旦为一个包含少量字符的元素设定了align属性，那么无论其属性值来自于W3C认可的三个值中的哪一个——center、left或right，该元素的显示效果就像其本身已实现了对齐。事实上，此时元素还是在原来的位置，与其他body的内容占据着同样的宽度，只是内部的文本根据属性值进行了对齐。如果为元素设置了固定的宽度，你会发现对齐属性依然控制着元素的内部文本，只是元素紧贴着左边距而已。为了使得固定宽度的元素居中，那么必须把它放置到另一个全宽度的容器元素中，并将该容器的对齐方式设置为center。

更令人困惑的是，过渡版W3C HTML 4规范允许使用justify（两端对齐）值，而严格的HTML 4和所有的XHTML规范都已从对齐属性的文本对齐方式中去除了该值（除table元素组件之外）。浏览器在对齐这些元素时则支持justify值。

表格单元格中的文本水平对齐

在W3C规范中，table元素（tbody、tr、td等）的子节点内部的文本可以根据center、justify、left和right值进行对齐。但是直到IE 7，IE系列浏览器仍不支持表格组件中的justify对齐方式。如果须要在IE表格的单元格中调整文本，可以将文本包装在一个p或div容器中，再将该容器的align属性设置为justify。

元素内部竖直文本的对齐

在表格组件中进行竖直对齐须要使用valign属性，该属性中的可用值类似于align的属性值，如baseline、bottom、middle和top。

颜色

颜色值可以用十六进制三元组（hexadecimal triplet）或对应的明语（plain-language）来表示。十六进制三元

组由3对十六进制数字组成，每个数字的取值范围为从00到FF，分别对应于红、绿、蓝3个颜色成分。将3对数字组合在一起，并在前面再加一个“#”号就可以表示一种颜色，如#rrggbb。因此，在全红（#FF0000）中仅包含red值（FF），其他2个色调均为空（00），而纯蓝则由#0000FF表示。颜色的这种表示方法中也可以使用小写字母。

这种数值表达方式可以表示1600余万种颜色，然而并非所有的显示器都能够完全显示这上千万种颜色。因此，您或许希望将颜色的选择限制在更为适度的范围之内，即著名的**Web调色板**（web palette）。在流行的色彩深度设置下，一个可以适用于所有浏览器的颜色参考可以在此链接中获取：*http://www.w3school.com.cn/html/html_colors.asp*（译注6）。

HTML推荐标准指定了一组由16种颜色组成的基本颜色库，并辅以明语命名。注意，这些颜色名并不区分大小写的。这些颜色名与对应的十六进制三元组列表如下：

Black	#000000	Maroon	#800000	Green	#008000	Navy	#000080
Silver	#C0C0C0	Red	#FF0000	Lime	#00FF00	Blue	#0000FF
Gray	#808080	Purple	#800080	Olive	#808000	Teal	#008080
White	#FFFFFF	Fuchsia	#FF00FF	Yellow	#FFFF00	Aqua	#00FFFF

换句话说，将颜色属性值设置为bgcolor="Aqua"与bgcolor="#00FFFF"将会产生相同的颜色效果。

多年以前，Netscape制定了一个更长的明语颜色对照表，该颜色表最初由X Windows System调色板采用，因此称为X11颜色。附录A（译注7）列出了详细的X11颜色，目前主流版本的浏览器均可以识别这些颜色名称。（译注8）

共享的 HTML 元素属性

在HTML 4.x、XHTML 1.x及主流浏览器中，大多数的元素都有很多的通用属性。因此，为了避免在参考列表中重复地描述这些属性，仅在此处对这些通用属性进行详细的说明。在本章的余下内容里，虽然这些共享属性不会再出现在每一元素的属性列表中，但实际上它们在每个元素中都可以使用（在每个元素属性列表中均会标明对应的浏览器或标准文档的版本信息）。很明显，由于一些可绘制元素的属性不能用在非绘制元素上，因此少数共享属性也可能没有任何意义。例如，因为在页面上style元素本身没有提供任何的可绘制内容，使得用户不能使用tab键进行定位选择，这使得在style元素上运用tabindex属性毫无意义。在一些情况下，虽然W3C规范并没有为非绘制元素实现公共属性，但由于W3C DOM为这些属性制定了可支持脚本的属性，因此浏览器仍然可以支持它们。如须查阅HTML 4和XHTML DTD中对特定共享属性的支持情况，请参考附录E。所有的共享属性如下。

属性

accesskey	class	contenteditable	dir	disabled
hidefocus	id	lang	language	repeat
repeat-max	repeat-min	repeat-start	repeat-template	style
tabindex	title	unselectable	xml:lang	

accesskey

IE *4/5* NN *n/a* Moz *0.9* Saf *1* Op *7* HTML *4*（见下文）

accesskey="某字符"

可选

一个单独的字符键可以让一个元素获得焦点（在某些浏览器中），也可以激活一个表单控件或链接动作。浏览器和操作系统共同决定键盘组合键，用户须要同时按下这些组合键和快捷键才能激活链接。例如，大多数

译注6：原书中提供的链接已无法访问。

译注7：此部分内容请读者访问 *http://www.china-pub.com/195581* 进行在线阅读。

译注8：颜色名称与真实颜色的对应关系的可参见此网页：*http://www.w3school.com.cn/html/html_colornames.asp*。

Windows浏览器须要用户同时按下Alt键和指定的快捷键，而Mac中的浏览器则使用Ctrl键。与此同时，在Opera中则使用完全不同的组合序列Shift+Esc+快捷键。

虽然这里列出的访问快捷键是一个被广泛共享的属性，但并非所有浏览器版本都完全实现了这一属性。例如，在HTML 4和大多数浏览器里，该属性可以使用于下列元素中：a、area、button、input、label、legend和textarea。IE 4则向这一列表增加了applet、body、div、embed、isindex、marquee、object、selec、span、table和td元素，同时删除了label和legend元素。而IE 5则增加了其他所有的可绘制元素，但要注意的是，除了input和其他表单相关的元素，必须为IE 5及更高版本中的元素分配一个tabindex属性（即使是简单地将其值设置为0），这样访问快捷键组合才可以让元素获得焦点。其他所有主流浏览器均支持在select元素中使用这一属性。

示例

```
<a href="http://www.megacorp.com/toc.html" accesskey="t">Table of Contents</a>
<h2 class="subsection" accesskey="2" tabindex="0">Part Two</h2>
```

值 单个字符。

默认值 无。

对象模型引用方式

```
[window.]document.links[i].accessKey
[window.]document.anchors[i].accessKey
[window.]document.formName.elementName.accessKey
[window.]document.forms[i].elements[j].accessKey
[window.]document.getElementById(elementID).accessKey
```

class

IE 3 NN 4 Moz 0.9 Saf 1 Op 7 HTML 4

```
class="类名 1[ ...类名 N]"
```

可选

这个标识符可以将元素和为某个类选择符而定义的样式单规则关联起来。请参见在线参考III。注意，class属性只能用于可见（可绘制）元素。

示例 `<a class="chapTitle" name="chapter3" id="chapter3">Chapter 3</a>`

值 区分大小写的标识符。可为此属性值分配多个类，只须将引号内的类名用空格隔开即可。

默认值 无。

对象模型引用方式 `[window.]document.getElementById(elementID).className`

contenteditable

IE 5.5 NN n/a Moz 1.7 Saf 1.3/2 Op 9 HTML n/a

```
contenteditable="功能开关"
```

可选

此属性值为布尔开关，以允许或禁止用户直接在Web页面上编辑元素内容。有关在Windows的IE浏览器（以及最近版本的Safari和Opera）中使用脚本进行编辑的详细信息，请访问*http://msdnmicrosoft.com/workshop/author/editing/editing_entry.asp*。有关Mozilla浏览器的相关内容，请参见第2章中有关document.designMode的属性描述。

示例 `<p id="userArea" contenteditable="true">Enter your text here.</p>`

值 `true | false | inherit`

默认值 `inherit`

对象模型引用方式 `[window.]document.getElementById(elementID).contentEditable`

dir

IE 5 NN n/a Moz 0.9 Saf 1 Op 7 HTML 4

`dir="显示方向"` 可选

决定文字绘制方向是从左到右，还是从右到左。此属性往往与lang属性配合使用，使用它时必须指定一个文字绘制方向以替代原值。

示例 `<a lang="ar" dir="rtl">Some Unicode Arabic text characters here</a>`

值 `ltr | rtl`

默认值 `ltr`

disabled

IE 4/5.5 NN n/a Moz 0.9 Saf 1 Op 7 HTML 4

`disabled="功能开关"` 可选

此属性值为布尔开关，以允许或禁止用户激活或访问一个元素。在HTML 4中此属性只对交互性的表单元素有效，而在IE 5.5及更高版本（仅限于Win 32系统）中也适用于其他大多数可绘制元素。被禁用的元素通常会被“灰化”，以便与其他元素相区别。

示例 `<input type="submit" name="sender" disabled="true">`

值 `true | false`

默认值 `false`

对象模型引用方式 `[window.]document.getElementById(elementID).disabled`

hidefocus

IE 5.5 NN n/a Moz n/a Saf n/a Op n/a HTML n/a

`hidefocus="功能开关"` 可选

此属性为布尔开关，以允许或禁止浏览器在元素获得焦点时在其外部显示虚线矩形框。如果在默认情况下某个元素可被聚焦，或者它拥有tabindex属性集，则该元素能够获得焦点。在一些只能使用键盘访问的情形下，焦点是必需的。一旦将此属性设置为“true”，则会失去可视化的焦点状态提示。这一属性只能在IE 5.5或更高版本（仅限于Win32系统）中使用。

示例 `<input type="image" src="sendme.jpg" hidefocus="true">`

值 `true | false`

默认值 `false`

对象模型引用方式 `[window.]document.getElementById(elementID).hideFocus`

id

IE 4 NN 4 Moz 0.9 Saf 1 Op 7 HTML 4

`id="元素标识符"` 可选

此属性是一个独一无二的标识符，它使得所属元素区别于文档中所有的其他元素可以将其属性值视为一个ID选择器，以便将一个元素与某个样式规则联系起来。另外，也可以为某元素分配一个唯一的ID，然后同时使用class属性，以便将该元素包括在一个元素组之中。详情请参见在线参考III。

大多数浏览器允许为非可绘制元素也指定id属性，不过如果代码须要通过验证，则要注意的是W3C的HTML 4规范和XHTML DTD并不允许这种做法。然而，由于所有的W3C DOM元素对象都具有一个id属性，因此如果脚本中必须引用这些元素，就很自然地要求为非可绘制元素也分配一个id。否则，在脚本中就须要使用其他方法来引用这些元素，如利用document.getElementsByTagName()返回的元素对象数组。

以前元素中只具有name属性，而现在id的属性值与name属性的值相同，并起到与name属性相同的功能。但目前浏览器表单控件仍需要name属性，以便将名称/值对（name/value pair）信息随表单一起提交至服务器，

同时，作为锚点（anchor）的元素也需要name属性。因此，id属性目前尚不能完全取代name属性。如果想要在脚本中引用一个元素，一定要记住为这个元素的id属性分配一个标识符。

示例 `<h2 id="sect3Head">Section Three</h2>`

值 区分大小写的标识符。

默认值 无。

对象模型引用方式 `[window.]document.getElementById(elementID).id`

13

lang

IE *4* NN *3* Moz *0.9* Saf *1* Op *9* HTML *4*

`lang="语言代码"` 可选

此属性指定了元素属性值与内容中使用的语种。这个属性提供的信息可以帮助浏览器正确地绘制元素内容及其所涉及的相关细节，例如如何处理连字（当被特定的字体支持，或是某种书面语言要求的时）、引号和断字等。其他应用程序和搜索引擎也可以使用这些信息，以辅助进行拼写检查时字典的选择和索引的创建。

示例 `<span lang="ede">Deutsche Bundesbahn</span>`

值 不区分大小写的语言代码。

默认值 由浏览器决定默认值。

对象模型引用方式 `[window.]document.getElementById(elementID).lang`

repeat,repeat-max,repeat-min, repeat-start, repeat-template

IE *n/a* NN *n/a* Moz *n/a* Saf *n/a* Op *9* HTML *n/a*

（见下文） 可选

Web Forms 2.0（*http://www.whatwg.org*）可以将任意HTML元素转换成一个重复块（repetition block），即一个或多个元素所组成的分组，但这须要使用文档中数目不定的多个实例。例如，一个包含产品号及数量的订购表单可以由一个table元素组成。在这个table中，可以放置两个文本输入控件，其中一个是number类型的input元素，而另一个则是text类型的input元素，后者还须要使用一个表达式以匹配公司的产品编号系统。在用户输入完成后，后台调用的AJAX会算出整个费用，并显示在表格中的另一个单元格内。如果用户期望继续添加订购，那么就须要再向表格添加一行，并在行内包含相同控件。但对应的名称要与之前的有所不同，以便浏览器能够正确提交每一行中的不同数据。尽管DHTML脚本能够在表格中执行这些动作，但Web Forms 2.0可以在不进行脚本编程的情况下使用重复块机制处理添加、删除及重复元素的移动等功能。

通过指定repeat属性，可以将一个表格行（或其他HTML容器）转换成一个重复块，代码如下所示：

```
<tr id="item" repeat="template">
   <td><input type="number" name="row[item].quantity" value="1"></td>
   <td><input type="text" name="row[item].product" value=""></td>
   ...
</tr>
```

14

从这段代码可以看到，每个控件的名称看起来就像是JavaScript的数组符号，并将重复块的ID（此时是item）作为数组的索引。表单提交时，控件名称中的数组符号会被一个从0开始整数所取代，本例中第一行就是row0.quantity和row0.product。

而add、move-down、move-up和remove类型的input和button元素将控制这些重复块。另外，这些控件的template属性则指向重复块外层容器的ID。

如果希望在表格开头使用多个重复行，那么只须要为单独的一行编写HTML代码，然后将repeat-start属性设置为初始显示时希望出现的行数。同样，也可以分别设置repeat-max和repeat-min属性来控制添加或删

除时重复块数目的上下限。

示例

```
<form action="..." method="POST">
<p>
<table>
<tr><th>Quantity</th><th>Item Number</th></tr>
<tbody>
    <tr id="item" repeat="template" repeat-start="2" repeat-min="1">
        <td><input type="number" name="row[item].quantity" value="1"></td>
        <td><input type="text" name="row[item].product" value=""></td>
        <td><input type="remove" /></td>
    </tr>
</tbody>
</table>
<button type="add" template="item">Add Item</button>
<p>
<input type="submit" />
</p>
</form>
```

值 对repeat而言，可以使用template常量或针对某个硬编码项的整数；对repeat-max、repeat-min和repeat-start，请使用一个整数；而针对repeat-template，请使用文档中某个作为模板使用的元素的ID值。

默认值 repeat-max：最大整数值；repeat-min：0；repeat-start：0（尽管Opera建议值为1）。

style

IE 4 NN 4 Moz 0.9 Saf 1 Op 7 HTML 4

`style="样式单特性"` 可选

通过本属性能够为当前元素设置一个或多个样式单规则属性。您可以使用CSS（仅针对Navigator 4浏览器）或JavaScript对style属性进行赋值。注意，style属性只能用于可见（可绘制）元素。

示例 `<span style="color: green; font-size: 18px">Big, green, and bold</span>`

值 应该将一个完整的CSS样式单规则放置在一对引号中。如须设置多个style属性值，请使用分号进行分隔。本书第4章将详细讲述样式单属性。

默认值 无。

对象模型引用方式 `[window.]document.getElementById(elementID).style`

tabindex

IE 4 NN n/a Moz 0.9 Saf 1 Op 7 HTML 4

`tabindex="整数值"` 可选

此属性为一个整数值，表示此元素在文档内所有可聚焦元素中进行跳格（tab）选择时的序列号。这种跳格选择顺序依照一个严格的规则。当用户在一个页面内进行跳格选择时，那些tabindex属性值不为0的元素将排列在前面。此时，无论元素在页面或文档中处在什么物理位置，焦点总是从tabindex属性值最小的元素开始聚焦，然后随着该属性值的增长，元素依次获得焦点。如果两个元素的tabindex属性值相同，则在文档中出现较早的元素将首先获得焦点。此后，那些未设置tabindex属性或其属性值为0的元素才能获得焦点。就这些元素而言，它们对获得焦点的次序与其在文档中出现的先后顺序相同。值得注意的是，并不总是从第一个焦点元素开始重载当前页面。因此，当可聚焦元素的逻辑顺序与它们在源代码中的顺序有所不同时，控制跳格选择的顺序显得更为重要。例如，可以不必逐个越过表格行而直接选择到表格末尾。

在HTML 4和大多数浏览器里，该属性可以使用在下列元素中：a、area、button、input、label、legend和textarea。IE 4向这一列表中增加了applet、body、div、embed、isindex、marquee、span、table和

td元素。而IE 5则能支持所有其他的可渲染元素。在IE浏览器和Mozilla 1.8版及更新版本中，如果将tabindex属性值设置为负数，则该元素将会从跳格选择序列中完全去除。

示例 `<a href="chapter3.html" tabindex="3">Chapter 3</a>`

值 0–32 767之间的任一整数。在IE浏览器和Mozilla 1.8版及更新版本中，如果将tabindex属性值设置为负数，则在进行跳格选择时会略过该元素。

默认值 无。

对象模型引用方式 `[window.]document.getElementById(elementID).tabIndex`

title

IE 3 NN n/a Moz 0.9 Saf 1 Op 7 HTML 4

`title="提示文字"`

可选

此属性描述一个元素的提示信息。对HTML元素而言，使用该属性将在页面上产生一些可视内容。光标在元素上驻留一段时间后，浏览器会将title的属性值以工具提示框（tooltip）的形式显示出来。然而，与table相关的col元素并不能显示任何内容，因此其title属性仅仅只是一个不可见的提示而已。如果要在表格中生成工具提示框，请在相关的table、tr、th或td元素中设置title属性。

而这个工具提示框的字体和颜色取决于浏览器，并且无法通过脚本进行修改。在IE/Windows系统中，工具提示是一个标准的浅黄色小方块；在IE/Mac系统中，它的外观风格和Mac操作系统中帮助系统里的卡通泡泡完全一样。而Mozilla浏览器的工具提示就是一个小方块，其外观与具体的操作系统版本无关。如果不设置该属性，则工具提示不会显示。

您可以为该属性设置任何描述性的文本。由于不是每个人都能看到这个提示，因此不要在这里放置关键的任务提示信息。另外，那些为满足无障碍网页标准而精心设计的浏览器也可以利用该属性，从而为视力受损的上网者朗读链接或非文本元素的提示信息。因此，不要忽视这个可能有用的小帮手，它的确能够描述页面上某个元素存在的目的。

尽管此属性的适用元素列表可以追溯到IE 3和HTML 3.2，但目前大多数支持它的元素是在IE 4和HTML 4.0后才逐渐出现的。

示例 `<span title="United States of America">U.S.A.</span>`

值 任何字符串。该字符串应该位于一对引号之中，双引号是推荐的标准格式，但也可使用单引号。

默认值 无。

对象模型引用方式 `[window.]document.getElementById(elementID).title`

Unselctable

IE 5.5 NN n/a Moz n/a Saf n/a Op 9 HTML n/a

`unselectable="功能开关"`

可选

此属性值为布尔开关，以允许或禁止用户选择该元素的任意部分。

示例 `<p unselectable="on">...</p>`

值 `on | off`

默认值 `off`

xml:lang

IE *n/a* NN *n/a* Moz *n/a* Saf *n/a* Op *9* HTML *X1.0*

`xml:lang="语言代码"` 可选

此属性表示HTML中对应lang属性的XML版本号，该版本号与W3C XML推荐标准中所述内容一致。只有在支持XML名空间的浏览器内，遵守XHTML标准的文档才能使用这个属性。XML处理器（而不是浏览器本身）会根据此属性的赋值决定如何解析文本并进行显示。例如，只有当浏览器和操作系统均支持对应的语言脚本规则时才能显示该元素。另外，即使它重复了xml:lang属性的设定，浏览器文档也应该继续使用lang属性。

示例 `<span lang="de" xml:lang="de">Deutsche Bundes bahn</span>`

值 不区分大小写的语言代码。

共享的事件处理属性

处理程序	IE/Windows	NN	Mozilla	IE/Mac	Safari	Opera	HTML
onactivate	5.5	n/a	n/a	n/a	n/a	n/a	n/a
onbeforeactivate	5.5	n/a	n/a	n/a	n/a	n/a	n/a
onbeforecopy	5	n/a	n/a	n/a	1.3/2	n/a	n/a
onbeforecut	5	n/a	n/a	n/a	1.3/2	n/a	n/a
onbeforedeactivate	5.5	n/a	n/a	n/a	n/a	n/a	n/a
onbeforeeditfocus	5	n/a	n/a	n/a	n/a	n/a	n/a
onbeforepaste	5	n/a	n/a	n/a	1.3/2	n/a	n/a
onblur	3/4	2	0.9	3/4	1	4	4
onclick	3/4	2	0.9	3/4	1	4	4
oncontextmenu	5	n/a	n/a	n/a	1.2	n/a	n/a
oncontrolselect	5.5	n/a	n/a	n/a	n/a	n/a	n/a
oncopy	5	n/a	n/a	n/a	n/a	n/a	n/a
oncut	5	n/a	n/a	n/a	n/a	n/a	n/a
ondblclick	4	4	0.9	4	1	7	4
ondeactivate	5.5	n/a	n/a	n/a	n/a	n/a	n/a
ondrag	5	n/a	n/a	n/a	1.3/2	n/a	n/a
ondragend	5	n/a	n/a	n/a	1.3/2	n/a	n/a
ondragenter	5	n/a	n/a	n/a	1.3/2	n/a	n/a
ondragleave	5	n/a	n/a	n/a	1.3/2	n/a	n/a
ondragover	5	n/a	n/a	n/a	1.3/2	n/a	n/a
ondragstart	5	n/a	n/a	n/a	1.3/2	n/a	n/a
ondrop	5	n/a	n/a	n/a	1.3/2	n/a	n/a
onfilterchange	4	n/a	n/a	n/a	n/a	n/a	n/a
onfocus	3/4	2	0.9	3/4	1	4	4
onfocusin	6	n/a	n/a	n/a	n/a	n/a	n/a
onfocusout	6	n/a	n/a	n/a	n/a	n/a	n/a
onhelp	4	n/a	n/a	5	n/a	n/a	n/a
onkeydown	4	4	0.9	4	1	5	4
onkeypress	4	4	0.9	4	1	5	4
onkeyup	4	4	0.9	4	1	5	4
onlosecapture	5	n/a	n/a	n/a	n/a	n/a	n/a
onmousedown	4	4	0.9	4	1	5	4

续表

处理程序	IE/Windows	NN	Mozilla	IE/Mac	Safari	Opera	HTML
onmouseenter	5.5	n/a	n/a	n/a	n/a	n/a	n/a
onmouseleave	5.5	n/a	n/a	n/a	n/a	n/a	n/a
onmousemove	4	4	0.9	4	1	5	4
onmouseout	3/4	2	0.9	3/4	1	4	4
onmouseover	3/4	2	0.9	3/4	1	4	4
onmouseup	4	4	0.9	4	1	5	4
onmousewheel	6	n/a	n/a	n/a	1.3/2	n/a	n/a
onmove	5.5	n/a	n/a	n/a	1	n/a	n/a
onmoveend	5.5	n/a	n/a	n/a	n/a	n/a	n/a
onmovestart	5.5	n/a	n/a	n/a	n/a	n/a	n/a
onpaste	5	n/a	n/a	n/a	n/a	n/a	n/a
onpropertychange	5	n/a	n/a	n/a	n/a	n/a	n/a
onreadystatechange	4	n/a	n/a	4	n/a	n/a	n/a
onresize	4	n/a	n/a	4	1.2	n/a	n/a
onresizeend	5.5	n/a	n/a	n/a	n/a	n/a	n/a
onresizestart	5.5	n/a	n/a	n/a	n/a	n/a	n/a
onselectstart	4	n/a	n/a	4	1.3/2	n/a	n/a

伴随着支持脚本的浏览器的发展历史，共享的事件处理属性的演变也很曲折。所有来自于IE 4、Mozilla 0.9和HTML 4的可渲染元素均拥有公共的鼠标和键盘事件处理属性，但在更早的浏览器中仅仅交互型元素能使用部分事件属性。例如，那些总是响应鼠标点击的元素（如表单的button控件、链接和图像映射）须要支持onclick事件。而链接和图像映射则往往须要支持mouseover和mouseout这两个属性。微软公司为其浏览器增加了大量事件处理属性指令，而苹果公司也在其Safari 1.3/2中继承了其中的一部分，并首先将它们应用于其Dashboard小程序。事件类型的更多细节或其他尚未提及的事件类型，请参见第3章。

标签参考（按字母次序排列）

<a>

IE *all* NN *all* Moz *all* Saf *all* Op *7* HTML *all*

```
<a>...</a>
```

HTML 结束标签：必选

根据其中name和href属性的状态，此元素非常适合扮演锚链或链接的角色。作为一个锚链时，此元素可以定义文档中一个已命名的位置。有了它，任何URL都可以通过在末尾附加一个“#”和锚链名称来引用文档URI。例如，*http://www.example.com/contents#a-c*。该名称一般会赋值给name属性，而在较新的浏览器中则会赋值给id属性。在默认情况下，定义为锚链的内容与环绕其四周的body元素的内容并没有视觉上的差异。

如果将一个URI赋值给href属性，那么这个元素就能化身为超文本链接的入口点。激活这个链接后，页面往往会跳转到href属性对应的URI地址，有时也可能只是将其他多媒体文件载入一个帮助程序或插件中，而不改变原来的页面。除非通过样式单修改了外观，否则浏览器中的链接总有着代表性的显眼外观，例如具有下划线的一串文字（或有边框的一个对象），并且其颜色也异于周围内容的当前颜色。链接有三种状态：未访问、已被用户激活及曾访问过，可以使用属性为链接的这些状态分别指定不同的颜色。但这种颜色控制方式并不受推荐，可以使用CSS的伪类进行控制，如:link、:active、:visited及新增的:hover状态。如果a元素中name（或id）和href属性均被赋值，那么它既是一个锚链也是一个链接。

示例

```
<a name="anchor3" id="anchor3">Just an anchor named "anchor3."</a>
```

```
<a href="#anchor3">A link to navigate to "anchor3" in the same document.</a>
<a name="anchor3" id="anchor3" href="http://www.example.com/index.html">
Go from here (anchor 3) to home page.</a>
```

对象模型引用方式

```
[window.]document.links[i]
[window.]document.anchors[i]
[window.]document.getElementById(elementID)
```

元素专有属性 charset、coords、datafld、datasrc、href、hreflang、methods、name、rel、rev、shape、target、type、urn

元素专有事件处理属性 无。

从第4版Navigator开始，作为锚链的元素不再拥有事件处理程序。

charset

IE *n/a* NN *n/a* Moz *0.9* Saf *all* Op *7* HTML *4*

```
charset="字符集"
```

可选

链接另一端的文本内容的编码方式。

示例 `<a charset="csISO5427Cyrillic" href="moscow.html">Visit Moscow</a>`

值 是否区分大小写取决于字符集注册表。

默认值 由浏览器决定。

coords

IE *6* NN *n/a* Moz *all* Saf *all* Op *7* HTML *4*

```
coords="坐标 1, ... 坐标 N"
```

可选

尽管该属性是为a元素而定义，但它也常因客户端图像映射而在area元素中得到使用。我们会发现，area元素“继承”了很多来自于a元素的属性和行为。详细内容请参见area元素。

datafld

IE *4* NN *n/a* Moz *n/a* Saf *n/a* Op *n/a* HTML *n/a*

```
datafld="列名称"
```

可选

此属性用于实现IE浏览器中的数据绑定功能，可以使用远程数据源中的列名代替link元素中的href属性。数据来源列必须包含一个有效的URI地址（无论是相对还是绝对地址）。同时，元素中还必须设定datasrc属性。注意，只能在IE 5/Mac操作系统下实现文本文件数据源的绑定。

示例 `<a datasrc="DBSRC3" datafld="newsURL">Late-Breaking News</a>`

值 区分大小写的标识符。

默认值 无。

对象模型引用方式

```
[window.]document.links[i].dataFld
[window.]document.getElementById(elementID).dataFld
```

datasrc

IE *4* NN *n/a* Moz *n/a* Saf *n/a* Op *n/a* HTML *n/a*

```
datasrc="数据源名称"
```

可选

为实现IE数据绑定功能，此属性用于指定页面内object元素的ID值，以便在进行远程数据访问时加载数据源对象。插入a元素的数据源内容由datafld属性决定，见上文。注意，只能在IE 5/Mac操作系统下实现文本文件数据源的绑定。

示例 `<a datasrc="DBSRC3" datafld="newsURL">Late-Breaking News</a>`

<a>

值 区分大小写的标识符。

默认值 无。

对象模型引用方式

```
[window.]document.links[i].dataSrc
[window.]document.getElementById(elementID).dataSrc
```

href

IE *all* NN *all* Moz *all* Saf *all* Op *7* HTML *all*

`href="URI"` 链接必须使用此属性

此属性指定了链接的目的URI地址。当这个URI地址是一个HTML文档时，该文档会被载入当前浏览器窗口（默认情况）或其他目标窗口（由target属性决定）。如果是其他文件格式，浏览器可能将目标内容载入插件或将目标文件保存在客户端所在的机器上。如果缺少href属性，则该元素无法区分自身是一个可点击的链接或仅仅只是一个锚链（如果已设置了name或id属性）。

示例

```
<a href="part1/chap3.html">Chapter 3</a>
```

值 可以是任何有效的URI地址，包括完整或相对的URL、同一页面上的锚链（锚链名以“#”号作为前导）、非标准的JavaScript伪URL（在支持脚本的浏览器中，它会触发一段脚本指令而不是直接导航到某个目的地）。注意，如果脚本在访问者的浏览器中禁止运行，则使用JavaScript伪URL的链接将会失效。此外，搜索引擎的蜘蛛程序也无法跟踪这种链接。

默认值 无。

对象模型引用方式

```
[window.]document.links[i].href
[window.]document.getElementById(elementID).href
```

其他所有涉及URL构成元素（如协议和主机名等）的链接对象属性。请参见第2章中的a元素说明。

Hreflang

IE *6* NN *n/a* Moz *09* Saf *all* Op *n/a* HTML *4*

`hreflang="语言代码"` 可选

此属性描述了链接目标文档所采用的语言代码。使用此属性时，须要同时设定href属性。在可能的情况下，这个属性主要用于告知浏览器为新的字符集做好准备。

示例

```
<a hreflang="HI" href="hindi/Chap3.html>Chapter 3 (in Hindi)</a>
```

值 不区分大小写的语言代码。

默认值 由浏览器决定默认值。

methods

IE *4* NN *n/a* Moz *n/a* Saf *n/a* Op *n/a* HTML *n/a*

`methods="http方法"` 可选

此属性能够为链接目标的功能给出一定的预告。浏览器获取此信息后，可以根据链接目标的相关动作来显示特定的颜色或图像。

示例

```
<a href="http://www.example.com/cgi-bin/search?chap3" methods="get">Chapter 3</a>
```

值 可以使用一个或多个HTTP方法，方法间使用逗号进行分隔。

默认值 无。

对象模型引用方式

```
[window.]document.links[i].Methods
[window.]document.getElementById(elementID).Methods
```

name

IE *all* NN *all* Moz *all* Saf *all* Op *7* HTML *all*

22

`name="元素标识符"`

链接必须使用此属性

此属性是用来描述文档中锚链位置的传统方式。其他链接元素可以通过设置`href`属性来指向一个锚链，此时，它的值是一个以“#”锚链名称结尾的URL地址。如果a元素中未使用`name`或`id`属性，则该元素不能作为一个锚链使用。在目前的浏览器中，虽然此属性与`id`属性等价，但由于XHTML 1.0并不推荐使用该属性，因此建议同时使用`name`和`id`属性（赋以相同的标识符名称），以保证在所有的浏览器中均可顺利生成标识符。如果在元素中同时设置了`name`和`href`属性，则该元素既可以看作是一个锚链，也可以视为一个链接。

示例 `<a id="sect3" name="sect3">Section III</a>`

值 区分大小写的标识符。

默认值 无。

对象模型引用方式

```
[window.]document.links[i].name
[window.]document.anchors[i].name
[window.]document.getElementById(elementID).name
```


rel

IE *3* NN *n/a* Moz *0.9* Saf *all* Op *n/a* HTML *4*

`rel="链接类型"`

可选

此属性用于定义当前元素与链接目标之间的关系，它也可叫做“前向链接”。无论如何也不要将该属性与`href`定义的链接目标文档相混淆。虽然HTML 4推荐标准定义了几种链接类型，但最终还是由浏览器决定如何对待这些链接类型。此属性在将来的程序应用中有明显的发挥空间，例如，在静态导航条中可以将a元素作为一个按钮来指向系列文档中的前文或下文。但在当前应用中，主要还是在`link`元素中`rel`使用属性。须要注意的是，元素中必须首先包含一个`href`属性后才能使用`rel`属性。

示例 `<a rel="next chapter" href="chapter3.html">Chapter 3</a>`

值

不区分大小写，但其选择范围仅限于HTML 4认可的标准链接类型。如下所示：

`alternate`	`appendix`	`bookmark`	`chapter`	`contents`
`copyright`	`glossary`	`help`	`index`	`next`
`prev`	`section`	`start`	`stylesheet`	`subsection`

另外，IE 3定义了4种固定取值：`same | next | parent | previous`，但其中只有`next`和`previous`获得了后续IE版本的支持。

默认值 无。

对象模型引用方式

```
[window.]document.links[i].rel
[window.]document.getElementById(elementID).rel
```

23

rev

IE *3* NN *n/a* Moz *0.9* Saf *all* Op *n/a* HTML *4*

`rev="链接类型"`

可选

此属性值表示相反的链接关系。类似于`rel`，`rev`属性所能实现的功能依然取决于浏览器，特别是浏览器如何解释并渲染HTML 4规范中定义的不同链接类型。假设有A和B两个文档，其中均包含指向对方的链接。B文档中的`rev`属性值用来描述两个文档之间的关系，则A文档中就使用`rel`属性来描述。但在主流浏览器里，使用a元素的`rel`或`rev`属性的应用实例非常少。

示例 `<a rev="previous" href="chapter2.html">Chapter 2</a>`

<a>

值 元素可以使用的标准链接类型列表，请在类型间使用空格进行分隔，类型名称不区分大小写。HTML 4认可并支持的链接类型请参考上文中的rel属性。

默认值 无。

对象模型引用方式

```
[window.]document.links[i].rev
[window.]document.getElementById(elementID).rev
```

shape

IE *n/a* NN *n/a* Moz *all* Saf *all* / Op *7* HTML *4*

```
shape="外形"
```

可选

此属性定义了服务器端图像映射区域的外形，具体的坐标由coords属性指定。详细内容请参见area元素。

target

IE *all* NN *all* Moz *all* Saf *all* Op *7* HTML *all*

```
target="窗口或框架名称"
```

可选

如果并不准备在当前窗口或框架中载入目标文档，就可以设置target属性，从而在该属性值对应的窗口或框架中载入目标文档。此时，框架或窗口名称必须在对应目标中被指定为标识符。例如，可以将frame元素的name或id属性设置为该名称，或通过脚本方法window.open()的第二个参数在创建新窗口时为窗口命名。如果疏忽了这个属性，那么目标文档将会取代链接所在的文档。另外，如果现有的框架或窗口中没有对应的目标名称，则浏览器会打开一个新的窗口以显示目标文档。须要注意的是，只有在href属性已赋值给元素之后，此属性方能发挥作用。

由于一个链接元素只能指定一个目标文档和一个载入目的地，因此，如果期望一个链接能够改变多个框架的内容，不妨采用元素的onclick事件处理程序来激发一个脚本，以加载多个文档。在使用中，请将每个框架的location.href属性值设定为期望的URL值。

由于框架和窗口已经超出了单纯的文档标记范畴，因此HTML 4和XHTML规范中的严格DTD并不支持在任何元素中使用target属性。事实上，在严格的环境下框架根本无法设置文档内容，这正是将DTD从HTML 4和XHTML中剥离出来的目的。如果文档必须支持这些严格的DTD，并且又期望使用target属性，不妨在页面载入后通过脚本设置链接、图像映射和表单的属性。

示例

```
<a target="display" href="chap3.html#sec2">Section 3.2</a>
<a target="_top" href="index.html">Start Over</a>
```

值

如果已通过目标元素的name属性指定了框架或窗口名称，就可以使用这个区分大小写的标识符。可以使用下列已保留的目标名称常量之一。

_blank
: 浏览器为目标文档打开一个新窗口。

_content
: 此属性值仅能在Mozilla系列浏览器中使用。当链接出现在浏览器的侧边栏中时，使用此属性值后无论主内容框架名称是什么，目标文档均会在主内容框架中显示。

_parent
: 目标文档会取代当前框架的框架集文档，如果不存在，则依照“_self”执行。

_self
: 目标文档替代在链接所在窗口或框架里的现有文档。

_top

目标文档将会取代任何已加载的框架集窗口，从而占据整个浏览器。如果在窗口中不存在框架集，则依照“_self”执行。

默认值 _self

对象模型引用方式

```
[window.]document.links[i].target
[window.]document.getElementById(elementID).target
```

type

IE 6 NN n/a Moz n/a Saf n/a Op n/a HTML n/a

type="*MIME类型*" 可选

此属性为目标文档或资源的类型给出提示。此信息可以帮助浏览器为需要多媒体播放器或插件的资源提前做好准备。

示例 View Devil's Ghost slope

值 不区分大小写的MIME类型。可从此处获取MIME类型目录：*http://www.iana.org/assignments/media-types/*。

默认值 无。

urn

IE 4 NN n/a Moz n/a Saf n/a Op n/a HTML n/a

urn="*urn*" 可选

本属性表示href属性指定的目标文档的URN（Uniform Resource Name，统一资源名称）版本号。使用此属性主要是为将来URN格式的URI提供支持，IEFT（请参见RFC 2141）一直在讨论这个不断发展的推荐标准。尽管从IE 4以来，IE浏览器都支持这个属性，但这个属性依然未能取代href属性的位置。

示例 Chapter 3

值 一个形式为“urn:*名空间ID*:*名空间特征字符串*”的有效URN地址。

默认值 无。

对象模型引用方式

```
[window.]document.links[i].urn
[window.]document.getElementById(elementID).urn
```

<abbr>

IE n/a NN n/a Moz 0.9 Saf all Op 7 HTML 4

<abbr>...</abbr> HTML结束标签：必选

abbr为文档主体文本内出现的缩写词提供了一种封装和枚举的机制。例如，假设一个网页须要包含您所在公司的地址。在该文档中的某一处，IA缩写代表Iowa（译注9）。那么这个缩写也许就会阻碍拼写检查器、语言翻译程序或语音合成器的正常工作，同时，搜索引擎在进行网页的相关度评估时也不会将单词“Iowa”考虑在内。但是，如果将“IA”这个缩写词汇放入abbr元素（并且为它指定一个title属性），那么就和使用全称没有任何差别。搜索引擎可以对该词计数，文本-语音转换程序也可以大声读出州的全称，而不是发出一些怪异的声音。正如HTML 4.0中引入的很多元素一样，这个元素主要也是用于辅助浏览器工作，虽然目前也许并没有得到应用，但在将来应该可以找到用武之地。

在Mozilla和Opera浏览器中，abbr元素的显示效果是一小段虚线。在Mozilla的上下文菜单中，为此元素提供了一个属性选项，这样就可以将元素内的属性与属性值都呈现给网页的浏览者。

译注9：美国的爱荷华州。

另外，还有一个名为acronym的元素也为首字母缩写词提供了类似的服务。在HTML 4.0推荐标准中，有一个名为“短语元素”的庞大分组，这两个属性就是其中的一部分。

示例

```
Ottumwa, <abbr title="Iowa">IA</abbr> 55334<br>
<abbr lang="de" title="und so weiter">usw.</abbr>
```

对象模型引用方式 [window.]document.getElementById(*elementID*)

元素专有属性 无。

元素专有事件处理属性 无。

<acronym>

IE *4* NN *n/a* Moz *0.9* Saf *all* Op *7* HTML *4*

<acronym>...</acronym>

HTML 结束标签：必选

acronym元素为文档主体文本内出现的首字母缩写词提供了一种封装和枚举的机制。例如，假设一个页面中包含一段关于国际贸易问题的讨论。在文档中的某处，“GATT”是“General Agreement on Tariffs and Trade”的缩写。这个缩写也许就会阻碍拼写检查器、语言翻译程序或语音合成器的正常工作，同时，搜索引擎在进行网页的相关度评估时也不会将单词“tariffs”考虑在内。”但是，如果将“GATT”这个缩写词汇放入acronym元素（并且为它指定一个title属性），那么就和使用全称没有任何差别。搜索引擎可以对该词计数，文本-语音转换程序也可以大声读出缩写词的全称。正如HTML 4.0中引入的很多元素一样，这个元素主要也是用于辅助浏览器工作，虽然目前也许并没有得到应用，但在将来应该可以找到用武之地。

在Mozilla和Opera浏览器中，acronym元素的显示效果是一小段虚线。在Mozilla的上下文菜单中，为此元素提供了一个**属性**选项，这样就可以将元素内的属性与属性值都呈现给网页的浏览者。

还有一个名为abbr的元素也为首字母缩写词提供了类似的服务。在HTML 4推荐标准中，有一个称为**“短语元素”**的庞大分组，这两个属性就是其中的一部分。

示例

```
<acronym title="General Agreement on Tariffs and Trade">GATT</acronym>
<acronym lang="it" title="Stati Uniti">s.u.</acronym>
```

对象模型引用方式 [window.]document.getElementById(*elementID*)

元素专有属性 无。

元素专有事件处理属性 无。

<address>

IE *all* NN *all* Moz *all* Saf *all* Op *7* HTML *all*

<address>...</address>

HTML 结束标签：必选

在HTML 4之前，address元素往往被视作一个图像格式编排标签，以便在页面上显示该页面作者的联系方式。主流浏览器均用斜体字显示address元素。但自从HTML 4以来，人们越来越关注于将实际内容与外观呈现分离开来，因此为此元素增添了更丰富的含义。由于作者信息也许会从meta元素的隐藏信息中分离出来，因此搜索引擎和未来的HTML或XML解析器也许会特别重视这个元素中的内容，以便收集作者信息。如果您既希望利用这个结构中暗含的语义，同时又想让它与其他内容一起显示，那么就须要为这个元素指定一个样式单规则以覆盖浏览器默认的格式编排方式。address元素可以包含任何标准的body元素，例如链接等。

示例

```
<address><p>Send comments to:<a href="mailto:jb@example.com">jb@example.com</a></p><address>
```

对象模型引用方式 [window.]document.getElementById(*elementID*)

元素专有属性 无。

元素专有事件处理属性 无。

<applet>

IE 3 NN 2 Moz all Saf all Op 7 HTML 3.2

```
<applet>...</applet>
```

HTML 结束标签：必选

可以将一段可执行的Java代码块以applet的形式嵌入到一个HTML文档中，一般称这个代码块为Java小程序。即使一个小程序只有一平方像素那么大，它在页面上也会占据一整块矩形区域。小程序有时须要从HTML文档中获取一些初始值。那么在小程序启动前，就可以使用一个或多个param元素向它传递参数（假设这个小程序能够接收这些参数）。这些param元素均放置在applet元素的起始和结束标签范围之内。

小程序的作者会将它们编译为一种后缀名为.class的类文件。为实现正常调用，必须将小程序的类文件同调用它的HTML文档放置在同一个目录或子目录下。而applet元素的关键属性将指引浏览器从子目录下载入特定的类文件。

所有为小程序设计的用户界面都已通过Java语言编写到小程序内容。因此，applet元素中相关属性的一个重要任务就是在页面中设置它的大小和其他相关位置特性。目前浏览器允许JavaScript脚本与小程序进行通信，也允许小程序访问文档中的其他元素。

值得注意的是，HTML 4并不赞成使用applet元素，而推荐使用用更为一般的object元素。虽然通过object元素嵌入小程序正在逐渐获得支持，但这一做法尚未形成共识。因此，在未来的一段时间内，浏览器会继续支持使用applet元素。

示例

```
<applet code="simpleClock.class" name="myClock" width="400" height="50">
<param name="bgColor" value="black">
<param name="fgColor" value="yellow">
</applet>
```

对象模型引用方式

```
[window.]document.applets[i]
[window.]document.appletName
[window.]document.getElementById(elementID)
```

元素专有属性 align、alt、archive、code、codebase、datafld、datasrc、height、hspace、mayscript、name、object、src、vspace、width

元素专有事件处理属性

处理程序	IE	其他	HTML	处理程序	IE	其他	HTML
onafterupdate	4	n/a	n/a	ondatasetcomplete	4	n/a	n/a
onbeforeupdate	4	n/a	n/a	onerrorupdate	4	n/a	n/a
ondataavailable	4	n/a	n/a	onrowenter	4	n/a	n/a
ondatasetchanged	4	n/a	n/a	onrowexit	4	n/a	n/a

align

IE 3 NN 2 Moz all Saf all Op 7 HTML 3.2

```
align="对齐常量"
```

可选

对齐属性决定了小程序所在的矩形框如何与环绕四周的其他内容进行对齐。如果想要了解不同浏览器中此属性的具体使用方式，可以参见本章前文中的“对齐常量”部分。

示例

```
<applet code="simpleClock.class" name="myClock" align="absmiddle"
width="400" height="50"></applet>
```

值 不区分大小写的对齐常量。

默认值 bottom

对象模型引用方式

```
[window.]document.applets[i].align
[window.]document.appletName.align
[window.]document.getElementById(elementID).align
```

alt

IE 3 NN 3 Moz all Saf all Op 7 HTML 3.2

```
alt="文本信息"
```

可选

如果浏览器不具备加载并运行Java小程序的能力，或浏览器在其选项中关闭了对Java的支持，那么网页将在原本应该显示applet元素的位置上显示alt的文本信息。在无法加载Java小程序的情况下，这个文本信息会报告页面访问者他错过了什么内容。与noscript和noframes元素不同的是，当无法使用Java小程序时，并没有对应的替代元素。但是在实际使用中，由于种种原因，当小程序加载失败时浏览器并不一定显示这条消息。

示例

```
<applet code="simpleClock.class" name="myClock" align="absmiddle"
alt="A Java clock applet." width="400" height="50"></applet>
```

值 使用引号括起来的任意字符串。

默认值 无。

archive

IE 7 NN 3 Moz all Saf all Op 7 HTML 4

```
archive="存档文件URL"
```

可选

可以将多个Java类文件打包放入一个未压缩的*.zip*存档文件中，以便浏览器能够一次载入全部类文件。另外，也可以采用先载入主类（由code属性指定），然后再让类加载器获取其他附加类的加载方式。但比较这两种方式，一次性载入能带来性能上的提升。而在一次性载入时，*zip*格式的存档文件地址就由archive的属性值指定。

为进一步阐明archive属性，一定要包含一个code属性，以便指出首先要载入的主类名称。要注意一点，这里使用的URL与codebase地址是相关的。

HTML规范允许使用由空格分隔的多个URL来指明其他附加类或资源文件。由于W3C一直视object为applet元素的继任者，这种设计正是为了使object元素也能使用此属性。

示例

```
<applet code="ScriptableClock.class" archive="myClock.zip" width="400" height="50"></applet>
```

值 该URI区分大小写。

默认值 无。

对象模型引用方式

```
[window.]document.applets[i].archive
[window.]document.appletName.archive
[window.]document.getElementById(elementID).archive
```

code

IE 3 NN 2 Moz all Saf all Op 7 HTML 3.2

```
code="文件名.class"
```

必选

此属性表示已启动的小程序的主类名称。如果codebase属性尚未指定，则code属性必须包含一个文件夹路径，该路径指明了加载小程序的HTML文档的存放路径。值得注意的是，不要遗漏*.class*扩展名，一定要完整地书写整个类名。另外，由于大多数服务器区分字符大小写，因此还应区分类名的大小写。

示例 `<applet code="applets/ScriptableClock.class" width="400" height="50"></applet>`

值 以*class*结尾的类名或与HTML文档相关的完整路径，注意区分大小写。

默认值 无。

对象模型引用方式

```
[window.]document.applets[i].code
[window.]document.appletName.code
[window.]document.getElementById(elementID).code
```

codebase

IE 3 NN 2 Moz all Saf all Op 7 HTML 3.2

`codebase="路径"` 可选

此属性指明了保存类文件的文件夹路径，该类文件由code或archive属性指定。要注意一点，codebase属性并不包含类文件的名称，它仅仅说明其所在路径。此时可以使用文件夹的完整URL路径作为属性值，但请勿使用超出其当前文档域的路径，否则安全方面的限制会阻止类文件的加载。另外也可以在code或object中使用全路径和文件名称，这样就可以不必使用codebase属性。

示例

```
<applet code="ScriptableClock.class" codebase="applets/" width="400" height="50"></applet>
```

值 区分大小写的路径名称，往往与保存当前HTML文档的文件夹相关。

默认值 无。

对象模型引用方式

```
[window.]document.applets[i].codeBase
[window.]document.appletName.codeBase
[window.]document.getElementById(elementID).codeBase
```

datafld, datasrc

其他信息请参见param元素，它用于IE与Java小程序之间的数据绑定。

height, width

IE 3 NN 2 Moz all Saf all Op 7 HTML 3.2

`width="像素值" height="像素值"` 必选

height和width这两个属性可以控制Java小程序在文档中所占据的空间大小。在一些浏览器版本中可以不设定这两个属性，此时就由小程序自身界面的高和宽决定它在文档中可见矩形框的大小。虽然它的显示特点有点类似于图片，但如果明确地指出这个对象的空间大小会大大提高浏览器中渲染引擎的效率。因此，正如我们为所有的图片和其他可视外部对象所做的，最好也养成为所有的applet元素设置这些空间属性值的习惯。

示例 `<applet code="ScriptableClock.class" width="400" height="50"></applet>`

值 正整数像素值。虽然不能通过设定0值来完全隐藏一个applet元素，但至少可以让它的长宽均只占据一个像素值。可以通过DHTML设置其display属性为“none”来隐藏applet元素。

默认值 无。

对象模型引用方式

```
[window.]document.applets[i].height
[window.]document.appletName.height
[window.]document.getElementById(elementID).height
[window.]document.applets[i].width
[window.]document.appletName.width
[window.]document.getElementById(elementID).width
```

hspace, vspace

IE 3 NN 2 Moz all Saf all Op 7 HTML 3.2

```
hspace="像素值" vspace="像素值"
```

可选

可以通过设置这两个属性为applet元素与周围的任何内容之间留下空白。vspace属性主要控制applet元素上下两端的空白空间大小，而hspace则控制左右两端的空间大小。对于能够领悟样式单的浏览器而言，如果有特别的需求，最好通过CSS的填充和边距等属性来控制单独一条边的空白空间大小。

示例

```
<applet code="ScriptableClock.class" width="400" height="50" hspace="3" vspace="4"></applet>
```

值 正整数像素值。

默认值 0

对象模型引用方式

```
[window.]document.applets[i].hspace
[window.]document.appletName.hspace
[window.]document.getElementById(elementID).hspace
[window.]document.applets[i].vspace
[window.]document.appletName.vspace
[window.]document.getElementById(elementID).vspace
```

id

IE 4 NN n/a Moz all Saf all Op 7 HTML 4

```
id="元素标识符"
```

可选

此属性是一个独一无二的标识符，它能够使所属元素区别于文档中的所有其他元素。可以将其属性值视为一个ID选择器，以便将一个元素与某个样式规则联系起来。另外，也可以为某元素分配一个唯一的ID，然后同时使用class属性，以便将该元素包括在一个元素组之中。详情请参见在线参考III。

如果为applet元素设置了id属性但未设置name属性，那么id的属性值也可以充当applet元素的名称，以便那些使用元素名称的脚本进行引用，如document.appletName。

示例

```
<applet id="clocker" code="ScriptableClock.class" width="400" height="50"></applet>
```

值 区分大小写的标识符。

默认值 无。

对象模型引用方式

```
[window.]document.applets[i].id
[window.]document.appletName.id
[window.]document.getElementById(elementID).id
```

mayscript

IE 4 NN 3 Moz 1.7.5 Saf all Op n/a HTML n/a

```
mayscript
```

可选

Navigator 3引入了一种称为“实时连接”的新技术，这种技术允许脚本与Java小程序之间进行双向通信（Mozilla系列浏览器并未实现该功能）。考虑到各种安全因素，小程序与脚本之间的通信功能必须由页面作者明确地开启后才能正常工作。将mayscript属性添加到applet元素的标签库后，能够利用文档对象和脚本的小程序就可以访问到这些项目。换句话说，HTML赋予这个小程序访问文档中相关脚本的能力。而这个属性就是一个简单的开关，一旦出现了这个属性的名称，就意味着打开了访问开关。

但小程序与JavaScript之间进行通信还需要另外一个步骤。即小程序的代码中必须导入一个特殊的Netscape类：*JSObject.class*。这个类和它的异常类已内置于IE 4及后续版本的Java支持环境之中。尽管在IE中这个功能执行起来并不完美，但这些小程序还是能够与脚本之间完成基本的通信。而对一些Mozilla系列浏览器的用户安装

程序而言，须要Java实时运行模块/插件正常完成安装。

示例 `<applet code="ScriptableClock.class" width="400" height="50" mayscript></applet>`

值 无须分配属性值。只要此属性名出现在元素中，就表示打开小程序和脚本之间的双向通信。

默认值 `Off`

name

IE 3 NN 2 Moz all Saf all Op 7 HTML 3.2

`name="元素标识符"` 可选

如果正在使用脚本调用一个小程序，那么为该applet元素指定一个独一无二的name属性值将有助于引用此小程序。这样，一旦编辑了页面，并移动或删除了页面上的多个小程序，也不用担心如何将索引值与数组形式的对象引用对应起来。在IE 4及后续版本或与W3C DOM兼容的浏览器中，可以忽略name属性，而在脚本中使用id的属性值来引用applet对象。

示例

```
<applet name="clock2" code="ScriptableClock.class" width="400" height="50"></applet>
```

值 区分大小写的标识符。

默认值 无。

对象模型引用方式

```
[window.]document.applets[i].name
[window.]document.appletName.name
[window.]document.getElementById(elementID).name
```

object

IE n/a NN n/a Moz n/a Saf n/a Op n/a HTML 4

`object="文件名"` 可选

此属性用于引用文件名对应的文件（与codebase的URI相关），以便在不同会话（session）之间保存applet元素的状态。如果使用得当，这个属性可以取代code属性，并且该数据文件就指向小程序的启动类文件。

示例 `<applet name="clock2" object="clockData.dat" width="400" height="50"></applet>`

值 该文件名区分大小写。

默认值 无。

对象模型引用方式

```
[window.]document.applets[i].object
[window.]document.appletName.object
[window.]document.getElementById(elementID).object
```

src

IE 4 NN n/a Moz n/a Saf n/a Op n/a HTML n/a

`src="URL"` 可选

IE浏览器将此属性作为“关联文件的”URL地址。此属性可能与HTML和Navigator规范中定义的archive属性相似，但它并不能代替code和codebase属性。

值 完整或相对URL路径。

默认值 无。

对象模型引用方式

```
[window.]document.applets[i].src
[window.]document.appletName.src
[window.]document.getElementById(elementID).aex
```

vspace

请参见hspace属性。

width

请参见height属性。

<area>

IE *all* NN *all* Moz *all* Saf *all* Op *7* HTML *3.2*

`<area>` HTML结束标签：禁用

map元素定义了一个客户端图像映射，它最后将与一个占据一定页面空间的图像或其他元素联系在一起。map元素的唯一工作就是为一个或多个area元素定义指定一个名称或标签环境。当用户与图像或其他对象的特定区域发生交互（比如点击）时，每个area元素均定义了页面应如何做出响应。

客户端图像映射区域就像一个链接，映射内的某个区域可以链接到一个URL，并且可以指定新文档加载时的目标框架或窗口。事实上，在最初的脚本文档对象模型中，正是像引用链接一样引用area元素。虽然并不会在导航条中使用客户端区域映射来占据一个细长的框架或框架集，但这种方法能让艺术家们创造性地设计出一种菜单样式，同时也可以让网页制作者将一副大图片的任意部分转化为特殊的链接。

示例

```
<map name="nav">
<area coords="20,30,120,70" href="contents.html" target="display">
<area coords="20,80,145,190" href="contact.html" target="display">
</map>
```

对象模型引用方式

```
[window.]document.links[i]
[window.]document.getElementById(elementID)
```

元素专有属性 alt、coords、href、nohref、shape、target

元素专有事件处理属性 无。

alt

IE *3* NN *n/a* Moz *all* Saf *all* Op *7* HTML *3.2*

`alt="文本信息"` 必选

非图形化的浏览器能借助alt属性显示图像热点的扼要说明。请记住，在手持电脑中往往有非图形化的浏览器，有时人们为了改善性能也会关闭普通浏览器的图像功能。另外，也别忽视那些视力受损的使用者。

示例

```
<area coords="20,30,120,70" href="contents.html" target="display" alt="Table of Contents">
```

值 使用引号括起来的任意字符串。

默认值 无。

对象模型引用方式 `[window.]document.getElementById(elementID).alt`

coords

IE *all* NN *all* Moz *all* Saf *all* Op *7* HTML *3.2*

`coords="坐标 1, ... 坐标 N"` 可选

尽管从W3C定义来看，area元素并未强制要求使用coords属性，但这并不意味着可以忽略这个属性，更不能期望在缺少这个属性的情况下area元素会表现得一如往常。coords属性用于定义区域轮廓，使得它们能

够与特殊的链接或脚本动作联系起来。通过一些第三方的开发工具的辅助，可以确定热点区域的坐标。还可以在图形程序中打开图像，通过程序实时显示光标的位置，然后将它们转换成coords的属性值。

这些坐标值是一个由逗号分隔的数值列表。如果两个区域重叠在一起，则优先使用在HTML代码中首先定义的区域。

示例 `<area coords="20,30,120,70" href="contents.html" target="display">`

值 每个坐标值都是一个长度值，但坐标的数值与它们的顺序都依赖于shape属性指定的形状，而这个形状与area元素有关。如果shape="rect"，则有四个坐标（左，上，右，下）；如果shape="circle"，则有3个坐标（圆心-x，圆心-y，半径）；如果shape="poly"，则有一组坐标值对，每对代表多边形的一个顶点（x1，y1，x2，y2，x3，y3，...xN，yN）。

默认值 无。

对象模型引用方式 `[window.]document.getElementById(elementID).coords`

href

IE *all* NN *all* Moz *all* Saf *all* Op *7* HTML *3.2*

`href=" URI"` 必选

此属性指定了与特定区域相关联的链接目标URI地址。当此URI地址是一个HTML文档时，该文档会被载入当前浏览器窗口（默认情况）或其他目标窗口（由target属性决定）。如果是其他文件格式，浏览器可能将目标内容载入插件或将目标文件保存在客户端所在的机器上。由于一些早期的浏览器并不区分area元素和a元素，因此在area中必须定义href属性，只有这样，早期的DOM才能读取URL相关的属性并使事件处理程序能够正常工作。

示例 `<area coords="20,30,120,70" href="contents.html" target="display">`

值 可以是任何有效的URI地址，包括完整或相对URL、同一页面上的锚链（锚链名以“#”号作为前导）、非标准的JavaScript伪URL（在支持脚本的浏览器中，它会触发一段脚本指令而不是直接导航到某个目的地）。

默认值 无。

对象模型引用方式

```
[window.]document.links[i].href
[window.]document.getElementById(elementID).h ef
```

其他所有涉及URL构成元素（如协议和主机名等）的链接对象属性。请参见第2章中的link对象说明。

nohref

IE *all* NN *all* Moz *all* Saf *all* Op *7* HTML *3.2*

`nohref` 可选

此属性会通知浏览器，由这些坐标定义的区域并未与任何链接相关联，即不包含任何href属性。一旦包含了此属性，则支持脚本的浏览器就不会将此area元素视作一个链接。于此同时，由于area元素缺少href属性，该元素也不会再响应任何用户事件。在IE 4和后续版本的浏览器及其他W3C DOM浏览器中，可以在脚本中设置相应的noHref属性为“true”或“false”来达到开启或关闭此属性的目的。

示例 `<area coords="20,30,120,70" nohref>`

值 如果此属性已出现，就意味着其属性值为true。但为了兼容XHTML，请使用nohref="nohref"的赋值形式。

默认值 Off

对象模型引用方式 `[window.]document.getElementById(elementID).noHref`

shape

IE *all* NN *all* Moz *all* Saf *all* Op *7* HTML *3.2*

shape="*外形名称*" 可选

此属性定义了客户端图像映射区域的外形，具体的坐标由coords属性指定。shape属性提示浏览器坐标点的应有数量。

示例

```
<area shape="poly" coords="20,20,20,70,65,45" href="contents.html" target="display">
```

值

外形名称不区分大小写。不同的浏览器定义了不同的外形名称，但circle、rect、poly和polygon这4个名称适用于各种浏览器。

外形名称	IE	其他	HTML	外形名称	IE	其他	HTML
circ	·	—	·	polygon	·	·	—
circle	·	·	·	rect	·	·	·
poly	·	·	·	rectangle	·	—	—

默认值 rect

对象模型引用方式 [window.]document.getElementById(*elementID*).shape

target

IE *all* NN *all* Moz *all* Saf *all* Op *7* HTML *3.2*

target="*窗口或框架名称*" 可选

如果并不准备在当前窗口或框架中载入目标文档，就可以通过设置target属性，使得文档在属性值对应的窗口或框架中载入。此时，框架或窗口名称必须在对应目标中指定为标识符。例如，可以将frame元素的name或id属性设置为该名称，或通过脚本方法window.open()的第二个参数在创建新窗口时为窗口命名。如果疏忽了这个属性，那么目标文档将会取代链接所在的文档。只有将href属性赋值给元素之后，此属性方能发挥作用。

由于一个area元素只能指定一个目标文档和一个载入目的地，因此如果期望一个链接能够改变多个框架的内容，不妨采用area元素的onclick事件处理程序来激发一个脚本以加载多个文档（请参考第2章以确定可用的浏览器版本）。此时要将每个框架的location.href属性值设定为期望的URL值。

由于框架和窗口已经超出了单纯的文档标记范畴，因此HTML 4和XHTML规范中的严格DTD并不支持在任何元素中使用target属性。事实上，在严格的环境下框架根本无法设置文档内容，这正是将DTD从HTML 4和XHTML中剥离出来的目的。如果文档必须有这些严格的DTD，并且又期望使用target属性，不妨在页面载入后通过脚本设置链接、图片映射表和表单的属性。

示例

```
<area coords="20,30,120,70" href="contents.html" target="display">
<area coords="140,30,180,70" href="index.html" target="_top">
```

值

如果已通过目标元素的name属性指定了框架或窗口名称，就可以使用这个区分大小写的标识符。可使用下面几个保留的目标名称常量之一。

_blank

浏览器为目标文档打开一个新窗口。

_content

此属性值仅能在Mozilla系列浏览器中使用。当链接出现在浏览器的侧边栏中时，使用此属性值后无论主

内容框架名称是什么，目标文档均会在主内容框架中显示。

_parent

目标文档会取代当前框架的框架集文档，如果不存在，则依照“_self”执行。

_self

目标文档替代在链接所在窗口或框架里的现有文档。

_top

目标文档将会取代任何已加载的框架集窗口，从而占据整个浏览器。如果在窗口中不存在框架集，则依照“_self”执行。

默认值 _self

对象模型引用方式

```
[window.]document.links[i].target
[window.]document.getElementById(elementID).target
```


IE *all* NN *all* Moz *all* Saf *all* Op *7* HTML *all*

...

HTML 结束标签：必选

b元素是HTML 4数个字体样式元素之一。它会以粗体的形式显示其包含的文本。可以嵌套使用这些字体样式元素以产生复合效果，例如，通过“bold-italic text”可以产生粗斜体。

粗体字体的具体显示效果则取决于浏览器，例如，字符大小的计算结果和当前载入的字体类型。因此，可能无法根据系统字体来推断浏览器的粗体显示效果。如果期望完全控制字体，使它按预期完美地呈现出来，最好使用样式单（如果目标浏览器支持，也可以使用可下载的字体）来指定一个真正的粗体。在字体支持的情况下，CSS的font-weight样式属性可以对文本的粗体效果提供很细致的控制。

最好的方法是，利用此元素的容器，通过样式单规则指定页面内部分甚至全部b元素的样式。例如，如果期望所有的b元素都呈现为红色，就可以将样式规则设置为“b {color:red}”。这种方式只需要很少的代码就可以控制所有的b元素。

因此，尽管HTML 4或XHTML 1.0并未抛弃b元素，但样式单并不推荐使用这个元素。

示例 This product is new and improved!

对象模型引用方式 [window.]document.getElementById(*elementID*)

元素专有属性 无。

元素专有事件处理属性 无。

<base>

IE *all* NN *all* Moz *all* Saf *all* Op *7* HTML *all*

<base>

HTML 结束标签：禁用

base元素定义在文档的head元素内部，用于浏览器指定当前文档中的URL路径。对于那些用于指定src和href属性的相对URL而言，此路径是它们的基准路径。base元素的指定路径应该是一个包含文档名称的完整路径（尽管浏览器趋向于支持文件夹路径）。浏览器可以根据这个路径计算出该文档所在文件夹的基准URL路径。如果将base元素设置为<base href="http://www.example.com/products/index.html">，那么在这个页面上，一个指向“*widgets/framitz801.html*”的链接就可以得到这样的一个全路径：“*http://www.example.com/products/widgets/framitz801.html*”。同样，使用“..”也可以表示上级目录的路径。例如，根据前文中base

元素的定义，在“*index.html*”页面中的一个img元素就可以设置为‘src="../images/logo.jpg"’。该地址等同于“*http://www.example.com/images/logo.jpg*”。

然而，当今的许多浏览器已经可以自动计算页面的基准路径，这样在未指定base元素的情况下也可以使用相对URL。这样一来，在本地完成页面开发后，部署页面时就不须要再更改页面中的base元素。HTML 4规范指出，如果一个文档缺少base元素，那么就会默认将该文档的URL作为基准URL。

当然，也可以使用base元素为页面中任意的链接形式的元素指定一个默认目标。这样一来，如果所有的链接都会在另一个框架中加载文档，那么一旦在base标签中指定了对应的目标框架，就不必在文档中反复使用target属性。如果想用一个单独的链接替换默认链接，也可以同时为base元素指定一个target属性（请参考target属性的相关说明）。

base元素中只有id属性能与其他元素属性通用。

示例 `<head> <base href="http://www.example.com/index.html" target="_top"> </head>`

对象模型引用方式 `[window.]document.getElementById(elementID)`

元素专有属性 href、target

元素专有事件处理属性 无。

href

IE *all* NN *all* Moz *all* Saf *all* Op *7* HTML *all*

`href="URL"`

可选

href属性是一个文档页面的URL路径，该URL的服务器路径将作为当前页面中所有相对路径的基准路径。一般来说，其属性值就是当前文档的URL路径，但为了便于组织文档和文件夹，它也可以使用其他路径。

示例 `<base href="http://www.example.com/products/index.html">`

值 必须是一个文档的完整的绝对URL路径。

默认值 当前文档的路径名。

对象模型引用方式

```
[window.]document.getElementsByTagName("base")[0].href
[window.]document.getElementById(elementID).href
```

target

IE *all* NN *all* Moz *all* Saf *all* Op *7* HTML *4*

`target="窗口或框架名称"`

可选

如果一个页面上的绝大多数链接和区域映射都会在另一个窗口或框架中加载，那么就可以设置base元素中的target属性，从而更方便地管理这些元素的目标位置。当然，也可以仅仅设置target而不设置href属性，这样就只会设置基准目标引用方式。

由于框架和窗口已经超出了单纯的文档标记范畴，因此HTML 4和XHTML规范中的严格DTD并不支持在任何元素中使用target属性。事实上，在严格的环境下，框架根本无法设置文档内容，这正是将DTD从HTML 4和XHTML中剥离出来的目的。如果文档必须支持这些严格的DTD，并且又期望使用target属性，不妨在页面载入后通过脚本设置链接、图片映射表和表单的属性。

示例 `<base target="rightFrame">`

值

如果已通过目标元素的name属性指定了框架或窗口名称，就可以使用这个区分大小写的标识符。可以使用下列已保留的目标名称常量之一。

_blank
浏览器为目标文档打开一个新窗口。

_content
此属性值仅能在Mozilla系列浏览器中使用。当链接出现在浏览器的侧边栏中时，使用此属性值后无论主内容框架名称是什么，目标文档均会在主内容框架中显示。

_parent
目标文档会取代当前框架的框架集文档，如果不存在，则依照“_self”执行。

_self
目标文档替代在链接所在窗口或框架里的现有文档。

_top
目标文档将会取代任何已加载的框架集窗口，从而占据整个浏览器。如果在窗口中不存在框架集，则依照“_self”执行。

默认值 _self

对象模型引用方式

```
[window.]document.getElementsByTagName("base")[0].target
[window.]document.getElementById(elementID).target
```


<basefont>

IE *all* NN *all* Moz *all* Saf *all* Op *7* HTML *3.2*

```
<basefont>
```

HTML结束标签：禁用

basefont元素用于向浏览器提示一些字体信息，当前页面内该元素之后所有文本都会根据这个字体信息进行渲染。可以在文档的head或body部分应用这个元素（微软的IE 4及后续版本建议仅在body部分使用），另外，每当需要为文档的某个部分设置基准字体时，也可以插入basefont元素。值得注意的是，basefont元素不一定适用于表格中的内容。因此，如果期望表格中的内容与basefont设置的内容两者风格一致，那么最好单独为表格元素设置字体样式。

basefont元素替换了浏览器中用户选项中定义的默认字体设置。与大多数字体相关的元素一样，HTML 4也反对使用basefont元素，并建议使用样式单，在HTML 4和XHTML的严格DTD中甚至取消了该元素。

示例 `<basefont face="Times, serif" size="4">`

元素专有属性 color、face、name、size

元素专有事件处理属性 无。

color

IE *3* NN *n/a* Moz *all* Saf *all* Op *7* HTML *4*

```
color="颜色三元组或颜色名"
```

可选

此属性为所有basefont元素以后的文本设置字体颜色。HTML 4反对使用此属性，建议使用CSS的color属性。

示例 `<basefont color="Olive">`

值 一个十六进制的三元组或明语颜色名称。目前可用的明语颜色名称请参见附录A。

默认值 由浏览器决定默认值。

对象模型引用方式 `[window.]document.getElementsByTagName("basefont")[0].color`

face

IE *4* NN *n/a* Moz *all* Saf *all* Op *7* HTML *4*

```
face=" 字体名称 1[, ... 字体名称 N]"
```

可选

对于那些受basefont元素约束的部分，可以使用此属性为它们设置一组字体作为默认字体。这一组字体名称

使用逗号进行分隔，浏览器从它们中的第一个开始，逐个与客户端系统中的字体进行比对，以设置默认字体，如果最终仍然找不到对应字体就使用浏览器的默认字体。比对时，face中的字体名称必须与系统中的字体名称完全匹配。如果放弃使用更佳的样式单属性而使用这个属性，那么建议您将通用字体（如serif、sans-serif等）列在最后。HTML 4反对使用此属性，建议使用CSS的font-family属性。

在IE 3中，此属性被改为name属性。

示例 `<basefont face="Bookman, Times Roman, serif">`

值 一个或多个字体名称，请在其中包含可识别的通用字体，如serif | sans-serif | cursive| fantasy | monospace。

默认值 浏览器的默认值。

对象模型引用方式 `[window.]document.getElementsByTagName("basefont")[0].face`

name

IE |3| NN *n/a* Moz *n/a* Saf *n/a* Op *n/a* HTML *n/a*

`name="字体名称"` 可选

这是face属性在IE 3中的描述形式。它仅接受一个单独的字体名称作为属性值。目前已不再使用这个属性。

值 一个单独的字体名称。

默认值 浏览器的默认值。

size

IE *all* NN *n/a* Moz *all* Saf *all* Op *7* HTML *3.2*

`size="字体相对大小"` 可选

size属性中引用的字体大小均为相对大小，它未同任何操作系统平台中的字体大小相关联。size属性值的可选范围是从1—7的整数值，而浏览器的默认字体大小是3。但具体的大小随着操作系统和浏览器的不同而发生变化。

用户可以在选项里调整默认的字体大小。但使用size属性后，它会取代默认设置。此外，size的属性值还与选项中的具体字体大小设置有关。通过在属性值前加上“+”或“-”，可以使得浏览器的默认字体大小在1—7这个范围内增大或减小。

示例

```
<basefont size="4">
<basefont size="+3">
```

值 任意一个由“+”、“-”和整数组成相对数值。

默认值 3

对象模型引用方式 `[window.]document.getElementsByTagName("basefont")[0].size`

<bdo>

IE *5* NN *n/a* Moz *all* Saf *all* Op *7* HTML *4*

`<bdo>...</bdo>` HTML 结束标签：必选

bdo元素的名称来源于“*bidirectional override*”。大多数元素中的lang和dir属性都是用于解决书写系统所带来的混乱问题，这些书写系统往往由不同书写方向的文本所组成，使得它们看起来杂乱不堪。由于文本处理过程中使用了不同的转换方式，必须明确地关闭双向算法，而bdo元素正适用于这种情况（译注10）。

示例 `<bdo dir="ltr">someMixedScriptTextHere</bdo>`

译注 10：例如，如果电子邮件系统格式化了一段包含英语和希伯来语的文本，则双向算法会不正确地转换这段文字。由于希伯来语文本的阅读顺序已经被电子邮件格式转换过了，但双向算法会再一次对该文本进行转换。

元素专有属性 无。
元素专有事件处理属性 无。

<bgsound>

IE *3* NN *n/a* Moz *n/a* Saf *n/a* Op *n/a* HTML *n/a*

`<bgsound>` HTML结束标签：可选

此元素指定一个声音文件，以便在用户浏览该网页时能够在背景中播放，它仅适用于IE浏览器。而且本元素只允许出现在head元素中。通过脚本，可以在加载声音文件后控制音量和重播次数。尽管我们建议使用一个结束标签，但由于在所有的标准规范中，都由起始标签中的属性维护声音，故而此处不再须要使用结束标签。此元素中仅有id属性能够与其他元素通用。

如果要使用这个标签，那么我强烈建议能够让用户做出自己的选择，并且将默认选择设置为关闭播放。设想一下，在办公环境里，突然从电脑中传出一阵背景音乐是一件多么令人吃惊且尴尬的事情啊。同时还要注意，由于下载需要时间，因此音乐开始播放的时刻可能会有所延迟。

示例 `<bgsound src="tunes/mazeppa.mid">`
对象模型引用方式 `[window.]document.getElementById(elementID)`
元素专有属性 `balance`、`loop`、`src`、`volume`
元素专有事件处理属性 无。

balance

IE *4* NN *n/a* Moz *n/a* Saf *n/a* Op *n/a* HTML *n/a*

`balance="带符号整数"` 可选

此属性决定了左右两个声道的声音输出。一旦对此属性赋值，则不能通过脚本进行修改。

示例 `<bgsound src="tunes/mazeppa.mid" balance="+2500">`
值 介于-10 000和+10 000之间的带符号整数。如果取值为0，则代表两边的声音输出达到平衡。负值使得左边声音增大，而正值则使右边声音增大。
默认值 0
对象模型引用方式 `[window.]document.getElementsByTagName("bgsound")[0].balance`

loop

IE *3* NN *n/a* Moz *n/a* Saf *n/a* Op *n/a* HTML *n/a*

`loop="整数值"` 可选

此属性值决定了声音播放的次数。如果缺少该属性，或其值不等于“-1”，则声音将至少播放一次。如果值等于“-1”，就意味着该音乐会一直播放，直到页面被卸载。与微软提供的IE SDK文档不同的是，在声音开始播放前似乎无法预先缓存声音文件。

示例 `<bgsound src="tunes/mazeppa.mid" loop="3">`
值 仅播放一次时不需要设置，即使设置为0也会播放一次。大于0的数值表示声音的播放次数，如果设置为-1则表示无限地循环播放。
默认值 -1
对象模型引用方式 `[window.] document.getElementsByTagName("bgsound")[0].loop`

src

IE *3* NN *n/a* Moz *n/a* Saf *n/a* Op *n/a* HTML *n/a*

`src="URL"` 可选

此属性指出将要播放的声音文件的URL地址。可以被播放的声音文件类型只受浏览器音频功能的限制。如，

IE不需要任何额外的插件就可以支持常用的音频格式，包括MIDI等。

示例 `<bgsound src="tunes/beethoven.mid">`

值 任何有效的完整或者相对URL地址，但IE浏览器或插件必须支持该文件格式。

默认值 无。

对象模型引用方式 `[window.] document.getElementsByTagName("bgsound")[0].src`

volume

IE *4* NN *n/a* Moz *n/a* Saf *n/a* Op *n/a* HTML *n/a*

46

`volume="有符号的整数"` 可选

该整数决定了背景声音的播放时的相对音量，具体的声音大小与客户计算机中用户选择的最大声音输出等级有关。此处最大的音量值为0，但即使设置为0，播放出的声音也只能和**声音**控制面板里设置的音量一样大。而它的最小可以调节到−10 000，但大多数用户设置的音量均高于此值。

示例 `<bgsound src="tunes/beethoven.mid" volume="-500">`

值 介于−10 000和0之间的带符号整数。

默认值 0

对象模型引用方式 `[window.] document.getElementsByTagName("bgsound")[0].volume`

<big>

IE *all* NN *all* Moz *all* Saf *all* Op *7* HTML *3.2*

`<big>...</big>` HTML结束标签：必选

big元素是HTML 4的数个字体样式元素中的一个，它使得其中的内容在显示时比前文的字体尺寸大一级，当然字体尺寸依然还是限制在HTML规定的1−7号之内。如果嵌套使用big元素，相关的效果会进行叠加，每增加一层嵌套就会将字体增大一级。而默认的字体大小则取决于浏览器、操作系统和用户设置。建议使用样式单规则，以便更精确地控制字体大小的显示。

示例 `<p>This product is <big>new</big> and <big>improved</big>!</p>`

对象模型引用方式 `[window.]document.getElementById(elementID)`

元素专有属性 无。

元素专有事件处理属性 无。

<blink>

IE *n/a* NN *all* Moz *1.01* Saf *n/a* Op *7* HTML *n/a*

`<blink>...</blink>` HTML结束标签：必选

Marc Andreessen先生（译注11）的这一杰作给网页带来了巨大的灾难。它能使元素中的所有内容以一种不受控制的狂乱方式不停地闪烁。元素中的内容越多，就越难在闪烁的间隙中看清它们。求您就别用这玩意了。

示例 `<blink>I dare you to read this...and not look at it.</blink>`

<blockquote>

IE *all* NN *all* Moz *all* Saf *all* Op *7* HTML *all*

47

`<blockquote>...</blockquote>` HTML结束标签：必选

blockquote元素主要用于在文档中突出一长段引用语。一直以来，blockquote元素的显示效果都是一个缩进的区块，这个区块的左、右边距都约为40像素，其上部和下部也留出了额外的空白空间。尽管目前更鼓励使用样式单创建这样的显示效果（无论是否使用blockquote元素均可），但浏览器很有可能继续使用这样的呈现方式。如果在一行内使用短引用，请参见q元素。

译注11：Mosaic发明人，网景公司的合伙创始人。

示例

```
<blockquote>Four score and seven years ago...shall not perish from the earth</blockquote>
```

对象模型引用方式 [window.]document.getElementById(*elementID*)

元素专有属性 cite

元素专有事件处理属性 无。

cite

IE *6* NN *n/a* Moz *all* Saf *all* Op *7* HTML *4*

```
cite="URL"
```

可选

此属性指出一个在线原文档的URL地址，该引用语就出自此文档。要注意，设置此属性根本无法从另一个文档中复制或摘录任何内容。至于这个属性存在的原因，也许是因为HTML 4推荐标准希望鼓励未来的浏览器和搜索引擎能够利用这些在线资源，为读者和上网者提供便利。通过在Mozilla的上下文菜单中设置这个元素的特性，可以为此属性指定的URL显示一个包含链接的小窗口。目前的主流浏览器尚未为该属性提供更多的功能。

值 一个指向网络上某文档的有效URL链接，既可使用相对URL，也可以使用绝对URL地址。

默认值 无。

元素专有事件处理属性 无。

<body>

IE *all* NN *all* Moz *all* Saf *all* Op *7* HTML *all*

```
<body>...</body>
```

HTML结束标签：可选

在一个HTML页面中，所有的序言在head部分内陈述完成之后，就由body元素来包含页面内的正式内容，用户可以通过浏览器窗口看到它们。当然，有些浏览器还可以将这些内容读给用户听。在样式单出现以前，页面作者一直都在body元素中设置整个文档的颜色和背景图。在HTML 4中，很多有用的属性都不再被推荐使用，它们将被相应的样式单规则所替代。尽管如此，主流浏览器依然会在未来的很多年中继续支持这些属性。

在body元素内还可以使用window元素的许多对象事件处理属性。例如，绝大多数文档中均已定义的对象事件模型有一个名为onload的事件处理程序，当前窗口或框架完成对文档的加载后，就会自动激发onload。正是在body元素中将这个事件处理程序指定为一个元素属性。

尽管从很多暗示来看，body元素就是document对象，但事实并非如此。document对象有很多附加的属性，如document.title，这些属性都定义在HTML文档中，而不是来自于body元素。在一个遵循W3C DOM的浏览器中，文档节点树更加明显地指出了根结点document和body元素之间的差别：document节点是html元素的父节点，而html元素则是head和body元素共同的父节点。

示例

```
<body background="watermark.jpg" onload="init( );">...</body>
```

对象模型引用方式 [window.]document.body

元素专有属性 alink、background、bgcolor、bgproperties、bottommargin、leftmargin、link、marginheight、marginwidth、nowrap、rightmargin、scroll、text、topmargin、vlink

元素专有事件处理属性

处理程序	IE	NN	Mozilla	Safari	Opera	HTML
onafterprint	5	n/a	n/a	n/a	n/a	n/a
onbeforeprint	5	n/a	n/a	n/a	n/a	n/a

续表

处理程序	IE	NN	Mozilla	Safari	Opera	HTML
onload	3	2	m18	1	7	4
onresize	4	4	m18	1	7	n/a
onscroll	4	n/a	1.0.1	1	n/a	n/a
onselect	4	n/a	n/a	n/a	n/a	n/a
onunload	3	2	m18	1	7	4

alink

IE *all* NN *all* Moz *all* Saf *all* Op *7* HTML *3.2*

49

```
alink="颜色三元组或颜色名"
```

可选

指定一个超链接被用户激活（被点击）之后的颜色。链接存在三种状态：未访问（unvisited）、激活（active）和已访问（visited），这个状态是其中之一。设置这个属性后，作为链接文字、图像的边框或嵌入a元素之中的对象均会显示这个颜色。目前已不再赞成使用这个属性，请使用CSS的:active伪类来指定一个a元素的样式规则，具体信息请参见第4章。

示例 `<body alink="#FF0000">...</body>`

值 一个十六进制的三元组或明语颜色名称。目前可用的明语颜色名称请参见附录A。

默认值 `#FF0000`，此颜色最具代表性。

对象模型引用方式

```
[window.]document.alinkColor
[window.]document.body.aLink
```

background

IE *all* NN *all* Moz *all* Saf *all* Op *7* HTML *3.2*

```
background="URL"
```

可选

此属性指定了一个图像文件，它将作为页面文本和其他内容的背景。与浏览器载入的其他普通图像不同的是，背景图会以未经缩放的原始大小进行加载，然后贴附在浏览器窗口或框架中，并充满整个可用的页面空间。小图片虽然下载迅速，但总是会很明显地在背景中重复显示。当然也可以使用GIF动画作为背景图，但这实在是一件令读者抓狂的做法。在选择背景图时，一定要选择相对柔和的图片，以便使主体内容更加引人注目。另外，如果一定要使用背景图，还要保证它足够淡雅，或仅仅占据主体内容之外的空闲区域。

HTML 4并不赞成使用此属性，而建议使用CSS的backgroud属性。

示例 `<body background="watermark.jpg">...</body>`

值 任何有效的图像文件URL地址，既可使用相对URL，也可以使用绝对URL地址。

默认值 无。

对象模型引用方式 `[window.]document.body.background`

bgcolor

IE *all* NN *all* Moz *all* Saf *all* Op *7* HTML *3.2*

```
bgcolor="颜色三元组或颜色名"
```

可选

此属性为整个文档设置填充色，此颜色位于文本和其他内容的图层之下。如果同时使用bgcolor和background属性，那么背景色会透过背景图的任何透明部分。HTML 4并不赞成使用此属性，而建议使用CSS的backgroud-color属性。

示例 `<body bgcolor="tan">...</body>`

50

值 一个十六进制的三元组或明语颜色名称。不设置属性值就默认为黑色（`#000000`，black）。目前可用的明语颜色名称请参见附录A。

默认值 由浏览器、浏览器版本和操作系统共同决定。

对象模型引用方式

```
[window.]document.bgColor
[window.]document.body.bgColor
```

bgproperties

IE 3 NN n/a Moz n/a Saf all Op n/a HTML n/a

`bgproperties="属性"` 可选

此属性只能在IE浏览器中使用，它决定了在页面滚动时，背景图片（由background属性或样式单设置）是随之滚动还是固定不动。这个属性的效果能够让用户感到既奇怪又有趣。将背景图保持在一个固定位置后，此时的屏幕显示非常有趣，滚动的内容滑过背景图时就像电影的致谢在背景图上滚动显示。

示例 `<body background="watermark.jpg" bgproperties="fixed">...</body>`

值 如果设置为“fixed”，则图像不会滚动。忽略此属性或使用空的属性值（""）可让图像与内容同步滚动。

默认值 无。

对象模型引用方式 `[window.]document.body.bgProperties`

bottommargin

IE 4 NN n/a Moz n/a Saf n/a Op n/a HTML n/a

`bottommargin="整数值"` 可选

此属性用于确定文档内容的最底端与可滚动页面底部之间空白空间的大小。当页面内容的长度或窗口大小使得窗口并不需要滚动时，设置这个属性并不会带来任何可视效果。使用默认值时，内容的底部会充满页面的底部，但在Macintosh版本的IE浏览器中，即使将此属性值设置为0，两者之间也存在约10像素宽的空白。一般说来，此值越大效果越明显。此属性有点类似于为body元素设置样式单margin-bottom属性的简化方法。

示例 `<body bottommargin="20">...</body>`

值 页面底部所需空白区域高度的像素值。属性值为空字符串时，等价于0。

默认值 0

对象模型引用方式 `[window.]document.body.bottomMargin`

leftmargin

IE 3 NN n/a Moz n/a Saf all Op 7 HTML n/a

`leftmargin="整数值"` 可选

此属性用于确定文档内容的左边缘与窗口中文档区域的左边缘之间的空白空间大小。此属性有点类似于为body元素设置样式单margin-left属性的简化方法。由于body元素是文档中可渲染元素的最外层父容器，因此该属性会为文档中所有的嵌套元素设置固定的左侧页空白。

示例 `<body leftmargin="25">...</body>`

值 页面左边缘所需空白区域的像素值。属性值为空字符串时，等价于0。

默认值 10（IE/Windows）；8（其他）。

对象模型引用方式 `[window.]document.body.leftMargin`

link

IE all NN all Moz all Saf all Op 7 HTML 3.2

`link="颜色三元组或颜色名"` 可选

此元素设定一个超文本链接尚未被访问时的颜色，此时该链接的URL地址尚不在浏览器的缓存之中。链接存

在3种状态：未访问（unvisited）、激活（active）和已访问（visited），这个状态是其中之一。设置这个属性后，作为链接文字、图像的边框或嵌入a元素之中的对象均会应用这个颜色。目前已不再赞成使用这个属性，请使用CSS的:link伪类来指定一个a元素的样式规则，具体信息请参见第4章。

示例 `<body link="#00FF00">...</body>`

值 一个十六进制的三元组或明语颜色名称。目前可用的明语颜色名称请参见附录A。

默认值 `#0000FF`

对象模型引用方式

```
[window.]document.linkColor
[window.]document.body.link
```

marginheight,marginwidth

IE *n/a* NN *4* Moz *n/a* Saf *all* Op *7* HTML *n/a*

```
marginheight="整数值", marginwidth="整数值"
```

可选

此属性类似于使用CSS样式单设置正文页空白的简化方法。指定marginheight为某像素值后，会为正文内容设定其顶部和底部的页空白，同理，marginwidth值则会设置左右两侧的页空白。

示例 `<body marginheight="20" marginwidth="10">...</body>`

值 每个值分别设置了两个方向上所需空白空间的像素值。属性值为空字符串时，等价于0。

默认值 `0`

nowrap

IE *4* NN *n/a* Moz *n/a* Saf *n/a* Op *n/a* HTML *n/a*

```
nowrap
```

可选

此属性控制过长的文本内容是否应该在正文宽度以内进行换行。

示例 `<body nowrap>...</body>`

值 如果此属性已出现，就意味着其属性值为true。

默认值 `false`

对象模型引用方式 `[window.]document.body.noWrap`

rightmargin

IE *4* NN *n/a* Moz *n/a* Saf *all* Op *7* HTML *n/a*

```
rightmargin="整数值"
```

可选

此属性用于确定文档内容的右边缘与窗口中文档区域的右边缘之间的空白空间大小。此属性有点类似于为body元素设置样式单margin-right属性的简化方法。由于body元素是文档中可渲染元素的最为外层父容器，因此该属性会为文档中所有的嵌套元素均设置固定的右侧空白。须要注意的是，IE/Mac并不会像Windows版本一样让其内容贴近窗口的右边缘。

示例 `<body rightmargin="25">...</body>`

值 页面右边缘所需空白区域的像素值。属性值为空字符串等价于边距为0。

默认值 `10`（Windows）；`0`（Macintosh）。

对象模型引用方式 `[window.]document.body.rightMargin`

scroll

IE *4* NN *n/a* Moz *n/a* Saf *n/a* Op *n/a* HTML *n/a*

```
scroll="功能开关"
```

可选

当内容空间超过当前页面大小时，本属性可以控制是否出现滚动条。如果没有滚动条，但又想让用户移动页

面，那么就必须提供一些脚本方法用以滚动窗口。值得注意的是，当文档过大时，无论是否将scroll属性设置为“no”，Mac系统下的IE浏览器始终会显示滚动条。而更为时髦的控制方法则是使用CSS中的overflow属性（IE中应使用overflowX和overflowY属性）。

示例 `<body scroll="no">...</body>`

值 yes或no，不区分大小写

默认值 yes

对象模型引用方式 `[window.]document.body.scroll`

text

IE *all* NN *all* Moz *all* Saf *all* Op *7* HTML *3.2*

`text="颜色三元组或颜色名"` 可选

此属性用于确定文档中正文内容的字符颜色。文档内为每个元素单独设置的颜色值可以替代文档中的全局设定值。默认的背景颜色由于受浏览器种类、版本和操作系统的影响而变化繁多，因此使用bgcolor属性或等价的样式单规则为文档中的文本设置颜色是一个非常明智的做法。目前已不赞成使用此属性，推荐使用样式单的color属性。

示例 `<body bgcolor="#FFFFFF" text="#c0c0c0">...</body>`

值 一个十六进制的三元组或明语颜色名称。目前可用的明语颜色名称请参见附录A。

默认值 #000000或black

对象模型引用方式

```
[window.]document.fgColor
[window.]document.body.text
```

topmargin

IE *3* NN *n/a* Moz *n/a* Saf *all* Op *7* HTML *n/a*

`topmargin="整数值"` 可选

此属性用于确定文档内容的上边缘与窗口中文档区域的上边缘之间的空白空间大小。此属性有点类似于为body元素设置样式单margin-top属性的简化方法。由于body元素是文档中可渲染元素的最为外层父容器，因此该属性会为文档中所有的嵌套元素均设置固定的上边距。将topmargin设置为0或空字符串（""）时，会将正文内容推至文档内容区域的最上端。

示例 `<body topmargin="0">...</body>`

值 页面顶部所需空白区域的像素值。属性值为空字符串时，等价于边距为0。

默认值 15（IE/Windows）；8（其他）。

对象模型引用方式 `[window.]document.body.topMargin`

vlink

IE *all* NN *all* Moz *all* Saf *all* Op *7* HTML *3.2*

`vlink="颜色三元组或颜色名"` 可选

本属性指定超文本链接被用户访问后的颜色，此时目标页面已存在于浏览器的缓存空间中。链接存在三种状态：未访问（unvisited）、激活（active）和已访问（visited），这个状态是其中之一。设置这个属性后，作为链接文字、图像的边框或嵌入a元素之中的对象均会应用这个颜色。目前已不再赞成使用这个属性，请使用CSS的:visited伪类来指定一个a元素的样式规则，具体信息请参见第4章。

示例 `<body vlink="teal">...</body>`

值 一个十六进制的三元组或明语颜色名称。目前可用的明语颜色名称请参见附录A。

默认值 由浏览器和操作系统共同决定。

对象模型引用方式

```
[window.]document.vlinkColor
[window.]document.body.vLink
```


IE *all* NN *all* Moz *all* Saf *all* Op *7* HTML *all*

```
<br>
```

HTML 结束标签：禁用

无论br元素的标签出现在哪里，该处都会被强制进行换行显示（由一个回车和换行组成）。虽然浏览器将br元素视为一个真正的换行符，但在p元素定义的段落内，元素之间的纵向距离更大。如果在环绕着浮动图像或对象的一段文本中包含br元素，那么新的一行将会在对象下方起始，而不是在环绕文本的下一行（须要通过设置clear属性或对应的样式单）。

示例 `<p>I think that I shall never see<br>A poem lovely as a tree.</p>`

对象模型引用方式 `[window.]document.getElementById(elementID)`

元素专有属性 clear

元素专有事件处理属性 无。

clear

IE *all* NN *all* Moz *all* Saf *all* Op *7* HTML *3.2*

```
clear="常量"
```

可选

如果当前文本环绕着一个浮动图像或其他对象，那么clear属性就可以告知浏览器如何处理br元素之后的下一行文本。其属性值既取决于嵌入了一个或多个图像的页面，也与下一行文本相对于这些图像的预期位置有关。

HTML 4并不赞同使用本属性，建议使用CSS2中的clear样式单属性。

示例 `<br clear="left">`

值 有4个可选的常量值：all | left | none | right。HTML 4.0中还包括一个默认值“none”。IE 3文档中列出了该值，但IE 4却没有，因此可以将属性值设为“none”，浏览器可能做出响应，也可能完全忽视它的存在，但最终的处理还是一致的。

默认值 none

对象模型引用方式 `[window.]document.getElementById(elementID).clear`

<button>

IE *4* NN *n/a* Moz *all* Saf *all* Op *7* HTML *4*

```
<button>...</button>
```

HTML 结束标签：必选

虽然button元素来自于对input元素子类型（包括button、submit和reset）的模仿，但它也具有一些额外的功能，特别是将它作为一个提交类型的按钮时，其功能十分强大。另外，按钮的标注内容不再作为其一个属性值，而是放置在button元素的起始标签和结束标签之间。其他元素也可以用于生成标注内容，甚至使用一个img元素都是可行的（尽管W3C强烈反对使用客户端图像映射）。尽管可以对button元素指定一个样式单，但也可以将其标注内容封装在一个元素（如一个span元素）中，并仅为其指定样式规则。

当button元素被指定了一个“submint”类型的type属性时，浏览器就会像处理其他表单元素一样将该按钮的name和value属性值以名称/值对的形式提交至服务器端。如果指定了其他类型的属性值，则不会有任何表单操作作用于该button元素。

从理论上说，button元素应该嵌入到一个form元素中，但在实际应用中，浏览器完全可以正常地显示一个

独立的button元素。当不须要通过此按钮与脚本一起引用相关表单元素（如一个文本框）时，这样的做法是可以接受的。一些脚本方法简化了表单元素之间的脚本操作，如可以读取even对象中的srcElement或target的form属性。

W3C之所以实现这个input元素的变体，主要是为了能够创建一个不同的、更富观赏性的按钮。实际上，对于这两个不同的元素，不同的浏览器生产商做出了不同的选择，例如，Mozilla中不加区分，在IE 7中则稍有不同，而Safari中则完全迥异。对于Web Forms 2.0（首先在Opera 9中得以实现）而言，button元素获得了其他表单控制元素所共有的许多属性，这使得它能够提交一个表单，但button元素的名称/值对并不会随着表单一起提交。

示例

```
<button type="button" oneClick="doSomething( );">Click Here</button>
<button type="submit" id="sender" value="infoOnly">Request Info</button>
<button type="reset"><img src="clearIt.gif" height="20" width="18"></button>
```

对象模型引用方式 [window.]document.getElementById(*elementID*)

元素专有属性 action、autofocus、datasrc、disabled、enctype、form、method、name、replace、target、template、type、value

元素专有事件处理属性

处理程序	IE	其他	HTML	处理程序	IE	其他	HTML
onafterupdate	4	n/a	n/a	onrowenter	4	n/a	n/a
onbeforeupdate	4	n/a	n/a	onrowexit	4	n/a	n/a

action, enctype, method, replace, target

当button元素的type属性设置为“submit”时，这些Web Forms 2.0功能可以扩展到此元素，详细描述请参见下文中的input元素。

autofocus

IE *n/a* NN *n/a* Moz *n/a* Saf *n/a* Op *n/a* HTML *n/a*

```
autofocus="自动聚焦"
```

可选

这是Web Forms 2.0扩展功能，页面载入后它让元素自动获得焦点。每个页面上只能有一个表单控件元素能拥有此属性。

datafld

IE *4* NN *n/a* Moz *n/a* Saf *n/a* Op *n/a* HTML *4*

```
datafld="列名称"
```

可选

此属性用于IE中的数据绑定，以便将一个远程数据源的列名与按钮的标注联系在一起。数据源的对应列必须是纯文本或HTML，请参见dataformatas。同时，button元素中还必须设定datasrc属性。注意，只能在IE 5/Mac操作系统下实现文本文件数据源的绑定。

HTML 4中依然保留此属性，但它已被XHTML 1.0删除。

示例

```
<button type="button" datasrc="DBSRC3" datafld="label" onClick="getTopStory( );">
Latest News</button>
```

值 区分大小写的标识符。

默认值 无。

对象模型引用方式 [window.]document.getElementById(*elementID*).dataFld

dataformatas

IE *4* NN *n/a* Moz *n/a* Saf *n/a* Op *n/a* HTML | *4* |

`dataformatas="数据类型"` 可选

本属性用于IE浏览器中的数据绑定。对于来自于数据源的原始资料，浏览器根据此属性值决定是将它们视为纯文本还是HTML标签。本属性的设置完全依赖于数据源的构造方式。

HTML 4中依然保留此属性，但它已被XHTML 1.0删除。

示例

```
<button type="button" datasrc="DBSRC3"dataformatas="HTML" datafld="label"
onClick="getTopStory( );"> Latest News</button>
```

值 IE浏览器可识别两种有效的设定值：text和html。

默认值 text

对象模型引用方式 [window.]document.getElementById(*elementID*).dataFormatAs

datasrc

IE *4* NN *n/a* Moz *n/a* Saf *n/a* Op *n/a* HTML | *4* |

`datasrc="数据源名称"` 可选

为实现IE数据绑定功能，此属性用于指定页面内object元素的ID值，以便在进行远程数据访问时加载数据源对象。通过datafld属性指定来自于数据源中的绑定内容。注意，只能在IE 5/Mac操作系统下实现文本文件数据源的绑定。

HTML 4中依然保留此属性，但它已被XHTML 1.0删除。

示例

```
<button type="button" datasrc="DBSRC3" datafld="label"
onClick="getTopStory( );"> Latest News</button>
```

值 区分大小写的标识符。

默认值 无。

对象模型引用方式 [window.]document.getElementById(*elementID*).dataSrc

form

IE *n/a* NN *n/a* Moz *n/a* Saf *n/a* Op *9* HTML *n/a*

`form="表单ID [表单ID] ..."` 可选

通过这个Web Forms 2.0扩展，无论表单是否嵌入了某控件，都可以将一个单独的表单控件元素与一个或多个表单联系在一起。由于button元素并没有像form的子元素那样受到限制，因此它可以放在form元素之外的任何地方。这个属性可以将button元素同页面上一个或多个表单元素联系起来。

示例 `<button type="submit" form="orderForm">Submit Order</button>`

值 页面上一个或多个form元素的ID值。此时，使用空格分隔多个ID值。

默认值 无。

name

IE *4* NN *n/a* Moz *all* Saf *all* Op *7* HTML *4*

`name="元素标识符"` 可选

对于button元素而言，根据type属性设置的不同，name属性可以扮演两种不同角色。针对所有的type属性设定，均可为name属性指定一个标识符，以便在脚本中引用该元素（一般而言，最好通过id属性引用一个元素）。但是对于“submit”类型的按钮而言，在表单提交时，name属性将作为名称/值对的一部分发送至服务器端。

示例 `<button type="submit" name="sender" value="infoOnly">Request Info</button>`

值 区分大小写的标识符。

默认值 无。

对象模型引用方式 `[window.]document.getElementById(elementID).name`

template

IE *n/a* NN *n/a* Moz *n/a* Saf *n/a* Op *9* HTML *n/a*

`template="元素 ID"` 可选

在Web Forms 2.0扩展中，可以将一个button元素与页面上的一个重复的HTML容器块联系在一起。通过类型为“add”的按钮可以将这个重复区块复制到页面上，例如，在表格中加入新的一行。根据type属性的设定值，此属性使按钮与将要增加、删除或移动的区块联系在一起。

示例 `<button type="remove" template="orderFormRow">Delete</button>`

值 页面上已设置为重复区块的元素的ID值。

默认值 无。

type

IE *4* NN *n/a* Moz *all* Saf *all* Op *7* HTML *4*

`type="按钮类型"` 可选

此属性决定了一个按钮的本质类型。按钮类型用于在事件处理程序中启动脚本动作。例如，“reset”类型的按钮和type属性为“reset”的input元素都有相同的作用，它们都用于将所有的元素恢复到默认值。同理，“submit”类型的按钮和type属性为“submit”的input元素也完全相同。由于type属性设置为“reset”或“submit”的button元素的潜在作用会影响到整个页面，因此它必须与一个表单联系起来。

示例 `<button type="reset"><img src="clearIt.gif" height="20" width="18"></button>`

值 3个标准HTML类型（button、reset和submit）之一，不区分大小写。当button与重复块一同使用时，Web Forms 2.0增加了4种额外选择：add | move-down | move-up和remove。

默认值 button

对象模型引用方式 `[window.]document.getElementById(elementID).type`

value

IE *4* NN *n/a* Moz *all* Saf *all* Op *7* HTML *4*

`value=" 文本"` 可选

当button元素是表单成员时，使用此属性可以为该元素预先指定一段文本，它将作为名称/值对的一部分提交到服务器端。

示例 `<button name="connections" id="connections" value="ISDN">ISDN</button>`

值 任何文本字符串。

默认值 无。

对象模型引用方式 `[window.]document.getElementById(elementID).value`

<canvas>

IE *n/a* NN *n/a* Moz *1.8* Saf *2* Op *9* HTML *n/a*

`<canvas></canvas>` HTML 结束标签：必选

canvas元素在文档内定义了一个矩形区域，脚本可以在这个区域内绘制直线和曲线，从而达到不借助于图片就可以显示图表等信息的目的。在实际应用中，往往使用单一颜色或渐变色填充该矩形区域。由于文档中的HTML元素实际上只是针对空间的占位符，因此几乎没有与此元素相关联的属性。关于如何在canvas元素中进行图形绘制的更多详细信息，请参见第2章中的canvas对象。

在Safari浏览器中，此元素并不需要结束标签，但对于其他浏览器而言，建议使用结束标签，这样就可以在文档中使用后备内容,例如，当浏览器不支持该元素时，显示文字或图像。然而，很不幸的是，在Safari 2中后备内容会与canvas一同显示出来。

60 canvas元素起源于Apple公司的Safari浏览器，但它也是在网页超文本技术工作小组（WHATWG）的网页应用程序1.0规范中才得以定型。更多相关细节可浏览*http://www. whatwg.org*。

示例 `<canvas id="myCanvas" height="300" width="300"></canvas>`

对象模型引用方式 `[window.]document.getElementById(elementID)`

元素专有属性 `height、width`

元素专有事件处理属性 无。

height, width

IE *n/a* NN *n/a* Moz *1.8* Saf *2* Op *9* HTML *all*

`height="长度值" width="长度值"` 可选

此长度值单位为像素，它指定了为脚本图形绘制而预留的矩形区域大小。

示例 `<canvas id="pieChart" height="400" width="400"></canvas>`

值 正整数像素值。

默认值 Height: 150；width: 300

对象模型引用方式

```
[window.]document.getElementById(elementID).height
[window.]document.getElementById(elementID).width
```

<caption>

IE *all* NN *all* Moz *all* Saf *all* Op *7* HTML *3.2*

`<caption>...</caption>` HTML 结束标签：必选

caption元素仅在table元素中使用（并紧跟在<table>标签之后），用于表示此表格的标题文本。须要注意的是，caption元素适用于整个表格，而表格的表头元素（th）则仅仅适用于表格的某一行或列。而且在table元素中，仅能识别一个caption元素。

caption元素简要地描述了表格的主体内容。而table元素的大段描述则往往编写在summary属性中，以便使用浏览器中的文本-语音转换技术。此元素的主要属性是align。尽管HTML 4反对使用此元素，但它的确可以决定在何处显示相关的表格标题。

示例

```
<table ...>
<caption class="tableCaptions">
   Table 3-2. Sample Inverse Framistan Values
</caption>
61 ...
</table>
```

对象模型引用方式 `[window.]document.getElementById(elementID)`

元素专有属性 `align、valign`

元素专有事件处理属性 无。

align

IE *all* NN *all* Moz *all* Saf *all* Op *7* HTML *3.2*

`align="对齐位置"` 可选

此属性决定标题在表格中的实际显示位置。并不是所有版本的浏览器均支持该属性的全部可选属性值,其中仅

有top和bottom适用于绝大多数浏览器。在IE和Opera浏览器中，当此属性指定为“left”或“right”时，标题依然位于表格的上方，而不是位于表格的左侧或右侧。

浏览器往往将标题水平居中显示在表格的上方或下方，而标题字体和正文字体相同，当然也可以使用标签或样式单改变其字体。如果标题长度宽于表格，那么超出部分将在第二行显示，并且依然保持水平居中的对齐方式。

HTML 4.0已不再赞成使用此属性，推荐使用样式单属性text-align和vertical-align。

示例 `<caption align="top">Table II.Stock List</caption>`

值

根据浏览器的不同版本，适用于此属性的属性值也不相同。可以根据实际工作中的部署环境从下表中选择合适的属性值。

值	IE 4+	NN 4	Mozilla	Safari	Opera	HTML 4
bottom	·	·	·	·	·	·
center	·	—	·	·	·	—
left	·	—	·	—	·	·
right	·	—	·	—	·	·
top	·	·	·	·	·	·

具体实现细节可参见本章前文部分，在“对齐常量”一节中已对容器盒内的文本对齐进行了相关讨论。

默认值 top（在IE浏览器中，如果valign属性也进行了设置，那么默认值为center）。

对象模型引用方式 `[window.]document.getElementById(elementID).align`

valign

IE *3* NN *n/a* Moz *n/a* Saf *n/a* Op *n/a* HTML *n/a*

`valign="对齐位置"`

可选

valign属性是IE浏览器的早期属性，它决定了在表格上方还是下方放置表格标题。尽管此属性值现在已是align属性的一部分，但IE浏览器对align相关属性值（left、center和right）的特殊处理方式，依然使得valign有一定的用武之地。例如，可以使用valign属性使标题位于表格下方，同时使用align="right"使得标题在底部右对齐。而通过HTML 4的属性，则无法实现这种组合形式。

示例 `<caption align="right" valign="bottom">Table 3-2. Fiber Content.</caption>`

值 两个可用值：bottom和top，且不区分字母大小写。

默认值 top

对象模型引用方式 `[window.]document.getElementById(elementID).valign`

IE *all* NN *all* Moz *all* Saf *all* Op *7* HTML *3.2*

`<center>...</center>`

HTML结束标签：必选

Netscape首先引入了center元素，在W3C认可的div元素投入使用之前，它得到了广泛的应用。但很明显的是，即使从HTML 3.2规范来看，HTML工作组也从未中意过此元素。然而，实际的使用最终取得了胜利，HTML 3.2规范还是收录了这个元素。HTML 4已不再赞成使用此属性，建议使用一个由样式单规则text-align:center控制的div元素来取代它。另外，也可以使用align="center"替换div元素和样式单，但HTML 4并不赞成这种方式。

由center元素包含的内容将沿一条轴线对齐，这条轴线贯穿最外层容器元素（一般是body或html元素）的中部。

示例 `<center>Don't do this.</center>`

对象模型引用方式 [window.]document.getElementById(*elementID*)
元素专有属性 无。
元素专有事件处理属性 无。

63

<cite>

IE *all* NN *all* Moz *all* Saf *all* Op *7* HTML *all*

<cite>...</cite>

HTML 结束标签：必选

在HTML 4推荐标准中，有一类元素统称为**短语元素**（phrase element），cite元素正是其中之一。这些元素为文档的特定部分指明了结构上的含义。它们中的cite元素就包含了对其他资源的引用和参考。它并不是一个有效的链接，而是一个指明了元素内容的简单符号。搜索引擎和其他的HTML文档解析器也许可以使用该信息，例如,利用它可以生成一个文档的参考书目。

浏览器可以自由决定如何（或是否）从其他body元素中区分出cite元素内容。目前主流浏览器使用斜体字进行区分，也可以使用任何合适的样式单来替换默认的显示形式。

示例

```
<p>Trouthe is the hyest thing that many may kepe.<br>
(Chaucer, <cite>The Franklin's Tale</cite>)</p>
```

对象模型引用方式 [window.]document.getElementById(*elementID*)
元素专有属性 无。
元素专有事件处理属性 无。

<code>

IE *all* NN *all* Moz *all* Saf *all* Op *7* HTML *all*

<code>...</code>

HTML 结束标签：必选

在HTML 4推荐标准中，有一类元素统称为**短语元素**（phrase element），code元素也是其中之一。这些元素为文档的特定部分指明了结构上的含义。它们中的code元素主要用于凸显出表示计算代码的成排字符（如程序指令、变量名称、关键字，等等）。

浏览器可以自由决定如何（或是否）从其他body元素中区分出code元素的内容。主流浏览器使用等宽字体显示code元素中的内容，同默认的正文字体相比，这个字体要明显小一些（注意，在Macintosh中的IE 4并未缩小字符）。当然也可以使用任何合适的样式单来替换默认的显示形式。

由于浏览器会将code元素内空白（包括回车符号）内容与文档内body元素的内容视为一致，采取相同的处理方式，因此必须手工插入br元素作为换行符。另外，请参见pre元素显示预格式化文本，它可以显示源代码内所有的空白内容。

示例

```
<p>Initialize a variable in JavaScript with the <code>var</code> keyword.</p>
```

64

对象模型引用方式 [window.]document.getElementById(*elementID*)
元素专有属性 无。
元素专有事件处理属性 无。

<col>

IE *3* NN *n/a* Moz *all* Saf *all* Op *7* HTML *4*

<col>

HTML 结束标签：禁用

col元素主要用于表格或表格列表分组内的列表子集，它为指定列表子集的宽度和其他特征（样式）提供了一条捷径。由于在table元素中其信息出现得较早，浏览器可以在载入全部表格代码前就准备开始绘制表格，

否则这时就只能进行一些表格空间上的估算工作。

可以将col元素与colgroup元素或多个col元素联合使用。具体的使用结构取决于想如何为单独的列或相邻的列指定宽度和其他样式。一个col元素可以应用于多个相邻的列。为其span属性分配一个整数值之后，浏览器就会将col元素的宽度或样式设置应用于上述数目的连续列中。span属性类似于colgroup元素的colspan属性。为了与colgroup协调一致，可以在一个colgroup集合内创建相关列的子集。

须要注意的是，无论在表格中如何编排列结构，由col和colgroup定义的列总数必须与表格中的实际列数相等。下面的3个框架示例展示了包含6列的4个表格：

```
<table>
<col span="6">
...
</table>

<table>
<col>
<col span="4">
<col>
...
</table>

<table>
<colgroup>
<col span="2"></colgroup>
<colgroup span="4">
...
</table>
```

但是目前在大多数浏览器中尚不能完全实现HTML 4规范对于col元素的定义。例如，HTML 4规定，对于列的调整应可精确到该列中的任一字符，例如，十进制资金数额中的小数点。而且在col元素之后的基本原理之中也添加这一特性。例如，可以构造这样一个表格：前3列按照一种风格设计，而第4列则指定另一种独有的调整方式，相关代码如下：

```
<html>
<head>
<style type="text/css">
  .colHdrs {color:black}
  .normColumn {color:green}
  .priceColumn {color:red}
</style>
</head>
<body>
<table>
<colgroup class="normColumn" span="3"></colgroup>
<col class="priceColumn" align="char" char=".">
<thead class="colHdrs">
<tr><th>Stock No.<th>In Stock<th>Description<th>Price</tr>
<tbody>
<tr><td>8832<td>Yes<td>Brass Frobnitz<td>$255.98</tr>
<tr><td>8835<td>No<td>Frobnitz (black)<td>$98</tr>
...
</table>
</body>
</html>
```

由于col和colgroup属性都应用于整列，因此在此例中thead元素的样式单规则将取代另外两种列样式中规定的颜色设置，并作用于由thead元素封装的行中。上面这个例子在IE 4及后续版本均可运行，但最后一列的

属性设置却会被忽略。

非IE浏览器对这一例子的支持主要取决于其DOM的实行情况。DOM将会通知脚本程序，col元素和它的属性是存在的（以特性的形式反映出来）。但在Mozilla 1.8、Safari 2和Opera 9中，col元素并未呈现出其预期的效果。

示例 `<col class="dateCols" width="15" align="right">`

对象模型引用方式 `[window.]document.getElementById(elementID)`

元素专有属性 `align、ch、char charoff、choff、span、valign、width`

元素专有事件处理属性 无。

align

IE *3* NN *n/a* Moz *n/a* Saf *n/a* Op *8* HTML *4*

`align="对齐常量"` **可选**

在col元素控制的列中，此属性确定其水平对齐的特点。HTML 4规范定义了一些在CSS规范中未曾涉及的align属性值。例如，在CSS中就没有针对单一字符的样式设置方式。

示例 `<col class="dateCols" width="15" align="right">`

值

HTML 4和IE/Opera浏览器支持两组属性值，如下表所示：

值	IE 和 Opera	HTML 4	值	IE 和 Opera	HTML 4
center	·	·	left	·	·
char	–	·	right	·	·
justify	–	·			

center、left和right这三个值不解自明，其含义甚至也可以推广至CSS的text-align属性。如果使用justify这种对齐方式，将会分散排列其内容，直到使它们与左右边缘平齐。而针对char这种对齐方式，在使用时必须为char属性指定一个字符，并将会以这个字符为中心进行对齐。

必须牢记的是，align属性将作用于col元素所涉及的每一列中的所有行，其中还包括为表格指定的全部th元素。如果希望调整起始列的样式，可以使用单独的align属性或text-align样式单属性为thead元素或th元素设定样式。

默认值 left

对象模型引用方式 `[window.]document.getElementById(elementID).align`

char

IE *n/a* NN *n/a* Moz *n/a* Saf *n/a* Op *n/a* HTML *4*

`char="某字符"` **可选**

char属性用于指定一个字符，该字符会作为某一列中文本对齐的参考点。只有当align属性设置为“char”时，此属性才有价值。而微软则提出了一个ch属性，它可以和标准的char属性相对应。可惜的是，浏览器无论如何都不会响应char或ch属性。

示例 `<col class="priceColumn" align="char" char=".">`

值 任意一个单独的字符。

默认值 无。

charoff

IE *n/a* NN *n/a* Moz *n/a* Saf *n/a* Op *n/a* HTML *4*

`charoff="长度值"` **可选**

charoff属性用于指定一个特定的偏移点，char属性所指定的字符将在单元格内的该位置上出现。提供此属

性主要是为了防止浏览器的默认定位不符合表格的设计目标。而微软则提出了一个choff属性，它可以和标准的charoff属性相对应。可惜的是，浏览器无论如何都不会响应这两个属性。

示例 `<col class="priceColumn" align="char" char="." charoff="80%">`

值 任意一个单位为像素的长度值或单元格空间的百分比。

默认值 无。

choff

参见charoff。

Span

IE 3 NN n/a Moz n/a Saf n/a Op n/a HTML 4

`span="列数"` 可选

此属性决定了相邻列的数量，col元素的属性和样式设置会作用于这些列。如果缺失此属性，那么col元素仅控制单独的一列。可以将span属性值各不相同的多个col元素组合在一起，以满足不同的列分组需求。

示例 `<col span="3">`

值 大于0的整数值。

默认值 1

对象模型引用方式 `[window.]document.getElementById(elementID).span`

valign

IE 4 NN n/a Moz n/a Saf n/a Op n/a HTML 4

`valign="对齐常量"` 可选

在col元素控制的列单元格中，此属性确定其中内容的竖直对齐形式。可以为列中的任意一个特定单元格更换竖直对齐方式。

示例 `<col valign="middle">`

值 IE 4及后续版本和HTML 4均可识别以下4个常量值：top、middle、bottom和baseline。当属性值设定为top和bottom时，其内容将会齐平或紧贴表格单元格的顶部或底部显示。设置为middle（默认值）时，其内容会恰好悬浮在单元格的竖直中心处。如果单元格内的内容可能会依照通常的窗口宽度（假设为表格宽度）分隔为多行，那么最好将同一行（或列）中所有单元格的valign属性均设置为baseline。这样就可以保证单元格内第一行文本的字符基线能够与行内其他单元格的文本对齐，而这是最符合审美观的编排方式。

默认值 middle

对象模型引用方式 `[window.]document.getElementById(elementID).valign`

width

IE 4 NN n/a Moz all Saf all Op 7 HTML 4

`width="长度"` 可选

此属性决定了由col元素控制的列的最大宽度。实际上，浏览器在渲染每一列时，其列宽不会小于单元格内连续字符串（例如，最长的一个单词）的宽度。当然，每一列的精确宽度也取决于其中文本的字体特征。

示例 `<col width="100">`

值 在IE中，width的属性值既可以是像素值（不须要添加“px”单位符号），也可以用整个表格宽度的百分比来表示，如width="25%"。

HTML 4规范还引入了一个新的长度度量方式，作为上面两种度量方式的补充。这种方式称为比例长度（proportional length），也可称为多级长度（MultiLength）。它展现了一种新的长度标记和布局方案。如果只有在所有的固定长度和百分比长度都获取后，才能根据它们计算col元素在表格空间中的可用宽度，那么这种比例长度就非常适用。使用比例长度标记（一个数字和一个星号*），就可以指引浏览器根据比例分配剩余空间。例如，如果所有其他列宽度都计算完毕之后还剩余100像素的水平空间，那么就可以将比例长度为三个col元素的width属性分别赋值为1*、3*和1*。这样就形成了总共5个成比例的分段。根据之前每一列在剩余空间里的比例长度，这100像素的可用空间就会划分为三段，分别为20像素、60像素和20像素。

默认值　由浏览器计算后决定。

<colgroup>

IE *3* NN *n/a* Moz *all* Saf *all* Op *7* HTML *4*

<colgroup>...</colgroup>　　HTML结束标签：可选

如果要为表格中的列子集指定宽度和其他特征（样式），那么使用colgroup元素能很快达到这一目的。由于在table元素中其信息出现得较早，浏览器就可以在载入全部表格代码前就准备开始绘制表格，否则这时就只能进行一些空间上的估算工作。

可以将colgroup元素与col元素或与多个colgroup元素联合起来使用。也可以定义一个colgroup，并在其中嵌套一些col元素，通过col元素可以将这些共享部分属性或样式设置的列定义为列子集。colgroup之后是否出现独立的col元素则决定了是否需要元素的结束标签。例如，如果完全使用colgroup元素来指定列分组，那么就不须要在HTML中使用结束标签，如下例所示：

```
<table>
<colgroup span="2" width="30">
<colgroup span="3" width="40">
<thead>
```

如果在colgroup元素之后还存在独立的col元素，那么就必须在独立的col元素之前明确地使用结束标签，如下例所示：

```
<table>
<colgroup class="leftCols">
<col width="30">
<col width="20">
</colgroup>
<col class="priceCol" width="25">
<thead>
...
```

具体的结构完全取决于您想怎样为单独或连续的列指定宽度或其他样式。可以使用span属性建立一个包含多个连续列的列分组。这与td元素中的colspan属性完全不同，colspan属性会将多个连续的单元格融合成一个单元格。此时，span元素仅仅只是帮助决定可视为一体的列数目，这样一来，无论每列中包含了什么具体内容，都可以同时为多个列指定相同的属性值和样式单。

须要注意的是，无论在表格中如何编排列结构，由col和colgroup定义的列总数必须与表格中的实际列数相等。例如，在下面的三个框架中，指定了几个含有6列的HTML 4表格：

```
<table>
<colgroup span="6">
...
</table>
<table>
<col>
<colgroup span="4">
```

```
<col>
...
</table>
<table>
<colgroup>
   <col span="2">
</colgroup>
<colgroup span="4">
...
</table>
```

但是目前在大多数浏览器中尚不能完全实现HTML 4规范中对于colgroup元素的定义。例如，HTML 4规定，一列内的对齐应该作用到该列中的每个字符，例如十进制资金数额中的小数点。在col元素之后的基本原理之中已添加这种特性（可以参见上文col元素中的例子）。

从语句构成上来看，colgroup和col元素之间存在着小小的区别。colgroup元素可以使分组中的行保持框架的完整性，在包含它们的table元素指定rules="groups"之后，显示效果会有所不同。此时，浏览器仅会在colgroup元素之间绘制线条（在IE中就是标准的表格边框线），而在col元素之间则不会绘制。

非IE浏览器对这一例子的支持主要取决于其DOM的实行情况。DOM将会通知脚本程序，colgroup元素和它的属性是存在的（以特性的形式反映出来）。但在Mozilla 1.8和Opera 9中，colgroup元素并未完成其计划中的任务而呈现出预期的效果。

示例 `<colgroup class="dateCols" width="15" align="right">`

对象模型引用方式 `[window.]document.getElementById(elementID)`

元素专有属性 `align、char、charoff、span、valign、width`

元素专有事件处理属性 无。

align

IE *3* NN *n/a* Moz *n/a* Saf *n/a* Op *8* HTML *4*

`align="对齐常量"` 可选

在colgroup元素控制的列中，此属性确定其水平对齐的特点。HTML 4规范定义了一些在CSS规范中未曾涉及的align属性值。例如，在CSS中就没有针对单一字符的样式设置方式。

示例 `<colgroup class="dateCols" width="15" align="right" span="3">`

值

HTML 4和IE浏览器支持两组属性值，如下表所示：

值	IE 和 Opera	HTML 4	值	IE 和 Opera	HTML 4
center	·	·	left	·	·
char	—	·	right	·	·
justify	—	·			

center、left和right这三个值不解自明。如果使用justify这种对齐方式，将会分散排列其内容，直到使它们与左右边缘平齐。而针对char这种对齐方式，在使用时必须为char属性指定一个字符，并将会以这个字符为中心进行对齐。在HTML 4规范的示例中，未包含字符的内容将会与同一列中其他行的文本内容右对齐。必须牢记的是，align属性将作用于col元素所涉及的每一列中的所有行，其中还包括为表格指定的全部th元素。如果希望调整列头元素的样式，可以使用单独的align属性或text-align样式单属性为thead元素或th元素设定样式。

默认值 left

对象模型引用方式 `[window.]document.getElementById(elementID).align`

char

IE *n/a* NN *n/a* Moz *n/a* Saf *n/a* Op *n/a* HTML *4*

```
char="某字符"
```

可选

char属性用于指定一个字符，该字符会作为某一列中文本对齐的参考点。只有当align属性设置为char时，此属性才有价值。而微软则提出了一个ch属性，它可以和标准的char属性相对应。可惜的是，浏览器无论如何都不会响应char或ch属性。

示例 `<colgroup class="priceCols" align="char" char="." span="2">`

值 任意一个单独的字符。

默认值 无。

charoff

IE *n/a* NN *n/a* Moz *n/a* Saf *n/a* Op *n/a* HTML *4*

```
charoff="长度值"
```

可选

charoff属性用于指定一个特定的偏移点，char属性所指定的字符将在单元格内的该位置上出现。提供此属性主要是为了防止浏览器的默认定位不符合表格的设计目标。而微软则提出了一个choff属性，它可以和标准的charoff属性相对应。可惜的是，浏览器无论如何都不会响应char或ch属性。

示例 `<colgroup class="priceColumn" align="char" char="." charoff="80%" span="2">`

值 任意一个单位为像素的长度值或单元格空间的百分比。

默认值 无。

span

IE *3* NN *n/a* Moz *n/a* Saf *n/a* Op *n/a* HTML *4*

```
span="列数"
```

可选

此属性决定了相邻的列数量，colgroup元素的属性和样式设置会作用于这些列。如果缺失此属性，那么colgroup元素仅控制单独的一列。可以将span属性值各不相同的多个colgroup元素组合在一起，以满足不同的列分组需求。

示例 `<colgroup span="3">`

值 大于0的整数值。

默认值 1

对象模型引用方式 `[window.]document.getElementById(elementID).span`

valign

IE *3* NN *n/a* Moz *n/a* Saf *n/a* Op *n/a* HTML *4*

```
valign="对齐常量"
```

可选

在colgroup元素控制的列单元格中，此属性确定其竖直对齐的特点。可以为列中的任意一个特定单元格更换竖直对齐方式。

示例 `<colgroup valign="middle">`

值 IE 4及后续版本和HTML 4均可识别以下4个常量值：top、middle、bottom和baseline。当属性值设定为top和bottom时，其内容将会齐平或紧贴表格单元格的顶部或底部显示。设置为middle（默认值）时，其内容会恰好悬浮在单元格的竖直中心处。如果单元格内的内容可能会依照窗口宽度(假设为表格宽度)分割为多行，那么最好将同一行(或列)中所有单元格的valign属性均设置为baseline。这样就可以保证单元格内文本的第一行的字符基线能够与行内其他单元格的文本对齐，而这是最符合审美观的编排方式。

默认值 middle

对象模型引用方式 [window.]document.getElementById(*elementID*).valign

width

IE *3* NN *n/a* Moz *all* Saf *n/a* Op *7* HTML *4*

`width="多倍长度"` 可选

此属性决定了由colgroup元素控制的列的最大宽度。实际上，浏览器在渲染每列时，其列宽不会小于单元格内连续字符串(例如，最长的一个单词)的宽度。当然，每一列的精确宽度也取决于其中文本的字体特征。

示例 `<colgroup width="100">`

值

width的属性值既可以是像素值（不须要添加“px”单位符号），也可以用整个表格宽度的百分比来表示，如width="25%"。

HTML 4.0规范还介绍了另一种可供选用的比例长度。在colgroup元素中指定width="0*"后，可以命令浏览器根据显示单元格内容所需的最小宽度绘制所有的列。由于浏览器须要进行这样的计算，因此它必须一次加载所有的表格内容，这也就避免了逐步绘制一个长表格的可能性。可以参见col元素中的width属性，以获取关于比例长度的更多信息。

默认值 由浏览器计算后决定。

<comment>

IE *all* NN *n/a* Moz *all* Saf *all* Op *7* HTML *n/a*

`<comment>...</comment>` HTML 结束标签：必选

comment元素是早期IE浏览器的一个产物。曾经期望使用这个明语标签来替代<!--comment-->注释元素。虽然IE浏览器不会显示comment元素中的内容，但其他所有浏览器均会显示，因此不要使用这个元素。为了尽量避免使用此元素，本书对其不加以详述。

<datalist>

IE *all* NN *all* Moz *all* Saf *all* Op *9* HTML *all*

`<option>...</option>` 结束标签：必选

datalist是Web Forms 2.0中的一种元素，那些与包含文本输入域的iput元素联合使用的预定义项就由它来提供语意内容。而嵌入在datalist元素中的option元素则用来指定每个单独的预定义项。

在使用时，必须首先为datalist元素的id属性指定一个标识符。此后，与此列表联合使用的表单控件元素中必须设置一个list属性，并且将其属性值指定为datalist的id值。虽然用户并非必须在这个数据列表选择中做出选择（在默认情况下，可以编辑该文本输入域），但这些选项可以为常用的或最近的记录加快数据输入速度。

示例

```
<label>Enter your operating system:<input type="text" name="os" list="oses" /></label>
<datalist id="oses">
   <option value="">
   <option value="Windows Vista">
   <option value="Windows XP">
   <option value="Windows 98">
   <option value="Mac OS X">
   <option value="Linux">
</datalist>
```

对象模型引用方式 [window.]document.getElementById(*elementID*)

元素专有属性 无。

元素专有事件处理属性 无。

<dd>

IE *all* NN *all* Moz *all* Saf *all* Op *7* HTML *all*

```
<dd>...</dd>
```

HTML 结束标签：可选

dd元素是dl、dt和dd 3元素小组的成员，这3个元素可用来在文档中创建一个定义列表。dl元素的标签首先将整个列表都括起来，然后使用前导的dt元素标签指明一个定义项，最后用前导的dd元素标签指出该条目的描述内容。下面这段代码扼要地定义了一个包含3个条目的定义列表序列：

```
<dl>
    <dt>Term 1</dt>
    <dd>Definition 1</dd>
    <dt>Term 2</dt>
    <dd>Definition 2</dd>
    <dt>Term 3</dt>
    <dd>Definition 3</dd>
</dl>
```

dt元素是一个行内元素，而dd元素可以包含块级内容，这些块级内容包括带花边的文字、图像和其他对象等。对dt和dd元素而言，可以不使用结束标签，因为下一个起始标签实际上就标志着前一个元素的结束。但对整个列表而言，必须使用结束标签来完成dl元素的封装。

尽管HTML规范并没有强制规定定义列表的显示方式，但在主流浏览器中，均使用左对齐来显示dt元素，并缩进显示任何紧随其后的dd元素。虽然目前浏览器尚未为此添加任何的特殊字体格式或可视化元素，但可以根据喜好自由指定显示样式。如果想要存储多个条目或描述，可以在源代码中放置多个dt或dd元素。

HTML一直作为上下文相关的标记语言而不断发展，那么就要力图避免在使用中仅将定义列表作为一种格式化的手段（获得文本缩进）。可以使用样式单或可调节的边距设定来完成格式化任务。

在Navigator 4中，任何通过class、id或style属性指派给dt和dd元素的样式都无法工作。如果希望为dt和dd元素都赋以同样的样式特性，那么也请为dl元素指定该样式；另外可以将每个dt和dd元素均放置在span元素之中，这样内嵌的dt和dd元素就会继承span元素的样式。对那些不会在NN 4中显示的文档而言，虽然完全没有必要这样做，但这种做法在其他浏览器中也是奏效的。

示例

```
<dl>
    <dt>Z-scale</dt>
    <dd>A railroad modeling scale of 1:220. The smallest mass-produced
    commercial model scale.</dd>
</dl>
```

对象模型引用方式 [window.]document.getElementById(*elementID*)

元素专有属性 无。

元素专有事件处理属性 无。

<del>

IE *4* NN *n/a* Moz *all* Saf *all* Op *7* HTML *4!*

```
<del>...</del>
```

HTML 结束标签：必选

del元素与其对应元素ins定义了一种格式，用于表示在编辑过程中已经标识为删除或插入的那部分内容。虽然这已远离了工作流管理方案的范畴，但在使用一款支持所见即所得的HTML编辑工具时，这些元素有助于在编辑过程中控制每一轮的修改内容。

del元素中包含的文本内容将会被加上删除线，而ins元素则会添加下划线。HTML 4规范为了保存修改时间和修改注释等隐藏信息，还引入了另外两种可能有用的属性。

示例

```
<p>Four score and <del cite="Fixed the math">eight</del><ins>seven</ins> years ago...</p>
```

对象模型引用方式 [window.]document.getElementById(*elementID*)

元素专有属性 cite、datetime

元素专有事件处理属性 无。

cite

IE *6* NN *n/a* Moz *all* Saf *all* Op *7* HTML *4*

`cite="文本"` 可选

此属性用于说明修改的原因或记录与元素相关联的其他信息，通常并不显示此信息。在Mozilla的上下文菜单中，为此元素提供了一个属性选项，这样就可以将元素内的属性与属性值都呈现给网页的浏览者。DHTML脚本也可以通过元素对象的cite属性访问这些信息，并添加描述信息。

示例

```
<del cite="Fixed the math --A.L.">eight</del>
```

值 任何字符串。该字符串应该位于一对引号之中，双引号是推荐的标准格式，而也可使用单引号。

默认值 无。

对象模型引用方式 [window.]document.getElementById(*elementID*).cite

datetime

IE *6* NN *n/a* Moz *all* Saf *all* Op *7* HTML *4*

`datetime="时间"` 可选

此属性表示删除时的日期与时间。一般来说，只有使用具备修改记录功能的HTML编辑工具，才会在文档中添加这种信息。日后在进行文档修改的审核跟踪时，就可以调用属性中记录的修改时间。在一个给定的del元素中，只能存在一个相关的datetime属性值。在Mozilla的上下文菜单中，为此元素提供了一个属性选项，这样就可以将元素内的属性与属性值都呈现给网页的浏览者。DHTML脚本也可以通过元素对象的datetime属性访问这些信息，并添加描述信息。

示例

```
<del datetime="2001-09-11T08:56:00-04:00">SomeDeleteTextHere</del>
```

值

datetime属性需要一个特殊的日期-时间格式，这样无论处于世界上哪一个时区，都可以根据这个信息推算出对应的准确时刻。该语法格式为：*yyyy-MM-ddThh:mm:ssTZD*。

yyyy

四位数字年份。

MM

两位数字月份（从01至12）。

dd

两位数字日期（从01至31）。

T

使用大写字母“T”将时间和日期分隔开来。

hh

两位数字的24小时制小时（从00至23）。

mm

两位数字分钟（从00至59）。

<dir>

ss

两位数字秒（从00至59）

TZD

时区标志符

77 时区标志符有两种格式。第一种格式仅使用大些字母“Z”，它代表UTC（Coordinated Universal Time，协调世界时，也称为“Zulu”）。另一种时间格式则使用“*hh:mm:ss*”，它指出相对于UTC的偏移量。这个时间偏移量还包含一个“+”或“-”，以及另一对“*hh:mm*”值。如果时区在格林尼治标准时间（Greenwich Mean Time，简称GMT，它实际上与UTC相同）以西，那么该地区时间要早于UTC，因此其偏移量运算符使用“-”。同理，如果在GMT以东，则偏移量使用正值。例如，太平洋标准时间（简称PST）比UTC早8小时，那么在PST时区为18：00时，UTC是第二天的凌晨2点。因此，以下这些例子均表示时间中的同一时刻（注意，出于突出的目的才加粗显示时区标志符，在实际使用中并不需要这样）：

2003-01-30T02:00:00**Z**	UTC	2003-01-29T18:00:00-**08:00**	Pacific Standard Time
2003-01-29T21:00:00-**05:00**	Eastern Standard Time	2003-01-30T13:00:00+**11:00**	Sydney, Australia

请参考ISO-8601标准以获取更多关于时间表示方法的细节信息。

默认值 无。

<dfn>

IE *3* NN *n/a* Moz *all* Saf *all* Op *7* HTML *3.2*

<dfn>...</dfn>

HTML 结束标签：必选

在HTML 4推荐标准中，有一类元素统称为**短语元素**（*phrase element*），dfn元素也是其中之一。这些元素为文档的特定部分指明了结构上的含义。dfn元素指明某条目在文档（在该文档内定义了此条目）中是第一次被使用。在文档中第一次使用一个重要的词汇条目时，所采取的一个常用处理技巧是使用斜体显示这个条目。通常这也是条目在文档中得到定义的位置，这样，在下文中使用该条目时，其含义就会十分明了。主流浏览器在默认情况下会用斜体显示dfn元素内的所有文本。当然，也可以很简单地使用样式单规则自定义dfn元素的样式。

示例

```
<p>Concerto composers usually provide a space for soloists to show off technical
skills while reminding the audience of various themes used throughout the
movement.This part of the concerto is called the <dfn> cadenza</dfn>.</p>
```

对象模型引用方式 [window.]document.getElementById(*elementID*)

元素专有属性 无。

元素专有事件处理属性 无。

<dir>

78

IE *all* NN *all* Moz *all* Saf *all* Op *7* HTML *all*

<dir>...</dir>

HTML 结束标签：必选

使用dir元素的最初设想，是允许浏览器生成条目的多栏式列表。但实际情况却是，每个浏览器都将dir元素视作ul元素，即显示条目的无序单列列表（往往以一个圆点作为开头）。HTML 4并不赞成使用dir元素，而严格的HTML 4或XHTML DTD甚至已经不再支持这个元素。由于dir元素将永远从浏览器的视野中消失，因此如果要保证代码的兼容性，应该在任何情况下都使用ul元素来代替它。此处所阐述的任何内容均适用于另一个不再赞成使用的元素——menu。

示例

```
Common DB Connector Types:
<dir>
    <li>DB-9</li>
    <li>DB-12</li>
    <li>DB-25</li>
</dir>
```

对象模型引用方式 [window.]document.getElementById(*elementID*)

元素专有属性 compact

元素专有事件处理属性 无。

compact

IE 3 NN n/a Moz n/a Saf n/a Op n/a HTML 3.2

compact 可选

该属性值是一个布尔变量，设计的初衷是让浏览器能够以另一种更为紧凑的样式显示列表，即与正常情况相比，使条目之间的行距更小。实际上，主流浏览器在响应此属性时并不会调整显示效果。

示例 <dir compact>...</dir>

值 如果此属性已出现，就意味着其属性值为true。

默认值 false

IE all NN all Moz all Saf all Op 7 HTML 3.2

... HTML 结束标签：必选

div元素为文档中的任何块级内容提供了结构和上下文关系。与那些被赋予了特殊内涵的结构化元素（如p元素）不同，作者们可以依靠div元素的属性和内嵌的内容，自由地为每个div元素都赋予不同的含义。只须要将相关的内容放入起始和结束标签之内，每个div元素都可以变成一个通用的块级容器。

如果须要使用一个样式单规则控制多个元素内容的样式，那么使用div元素进行包装是最合适的选择。例如，如果一段文字包含3个段落，那么就可以使用div元素封装这3个p元素，div元素的样式单就能为它们定义所需的字体样式，从而避免单独为它们指定字体样式。这个样式单既可以在div内嵌的style属性中定义，也可以通过class或id属性来指定，具体采取哪种方式取决于其余的文档结构。

div元素是块级元素，如果想为成行的内容选择一个更为合适的容器，则请使用span元素。

示例 ...

对象模型引用方式 [window.]document.getElementById(*elementID*)

元素专有属性 align、datafld、dataformatas、datasrc、nowrap

元素专有事件处理属性 无。

align

IE 3 NN 2 Moz all Saf all Op 7 HTML 3.2

align="对齐常量" 可选

若想获取块级元素内水平对齐的详细信息，请参考本章前文中“对齐常量”一节。

示例 Part IV

值 常量。Navigator 4、IE 4及后续版本和当前的浏览器可识别由HTML 4指定的4种常量：center | left | right和justify。但Macintosh系统下的IE 4浏览器无法识别justify。

默认值 left或right，取决于当前语言的显示方向。

对象模型引用方式 [window.]document.getElementById(*elementID*).align

datafld

IE 4 NN n/a Moz n/a Saf n/a Op n/a HTML 4

datafld="列名称" 可选

此属性用于IE中的数据绑定，以便将一个远程数据源的列与div元素的HTML内容联系在一起。数据源的对应列必须是HTML（参见dataformatas部分），div元素中还必须设置datasrc和dataformatas属性。注意，只能在IE 5/Mac操作系统下实现文本文件数据源的绑定。

HTML 4中依然保留此属性，但它已被XHTML 1.0删除。

示例

值 区分大小写的标识符。

默认值 无。

对象模型引用方式 [window.]document.getElementById(*elementID*).dataFld

dataformatas

IE 4 NN n/a Moz n/a Saf n/a Op n/a HTML 4

dataformatas="数据类型" 可选

本属性用于IE浏览器中的数据绑定。对于来自于数据源的原始资料，浏览器根据此属性值决定将它们视为纯文本亦或是HTML标签。div元素仅接收HTML格式的数据。注意，只能在IE 5/Mac操作系统下实现文本文件数据源的绑定。

HTML 4中依然保留此属性，但它已被XHTML 1.0删除。

示例

值 IE浏览器可识别两种有效的设定值：text和html。

默认值 text

对象模型引用方式 [window.]document.getElementById(*elementID*).dataFormatAs

datasrc

IE 4 NN n/a Moz n/a Saf n/a Op n/a HTML 4

datasrc="数据源名称" 可选

为实现IE数据绑定功能，此属性用于指定页面内object元素的ID值，以便在进行远程数据访问时加载数据源对象。通过datafld属性指定来自于数据源中的绑定内容。注意，只能在IE 5/Mac操作系统下实现文本文件数据源的绑定。

HTML 4中依然保留此属性，但它已被XHTML 1.0删除。

示例

值 区分大小写的标识符。

默认值 无。

对象模型引用方式 [window.]document.getElementById(*elementID*).dataSrc

nowrap

IE 4 NN n/a Moz n/a Saf n/a Op 7 HTML n/a

nowrap 可选

nowrap属性使div元素的普通块样式失效。设置此属性后，除非被其他元素打断，否则文本内容在显示时不会自动换行。随意使用此属性可能会使页面过宽，从而导致用户长时间地拖动水平滚动条。

示例 `<div id="bigBlock" nowrap>...</div>`

值 如果此属性已出现，就意味着其属性值为true。

默认值 `false`

<dl>

IE *all* NN *all* Moz *all* Saf *all* Op *7* HTML *all*

`<dl>...</dl>` HTML 结束标签：必选

dl元素是dl、dt和dd 3元素小组的成员，这3个元素可用来在文档中定义一个列表。dl元素的标签首先将整个列表都括起来，然后使用前导的dt元素标签指明一个定义项，最后用前导的dd元素标签指出该项目的描述内容。下面这段代码扼要地定义了一个包含3个条目的定义列表序列：

```
<dl>
    <dt>Term 1</dt>
    <dd>Definition 1</dd>
    <dt>Term 2</dt>
    <dd>Definition 2</dd>
    <dt>Term 3</dt>
    <dd>Definition 3</dd>
</dl>
```

但对整个列表而言，必须使用结束标签来完成dl元素的封装。值得注意的是，dl元素是整个列表的容器，这意味着内嵌的的dt和dd元素均会继承dl元素中指定的样式单规则。可以在dt和dd元素中取消那些不必要的继承样式。

尽管HTML规范并没有强制规定定义列表的显示方式，但在主流浏览器中，均使用左对齐来显示dt元素，并缩进显示任何紧随其后的dd元素。虽然目前浏览器尚未为此添加任何的特殊字体格式或可视化元素，但可以根据喜好自由指定显示样式。如果想要存储多个条目或描述，可以在源代码中放置多个dt或dd元素。

HTML一直作为上下文相关的标记语言而不断发展，那么就要力图避免在使用中仅将定义列表作为一种格式化的手段（获得文本缩进）。可以使用样式单或可调节的边距设定来完成格式化任务。

示例

```
<dl>
    <dt>Z-scale</dt>
    <dd>A railroad modeling scale of 1:220. The smallest mass-produced
    commercial model scale.</dd>
</dl>
```

对象模型引用方式 `[window.]document.getElementById(elementID)`

元素专有属性 `compact`

元素专有事件处理属性 无。

compact

IE *3* NN *3* Moz *n/a* Saf *n/a* Op *n/a* HTML *3.2*

`compact` 可选

当设置compact值为true时（即它出现在dl元素的标签中），它命令浏览器在空间允许的情况下在同一行显示相关的dt和dd元素对。此空间（在IE和Navigator浏览器中均可确定）的判断准则与分配给dd元素的缩进量（不同操作系统，缩进量略有不同）有关。如果开启compact属性，那么当dt元素比缩进量要小时，dd元素就会提升到dt元素所在行，并显示在dt元素之后。由于均衡字体的字符宽度变化繁多，因此无法通过一个计算dt元素所包含的字符数量，使得dd元素能够顺利地在同一行显示。但这种紧凑的格式完全可以用于仅包含少量几个字符的dt元素。HTML 4并不赞成使用此属性，而严格的HTML 4或XHTML DTD甚至已经不再支持这个元素。

示例 <dl compact>*ListItems*</dl>

值 出现属性名称即意味着开启此功能。

默认值 Off

对象模型引用方式 [window.]document.getElementById(*elementID*).compact

<!DOCTYPE>

IE *all* NN *all* Moz *all* Saf *all* Op *7* HTML *3.2*

<!DOCTYPE...> HTML结束标签：禁用

DOCTYPE并不是一个HTML元素，而是标准通用标记语言（Standard Generalized Markup Language，简称SGML）格式中的一个注解（与HTML中<!-- ... -->形式的注解相同）。此元素必须是文档中的第1个元素，而且必须位于<html>标签之前。它会告知浏览器该文档中HTML源代码在设计时所遵循的文档类型定义（document type definition，简称DTD）。所有的浏览器都拥有一个默认的文档类型，它定义了浏览器会支持哪些元素和元素属性（它们都获得了浏览器的内部支持，否则就会出现漏洞）。为一个文档指定一个更新的DTD并不能促使一个旧浏览器支持那些未曾编写成代码的元素和属性。相反地，指定一个严格的DTD不会阻止浏览器识别并支持向后兼容性或其专有的元素和属性。

DOCTYPE元素总包含几个未分类的属性值，通过属性值就可以获知指定了哪些细节，如最外层的文档标签（一般是html）、DTD的负责组织、实际DTD文件（称为系统标识符）的地址、定义的明语名称（如果需要，还可以加上版本号），等等。例如，下面这个DOCTYPE就涉及一个包含所有不赞成使用的元素和属性的HTML 4.01 DTD：

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
      "http://www.w3.org/TR/html401/loose.dtd">
```

下一个例子则指向XHTML 1.1 DTD，它并不包含不推荐使用的条目或框架：

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.1//EN"
      "http://www.w3.org/TR/xhtml11/DTD/xhtml11.dtd">
```

此外，如果要指定一个XHTML DTD，还须要在DOCTYPE 声明之前包含以下任一个SGML处理指令标签。

```
<?xml version="1.0"?>
<?xml version="1.0" encoding="UTF-8"?>
```

第二个标签中还包含一个字符集编码设置，也可以在<meta>标签中定义这部分内容。W3C HTML和XHTML校验程序建议在文档中声明其字符编码方式。

在大多数情况下，网页作者都会引入一个DOCTYPE元素，以便在发布至网络前促进HTML源代码的有效性。但是，就算基于相同的DOCTYPE注释细节，一些现代浏览器的处理也略有不同。根据DOCTYPE属性值的细节信息，IE 6及后续版本、Mozilla和部分其他浏览器会决定在两种“模式”中的哪一种下运行。一种模式主要针对后向兼容性，它面向一些以前的实现，也偏离了W3C标准；而另一种模式则驱使浏览器的行为更遵守W3C的推荐标准。这两种模式之间的区别主要集中在细致的布局细节，目前的CSS和DOM规范更谨慎地定义了这一内容。对于简单的布局，也许无法看出这两种模式之间的差别。一旦页面依赖表格的样式单或背景、表单控件对齐（特别是当它们在表格中时）、精确的字体大小或字符间距控制，以及遵循文档边界和既有元素大小的像素级精确CSS定位，那么你就必须注意文档内的DOCTYPE细节。

尽管很难顾及每个兼容性的细节，但有两个主要的建议能够保证相关工作一切正常。首先，如果乐于使用当前页面或模板的布局，那么最好继续使用能够保持后向兼容性的模式。Mozilla的工程师称它为“quirks”模式，微软则没有提出任何特殊的名称。但是，一旦要使用新的内容，特别是针对更新的浏览器，那么最好向

“严谨（strict）”（Mozilla）或“标准兼容（standards-compatible）”（IE）模式靠拢。

目前常用的DOCTYPE属性值多得惊人，而且并不能在浏览器间完全同步使用控制规则，以决定哪个属性能够强制浏览器进入某个特殊模式。这里列举了几个更常用的DOCTYPE标签，它们均可迫使各种现代浏览器进入向后兼容性模式：

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 3.2 Final//EN">
<!DOCTYPE HTML PUBLIC "-//IETF//DTD HTML 3.0//EN">
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">
```

还有另外两个值得注意的地方。首先，以上这些例子中声明的HTML DTD均不晚于HTML 4.01，也都不是XHTML。其次，这些例子中均未包含指向参考*dtd*文件的系统标识符地址。另外，如果完全忽略了DOCTYPE元素，那么浏览器会提供等价的旧版内置DTD。

注意：IE 6有一个缺陷，如果指定一个XHTML的DOCTYPE声明，并且在文档开头放置<? xml ... ?>声明，就会导致浏览器回到quirks模式。IE 7已经修正了这个问题。

这里有一些常用的DOCTYPE标签，它们均可迫使现代浏览器进入现代的、基于标准的模式：

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0 Transitional//EN"
       "http://www.w3.org/TR/REC-html40/loose.dtd">
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0 Frameset//EN"
       "http://www.w3.org/TR/REC-html40/frameset.dtd">
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0//EN"
       "http://www.w3.org/TR/REC-html40/strict.dtd">
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
       "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Frameset//EN"
       "http://www.w3.org/TR/xhtml1/DTD/xhtml1-frameset.dtd">
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
       "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.1//EN"
       "http://www.w3.org/TR/xhtml11/DTD/xhtml11.dtd">
```

无论是否包含URL，所有的HTML 4.x/strict和XHTML DTD均可开启标准兼容模式。包含HTML 4.x过渡标准和框架集DTD也会启动标准兼容模式。

在最新的浏览器中，有关DTD的不同选择对DOM和CSS的影响请参见第2章（“关于client-与offset-属性集”一节、body对象及document、compatMode属性）和第3章（height和width属性）。附录E列举了当前最流行的HTML 4.01和XHTML 1.0 DTD支持哪些HTML 4元素和属性。

对象模型引用方式 [window.]document.firstChild

元素专有属性 属性值未分类。

元素专有事件处理属性 无。

<dt>

IE *all* NN *all* Moz *all* Saf *all* Op *7* HTML *all*

```
<dt>...</dt>
```

HTML结束标签：可选

dt元素是dl、dt和dd 3元素小组的成员，这3个元素可用来在文档中定义一个列表。dl元素的标签首先将整个列表都括起来，然后使用前导的dt元素标签指明一个定义项，最后用前导的dd元素标签指出该项目的描述内容。下面这段代码扼要地定义了一个包含3个条目的定义列表序列：

```
<dl>
    <dt>Term 1</dt>
    <dd>Definition 1</dd>
    <dt>Term 2</dt>
```

```
    <dd>Definition 2</dd>
    <dt>Term 3</dt>
    <dd>Definition 3</dd>
</dl>
```

dt元素是一个行内元素，而dd元素则可以包含块级内容，这些块级内容包括带花边的文字、图像和其他对象等。对dt和dd元素而言，在HTML中可以不使用结束标签，因为下一个起始标签实际上就标志着前一个元素的结束。但对整个列表而言，必须使用结束标签来完成dl元素的封装。

尽管HTML规范并没有强制规定定义列表的显示方式，但在主流浏览器中，均使用左对齐来显示dt元素，并缩进显示紧随其后的每个dd元素。虽然目前浏览器尚未为此添加任何特殊字体格式或可视化元素，但可以根据喜好自由指定显示样式。如果想要存储多个条目或描述，可以在源代码中放置多个dt或dd元素。

由于HTML一直作为上下文相关的标记语言而不断发展，因此要力图避免在使用中仅将定义列表作为一种格式化的手段（获得文本缩进）。如果只是为了排版，那么可以使用样式单或可调节的边距设定来完成格式化任务。

在Navigator 4中，任何通过class、id或style属性指派给dt和dd元素的样式都无法工作。如果希望为dt和dd元素都赋以同样的样式特性，那么也请为dl元素指定该样式；另外可以将全部dt和dd元素均放置在span元素之中，这样内嵌的dt和dd元素就会继承span元素的样式。对那些不会在NN 4中显示的文档而言，虽然完全没有必要这样做，但这种做法在其他浏览器中也是奏效的。

示例

```
<dl>
    <dt>Z-scale</dt>
    <dd>A railroad modeling scale of 1:220. The smallest mass-produced
    commercial model scale.</dd>
</dl>
```

对象模型引用方式 [window.]document.getElementById(*elementID*)

元素专有属性 无。

元素专有事件处理属性 无。

IE *all* NN *all* Moz *all* Saf *all* Op *7* HTML *all*

```
<em>...</em>
```

HTML 结束标签：必选

在HTML 4推荐标准中，有一类元素统称为**短语元素**（phrase element），em元素就是其中之一。这些元素为文档的特定部分指明了结构上的含义。em元素会与文本的显示效果完全不同，以便指出需要强调的内容。

浏览器可以自由决定如何（或是否）从其他body元素中区分出em元素的内容。Navigator和IE均使用斜体字的方式突出em元素的内容。可以使用任何合适的样式单来替换默认的显示形式。

示例

```
<p>The night was dark, and the river's churning waters were <em>very</em> cold.</p>
```

对象模型引用方式 [window.]document.getElementById(*elementID*)

元素专有属性 无。

元素专有事件处理属性 无。

<embed>

IE *3* NN *2* Moz *all* Saf *all* Op *7* HTML *n/a*

```
<embed>...</embed>
```

HTML 结束标签：可选/必选

在embed元素中可以加载一些非浏览器自带的内容，如多媒体和其他文件类型等。这些外部数据往往需要插

件或辅助程序以正确地加载并显示文件内容。尽管从Navigator 2和IE 3开始，这些浏览器就开始支持这个元素，但HTML标准表从未吸纳过它。而且HTML 4推荐标准建议在浏览器中使用object来加载那些一直由embed元素处理的外部数据。Navigator 4和IE 4浏览器及它们的后续版本也都支持object元素。一般说来，只要所依托的浏览器能支持object，最好就使用这个它来处理嵌入式元素。

虽然一直都将object元素作为标准化的目标，但并非所有浏览器都能在各种操作系统中支持这个元素。例如，Mozilla就更倾向于使用embed元素，而object则得到IE的青睐。为了适应这种差异，不妨同时指定这两种元素，这样浏览器就可以从中选择一个。例如，下面这段代码演示了如何在一个已安装Flash插件的浏览器中载入一段Flash动画：

```
<object width="425" height="350">
   <param name="movie" value="http://www.example.com/vid/2bq"></param>
   <embed src="http://www.example.com/vid/2bq"
   type="application/x-shockwave-flash" width="425"
   height="350"></embed>
</object>
```


须要牢记的是，浏览器会根据数据类型启动对应的插件，而针对同种数据类型的控制面板也会有不同的外观，这与浏览器、操作系统和用户为这种数据类型所安装的插件都有关系。虽然可以将插件的控制面板和环绕文字或其他元素谨慎地整合在一起，但这是一件很危险的事情。

尽管embed元素拥有为数众多的属性，但依然要特别关注每个属性的浏览器兼容性。由于IE/Windows、IE/Mac和Navigator中的插件技术并不完全一致，因此对应的属性设置也不尽相同。尽管如此，如果嵌入的元素并不依赖于某个并未被浏览器支持的属性设置，那么文档中包含的embed元素就可以在这些浏览器品牌下正常工作。然而，有些插件须要请求或接收一些并未包含在此元素中的属性名称/值对。不过，至少在Navigator和Mozilla浏览器中，所有的属性（包括那些往往被浏览器忽略的属性）和其属性值都可传入插件。因此，必须查阅插件的帮助文档，以确定是否需要额外的属性。如前文所示，可以为object元素定制并添加任意数量的param元素，从而成功地避免这种特定的属性问题。

示例　　`<embed name="jukebox" src="jazz.aif" height="100" width="200"></embed>`

对象模型引用方式

```
[window.]document.embeds[elementName]
[window.]document.getElementById(elementID)
```

元素专有属性　　align、alt、height、hidden、name、pluginspage、pluginurl、src、type、units、width

元素专有事件处理属性　　无。

align

IE 4　NN n/a　Moz all　Saf all　Op 7　HTML n/a


```
align="对齐位置"
```

可选

如果内嵌对象或播放器的控制面板会在页面上占据一定的空间，那么align属性就会决定该对象的显示位置与临近元素的最外层容器之间的关系。如果还有文本位于embed元素起始和结束标签之间，那么align属性还会影响这些文本在对象矩形空间内的显示方式。

本章前文所述的有关对齐常量的大部分规则，也适用于embed元素。由于HTML页面的作者往往无法掌控插件的控制面板，因此很难控制页面上的精确布局。在这种情况下，适用于某个控制面板的元素尺寸总是无法应用于另一个面板。

HTML 4并不赞成使用align属性，而推荐使用样式单的align属性。但是，如果因后向兼容性而使用embed元素，那么可以继续使用align属性。

示例 `<embed src="jazz.aif" align="left" height="100" width="200"></embed>`

值

每个浏览器均为此属性定义了不同的属性值集合。可以根据实际工作中的部署环境从下表中选择合适的属性值：

值	IE	其他	值	IE	其他
absbottom	·	—	middle	·	·
absmiddle	·	—	right	·	·
baseline	·	—	texttop	·	—
bottom	·	·	top	·	·
left	·	·			

默认值 bottom

对象模型引用方式

```
[window.]document.embeds[elementName].align
[window.]document.getElementById(elementID).align
```

alt

IE *4* NN *n/a* Moz *n/a* Saf *n/a* Op *n/a* HTML *n/a*

`alt="文本信息"` 可选

如果IE浏览器无法加载并播放外部多媒体文件，那么`alt`属性指定的文本信息就可以在`embed`元素应该出现的位置显示出来。在无法加载数据的情况下，这个文本信息会报告页面访问者他错过了什么内容。当然，IE浏览器还会显示一个对话框，提示用户如何从网络获取插件的相关信息。

请谨慎使用`alt`属性。如果这个外部数据并不是页面内的关键内容，那么当浏览器无法显示这些内容时，可能并不须要提醒用户缺少媒体控制器，而直接载入剩余的页面内容。在这种情况下，提醒信息可能比缺少媒体播放器更容易干扰用户。

在Navigator浏览器内，有另一个类似的可用元素——`noembed`。

示例 `<embed src="jazz.aif" alt="Sound media player" height="10" width="20"></embed>`

值 使用引号括起来的任意字符串。

默认值 无。

height, width

IE *3* NN *2* Moz *all* Saf *all* Op *7* HTML *n/a*

`height="长度值" width="长度值"` 必选

通过`height`和`width`属性，可以控制一个内嵌对象（或一个插件的控制面板）在页面内所占据的空间大小。在一些浏览器版本中可以不设定这两个属性，此时就由插件自身界面的高和宽决定它在文档中可见矩形框的大小。无论什么情况，最好为每个插件控制面板指定确切的尺寸大小。这是由于控制面板在不同浏览器中的大小不同，甚至在同一个浏览器内，不同插件的控制面板大小也不相同。有时，如果在页面上未能为控制面板提供足够的空间，控制面板可能无法显示。但如果为控制面板指定的空间过大，浏览器在页面上则会保留剩余的空白空间，这也会干扰页面的预期设计构想。

示例 `<embed src="jazz.aif" height="150" width="250"></embed>`

值 由引号括起来的正整数或百分比。虽然不能通过设定其值为0来完全隐藏一个内嵌对象的控制面板，但至少可以让它的长宽均减少到只占据一个像素值。可以通过DHTML设置其`display`属性为“`none`”来隐藏一个插件。

默认值 无。

对象模型引用方式

```
[window.]document.embeds[i].height
[window.]document.getElementById(elementID).height
[window.]document.embeds[i].width
[window.]document.getElementById(elementID).width
```

90

hidden

IE 4 NN n/a Moz 0.9 Saf n/a Op 7 HTML n/a

```
hidden="true" | "false"
```

可选

hidden属性是一个控制开关，通过它可以设定是否允许在屏幕上显示内嵌数据的插件控制面板。如果设置了hidden属性，那么height和width属性将失去作用。

示例 `<embed src="soothing.aif" hidden="true"></embed>`

值 True | false

默认值 false

对象模型引用方式

```
[window.]document.embeds[i].hidden
[window.]document.getElementById(elementID).hidden
```


name

IE 3 NN 2 Moz all Saf all Op 7 HTML n/a

```
name="元素标识符"
```

可选

如果正在使用脚本调用一个插件（尤其是在Navigator浏览器中通过LiveConnect技术调用插件时），那么为该内嵌元素指定一个独一无二的name属性值将有助于引用此元素。这样，如果一旦对页面进行了编辑，并移动或删除了页面上的多个内嵌元素，也不用担心如何将索引值与数组形式的对象引用对应起来（document.embeds[embedName]）。如果是支持W3C DOM的浏览器，那么可以使用id属性和对应的document.getElementById()方法。

示例 `<embed name="jukebox" id="jukebox" src="jazz.aif" height="15" width="25"></embed>`

值 区分大小写的标识符。

默认值 无。

对象模型引用方式

```
[window.]document.embeds[i].name
[window.]document.getElementById(elementID).name
```

pluginspage

IE 4 NN 4 Moz all Saf all Op 7 HTML n/a

```
pluginspage="URL"
```

可选

如果浏览器中现有的插件或辅助程序并不支持embed元素内src属性所指定的数据文件的MIME类型，那么pluginspage属性就可为相关插件的下载和安装提供URL链接。而Mozilla浏览器则一直尝试使用它自己的插件记录作为下载来源。

示例

```
<embed name="jukebox" src="jazz.aif" height="150" width="250"
pluginspage="http://www.example.com/plugin/install/index.html">
</embed>
```

91

值 任何有效的URL。

默认值 无。

pluginurl

IE *n/a* NN |4| Moz *n/a* Saf *n/a* Op *n/a* HTML *n/a*

```
pluginurl="URL"
```

可选

仅有Navigator 4浏览器引入了一种称为智能升级的功能，它允许在一定程度上为浏览器组件进行自动安装。如果用户浏览器尚未安装必要的插件以播放embed元素的数据类型，那么pluginurl就能够指定一个Java归档文件（JAR），这个JAR中含有自动安装时所需的插件和安全检测所需的数字签名对象。获取相关的JAR文件后，由Netscape的Java安装管理器完成安装过程。在Netscape公司的JAR文件打包工具（JAR Packager）的帮助下，可以创建包含数字签名并已压缩的JAR文件，其压缩方式和*zip*文件非常类似。

可以在embed元素标签中同时包含pluginspage和 pluginurl两个属性，以应对不同浏览器版本的要求。例如，Navigator 2和3响应pluginspage属性，而Navigator 4则优先响应pluginurl。

示例

```
<embed name="jukebox" src="jazz.aif" height="150" width="250"
pluginurl="http://www.example.com/plugin/install.jar"></embed>
```

值 任何指向JAR文件的有效URL。

默认值 无。

src

IE *3* NN *2* Moz *all* Saf *all* Op *7* HTML *n/a*

```
src="URL"
```

可选

src属性指定了一个文件的URL地址，插件将会播放该文件中的数据内容。在大多数情况下，使用ebmed元素时此属性是必需的，但也有少数例外。请参见type属性。浏览器曾经主要依靠文件扩展名来决定加载哪个插件（根据浏览器针对插件和辅助程序的选项设置），而现在往往根据文件内容类型来做出抉择。

示例 `<embed name="babyClip" src="Ugachaka.avi" height="150" width="250"></embed>`

值 完整或相对URL路径。

默认值 无。

对象模型引用方式

```
[window.]document.embeds[i].src
[window.]document.getElementById(elementID).src
```

type

IE *n/a* NN *2* Moz *all* Saf *all* Op *7* HTML *n/a*

```
type="MIME 类型"
```

可选

早期的Navigator浏览器期望插件并不需要任何外部数据文件。那么这种插件就更接近于一个小程序（applet）。如果在文档中放置这样的一个插件，仍然使用embed元素，但仅须要指定其MIME类型而不须要在src属性中设置数据文件的URL地址。当然，这里假设MIME类型具有一种特殊的性质，即在浏览器的设定中只有一种合适的插件对应于该MIME类型。然而，现今的浏览器会根据type属性决定使用哪个插件播放外部媒体文件。因此，在embed元素标签中须要同时指定src和type属性。

示例

```
<embed src="hooha.fbz" type="application/x-frobnitz" height="150" width="250"> </embed>
```

值 任何一个有效的MIME类型名称字符串，它包含类型和子类型，两者之间使用正斜杠（/）分隔。整个字符串要使用引号括起来。可从此处获取MIME类型目录：*http://www.iana.org/assignments/media-types/*。

默认值 无。

units

IE 3 NN n/a Moz n/a Saf n/a Op n/a HTML n/a

`units="度量单位类型"` 可选

units属性用于指定元素中height和width属性值的对应度量单位类型。IE浏览器会忽视此属性设置，并以像素作为度量单位。

示例 `<embed src="jazz.aif" height="150" width="250" units="en"></embed>`

值 在IE浏览器中，可用值为px或em。

默认值 px

对象模型引用方式

```
[window.]document.embeds[i].units
[window.]document.getElementById(elementID).units
```

width

参见height属性。

<fieldset>

IE 4 NN n/a Moz all Saf all Op 7 HTML 4

`<fieldset>...</fieldset>` HTML结束标签：必选

fieldset元素是表单元素的一种结构容器，以便与表单元素的功能限制有所区别。事实上，可以在一个form元素内定义多个fieldset元素，以便为表单元素的逻辑分组提供上下文关系。例如，一个fieldset元素可能用于包含姓名和地址信息的文本输入区，而另一个fieldset则可能专用于包含信用卡信息的文本输入区。

为了突出显示这个元素，浏览器会在每个fieldset容器的内部自动绘制一个隔线框，包围住内部的表单元素。在fieldset元素的起始标签之后，如果定义一个legend标签，就可以在隔线框内嵌入一个提示性的标注。默认情况下，这个方框的宽度与其相邻的最外层容器（一般是body或html元素）的宽度保持一致。因此，如果期望这个方框能够比较紧凑地围绕在可见的表单元素周围，那么必须设置fieldset的width样式单属性。

示例

```
<form method="POST" action="...">
<fieldset>
<legend>Credit Card Information</legend>
...inputElementsHere...
</fieldset>
</form>
```

对象模型引用方式 `[window.]document.getElementById(elementID)`

元素专有属性 align、form

元素专有事件处理属性 无。

align

IE 4 NN n/a Moz n/a Saf n/a Op n/a HTML n/a

`align="对齐位置"` 可选

align属性仅在IE浏览器中才能发挥作用，而且其具体的显示效果在不同的操作系统下大为不同。从理论上来说，此属性应该能够控制fieldset内包含的所有input元素的对齐方式。Macintosh系统中的IE浏览器的确遵循这一设定，但在Windows系统的IE（尤其是IE 6）浏览器中，此属性对fieldset元素几乎没有任何影响。因此不妨优先使用默认设置，须要进行修改时则利用样式单进行替换。

示例 `<fieldset align="right">...</fieldset>`

值 适用的属性值为：left | center | right。

默认值 left

94 **对象模型引用方式** [window.]document.getElementById(*elementID*).align

form

IE *n/a* NN *n/a* Moz *n/a* Saf *n/a* Op *9* HTML *n/a*

form="*formID [formID] ...*" 可选

通过这个Web Forms 2.0扩展，无论表单是否包含了fieldset，都可以将该fieldset元素与一个或多个表单联系在一起。

值 页面上一个或多个form元素的ID值。此时，请使用空格分隔多个ID值。

默认值 无。

IE *all* NN *all* Moz *all* Saf *all* Op *7* HTML *3.2*

... HTML结束标签：必选

font是容器元素，会根据这个元素属性定义的字体特征显示此元素内的内容。HTML 4已不再赞成使用此元素，而推荐使用样式单中其他可用的字体属性，当然，同样也可以在其他元素中使用它们。另外，还可以通过专用的span容器来控制单行文本的字体。尽管如此，为了能够兼容那些为早期浏览器而设计的网页，此元素还能使用相当长的一段时间。

在Navigator 4的嵌套表格中，样式单继承时常出错。因此，通过在td元素中插入font来控制其内容，可以大大加强对样式设计的控制力。进一步说，如果某个浏览器不允许执行CSS字体规则，那么font元素还可以作为最后的一根救命稻草。

在font元素的生命期内，成熟的浏览器在不断为它增加实用的新属性。Navigator 4就引入了一些专用属性，但为了实现跨浏览器的兼容性，这些属性逐渐被样式单所代替。

示例

对象模型引用方式 [window.]document.getElementById(*elementID*)

元素专有属性 color、face、point-size、size、weight

元素专有事件处理属性 无。

color

95 IE *3* NN *2* Moz *all* Saf *all* Op *7* HTML *3.2*

color="颜色三元组或名称" 可选

此属性为font元素包含的所有文本设置字体颜色。HTML 4不再赞成使用此属性，而推荐使用样式单。

示例 ...

值 一个十六进制的三元组或明语颜色名称。目前可用的明语颜色名称请参见附录A。

默认值 浏览器的默认值。

对象模型引用方式 [window.]document.getElementById(*elementID*).color

face

IE *3* NN *3* Moz *all* Saf *all* Op *7* HTML *4*

face="字符集名称 *1*[, ... 字符集名称 *N*]" 可选

通过此属性，可为font元素包含的文本指定可用的字符集层次。这字符集名称使用逗号进行分隔，浏览器从它们中的第一个开始，逐个与客户端系统中的字符集进行比对，以设置默认字体，如果最终仍然找不到对应字符

集就使用浏览器的默认字符集。比对时，字符集名称必须与系统中的字符集名称完全匹配。如果使用这个属性（而不是更好的font-family样式单属性），那么建议您将通用字体（如serif、sans-serif等）列在最后。

示例 `<font face="Bookman, Times Roman, serif">...</font>`

值 一个或多个字符集名称，请包含可识别的通用字体，如serif、sans-serif、cursive、fantasy和monospace。

默认值 浏览器的默认值。

对象模型引用方式 `[window.]document.getElementById(elementID).face`

point-size

IE *n/a* NN *|4|* Moz *n/a* Saf *n/a* Op *n/a* HTML *n/a*

`point-size=" 尺寸"` 可选

point-size是Navigator 4浏览器中特有的一个非CSS等价属性，通过它可以指定磅（point）值，从而设置字体尺寸。此时就不须要再通过size属性来指定字体尺寸。如果同时指定了point-size属性和font-size样式属性，那么优先使用样式属性。另外，如果要实现跨浏览器部署，那么建议使用样式单属性来精确控制字体的磅值或像素尺寸。

示例 `<font point-size="14">...</font>`

值 用以描述磅尺寸的一个正整数。

默认值 浏览器的默认值。

size

IE *all* NN *all* Moz *all* Saf *all* Op *7* HTML *3.2*

`size="字体相对大小"` 可选

size属性中引用的字体大小均为相对大小，它未同任何操作系统平台中的字体大小相关联。size属性值的可选范围是1–7的7个整数值，而浏览器的默认字体大小是3。但具体的字体大小会随着操作系统和浏览器的不同而发生变化。

虽然用户可以在选项里调整默认的字体大小,但使用size属性后，会取代默认设置。此外，size的属性值还与首选项中的具体字体大小设置有关。在首选项中通过“+”和“-”两个符号可以调整属性，使得浏览器的默认字体大小在1–7这个范围内增大或减小。

示例

```
<font size="4">...</font>
<font size="+3">...</font>
```

值 既可以是一个整数（可以不使用引号括起来），也可以是一个由“+”或“-”和整数组成的相对数值（必须用引号括起来）。

默认值 3

对象模型引用方式 `[window.]document.getElementById(elementID).size`

weight

IE *n/a* NN *|4|* Moz *n/a* Saf *n/a* Op *n/a* HTML *n/a*

`weight=" boldnessValue"` 可选

weight是Navigator 4浏览器中特有的一个非CSS等价属性，通过这个普通的属性就可以设置字体粗细，而不须要再通过其他样式单规则指定字体粗细。此属性有时并不可靠，因此最好使用font-weight这一CSS样式属性设置字体粗细。

值 起始值为100，每次可递增100，直到最大值900。

默认值 未知。

<form>

IE *all* NN *all* Moz *all* Saf *all* Op *7* HTML *all*

<form>...</form>

HTML 结束标签：必选

当网页表单访问者和服务器之间进行通信时，无论HTML表单有多么重要，form元素从本质上看都仅仅是一个控件容器。页面中往往通过input元素创建表单控件，当然也存在例外。在Navigator 4或更早期的版本中，即使input元素的用户交互并不是用于向服务器提交数据，这些input元素仍然必须放置在form元素之中才能显示。

虽然一个文档中可以容纳任意多个form元素，但客户端每次仅能通过其中的一个表单提交控件设置。因此，只有当控件分组能够彼此独立地进行提交时，才有必要将这些连续的表单控件分装到多个form元素之中。如果既想维持一个表单结构，又想为控件进行逻辑或结构分组，那么请使用fieldset元素来创建必要的控件分组。

一旦表单被提交到服务器，那么所有包含name属性的控件均会将其名称和值以名称/值对的形式传递到服务器，以便进行下一步的处理工作。某些情况下，也会通过浏览器的E-mail模块将它们以附件或消息的形式发送出去。运行在服务器端的通用网关接口（Common Gateway Interface，简称CGI）程序能接收并解析这些名称/值对，以便进行下一步的处理工作，例如，向服务器的数据中插入一条数据，或初始化一条关键字查询。通过action属性中指定的URL可以调用服务器端程序。

在浏览器内部，每个提交过程都包含几个明确的步骤。首先，浏览器从表单控件的名称/值对中整理出表单数据集合。名称来自于控件中name属性的属性值，而控件的值则取决于控件类型。例如，提交时文本框内的内容就是文本输入元素的值；对于单选按钮组内的一个单选按钮而言，其组内所有控件的name属性值均相同，而已被选中的按钮控件的value属性值会作为整个分组的值插入名称/值对之中。

W3C推荐标准建议，在名称/值对中使用表单控件的id属性值。但直到目前可用的最新型浏览器为止，依然只能将name属性作为名称/值对所需的标识符。

提交过程的第二步是对每个名称/值对中的文本进行编码。首先，使用“+”代替空格符号。然后，用转义符替换RFC 1738中定义的保留字符，而其他所有非字母或数字字符则会进行十六进制转换，例如*%HH*，此处的*HH*就是某个字符对应的十六进制ASCII值。最后，使用“=”分隔每个名称/值对组合中的名称和值，并用“&”分隔不同的名称/值对。

在提交的最后一步中，method属性值决定了如何向服务器传输已转义的表单数据集合。如果method属性值为“get”，那么表单数据集合会附加在action属性指定的URL之后，这两者之间使用“?”进行分隔。如果是“post”或是默认的“enctype”方法，那么数据集合就会以一种消息的形式传输至服务器端，当然这种方式不同与E-mail。如果通过“GET”方法提交数据，字符长度会受到限制，而使用“POST”方法时，不仅数据长度不受限制，浏览器的地址栏也不会显示表单的提交位置。

在现代浏览器中，可以在脚本的辅助下取消表单的提交过程，通常是通过“onsubmit”事件触发有效性检查或其他操作。在提交表单前就会激活这个事件。如果事件处理程序运算后“return false”，或默认的事件被取消（如，evt.preventDefault()），那么表单就不会被提交，并且用户可以继续编辑表单元素。

Web Forms 2.0规范（Opera 9首先实现）提供了内部验证机制，这种机制使得表单控件更为智能。例如，可以将一个正则表达式作为一个文本输入域的属性。其他详细信息，请参见本章中的input元素和第2章中的ValidityState对象。

示例

```
<form name="orders" method="POST" action="http://www.example.com/cgi-bin/order">...</form>
```

对象模型引用方式

```
[window.]document.forms[i]
[window.]document.forms[formName]
[window.]document.formName
[window.]document.getElementById(elementID)
```

元素专有属性 accept、accept-charset、acceptcharset、action、autocomplete、data、enctype、method、name、replace、target

元素专有事件处理属性

处理程序	IE	NN	其他	HTML
onreset	4	3	all	4
onsubmit	3	2	all	4

accept

IE *n/a* NN *n/a* Moz *n/a* Saf *n/a* Op *9* HTML *4*

```
accept="MIME类型列表"
```

可选

当input元素的类型为“file”时，才须要使用accept属性。此属性用于说明表单提交时哪些MIME类型的文件能够上传至服务器端。使用这个属性后，可以在文件对话框中过滤掉其他的文件类型，以便选择合适的待上传文件。从某种程度上说，此属性提供了客户端的文件类型检查功能，因此不会将那些不合规的MIME文件类型发送到服务器端。

示例 `<form accept="text/html, image/gif" ...>...</form>`

值 不区分大小写的MIME类型（内容类型）。如果使用多种类型，请在列表中使用逗号进行分隔。可从此处获取MIME类型目录：*http://www.iana.org/assignments/media-types/*。

默认值 无。

accept-charset,acceptcharset

IE *5* NN *n/a* Moz *all* Saf *all* Op *7* HTML *4*

```
accept-charset="字符集名称列表"
```

可选

客户端表单的字符集信息将通过accept-charset属性告知服务器，以便服务器为解析表单信息做准备。这种带连字符的属性名称来自于HTML 4规范，但IE 5及其后续版本还支持另外一种不带连字符的属性名称。

示例 `<form accept-charset="it, es" ...>...</form>`

值 是否区分大小写取决于字符集注册表。可以使用逗号分隔多个字符集。如果服务器端为客户端生成表单，那么就使用保留值“unknown”来假定表示其字符集。

默认值 unknown

action

IE *all* NN *all* Moz *all* Saf *all* Op *7* HTML *all*

```
action="URL"
```

可选

此属性指定表单提交时须要访问的URL。如果表单是提交到服务器端以便进行更进一步的处理，那么此URL可能指向一个CGI程序或一个包含服务器端脚本的HTML页面。这些脚本会首先在服务器端执行，然后生成的HTML页面才会下载到客户端。作为表单提交的结果，服务器会返回一个HTML页面，并在客户端显示。如果返回的结果须要投递到不同的框架或窗口，那么就必须指定相应的target属性。

通过E-mail实现表单提交（为action属性指定一个“mailto:”的URL）时，所有浏览器均无法非常稳定地完成提交的过程，这样可能会丢失很多用户所提交的表单信息。如果使用CGI处理表单信息超出了开发者的技术能力范围，不妨使用第三方提供的一种FormMail服务程序，它可以通过E-mail传递表单信息。

如果未设置action属性，那么表单提交后当前页面会直接重载，并且所有的表单元素都会恢复到它们的默认状态。

示例 `<form method="POST" action="http://www.example.com/orders/order.html">`

值 完整或相对URL路径。

默认值 无。

对象模型引用方式

```
[window.]document.forms[i].action
[window.]document.forms[formName].action
[window.]document.formName.action
[window.]document.getElementById(elementID).action
```

autocomplete

IE 5 NN n/a Moz n/a Saf n/a Op n/a HTML n/a

`autocomplete="功能开关"` 可选

如果IE用户在选项中允许“表单自动完成”这一功能，那么autocomplete属性就能控制整个表单的填写。IE会保存（通过一种加密的方式）过去填写过的文本和密码等条目，当用户再次访问时，浏览器就会列出与以前输入条目相符的内容，以便完成表单条目的填写。如果表单域所需的数据可能会保存在用户的电子名片（vCard）中，那么可以为“text”和“password”类型的input元素指定vcard-name属性，当浏览器找到了与其名称相符的用户选项条目时，就会提前或辅助完成该表单域的填写工作。关于HTML表单中如何实现“自动完成”的更多细节，请访问此链接：*http://msdn.microsoft.com/en-us/library/ms533032(VS.85).aspx*。

示例 `<form method="POST" action="register.pl" autocomplete="on">`

值 on | off。

默认值 off

对象模型引用方式

```
[window.]document.forms[i].autoComplete
[window.]document.forms[formName].autoComplete
[window.]document.formName.autoComplete
[window.]document.getElementById(elementID).autoComplete
```

data

IE n/a NN n/a Moz n/a Saf n/a Op n/a HTML n/a

`data="URI"` 可选

Web Forms 2.0扩展，它允许表单从一个外部XML文件中获取表单控件的初始值。Web Forms 2.0规范为使用XML文件提供了详细的结构和命名空间说明。如须获取更多信息请访问此链接：*http://www.whatwg.org*。

示例

```
<form name="registration" method="POST" action="register.pl" data="form/reg_default.xml">
```

值 统一资源标识符。

默认值 无。

enctype

IE all NN all Moz all Saf all Op 7 HTML all

`enctype="MIME 类型"` 可选

此属性设置了随同表单一起提交至服务器的数据的MIME类型。一般情况下（method属性值为“POST”时），其默认值就是当前内容的类型。如果使用了一个类型为“file”的input元素，那么enctype的属性值就应该指定为“multipart/form-data”。

示例

```
<form method="POST" enctype="text/plain" action="mailto:orders@example.com">...</form>
```

值 MIME类型（内容类型）。如果使用多种类型，请在列表中使用逗号进行分隔。

默认值 application/x-www-form-urlencoded

对象模型引用方式

```
[window.]document.forms[i].encoding
[window.]document.forms[formName].encoding
[window.]document.formName.encoding
[window.]document.getElementById(elementID).encoding
```


method

IE *all* NN *all* Moz *all* Saf *all* Op *7* HTML *all*

```
method="GET" | "POST"
```

可选

可以通过“GET”和“POST”这两种HTTP方法提交表单。这两种方法决定了表单元素的数据提交方式，使用“GET”方法时，这些数据会附加在action属性指定的URL之后，而使用“POST”方法时，则会作为事务消息的正文进行发送。事实上，如果在form元素中未设置action和method属性，那么表单提交时会直接重载同一个文档，并将表单控件都恢复到初始值。

示例 <form method="POST" action="http://www.example.com/orders/order.html">...</form>

值 GET或POST。大写或小写属性值均可获得服务器响应。Web Forms 2.0另外增加了两种属性值：put和delete。

默认值 GET

对象模型引用方式

```
[window.]document.forms[i].method
[window.]document.forms[formName].method
[window.]document.formName.method
[window.]document.getElementById(elementID).method
```

name

IE *3* NN *2* Moz *all* Saf *all* Op *7* HTML *4.01*

```
name="元素标识符"
```

可选

此属性为整个form元素指定了一个标识符。在为以往的浏览器编写脚本时，这个属性值特别有用，通过它可以引用表单或其内嵌控件。为了实现这一功能，较新的浏览器建议使用更好的id属性，但表单提交时依然需要name属性。

示例

```
<form name="orders" id="orders" method="POST"
action="http://www.example.com/cgi-bin/order">
...
</form>
```

值 区分大小写的标识符。

默认值 无。

对象模型引用方式

```
[window.]document.forms[i].name
[window.]document.forms[formName].name
[window.]document.formName.name
[window.]document.getElementById(elementID).name
```

replace

IE *n/a* NN *n/a* Moz *n/a* Saf *n/a* Op *9* HTML *n/a*

```
replace="类型"
```

可选

Web Forms 2.0扩展版引入了这个指令，它指明了在表单提交后如何处理服务器返回的数据。目前有两种处理

方式，默认方式是使用服务器的响应数据替换原文档。如果已为form元素的data属性指定了一个URL，那么浏览器会将返回的数值填充到表单中，而不是重新载入表单的初始化数据，这是第二种处理方式。

示例

```
<form name="registration" method="POST" action="register.pl"
data="form/reg_default.xml" replace="values">
```

值 document或values

默认值 document

target

IE *all* NN *all* Moz *all* Saf *all* Op *7* HTML *4.01*

target="窗口或框架名称" 可选

如果从服务器返回的文档并不准备在当前窗口或框架中载入，就可以通过设置target属性，使得对应的窗口或框架载入该文档。此时，框架或窗口名称必须在对应目标中指定为标识符。例如，可以通过frame元素的name属性指定框架名称，或通过脚本方法window.open()的第二个参数在创建新窗口时为窗口命名。如果疏忽了这个属性，那么返回的文档将会取代表单元素所在的文档。如果现有的框架或窗口中没有对应的目标名称，则浏览器会打开一个新的窗口以显示返回的文档。

如果该表单位于子窗口中，同时又期望在主窗口中打开返回的文档，那么必须首先使用脚本获取主窗口的name属性值，然后将这个值作为表单的target属性值。

由于框架和窗口已经超出了单纯的文档标记范畴，因此HTML 4和XHTML规范中的严格DTD并不支持在任何元素中使用target属性。事实上，在严格的环境下框架根本无法设置文档内容，这正是将DTD从HTML 4和XHTML中剥离出来的目的。如果文档必须支持这些严格的DTD，并且又期望使用target属性，不妨在页面载入后通过脚本设置链接、图片映射表和表单的属性。

此外，每个表单元素都只能获得一个返回的文档和一个载入目的地。如果期望通过提交一个表单就能改变多个框架的内容，那么就须要在返回的文档中包含脚本代码，通过其onload事件处理程序在不同的框架中加载或动态写入文档。同时将每个框架的location.href属性值设定为期望的URL值。

示例

```
<form method="POST" action="http://www.example.com/cgi-bin/order" target="new"> ... </form>
```

值

如果已通过目标元素的name属性指定了框架或窗口名称，就可以使用这个区分大小写的标识符。目前，有如下4个作为常量使用的专用目标名称。

_blank

浏览器为目标文档打开一个新窗口。

_parent

目标文档会取代当前框架的框架集文档，如果不存在，则依照“_self”执行。

_self

目标文档替代在链接所在窗口或框架里的现有文档。

_top

目标文档将会取代任何已加载的框架集窗口，从而占据整个浏览器。如果在窗口中不存在框架集，则依照“_self”执行。

默认值 _self

对象模型引用方式

```
[window.]document.forms[i].targe
[window.]document.forms[formName].target
[window.]document.formName.target
[window.]document.getElementById(elementID).target
```

<frame>

IE *3* NN *2* Moz *all* Saf *all* Op *7* HTML *4*

`<frame>` HTML结束标签：禁用

frame元素定义了一种独立的窗口空间，它通常是整个浏览器窗口中的一部分。frame元素必须定义在frameset元素之中。而frameset则用于定义一个框架组的行列排列方式。

浏览器会将浏览器程序窗口中的框架视为一个单独的浏览器窗口。这样一来，每个框架窗口都能够彼此独立地加载其页面内容。尽管frame元素中并没有必须设置的属性，但依然建议为它指定一个name属性，这样就可以调用这个框架显示其他链接或表单的目标文档或返回文档。为框架指定name属性也能使脚本编程受益匪浅，因为这一做法能够大大提高脚本代码的可读性，使得框架及其内容的调用过程一目了然。值得注意的是，在最新的W3C DTD中，frame元素仅仅在HTML 4.01过渡DTD中有效，而frameset的DTD在HTML 4.01和XHTML 1.0中均有效。请参见附录E。

示例

```
<frameset cols="150,*">
    <frame name="navbar" src="nav.html">
    <frame name="main" src="page1.html">
</frameset>
```

对象模型引用方式

```
[window.]frameName
[window.]frames[i]
[window.]document.getElementById(elementID)
```

元素专有属性 allowtransparency、bordercolor、datafld、datasrc、frameborder、height、longdesc、marginheight、marginwidth、name、noresize、scrolling、security、src、width

元素专有事件处理属性 无。

allowtransparency

IE *5.5* NN *n/a* Moz *n/a* Saf *n/a* Op *n/a* HTML *n/a*

`allowtransparency="功能开关"` 可选

使用allowtransparency属性后，可使框架的背景变为透明。此元素更适用于iframe元素。详细内容请参见iframe元素。

bordercolor

IE *4* NN *3* Moz *all* Saf *n/a* Op *n/a* HTML *n/a*

`bordercolor="颜色三元组或颜色名"` 可选

如果既希望框架集显示边框，又想使其内某框架子集的边框颜色与其他框架子集的不同，就可以通过bordercolor属性为每个frame元素单独指定边框颜色。由于每个浏览器和操作系统的渲染技术各不相同，因此在框架集中使用不同的边框颜色可能会带来问题。与此同时，边框的精确宽度和位置以及其颜色均取决于不同的浏览器版本和操作系统。因此，如果为框架集中的部分框架设置特殊的颜色，最好尽可能在不同的浏览器版本和操作系统下进行多次测试，以便更好地评估框架颜色的视觉效果。如果为单独的一个框架边框设置颜色，在IE 6或IE 7中的显示结果无法预测。

示例

```
<frame name="navbar" src="nav.html" bordercolor="salmon">
```

值 一个十六进制的三元组或明语颜色名称。目前可用的明语颜色名称请参见附录A。

默认值 无。

对象模型引用方式 [window.]document.getElementById(*elementID*).borderColor

105

datafld

IE *4* NN *n/a* Moz *n/a* Saf *n/a* Op *n/a* HTML *n/a*

```
datafld="列名称"
```

可选

此属性用于实现IE浏览器中的数据绑定功能，可以使用远程数据源中的列名代替frame元素中的src属性。数据来源列必须包含一个有效的URI地址（无论是相对还是绝对地址）。此时，元素中还必须设定datasrc属性。注意，只能在IE 5/Mac操作系统下实现文本文件数据源的绑定。

示例 <frame datasrc="DBSRC3" datafld="newsURL">

值 区分大小写的标识符。

默认值 无。

对象模型引用方式 [window.]document.getElementById(*elementID*).dataFld

datasrc

IE *4* NN *n/a* Moz *n/a* Saf *n/a* Op *n/a* HTML *n/a*

```
datasrc="数据源名称"
```

可选

为实现IE数据绑定功能，此属性用于指定页面内object元素的ID值，以便在进行远程数据访问时加载数据源对象。此时将通过datafld属性指定来自于数据源中的绑定内容。注意，只能在IE 5/Mac操作系统下实现文本文件数据源的绑定。

示例 <frame datasrc="DBSRC3" datafld="newsURL">

值 区分大小写的标识符。

默认值 无。

对象模型引用方式 [window.]document.getElementById(*elementID*).dataSrc

frameborder

IE *3* NN *3* Moz *all* Saf *all* Op *8* HTML *4*

```
frameborder="功能开关"
```

可选

此属性用于控制是否显示框架集内的一个单独框架的边框。它会覆盖其frameset元素容器中的frameborder属性设置。在大部分浏览器和操作系统中，仅控制一个单独框架的边框可能会带来问题。如果周边其他框架均显示边框，那么仅仅隐藏一个框架的可能毫无作用。虽然可以大胆试验隐藏和显示边框的不同效果，但一定要保证访问者通过其操作系统和浏览器浏览页面时的显示效果。因此，使用整个frameset元素的frameborder属性来控制边框的隐藏和显示往往更为简便。

示例 <frame name="navbar" src="nav.html" frameborder="0">

值 根据使用场合的不同，适用于此属性的开关值也不相同。HTML 4指定1为开，0为关，而Navigator 3和4则使用“yes”和“no”表示开关。IE 4及后续版本和某些其他浏览器则支持以上两种类型的属性值。

106

默认值 1

对象模型引用方式 [window.]document.getElementById(elementID).frameBorder

height, width

IE *4* NN *n/a* Moz *n/a* Saf *n/a* Op *n/a* HTML *n/a*

```
height="像素值" width="像素值"
```

可选

根据微软公司发布的HTML文档，IE浏览器中height和width属性可以控制框架的尺寸。但实际上，这些属

性并不能直接控制框架集中框架的外观。与之相反，须要通过frameset的cols和rows属性管理框架最初的几何外形。因此，不要使用这些属性。

longdesc

IE 6 NN n/a Moz all Saf all Op n/a HTML 4

```
longdesc="URL"
```

可选

此属性可指定一个文档的URL地址，该文档中的内容比元素的title属性值更丰富，因此可以更细致地描述该元素。在未来的浏览器中，此属性可以作为元素的一个注释信息，以便那些无法看到屏幕的用户也能获知相关信息。Mozilla中，通过在上下文菜单中设置这个元素的特性，可以为此属性指定的URL显示一个包含链接的小窗口。目前主流的浏览器尚未为该属性提供更多的功能。

示例 `<frame longdesc="navDesc.html" title="Navigation Bar" src="navbar.html">`

值 任何有效的完整或相对URL地址。

默认值 无。

对象模型引用方式 `[window.]document.getElementById(elementID).longDesc`

marginheight, marginwidth

IE 3 NN n/a Moz all Saf all Op 7 HTML 4

```
marginheight="像素值" marginwidth="像素值"
```

可选

这两个属性值用于表示框架的内边框与显示内容之间空白距离的像素值。marginheight属性控制框架的上下边距高度，而marginwidth则控制左右边距宽度。HTML 4规范将它们的默认值留给浏览器来处理。

如果没有任何其他提示，浏览器会在框架内自动插入14个（IE/Windows）或8个像素（其他）的空白空间。但是，一定要注意的是，如果想尝试改变默认状态，改变其中的任何一个可能会导致另一个值变为0。因此，最好同时设置这两个值，以防止框架中的内容完全充满整个框架空间。

示例 `<frame src="navbar.html" marginheight="20" marginwidth="14">`

值 任何大于或等于0的整数。

默认值 14（IE/Windows）；8（其他）。

对象模型引用方式

```
[window.]document.getElementById(elementID).marginHeight
[window.]document.getElementById(elementID).marginWidth
```

name

IE 3 NN 2 Moz all Saf all Op 7 HTML 4

```
name="元素标识符"
```

可选

如果某框架中的链接或表单必须在其他框架中加载目标文档或返回的文档，那么就可以在这些元素中设置target属性，以指定用于接收新内容的框架。为了将这些内容导向一个框架，该框架必须为其name属性指定属性值。该属性值与a或form元素中target的属性值完全相同。此外，客户端脚本还可以利用框架的name值引用其他框架或其他框架中的内容。因此，为所有的框架都指定一个唯一的标识符名称是一个很好的编程习惯。

XHTML不再推荐使用此属性。为了通过Frameset XHTML DTD验证，可以为元素的name和id属性指定相同的标识符。

示例 `<frame name="navbar" id="navbar" src="nav.html">`

值 区分大小写的标识符。

默认值 无。

对象模型引用方式

```
[window.]frameName.name
```

<frame>

```
[window.]frames[i].name
[window.]document.getElementById(elementID).name
```

noresize

IE 3 NN 2 Moz all Saf all Op 7 HTML 4

```
noresize
```

可选

当用户沿着框架边框的垂线拖动边框时，框架尺寸会随之而改变。如果在元素中出现noresize属性，那么浏览器就会防止用户手动改变框架边缘的位置。这时，受影响的框架元素的所有边缘都无法移动，这意味着伸展到框架集中其他框架的边缘也无法移动。

示例 `<frame src="navbar.html" noresize>`

值 一旦在HTML代码中出现此属性，就无法改变框架尺寸。

默认值 默认情况下，框架均可改变尺寸。

对象模型引用方式 `[window.]document.getElementById(elementID).noResize`

scrolling

IE 3 NN 2 Moz all Saf all Op 7 HTML 4

```
scrolling="auto" | "no" | "yes"
```

可选

当框架中加载的内容超出其可视区域时，浏览器会默认添加水平和/或竖直滚动条。如果设计框架时并不期望有滚动条出现，那么出现的滚动条就会占据框架内容的部分空间，从而影响到部分内容的布局。同时，由于在不同浏览器和操作系统中，字体的默认尺寸各不相同，因此在不同的客户端中，同样的文本内容也会有不同的显示效果。如果须要阻止框架中出现滚动条，可以将scrolling设置为no；而如果希望框架总是包含滚动条，则须要将其设置为yes。在后一种情况中，如果内容显示时并不须要滚动，那么滚动条会被禁用。在一些老版本的Navigator浏览器中，就算框架中随后加载的内容并不需要滚动条，它们依然会自动显示。目前的这种情形已有所好转，在Navigator 4和其他主流浏览器中，只有需要时滚动条才会出现。

在将scrolling属性设置为no之前，应该进行详尽的测试，以保证在所有浏览器和平台下，关键内容都能够在框架中正常显示。如果既不允许框架滚动，又设置了noresize属性，那么部分用户可能无法看到框架中的全部内容。

示例 `<frame src="navbar.html" scrolling="no">`

值 常量：auto | no | yes。

默认值 auto

对象模型引用方式 `[window.]document.getElementById(elementID).scrolling`

security

IE 6 NN n/a Moz n/a Saf n/a Op n/a HTML n/a

```
security="restricted"
```

可选

如果激活此属性，那么就会将该框架的安全级别提高，达到Windows安全选项设置中的“受限”级别。这样一来，框架内容就有可能无法执行任何脚本。

示例 `<frame src="navbar.html" security="restricted">`

值 常量：restricted。

默认值 无。

src

IE 3 NN 2 Moz all Saf all Op 7 HTML 4

```
src="URL"
```

可选

此属性定义了一个URL，frame元素将要加载此链接中的内容。它既可以是绝对URL，也可以是一个包含框

架集说明的相对URL文档路径。也可以在框架中使用由脚本返回的javascript伪URL属性值。例如，当框架集载入时期望某个框架保持空白，那么就可以在框架集文档中定义方法，用以返回一个空白的HTML页面。从子框架的角度看，可以调用下面这个方法来为每个显示空白的框架设置src属性：

```
<html>
<script language="JavaScript">
function blank( ) {
   return "<html></html>"
}
</script>
<frameset cols="50%,50%">
   <frame name="leftFrame" src="javascript:parent.blank( )">
   <frame name="rightFrame" src="javascript:parent.blank( )">
</frameset>
</html>
```

在某些浏览器中还可以使用“about:blank”来达到显示空白页面的目的，这时会绘制一个内建空白页面。

示例 `<frame src="navbar.html">`

值 完整或相对URL，或者javascript伪URL。

默认值 无。

对象模型引用方式 `[window.]document.getElementById(elementID).src`

width

参见height属性。

<frameset>

IE 3 NN 2 Moz all Saf all Op 7 HTML 4

`<frameset>...</frameset>`

可选

frameset元素用于在浏览器程序窗口中定义多框架的布局形式。它的主要任务就是定义矩形框架的行列布局格式。frameset元素中已定义的所有属性均会直接应用于内嵌的frame元素，如果某个frame需要特定的属性值，可以为它单独指定同名属性。在HTML文档中，frameset元素的标签往往会替代body元素。

在使用中，可以在一个frameset中内嵌另一个frameset。这种做法可以进一步切分框架集中的某个框架，从而将其划分为两个或多个框架。例如，下面的示例代码定义了一个3行2列的frameset元素，这样就可以得到一个包含6个框架的框架集：

```
<frameset rows="33%, 33%, 34%" cols="50%, 50%">
   <frame name="r1c1" id="r1c1"...>
   <frame name="r1c2" id="r1c2"...>
   <frame name="r2c1" id="r2c1"...>
   <frame name="r2c2" id="r2c2"...>
   <frame name="r3c1" id="r3c1"...>
   <frame name="r3c2" id="r3c2"...>
</frameset>
```

图1-1显示了框架的实际组织形式。

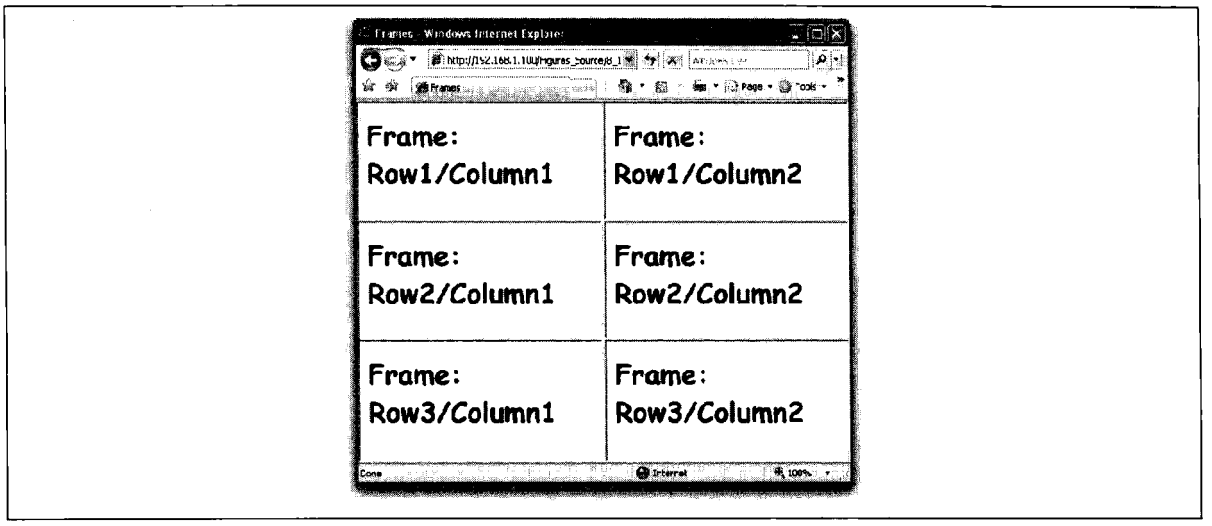


图 1-1：三行两列的框架集

另一方面，如果框架集中的框架内嵌了一个frameset，那么这个框架会根据内嵌框架集的设定切分为多个子框架。请参考这个嵌套的框架集：

```
<frameset rows="33%, 33%, 34%">
   <frame name="r1" id="r1"...>
   <frameset cols="50%, 50%">
      <frame name="r2c1" id="r2c1"...>
      <frame name="r2c2" id="r2c2"...>
   </frameset>
   <frame name="r3" id="r3"...>
</frameset>
```

其框架排列方式如图1-2所示。

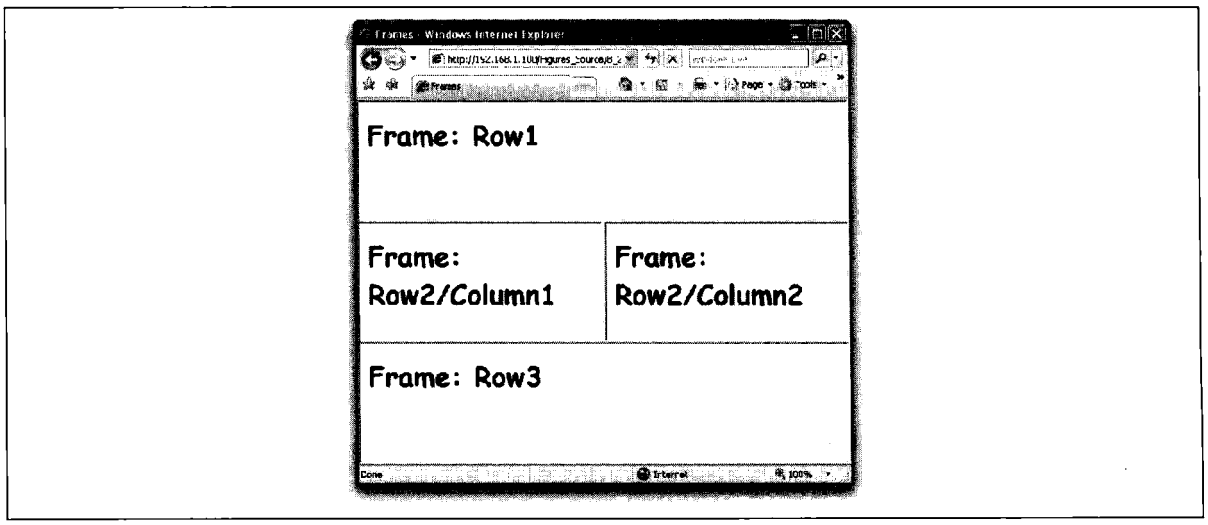


图 1-2：嵌套的框架集

可以根据页面的实际设计需要，任意反复嵌套frameset元素。但须要注意，在某些操作系统中，这种嵌套的

布局形式会占用大量的内存资源。同时，尽管这种结构会使页面设计具有逻辑意义，但并不是所有的用户都喜欢那些带边框的框架。

根据最外层frameset元素的title属性，浏览器窗口的标题栏会显示相应内容。而在框架集内部加载的文档则无法控制标题栏的显示，尽管如此，为了支持脚本或未来的一些潜在应用，依然建议为每个框架文档均指定title属性。

此外，还可以提供一个调用脚本的链接或按钮，以便用户消除框架。该脚本可以为top.location.href属性设置一个URL地址，以对应于框架集中最重要的页面的URL地址。

在HTML 4和XHTML的严格DTD中，已经明确地不再支持frameset和frame元素，以及其他元素中用于指向框架的target属性。这些文档类型定义规范已经将框架排除在文档标记之外。但使用过渡版HTML 4和框架集DTD或XHTML框架集DTD，仍然可以验证带框架的文档。

示例

```
<frameset cols="150,*">
   <frame name="navbar" id="navbar" src="nav.html">
   <frame name="main" id="main" src="page1.html">
</frameset>
```

对象模型引用方式 [window.]document.getElementById(*elementID*)

元素专有属性 border、bordercolor、cols、frameborder、framespacing、rows

元素专有事件处理属性

处理程序	IE	NN	其他	HTML	处理程序	IE	NN	其他	HTML
onload	3	2	all	4	onunload	3	2	all	4

border

IE *4* NN *3* Moz *all* Saf *all* Op *7* HTML *n/a*

```
border="像素值"
```

可选

在默认情况下，框架会显示3D边框。而在不同的浏览器和操作系统中，边框的厚度也会有所不同。可通过设定frameset元素的border属性值，来调整边框的厚度。但是在嵌套的框架集中，只有最外层的frameset元素才会响应border的属性设定。值得注意的是，这个属性只是控制内部框架的边框，而非框架集的外围边框。

在Windows和Macintosh操作系统中，Navigator和Mozilla浏览器内边框的默认厚度相同，其显示效果与border设置为5时相同。而在IE浏览器内，Windows系统下的默认值为6而Mac系统的默认值仅为1（尽管实际的显示效果远大于1像素）。正因为如此，就算设置了相同的border属性值，其显示效果在不同的浏览器上也会有所不同。另外，如果属性值较小，部分浏览器的表现也比较奇怪。例如，当设定的宽度小于等于4时，Windows操作系统内的IE 6将不再显示边框；而Mozilla浏览器在面对小于等于2的值时，则会失去3D显示效果。

框架边框的这种混乱现状阻止了它的具体使用，人们往往将border设置为0。但在有些情况下，如果框架内容需要滚动条，那么边框可以消除视觉显示时的一些问题。例如，如果没有边框，那么滚动条看起来就像悬浮在浏览器窗口中，部分浏览者会不适应这种显示效果。

在HTML 4规范中，并未包含border属性，这会让用户觉得应该使用样式单的边框属性替代框架的border属性。然而，在大多数情况下，边框相关的CSS样式属性仅能影响整个框架集四周的边框，对于框架之间的边框却无能为力。

示例

```
<frameset cols="150,*" border="0">...</frameset>
```

值 整数值。设置为0时则完全不显示边框。尽管此值用于指定框架集中边框厚度的精确像素值，但在不同的操作系统或浏览器中，具体的显示效果可能并不完全遵从此属性值。

默认值 参见上文中的详细说明。

对象模型引用方式 [window.]document.getElementById(*elementID*).border

bordercolor

IE *4* NN *3* Moz *all* Saf *n/a* Op *n/a* HTML *n/a*

`bordercolor="颜色三元组或颜色名"` 可选

此属性值用于确定框架集中所有可见边框的颜色。如果在嵌套的frameset元素或独立的frame元素中已指定了bordercolor，那么它可能会替代最外层frameset元素的bordercolor属性设置。当邻接的框架拥有不同的边框颜色时，浏览器会自动解决其带来的显示冲突。因此，如果使用混合边框颜色，请仔细测试所选用的颜色组合。

示例 `<frameset cols="150,*" bordercolor="salmon">...</frameset>`

值 一个十六进制的三元组或明语颜色名称。目前可用的明语颜色名称请参见附录A。

默认值 突出显示的黑色或蓝色，并具有灰色阴影。

对象模型引用方式 [window.]document.getElementById(*elementID*).borderColor

cols

IE *3* NN *2* Moz *all* Saf *all* Op *7* HTML *4*

`cols="列长度值列表"` 可选

此属性定义了框架集中各框架所占据的列尺寸或比例。如果要使用frameset元素创建包含多列的框架，那么就必须为cols属性指定一个值列表，值列表中的每一个值均对应于框架集中的一列。

有如下3种方式定义列尺寸：

- 绝对的像素尺寸；
- 相对于整个框架集宽度的百分比；
- 在其他列使用像素或百分比确定了宽度后，可使用通配符（*）表示剩余的全部可用空间。

在用户改变整个浏览器窗口大小时，如果期望框架的宽度仍然保持不变，那么就可以使用一个绝对的像素值定义框架的列尺寸。当框架中显示一个固定宽度的对象时（如显示一个图片），这种方式尤为有效。如果期望框架的宽度能够随着浏览器窗口的大小同步调整，始终占据一个固定的百分比，那么就可以使用第2种定义方式。如果cols属性中的值均采用百分比的形式，那么这些百分比数值之和应该为100%。当其和不等于100%时，浏览器会自动进行适当的调整。在第3种方式中，在其他框架集中部分框架宽度均已明确指定后，可以使用“*”指定列宽，那么浏览器会计算该框架的实际宽度。在值列表中，使用逗号“,”分隔各数值。

此外，在值列表中，可以混合使用以上3种方式定义的数值。例如，假设某框架集中共包含3纵列。如果期望其最左端的列宽为150像素，而中间一列始终保持框架集宽度的50%，那么就可以进行如下设置：

```
<frameset cols="150,50%,*">
```

在不同浏览器的宽度调节下，此框架集中右侧的两个框架的实际宽度会有所不同。在本例中，最右侧框架的宽度约等于框架集宽度的一半再减去150像素，而最左侧框架则占据了150像素。

有时还可以在右侧定义一个不可见的列。这时对可见列的宽度须要使用百分比数值，并且保证它们的宽度之和为100%。然后使用“*”指定最后一列的宽度。

通过在frameset元素标签内指定cols和rows属性值，可以创建一个常规的框架网格。对于那些非常规的排列方式，就必须使用嵌套的frameset元素，请参见本节前文中frameset元素的相关说明。

示例 `<frameset cols="25%,50%,25%">...</frameset>`

值 使用逗号分隔的数值列，数值形式可为像素值、百分比或通配符（*）。当数值中包含通配符

时，Macintosh系统中IE 4浏览器的显示有时会不正确。

默认值 100%

对象模型引用方式 [window.]document.getElementById(*elementID*).cols

frameborder

IE *3* NN *3* Moz *all* Saf *all* Op *n/a* HTML *n/a*

`frameborder="功能开关"` 可选

框架的边框实际上就是框架之间的分隔线，此属性即用于控制是否显示框架集内所有框架的边框。frame元素中的frameborder属性会替代frameset元素中的同名属性，但有时并不能将这些独立设置frameborder属性的框架从原有的框架集组合中正确除去。因此，在框架中使用frameborder属性时要小心谨慎，最好能在所有的浏览器和操作系统平台下均进行测试。

示例 `<frameset cols="25%,50%,25%" frameborder="no">...</frameset>`

值 根据所处浏览器的不同，适用于此属性的开关值也不相同。Navigator浏览器使用“yes”和“no”，而IE 4及后续版本的浏览器除可以使用“yes”和“no”外，还可以接受1和0。因此，为了保证浏览器间的兼容性，建议使用“yes”和“no”作为属性值。

默认值 yes

对象模型引用方式 [window.]document.getElementById(*elementID*).frameBorder

framespacing

IE *3* NN *n/a* Moz *n/a* Saf *n/a* Op *n/a* HTML *n/a*

`framespacing="像素长度值"` 可选

在IE浏览器中，border属性的早期版本就是framespacing。为了保证后向兼容性，目前的IE浏览器依然支持这个旧版的属性。在支持不同操作系统的IE浏览器中，framespacing属性的效果更为统一，例如，当边框厚度设置为10像素时，Windows和Mac版的IE浏览器的显示效果完全一致。因此，如果仅仅在IE浏览器下进行部署，则framespacing更为适用，这样可以保证在不同操作系统下边框的显示效果基本一致。

示例 `<frameset cols="25%,50%,25%" framespacing="7">...</frameset>`

值 正整数。然而与border不同的是，将frameborder设置为0时依然会显示边框。因此，请使用frameborder属性来完全隐藏边框。

默认值 2

对象模型引用方式 [window.]document.getElementById(*elementID*).frameSpacing

rows

IE *3* NN *2* Moz *all* Saf *all* Op *7* HTML *4*

`rows="行长度值列表"` 可选

此属性定义了框架集中各框架所占据的行尺寸或比例。如果要使用frameset元素创建包含多行的框架，那么就必须为rows属性指定一个值列表，值列表中的每一个值均对应于框架集中的一行。

有如下三种方式定义行尺寸：

- 绝对的像素尺寸；
- 相对于浏览器窗口整个框架集高度的百分比；
- 在其他行使用像素或百分比确定了宽度后，可使用通配符（*）表示浏览器窗口中剩余的全部可用空间。

当用户改变整个浏览器窗口大小时，如果期望框架的高度仍然保持不变，那么就可以使用一个绝对的像素值定义框架的行尺寸。当框架中显示一个固定高度的对象时（如显示一个图片），这种方式尤为有效。如果期

望框架的高度能够随着浏览器窗口的大小同步调整，始终占据一个固定的百分比，那么就可以使用第二种定义方式。如果rows属性中的值均采用百分比的形式，那么这些百分比数值之和应该为100%。当其和不等于100%时，浏览器会自动进行适当的调整。在第三种方式中，在其他框架集中部分框架宽高度均已明确指定后，就可以使用“*”指定行高，那么浏览器会计算该框架的实际高度。在值列表中，使用逗号分隔各数值。

在值列表中，可以混合使用以上三种方式定义的数值。例如，假设某框架集中共包含三横行。如果期望其最底端的行高为80像素，以容纳导航栏，而中间一行始终保持框架集高度的50%，那么就可以进行如下设置：

```
<frameset rows="*,50%,80">
```


在不同浏览器的高度调节下，此框架集中上部的两个框架的实际高度会有所不同。在本例中，顶端的框架的高度约等于框架集高度的一半再减去80像素，而底部框架则占据了80像素高。

有时，可以在底部定义一个不可见的行。这时，对可见行的高度须要使用百分比数值，并且保证它们的和为100%。然后使用“*”指定最后一行的高度。

通过在frameset元素标签内指定cols和rows属性值，可以创建一个常规的框架网格。对于那些非常规的排列方式，就必须使用嵌套的frameset元素，请参见本节上文中frameset元素的相关说明。

示例 `<frameset rows="25%,50%,25%">...</frameset>`

值 使用逗号分隔的数值，数值形式可为像素值、百分比或通配符（*）。当数值中包含通配符时，Macintosh系统中IE 4浏览器的显示有时会不正确。

默认值 100%

对象模型引用方式 `[window.]document.getElementById(elementID).rows`

<h1>,<h2>,<h3>, <h4>,<h5>,<h6>

IE *all* NN *all* Moz *all* Saf *all* Op *7* HTML *all*

```
<h1>...</h1>, <h2>...</h2>, <h3>...</h3>
<h4>...</h4>, <h5>...</h5>, <h6>...</h6>
```

HTML 结束标签：必选

HTML定义了一系列的标题元素，这类标题共分6个等级，每个等级中均包含一个数字，以表示该标题之相关部分的重要程度。h1元素表示最重要，而h6则表示最不重要。HTML文档解析器就可以根据页面内的标题标签创建内容表格。因此，为了构造合适的文档结构，应当按照次序依次使用这些标题等级，而不要为了美观刻意地跳跃使用。

每一级标题对应的默认字体、字号及其他特征均由浏览器决定。每个标题元素均会在同一行中显示，并不会加入换行符。段落元素在其分块内容之前应使用标题元素指定标题。在Windows系统中，IE 7和Firefox 1.5内这6种标题等级的显示效果请见图1-3。其他浏览器版本和操作系统的显示效果基本上也与此相似。

可以使用样式单规则改变全部或单独一个标题的显示效果。

示例

```
<h1>The Solar System</h1>
<p>Floating gracefully within the Milky Way galaxy is our Solar System. ...</p>
<h2>The Sun</h2>
<p>At a distance of 93,000,000 miles from Earth, the Sun...</p>
<h3>The Planets</h3>
<p>Nine recognized planets revolve around the Sun. ...</p>
<h4>Mercury</h4>
...
```

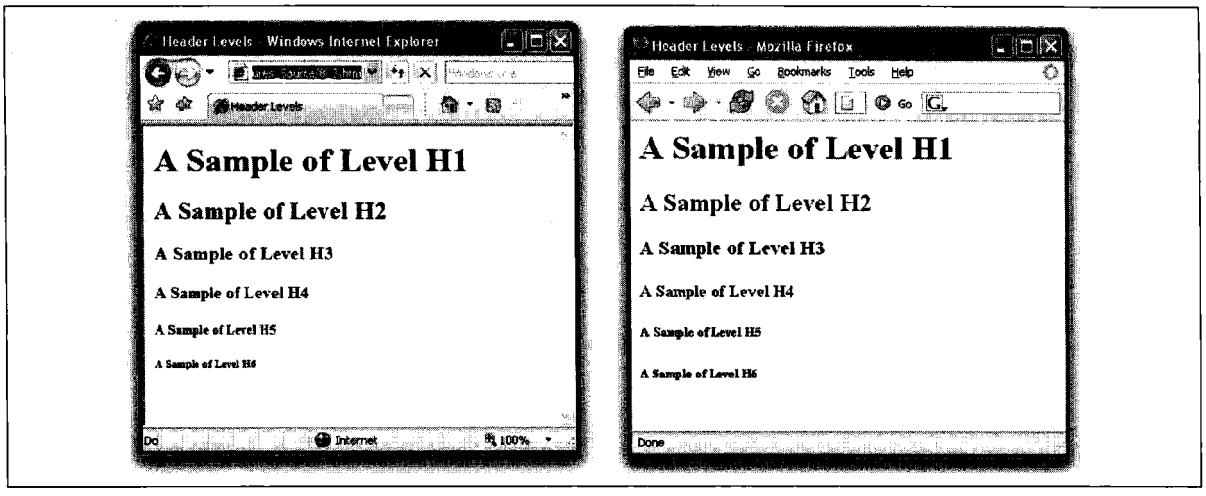


图 1-3：IE 7 和 Firefox 1.5 中的标题

对象模型引用方式 [window.]document.getElementById(*elementID*)
元素专有属性 align
元素专有事件处理属性 无。

align

IE *all* NN *all* Moz *all* Saf *all* Op *7* HTML *3.2*

align="对齐位置" 可选

此属性决定了元素文本在元素块中如何对齐。主流浏览器支持3种对齐属性值：center、left和right。另外，HTML 4过渡版本还为多行文本增加了一种两端对齐的对齐方式。

HTML 4.0已不再赞成使用align属性，推荐使用样式单属性text-align。

示例

```
<h1 align="center">Article I</h1>
```

值

下文中的表格列出了align属性的可用值。这些可用值在使用时并不区分大小写。

值	IE 4+	其他	HTML4	值	IE 4+	其他	HTML4
center	·	·	·	left	·	·	·
justify	—	—	·	right	·	·	·

默认值 left

对象模型引用方式 [window.]document.getElementById(*elementID*).align

<head>

IE *all* NN *all* Moz *all* Saf *all* Op *7* HTML *all*

<head>...</head> HTML 结束标签：可选

head元素包含着文档的很多信息，这些信息往往不会显示在浏览器窗口中。浏览器最多只会在窗口的标题栏上显示title元素的内容。

那些有助于浏览器处理文档数据的相关元素几乎都放置在head元素之中。还有一种通过meta元素表征的数据类型也可以起到辅助作用，搜索引擎和文档解析器可以从中获取作者提供的抽象信息，以便更好地了解文档内容。下表列出了在3种不同规范中可嵌入到head元素中的相关元素。

<hr>

尽管HTML 4或XHTML标准并未明确支持id属性，但浏览器依然认为这个属性是这两种标准所支持的W3C DOM元素通用属性，从而允许使用。

元素	IE 4+	NN 4 和 Mozilla	HTML 4 和 XHTML
base	.	.	.
basefont	.	.	—
bgsound	.	—	—
isindex	—	.	.
link	.	.	.
meta	.	.	.
nextid	.	—	—
object	—	—	.
script	.	.	.
style	.	.	.
title	.	.	.

119

示例

```
<head>
<meta name="Author" content="Danny Goodman">
<style type="text/css">
   h1 {color:cornflowerblue}
</style>
</head>
```

对象模型引用方式

```
[window.]document.getElementsByTagName("head")[0]
[window.]document.getElementById(elementID)
```

元素专有属性 profile

元素专有事件处理属性 无。

profile

IE *6* NN *n/a* Moz *all* Saf *all* Op *n/a* HTML *4*

```
profile="URL列表"
```

可选

元数据配置文件是一个定义了一个或多个元数据属性特征的独立文件。而W3C将这一属性的具体应用留给了浏览器生产商。但现有的浏览器并不会对此属性设置做出任何特殊反应。

示例

```
<head profile="http://www.example.com/profiles/common">
   <meta name="Author" content="Jane Smith">
   <meta name="keywords" content="benefits,insurance,">
   ...
</head>
```

值 任何有效的URL或浏览器配置文件。

默认值 浏览器的默认值

120

对象模型引用方式

```
[window.]document.firstChild.firstChild.profile
[window.]document.getElementsByTagName("head")[0].profile
[window.]document.getElementById(elementID).profile
```

<hr>

IE *all* NN *all* Moz *all* Saf *all* Op *7* HTML *all*

```
<hr>
```

HTML结束标签：禁用

根据浏览器的视觉规则，hr元素在一系列属性的控制之下可以绘制出一条水平分隔线。作为一个块级元素，

hr元素会在单独的一行上显示水平线，这种效果看起来就像元素前后都使用了br元素一样。hr不是一个内容容器元素，而且由于HTML 4建议使用对应的样式单规则，因此一些使用很久的属性也没有得到HTML 4的支持。HTML 推荐规范让浏览器生产商来决定其默认的显示规格。

示例 `<hr align="center" width="80%" />`

对象模型引用方式 `[window.]document.getElementById(elementID)`

元素专有属性 `align、color、noshade、size、width`

元素专有事件处理属性 无。

align

IE *all* NN *all* Moz *all* Saf *all* Op *7* HTML *3.2*

`align="对齐位置"` 可选

此属性决定了hr元素的显示位置与临近的最外层容器（一般是body或html元素）之间的空间关系。HTML 4已不再赞成使用align属性，推荐使用样式单属性align。

示例 `<hr align="right" />`

值 可选用center、left和right之一，均不区分大小写。

默认值 center

对象模型引用方式 `[window.]document.getElementById(elementID).align`

color

IE *4* NN *n/a* Moz *all* Saf *all* Op *n/a* HTML *n/a*

`color="颜色三元组或名称"` 可选

此属性用于设置hr元素的颜色。设置color属性后，会一并打开noshade属性。

示例 `<hr color="salmon">`

值 一个十六进制的三元组或明语颜色名称。目前可用的明语颜色名称请参见附录A。

默认值 无。

对象模型引用方式 `[window.]document.getElementById(elementID).color`

noshade

IE *all* NN *all* Moz *all* Saf *all* Op *7* HTML *3.2*

`noshade` 可选

如果出现了noshade属性，那么浏览器就会使用平面样式绘制水平线，而不是默认的3D样式。本属性仅用于IE浏览器，一旦设置了color属性，那么浏览器就会将线条的默认样式转换为无阴影的样式。

示例 `<hr noshade>`

值 一旦此属性出现即表示使用无阴影的显示样式。

默认值 Off

对象模型引用方式 `[window.]document.getElementById(elementID).noShade`

size

IE *all* NN *all* Moz *all* Saf *all* Op *7* HTML *3.2*

`size="像素值"` 可选

通过设置size属性，可以改变hr元素的默认线条厚度。HTML 4已不再赞成使用size属性，推荐使用样式单属性height。

示例 `<hr size="4">`

值 正整数。如果设置为0，其线条厚度为1像素。

默认值 2

<html>

对象模型引用方式 [window.]document.getElementById(*elementID*).size

122 ## width

IE *all* NN *all* Moz *all* Saf *all* Op *7* HTML *3.2*

width="长度值" 可选

此属性决定了hr元素线条宽度，该值即可以是精确像素宽度，也可用宽度的百分比（与容器元素有关）来表示。HTML 4并不赞成使用此属性，而建议使用样式单属性width。

示例 <hr width="75%">

值 任意一个单位为像素的长度值，或者可用空间的百分比。

默认值 100%

对象模型引用方式 [window.]document.getElementById(elementID).width

<html>

IE *all* NN *all* Moz *all* Saf *all* Op *7* HTML *all*

<html>...</html> HTML 结束标签：必选

html元素是整个文档内容的容器，其中也包括上文提及的head元素。通常，html元素的起始标签紧跟在文档类型定义声明（参见本章上文中提及的DOCTYPE元素）之后，位于HTML文件中的第二行。如果文件内未提供DTD（即使用浏览器默认的DTD），那么html的起始标签就会放置在文件的第一行。而结束标签则应放置在文件的最后一行，当然它并不须要独自占据一整行。

尽管html元素本身并不会显示，但在一个完全遵从W3C的环境下，文档上下文将根据它进行定位。此元素已应用于支持W3C DOM的所有浏览器（包括 Mozilla、Safari、Opera）和IE 6中，它往往位于文档起始部分的DOCTYPE定义之后。但Windows版本的IE浏览器（从IE 4到IE 5.5）则将根据body元素进行文档上下文定位。如果不为body元素指定页边距、边框或进行填充，根本无法发现这两者之间的区别。

在HTML 4和XHTML标准中，并未明确支持在html元素中使用id、class或style属性，但目前的DHTML浏览器均支持这些属性。

示例

```
<html>
<head>
   ...
</head>
<body>
   ...
</body>
</html>
```

元素专有属性 version、xmlns

123 **元素专有事件处理属性** 无。

version

IE *6* NN *n/a* Moz *all* Saf *all* Op *n/a* HTML *3.2*

version="字符串" 可选

HTML 4并不赞成使用version属性，而且这一属性也从未在主要的浏览器中得到采用，直到人们期望通过它声明相关的标准规范时，浏览器生产商才引入这个将要消失的属性。可以使用这个属性指定文档所支持的HTML DTD版本，这个信息通常应用在单独的DTD声明中，并位于文档内html元素之前。但在实际应用中请勿使用此属性。

值 任何字符串。该字符串应该位于一对引号之中，双引号是推荐的标准格式，但也可使用单引号。

默认值 无。

xmlns

IE 5 NN n/a Moz all Saf all Op 7 HTML x1.0

`xmlns="命名空间"`

可选

在每个XHTML文档的html元素中，均应包含W3C属性及其对应的URI值。微软使用这一属性后，可以使得Windows系统中的IE 5及后续版本浏览器能够引用非HTML的元素对象，例如微软所实现的各种行为（可应用于任意元素的外部脚本模块）。

根据其用法的不同，xmlns属性有两种赋值运算符。如果使用等号（=）作为赋值运算符，那么就会为默认的命名空间指定一个命名空间规范（通常是一个URI），以供当前元素及其所有子元素使用。如果使用冒号（:）作为运算符，则会为命名空间前缀指定对应的命名空间规范，这样一来，当使用此前缀为属性指定特定的命名空间时，该前缀可作为命名空间的快捷方式。

例如，在XHTML文档中，可使用此属性表明当前文档将使用W3C XHTML DTD，作为文档中元素与属性的含义的来源。

```
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="en" lang="en">
```

当使用来源不同的DTD共同管理元素和属性规则时，就须要在文档中使用命名空间前缀，这些命名空间往往是一些XML文档。例如：

```
<html xmlns:au="http://www.writers.org/2002/author-guidelines">
```

一旦指定了前缀au，那么在<html>元素作用域内的任意其他元素均可使用此前缀，以代表一个由DTD与命名空间URI定义的属性。从理论上说，这样就可以让“自定义”的属性得以生效。

```
<p au:license="Creative Commons">...</p>
```

示例（仅适用于Windows下IE 5及后续版本）

```
<html xmlns:MSIE>...</html>
```

值 对XHTML而言，确定的URI字符串为：http://www.w3.org/1999/xhtml。对IE 5和较新的命名空间参考而言，前缀名（内定值为MSIE）或前缀加URI的形式都可以作为前缀的另一种标示符。注意使用冒号作为分隔符。

默认值 无。

IE all NN all Moz all Saf all Op 7 HTML all

`<i>...</i>`

HTML结束标签：必选

i元素是HTML 4数个字体样式元素之一。它会以斜体的形式显示其包含的文本。可以嵌套使用这些字体样式元素以产生复合效果，例如，通过“bold-italic text”可以产生粗斜体。

而浏览器会决定是用斜体显示系统字体，还是加载当前特定字体的斜体版本。因此，可能无法根据系统字体来推断浏览器的斜体显示效果。如果期望完全控制字体，使它按预期完美地呈现出来，最好使用样式单（如果目标浏览器支持，也可以使用可下载的字体）来指定一个真正的斜体。

最好的方法是利用此元素的容器，通过样式单规则指定页面内部分甚至全部i元素的样式。例如，如果希望使用红色显示所有i元素的内容。那么，就可以指定样式规则“i{color:red}”，这么短小的一段代码就使所有的i元素均改变颜色。

尽管HTML 4依然赞成使用此元素，但可以预见的是，将来它会被样式单属性font-style所取代。

示例 `<p>This product is <i>new</i> and <i>improved</i>!</p>`

对象模型引用方式 `[window.]document.getElementById(elementID)`

元素专有属性 无。

元素专有事件处理属性 无。

<iframe>

IE 3 NN n/a Moz all Saf all Op 7 HTML 4

`<iframe>...</iframe>` HTML结束标签：必选

iframe元素可以在连续的文档内容中创建一个内嵌的框架。该框架是一个矩形空间，可以在这个空间内载入任何其他的HTML文档，也可以通过脚本向空间内动态写入内容。如果为iframe元素指定了name属性值，那么就可将它作为a、form或其他元素中target的属性值，这样就可以在这个iframe中加载对应的目的或返回文档。

125

iframe元素如果嵌入在一段内容中间，那么处于其起始标签之前的内容会在iframe元素的矩形空间之前显示，而结束标签之后的内容则会在矩形空间之后显示。与环绕在img或object元素四周的文本相似，位于iframe元素之前的文本也会采取同样的方式进行排列。

支持iframe元素的浏览器会忽略文档内容之间的起始和结束标签。其他浏览器则会将这些内容作为内嵌的HTML内容进行显示，以便用户了解到他们遗漏了什么信息，也许还会为这些信息提供一个链接。在Navigator 4中，在功能性与行为上均与iframe元素最接近的是ilayer元素。

在大多数方面，iframe与frame元素很相似，但它并不须与同框架集一起使用。事实上，此元素与frame元素非常相似，但如果通过框架引用语法（window.*frameName*），其返回对象与window对象的类型相同，而不是一个frame元素对象。

尽管iframe、frame和frameset元素均可以通过过渡版HTML 4 DTD中的验证，但只有iframe可以通过过渡版XHTML DTD的验证。在HTML和XHTML的框架集DTD中，iframe均可通过验证。

示例

```
<iframe src="quotes.html" width="150" height="90">
<a href="quotes.html" target="new" style="color:darkred">
 Click here to see the latest quotes </a>
</iframe>
```

对象模型引用方式

`[window.]document.getElementById(elementID)`

元素专有属性 align、allowtransparency、datafld、datasrc、frameborder、height、hspace、longdesc、marginheight、marginwidth、name、scrolling、security、src、vspace、width

元素专有事件处理属性 无。

align

IE 3 NN n/a Moz all Saf all Op 7 HTML 4

`align="对齐常量"` 可选

本属性决定了在四周环绕的上下文内容中，iframe矩形空间的对齐方式。对于此元素属性的可用值，可参见本章前文中的“对齐常量”。

示例 `<iframe src="quotes.html" width="150" height="90" align="center"></iframe>`

值 不区分大小写的对齐常量。

默认值 left

126

对象模型引用方式 `[window.]document.getElementById(elementID).align`

allowtransparency

IE 5.5 NN n/a Moz n/a Saf n/a Op n/a HTML n/a

`allowtransparency="true"` 可选

此属性用于决定iframe元素的背景平面是否透明。如果要透过iframe显示主文档的内容，那么须要将allowtransparency设置为“true”，同时，iframe的background-color样式属性须要保持为默认设置或明确设置为“transparent”。值得注意的是，无论iframe中加载了什么文档，该透明作用均会影响iframe元素。因此，如果只想通过背景样式控制iframe元素，那么就必须将iframe的allowtransparency属性值设置为“true”，并且设置iframe下方出现的元素背景（此时假设iframe中加载的文档背景为透明状态）。如果将allowtransparency属性设置为“false”，那么iframe元素相关的背景样式就无法显示，此时iframe元素中内嵌的文档背景样式则是可见的。

示例

```
<iframe src="album.html" height="400" width="400" allowtransparency="true"></iframe>
```

值 常量：true | false。

默认值 false

对象模型引用方式 [window.]document.getElementById(*elementID*).allowTransparency

datafld

IE 4 NN n/a Moz n/a Saf n/a Op n/a HTML n/a

`datafld="列名称"` 可选

此属性用于实现IE浏览器中的数据绑定功能，可以使用远程数据源中的列名代替iframe元素中的src属性。数据来源列必须包含一个有效的URI地址（无论是相对还是绝对地址）。此时，元素中还必须设定datasrc属性。注意，只能在IE 5/Mac操作系统下实现文本文件数据源的绑定。

示例 `<iframe datasrc="DBSRC3" datafld="newsURL"></iframe>`

值 区分大小写的标识符。

默认值 无。

对象模型引用方式 [window.]document.getElementById(elementID).dataFld

datasrc

IE 4 NN n/a Moz n/a Saf n/a Op n/a HTML n/a

`datasrc="数据源名称"` 可选

为实现IE数据绑定功能，此属性用于指定页面内object元素的ID值，以便在进行远程数据访问时加载数据源对象。通过datafld属性指定来自于数据源中的绑定内容。注意，只能在IE 5/Mac操作系统下实现文本文件数据源的绑定。

示例 `<iframe datasrc="DBSRC3" datafld="newsURL"></iframe>`

值 区分大小写的标识符。

默认值 无。

对象模型引用方式 [window.]document.getElementById(*elementID*).dataSrc

frameborder

IE 3 NN n/a Moz all Saf all Op n/a HTML 4

`frameborder="功能开关"` 可选

此属性用于控制iframe元素是否显示边框。如果期望iframe的内嵌文档看起来就像是嵌入在主文档中的一部分，那么可以关闭frameborder属性。

示例 `<iframe src="quotes.html" width="150" height="90" frameborder="no"></iframe>`

值 根据使用场合的不同，适用于此属性的开关值也不相同。HTML 4指定1为开，0为关，在IE 4或部分浏览器中，还运行使用“yes”和“no”作为属性值。

默认值 1

对象模型引用方式 `[window.]document.getElementById(elementID).frameBorder`

height,width

IE *4* NN *n/a* Moz *all* Saf *all* Op *7* HTML *4*

`height="像素值" width="像素值"` 可选

这些属性用于确定iframe的尺寸大小。HTML 4不再赞成使用这两个属性，建议使用CSS的height和width样式单属性。

示例 `<iframe src="news.html" height="200" width="200">`

值 任意一个单位为像素的长度值，或者可用空间的百分比。

默认值 width为300像素，height为150像素。

对象模型引用方式

```
[window.]document.getElementById(elementID).height
[window.]document.getElementById(elementID).width
```

hspace,vspace

IE *4* NN *n/a* Moz *n/a* Saf *n/a* Op *7* HTML *n/a*

`hspace="像素值" vspace="像素值"` 可选

这两个属性用于设定在文档内容中iframe元素四周的填充区域大小。hspace属性控制左右两侧的填充（水平填充），而vspace属性则控制顶部和底部的填充（竖直填充）。使用填充后，会在框架四周形成一个空白的缓冲区。另外，还可以调整样式单的页边距来达到这一效果，这种做法特别适合于控制单独一条边的空白空间。同时，使用样式单也具有更好的跨浏览器兼容性。

示例 `<iframe src="news.html" hspace="20" vspace="10">`

值 正整数。

默认值 0

对象模型引用方式

```
[window.]document.getElementById(elementID).hspace
[window.]document.getElementById(elementID).vspace
```

longdesc

IE *6* NN *n/a* Moz *all* Saf *all* Op *n/a* HTML *4*

`longdesc="URL"` 可选

此属性可指定一个文档的URL地址，该文档中的内容比元素的title属性值丰富，因此可以更细致地描述该元素。在未来的浏览器中，此属性可以作为元素的一个注释信息，以便那些无法看到屏幕的用户也能获知相关信息。Mozilla中，通过在上下文菜单中设置这个元素的特性，可以为此属性指定的URL显示一个包含链接的小窗口。目前主流的浏览器尚未为该属性提供更多的功能。

示例

```
<iframe longdesc="newsDesc.html" title="Navigation Bar" src="news.html"> </iframe>
```

值 任何有效的完整或相对URL地址。

默认值 无。

marginheight,marginwidth

IE 3 NN n/a Moz all Saf all Op 7 HTML 4

```
marginheight="像素值" marginwidth="像素值"
```

可选

这两个属性值用于表示框架的内边框与显示内容之间空白距离的像素值。marginheight属性控制框架的上下边距高度，而marginwidth则控制左右边距宽度。HTML 4规范将它们的默认值留给浏览器来处理。

在框架内，浏览器会插入默认的空白空间，其宽度介于8像素和14像素之间。但是一定要注意的是，如果想尝试改变默认状态，改变其中的任何一个可能会导致另一个值变为0。因此，最好同时设置这两个值，以防止框架中的内容完全充满整个框架空间。

129

示例 `<iframe src="news.html" marginheight="20" marginwidth="14"></iframe>`

值 任何大于或等于0的整数。

默认值 由浏览器和操作系统共同决定。

对象模型引用方式

```
[window.]document.getElementById(elementID).marginHeight
[window.]document.getElementById(elementID).marginWidth
```

name

IE 3 NN n/a Moz all Saf all Op 7 HTML 4

```
name="元素标识符"
```

可选

如果某框架中的链接或表单必须在其他框架中加载目标文档或返回的文档，那么就可以在这些元素中设置target属性，以指定用于接收新内容的框架。为了将这些内容导向一个框架，该框架必须为其name属性指定属性值。该属性值与a或form元素中target的属性值完全相同。

示例 `<iframe name="news" id="news" src="news.html"></iframe>`

值 区分大小写的标识符。

默认值 无。

对象模型引用方式 `[window.]document.getElementById(elementID).name`

scrolling

IE 3 NN n/a Moz all Saf all Op 7 HTML 4

```
scrolling="功能开关"
```

可选

当内嵌框架中加载的内容超出其可视区域时，浏览器会默认添加水平和/或竖直滚动条。如果设计框架时并不期望有滚动条出现，那么出现的滚动条就会占据框架内容的部分空间，从而影响到部分内容的布局。同时，由于在不同浏览器和操作系统中，字体的默认尺寸各不相同，因此在不同的客户端中，同样的文本内容也会有不同的显示效果。如果须要阻止框架中出现滚动条，可以将scrolling设置为no，如果希望框架总是包含滚动条，则须要将其设置为yes。在后一种情况中，如果显示内容时并不需要滚动，那么滚动条虽然依然可见，但会处于禁用状态。

在将scrolling属性设置为no之前，应该进行详尽的测试，以保证在所有浏览器和平台下，关键内容都能够在框架中正常显示。如果不允许滚动框架中的内容，那么部分用户可能无法看到框架中的全部内容。

当然，也可以使用CSS的overflow样式属性来代替这两个属性。另外，微软还通过其overflow-x和overflow-y样式属性为滚动条提供了额外的轴向控制。

130

示例 `<iframe src="news.html" scrolling="no"></iframe>`

值 常量值：auto | no | yes。

默认值 auto

对象模型引用方式 `[window.]document.getElementById(elementID).scrolling`

security

IE *6* NN *n/a* Moz *n/a* Saf *n/a* Op *n/a* HTML *n/a*

`security="restricted"`

可选

如果激活此属性，那么就会将该内嵌框架的安全级别提高，达到Windows安全选项设置中的受限摄级别。这样一来，框架内容就有可能无法执行任何脚本。

示例 `<iframe src="news.html" security="restricted"></iframe>`

值 常量：`restricted`

默认值 无。

src

IE *3* NN *n/a* Moz *all* Saf *all* Op *7* HTML *4*

`src="URL"`

可选

此属性定义了一个URL，`iframe`元素将要加载的这URL地址中的内容。它既可以是绝对URL，也可以是一个包含框架集说明的相对URL文档路径。此外，也可以在框架中使用由脚本返回的`javascript`伪URL属性值。如果忽略了`src`属性，那么框架中将是一片空白。

示例 `<iframe src="news.html"></iframe>`

值 完整或相对URL，以及`javascript`伪URL。

默认值 无。

对象模型引用方式 `[window.]document.getElementById(elementID).src`

vspace

参见hspace属性。

width

参见height属性。

<ilayer>

IE *n/a* NN *4* Moz *n/a* Saf *n/a* Op *n/a* HTML *n/a*

`<ilayer>...</ilayer>`

HTML结束标签：必选

`ilayer`元素与Navigator 4中特有的`layer`元素相对应，它是`layer`的内嵌版本。在某些情况下，IE中`ilayer`元素与`iframe`元素的工作方式相同，但Navigator 4的对象模型则将`ilayer`视为一个可定位元素，例如，就像一个拥有CSS `position`属性的块级元素，可进行定位。因此，`ilayer`元素与`layer`元素拥有很多相同的属性，同时，也是根据Navigator 4的定位、尺寸调整和可定位元素的排列方式来命名这些属性。

`ilayer`元素的内容既可以从`src`属性指定的单独文件中读取，也可以使用起始和结束标签包含在当前文档中。还可以在同一个`ilayer`元素中同时引入这两种类型的内容。在这种情况下，`src`属性指定的文档内容首先显示，而后，附加的内容才会出现在外部文档内容的矩形框之后的一行。

示例 `<ilayer id="thingy1" src="quotes.html" width="150" height="90"></ilayer>`

对象模型引用方式 `[window.]document.layerName`

元素专有属性 `above`、`background`、`below`、`bgcolor`、`clip`、`height`、`id`、`left`、`name`、`src`、`top`、`visibility`、`width`、`z-index`

元素专有事件处理属性

处理程序	NN	其他	HTML	处理程序	NN	其他	HTML
onblur	4	n/a	n/a	onmouseout	4[a]	n/a	n/a
onfocus	4	n/a	n/a	onmouseover	4[a]	n/a	n/a
onload	4	n/a	n/a	onmouseup	4[a]	n/a	n/a
onmousedown	4[a]	n/a	n/a				

[a] 仅适用于事件捕捉模式

above

IE *n/a* NN *|4|* Moz *n/a* Saf *n/a* Op *n/a* HTML *n/a*

```
above="层 ID"
```

可选

此属性指出在叠放次序中，放置在当前ilayer元素之前（之上）的可定位元素。这是设置z-index属性的一种变相方法，但它并不依靠随机编号体系。如果使用了above属性，那么就不要在同一个ilayer元素中使用below或z-index属性。

示例 `<ilayer id="thingy4" src="quotes.html" above="thingy3"></ilayer>`

值 区分大小写的标识符。

默认值 无。

对象模型引用方式 `[window.]document.layerName.above`

background

IE *n/a* NN *|4|* Moz *n/a* Saf *n/a* Op *n/a* HTML *n/a*

```
background="URL"
```

可选

此属性指定了一个图像文件，它将作为ilayer元素中文本和其他内容的背景。与浏览器载入的其他普通图像不同的是，背景图会以未经缩放的原始大小进行加载，然后贴附并充满整个可用的页面空间。小图片虽然下载迅速，但总是会很明显地在背景中重复显示。也可以使用GIF动画作为背景图，但这实在是一件令读者抓狂的做法。在选择背景图时，一定选择相对柔和的图片，以便使主体内容更加引人注目。如果一定要使用背景图，那么请选用非常淡雅的图片。

示例 `<ilayer id="thingy4" src="quotes.html" background="blueCrinkle.jpg"></ilayer>`

值 任何有效的图像文件URL地址，既可使用相对URL，也可以使用绝对URL地址。

默认值 无。

对象模型引用方式 `[window.]document.layerName.background`

below

IE *n/a* NN *|4|* Moz *n/a* Saf *n/a* Op *n/a* HTML *n/a*

```
below="层 ID"
```

可选

此属性指出在叠放次序中，放置在当前ilayer元素之后（之下）的可定位元素。这是设置z-index属性的一种变相方法，但它并不依靠随机编号体系。如果使用了below属性，那么就不要在同一个ilayer元素中使用above或z-index属性。

示例 `<ilayer id="thingy4" src="quotes.html" below="thingy5"></ilayer>`

值 区分大小写的标识符。

默认值 无。

对象模型引用方式 `[window.]document.layerName.below`

133

bgcolor

IE *n/a* NN |4| Moz *n/a* Saf *n/a* Op *n/a* HTML *n/a*

```
bgcolor="颜色三元组或颜色名"
```

可选

此属性为整个矩形层设置填充色，它将位于文本和其他内容之下。如果同时使用bgcolor和background属性，那么背景色会透过背景图的任何透明部分。

示例 `<ilayer src="quotes.html" bgcolor="tan"></ilayer>`

值 一个十六进制的三元组或明语颜色名称。不设置属性值就默认为黑色（#000000，black）。目前可用的明语颜色名称请参见附录A。

默认值 由操作系统决定。

对象模型引用方式 `[window.]document.layerName.bgColor`

clip

IE *n/a* NN |4| Moz *n/a* Saf *n/a* Op *n/a* HTML *n/a*

```
clip="[左侧像素值，顶端像素值,] 右侧像素值，底部像素值"
```

可选

裁剪区域是整个ilayer内容中的矩形可见区域。只能看见裁剪域矩形框内的页面内容。相关内容进入ilayer元素后，显示这些内容所需的空间将决定clip属性的默认值。设置clip后，可以控制过长的内容，如果它们的尺寸超出了页面设计时确定的矩形区域，那么不会显示超出部分。

示例 `<ilayer src="quotes.html" clip="50,50"></ilayer>`

值 当元素进入文档时，从元素的左上边缘处开始计算clip属性的像素值。这些值的属性从左边缘开始，依顺时针方向环绕矩形框四周，分别为：左、上、右、下。如果仅仅提供两个值，那么Navigator会假定左侧和上端属性值为0，这意味着只会调整右侧和底部边缘。因此，设置为“50,50”时，意味着裁剪区域是一个从层左上角起始的矩形块，面积为50像素×50像素。如果期望保持可视面积不变，但从距离左边缘10像素的位置开始创建区域，那么clip属性就应设置为“10,0,60,50”。

默认值 ilayer内容的自然可视空间大小。

对象模型引用方式

```
[window.]document.layerName.clip.left
[window.]document.layerName.clip.top
[window.]document.layerName.clip.right
[window.]document.layerName.clip.bottom
```

134

height,width

IE *n/a* NN |4| Moz *n/a* Saf *n/a* Op *n/a* HTML *n/a*

```
height="像素值" width="像素值"
```

可选

这两个属性决定了层进入文档时的最小尺寸。然而，一旦向层中注入了内容，那么这两个属性设置依然无法限制可见空间的尺寸，在x和y轴方向上均会超出限定值。例如，如果要在ilayer中显示一个长为90像素、宽为120像素的图像，那么当ilayer元素的height和width属性值过小时，就会扩展该层的可视尺寸，以便显示整张图像。对文本或其他内容也会同样处理，为了显示全部内容，将扩展可视空间。为限制内容的可见部分，请设置clip属性。

在其他已定位的内容下创建彩色或有图案的矩形（使用bgcolor或background属性）作为衬底时，为height和width属性设置特定的大小是很有用的。当没有任何内容要溢出ilayer边框时，裁剪域将保持height和width属性指定的尺寸。

示例 `<ilayer bgcolor="yellow" height="100" width="100"></ilayer>`

值 由引号括起来的正整数或百分比。将这两个属性值减小为0时，不仅可以掩藏元素（也可以通过visibility属性到达这一目的），还可以防止元素占据任何的页面空间。

默认值 ilayer内容的自然可视空间大小。

对象模型引用方式

```
[window.]document.layerName.height
[window.]document.layerName.width
```

id

IE *n/a* NN |4| Moz *n/a* Saf *n/a* Op *n/a* HTML *n/a*

```
id="元素标识符"
```

可选

此属性是一个独一无二的标识符，使所属元素区别于文档中所有的其他元素。此标识符将被用作above和below属性的属性值。另外，脚本也可以使用id属性值作为ilayer元素的名称，以便实现对象引用。

示例 `<ilayer id="oldYeller" bgcolor="yellow" height="100" width="100"></ilayer>`

值 区分大小写的标识符。

默认值 无。

对象模型引用方式 `[window.]document.layerName.name`

left,top

IE *n/a* NN |4| Moz *n/a* Saf *n/a* Op *n/a* HTML *n/a*

```
left="像素值" top="像素值"
```

可选

这两个属性定义了层的左、上边缘与原有显示位置相比的偏移量。由于ilayer元素是内嵌层，因此它相对页面的精确显示位置会发生变化，这意味着在连续的HTML内容中的任一位置均可显示ilayer元素。一旦使用了这两个属性之一，Navigator 4就会在文档中保留ilayer元素曾经显示过的区域，由于元素位置改变而空出的空间并不会被周围的内容所占据。因此，只有在为定制的内容构建完美外观时，才会在ilayer元素中使用这些属性，如在流动的图像之间插入ilayer。

示例 `<ilayer bgcolor="yellow" left="10" top="50"></ilayer>`

值 正整数值，可不使用引号。

默认值 0

对象模型引用方式

```
[window.]document.layerName.left
[window.]document.layerName.top
```

name

IE *n/a* NN |4| Moz *n/a* Saf *n/a* Op *n/a* HTML *n/a*

```
name="元素标识符"
```

可选

此属性是一个独一无二的标识符，使所属元素区别于文档中所有的其他元素。此标识符可作为above和below属性的属性值。在进行对象引用时，name属性和id属性可互换。

示例 `<ilayer name="oldYeller" bgcolor="yellow" height="100" width="100"></ilayer>`

值 区分大小写的标识符。

默认值 无。

对象模型引用方式 `[window.]document.layerName.name`

src

IE *n/a* NN |4| Moz *n/a* Saf *n/a* Op *n/a* HTML *n/a*

```
src="URL"
```

可选

为了在ilayer元素中加载外部HTML文件，须要在src属性中指定该文件的URL地址。在ilayer中，首先显示从src指定的URL中载入内容，紧随其后的才是元素的起始和结束标签中所包含的HTML内容。如果忽略

了src属性，那么只会显示标签之间的内容。在ilayer加载文档后，仍然可以通过脚本改变对应的对象属性（即src），以便在不重载主文档的情况下动态改变ilayer元素中的内容。

示例 `<ilayer src="quotes.html"></ilayer>`

值 完整或相对URL路径。

默认值 无。

对象模型引用方式 `[window.]document.layerName.src`

top

参见left属性。

visibility

IE *n/a* NN |4| Moz *n/a* Saf *n/a* Op *n/a* HTML *n/a*

`visibility="可见性常量"` 可选

Navigator 4根据此属性决定是否显示ilayer元素。在默认情况下，此图层会继承其临近的最外层图层的visibility属性设置。ilayer元素是文档主体的一部分，由于主体通常是可见的，因此ilayer在默认情况下也是可见的。当页面加载时如果要隐藏层，那么只须将visibility属性设置为“hidden”。只有当ilayer元素内嵌在另一个layer元素中，并且该layer元素的visibility值为“hidden”（或继承父层的“hidden”）时，才须要将ilayer元素的visibility设置为“show”。

但无论如何设置visibility属性，ilayer元素均会在文档中占据一定的空间。这样一来，Navigator 4就不必重载相关文档，直接通过脚本动态改变可见性。由于Navigator 4不会自动地对已改变的内容重新排版，这种做法就可以规避这一窘境的发生。

示例 `<ilayer src="quotes.html" visibility="hidden"></ilayer>`

值 选用以下常量之一：hidden | inherit | visible

默认值 inherit

对象模型引用方式 `[window.]document.layerName.visibility`

width

参见height属性。

z-index

IE *n/a* NN |4| Moz *n/a* Saf *n/a* Op *n/a* HTML *n/a*

`z-index="层号"` 可选

对文档内所有的层而言，此属性可以控制它们相对于临近的最外层层容器在Z轴上的叠放次序。如果在同一个容器（如文档的基础层）内，两个或多个可定位元素拥有相同的z-index值，那么HTML源代码中元素的加载顺序就决定了页面上元素的叠放次序，较晚加载的元素将放置在较早加载的元素之前。在所有可定位元素中，z-index的默认值均为0。因此，如果希望将一个可定位元素放置在其他所有元素之前，只须要简单地将它的z-index属性设置为任意一个正整数。通过脚本则可以实时调整可定位元素的叠放次序。请参见above和below元素。

示例 `<ilayer src="quotes.html" z-index="1"></ilayer>`

值 任意一个整数值。

默认值 0

对象模型引用方式 `[window.]document.layerName.zIndex`

IE *all* NN *all* Moz *all* Saf *all* Op *7* HTML *all*

<img> HTML 结束标签：禁用

任何浏览器能够处理的MIME类型的图像，都可以通过img元素进行显示。这些图片一般是GIF或JPEG格式，但目前的浏览器还能解码PNG和BMP等位图图像。除非帮助程序将这些文件类型交由外部程序处理，否则它们均由浏览器本身进行处理。img元素是一种内嵌元素，它可以出现在文档中的任一位置，甚至可以在一行文本的中间显示。HTML 4已不再赞成使用此元素中大量用于控制视觉效果的属性，请使用对应的样式单规则。然而，为了兼容那些挂满图片的旧网页，仍可以在很多浏览器中继续安全地使用这些属性。同时还要注意，如果要在Navigator 4中使用样式单控制img元素的边框和边距，那么必须将img元素内嵌在div或span元素内，然后对容器元素应用样式单。在其他所有能够感知CSS的浏览器中，这种设置方式可正常工作，因此可以通过这种方式保证在不同浏览器环境下均可使用样式单。

如果想让整张图片成为一个可点击的链接，那么请将img元素放置在一个a元素内。然后，将border属性设置为0，就会消除围绕在图像四周的链接外框。对图像映射（图片的不同区域链接到不同的目标）而言，HTML推荐标准建议使用客户端图像映射（使用usemap属性）而不是服务器端图像映射（使用ismap属性）。如果不是图片链接，那么IE 4及后续版本和W3C DOM浏览器均可在图像中指定onclick事件处理程序。这种方式的唯一缺点是，须要通过CSS伪类:hover或其他事件来控制鼠标指针的样式。

为了兼容早期的支持脚本的浏览器，最好在所有的img元素标签中均使用height和width属性。在img元素中指定了这两个属性后，浏览器不再须要在等待完成图片加载之后再决定其尺寸以组织页面内其他内容，因此能够大大加快浏览器显示页面的速度。

示例

```
<img src="mtRushmore.jpg" height="90" width="120" alt="Mount Rushmore" />
```

对象模型引用方式

```
[window.]document.imageName
[window.]document.images[i]
[window.]document.getElementById(elementID)
```

元素专有属性

align、alt、border、datafld、datasrc、dynsrc、galleryimg、height、hspace、ismap、longdesc、loop、lowsrc、name、src、start、suppress、usemap、vspace、width

元素专有事件处理属性

处理程序	IE	NN	其他	HTML	处理程序	IE	NN	其他	HTML
onabort	4	3	all	n/a	onload	4	3	all	n/a
onerror	4	3	all	n/a					

align

IE *all* NN *all* Moz *all* Saf *all* Op *7* HTML *all*

align="对齐位置" 可选

此属性决定了img元素相对于其临近的最外围容器元素和四周内容的相对位置。某些设置还可以让图像浮动到页面的左、右边缘，并且让周围的文字环绕在图片四周。

本章前文所述的有关对齐常量的大部分规则，也适用于img元素。HTML 4并不赞成使用align属性，而推荐使用align这一样式单属性。

示例

```
<img src="surferDude.gif" align="right" alt="Surfer" height="100" width="200">
```

值

每个浏览器均为此属性定义了不同的属性值集合。尽管align属性自诞生一来一直沿用至今，但并不是每个属性值都能够延续下来。一些使用不广的属性值，如absmiddle和baseline，在Navigator 3和IE 4中可以使用，

但并没有收录在W3C指令集中。因此，为页面中相邻的多个图像设置不同的对齐方式可能会出现问题，这些图像和周围内容的显示和定位往往无法预料。可以根据实际工作中的部署环境从下表中选择合适的属性值：

值	IE 4+	NN 3+/Mozilla/Safari	Opera	HTML 4
absbottom	•	•	•	—
absmiddle	•	•	•[a]	—
baseline	•	•	•	—
bottom	•	•	•	•
left	•	•	•	•
middle	•	•	•	•
right	•	•	•	•
texttop	•	•	—	—
top	•	•	•	•

[a] 与其他浏览器不同，Opera 并不区分 absmiddle 和 middle。它把这两个属性均视为 middle，其处理方式与其他浏览器没有区别。

默认值 bottom

对象模型引用方式

```
[window.]document.imageName.align
[window.]document.images[i].align
[window.]document.getElementById(elementID).align
```

alt

IE all NN all Moz all Saf all Op 7 HTML all

```
alt="文本信息"
```

必选

虽然在这个世界上充斥着很多图形浏览器，但须要牢记的是并非所有的浏览器均会下载图像，也不是所有的上网者能够看见图像。除了那些使用VT100终端的浏览器无法显示图片，如Lynx等，通过避免下载、显示图片，掌上电脑也可以有更好的效果。而那些视觉受损的用户可能无法看到图像，但如果能使得他们了解图片的大致内容也是有所裨益的。当img元素出现在页面中时，文本浏览器可以在对应的地方显示img元素内alt属性的相关文本。那些能够朗读页面文本的浏览器也可以读出alt的文本内容。alt属性应该包含有关图像的扼要说明，而对那些用作空间填充物的图像而言，使用空字符串即可。HTML推荐标准非常看重这个功能，因此它要求在img元素中必须使用alt属性。

示例 <img src="navbar.gif" usemap="#nav" alt="Navigation Bar" width="400" height="50">

值 使用引号括起来的任意字符串。

默认值 无。

对象模型引用方式

```
[window.]document.imageName.alt
[window.]document.images[i].alt
[window.]document.getElementById(elementID).alt
```

border

IE all NN all Moz all Saf all Op 7 HTML 3.2

```
border="像素值"
```

可选

此属性用于控制img元素四周边框的厚度。img元素边框的默认颜色为黑色，但当它内嵌在a元素之内时，会根据链接的状态使用不同的边框颜色。如果要为平面边框设置不同的颜色，请使用样式单。但在Navigator 4中，请先将img放置在div或span元素内。当链接包围在图像四周时，通过将border属性设置为0，也可以让带颜色的边框完全消失。

示例 <img src="surferDude.gif" alt="Surfer" border="3" height="100" width="200">

值 任意整数值。

默认值 0

对象模型引用方式

```
[window.]document.imageName.border
[window.]document.images[i].border
[window.]document.getElementById(elementID).border
```

datafld

IE 4 NN n/a Moz n/a Saf n/a Op n/a HTML n/a

```
datafld="列名称"
```

可选

此属性用于IE中的数据绑定，以便将一个远程数据源的列名与img元素内src属性所需的URL联系在一起。同时，img元素中还必须设定datasrc属性。注意，只能在IE 5/Mac操作系统下实现文本文件数据源的绑定。

示例

```
<img datasrc="DBSRC3" alt="Current Radar" datafld="img3URL" height="100" width="150">
```

值 区分大小写的标识符。

默认值 无。

对象模型引用方式

```
[window.]document.imageName.dataFld
[window.]document.images[i].dataFld
[window.]document.getElementById(elementID).dataFld
```

datasrc

IE 4 NN n/a Moz n/a Saf n/a Op n/a HTML n/a

```
datasrc="数据源名称"
```

可选

为实现IE数据绑定功能，此属性用于指定页面内object元素的ID值，以便在进行远程数据访问时加载数据源对象。通过datafld属性可指定来自于数据源中的绑定内容。注意，只能在IE 5/Mac操作系统下实现文本文件数据源的绑定。

示例

```
<img datasrc="DBSRC3" alt="Current Radar" datafld="img3URL" height="100" width="150">
```

值 区分大小写的标识符。

默认值 无。

对象模型引用方式

```
[window.]document.imageName.dataSrc
[window.]document.images[i].dataSrc
[window.]document.getElementById(elementID).dataSrc
```

dynsrc

IE 4 NN n/a Moz n/a Saf n/a Op n/a HTML n/a

```
dynsrc="URI"
```

可选

IE 4及后续版本允许通过img元素显示视频剪辑和VRML（Virtual Reality Modeling Language），以作为embed或object元素的一种替代。为了帮助浏览器区分动态和静态图像源文件，可以使用dynsrc代替src属性，以便加载视频剪辑。这样一来，img元素的矩形空间即可用于播放视频剪辑。请参见start属性，以及用于控制播放次数的loop属性。

示例 `<img dynsrc="snowman.avi" loop="3" height="100" width="150">`

值 任何有效的完整或相对URL地址。

默认值 无。

对象模型引用方式

```
[window.]document.images[i].dynsrc
[window.]document.imageName.dynsrc
[window.]document.getElementById(elementID).dynsrc
```

galleryimg

IE |6| NN *n/a* Moz *n/a* Saf *n/a* Op *n/a* HTML *n/a*

```
galleryimg="功能开关"
```

可选

在IE 6中，如果一个图片尺寸大于130像素×130像素，那么通过这个属性可以决定在鼠标滑过时是否在图片上显示Windows操作系统的“我的图片”工具栏。这个工具栏可以为图像提供保存、打印或E-mail等功能的点击式快捷方式。但目前在该工具栏中出现哪些按钮还无法控制。

示例

```
<img src="rushmore.jpg" alt="Mount Rushmore" height="240" width="550" galleryimg="no">
```

值 常量：yes、true、no和false。

默认值 yes

对象模型引用方式

```
[window.]document.images[i].galleryImg
[window.]document.imageName.galleryImg
[window.]document.getElementById(elementID).galleryImg
```

height,width

IE *all* NN *all* Moz *all* Saf *all* Op *7* HTML *3.2*

```
height="长度值" width="长度值"
```

可选

这两个属性决定了页面中为图像预留的空间尺寸，此时并不须要考虑图像的实际尺寸。为了更好地显示图像，同时兼顾脚本兼容性，应该使用图像的实际像素高和宽为这两个属性赋值。如果使用其他尺寸，那么浏览器会拉伸或缩放图像，以适应由它们定义的空间大小，这时的显示效果可能不佳。

示例 `<img src="surferDude.gif" alt="Surfer" height="100" width="200">`

值 正整数或百分比。

默认值 图像的实际尺寸。

对象模型引用方式

```
[window.]document.imageName.height
[window.]document.images[i].height
[window.]document.getElementById(elementID).height
[window.]document.imageName.width
[window.]document.images[i].width
[window.]document.getElementById(elementID).width
```

hspace,vspace

IE *all* NN *all* Moz *all* Saf *all* Op *7* HTML *3.2*

```
hspace="像素值" vspace="像素值"
```

可选

这两个属性决定了img元素的边距，即环绕在其可见内容四周的白色填充域的大小。hspace确定了图像框左右两侧的边距大小，而vspace则决定了上下两端的边距大小。HTML 4不再赞成使用这两个属性，建议使用CSS中与边距或填充相关的样式单属性。

示例 `<img src="desk3.gif" alt="My Desktop" vspace="10" hspace="10">`

值 整数值，表示img元素框对应边边距宽度的像素值。

默认值 0

对象模型引用方式

```
[window.]document.imageName.hspace
```

```
[window.]document.images[i].hspace
[window.]document.getElementById(elementID).hspace
[window.]document.imageName.vspace
[window.]document.images[i].vspace
[window.]document.getElementById(elementID).vspace
```

ismap

IE *all* NN *all* Moz *all* Saf *all* Op *7* HTML *all* 143

```
ismap
```

可选

ismap是一个布尔属性，它通知浏览器该img元素此时将作为服务器端图像映射。为了将一个图像转化为一个服务器端图像映射，首先须要将img元素内置于一个a元素中，同时将这个a元素的href属性指向一个CGI程序的URL地址，用以解析点击时的坐标信息。就像使用GET方法提交表单时，须要将表单数据附加在action属性的URL之后一样，浏览器也会将点击时的坐标信息附加在URL后。例如，如果用户点击的坐标点为（50，25），那么浏览器就会将“http://www.example.com/cgi-bin/nav.pl?50,25”发送给服务器。服务器端一个名为“nav.pl”的程序可能会检查该坐标点出现的区域，然后返回一个相关的HTML内容给客户端。

当然，浏览器也允许使用客户端图像映射（请参见usemap属性），由于不须要与服务器之间通信以检查点击时的坐标位置，用户使用这种图像映射时会更为快捷。

示例

```
<a href="http://www.example.com/cgi-bin/nav" target="main">
<img src="navbar.gif" alt="Navigation Bar" ismap height="90" width="120"></a>
```

值 如果此属性出现，即表示打开此功能。

默认值 Off

对象模型引用方式

```
[window.]document.imageName.isMap
[window.]document.images[i].isMap
[window.]document.getElementById(elementID).isMap
```

longdesc

IE *6* NN *n/a* Moz *all* Saf *all* Op *n/a* HTML *4*

```
longdesc="URL"
```

可选

此属性可指定一个文档的URL地址，该文档中的内容比元素中alt或title属性值更为丰富，因此可以更细致地描述该元素。在未来的浏览器中，此属性可以作为元素的一个注释信息，以便那些无法看到屏幕的用户也能获知相关信息。Mozilla中，通过在上下文菜单中设置这个元素的特性，可以为此属性指定的URL显示一个包含链接的小窗口。目前主流的浏览器尚未为该属性提供更多的功能。

示例 `<img longdesc="navDesc.html" alt="Navigation Bar" src="navbar.jpg">`

值 任何有效的完整或相对URL地址。

默认值 无。

对象模型引用方式

```
[window.]document.imageName.longDesc
[window.]document.images[i].longDesc
[window.]document.getElementById(elementID).longDesc
```

loop

IE *3* NN *n/a* Moz *n/a* Saf *n/a* Op *n/a* HTML *n/a* 144

```
loop="整数值"
```

可选

如果使用dynsrc属性指定了视频剪辑，即可通过loop属性控制该剪辑在加载后的播放次数。如果将loop值设定为0，那么会加载该剪辑，但不会开始播放。用户双击图像后，尚未运行的视频剪辑将开始播放。由于页面

内没有明显的控制标志，因此应该为用户提供相应的使用说明。但要注意，此属性无法控制*gif*动画的回放。

示例　<img dynsrc="snowman.avi" alt="Snowman Movie"loop="3" height="100" width="150">

值　任何大于或等于0的整数。

默认值　1

对象模型引用方式

```
[window.]document.imageName.loop
[window.]document.images[i].loop
[window.]document.getElementById(elementID).loop
```

lowsrc

IE *4* NN *3* Moz *all* Saf *n/a* Op *n/a* HTML *n/a*

```
lowsrc="URL"
```

可选

由于并非所有用户都拥有快速的网络连接速度，因此高清晰度的图像可能花费较长的时间才能下载到客户端。为了填补这段空虚的等待时间，作者可以通过lowsrc属性指定一个低清晰度或替代的图像，它将首先显示在文档空间中。注意，这个低清晰度的图像应该与src指定的原图拥有相同的像素尺寸。

示例

```
<img src="navbar.jpg" alt="Navigation Bar" lowsrc="navbarBW.jpg" height="60" width="300">
```

值　任何有效的完整或相对URL地址。

默认值　无。

对象模型引用方式

```
[window.]document.imageName.lowsrc
[window.]document.images[i].lowsrc
[window.]document.getElementById(elementID).lowsrc
```

name

IE *4* NN *3* Moz *all* Saf *all* Op *7* HTML *4*

```
name="元素标识符"
```

可选

由于在早期文件中并不存在id属性，因此在使用脚本操作图像时为了保证后向兼容性，可以使用name属性值引用img对象。与通过数值索引从document.images数组中获得的img对象相比，使用名称来引用img对象往往更为可靠，无论何时重新排列或删除了图像，均可保持针对剩余的已命名图像的引用。当然，如果只在现代的浏览器中使用，那么可以使用id来替代name属性。

示例

```
<img name="mugshot" id="mugshot" alt="My face" src="janem.jpg" height="90" width="80">
```

值　区分大小写的标识符。

默认值　无。

对象模型引用方式

```
[window.]document.images[i].name
[window.]document.imageName.name
[window.]document.getElementById(elementID).name
```

src

IE *all* NN *all* Moz *all* Saf *all* Op *7* HTML *all*

```
src="URL"
```

必选

本属性指定了一个文件的URL地址，img元素将显示该文件中包含的图像数据。一般来说，如果希望通过img元素显示任何图像，均应指定src属性，但也有例外，如在IE中使用dynsrc属性播放视频剪辑或使用datasrc进行数据绑定时就不须要使用这个属性。但值得注意的是，只有当浏览器能够处理该图像的MIME类型时，

才能正常显示。在万维网上，GIF和JPEG是最常用的图像格式。此外，在HTML或XHTML验证中，也必须使用src属性，它们并不接受IE提供的另外两个替代属性dynsrc和datasrc。

示例 `<img src="surferDude.gif" alt="Surfer" height="100" width="200">`

值 完整或相对URL路径。

默认值 无。

对象模型引用方式

```
[window.]document.images[i].src
[window.]document.imageName.src
[window.]document.getElementById(elementID).src
```

start

IE 4 NN n/a Moz n/a Saf n/a Op n/a HTML n/a

`start="视频开启形式"` 可选

一旦在IE浏览器中为img元素指定了dynsrc属性，就可以通过img元素播放视频剪辑。此时，视频有两种开启方式，一种是在视频文件加载完成后立即开始播放，另一种则是当用户鼠标指针位于图像框之上时才开始播放。通过start属性，可以选用最适合于页面的启动方式。

示例 `<img dynsrc="snowman.avi" loop="1" start="mouseover" height="100" width="150">`

值 两个常量fileopen或mouseover的其中之一，使用时均不区分大小写。

默认值 fileopen

对象模型引用方式

```
[window.]document.images[i].start
[window.]document.imageName.start
[window.]document.getElementById(elementID).start
```

suppress

IE n/a NN 4 Moz n/a Saf n/a Op n/a HTML n/a

`suppress="功能开关"` 可选

一旦在Navigator 4中使用了此属性，那么在图像下载过程中将不会在页面上显示图像图标和alt文本，也不会突出显示图像域。但是，如果图像加载失败，无论是否开启此属性，手动图标均会显示在图像空间上。

示例 `<img src="surferDude.gif" alt="Surfer" height="150" width="250" suppress="true">`

值 布尔字符串：true或false。

默认值 false

usemap

IE all NN all Moz all Saf all Op 7 HTML 3.2

`usemap="映射 URL"` 可选

通过map和area元素，可以定义一个客户端图像映射。可将map元素看作一个包含一个或多个area元素的已命名的容器。每个area元素均在图像上设定了一个“热点”区域，并且指定了对应的链接目标（或其他设定），当用户点击该区域时就会做出相应的响应。而usemap属性的作用就是将同一个文档中的img元素和一个已命名的map元素联系起来。这样一来，map元素的名称即可视为一个定位标记，而该map元素的“地址”由“#”和元素名称组成。

示例

```
<img src="navbar.gif" alt="Navigation Bar" usemap="#navigation" height="90" width="120">
```

值 位于同一文档中的map元素URL地址，即#符号加上map元素的名称。

默认值 无。

对象模型引用方式

```
[window.]document.imageName.useMap
[window.]document.images[i].useMap
[window.]document.getElementById(elementID).useMap
```

vspace

参见hspace属性。

width

147

参见height属性。

<input>

IE *all* NN *all* Moz *all* Saf *all* Op *7* HTML *all*

`<input>` HTML结束标签：禁用

尽管并不是所有的input元素都会显示在页面上，但仍将input元素称为表单控件。在大多数情况下，用户可以通过input元素输入文本、点击按钮，或者从列表做出选择。当form元素提交时，交互产生的数据也会一并提交到服务器端程序中。另外，input元素还可以直接应用在客户端上，作为用户与客户端脚本交互的一种途径。服务器端程序通常也将会话数据内嵌隐藏在页面内的input元素之中，这样一来，在下一个表单提交时这些数据也会同时提交。通过这种方式，当须要通过多个表单页面收集数据时，在页面传递过程中，就不须要在服务器上临时保存数据。

在HTML 4之前，应将input元素内嵌在form元素中。此后，业内逐渐放松了这个限制，但为了显示input元素，在Navigator 4中依然须要将form元素作为容器。

此元素中，type属性决定了页面中显示的具体控件类型。在标准的HTML中，可选用以下几种可用值之一：button、checkbox、file、hidden、image、password、radio、reset、submit或text（注1）。而Web Foms 2.0规范（首先由Opera 9提供了内置的支持，在*http://www.whatwg.org/specs/*中可查看相关细节）则提供了其他的输入类型：add、date、datetime、datetime-local、email、month、move-up、move-down、number、range、time、url及week。一般情况下，并不是以上所有的input元素类型都能够利用全部的input元素属性，对不同的元素类型而言，相同的一个属性可能具有不同的作用效果。下文列出了每种input元素属性可应用于哪几种类型之中。尽管textarea元素也拥有其自身的标签，但往往将它也视为一个表单控件。

示例

```
<form method="POST" action="http://www.example.com/cgi-bin/query.pl">
First Name: <input type="text" name="first" id="first" maxlength="15"><br>
Last Name: <input type="text" name="last" id="last" maxlength="25"><br>
ZIP Code: <input type="text" name="zip" id="zip" maxlength="10"><br>
<input type="reset">
<input type="submit">
</form>
```

对象模型引用方式

148

```
[window.]document.formName.inputName          // 但不能在 image 类型中使用
[window.]document.forms[i].elements[j]        // 但不能在 image 类型中使用
[window.]document.getElementById(elementID)
```

注1：Apple 还采用了另外两种附加的 input 元素类型：search 和 range，它们主要面向使用 Dashboard 小程序的用户。其中，search 类型会显示一个 Mac OS X 中的搜索域，而 range 类型则会显示一个滑块控件。尽管这些元素可以在 Safari 2 中显示，但文本并未记录相关细节。请通过此链接查阅 Safari 相关的 HTML 参考：*http://developer.apple.com/documentation/AppleApplications/Reference/ SafariHTMLRef/ SafariHTMLRef.pdf*。

元素专有属性

accept、accesskey、action、align、alt、autocomplete、autofocus、border、checked、datafld、datasrc、disabled、dynsrc、enctype、form、height、hspace、inputmode、ismap、list、loop、lowsrc、max、maxlength、method、min、name、pattern、readonly、replace、required、size、src、start、step、template、type、usemap、value、vcard_name、vspace、width

元素专有事件处理属性

处理程序	IE	NN	Opera	其他	HTML
onafterupdate	4	n/a	n/a	n/a	n/a
onbeforeupdate	4	n/a	n/a	n/a	n/a
onchange	3	2	all	all	4
onformchange	n/a	n/a	9	n/a	n/a
onforminput	n/a	n/a	9	n/a	n/a
oninvalid	n/a	n/a	9	n/a	n/a
onselect	3	2	all	all	4

注意，并不是所有的事件均能适用于全部input类型。

accept

IE *6* NN *n/a* Moz *all* Saf *all* Op *7* HTML *4*

`accept="MIME类型列表"` 可选

此属性用于指定一个或多个MIME类型，只有这些类型的文件才能在表单提交时一并上传到服务器。在提交表单之前，浏览器往往期望能够对file类型的input元素进行文件类型过滤，因此HTML 4提供了这个属性。但在目前主流浏览器中，在文件选择或提交时，该属性并没有发挥任何的实际效果。

input类型 file

示例 `<input type="file" accept="text/html, image/gif" ...>`

值 MIME类型（内容类型）。如果使用多种类型，请在列表中使用逗号进行分隔。Web Forms 2.0允许使用通配符列出可用的MIME类型范围。

默认值 无。

accesskey

IE *4* NN *n/a* Moz *all* Saf *all* Op *7* HTML *4*

`accesskey="某字符"` 可选

请参见本章前文，在共享属性中已进行了详细说明。但在file类型的input元素中，点击快捷键组合后，光标会出现在对应的文件地址文本框中，而不会自动点击“浏览”按钮。

input类型 所有的可显示类型。

action

IE *n/a* NN *n/a* Moz *n/a* Saf *n/a* Op *9* HTML *n/a*

`action="URI"` 可选

当点击input元素以提交嵌套表单时，这个Web Forms 2.0扩展允许其目标URI可以与常规的表单提交地址不同。

input类型 image、submit

align

IE *all* NN *all* Moz *all* Saf *all* Op *7* HTML *3.2*

`align="对齐常量"` 可选

本属性决定了在环绕四周的上下文中，input元素矩形空间的对齐方式。对于此元素属性的可用值，可参见本章前文“对齐常量”中对img元素的相关描述。

input类型 image

示例 <input type="image" name="icon" src="icon.gif" align="absmiddle">

值 适用于图像矩形框外围元素的任意一种对齐常量值。

默认值 left

对象模型引用方式

```
[window.]document.formName.inputName.align
[window.]document.forms[i].elements[j].align
[window.]document.getElementById(elementID).align
```

alt

IE 4 NN n/a Moz all Saf all Op 7 HTML 4

```
alt="文本信息"
```

可选

此属性提供了input元素的文本描述，在图像下载时可用它替代图像，也可用于文本-语音转换浏览器。它与img元素中的alt属性很类似。

input类型 image

示例 <input type="image" name="icon" src="sndIcon.gif" alt="Sound Icon">

值 使用引号括起来的任意字符串。

默认值 无。

对象模型引用方式

```
[window.]document.formName.inputName.alt
[window.]document.forms[i].elements[j].alt
[window.]document.getElementById(elementID).alt
```

autocomplete

IE 5 NN n/a Moz n/a Saf n/a Op n/a HTML n/a

```
autocomplete="功能开关"
```

可选

如果IE用户在选项中允许浏览器自动对表单进行填写，那么autocomplete属性就能控制整个表单的填写。IE会保存（通过一种加密的方式）过去填写过的文本和密码等条目，当用户再次访问时，浏览器就会列出与以前输入条目相符的内容，以便完成表单条目的填写。如果表单域所需的数据保存在用户的电子名片（vCard）中，那么可以为“text”和“password”类型的input元素指定vcard-name属性，如果浏览器找到了与其名称相符的用户选项条目，就会提前或辅助完成该表单域的填写工作。关于HTML表单中如何实现自动完成的更多细节，请访问此链接：*http://msdn.microsoft.com/en-us/library/ms533032(VS.85).aspx*。

Web Forms 2.0也指明了类似的使用方式，但它并不需要微软的“vCard”专利技术。相反，它允许所有支持这一功能的浏览器采用其自身的处理机制保存数据，以便在页面上实现自动完成功能。

input类型 password、text。Web Forms 2.0中任意一个与文本相关的input类型均可使用此属性。

示例

```
<input type="text" name="homephone" vcard_name="vCard.Home.Phone" autocomplete="on">
```

值 on | off。

默认值 off

对象模型引用方式

```
[window.]document.formName.inputName.autoComplete
[window.]document.forms[i].elements[j].autoComplete
[window.]document.getElementById(elementID).autoComplete
```

autofocus

IE *n/a* NN *n/a* Moz *n/a* Saf *n/a* Op *n/a* HTML *n/a*

```
autofocus="自动聚焦"
```

可选

在页面加载完成后，这个Web Forms 2.0 扩展会将焦点聚集在元素上。每个页面中只能有一个表单控件元素能拥有此属性。

input类型 所有的可显示类型。

border

IE *4(Mac)* NN *4* Moz *all* Saf *n/a* Op *n/a* HTML *n/a*

151

```
border="像素值"
```

可选

此属性用于控制input元素四周边框的厚度。在默认状态下，仅有Navigator 4会为图像提供边框。将border属性设置为0后，可消除边框。

input类型 image

示例

```
<input type="image" name="icon" src="sndIcon.gif" border="0">
```

值 任意整数值。

默认值 2（Navigator 4）或0（其他浏览器）。

对象模型引用方式

```
[window.]document.formName.inputName.border
[window.]document.forms[i].elements[j].border
[window.]document.getElementById(elementID).border
```

checked

IE *all* NN *all* Moz *all* Saf *all* Op *7* HTML *4*

```
checked
```

可选

加载页面时，这个布尔属性可决定是否开启当前checkbox或radio类型的input元素。如果单选按钮已经分组，那么在一个分组中只有一个input元素能够拥有checked属性。完成页面加载之后，还可以使用脚本改变此属性值。提交表单时，如果某个input元素拥有checked属性，那么就会将其名称/值对作为表单数据的一部分。而该元素的name和value属性的属性值则构成了它的名称/值对。当复选框或单选按钮被选中时，如果其value属性并未赋值，那么该属性会自动赋值为“active”或“on”。由于复选框只有名称各不相同，因此这种做法是可取的。然而在同一个分组中，所有单选按钮必须拥有相同的名称，只有这样浏览器才能自动处理组内已选和未选的按钮。请参见下文所述的name属性。

input类型 checkbox、radio

示例

```
<input type="checkbox" name="addToList" checked>Send email updates to this web site.
```

值 如果此属性出现，即表示开启此功能。

默认值 off

对象模型引用方式

```
[window.]document.formName.inputName.checked
[window.]document.forms[i].elements[j].checked
[window.]document.getElementById(elementID).checked
```

datafld

IE *4* NN *n/a* Moz *n/a* Saf *n/a* Op *n/a* HTML *4*

152

```
datafld="列名称"
```

可选

此属性用于IE中的数据绑定，以便将一个远程数据源的列名与不同的input元素联系在一起。此时元素中必须设定datasrc属性。注意，只能在IE 5/Mac操作系统下实现文本文件数据源的绑定。

HTML 4中依然保留此属性，但它已被XHTML 1.0删除。

input类型 button、checkbox、hidden、password、radio、text

示例 `<input type="text" name="first" datasrc="DBSRC3" datafld="firstName">`

值 区分大小写的标识符。

默认值 无。

对象模型引用方式

```
[window.]document.formName.inputName.dataFld
[window.]document.forms[i].elements[j].dataFld
[window.]document.getElementById(elementID).dataFld
```

datasrc

IE 4 NN n/a Moz n/a Saf n/a Op n/a HTML 4

```
datasrc="数据源名称"
```

可选

为实现IE数据绑定功能，此属性用于指定页面内object元素的ID值，以便在进行远程数据访问时加载数据源对象。此时须要通过datafld属性指定来自于数据源中的绑定内容。注意，只能在IE 5/Mac操作系统下实现文本文件数据源的绑定。

HTML 4中依然保留此属性，但它已被XHTML 1.0删除。

input类型 button、checkbox、hidden、password、radio、text

示例 `<input type="text" name="first" datasrc="DBSRC3" datafld="firstName">`

值 区分大小写的标识符。

默认值 无。

对象模型引用方式

```
[window.]document.formName.inputName.dataSrc
[window.]document.forms[i].elements[j].dataSrc
[window.]document.getElementById(elementID).dataSrc
```

disabled

IE 4 NN n/a Moz all Saf all Op 7 HTML 4

```
disabled
```

可选

如果一个input元素处于“禁用”状态下，那么它在屏幕上会被灰化显示，而且用户也无法对其进行操作。被禁用的表单控件无法接收焦点，进行跳格选择时也无法被激活。IE/Windows扩展了其含义，即当一个选择分组中的某个单选按钮被禁用时，该组会被全部禁用。在IE/Mac或其他浏览器中，情况并非如此，它们允许通过脚本让分组中未选中的按钮取消禁用按钮的选中状态。

表单提交时并不会将已禁用的input元素的名称/值对发送至服务器。因此，作为提交按钮的input元素一旦被禁用，将无法提交该表单。

由于disabled属性是布尔类型，因此在HTML格式中，如果出现此属性，即意味着它的值为“true”。当然，也可以通过脚本在事后调整此属性值，请参见第2章中的input对象。

input类型 所有input类型均可使用此属性。

示例 `<input type="submit" disabled value="Ready to Submit">`

值 一旦此属性出现，即表示禁用该元素。

默认值 false

对象模型引用方式

```
[window.]document.formName.inputName.disabled
```

```
[window.]document.forms[i].elements[j].disabled
[window.]document.getElementById(elementID).disabled
```

dynsrc

IE 4 NN n/a Moz n/a Saf n/a Op n/a HTML n/a

```
dynsrc="URL"
```

可选

IE 4及后续版本允许通过image类型的input元素播放视频剪辑和VRML。为了帮助浏览器区分动态和静态图像源文件，可以使用dynsrc代替src属性，以便加载视频剪辑。这样一来，input元素的矩形空间即可用于播放视频剪辑。请参见start属性，以及用于控制播放次数的loop属性。

input类型 image

示例

```
<input type="image" dynsrc="submit.avi" alt="Submit Button" loop="3" height="100" width="150">
```

值 任何有效的完整或相对URL地址。

默认值 无。

对象模型引用方式

```
[window.]document.formName.inputName.dynsrc
[window.]document.forms[i].elements[j].dynsrc
[window.]document.getElementById(elementID).dynsrc
```

enctype

IE n/a NN n/a Moz n/a Saf n/a Op 9 HTML n/a

```
enctype="MIME 类型"
```

可选

通过enctype与其他属性（如action）的协助，在提交封闭的表单时，这个Web Forms 2.0扩展允许其提交目标URI与附件的MIME类型都与常规的表单不同。

input类型 image、submit

form

IE n/a NN n/a Moz n/a Saf n/a Op 9 HTML n/a

```
form=" formID [formID] ..."
```

可选

通过这个Web Forms 2.0扩展，无论表单是否嵌入了某个控件，都可以将一个单独的表单控件元素与一个或多个表单联系在一起。由于input元素并没有像form的子元素那样受到限制，因此它可以放在form元素之外的任何地方。通过form属性可以将input元素同页面上一个或多个表单元素联系起来。

input类型 所有的可显示类型。

示例

```
<input type="text" id="searchText" form="GoogleSearch" />
```

值 页面上一个或多个form元素的ID值。此时，使用空格分隔多个ID值。

默认值 无。

height, width

IE 4 NN 4 Moz all Saf all Op 7 HTML n/a

```
height="像素值" width="像素值"
```

可选

这两个属性可用来定义image类型的input元素的尺寸大小。如果忽略了这两个属性，那么浏览器一直要等到图像完全加载后才能计算元素在页面上所占的空间大小。

input类型 image

示例

```
<input type="image" src="submit.jpg" height="20" width="60">
```

值 正整数。

默认值 图片的默认尺寸。

hspace, vspace

IE *4* NN *n/a* Moz *all* Saf *all* Op *7* HTML *n/a*

`height="像素值" width="像素值"` 可选

这两个属性可定义图像（即input元素）四周的填充区间，从而使得图像可与其他内容保持一定的最小距离。hspace控制左右两侧的填充厚度，而vspace则用于控制上下两端的填充厚度。在Safari中，其他类型的input元素也可使用此属性。

input类型 image

示例

```
<input type="image" src="submit.jpg" alt="Submit Button" height="20" width="60" hspace="10"
vspace="20">
```

值 正整数。

默认值 0

inputmode

IE *n/a* NN *n/a* Moz *n/a* Saf *n/a* Op *n/a* HTML *n/a*

`inputmode="脚本标记[修定标记]"` 可选

这个Web Forms 2.0 扩展（完全继承自W3C XForms 1.0规范，请参阅*http://www.w3.org/TR/xforms/sliceE.html*）会引导浏览器为某种书面语言显示合适的文本输入用户界面。请查阅W3C XForms 1.0文档，以获取更多详细信息。

input类型 email、password、text、url

示例 `<input type="text" id="searchText" inputmode="hiragana" />`

值 每种书面语言均伴随着一个可选择的修订标记。这些修订标记通常相当于Unicode脚本，请参见*http://www.unicode.org/unicode/reports/tr24/*。

默认值 无。

ismap

IE *all* NN *all* Moz *all* Saf *all* Op *7* HTML *all*

`ismap` 可选

ismap是一个布尔属性，它通知浏览器，image类型的input元素此时将作为服务器端图像映射。与img元素不同的是，image类型的input元素本身就有一个相关动作，即表单提交动作，因此并不须要将其放置在a元素之中。浏览器会将点击的坐标信息附加在表单动作的URL之后。客户端图像映射的相关细节，请参见usemap属性。

input类型 image

示例

```
<input type="image src="navbar.gif" alt="Navigation Bar" ismap height="90" width="120">
```

值 如果此属性出现，即表示打开此功能。

默认值 Off

对象模型引用方式

```
[window.]document.formName.inputName.isMap
[window.]document.forms[i].elements[j].isMap
[window.]document.getElementById(elementID).isMap
```

list

IE *n/a* NN *n/a* Moz *n/a* Saf *n/a* Op *9* HTML *n/a*

`list="数据列表元素 ID"` 可选

通过这个Web Forms 2.0扩展可以为输入类型定制预定义条目，当然，在input元素中也可以使用那些未包含在此列表中的文本条目。这些预定义条目编写于option元素中，并包含在Web Forms 2.0的datalist元素之内。而list属性的属性值就是包含这些预定义条目的datalist元素的ID值。在Opera中，当input元素被选中

时，这些预定义条目会以选择列表的形式显示在元素下方。

input类型 date、datetime、datetime-local、email、month、number、range、text、time、url、week

示例

```
<label>Enter your operating system:<input type="text" name="os" list="oses" /></label>
<datalist id="oses">
   <option value="">
   <option value="Windows Vista">
   <option value="Windows XP">
   <option value="Windows 98">
   <option value="Mac OS X">
   <option value="Linux">
</datalist>
```

值 相关datalist元素的ID值。

默认值 无。

loop

IE *3* NN *n/a* Moz *n/a* Saf *n/a* Op *n/a* HTML *n/a*

```
loop="整数值"
```

可选

如果使用dynsrc属性指定了视频剪辑，那么就可通过loop属性控制该剪辑在加载后的播放次数。如果将loop值设定为0，那么会加载该剪辑，但不会播放。在用户双击图像后，尚未运行的视频剪辑将开始播放。由于页面内没有明显的控制标志，因此应该为用户提供相应的使用说明。

input类型 image

示例

```
<input type="image" dynsrc="snowman.avi" alt="Snowman Movie" loop="3" height="100" width="150">
```

值 任何大于或等于0的整数。

默认值 1

对象模型引用方式

```
[window.]document.formName.inputName.loop
[window.]document.forms[i].elements[j].loop
[window.]document.getElementById(elementID).loop
```

lowsrc

IE *4* NN *all* Moz *all* Saf *n/a* Op *n/a* HTML *n/a*

```
lowsrc="URL"
```

可选

如果src属性指定的图像文件须要很长时间才能完成下载，那么可以通过此属性提供一个替代的低分辨率图像，在页面加载时将首先显示这个图像。

input类型 image

示例

```
<input type="image" src="navbar.jpg" alt="Navigation Bar" lowsrc="navbarBW.jpg"
height="60" width="300">
```

值 任何有效的完整或相对URL地址。

默认值 无。

对象模型引用方式

```
[window.]document.formName.inputName.lowsrc
[window.]document.forms[i].elements[j].lowsrc
[window.]document.getElementById(elementID).lowsrc
```

max, min

IE *n/a* NN *n/a* Moz *n/a* Saf *n/a* Op *9* HTML *n/a*

`max="取值范围最大值", min="取值范围最小值"` 可选

这个Web Forms 2.0扩展允许为某些类型的input元素指定取值范围，这些数据类型包括：数字、日期、时间及文件上传等。如果用户输入的数据超出了这个范围，那么浏览器会设置ValidityState对象的rangeUnderflow或rangeOverflow属性，这些脚本可能会向用户提供更进一步的错误提示信息。

input类型 date、datetime、datetime-local、file、month、number、range、time、week

示例 `<input type="time" name="apptTime" min="09:00" max="17:00" />`

值 number和range类型，整数或浮点数；data类型，根据ISO 8601指定的格式编写日期值，如2007-03-15；时间与日期的组合类型，根据ISO 8601指定的格式编写时间日期值，如2007-03-15T08:00:00；month类型，ISO格式的月份值，如2007-03；week类型，ISO 8601格式的日期值，如2007W3；file类型，正整数值，表明可随表单一同上传的文件数量。

默认值 无。

maxlength

IE *all* NN *all* Moz *all* Saf *all* Op *7* HTML *all*

`maxlength="字符数量"` 可选

此属性定义了在input元素的文本域中可输入字符的最大数量。在实际使用时，当用户输入的字符数量超过maxlength定义的值时，浏览器会发出蜂鸣声或给出其他提示。注意，maxlength和size属性之间并没有任何内在的联系。如果maxlength允许的最大字符数超过了元素宽度可显示的字符数，那么浏览器会在元素中使用水平滚动（尽管对很多用户而言这并不方便），以便输入或修改文本内容。

input类型 password、text

示例 `<input type="text" name="ZIP" maxlength="10">`

值 正整数。

默认值 无限制。

对象模型引用方式

```
[window.]document.formName.inputName.maxLength
[window.]document.forms[i].elements[j].maxLength
[window.]document.getElementById(elementID).maxLength
```

method

IE *n/a* NN *n/a* Moz *n/a* Saf *n/a* Op *9* HTML *n/a*

`method="GET" | "POST"` 可选

通过其他属性（如action）的协助，在提交嵌套的表单时，这个Web Forms 2.0扩展允许其提交目标URI甚至提交方式都与常规的表单不同。

input类型 image、submit

name

IE *all* NN *all* Moz *all* Saf *all* Op *7* HTML *all*

`name="元素标识符"` 可选/必选

如果input元素是表单的一部分，并且其元素值会随表单一起提交至服务器端，那么就必须使用name属性。而对那些为了方便脚本调用而使用的表单而言，input元素往往并不需要name属性。

为input元素指定的name属性值将作为名称/值对的一部分提交至服务器端。而处于同一个互斥组内的单选按钮必须拥有相同的name属性值。在某些浏览器中，如果忽略了name属性往往会造成用户无法使用复选框或单选按钮形式的input元素。

HTML 4和XHTML规范建议在整个页面内均使用id属性代替name属性。但是，如果input元素只有id属性而没有name属性，目前主流的浏览器依然无法提交表单数据。为了对所有元素都保持DHTML DOM脚本操作一致性，最好同时使用这两个属性，甚至在这两个属性上使用相同的标识符。但要注意，单选按钮是个例外，在一个分组内其name值是共享的，而id值则是彼此不同的。同时使用这两个属性值后，可以通过name属性将元素的值传递到服务器，而脚本则可通过ID访问元素对象。对那些可以使用引用语法（IE下可用document.all.elementID，而W3C DOM浏览器下则可用document.getElementById(elementID)）的浏览器而言，这种做法非常实用。严格的XHTML DTD也可验证input元素中的name属性，这样所有通过验证的页面都可以在以前和现在的浏览器中使用。

input类型 所有input类型均可使用此属性。

示例
```
<input type="text" name="ZIP" id="ZIP" maxlength="10">
```

值 区分大小写的标识符。

默认值 无。

对象模型引用方式
```
[window.]document.formName.inputName.name
[window.]document.forms[i].elements[j].name
[window.]document.getElementById(elementID).name
```

pattern

IE *n/a* NN *n/a* Moz *n/a* Saf *n/a* Op *9* HTML *n/a*

```
pattern="正则表达式"
```
可选

Web Forms 2.0扩展，当必须对用户输入进行验证时，它允许指定一个正则表达式来进行判断。

input类型 email、password、text、url

示例
```
<input pattern="[A-Z][0-9]{7}" name="partNum"
  title="A part number is an uppercase letter followed by seven numbers." />
```

值 正则表达式（与JavaScript中的正则表达式相同，不能放置在斜线符号中）。

默认值 无。

readonly

IE *4* NN *n/a* Moz *all* Saf *all* Op *7* HTML *4*

```
readonly
```
可选

当input元素中出现readonly属性时，用户就不能在页面上编辑其文本域中的内容。此时只能通过脚本对这些内容进行修改。进行跳格选择时，被标示为readonly的元素域将无法接收焦点。但在Macintosh系统下的IE 4及后续版本中，文本域依然可以接收焦点，当用户尝试着进行编辑时还会发出蜂鸣声。

input类型 password、text

示例
```
<input type="text" name="ZIP" readonly>
```

值 如果此属性已出现，就意味着其属性值为true。

默认值 false

对象模型引用方式
```
[window.]document.formName.inputName.readOnly
[window.]document.forms[i].elements[j].readOnly
[window.]document.getElementById(elementID).readOnly
```


replace

IE *n/a* NN *n/a* Moz *n/a* Saf *n/a* Op *9* HTML *n/a*

```
replace="类型"
```
可选

这个Web Forms 2.0扩展将一些指令与表单提交控件联系在一起，用以在表单提交后处理服务器返回数据的相关指令。目前有两种处理方式，默认方式是使用服务器的响应数据替换原文档。如果已为form元素的data

属性指定了一个URL，那么浏览器会将返回的数值填充到表单中，而不是重新载入表单的初始化数据，这是第二种处理方式。

input类型 image、submit

示例

```
<input type="submit" replace="values"
  action="http://example.com/custRecord.jsp" method="POST" />
```

值 document | values。

默认值 document

required

IE *n/a* NN *n/a* Moz *n/a* Saf *n/a* Op *9* HTML *n/a*

```
required="required"
```

可选

这个Web Forms 2.0 扩展表示提交时该input元素的值是否必须存在。此值设置为true后，一旦元素未设定值，那么就会为ValidityState对象设置missingValue属性值。

input类型 checkbox、date、datetime、datetime-local、email、file、month、number、password、radio、range、text、time、url、week

size

IE *all* NN *all* Moz *all* Saf *all* Op *7* HTML *all*

```
size="元素宽度"
```

可选

实际上，通过可显示的字符的数量，size属性可以限制input元素文本域的宽度。根据元素的字体设置（或默认字体设置），可计算出实际的显示宽度，但真实结果往往并不理想。由于字体样式多变（在新的浏览器中还可以指定字体样式和尺寸），有时计算出的结果会导致意想不到的狭窄文本域。因此，如果未在不同浏览器和操作系统下进行大量测试以验证具体的效果，就随意为size和maxlength属性设置属性值，往往会带来很糟糕的结果。例如，在均衡字体中，当所有字符都是“m”或“W”时，其宽度就会比其他字符组合宽得多。HTML 4规范指出，size属性还可应用于其他input元素类型，以指明其像素宽度，但就目前的浏览器来看，实际情况并非如此。与此同时，可以使用CSS属性加大按钮的宽度，而其默认宽度与value属性字符串的宽度一致。

input类型 password、text

示例 <input type="text" name="ZIP" maxlength="10" size="12">

值 正整数。

默认值 20

对象模型引用方式

```
[window.]document.formName.inputName.size
[window.]document.forms[i].elements[j].size
[window.]document.getElementById(elementID).size
```

src

IE *all* NN *all* Moz *all* Saf *all* Op *7* HTML *all*

```
src="URL"
```

必选

本属性指定了一个文件的URL地址，image类型的input元素将显示该文件中包含的图像数据。但值得注意的是，只有当浏览器能够处理该图像的MIME类型时，才能正常显示。在万维网上，GIF和JPEG是最常用的图像格式。

input类型 image

示例 <input type="image" name="icon" src="sndIcon.gif" border="0">

值 完整或相对URL路径。

默认值 无。

对象模型引用方式

```
[window.]document.formName.inputName.src
[window.]document.forms[i].elements[j].src
[window.]document.getElementById(elementID).src
```

start

IE 4 NN n/a Moz n/a Saf n/a Op n/a HTML n/a

```
start="视频开启形式"
```

可选

一旦在IE浏览器中为image类型的input元素指定了dynsrc属性，那么该元素即可播放视频剪辑。此时，视频有两种开启方式，一种是在视频文件加载完成后立即开始播放，另一种则是当用户鼠标指针位于图像框之上时才开始播放。通过start属性，可以选用最适合于页面的启动方式。

input类型 image

示例

```
<input type="image dynsrc="submit.avi" alt="Submit Button" loop="1" start="mouseover"
height="100" width="150">
```

值 两个常量之一：fileopen | mouseover，使用时均不区分大小写。

默认值 fileopen

对象模型引用方式

```
[window.]document.formName.inputName.start
[window.]document.forms[i].elements[j].start
[window.]document.getElementById(elementID).start
```

step

IE n/a NN n/a Moz n/a Saf n/a Op 9 HTML n/a

```
step="精度"
```

可选

Web Forms 2.0 扩展，它可为input元素指定允许使用的递增值。设置了min或/和max属性后，这两个属性就限制了用户输入数据的范围，否则就会以0作为取值范围。须要注意的是，对日期相关的元素而言，起始点是“1970-01-01T00:00:00.0Z”。

input类型 date、datetime、datetime-local、file、month、number、range、time、week

示例

```
<input type="time" name="apptTime" min="09:00" max="17:00" value="09:00" step="900"/>
```

值 number和range类型，整数或浮点数；date、week和month类型，表示天、周或月的整数值；时间与日期的组合类型，则使用秒数。

默认值 60（datetime、datetime-local、time）；1（其他类型）。

target

IE n/a NN n/a Moz n/a Saf n/a Op 9 HTML n/a

```
target="窗口或框架名称"
```

可选

input元素被点击后，这个Web Forms 2.0扩展允许其内嵌表单的返回点与常规表单有所不同，使得其返回页面出现在另一个窗口或框架之中。

input类型 image、submit

template

IE n/a NN n/a Moz n/a Saf n/a Op 9 HTML n/a

```
template="重复块 ID"
```

可选

这个Web Forms 2.0 扩展可将一个add类型的input元素与另一个元素（通常是tr元素）联系起来，并将它作为一个模板，不断地向表单中增加整块的内容。此时，在重复使用的元素中，须要将其repeat属性设置为

“template”，而add元素的template属性则应指向该重复元素的ID。例如，点击add元素后，就会向表格中增加新的一行，该行的单元格排列方式与模板对象完全相同。

input类型 add

示例

```
<form>
  <table>
    <tr>
      <th>Product</th>
      <th>Quantity</th>
    </tr>
    <tr id="order" repeat="template">
      <td><input type="text" name="row[order].product" value=""></td>
      <td><input type="text" name="row[order].quantity" value="1"></td>
    </tr>
  </table>
  <p><button type="add" template="order">Add Row</button></p>
  <p><button type="submit">Submit</button></p>
</form>
```

值 重复元素的id属性值。

默认值 无。

type

IE *all* NN *all* Moz *all* Saf *all* Op *7* HTML *all*

type="元素类型"

必选

此属性用于通知浏览器如何显示input元素，或者仅仅是告知浏览器是否应显示该元素。这里列出了所有浏览器中均可使用的各种选择。

类型	描述
button	可点击的按钮，须要通过脚本实现其响应动作。通过value属性可以设定按钮上的标注文字。如果要使用HTML设计按钮上的文本，那么请直接使用button元素
checkbox	独立的复选框，它能提供两种状态以供选择：激活/开启、非激活/关闭。通过input元素标签之前或之后的HTML文本，可以设定其标注文字。请参见lable元素。value属性的属性值会同表单一同提交
file	包含一个按钮和一个文本域，使得用户可以选择一个本地文件，并上传到服务器端。点击该按钮后，会出现一个文件选择对话框，用户选择文件后，该文件名称或路径会显示在文本域中。在提交时，表单中action属性指定的URI会激活服务器端的CGI脚本程序，从而接收上传文件
hidden	隐藏域，往往用于在几次连续的提交过程中缓存数据库或状态数据。这样一来，用户就不用反复填写重复的内容，服务器也不必存储临时数据。名称/值对会同表单一同提交
image	一种图形化的按钮，其唯一的默认动作就是提交表单。点击图像时对应的坐标点（x,y）会作为两个名称/值对提交至服务器端，其形式如下：inputName.x= n&inputName.y=m
password	也是一种文本域，但用户输入的每个字符都显示为一个圆点（·）或星号（*），以防止其他人偷窥到用户的个人隐私。元素中的明语字符会作为其值提交至服务器端
radio	开/关按钮关联组中一个按钮。将多个单选按钮的name属性赋以相同值后，可将它们组合成一个关联组。通过点击激活组内某个按钮后，其他按钮将全部处于非选中状态。已激活按钮的value属性值会同表单一同提交
reset	该按钮的唯一工作就是将表单中的元素恢复到加载完成时的初始状态。通过value属性可以自定义按钮标注的文字内容

续表

类型	描述
submit	一种图形化的按钮，其唯一的工作就是提交表单。通过value属性可以自定义按钮标注的文字。如果已为该元素指定了name和value属性，那么这些属性值也会随着表单一并提交
text	单行文本域，用户可在其中输入文本。这些文本将作为元素属性值，以URL编码的形式提交至服务器端。如果要使用多行文本域，请参见textarea元素

Apple公司在Safari 2中还支持另外两种类型——search和range，但这两种类型主要面向Dashboard小程序的用户，并没有应用在网页上。

Web Forms 2.0标准还引入了一些智能输入类型，它们大多都提供了一些用户接口功能，以便用户输入正确的表单数据。当然，目前HTML的原生类型也在数据输入校验功能上得到了一定的加强。下表中罗列了这些新的类型。

类型	描述
add	可点击的按钮，点击后可反复插入已编写好的 HTML 内容块
date	日期的专用文本输入域，其格式由 ISO 8601 指定，如 2008-04-04
datetime	日期和 UTC 时间的专用文本输入域，其格式由 ISO 8601 指定，如 2008-04-04T21:45:00Z
datetime-local	日期和本地时间的专用文本输入域，其格式由 ISO 8601 指定，如 2008-04-04T21:45:00
email	E-mail 地址的专用文本输入域，其格式由 RFC 2882 指定
month	年份与月份数字组合的专用文本输入域，如 2009-03
move-down	可点击的按钮，点击后可将最接近它的上级重复块下移一格，例如，将表格中的某行下移一行
move-up	可点击的按钮，点击后可将最接近它的上级重复块上移一格，例如，将表格中的某行上移一行
number	仅能输入数字的专用文本输入域
range	辅助用户在预定义的范围内选择某个值。具体的设计实现还取决于浏览器生产商，而滑块控件是其中最常见的一种
remove	可点击的按钮，点击后将移除某个 HTML 重复块
time	时间的专用文本输入域，其格式由 ISO 8601 指定，如 21:47:00
url	互联网地址的专用文本输入域，其格式由 RFC 3987 指定，请参见 *http://www.ietf.org/rfc/rfc 987.txt*
week	年份与星期组合的专用文本输入域，如 2007W22

input类型 所有input类型均可使用此属性。

示例

```
<input type="button" value="Toggle Sound" onclick="toggleSnd( )">
<input type="checkbox" name="connections" value="ISDN">ISDN
<input type="file" name="uploadFile">
<input type="hidden" name="prevState" id="prevState" value="modify">
<input type="image" name="graphicSubmit" src="submit.jpg" height="40" width="40">
<input type="password" name="password" maxlength="12" size="20">
<input type="radio" name="creditCard" value="Visa">Visa
<input type="reset">
<input type="submit" value="Send Encrypted">
Social Security Number:<input type="text" name="ssn" value="###-##-####"
onclick="validateSSN(this)">
```

值 任何已知的input元素类型：button | checkbox | file | hidden | image | password | radio | reset | submit | text。

默认值 text

对象模型引用方式

```
[window.]document.formName.inputName.type
[window.]document.forms[i].elements[j].type
[window.]document.getElementById(elementID).type
```

usemap

IE 6 NN n/a Moz all Saf all Op 7 HTML 4

```
usemap="映射 URL"
```

可选

通过map和area元素，可以为image类型的input元素定义一个客户端图像映射。map元素可看作一个已命名的容器，其中容纳了一个或多个area元素。每个area元素均在图像上设定了一个热点区域，并且指定了对应的链接目标（或其他设定），当用户点击该区域时就会做出相应的响应。而usemap属性的作用就是将同一个文档中image类型的input元素和一个已命名的map元素联系起来。这样一来，map元素的名称即可视为一个定位标记，该map元素的地址由#和元素名称组成。

input类型 image

示例

```
<input type="image" src="submit.gif" alt="Submit Button" usemap="#submitter" height="90"
width="120">
```

值 位于同一文档中的map元素URL地址，即#符号加上map元素的名称。

默认值 无。

对象模型引用方式

```
[window.]document.formName.inputName.useMap
[window.]document.forms[i].elements[j].useMap
[window.]document.getElementById(elementID).useMap
```

value

IE all NN all Moz all Saf all Op 7 HTML all

```
value="文本"
```

可选/必选

为input元素预先指定一个值，它将作为名称/值对的一部分提交至服务器。尽管并不会提交某些input元素类型（如，未选择的单选按钮），但一旦提交了这些元素（button和reset类型的input元素除外），就能将其预定义值传送至服务器端。

对text和password类型的input元素而言，value属性包含了其默认输入。如果用户改变了文本域中的内容，value值也会随之发生改变，但此时源代码依然保持原样。点击reset按钮重置表单后，默认值就会重新显示在文本域中。

目前仅checkbox和radio类型的input元素必须使用value属性。对那些显示为标准可点击按钮的input元素而言，value属性定义了按钮上文字标注的内容。

input类型 所有input类型均可使用此属性。

示例 `<input type="checkbox" name="connections" value="ISDN">ISDN`

值 任何文本字符串。

默认值 无。

对象模型引用方式

```
[window.]document.formName.inputName.defaultValue
[window.]document.forms[i].elements[j].defaultValue
[window.]document.getElementById(elementID).defaultValue
[window.]document.formName.inputName.value
[window.]document.forms[i].elements[j].value
[window.]document.getElementById(elementID).value
```

vcard_name

IE 5 NN n/a Moz n/a Saf n/a Op n/a HTML n/a

vcard_name=" *vcard域*" 可选

如果IE用户在选项中允许表单自动完成这一功能，那么autocomplete属性就能控制整个表单的填写。如果表单域所需的数据可能保存在用户的电子名片（vCard）中，那么可以为“text”和“password”类型的input元素指定vcard-name属性，如果浏览器找到了与其名称相符的用户选项条目，就会提前或辅助完成该表单域的填写工作。关于HTML表单中如何实现自动完成的更多细节，请访问此链接：*http://msdn.microsoft.com/en-us/library/ms533032 (VS.85).aspx*。

input类型 password、text

示例

```
<input type="text" name="homephone" vcard_name="vCard.Home.Phone" autocomplete="on">
```

值 以下常量：vCard.Business.City | vCard.Business.Country | vCard.Business.Fax | vCard.Business.Phone | vCard.Business.State | vCard.Business.StreetAddress | vCard.Business.URL | vCard.Business.Zipcode | vCard.Cellular | vCard.Company | vCard.Department | vCard.DisplayName | vCard.Email | vCard.FirstName | vCard.Gender | vCard.Home.City | vCard.Home.Country | vCard.Home.Fax | vCard.Home.Phone | vCard.Home.State | vCard.Home. StreetAddress | vCard.Home.Zipcode | vCard.Homepage | vCard.JobTitle | vCard.LastName | Card.MiddleName | vCard.Notes | vCard.Office | vCard.Pager。

默认值 无。

对象模型引用方式

```
[window.]document.formName.inputName.vcard_name
[window.]document.forms[i].elements[j].vcard_name
[window.]document.getElementById(elementID).vcard_name
```

vspace

参见hspace属性。

width

参见height属性。

<ins>

IE 4 NN n/a Moz all Saf all Op 7 HTML 4

<ins>...</ins> HTML结束标签：必选

ins元素与其配套的del元素定义了一种格式，用于表示在编辑过程中已经标识为删除或插入的那部分内容。虽然这已远离了工作流管理方案的范畴，但在使用一款支持所见即所得的HTML编辑工具时，这些元素有助于在编辑过程中控制每一轮的修改内容。

在显示时，支持此元素的浏览器会为元素中包含的文本加上下划线，而del元素中的文本则会加上删除线。HTML 4规范为了保存修改时间和修改注释等隐藏信息，还引入了另外两种可能有用的属性。

示例

```
<p>Four score and <del cite="Fixed the math">eight</del><ins>seven</ins> years ago...</p>
```

对象模型引用方式 [window.]document.getElementById(*elementID*)

元素专有属性 cite、datetime

元素专有事件处理属性 无。

cite

IE 6 NN n/a Moz all Saf all Op 7 HTML 4

```
cite="文本"
```

可选

此属性用于说明修改的原因或记录与元素相关联的其他信息，通常并不显示此信息。在Mozilla的上下文菜单中，为此元素提供了一个属性选项，这样就可以将元素内的属性与属性值都呈现给网页的浏览者。DHTML脚本也可以通过元素对象的cite属性访问这些信息，并添加描述信息。

示例 `<ins cite="Fixed the math A.L.">seven</ins>`

值 任意字符串。该字符串应该位于一对引号之中，双引号是推荐的标准格式，也可使用单引号。

默认值 无。

对象模型引用方式 `[window.]document.getElementById(elementID).cite`

datetime

IE 6 NN n/a Moz all Saf all Op 7 HTML 4

```
datetime="时间字符串"
```

可选

此属性表示插入时的日期与时间。一般来说，只有使用具备修改记录功能的HTML编辑工具，才会在文档中添加这种信息。日后在进行文档修改的审核跟踪时，就可以调用属性中记录的修改时间。在一个给定的ins元素中，只能存在一个相关的datetime属性值。在Mozilla的上下文菜单中，为此元素提供了一个属性选项，这样就可以将元素内的属性与属性值都呈现给网页的浏览者。另外，DHTML脚本也可以通过元素对象的datetime属性访问这些信息，并添加描述信息。

示例 `<ins datetime="2001-09-11T08:56:00-04:00">SomeInsertedTextHere</ins>`

值

datetime属性需要一个特殊的日期-时间格式，这样无论处于世界上哪一个时区，都可以根据这个信息推算出对应于的准确时刻。该语法格式为：*yyyy-MM-ddThh:mm:ssTZD*。

yyyy

四位数字年份。

MM

两位数字月份（从01至12）。

dd

两位数字日期（从01至31）。

T

使用大写字母将时间和日期分隔开来。

hh

两位数字的24小时制小时（从00至23）。

mm

两位数字分钟（从00至59）。

ss

两位数字秒（从00至59）。

TZD

时区标志符。

时区标志符有两种格式。第1种格式仅使用大写字母，它代表UTC（Coordinated Universal Time，协调世界时，也称为“Zulu”）。另一种时间格式则使用*hh:mm:ss*，它指出相对于UTC的偏移量。这个时间偏移量还包含

一个“+”或“-”，以及另一对*hh:mm*值。如果时区在格林尼治标准时间（Greenwich Mean Time，简称GMT，它实际上与UTC相同）以西，那么该地区时间要早于UTC，因此其偏移量运算符使用“-”。同理，如果在GMT以东，则偏移量使用正值。例如，太平洋标准时间（简称PST）比UTC早8小时，那么在PST时区为18:00时，UTC是第二天的凌晨2点。因此，以下这些例子均表示时间中的同一时刻（注意，出于突出的目的才加粗显示时区标志符，在实际使用中并不需要这样）：

2003-01-30T02:00:00**Z**	UTC	2003-01-29T18:00:00-**08:00**	Pacific Standard Time
2003-01-29T21:00:00-**05:00**	Eastern Standard Time	2003-01-30T13:00:00+**11:00**	Sydney, Australia

请参考ISO-8601标准以获取更多关于时间表示方法的细节信息。

默认值 无。

<isindex>

IE *all* NN *all* Moz *all* Saf *all* Op *n/a* HTML *all*

```
<isindex>
```

HTML结束标签：禁用

isindex是早期HTML遗留下来的一个元素，目前HTML 4已不赞成使用此元素，建议使用text类型的input元素取代该元素。isindex元素标签位于head元素内。在现代浏览器中，一般将其作为普通的文本域显示在两个hr元素之中。当用户向其中输入了文本内容，并按下回车键，那么会对文本域中的内容进行URL编码（使用“+”号代替空格），并同当前文档的URL一起发送至服务器端。服务器端的某个CGI程序必须知晓如何处理这个URL，并且还要返回一个HTML以显示在当前的窗口或框架中。

示例 `<head><isindex prompt="Enter a search string:"></head>`

对象模型引用方式 `[window.]document.getElementById(elementID)`

元素专有属性 `prompt`

元素专有事件处理属性 无。

prompt

IE *all* NN *all* Moz *all* Saf *all* Op *n/a* HTML *4*

```
prompt="信息"
```

可选

此属性用于指定与元素一同出现的提示信息。

示例 `<isindex prompt="Enter a search string:">`

值 任何使用引号括起的字符串。

默认值 无。

<kbd>

IE *all* NN *all* Moz *all* Saf *all* Op *7* HTML *all*

```
<kbd>...</kbd>
```

HTML结束标签：必选

在HTML 4推荐标准中，有一类元素统称为短语元素（phrase element），kbd元素也是其中之一。这些元素为文档的特定部分指明了结构上的含义。如果可以预计到用户将在键盘上输入某些文本，以填写文本域或发出某些命令，那么就可以使用kbd元素来显示这些文本内容。

浏览器可以自由决定如何（或是否）从其他body元素中区分出kbd元素的内容。大多数的浏览器选择使用等宽字体来显示这些文字。可以使用任何合适的样式单来替换默认的显示形式。

示例 `<p>If you don't know the answer, type <kbd>NONE</kbd> into the text box.</p>`

对象模型引用方式 `[window.]document.getElementById(elementID)`

元素专有属性 无。

元素专有事件处理属性 无。

<keygen>

IE *n/a* NN *3* Moz *all* Saf *all* Op *7* HTML *n/a*

<keygen> HTML结束标签：禁用

当服务器希望通过加密密匙封装表单时，keygen元素可以帮助表单在提交时使用密匙进行加密。此时，客户端浏览器必须安装数字证书。如果在form元素中加入了keygen元素，那么用户将看到两个不同的效果。首先，在表单中keygen元素出现的位置，会显示当前可用加密密匙长度的选择列表。另外，当用户提交表单时，用户可能看到一个或多个安全相关的确认对话框。此元素基于Navigator的公用密匙加密系统和Netscape的证书管理系统（CMS）而构建。请参见其参考文档：*http://www.redhat.com/docs/manuals/cert-system/admin/overview.htm*。

171

示例

```
<form ...>
...
<keygen name="encryptedOrder" challenge="39457582201">
</form>
```

元素专有属性 challenge、keytype、name、pqg

元素专有事件处理属性 无。

challenge

IE *n/a* NN *3* Moz *all* Saf *all* Op *7* HTML *n/a*

challenge="口令字符串" 必选

如果服务器能够解析口令字符串以便校验加密数据包，那么就可以在challenge属性中设置口令字符串。如果此属性的值为空字符串，那么会将密匙编码为一个IA5STRING。

示例 <keygen name="encryptedOrder" challenge="39457582201">

值 任意字符串。

默认值 空字符串。

keytype

IE *n/a* NN *3* Moz *all* Saf *n/a* Op *n/a* HTML *n/a*

keytype="密匙类型" 可选/必选

此属性用于设定密匙类型，在提交表单数据时CMS可依此设置生成密匙。另外，仅当次级类型为“DSA”时才须要使用此属性。

示例 <keygen name="encryptedOrder" challenge="39457582201" keytype="DSA">

值 RSA或DSA。

默认值 RSA

name

IE *n/a* NN *3* Moz *all* Saf *all* Op *7* HTML *n/a*

172

name="标识符" 必选

可将整个表单数据编码为一个值，从而成为名称/值对中的一部分提交至服务器。而name属性则指定了该名称/值对中的“名称”部分。如果服务器成功地对加密包进行了解码，那么服务器就能进一步处理表单中相关元素的名称/值对。

示例 <keygen name="encryptedOrder" challenge="39457582201">

值 该标识符不区分大小写。

默认值 无。

pqg

IE *n/a* NN *3* Moz *all* Saf *n/a* Op *n/a* HTML *n/a*

`pqg="dss 参数"` 可选/必选

如果keytype属性的密匙类型为DSA，那么就必须为pgg属性也指定相关的参数值。获取这些数值的算法说明可参见：*ftp://ftp.ietf.org/internet-drafts/draft-ietf-pkix-ipki-pkalgs-05.txt*。

<label>

IE *4* NN *n/a* Moz *all* Saf *all* Op *7* HTML *4*

`<label>...</label>` HTML 结束标签：必选

通过label元素，可以为input元素的标注文字定义结构和容器。由于在表单控件的标签内，往往没有可显示的标注文字，通过label元素可以为网页作者提供一个途径，以便他们为控件附上相关的说明内容。同时，使用label元素可以简便地统一表单内所有标注文字的样式。

有两种方式可以为表单控件添加标注说明。第1种方式须要将控件的id属性值赋给label元素的for属性。而第2种方式则须要将input元素内嵌在一个label元素之内。值得注意的是，如果标注和控件都是正文主体的一部分，那么才能使用第2种方式，但当它们彼此独立，且分别位于表格内不同的td元素之内时，就必须使用第1种方式。而标注是显示在控件之前还是之后，则完全由标签在源代码中的相对位置决定。

示例

```
<form>
<label>Company:<input type="text" name="company"></label><br>
<label for="stateEntry">State:</label>
<input type="text" name="state" id="stateEntry">
...
</form>
```

对象模型引用方式 `[window.]document.getElementById(elementID)`

元素专有属性 accesskey、datafld、dataformatas、datasrc、for

元素专有事件处理属性 无。

accesskey

IE *4* NN *n/a* Moz *all* Saf *all* Op *7* HTML *4*

`accesskey="某字符"` 可选

一个字符键，通过它和其他组合键可以聚焦或激活一个相关的input元素。本章前文的共享属性中已进行了详细说明。

示例 `<label for="stateEntry" accesskey="s">State:</label>`

值 单个字符。

默认值 无。

对象模型引用方式 `[window.]document.getElementById(elementID).accessKey`

datafld

IE *4* NN *n/a* Moz *n/a* Saf *n/a* Op *n/a* HTML *n/a*

`datafld="列名称"` 可选

此属性用于IE中的数据绑定，以便将一个远程数据源的列名与input元素标注联系在一起。数据源的对应列必须是纯文本或HTML，请参见dataformatas。同时，label元素中还必须设定datasrc属性。注意，只能在IE 5/Mac操作系统下实现文本文件数据源的绑定。

示例

```
<label for="stateEntry" datasrc="DBSRC3" datafld="label" dataformatas="HTML"> </label>
```

值 区分大小写的标识符。

默认值 无。

对象模型引用方式 [window.]document.getElementById(*elementID*).dataFld

dataformatas

IE *4* NN *n/a* Moz *n/a* Saf *n/a* Op *n/a* HTML *n/a*

```
dataformatas="数据类型"
```

可选

本属性用于IE浏览器中的数据绑定。对于来自于数据源的原始资料，浏览器根据此属性值决定将它们视为纯文本亦或是HTML标签。本属性的设置完全依赖于数据源的构造方式。注意，只能在IE 5/Mac操作系统下实现文本文件数据源的绑定。

示例


```
<label for="stateEntry" datasrc="DBSRC3" datafld="label" dataformatas="HTML"> </label>
```

值 IE浏览器可识别两种有效的设定值：text和html。

默认值 text

对象模型引用方式 [window.]document.getElementById(*elementID*).dataFormatAs

datasrc

IE *4* NN *n/a* Moz *n/a* Saf *n/a* Op *n/a* HTML *n/a*

```
datasrc="数据源名称"
```

可选

为实现IE数据绑定功能，此属性用于指定页面内object元素的ID值，以便在进行远程数据访问时加载数据源对象。通过datafld属性指定来自于数据源中的绑定内容。注意，只能在IE 5/Mac操作系统下实现文本文件数据源的绑定。

示例

```
<label for="stateEntry" datasrc="DBSRC3" datafld="label" dataformatas="HTML"> </label>
```

值 区分大小写的标识符。

默认值 无。

对象模型引用方式 [window.]document.getElementById(*elementID*).dataSrc

for

IE *4* NN *n/a* Moz *all* Saf *all* Op *7* HTML 4

```
for="input 元素标识符"
```

可选

此属性是一个独一无二的标识符，它与input元素的id属性值相等，从而将此label元素与对应的input元素联系起来。如果未将input元素内嵌在label元素内，那么才须要使用for属性，此时，for属性可看作是这两个元素之间的联系纽带。

示例 `<label for="stateEntry">State:</label>`

值 区分大小写的标识符。

默认值 无。

对象模型引用方式 [window.]document.getElementById(*elementID*).htmlFor

<layer>

IE *n/a* NN *|4|* Moz *n/a* Saf *n/a* Op *n/a* HTML *n/a*

```
<layer>...</layer>
```

HTML结束标签：必选

在Navigator 4的对象模型中，layer是一种可定位的元素。它们类似于块级元素，其CSS的position属性值为“absolute”。因此，layer元素的很多属性都是根据Navigator 4的定位、尺寸调整和可定位元素的排列

方式来命名的。迁移到Mozilla后，浏览器废除了这个元素，同时，新浏览器和W3C标准也未实现此元素。

layer元素的内容即可以从src属性指定的单独文件中读取，也可以使用起始和结束标签包含在当前文档中，还可以在同一个layer元素中同时引入这两种类型的内容。src属性指定的文档内容首先显示，而后，附加的内容才出现在外部文档内容的矩形框之后的一行中。

layer元素可以出现在文档中的任意位置，它还可以覆盖其他层的内容，其中也包括文档的基础层，如body等。与链接或脚本控件不同的是，独立的层内容可以在不重载其他页面内容的情况下单独改变。此外，还可以嵌套使用layer元素。

示例 `<layer bgcolor="yellow" src="instrux.html" width="200" height="300"></layer>`

对象模型引用方式 `[window.]document.layerName`

元素专有属性 above、background、below、bgcolor、clip、height、id、left、pagex、pagey、src、top、visibility、width、z-index

元素专有事件处理属性

处理程序	NN	IE/其他	HTML	处理程序	NN	IE/其他	HTML
onblur	4	n/a	n/a	onmouseout	4[a]	n/a	n/a
onfocus	4	n/a	n/a	onmouseover	4[a]	n/a	n/a
onload	4	n/a	n/a	onmouseup	4[a]	n/a	n/a
onmousedown	4[a]	n/a	n/a				

a 仅用于事件捕获模式。

above

IE *n/a* NN *|4|* Moz *n/a* Saf *n/a* Op *n/a* HTML *n/a*

`above="层 ID"` 可选

此属性指出在叠放次序中，放置在当前layer元素之前的可定位元素。这是设置z-index属性的一种变相方法，但它并不依靠随机编号体系。如果使用了above属性，那么就不要在同一个layer元素中使用below或z-index属性。

示例

```
<layer id="instrux" bgcolor="yellow" src="instrux.html" above="help1" width="200"
height="300"></layer>
```

值 区分大小写的标识符。

默认值 无。

对象模型引用方式 `[window.]document.layerName.above`

background

IE *n/a* NN *|4|* Moz *n/a* Saf *n/a* Op *n/a* HTML *n/a*

`background="URL"` 可选

此属性指定了一个图像文件，它将作为layer元素中文本和其他内容的背景图。与浏览器载入的其他普通图像不同的是，背景图会以未经缩放的原始大小进行加载，然后贴附并充满整个可用的页面空间。小图片虽然下载迅速，但总是会很明显地在背景中重复显示。也可以使用GIF动画作为背景图，但这实在是一件令读者抓狂的做法。在选择背景图时，一定要选择相对柔和的图片，以便使主体内容更加引人注目。如果一定要使用背景图，那么请选用非常淡雅的图片。

示例

```
<layer background="blueCrinkle.jpg" src="instrux.html" width="200" height="300"> </layer>
```

值 任何有效的图像文件URL地址，既可使用相对URL，也可以使用绝对URL地址。

默认值 无。

对象模型引用方式 [window.]document.*layerName*.background

below

IE *n/a* NN |4| Moz *n/a* Saf *n/a* Op *n/a* HTML *n/a*

`below="层 ID"` 可选

此属性指出在叠放次序中，放置在当前layer元素之后（之下）的可定位元素。这是设置z-index属性的一种变相方法，但它并不依靠随机编号体系。如果使用了below属性，那么就不要在同一个layer元素中同时使用above或z-index属性。

示例

177

```
<layer bgcolor="yellow" src="instrux.html" width="200" height="300" below="thankyou">
</layer>
```

值 区分大小写的标识符。

默认值 无。

对象模型引用方式 [window.]document.*layerName*.below

bgcolor

IE *n/a* NN |4| Moz *n/a* Saf *n/a* Op *n/a* HTML *n/a*

`bgcolor="颜色三元组或颜色名"` 可选

此属性为整个矩形层设置填充色，此颜色位于文本和其他内容之下。如果同时使用bgcolor和background属性，那么背景色会透过背景图的任何透明部分。

示例 `<layer bgcolor="yellow" src="instrux.html" width="200" height="300"></layer>`

值 一个十六进制的三元组或明语颜色名称。不设置属性值就默认为黑色（#000000，black）。目前可用的明语颜色名称请参见附录A。

默认值 由操作系统决定。

对象模型引用方式 [window.]document.*layerName*.bgColor

clip

IE *n/a* NN |4| Moz *n/a* Saf *n/a* Op *n/a* HTML *n/a*

`clip="[左侧像素值，顶端像素值,] 右侧像素值，底部像素值"` 可选

裁剪区域是整个layer内容中的矩形可见区域。只能看见裁剪域矩形框内的页面内容。clip属性的默认值可能是其内容的默认尺寸，也可能是layer元素的宽度乘以连续内容的长度。设置clip后，可以控制住过长的内容，如果它们的尺寸超出了页面设计时确定的矩形区域，那么不会显示出超出的部分。

示例

```
<layer bgcolor="yellow" src="instrux.html" clip="50,50" width="200" height="300"> </layer>
```

值 当元素进入文档时，从元素的左上边缘处开始计算clip属性的像素值。这些值的属性从左边缘开始，依顺时针方向环绕矩形框四周，分别为：左、上、右、下。如果仅仅提供两个值，那么Navigator 4会假定左侧和上端属性值为0，这意味着只会调整右侧和底部边缘。因此，设置为“50,50”时，意味着裁剪区域是一个从层左上角起始的矩形块，面积为50像素×50像素。如果期望保持可视面积不变，但从距离左边缘10像素的位置开始创建区域，那么clip属性就应设置为“10,0,60,50”。

178

默认值 layer内容的自然可视空间大小。

对象模型引用方式

```
[window.]document.layerName.clip.left
[window.]document.layerName.clip.top
```

```
[window.]document.layerName.clip.right
[window.]document.layerName.clip.bottom
```

height, width

IE *n/a* NN |4| Moz *n/a* Saf *n/a* Op *n/a* HTML *n/a*

```
height="长度值" width="长度值"
```

可选

这两个属性定义了layer元素的最小尺寸。然而，在初始加载过程中一旦向层中注入了内容，那么这两个属性设置依然无法限制可见空间的尺寸，在x和y轴方向上均会超出限定值。例如，如果要在layer中显示一个长为90像素、宽为120像素的图像，那么当layer元素的height和width属性值过小时，就会扩展该层的可视尺寸，以便显示整张图像。对文本或其他内容也会同样处理，为了显示全部内容，将扩展可视空间。为限制内容的可见部分，请设置clip属性。

在其他已定位的内容下创建彩色或有图案的矩形（使用bgcolor或background属性）作为衬底时，为height和width属性设置特定的大小是很有用的。当没有任何内容要溢出layer边框时，裁剪域将保持height和width属性指定的尺寸。

示例 `<layer bgcolor="yellow" src="instrux.html" width="200" height="300"></layer>`

值 正整数或百分比。将这两个属性值减小为0时，不仅可以掩藏元素（也可以通过visibility属性到达这一目的），还可以防止元素占据任何的页面空间。

默认值 layer内容的自然可视空间大小。

对象模型引用方式

```
[window.]document.layerName.height
[window.]document.layerName.width
```

id

IE *n/a* NN |4| Moz *n/a* Saf *n/a* Op *n/a* HTML *n/a*

```
id="元素标识符"
```

可选

此属性是一个独一无二的标识符，使所属元素区别于文档中所有的其他元素。此标识符可被用作above和below属性的属性值。脚本也可以使用id属性值作为layer元素的名称，以便实现对象引用。

示例

```
<layer id="oldYeller" bgcolor="yellow" src="instrux.html" width="200" height="300"> </layer>
```

值 区分大小写的标识符。

默认值 无。

对象模型引用方式 `[window.]document.layerName.name`

left, top

IE *n/a* NN |4| Moz *n/a* Saf *n/a* Op *n/a* HTML *n/a*

```
left="像素值" top="像素值"
```

可选

这两个属性定义了层的左、上边缘与其原有显示位置相比的偏移量。除非通过pagex和pagey属性设定了元素在文档中的绝对位置，否则其准确位置还会发生变化。Navigator对layer的处理不同于ilayer，它不会在文档中为layer元素保留显示空间。如果未将layer元素放置在其他位置，那么它会被放置在其自身的平面内，而周围的源代码内容则会被收缩起来，覆盖在layer元素的内容之上。

示例

```
<layer bgcolor="yellow" src="instrux.html" width="200" height="300" left="10" top="50">
</layer>
```

值 正整数值，可不使用引号。

默认值 0

对象模型引用方式

```
[window.]document.layerName.left
[window.]document.layerName.top
```

pagex, pagey

IE *n/a* NN |4| Moz *n/a* Saf *n/a* Op *n/a* HTML *n/a*

```
pagex="像素值" pagey="像素值"
```

可选

为了精确定位layer元素，可以使用文档（即页面）的左上角作为参考点。设置了pagex和/或pagey属性后，就可以确定layer元素左、上边缘与整个文档对应边缘之间的偏移量。因此，对一个可在竖直方向滚动的页面而言，其零点也能位于浏览器窗口的可视区域之上。

示例

```
<layer bgcolor="yellow" src="instrux.html" width="200" height="300" pagex="50" pagey="350">
</layer>
```

值 正整数值，可不使用引号。

默认值 0

对象模型引用方式

```
[window.]document.layerName.pageX
[window.]document.layerName.pageY
```

src

IE *n/a* NN |4| Moz *n/a* Saf *n/a* Op *n/a* HTML *n/a*

```
src="URL"
```

可选

为了在layer元素中加载外部HTML文件，须要在src属性中指定该文件的URL地址。在layer中，首先显示从src指定的URL中载入的内容，紧随其后的才是元素的起始和结束标签中包含的HTML内容。如果忽略了src属性，那么只会显示标签之间的内容。layer加载文档后，仍然可以通过脚本改变对应的对象属性（即src），以便在不重载主文档的情况下动态改变layer元素中的内容。

示例 `<layer bgcolor="yellow" src="instrux.html" width="200" height="300"></layer>`

值 完整或相对URL路径。

默认值 无。

对象模型引用方式 `[window.]document.layerName.src`

top

参见left属性。

visibility

IE *n/a* NN |4| Moz *n/a* Saf *n/a* Op *n/a* HTML *n/a*

```
visibility="可见性常量"
```

可选

Navigator 4根据此属性决定是否显示layer元素。在默认情况下，此图层会继承其临近的最外层图层的visibility属性设置。layer元素是文档主体的一部分，由于主体通常是可见的，因此layer在默认情况下也是可见的。如果在页面加载时须要隐藏某个层，那么请将其visibility属性设置为“hidden”。只有当layer元素内嵌在另一个layer或ilayer元素中，并且该元素的visibility值为“hidden”（或继承其父层的“hidden”值）时，才须要将layer元素的visibility设置为“hidden”。

示例

```
<layer bgcolor="yellow" src="instrux.html" width="200" height="300" pagex="50" pagey="350"
visibility="hidden"></layer>
```

值 选用以下常量之一：hidden | inherit | visible。

默认值 inherit

对象模型引用方式 [window.]document.*layerName*.visibility

181

width

参见height属性。

z-index

IE *n/a* NN |4| Moz *n/a* Saf *n/a* Op *n/a* HTML *n/a*

```
z-index="层号"
```

可选

对文档内所有的层而言，此属性可以控制它们相对于临近的最外层层容器在Z轴上的叠放次序。如果在同一个容器（如文档的基础层）内，两个或多个可定位元素拥有相同的z-index值，那么HTML源代码中元素的加载顺序就决定了页面上元素的叠放次序，较晚加载的元素将放置在较早加载的元素之前。而在所有可定位元素中，z-index的默认值均为0。因此，如果希望将一个可定位元素放置在其他所有元素之前，只须要简单地将它的z-index属性设置为任意一个正整数。通过脚本可以实时调整可定位元素的叠放次序。请参见above和below元素。

示例

```
<layer bgcolor="yellow" src="instrux.html" width="200" height="300" z-index="1"> </layer>
```

值 任意一个整数值。

默认值 0

对象模型引用方式 [window.]document.*layerName*.zIndex

<legend>

IE *4* NN *n/a* Moz *all* Saf *all* Op *7* HTML *4*

```
<legend>...</legend>
```

HTML结束标签：必选

对fieldset元素而言，legend元素就像是它的一个标注。在可视化的浏览器中，这通常意味着fieldset元素的外框上也会显示相关的标注内容。同时，当用户操作表单时，具有文本-语音转换功能的浏览器可能还会大声读出标注内容。为了使得标注内容更为醒目，最好将legend元素就放置在fieldset元素的起始标签之后。由于legend元素的内容也是HTML内容，因此在需要的时候可以通过样式单使标注突出出来。

示例

```
<form method="POST" action="...">
<fieldset>
<legend>Credit Card Information</legend>
...inputElementsHere...
</fieldset>
</form>
```

182

对象模型引用方式 [window.]document.getElementById(*elementID*)

元素专有属性 accesskey、align

元素专有事件处理属性 无。

accesskey

IE *5* NN *n/a* Moz *all* Saf *n/a* Op *n/a* HTML *4*

```
accesskey="某字符"
```

可选

一个单独的字符键，通过它可聚焦或激活表单中第一个与legend元素相关联的可聚焦控件。请参见本章前文，在共享属性中已进行了详细说明。

示例 `<legend accesskey="c">Credit Card Information</legend>`

值 单个字符。

默认值 无。

对象模型引用方式 `[window.]document.getElementById(elementID).accessKey`

align

IE 4 NN n/a Moz all Saf n/a Op n/a HTML 4

`align="对齐位置"` 可选

此属性用于控制fieldset元素内legend元素的对齐状态。有关容器盒内文本对齐的相关内容，请参考本章前文中的“对齐常量”部分。

示例 `<legend align="right">Credit Card Information</legend>`

值 HTML 4中的可用值为：bottom | left | right | top。另外，IE 4及后续版本和Mozilla浏览器还增加了另一个值——center。

默认值 left

对象模型引用方式 `[window.]document.getElementById(elementID).align`

IE all NN all Moz all Saf all Op 7 HTML all

`<li>...</li>` HTML结束标签：可选

li元素是一种独立的列表项，通常内嵌在ol或ul列表容器中。而是否在li项前加上有序的数字、字符或无顺序含义的符号标示，则由其外部容器决定。同时，还有一类特殊的样式单属性被用来设计列表格式。

但是，如果在Navigator 4中通过样式单调整li元素的颜色，那么只会为li项的先导符号加上颜色。如果要改变元素内文本的颜色，须要将li元素内嵌在span元素之中，并且为span元素设置相应的样式单。在其他能够感知CSS的浏览器中，这种设置方式可正常工作。

示例

```
<ul>
   <li>Larry</li>
   <li>Moe</li>
   <li>Curly</li>
</ul>
```

对象模型引用方式 `[window.]document.getElementById(elementID)`

元素专有属性 type、value

元素专有事件处理属性 无。

type

IE all NN all Moz all Saf all Op 7 HTML 3.2

`type="标注类型"` 可选

type属性为浏览器显示li项的先导符号或序号提供了一定的灵活性。其属性值分为两大类，分别应用于ol和ul元素的li项中。对ul这种无序列表而言，可以指定圆点、圆圈或方块这三种不同类型的先导符号，而在ol这种有序列表中，则可使用大/小写字符、大/小写罗马数字或阿拉伯数字。HTML 4.0已不再赞成使用type属性，推荐使用样式单属性list-style-type。

须要注意的是，在第4版浏览器中，为一个li元素设置了type属性后，除非也为其后的li元素另外设置type属性，否则还会影响随后的所有li元素。而后续的浏览器则仅限于影响当前的li元素。尽管如此，最好还是为ol或ul元素设置type属性，并让它管理所有内嵌li元素的类型。

示例 `<li type="square">Chicken Curry</li>`

值

在ul元素中使用时，可用值为disc、circle和square。而在ol元素中使用时，可用值则为A、a、I、i和1。浏览器会自动完成排序工作，如下表所示。

类型	示例	类型	示例
A	A, B, C, ...	i	i, ii, iii, ...
a	a, b, c, ...	1	1, 2, 3, ...
I	I, II, III, ...		

默认值 1和disc。

对象模型引用方式 `[window.]document.getElementById(elementID).type`

value

IE *all* NN *all* Moz *all* Saf *all* Op *7* HTML *3.2*

`value="数字"` 可选

仅当li元素内嵌在ol元素中时，才能使用value属性。通过此属性，可以为列表项手动设置其起始排序数字。这样就可以很方便地使用一些其他连续文本打断ol元素列表。

即使将此属性设置为数字，它也不会影响到type属性的设置值。例如，当type为“A”时，将value设置为“3”，就意味着li元素将从字符“C”开始排序。

示例 `<li value="3">Insert Tab C into Slot M. Tighten with a wingnut.</li>`

值 正整数。

默认值 1

对象模型引用方式 `[window.]document.getElementById(elementID).value`

<link>

IE *3* NN *4* Moz *all* Saf *all* Op *7* HTML *all*

`<link>` HTML 结束标签：禁用

与a元素（当它包含href属性时会将它通俗地称为“链接”）不同，link元素位于head元素之中，文档使用它与外部文档建立起联系，而这些外部文档既可能是样式单定义文件也可能是字体定义文件。总的来说，浏览器已经完全开发了此元素的预期功能。网页作者利用此元素的大量属性，能够在当前文档与潜在的相关文档之间建立起联系。从理论上来说，某些联系可以显示为文档或浏览器控件的一部分。但与HTML 4规范相比，目前浏览器中对此元素的实现程度还非常薄弱。与此同时，HTML 4规范中定义的某些属性和所有的事件处理程序，以及本章前文列出的一些共享属性通常应用于页面上的可显示元素，因此在此元素中并不十分有用。link元素也不会不给页面带来明确的显示内容。之所以将某些属性与link元素联系在一起，往往是由于疏忽或只是为了在渲染引擎内保持一致性。

将link元素用于导入样式单时，其title属性会充当一种不很直观的角色。将rel属性设置为“stylesheet”后，如果添加了title属性，就可以命令现代浏览器将导入的样式单作为“作者优先”样式单。如果浏览器允许用户从多个作者提供的样式单中作出选择，那么优先的样式单会位于选择列表的首位，例如，通过点击Firefox菜单选择的“查看”→“页面风格”就可看到这个列表。

示例

```
<head>
<title>Section 3</title>
<link rev="Prev" href="sect2.html">
```

```
<link rel="Next" href="sect4.html">
<link rel="stylesheet" type="text/css" href="myStyles.css">
</head>
```

对象模型引用方式 [window.]document.getElementById(*elementID*)

元素专有属性 charset、href、hreflang、media、rel、rev、src、target、type

元素专有事件处理属性 无。

charset

IE *n/a* NN *n/a* Moz *all* Saf *all* OP *7* HTML *4*

`charset="字符集"` 可选

链接另一端的文本内容的编码方式。

示例 `<link charset="csISO5427Cyrillic" href="moscow.html">`

值 是否区分大小写取决于字符集注册表，请参见*http://www.iana.org/assignments/ character-sets*。

默认值 由浏览器决定。

href

IE *n/a* NN *n/a* Moz *all* Saf *all* Op *7* HTML *4*

`href="URI"` 必选

此属性指定了与link元素相关联的文档URI，尽管并不涉及任何的页面导航，W3C术语仍然称之为“目标”。出于同样的目的，在Navigator 4中还可以使用`src`属性。

示例 `<link rel="Prev" href="sect2.html" src="sect2.html">`

值 任何有效的完整的或相对URL地址。

默认值 无。

对象模型引用方式 [window.]document.getElementById(*elementID*).href

hreflang

IE *6* NN *n/a* Moz *all* Saf *all* Op *7* HTML *4*

`hreflang="语言代码"` 可选

此属性描述了链接的目标文档所采用的语言代码。使用此属性时，须要同时设定`href`属性。在可能的情况下，这个属性主要用于通知浏览器为新的字符集做好准备。

示例 `<link hreflang="HI" href="hindi/Chap3.html">`

值 不区分大小写的语言代码。

默认值 浏览器的默认值。

对象模型引用方式 [window.]document.getElementById(*elementID*).hrefLang

media

IE *4* NN *n/a* Moz *all* Saf *all* Op *7* HTML *4*

`media="描述符列表"` 可选

本属性用于设置`href`属性所指定的目标文档内容所需要的输出设备。当浏览器能够将文档内容进行修正，以便适合不同的设备（如手持电脑、文本—语音转换器或模糊电视设备等）时，`media`属性才会大有作用。目前，可以用这个属性指定单独的外部样式单，这样一来，将页面发送至打印机时就可以应用这些样式。HTML 4规范为预期设备定义了一系列的常量值，但这个列表尚未完全确定，以便将来的浏览器能够为其他的媒体或装置修改输出设备。

示例 `<link rel="Glossary" href="gloss.html" media="screen, tv, handheld">`

值 常量。在用括号括起的字符串中，可以将多个值组合在一起，彼此之间使用逗号分隔。HTML 4定

义了下列值：all | aura | braille | handheld | print | projection | screen | tty | tv。目前的浏览器仅支持3个值：all | print | screen。

默认值 screen

对象模型引用方式 [window.]document.getElementById(*elementID*).media

rel

IE 3 NN 4 Moz all Saf all Op 7 HTML 3.2

rel="链接类型" 可选

此属性用于定义当前元素与链接目标之间的关系，虽然HTML 4推荐标准定义了几种链接类型，但最终还是由浏览器决定如何对待这些链接类型。注意，元素中必须首先包含一个href属性才能使用rel属性。

示例 <link rel="Next" href="sect6.html">

值 文档和浏览器可使用的标准链接类型列表，不同的类型请用空格进行分隔，类型名称不区分大小写。HTML 4认可的链接类型如下：alternate、appendix、bookmark、chapter、contents、copyright、glossary、help、index、next、prev、section、start、stylesheet、subsection。

当前的浏览器使用stylesheet来链接一个外部CSS文件。Navigator 4可以识别stylesheet和fontdef。在Windows操作系统中，IE 5及后续版本还允许使用“快捷方式图标”，以便为定制图标文件（*.ico*）指定一个URL。当用户将页面添加到收藏夹时，这个图标就会显示在收藏夹内。

默认值 无。

对象模型引用方式 [window.]document.getElementById(*elementID*).rel

rev

IE 4 NN n/a Moz all Saf all Op n/a HTML 4

rev="链接类型" 可选

此属性值表示相反的链接关系。类似于rel属性，rev属性所能实现的功能依然取决于浏览器，特别是浏览器如何解释并渲染HTML 4规范中定义的不同链接类型。假设有A和B两个文档，其中均包含指向对方的链接。如果B文档中的rev属性值用来描述两个文档之间的关系，则A文档中就使用rel属性来描述。目前主流的浏览器尚未为该属性提供实际的功能。

示例 <link rev="Prev" href="sect4.html">

值 不区分大小写，但其选择范围仅限于HTML 4认可的标准链接类型。HTML 4认可的链接类型请参考上文中的rel属性。

默认值 无。

对象模型引用方式 [window.]document.getElementById(*elementID*).rev

src

IE n/a NN 4 Moz n/a Saf all Op 7 HTML n/a

src="*URL*" 可选

此属性指定了链接的目的URL地址。对于将要被置入浏览器的内容而言，它指出了资源的所在位置。

示例 <link rel="fontdef" src="fonts/garamond.pfr" href="fonts/garamond.pfr">

值 任何有效的完整的或相对URL地址。

默认值 无。。

target

IE 4 NN n/a Moz all Saf all Op 7 HTML 4

target="窗口或框架名称" 可选

HTML 4提供的target属性是用于指出link元素中href属性的显示路径，例如，指向下一个页面的链接。

值

如果已通过目标元素的name属性指定了框架或窗口名称，就可以使用这个标识符。目前，有如下4个作为常量使用的专用目标名称。

_blank

浏览器为目标文档打开一个新窗口。

_parent

目标文档会取代当前框架的框架集文档，如果不存在，则依照“_self”执行。

_self

目标文档替代在链接所在窗口或框架里的现有文档。

_top

目标文档将会取代任何已加载的框架集窗口，从而占据整个浏览器。如果在窗口中不存在框架集，则依照“_self”执行。

Windws系统下的IE浏览器还应用了另外两个属性值：_search（IE 5及后续版本）和_media（IE 6）。这两个属性据说可以使浏览器直接将链接内容分别加载到浏览器的搜索框和媒体栏中。由于浏览器目前仍不支持链接和搜索框、媒体栏之间的联系，因此这两个属性在这方面比较有用，但具体的实现细节尚不清楚。

默认值 _self

对象模型引用方式 [window.]document.getElementById(*elementID*).target

type

IE *4* NN *4* Moz *all* Saf *all* Op *7* HTML *4*

type="*MIME类型*" 可选

此属性为目标文档或资源的类型给出提示。为让浏览器对链接的样式单类型做出准备，这个属性已在实际中得到了广泛的应用。

示例 <link rel="stylesheet" type="text/css" href="styles/mainStyle.html">

值 不区分大小写的MIME类型。可从此处获取MIME类型目录：*http://www.iana.org/assignments/media-types/*。

默认值 无。

对象模型引用方式 [window.]document.getElementById(*elementID*).type

<listing>

IE *all* NN *all* Moz *all* Saf *all* Op *7* HTML *<4*

<listing>...</listing> HTML结束标签：必选

作为一个块级元素，listing元素会使用单间距字体显示它所包含的内容，例如，使用132列宽显示的计算机代码清单。在大多数浏览器中，此时的字体尺寸比默认尺寸小。而且浏览器还须要为元素内容中的回车和其他空白空间留下位置。很早以前HTML就不再推荐使用此元素，HTML 4.0规范也删除了这个元素。因此，建议使用pre元素代替listing元素。

示例

```
<listing>
&lt;script language="JavaScript"&gt;
  document.write("Hello, world.")
&lt;/script&gt;
</listing>
```

对象模型引用方式 [window.]document.getElementById(*elementID*)

元素专有属性 无。

元素专有事件处理属性 无。

<map>

IE *all* NN *all* Moz *all* Saf *all* Op *7* HTML *3.2*

```
<map>...</map>
```

HTML结束标签：必选

map元素是一种容器，其中包含的area元素定义了客户端图像映射的区域划分与对应的热点链接。将img元素转变为客户端图像映射时，主要是使用map元素中的标识符——name属性，usemap就指向该属性。当然，map中还有很多其他与样式相关的元素，将它们应用于map元素后，map中内嵌的其他元素就会继承这些属性。

示例

```
<img src="images/logo.gif" alt="Scroll to the bottom for navigation links."
height="300" width="250" usemap="#navigation">
<map name="navigation">
<area shape="rect" coords="0,0,100,100" href="products.html">
<area shape="rect" coords="0,100,300,100" href="support.html">
</map>
```

对象模型引用方式 [window.]document.getElementById(*elementID*)

元素专有属性 name

元素专有事件处理属性 无。

name

IE *all* NN *all* Moz *all* Saf *all* Op *7* HTML *3.2*

```
name="标识符"
```

必选

img元素的usemap属性将指向此属性定义的标识符。由于usemap属性实际上属于一种URL类型，因此其属性值类似于anchor元素链接，即属性值名称前将放置一个“#”号。目前也仅在usemap属性中会使用这种定义用法。尽管XHTML偏向于使用id属性来替代name属性，但浏览器依然会将name属性作为图像和区域映射之间的联系纽带。严格的HTML 4和XHTML DTD目前也继续支持name属性。

示例 <map name="navigation"> ...</map>

值 区分大小写的唯一标识符。

默认值 无。

对象模型引用方式 [window.]document.getElementById(*elementID*).name

<marquee>

IE *3* NN *n/a* Moz *1.0.1* Saf *all* Op *8* HTML *n/a*

```
<marquee>...</marquee>
```

HTML结束标签：可选

IE浏览器首先使用了marquee元素。通过它可以在页面上的一个滚动区域内显示一些HTML内容。滚动显示的内容包含在元素的起始和结束标签之内。现在主流浏览器均支持该元素，但IE浏览器仍然是支持得最好的一个。

示例

```
<marquee behavior="slide" direction="left" width="250" bgcolor="white">
Check out our monthly specials.
</marquee>
```

对象模型引用方式 [window.]document.getElementById(*elementID*)

元素专有属性 behavior、bgcolor、datafld、dataformatas、datasrc、direction、height、hspace、loop、scrollamount、scrolldelay、truespeed、vspace、width

元素专有事件处理属性

处理程序	IE	其他	HTML	处理程序	IE	其他	HTML
onafterupdate	4	n/a	n/a	onfinish	4	n/a	n/a
onbounce	4	n/a	n/a	onstart	4	n/a	n/a

behavior

IE *3* NN *n/a* Moz *1.0.1* Saf *all* Op *8* HTML *n/a*

```
behavior="运动方式"
```

可选

通过此属性可以设置marquee元素矩形框内HTML内容的运动方式。目前有3种运动方式可供选择。

示例

```
<marquee behavior="slide" direction="left" width="250" bgcolor="white">...</marquee>
```

值

可选用以下几种方式之一，它们均不区分大小写。

`alternate`

相关内容在两侧边缘之间来回移动。Mozilla直到1.7版，它仍然仅支持这一个属性值。

`scroll`

相关内容会在框内循环滚动，看起来它会从一端消失，然后又从另一端进入视野，`direction`属性可指定其滚动方向。

`slide`

相关内容首先会进入视野，它走完全程后会静止在一端，仍然由`direction`控制滚动的方向。

默认值 `scroll`

对象模型引用方式 `[window.]document.getElementById(elementID).behavior`

bgcolor

IE *3* NN *n/a* Moz *1.0.1* Saf *all* Op *8* HTML *n/a*

```
bgcolor="颜色三元组或颜色名"
```

可选

此属性为marquee元素的整个矩形空间设置填充色，此颜色位于文本和其他内容之下。

示例

```
<marquee behavior="slide" direction="left" width="250" bgcolor="white">...</marquee>
```

值 一个十六进制的三元组或明语颜色名称。不设置属性值就默认为黑色（`#000000`，**black**）。目前可用的明语颜色名称请参见附录A。

默认值 随着浏览器、浏览器版本和操作系统的不同而改变。

对象模型引用方式 `[window.]document.getElementById(elementID).bgColor`

datafld

IE *4* NN *n/a* Moz *n/a* Saf *n/a* Op *n/a* HTML *n/a*

```
datafld="列名称"
```

可选

此属性用于IE中的数据绑定，以便将一个远程数据源的列名与marquee元素中的滚动的HTML内容联系在一起。数据源的对应列必须是纯文本或HTML，请参见`dataformatas`。此时marquee元素中还必须设定`datasrc`属性。注意，只能在IE 5/Mac操作系统下实现文本文件数据源的绑定。

示例

```
<marquee behavior="slide" direction="left" width="200"
datasrc="DBSRC3" datafld="news" dataformatas="HTML"></marquee>
```

值 区分大小写的标识符。

默认值 无。

对象模型引用方式 [window.]document.getElementByld(*elementID*).dataFld

dataformatas

IE *4* NN *n/a* Moz *n/a* Saf *n/a* Op *n/a* HTML *n/a*

193

```
dataformatas="数据类型"
```

可选

本属性用于IE浏览器中的数据绑定。对于来自于数据源的原始资料，浏览器根据此属性值决定将它们视为纯文本亦或是HTML标签。本属性的设置完全依赖于数据源的构造方式。注意，只能在IE 5/Mac操作系统下实现文本文件数据源的绑定。

示例

```
<marquee behavior="slide" direction="left" width=200
datasrc="DBSRC3" datafld="news" dataformatas="HTML"></marquee>
```

值 常量：text | html。

默认值 text

对象模型引用方式 [window.]document.getElementById(*elementID*).dataFormatAs

datasrc

IE *4* NN *n/a* Moz *n/a* Saf *n/a* Op *n/a* HTML *n/a*

```
datasrc="数据源名称"
```

可选

本属性用于IE浏览器中的数据绑定。对于来自于数据源的原始资料，浏览器根据此属性值决定将它们视为纯文本亦或是HTML标签。注意，只能在IE 5/Mac操作系统下实现文本文件数据源的绑定。

示例

```
<marquee behavior="slide" direction="left" width="200"
datasrc="DBSRC3" datafld="news" dataformatas="HTML"></marquee>
```

值 区分大小写的标识符。

默认值 无。

对象模型引用方式 [window.]document.getElementById(*elementID*).dataSrc

direction

IE *3* NN *n/a* Moz *1.0.1* Saf *all* Op *8* HTML *n/a*

```
direction="滚动方向"
```

可选

marquee元素中的内容共有4种滚动方向（Safari仅支持向左和向右两种方向）。对书面语言而言，从左到右的书写方式最具可读性，同理，从左到右或从上向下进行内容滚动也最便于阅读。

示例 `<marquee behavior="slide" direction="left" width="200">...</marquee>`

值 4种可用的滚动方向之一：down | left | right | up。

默认值 left

对象模型引用方式 [window.]document.getElementById(*elementID*).direction

194

height, width

IE *3* NN *n/a* Moz *1.0.1* Saf *all* Op *8* HTML *n/a*

```
height="长度值" width="长度值"
```

可选

marquee元素在页面上会显示为一个矩形空间。通过设置height和width属性值，可以重新设置这个矩形空间的默认尺寸。marquee元素包含内容的最大字体的字符尺寸，会决定height属性的默认值。而默认的width值则与其相邻的最外层容器宽度一致，例如文档的body元素宽度。width属性定义了每次水平滚动时使用的空间大小。如果将marquee元素内嵌在td元素中，并让浏览器自动计算表格单元格的宽度，那么就必须指定其width值，否则浏览器可能会将width设置为1，使得滚动内容不可见。

如果希望在这个矩形空间四周再附加一些空白区域，可以使用hspace和vspace属性。

值得注意的是，在Mozilla系列浏览器中，只能使用百分比或通过CSS的width属性指定marquee的宽度。而在Safari中，虽然可以拉伸整个元素，但其内的文本并不能在竖直方向上滚动。

示例 `<marquee behavior="slide" direction="left" height="20" width="200">...</marquee>`

值 任意一个单位为像素的长度值，或者可用空间的百分比。

默认值 width为100%，height为12像素。

对象模型引用方式

```
[window.]document.getElementById(elementID).height
[window.]document.getElementById(elementID).width
```

hspace, vspace

IE *3* NN *n/a* Moz *n/a* Saf *all* Op *n/a* HTML *n/a*

`hspace="像素值" vspace="像素值"` 可选

IE浏览器提供这两个属性来设置marquee元素四周的填充。hspace属性控制左右两侧的填充（水平填充），而vspace属性则控制顶部和底部的填充（竖直填充）。使用填充后，会在marquee四周形成一个空白的缓冲区。另外，还可以调整样式单的页边距来达到这一效果，这种做法特别适合于控制单独一条边的空白空间。

示例

```
<marquee behavior="slide" direction="left" height="20" width="200"
hspace="10" vspace="15">...</marquee>
```

值 正整数。

默认值 0

对象模型引用方式

```
[window.]document.getElementById(elementID).hspace
[window.]document.getElementById(elementID).vspace
```

loop

IE *3* NN *n/a* Moz *n/a* Saf *all* Op *8* HTML *n/a*

`loop="整数值"` 可选

此属性可以设置marquee元素显示其内容时的滚动次数。完成最后一次滚动后，相关内容会停留在一个固定的位置。由于连续的动画显示可能会干扰页面访问者，因此，最好让相关内容滚动几次后就不再滚动。

示例

```
<marquee behavior="slide" direction="left" height="20" width="200" loop="3">...</marquee>
```

值 设置为正整数值时，完成该数值指定的滚动次数后就会停止下来。如果设置为“-1”或“infinite”，则不会停止滚动。

默认值 -1

对象模型引用方式 `[window.]document.getElementById(elementID).loop`

scrollamount

IE *3* NN *n/a* Moz *1.0.1* Saf *all* Op *8* HTML *n/a*

`scrollamount="像素值"` 可选

通过浏览器不断地将显示内容清除并重绘，marquee中的内容就像动画一样在页面上改变位置，移动的方向则由其direction属性值来决定。增加每次位置改变之间的偏移量，可以加快滚动的速度；相反地，减小偏移量则可以使得滚动变慢。请同时参考scrolldelay属性。

示例

```
<marquee behavior="slide" direction="left" height="20" width="200" scrollamount="2">
...</marquee>
```

值 正整数。

默认值 6

对象模型引用方式 [window.]document.getElementById(*elementID*).scrollAmount

scrolldelay

IE *3* NN *n/a* Moz *1.0.1* Saf *all* Op *8* HTML *n/a*

196

```
scrolldelay="毫秒数"
```

可选

由于每次重绘都会产生位置改变，因此改变重绘的频率也可以影响滚动的速度。加大scrolldelay值就会降低滚动速度，而减小其值则会使滚动加速。但要注意，在一些速度较慢的计算机上，无论将scrolldelay值设置为多少，都无法观察到滚动速度的增加，请参考truespeed属性。

示例

```
<marquee behavior="slide" direction="left" height="20" width="200" scrolldelay="100">
...</marquee>
```

值 任意正整数值，用以描述每次内容重绘之间的毫秒间隔。

默认值 85（IE/Windows）；95（Macintosh）。

对象模型引用方式 [window.]document.getElementById(*elementID*).scrollDelay

truespeed

IE *4* NN *n/a* Moz *n/a* Saf *all* Op *n/a* HTML *n/a*

```
truespeed
```

可选

marquee元素包含一个内置的速度阀值，以免由于滚动速度过快而导致访问者无法看清相关内容。如果确实希望相关内容能够快速滚动，那么就可以使用truespeed属性，以便在scrolldelay小于60毫秒时通知浏览器仍根据其属性值设置速度。

示例

```
<marquee behavior="slide" direction="left" height="20" width="200"
scrolldelay="45" truespeed>...</marquee>
```

值 如果此属性已出现，就意味着其属性值为true。

默认值 false

对象模型引用方式 [window.]document.getElementById(*elementID*).trueSpeed

vspace

参见hspace属性。

width

参见height属性。

<menu>

IE *all* NN *all* Moz *all* Saf *all* Op *7* HTML *all*

197

```
<menu>...</menu>
```

HTML结束标签：必选

使用menu元素的最初设想，是允许浏览器生成条目的单栏式列表。但实际情况却是，每个浏览器都将menu元素视作ul元素，即显示条目的无序单列列表（往往以圆点作为开头）。HTML 4并不赞成使用menu元素。由于menu元素将永远从浏览器的视野中消失，因此如果要保证代码的兼容性，应该在任何情况下都使用ul元素来代替它。此处所阐述的任何内容还适用于另一个不再赞成使用的元素——dir。

示例

```
Common DB Connector Types:
<menu>
    <li>DB-9</li>
    <li>DB-12</li>
    <li>DB-25</li>
</menu>
```

对象模型引用方式 [window.]document.getElementById(*elementID*)

元素专有属性 compact

元素专有事件处理属性 无。

compact

IE *6* NN *n/a* Moz *all* Saf *n/a* Op *n/a* HTML *3.2*

compact

可选

该属性值是一个布尔变量，设计的初衷是让浏览器能够以另一种更为紧凑的样式显示列表，即与正常情况相比，使条目之间的行距更小。

示例 <menu compact>...</menu>

值 如果此属性已出现，就意味着其属性值为true。

默认值 false

对象模型引用方式 [window.]document.getElementById(*elementID*).compact

<meta>

IE *all* NN *all* Moz *all* Saf *all* Op *7* HTML *all*

<meta>

HTML结束标签：禁用

meta元素可以承载一些关于文档的隐藏信息。某些浏览器会响应此元素以获取头信息，虽然在向服务器请求文档时，服务器并不会发送这些头信息，但它们对文档而言可能非常重要。同时，还可以使用此元素内嵌一些文档信息，搜索引擎即可通过它们为互联网上的文档建立索引并进行分类。

文档中可以包含多个meta元素，而所有的meta元素都会内嵌在head元素之中。通过它的元素属性，可以决定每个meta元素的特定作用。meta元素实际上已经简化为一个名称/值对，这种形式无论是对服务器还是客户端都很有价值。例如，某些属性设置可以让页面在一段时间之后重载或重定向到另一个页面，而目前大多数浏览器均可识别这种属性设置。在指定了这样的meta元素之后，浏览器就会不断地加载最新页面，因此，如果某个页面须要每分钟定时更新其页面内容，那么这些属性就非常有用。

除此之外，在HTML 4中，meta元素内还可以设置其他元素或属性所包含的元数据。因此，请根据实际使用中的服务器和浏览器环境选择最佳的设置方式。请参考address、del、ins、link及title元素，以及head元素的profile属性。

有很多神话环绕在meta元素周围。但实际上有些属性并不能应用于某些浏览器之中（例如控制浏览器缓存的属性），同样，也不是所有的搜索引擎都能够响应meta标签属性。与此同时，目前的标准也并不赞成使用某些常用功能，例如refresh。虽然现在还没有可用属性的标准规范，但下面这个示例中列出了HTML 4和XHTML中W3C校验程序认可的属性值。

示例

```
<head profile="http://www.example.com/profiles/common">
    <meta name="Author" content="Jane Smith">
    <meta name="keywords" content="benefits,insurance,plan">
    <meta http-equiv="refresh"
```

```
    content="1;URL=http://www.example.com/truindex.html">
    <meta http-equiv="Content-Type" content="text/html;
    charset=ISO-8859-5">
</head>
```

元素专有属性 content、http-equiv、name、scheme

元素专有事件处理属性 无。

content

IE *all* NN *all* Moz *all* Saf *all* Op *7* HTML *all*

```
content="字符串"
```

必选

此属性等价于名称/值对中的属性值。它通常与name或http-equiv属性一起使用，而这两个属性则扮演了名称/值对中名称的角色。根据name或http-equiv属性值的不同，须要为content指定不同的属性值。有时，content还会包含多个值。在这种情况下，可以用逗号、分号或其他任意浏览器认可的符号分隔这些不同的值。这些属性值将组成有一定含义的名称/值对，例如下例中meta元素的“refresh”和对应的内容属性值。在content的属性值中，第一个值为数字，它指定了在加载另一个文档前须要延迟的秒数；而第二个属性值则为URL，它指定了延迟后所加载的文档地址。

示例

```
<meta http-equiv="refresh" content="2;URL=http://www.example.com/basicindex.html">
```

值 任意字符串。该字符串应该位于一对引号之中，双引号是推荐的标准格式，也可使用单引号。

默认值 无。

对象模型引用方式 [window.]document.getElementById(*elementID*).content

http-equiv

IE *all* NN *all* Moz *all* Saf *all* Op *7* HTML *all*

```
http-equiv="标识符"
```

必选

当服务器通过HTTP协议将文档发送至客户端时，还会同时发送一些HTTP头数据，它们将作为有关文档内容的客户端命令。在文档中指定了http-equiv属性后，就可以向HTTP头添加一些meta元素。浏览器会将http-equiv和content属性值转换为HTTP响应头的格式——“名称:值”，然后将它们视为服务器直接发送的数据进行处理。

网页标准定义了很多HTTP头（请参考《Webmaster in a Nutshell》一书，作者Stephen Spainhour和Valerie Quercia，O'Reilly出版），下文的示例中已向大家展示了一些常用的值。但并不是所有的浏览器均会响应这些头类型，而某些浏览器还会响应自定义的类型，例如，IE 6定义了“MSTHEMECOMPATIBLE”头类型。另外，尽管必须在meta元素中使用http-equiv或name属性，但这两者不能同时在同一个meta元素中使用。

示例

```
<meta http-equiv="refresh"
content="1,http://www.example.com/truindex.html">
<meta http-equiv="Content-Type" content="text/html; charset=ISO-8859-5">
<meta http-equiv="expires" content="Sun, 15 Jan 1998 17:38:00 GMT">
```

值 任何字符串标识符。

默认值 无。

对象模型引用方式 [window.]document.getElementById(*elementID*).httpEquiv

name

IE *all* NN *all* Moz *all* Saf *all* Op *7* HTML *all*


```
name="标识符"
```

可选

此属性是用于名称/值对的标识符，它是meta元素的一部分。通过name的属性值很明确地表明了使用该meta

元素的目的，如“author”或“keywords”。尽管必须在meta元素中使用http-equiv或name属性，但这两者不能同时在同一个meta元素中使用。

示例

```
<meta name="Author" content="Jane Smith">
<meta name="keywords" content="benefits,insurance,plan">
```

值 任何字符串标识符。

默认值 无。

对象模型引用方式 [window.]document.getElementById(*elementID*).name

scheme

IE *6* NN *n/a* Moz *all* Saf *n/a* Op *n/a* HTML *4*

```
scheme="标识符"
```

可选

此元素为文档内的元数据提供了进一步的组织层次。例如，某大学拥有几个图书馆，那么就可为每个图书馆均指定一个对应的文档。假设浏览器须要解释其元数据信息，那么一种处理方式就是在meta元素标签中为每个图书馆创建不同的name属性值，例如，name="law"、name="main"、name="engineering"，等等。但是，可能还须要将name属性值与不同的大学联系起来。这时，就可以通过scheme属性将元数据与特定的大学联系起来，如scheme="Harvard"。此时，其他大学图书馆系统也可以使用相同组织形式的name属性，仅通过scheme就可将meta元素与特定的大学联系起来。另外再强调一点，此时已假设浏览器有能力对相关的元知识进行特殊处理。

示例 <meta scheme="Chicago" name="classicalFM" content="98.7">

值 任何字符串标识符。

默认值 无。

对象模型引用方式 [window.]document.getElementById(*elementID*).scheme

<multicol>

IE *n/a* NN *3* Moz *n/a* Saf *n/a* Op *n/a* HTML *n/a*

```
<multicol>...</multicol>
```

HTML 结束标签：必选

此元素是Navigator 3和4中的特定元素，它可以将一些内容显示在任意数量的连续等宽列中。该元素排列内容的方式可能会让人想起桌面出版程序，它会将内容自动地按顺序排列在页面上早已定义好的列空间中。尽管在HTML和IE中并没有与之对应的等价元素，但在CSS3中有一个议案与多行布局属性有关，请参见*http://www.w3.org/TR/css3-multicol/*。

示例 <multicol cols="2" gutter="20" width="500">*LongFlowingHTMLContent*</multicol>

元素专有属性 cols、gutter、width

元素专有事件处理属性 无。

cols

IE *n/a* NN *3* Moz *n/a* Saf *n/a* Op *n/a* HTML *n/a*

```
cols="列数量"
```

必选

此属性定义了浏览器中排列与显示元素内容的总列数。对于已指定宽度的内容而言，如果让每列的长度相同可使浏览器的显示达到最佳效果。CSS中建议使用的对应属性是column-number。

示例 <multicol cols="2" gutter="20" width="500">*LongFlowingHTMLContent*</multicol>

值 正整数。

默认值 1

gutter

IE *n/a* NN *3* Moz *n/a* Saf *n/a* Op *n/a* HTML *n/a*

```
gutter="像素值"
```

可选

此属性用于指定两列之间空白空间的像素值。设置此属性后，浏览器会从可用宽度总数中减去全部列间空白宽度，以得到每一列的宽度。CSS中建议使用的对应属性是column-gap。

示例 `<multicol cols="2" gutter="20" width="500">LongFlowingHTMLContent</multicol>`

值 正整数。

默认值 10

width

IE *n/a* NN *3* Moz *n/a* Saf *n/a* Op *n/a* HTML *n/a*

```
width="元素宽度"
```

可选

此属性定义了所有列的总宽度，其中包括列间空白。可以使用像素值或临近的外层容器宽度的百分比来设置其属性值。CSS中建议使用的对应属性是width。

示例 `<multicol cols="2" gutter="20" width="500">LongFlowingHTMLContent</multicol>`

值 任意一个单位为像素的长度值，或者可用空间的百分比。

默认值 100%

<nextid>

IE *all* NN *n/a* Moz *n/a* Saf *n/a* Op *n/a* HTML *<2*

```
<nextid>
```

HTML结束标签：禁用

曾经有一段时间期望使用nextid元素来辅助文档编辑程序，并将它放置在文档的head元素中。但从HTML 2.0开始就不再赞成使用这个元素，它也闲置已久。

<nobr>

IE *all* NN *all* Moz *all* Saf *all* Op *7* HTML *n/a*

```
<nobr>...</nobr>
```

HTML结束标签：必选

nobr元素会命令浏览器在显示内容时不要进行文本换行。使用这个元素后，即使在元素内容的源代码中包含回车符，浏览器也在一行中顺序显示文本内容。尽管这也许会给页面布局带来一些便利，但也意味用户必须滚动页面以便看到所有的文本，而大多数用户其实并不乐意这样做。虽然在商业浏览器中依然可以使用nobr元素，但正式的HTML推荐标准中已不再提及此元素。

示例

```
<nobr>
Now is the time for all good men to come to the aid of their country, even if
the text forces them to scroll horizontally.
</nobr>
```

对象模型引用方式 `[window.]document.getElementById(elementID)`

元素专有属性 无。

元素专有事件处理属性 无。

<noembed>

IE *n/a* NN *2* Moz *all* Saf *all* Op *7* HTML *n/a*

```
<noembed>...</noembed>
```

HTML结束标签：必选

当浏览器无法使用插件时，通过noembed元素可以给出相关的提示信息。在允许使用插件的浏览器中，不会

显示包含在noembed元素的起始和结束标签之内的所有内容，但在其他浏览器中就会显示这些内容，当然，并不会显示标签。还没有可供此元素使用的属性。

示例

```
<embed name="jukebox" src="jazz.aif" height="100" width="200"></embed>
<noembed>
To play the music associated with this page, you need a modern graphical browser.
</noembed>
```

元素专有属性 无。

元素专有事件处理属性 无。

<noframes>

IE *3* NN *2* Moz *all* Saf *all* Op *7* HTML *4*

`<noframes>...</noframes>` HTML 结束标签：必选

当浏览器无法显示框架时，会显示noframes元素中包含的HTML内容。而那些可以显示框架的浏览器则会忽略noframes元素及它包含的所有内容。此元素包含的内容应该提示用户目前使用了框架，或者为用户提供一个不包含框架的页面链接。经常将noframes元素放置在frameset元素内。但HTML 4规范认为，在可绘制的文档内可以将它内嵌在任意一个元素中。在使用iframe元素时，这个元素也很有用处，它可以让那些不支持iframe的浏览器提醒用户遗漏了哪些信息。

层叠样式单能支持noframes元素的所有标准属性。但这看起来很怪异，难道一个浏览器会支持CSS而不支持框架吗？不过Navigator 4就是一个很好的例子，它恰好就不支持iframe。

示例

```
<frameset cols="150,*">
    <frame name="navbar" src="nav.html">
    <frame name="main" src="page1.html">
    <noframes>Your browser does not support frames.
    Click <a href="noFramesIndex.html">here</a> for a frameless version.
    </noframes>
</frameset>
```

元素专有属性 无。

元素专有事件处理属性 无。

<nolayer>

IE *n/a* NN *4* Moz *n/a* Saf *n/a* Op *n/a* HTML *n/a*

`<nolayer>...</nolayer>` HTML 结束标签：必选

如果某些浏览器不能识别layer元素，Navigator 4就为这种情况所需的提示信息提供了一个标签载体。Navigator 4不会显示包含在nolayer元素的起始和结束标签之内的所有内容，但在其他浏览器中就会显示这些内容，当然，并不会显示标签。尽管可以将nolayer元素放置在任意位置，但值得注意的是，它无法像layer元素那样进行定位。无法使用层的浏览器均会显示nolayer元素中的相关内容。

目前还没有可供此元素使用的属性。如果要为此元素设置样式单规则，有些浏览器会忽略这些设置，例如IE。然而，可以将nolayer元素内置在一个div或span元素中，然后通过容器元素为提示信息设置样式规则。

示例

```
<layer bgcolor="yellow" src="instrux.html" width=200 height=300></layer>
<nolayer>
You are not seeing some content that requires Netscape Navigator 4 to view.
</nolayer>
```

元素专有属性 无。

元素专有事件处理属性 无。

<noscript>

IE 4 NN 3 Moz all Saf all Op 7 HTML 4

<noscript>...</noscript>

HTML结束标签：必选

如果浏览器无法运行当前文档中的脚本，那么此元素内的相关内容就会显示在页面上。如果用户禁用了浏览器的脚本功能，那么源代码中出现脚本的任何位置均会显示noscript元素的相关内容。对于老式的浏览器以及那些不支持脚本的浏览器，它们会忽略noscript元素，并将该元素显示在临近的最外层容器内容之中。当浏览器不支持或不允许脚本的语言类型时，HTML 4.0推荐标准要求浏览器也显示noscript元素的相关内容。同时，如果某个兼容HTML 4的浏览器缺少脚本功能，那么它也应该显示noscript元素内容。但在实际应用中，只有关闭了浏览器的脚本功能后，支持脚本的浏览器才会显示这些内容。

noscript元素的所有标准属性均可支持HTML 4的样式单设置和相关事件操作。

示例

```
<noscript>
This document contains programming that requires a scriptable browser, such as Microsoft
Internet Explorer or Netscape Navigator. You may not have full access to this page's powers
at this time.
</noscript>
```

元素专有属性 无。

元素专有事件处理属性 无。

<object>

IE 3 NN 4 Moz all Saf all Op 7 HTML 4

<object>...</object>

HTML结束标签：必选

如果浏览器不能直接支持某些数据类型，那么通过此元素可以为浏览器提供相关信息，以便加载并显示这些数据类型。如果浏览器必须加载某些外部程序（如Java小程序、插件、ActiveX对象或其他辅助程序），那么object元素内的相关信息就会显示出来，而它的属性和其他可选的相关param元素则内嵌在元素之中。尽管目前的浏览器还能识别类似的其他元素，如applet、embed等，但HTML规范指出，未来的趋势是完全使用object元素来整合这些相关元素。实际上IE一直以来也都乐于使用object元素。

HTML 4规范还允许嵌套使用object元素，这样一来，当浏览器中无可用插件或其他辅助手段时，浏览器就可以加载其他的替代内容。实质上，浏览器应该能够遍历嵌套的object元素，直到找到一个可以处理的为止。如下例所示，外层的object元素将加载一段MPEG2视频，如果没有可用的播放器，那么浏览器应该查找其相邻的嵌套object元素，它仅显示了视频的一个静态JPEG画面，如果该浏览器不是一个图形浏览器，那么它应该直接显示object元素嵌套中最内层的HTML内容，这个HTML内容可能并不是一个object元素，在本例中就是一段文字。

```
<div>
<object data="proddemo.mpeg" type="application/mpeg">
   <object data="prodStill.jpg" type="image/jpeg">
      The all-new Widget 3000!
   </object>
</object>
</div>
```

请参见embed元素，以便了解如何混合使用object和embed元素以适应不同的浏览器种类。

请阅读相关对象与插件的参考文档，以便掌握不同内容类型所需的属性与属性值。另外，这些参考文档还会指明哪些浏览器品牌和操作系统能够运行相关的对象或插件。

示例

206

```
<object id="earth" classid="clsid:83A38BF0-B33A-A4FF-C619A82E891D">
<param name="srcStart" value="images/earth0.gif">
<param name="frameCount" value="12">
<param name="loop" value="-1"
<param name="fps" value="10">
</object>
```

元素专有属性 align、alt、archive、border、classid、code、codebase、codetype、data、declare、height、hspace、name、standby、type、usemap、vspace、width

元素专有事件处理属性 无。

align

IE 3 NN 4 Moz all Saf all Op 7 HTML 4

`align="对齐常量"` 可选

本属性决定了在环绕四周的上下文中，object元素矩形空间的对齐方式。请参见本章前文中“对齐常量”的有关内容，其中已讨论了元素与元素盒外部相关内容的对齐问题。

示例 `<object ... align="baseline"></object>`

值 常量。参见“对齐常量”。

默认值 bottom

对象模型引用方式 `[window.]document.getElementById(elementID).align`

archive

IE 6 NN n/a Moz all Saf all Op n/a HTML 4

`archive="URI列表"` 可选

此属性提供了一个用空格分隔的文件URI列表，object元素可以加载并运行这些文件。在archive属性中明确指定了文件列表后，浏览器就不需要等待object元素中的内容调用这些文件。相反地，浏览器会同时下载这些文件与元素内的主要内容。archive属性中还可以包含一些已分配给classid或data属性的URI，但在这两个属性中，至少要保证有一个指向主内容的URI。目前主流的浏览器尚未为该属性提供实际的功能。

示例 `<object ... archive="/images/anim3.gif/images/anim4.gif"></object>`

207

值 完整或相对URL路径。

默认值 无。

对象模型引用方式 `[window.]document.getElementById(elementID).archive`

border

IE 6 NN n/a Moz all Saf n/a Op n/a HTML 4

`border="像素值"` 可选

此属性定义了object元素四周边框的厚度。HTML 4不再赞成使用此属性，而推荐使用样式单。在IE 5/Mac中，如果使用object元素为客户端图像映射加载图像，那么可以将border属性设置为0，以去掉典型的链接边框。

示例 `<object ... border="4"></object>`

值 任意整数值。

默认值 无。

对象模型引用方式 `[window.]document.getElementById(elementID).border`

classid

IE 3 NN 4 Moz all Saf all Op 7 HTML 4

`classid="URL"` 可选

此属性指定了对象实现的URL。这样就可以指引浏览器加载一个程序、applet或插件类文件。在IE中，这个URL可以指向一个CLSID目录，而在该目录下则保存着所有已注册ActiveX控件的ID，例如DirectAnimation控件。这些classid值需要从ActiveX控件的提供者那里获取，也可以通过Regedit程序在注册表中找到对应的值。在Navigator 4和Mozilla中，当data属性指定的数据类型所需的插件尚未安装时，Java归档文件（JAR）安装管理器会尝试着从classid指定的URL安装插件。该属性虽然可用于加载Java小程序（IE 4引入了一个code属性来处理它），但在目前的浏览器中并不能用这种方式支持Java小程序。

示例 `<object id="earth" classid="clsid:83A38BF0-B33A-A4FF-C619A82E891D"></object>`

值 完整或相对URL路径。

默认值 无。

对象模型引用方式 `[window.]document.elementID.classid`

code

IE 4 NN n/a Moz n/a Saf n/a Op n/a HTML n/a

`code="文件名.class"` 可选

在IE中，使用code属性可以使得object元素扮演与applet元素相同的角色，它们使用相同类型的属性。此时code的属性值就是Java小程序的类文件的名称。如果这个类文件保存在一个目录中而不是文档内，那么就需要通过codebase属性指定该目录的路径，这与它在applet元素中的用法一样。相关的参数则通过内嵌在object元素中的param元素传递给applet小程序。不过IE保留classid属性的目的似乎仅仅就是为了引用ActiveX控件。

示例 `<object code="fileReader.class" codebase="classes"></object>`

值 applet小程序的类文件名。

默认值 无。

对象模型引用方式 `[window.]document.elementID.code`

codebase

IE 3 NN 4 Moz all Saf all Op 7 HTML 4

`codebase="路径"` 可选

针对由code或classid属性指定的类文件，此属性指明了保存该类文件的文件夹路径。要注意一点，codebase属性并不包含类文件的名称，它仅仅说明其所在路径。可以使用文件夹的完整URL路径作为属性值，但请勿使用超出其当前文档域的路径。

示例 `<object code="fileReader.class" codebase="classes"></object>`

值 区分大小写的路径名称，往往与保存当前HTML文档的文件夹相关。

默认值 无。

对象模型引用方式 `[window.]document.getElementById(elementID).codeBase`

codetype

IE 3 NN n/a Moz all Saf all Op n/a HTML 4

`codetype="MIME类型"` 可选

此属性为classid属性所指出的对象类型给出了相应的提示信息。此信息可以帮助浏览器为需要多媒体播放器或插件的资源提前做好准备。尽管这一信息往往与data属性指定的URL所链接的内容有关，但在缺少codetype属性的情况下，那么浏览器只会继续查看type属性的设定。如果这两个属性均未使用，那么浏览器才会从下载的资源中获取内容类型信息。

示例

```
<object classid="clsid:83A38BF0-B33A-A4FF-C619A82E891D"
codetype="application/x-crossword"></object>
```

209

值 MIME类型不区分大小写。MIME类型目录可从此出获得：*http://www.iana.org/assignments/media-types/*。

默认值 无。

对象模型引用方式 `[window.]document.getElementById(elementID).codeType`

data

IE *3* NN *4* Moz *all* Saf *all* Op *7* HTML *4*

`data="URL"`

可选

此属性指明了一个文件的URL地址，object元素将加载该文件的数据信息。对那些使用兼容对象或插件就可打开、显示或播放的数据类型而言，使用data与type属性就足以加载插件并加载数据内容。如果需要非常特殊的插件或ActiveX控件，那么就需要同时使用classid属性以指定具体的对象实现。此时，可以使用codetype或type属性来指定数据类型。如果指定了codebase属性，那么将根据它的属性值计算文件的相对URL路径，否则文件URL路径只与所在文档的URL有关。

示例 `<object data="proddemo.mpeg" type="application/mpeg"></object>`

值 完整或相对URL路径。

默认值 无。

对象模型引用方式 `[window.]document.getElementById(elementID).data`

declare

IE *6* NN *n/a* Moz *all* Saf *all* Op *n/a* HTML *4*

`declare`

可选

declare属性会命令浏览器仅将当前的object元素视为一个声明，而不再对其进行实例化工作。浏览器可利用这个机会提前缓冲那些不需要加载或运行对象的数据。如果另一个拥有相同classid或data属性值的object元素并未声明declare属性，那么浏览器会让该对象正常运行起来。目前主流的浏览器尚未为该属性提供实际的功能。

示例 `<object classid="clsid:83A38BF0-B33A-A4FF-C619A82E891D" declare></object>`

值 如果此属性已出现，就意味着其属性值为true。

默认值 false

对象模型引用方式 `[window.]document.getElementById(elementID).declare`

210

height, width

IE *3* NN *4* Moz *all* Saf *all* Op *7* HTML *4*

`height="长度值" width="长度值"`

可选

通过height和width属性，可以控制一个内嵌对象（或一个插件的控制面板）在页面内所占据的空间大小。在一些浏览器版本中也可以不对这两个属性进行设定，此时就由插件自身界面的高和宽决定它在文档中可见矩形框的大小。由于插件控制面板或图片大小会随着浏览器的不同而发生改变，甚至不同的插件在同一款浏览器下尺寸也会有所不同，因此只要条件允许，最好为每个插件控制面板或图片均指定确切的尺寸大小。在某些情况下，如果没有在页面上为对象提供足够的高度空间，该对象可能无法显示。另外，如果播放器加载失败，有时也不会保留其所占空间。如果为实际对象或其控制面板指定的空间过大，浏览器在页面上会保留剩余的空白空间，而这也会干扰页面的预期设计构想。

当通过脚本可以控制某个对象，且并不希望显示其控制面板时，可以将其尺寸大小设置为0或1。同时，将标签放置在文档的末尾。

示例 `<object data="blues.aif" height="150" width="250"></object>`

值 由引号括起来的正整数或百分比。

默认值 无。

对象模型引用方式

```
[window.]document.getElementById(elementID).height
[window.]document.getElementById(elementID).width
```

hspace, vspace

IE 3 NN n/a Moz all Saf n/a Op 7 HTML 4

```
hspace="像素值" vspace="像素值"
```

可选

环绕在object元素的矩形空间四周的空白填充空间大小由这两个属性来表示。hspace确定了矩形框左右两侧的边距大小，而vspace则决定了上下两端的边距大小。

示例

```
<object data="blues.aif" height="150" width="250" vspace="10" hspace="10"> </object>
```

值 表示object元素矩形框对应边边距宽度的整数像素值。

默认值 0

对象模型引用方式

```
[window.]document.getElementById(elementID).hspace
[window.]document.getElementById(elementID).vspace
```

name

IE 6 NN n/a Moz all Saf all Op 7 HTML 4

```
name="元素标识符"
```

可选

HTML 4参考规范为object元素提供了name属性，这样它就可以作为表单的一部分被提交至服务器端。此时name属性的作用与input元素的name属性完全相同，可以作为被提交数据的标签来使用。如果对象将通过HTML表单进行提交，那么编写object元素内加载的代码时必须使其拥有一个返回值。当前的主流浏览器支持此属性以便进行兼容性声明，但并不会对其值做出任何响应。另外，请使用id属性来指定标识符，以便通过脚本引用对象。

示例

```
<object name="embedded" classid="clsid:83A38BF0-B33A-A4FF-C619A82E891D"
height="150" width="250"></object>
```

值 区分大小写的标识符。

默认值 无。

对象模型引用方式 `[window.]document.getElementById(elementID).name`

standby

IE 6 NN n/a Moz all Saf n/a Op n/a HTML 4

```
standby="HTML 文本"
```

可选

加载object元素时将显示对应的HTML内容。除IE 5/Mac外，此属性尚未应用在大多数的主流浏览器中。正如图片加载时在img元素空间内会alt信息一样，我们也可以假想这个属性指定的信息将显示在object元素的矩形空间内。

示例

```
<object classid="clsid:83A38BF0-B33A-A4FF-C619A82E891D"
height="150" width="250" standby="Loading movie..."></object>
```

值 任意HTML内容均适用。

默认值 无。

type

IE 3 NN 4 Moz all Saf all Op 7 HTML 4

`type="MIME类型"` 必选

对于data属性所指出的数据类型，此属性给出了相应的提示信息。这些信息可以帮助浏览器为需要多媒体播放器或插件的资源提前做好准备。data元素首先会为此信息查看codetype属性。一旦未指定codetype属性，那么浏览器将接着检查type属性的设定。如果这两个属性均未使用，那么浏览器将会尝试着从下载的资源中获取内容类型信息。为了安全起见，最好每次都为图像数据指定一个MIME类型，如image/jpeg或image/gif。

示例 `<object data="movies/prodDemo.mpeg" type="application/mpeg"></object>`

值 不区分大小写的MIME类型。MIME类型目录可从此处获得：*http://www.iana.org/assignments/media-types/*。

默认值 无。

对象模型引用方式 `[window.]document.getElementById(elementID).type`

usemap

IE 6 NN n/a Moz all Saf all Op 7 HTML 4

`usemap="映射URL"` 可选

HTML 4规范为object元素列出了usemap属性。这样一来，就为在object元素中加载图像映射提供了可能性。目前的浏览器已支持此功能。

此时需要将图像的URI指定给data属性，并将type属性设置为该图像的MIME类型。然后再通过一个或多个内嵌的area元素创建一个独立的map元素，并且将map元素的name属性值赋予object元素的usemap属性。由于IE/Windows会填充图像并自动添加滚动条，因此这一功能不能完全实现垮浏览器部署。

示例

```
<object data="navbar.jpg" type="image/jpeg" alt="Navigation Bar" usemap="#navbarMap"
border="0"></object>
```

值 请参见img元素的usemap属性。

默认值 无。

对象模型引用方式 `[window.]document.getElementById(elementID).useMap`

vspace

参见hspace属性。

width

参见height属性。

IE all NN all Moz all Saf all Op 7 HTML all

`<ol>...</ol>` HTML结束标签：必选

ol元素是为有序条目列表提供的一种容器。“有序列表”意味着在显示相关条目时，在每个条目前面会使用序列数字或字符。具体的形式取决于type属性或list-style-type样式单的设定。而通过嵌套的li元素，可定义每一个列表条目的内容。如果为ol元素提供了对应的样式单规则，那么内嵌的li元素也会继承该规则。

示例

```
<ol>
    <li>Choose Open from the File menu.</li>
    <li>Locate the file you wish to edit, and click on the filename.</li>
```

```
    <li>Click the Open button.</li>
</ol>
```

对象模型引用方式 [window.]document.getElementById(*elementID*)
元素专有属性 compact、start、type
元素专有事件处理属性 无。

compact

IE *4* NN *n/a* Moz *all* Saf *all* Op *n/a* HTML *3.2*

```
compact
```

可选

该属性值是一个布尔变量，设计的初衷是让浏览器能够以另一种更为紧凑的样式显示列表，即与正常情况相比，使条目之间的行距更小。尽管为了顾及兼容性，依然支持compact属性，但它在主流浏览器上已失去了作用。请使用样式单来控制元素尺寸大小与行间距。

示例 `<ol compact>...</ol>`
值 如果此属性已出现，就意味着其属性值为true。
默认值 false

start

IE *all* NN *all* Moz *all* Saf *all* Op *7* HTML *all*

```
start="数字"
```

可选

通过此属性可以为ol元素中的条目序列指定一个自定义的起始数字。当条目序列必须被主体内容打断时，这个功能会带来很多便利。尽管此属性值是一个数字，但在type属性控制下会显示对应的阿拉伯数字、罗马数字或字母。

示例 `ol start="5"> ...</ol>`
值 正整数。
默认值 无。
对象模型引用方式 [window.]document.getElementById(*elementID*).start

type

IE *all* NN *all* Moz *all* Saf *all* Op *7* HTML *3.2*

```
type="标注类型"
```

可选

type属性为如何在浏览器中显示序列号提供了一定的灵活性。针对一个有序列表，可使用大/小写字符、大/小写罗马数字或阿拉伯数字。但HTML 4.0已不再赞成使用type属性，推荐使用样式单属性list-style-type。

示例 `<ol type="a">...</ol>`
值

可用的值为A|a|I|i|1。列表会自动按顺序排列，如下表所示：

类型	示例	类型	示例
A	A, B, C, ...	i	i, ii, iii, ...
a	a, b, c, ...	1	1, 2, 3, ...
I	I, II, III, ...		

默认值 1
对象模型引用方式 [window.]document.getElementById(*elementID*).type

<optgroup>

IE *5(Mac)/6(Win)* NN *n/a* Moz *all* Saf *all* Op *7* HTML *4*

```
<optgroup>...</optgroup>
```

HTML结束标签：必选

在一个select元素内，optgroup是option元素的容器。每个optgroup都能代表select元素选择列表项中的

一个子组合。每个浏览器和操作系统均设计了默认的显示方式，以区分optgroup标签与其他可选择的缩进选项。例如，在IE 5/Mac中，optgroup元素出现后，浏览器会将一个弹出菜单变为一种两层的层次菜单。

示例

```
<select name="carCos">
   <optgroup label="American">
      <option value="General Motors">General Motors</option>
      <option value="Ford">Ford Motor Company</option>
      <option value="Chrysler">DaimlerChrysler</option>
   </optgroup>
   <optgroup label="Japanese">
      <option value="Toyota">Toyota</option>
      <option value="Honda">Honda</option>
      <option value="Nissan">Nissan</option>
   </optgroup>
</select>
```

215

对象模型引用方式 [window.]document.getElementById(*elementID*)

元素专有属性 disabled、label

元素专有事件处理属性 无。

disabled

IE *5/6* NN *n/a* Moz *all* Saf *n/a* Op *7* HTML *4*

`disabled` 可选

出现此属性后，将禁用optgroup元素及其内嵌的option元素。其他未使用此属性的optgroup元素依然有效。

示例 `<optgroup label="Engineering" disabled>`

值 如果此属性已出现，就意味着其属性值为true。

默认值 false

对象模型引用方式 [window.]document.getElementById(*elementID*).disabled

label

IE *5/6* NN *n/a* Moz *all* Saf *all* Op *7* HTML *4*

`label="标签文本"` 必选

通过此属性定义为optgroup提供的select元素文本内容。此处应使用纯文本，而非HTML代码，用户也无法从列表中选择该文本。

示例 `<optgroup label="Engineering" disabled>`

值 任意字符串。该字符串应该位于一对引号之中，双引号是推荐的标准格式，也可使用单引号。

默认值 无。

对象模型引用方式 [window.]document.getElementById(*elementID*).label

<option>

IE *all* NN *all* Moz *all* Saf *all* Op *7* HTML *all*

`<option>...</option>` HTML 结束标签：可选

无论select列表是弹出菜单还是滚动列表，都是通过option元素定义的出现在列表中的各个条目。与一个select元素联合使用的option元素必须内嵌在select元素的起始与结束标签之间。

当该元素作为form元素的一部分被提交时，select元素会提供对应的名称/值对。此时，select元素的name属性和选中项的value属性将作为名称/值对进行提交。因此，为选择列表中的每一个option元素指定有意义的value属性值是非常重要的。为了提高易用性，可以使用value属性为用户隐藏不便于理解的属性值（但

216

便于服务器或脚本处理），而在选择列表中显示那些易于用户理解的信息。在option元素的起始标签之后，输入选择列表中供人阅读的信息。由于可以用过下一个option元素的起始标签或select元素的结束标签来界定一个option元素，因此此元素的结束标签是可选的。关于将option元素进行多级菜单分组的详细信息，请参见optgroup属性。

示例

```
<select name="chapters">
    <option value="1">Chapter 1</option>
    <option value="2">Chapter 2</option>
    <option value="3">Chapter 3</option>
    <option value="4">Chapter 4</option>
</select>
```

对象模型引用方式

```
[window.]document.formName.selectName.optionName
[window.]document.forms[i].elements[j].options[k].optionName
[window.]document.getElementById(elementID)
```

元素专有属性 disabled、label、selected、value

元素专有事件处理属性 无。

disabled

IE *5(Mac)* NN *n/a* Moz *all* Saf *n/a* Op *7* HTML *4*

```
disabled
```

可选

出现这个属性时，将禁用选择列表中的对应option元素。值得注意的是，尽管IE 5/Mac会禁止在列表中选择该选项，但直到版本7，IE/Windows并未为此属性提供任何其他的功能。

示例 `<option value="Met101" disabled>Meteorology 101</option>`

值 如果此属性已出现，就意味着其属性值为true。

默认值 false

对象模型引用方式

```
[window.]document.formName.selectName.optionName.disabled
[window.]document.forms[i].elements[j].options[k].optionName.disabled
[window.]document.getElementById(elementID).disabled
```

label

IE *n/a* NN *n/a* Moz *all* Saf *all* Op *7* HTML *4*

```
label="标签文本"
```

必选

考虑到开发者可能会构建多层选择列表，HTML 4.0引入了label属性。在进行分层显示时，HTML 4.0试图将label作为option元素的一种更为短小的代替者。此时，它会代替option元素原有的正常文本。值得注意的是，IE 5/Mac和Safari会错误地替换元素文本并显示label属性值。当前其他浏览器并未实现此属性的功能，但Opera 9在Web Forms 2.0中使用此属性为datalist元素中的条目提供友好的标签提示，例如，当选项的value属性值为URL时，它可以表示一个网页站点的标题。

示例 `<option label="Meteo 101" value="met101">Meteorology 101</option>`

值 任意字符串。该字符串应该位于一对引号之中，双引号是推荐的标准格式，也可使用单引号。

默认值 无。

selected

IE *all* NN *all* Moz *all* Saf *all* Op *7* HTML *all*

```
selected
```

可选

当此属性出现时，可以在select元素中预先选择对应的条目。如果将select元素设置为multiple，那么可

以同时为多个option元素设置selected属性。

示例 `<option value="met101" selected>Meteorology 101</option>`

值 如果此属性已出现，就意味着其属性值为true。

默认值 false

对象模型引用方式

```
[window.]document.formName.selectName.optionName.selected
[window.]document.forms[i].elements[j].options[k].selected
[window.]document.getElementById(elementID).selected
```

value

IE *all* NN *all* Moz *all* Saf *all* Op *7* HTML *all*

```
value="文本"
```

可选

此属性将一个字符串与option元素联系起来，它的具体内容与select元素中显示的文本可以相同也可以不同。当select元素作为表单的一部分被提交至服务器端时，如果用户选择了对应的选项，那么value的属性值将会被指定给select元素的名称/值对。为便于脚本操作，value属性通常会包含URL或对象的字符串表达式，然后脚本将处理这些数据。

示例 `<option value="met101">Meteorology 101</option>`

值 任意字符串。该字符串应该位于一对引号之中，双引号是推荐的标准格式，也可使用单引号。

默认值 无。

对象模型引用方式

```
[window.]document.formName.selectName.optionName.value
[window.]document.forms[i].elements[j].options[k].optionName.value
[window.]document.getElementById(elementID).value
```


<output>

IE *n/a* NN *n/a* Moz *n/a* Saf *n/a* Op *9* HTML *n/a*

```
<output>...</output>
```

HTML 结束标签：必选

在脚本的控制之下，Web Forms 2.0中的output元素可以结合表单控件来显示HTML内容，通常它们是一些表单事件发生后的结果。与span类似，此元素是一个行内元素，但与其他表单元素不同，它的值并不会随着表单一同提交。然而这个元素确实成为了表单的elements数组的成员之一。

当用户与表单互动时，通常使用它显示提示信息或临时信息。例如，可以将一个初始化为空的output元素紧临表单的一个文本输入域放置。然后通过一个事件处理程序监控用户对文本域的输入(通过forminput事件)，当数据输入不正确时，可以检查各种ValidityState对象属性，并显示相关的错误信息。

示例

```
<script type="text/javascript"> // <![CDATA[function validateField(evt) {
  var form = evt.target.form;
  var errField = form.elements[evt.target.name + "Error"];
  if (evt.target.validity.typeMismatch) {
    errField.value = 'You must enter a time (hh:mm).';
  } else if (evt.target.validity.stepMismatch) {
    errField.value = 'Appointments must begin at 0, 15, 30, or 45 past the hour.';
  } else if (evt.target.validity.rangeUnderflow) {
    errField.value = 'The earliest appointment is 9:00 am.';
  } else if (evt.target.validity.rangeOverflow) {
    errField.value = 'The last appointment is 5:00 pm.';
  } else if (evt.target.validity.valueMissing) {
    errField.value = 'You must enter a time.';
  } else {
```

```
    errField.value = '';
  }
  evt.preventDefault( );
// ]]>
}
</script>
...
<form action="..." method="get" >
<p><label>Desired appointment time:
<input type="time" name="apptTime" min="09:00" max="17:00" value="09:00"
  step="900" required="required" onforminput="validateField(event)" />
</label>
<output name="apptTimeError" />
</p>
<p>
<input type="submit" />
</p>
</form>
```

对象模型引用方式 [window.]document.getElementById(*elementID*)

元素专有属性 for、form、name

元素专有事件处理属性 无。

for

IE *n/a* NN *n/a* Moz *n/a* Saf *n/a* Op *n/a* HTML *n/a*

`for="elementID [elementID] ..."` 可选

此属性可用于为与option元素相关的一个或多个其他元素指定ID。具体的使用方式由浏览器决定。

值 由空格分隔的一个或多个元素ID属性。

默认值 无。

对象模型引用方式 无。

form

IE *n/a* NN *n/a* Moz *n/a* Saf *n/a* Op *9* HTML *n/a*

`form=" formID [formID] ..."` 可选

此属性可以将output元素与一个或多个表单联系起来。由于Web Forms 2.0中的output元素并没有像form的子元素那样受到限制，因此它可以放在form元素之外的任何地方。而通过form属性则可以将output元素与页面上一个或多个表单元素联系起来。

input类型 所有的可显示类型。

示例 `<output form="appointmentForm" name="apptTimeError" />`

值 页面上一个或多个form元素的ID值。此时，使用空格分隔多个ID值。

默认值 无。

name

IE *all* NN *all* Moz *all* Saf *all* Op *9* HTML *all*

`name="元素标识符"` 可选

通过表单元素队列，脚本可以使用一个标识符来访问表单对象。当然，也可以使用id属性来指定标识符，并在脚本中使用document.getElementById()。

示例 `<output form="appointmentForm" name="apptTimeError" />`

值 区分大小写的标识符。

默认值 无。

<param>

对象模型引用方式

```
[window.]document.formName.elementName.name
[window.]document.forms[i].elements[j].name
[window.]document.getElementById(elementID).name
```

<p>

IE *all* NN *all* Moz *all* Saf *all* Op *7* HTML *all*

```
<p>...</p>
```

HTML 结束标签：可选

每个p元素均定义了文档中的一个段落结构元素。在HTML 4中，p元素正式成为了一个块级元素，这意味着p元素的内容将从其自身的行开始显示，而p元素的之后的内容将在新的一行中显示。在使用时也不应将其他的块级元素内嵌在一个p元素内。如果忽略了结束标签（XHTML中不允许这样做），那么此元素将在另一个块级元素的起始标签之前自动结束。

p元素的本质随着时间不断改变。在早期的HTML实现中，此元素仅仅表示一个段落换行，即拥有额外行距的一个新行。而从第4版开始，浏览器使用一种混合方式来显示p元素，此时，p元素的起始标签会在块前插入一行空白。这意味着如果不通过CSS进行定位，p元素将无法放置在页面最上端的起始部位。请将p元素作为一种结构元素来使用，而不要作为一种格式化的手段。

需要注意的是，p元素的内容无法识别源代码中的额外的空白内容。而其他元素，例如pre，将会按照源代码排版的格式显示其内容。

示例

```
<p>This is a simple, one-sentence paragraph.</p>
<p>This second paragraph starts on its own line, with a little extra line spacing.</p>
```

对象模型引用方式 `[window.]document.getElementById(elementID)`

元素专有属性 `align`

元素专有事件处理属性 无。

align

IE *all* NN *all* Moz *all* Saf *all* Op *7* HTML *3.2*

```
align="对齐位置"
```

可选

此属性决定了在p元素所占据的矩形盒中，相关段落文本如何对齐。请参考本章前文“对齐常量”中，有关块级元素内文本水平对齐的相关内容。

HTML 4.0已不再赞成使用align属性，推荐使用样式单属性text-align。

示例 `<p align="center">...</p>`

值 文本共有4种对齐值方式：center、justify、left和right，但在严格的HTML或XHTML DTD中，justify无效。

默认值 `left`

对象模型引用方式 `[window.]document.getElementById(elementID).align`

<param>

IE *3* NN *2* Moz *all* Saf *all* Op *7* HTML *3.2*

```
<param>
```

HTML 结束标签：禁用

在加载Java小程序或对象时，可以使用内嵌在applet或object对象内的param元素向它们传递参数。在不须要重新编写小程序或对象的前提下，这些参数为HTML作者调整小程序或对象的设置提供了一些途径。每个参数往往都会传递一个由name和value属性组成的名称/值对。每个applet或object元素均可拥有多个param元素。小程序或对象的帮助文档应该为这些参数传递提供足够的信息。

示例

```
<applet code="simpleClock.class" name="myClock" width="400" height="50">
<param name="bgColor" value="black">
<param name="fgColor" value="yellow">
</applet>
```

对象模型引用方式 [window.]document.getElementById(*elementID*)

元素专有属性 datafld、dataformatas、datasrc、name、type、value、valuetype

元素专有事件处理属性 无。

datafld

IE *4* NN *n/a* Moz *n/a* Saf *n/a* Op *n/a* HTML *n/a*

```
datafld="列名称"
```

可选

此属性用于IE中的数据绑定，以便将一个远程数据源的列名与传递给Java小程序或对象的参数联系在一起。在下面的例子中，从一列名为backColor的数据源获取数据，并指定给value属性，而在元素标签中，这个属性并没有明确地显示出来。另外，在object和applet元素中，还可能存在更为复杂的关系。注意，datafld仅能在IE 5/Mac组合提供的文本文件数据源中正常工作。

示例 <param name="bgColor" datasrc="DBSRC2" dataformatas="text" datafld="backColor">

值 区分大小写的标识符。

默认值 无。

dataformatas

IE *4* NN *n/a* Moz *n/a* Saf *n/a* Op *n/a* HTML *n/a*

```
dataformatas="数据类型"
```

可选

本属性用于IE浏览器中的数据绑定。对于来自于数据源的原始资料，浏览器根据此属性值决定将它们视为纯文本亦或是HTML标签。此属性设置完全依赖于数据源的构成形式及param元素的所期望的数据类型。注意，dataformatas仅能在IE 5/Mac组合提供的文本文件数据源中正常工作。

示例 <param name="bgColor" datasrc="DBSRC2" dataformatas="text" datafld="backColor">

值 IE浏览器可识别两种有效的设定值：text和html。

默认值 text

datasrc

IE *4* NN *n/a* Moz *n/a* Saf *n/a* Op *n/a* HTML *n/a*

```
datasrc="数据源名称"
```

可选

为实现IE数据绑定功能，此属性用于指定页面内object元素的ID值，以便在进行远程数据访问时加载数据源对象。通过datafld属性指定来自于数据源中的绑定内容。注意，datasrc仅能在IE 5/Mac组合提供的文本文件数据源中正常工作。

示例 <param name="bgColor" datasrc="DBSRC2" dataformatas="text" datafld="backColor">

值 区分大小写的标识符。

默认值 无。

name

IE *3* NN *2* Moz *all* Saf *all* Op *7* HTML *3.2*

```
name="元素标识符"
```

必选

通过此属性可以为applet或object元素所期望的参数指定标识符。每个参数往往会提供一个名称/值对。例如，applet对象会包含一个例程，它通过名称提取每个参数并将值传递至applet中的变量。小程序或对象元素的帮助文档应该为param元素提供一份对应的名称与值类型列表。

示例 `<param name="loop" value="4">`

值 区分大小写的标识符。

默认值 无。

type

IE *6* NN *n/a* Moz *all* Saf *all* Op *7* HTML *4*

`type="MIME类型"` **可选**

当valuetype属性设定为“ref”时，由value属性的URL值所引用的文件类型将通过type属性通知浏览器。除此之外，针对valuetype属性的其他设置，type属性均可忽略。

示例

```
<param name="help" value="http://www.example.com/help.html" valuetype="ref" type="text/html">
```

值 MIME类型不区分大小写。MIME类型目录可从此出获得：*http://www.iana.org/assignments/media-types/*。

默认值 无。

对象模型引用方式 `[window.]document.getElementById(elementID).type`

value

IE *3* NN *2* Moz *all* Saf *all* Op *7* HTML *3.2*

`value="运行时参数值"` **可选**

此属性表示在执行程序或加载数据时，将传递给applet或object元素的参数值。参数值将通过字符串的形式进行传递，而applet或object会在其内部将数据转化成所需要的类型。有时对object而言，param元素的name属性已经完全足够，因此并非必须列出value属性。值得注意的是，除非可以通过脚本操作applet或object，或者它们已暴露了一些用于修改参数值的方法，否则当它们加载相关的参数之后，脚本就无法动态修改这些参数值。

示例 `<param name="loop" value="4">`

值 任意字符串。

默认值 无。

对象模型引用方式 `[window.]document.getElementById(elementID).value`

valuetype

IE *6* NN *n/a* Moz *all* Saf *all* Op *7* HTML *4*

`valuetype="参数值类型"` **可选**

object元素参数共有3种不同类型：data、object和ref。valuetype属性使用这些常量来通知浏览器如何将value的属性值传递至对象。当valuetype为data类型时，通过纯文本方式传递value属性。当valuetype为object时，value属性包含一个标识符，它一般是同一文档中另一个object对象的id属性值。这个对象可能已经设置了declare属性，现在就可以通过传递参数值来实例化该对象。而使用ref时，value属性值为一个URL地址，它指向一个文件或存储运行时参数值的其他资源。

示例

```
<param name="anime" value="http://www.example.com/params/animation.txt" valuetype="ref"
type="text/html">
```

值 有3种可用的常量值：data | object | ref。

默认值 data

对象模型引用方式 `[window.]document.getElementById(elementID).valueType`

<plaintext>

IE *all* NN *all* Moz *all* Saf *all* Op *7* HTML *<4*

```
<plaintext>...</plaintext>
```

HTML 结束标签：可选

作为一个块级元素，plaintext元素将通过等宽字体显示它的内容。位于其起始标签之后的所有的文档源代码均会以纯文本的形式原样显示。须要注意的是，一旦使用了plaintext元素，就无法将其关闭。甚至其结束标签也会原样显示出来。很早以前HTML就不再推荐使用此元素，HTML 4.0规范也删除了这个元素。因此，建议使用pre元素代替此元素。

示例

```
<p>The rest of the HTML code follows:</p>
<plaintext>
...
</plaintext>
```

对象模型引用方式 [window.]document.getElementById(*elementID*)

元素专有属性 无。

元素专有事件处理属性 无。

<pre>

IE *all* NN *all* Moz *all* Saf *all* Op *7* HTML *all*

```
<pre>...</pre>
```

HTML 结束标签：必选

pre元素定义了一个包含预格式化文本的数据块。在pre元素中，预格式化的文本通常使用等宽字体显示，更为重要的是，相关内容中的空白部分（字间的多个空格及空白行）会完全保留在源代码中。与plaintext元素不同，pre元素并不会忽略HTML标签。相反，它会跳过这些标签以进行正常的显示。如果须要在预定义文本块中显示HTML标签，那么请使用“<”和“>”代替“<”和“>”符号。这样就可以防止这些标签被解释为真正的HTML标签，但又保证能够在预定义文本中正常显示它们。

为了在源代码的新一行内开始显示相关内容，浏览器将忽视紧随在pre元素的起始标签之后的空白换行符，请参考下文中的示例代码。

HTML 4规范则坚持在pre元素中保留其等宽字体的尺寸和行间距。下列元素不应包含在pre元素中：applet、basefont、big、font、img、object、small、sub和sup。这些元素会破坏pre元素的固定字符间距。同时，HTML 4推荐标准还鼓励作者们避免通过样式单来改变等宽字体设置。

最后一个警告与在pre元素内使用tab进行缩进或文本对齐有关。浏览器不会以完全相同的方式显示tab符号。因此，可以使用空格符号或让pre元素保留的空白区域来完成排版的工作，以避免出现潜在的问题。在pre元素中，无须使用不间断空格（ ）。

示例

```
<p>Here is the script example:</p>
<pre>
&lt;script language="JavaScript"&gt;
   document.write("Hello, world.")
&lt;/script&gt;
</pre>
```

对象模型引用方式 [window.]document.getElementById(*elementID*)

元素专有属性 cols、width、wrap

元素专有事件处理属性 无。

<q>

cols

IE *n/a* NN *all* Moz *all* Saf *n/a* Op *n/a* HTML *n/a*

226

```
cols="列数量"
```

可选

此属性表示预格式化代码中每行文本所容纳的最大字符数量。这个专有属性会自动将wrap属性设置为true。如果不使用此属性，那么源代码的原有格式（或width属性）将控制行宽。

示例 `<pre cols="80">...</pre>`

值 任意正整数。

默认值 无。

width

IE *n/a* NN *n/a* Moz *all* Saf *n/a* Op *n/a* HTML *4*

```
width="列数"
```

可选

HTML 4规范引入了width属性以设置每行预格式化文本能够显示的最大字符数。HTML 4假定将来能够支持这一属性的浏览器会包装文本行，以使得文字不会从中间断裂开来。如果不使用此属性，那么源代码的原有格式将控制行宽。Navigator和Mozilla支持cols属性的这种功能，而其Mozilla还支持width属性。须要注意的是，CSS的width属性无法影响此元素。

示例 `<pre width="80">...</pre>`

值 任意正整数。

默认值 无。

wrap

IE *n/a* NN *all* Moz *all* Saf *n/a* Op *n/a* HTML *n/a*

```
wrap
```

可选

wrap属性出现时，它会命令Navigator与Mozilla浏览器对预格式化文本进行包装，以使得这些文本不会超出浏览器窗口或框架的右边缘。设置cols属性后，会自动将wrap设置为true。

示例 `<pre wrap>...</pre>`

值 一旦出现此属性，就表示其值为true。

默认值 false

<q>

IE *4* NN *n/a* Moz *all* Saf *all* Op *7* HTML *4*

```
<q>...</q>
```

HTML 结束标签：必选

q元素主要用于在文档中突出一行引用语。HTML 4规范指出，浏览器应该根据文档语言自动为q元素的内容加上引用符号，这样一来，网页作者则无须使用引号。除Windows系统下的IE浏览器（从版本7开始）外，所有的主流浏览器均支持这一需求。虽然在IE 5/Mac中也会插入引号，但如果元素的父容器不是左对齐，那么引号会浮动到错误的位置。如果需要在跨浏览器环境中为引用的文字加上引号，那么只能自己为它们加上引号而不是借助于q元素。请参见blockquote元素，以了解关于块级引用的相关信息。

227

示例

```
<p>The preamble to the u.s. Constitution begins,<q>We the People of the United States</q></p>
```

对象模型引用方式 [window.]document.getElementById(*elementID*)

元素专有属性 cite

元素专有事件处理属性 无。

cite

IE 6 NN n/a Moz all Saf all Op 7 HTML 4

`cite="URL"` 可选

此属性指出一个在线原文档的URL地址，该引用语就出自此文档。要注意的是，即使设置此属性也根本无法从另一个文档中复制或摘录任何内容。至于这个属性存在的原因，也许是因为HTML 4推荐标准希望鼓励未来的浏览器和搜索引擎能够利用这些在线资源，为读者和上网者提供便利。IE 6尚未为该属性提供实际的功能。

值 一个指向网络上某文档的有效URL链接，既可使用相对URL也可以使用绝对URL地址。

默认值 无。

<rb>

IE n/a NN n/a Moz n/a Saf n/a Op n/a HTML X1.1

`<rb>...</rb>` 结束标签：必选

在使用深色加强显示的内容区域中，rb元素用于表示其中的基础文本。添加ruby注释后，rb元素内的文本仍然是通过正常的方式进行显示。IE 5及后续浏览器均支持使用ruby来控制文本，但它们并未明确支持rb元素。同时，与装入rt元素的文本不同，rb元素会自动应用于文本之上。

示例

```
<ruby>
  <rb>03</rb><rt>Month</rt>
  <rb>04</rb><rt>Day</rt>
  <rb>2003</rb><rt>Year</rt>
</ruby>
```

对象模型引用方式 `[window.]document.getElementById(elementID)`

元素专有属性 无。

元素专有事件处理属性 无。

<rbc>, <rtc>

IE n/a NN n/a Moz n/a Saf n/a Op n/a HTML X1.1

`<rbc>...</rbc> <rtc>...</rtc>` 结束标签：必选

如果希望将相邻的连续ruby基础项及其相关的ruby文本项串联起来，那么可以将它们分别放置在rbc和rtc容器中以便进行分组。在rbc和rtc元素中的条目数量应该相同，这样浏览器才能将对应的基础项与ruby文本放置在一起。在不支持ruby文本的浏览器中，使用这种手段也无法正常降格显示相关内容。

示例

```
<ruby>
  <rbc>
    <rb>03</rb><rb>04</rb><rb>2003</rb>
  </rbc>
  <rtc>
    <rt>Month</rt><rt>Day</rt><rt>Year</rt>
  </rtc>
</ruby>
```

对象模型引用方式 `[window.]document.getElementById(elementID)`

元素专有属性 无。

元素专有事件处理属性 无。

<rp>

IE 5 NN n/a Moz n/a Saf n/a Op n/a HTML X1.1

`<rp>...</rp>` 结束标签：必选

对那些不能直接支持ruby标记的浏览器而言，rp元素可以加强浏览器之间的兼容性。不支持ruby的浏览器会

将rp和rt元素的内容视为行内文本。rp元素让网页作者有机会在ruby文本周边引入括号或其他字符，使这些文本看起来就像是行内的标签。从传统意义上来说，每个rp元素的内容都可以被看作一个左或右括号。在每个rt元素的前部和后部都伴随着一个完整的rp元素。支持ruby文本的浏览器将忽视rp元素内容，与此同时，也不会显示这些括号。

示例

```
<ruby>
  <rb>03</rb><rp>(</rp><rt>Month</rt><rp>)</rp>
  <rb>04</rb><rp>(</rp><rt>Day</rt><rp>)</rp>
  <rb>2003</rb><rp>(</rp><rt>Year</rt><rp>)</rp>
</ruby>
```

对象模型引用方式 [window.]document.getElementById(*elementID*)

元素专有属性 无。

元素专有事件处理属性 无。

<rt>

IE *5* NN *n/a* Moz *n/a* Saf *n/a* Op *n/a* HTML *X1.1*

```
<rt>...</rt>
```

结束标签：必选

rt元素中所包含的文本内容，一般是对应rb元素的一种注释与说明。支持ruby文本的浏览器在显示rt元素时，其字体大小往往会比基本文本的字体略小。通过样式单，可以为它们替换字体集。另外，还可以通过rt元素的xml:lang属性为ruby文本指定另一种语言设置。

示例

```
<ruby>
  <rb>03</rb><rt>Month</rt>
  <rb>04</rb><rt>Day</rt>
  <rb>2003</rb><rt>Year</rt>
</ruby>
```

对象模型引用方式 [window.]document.getElementById(*elementID*)

元素专有属性 name、rbspan

元素专有事件处理属性 无。

name

IE *5* NN *n/a* Moz *n/a* Saf *n/a* Op *n/a* HTML *n/a*

```
name="元素标识符"
```

可选

本属性用于为rt元素设置一个标识符。

值 该字符串区分大小写。

默认值 无。

rbspan

IE *n/a* NN *n/a* Moz *n/a* Saf *n/a* Op *n/a* HTML *X1.1*

```
rbspan="整数值"
```

可选

在某些情况下，一个rt元素可能要跨越两个或多个连续的rb元素。此时可以将rb元素的数量指定给rt元素的rbspan属性。这一机制与td元素的colspan属性很类似。

值 表示rb元素数量的整数值。

默认值 1

<ruby>

IE 5 NN n/a Moz n/a Saf n/a Op n/a HTML X1.1

<ruby>...</ruby> 结束标签：必选

ruby文本是一种小字体的注释文本，它往往出现在主要正文之前或之后，而在竖直排列的书写系统中，则出现在正文的一侧。这个名称来自于印刷术中的一种小字体，它常用于设计较小的注释文本。在象形文字中，ruby文本很常见，此时ruby文本可以为象形文字符号提供发音指南。当然，ruby文本也同样适用于拉丁字母系统的语言中。

包括正文在内，对所有受ruby标记影响的内容而言，ruby元素是一种主要的容器。此时，主文本称之为“ruby基础文本”而注释则被成为“ruby文本”。以上的两种类型各自拥有一种标签（分别为rb和rt），并且这些标签都必须装入ruby元素之中。从IE 5开始，Windows和Mac系统中的IE浏览器开始支持ruby标记。

W3C对ruby标记规范的开发与HTML推荐标准彼此独立，W3C还将ruby标记添加至XHTML 1.1标准，使之成为第一个利用XHTML扩展特性的模块。

示例

```
<ruby>
  <rb>03</rb><rt>Month</rt>
  <rb>04</rb><rt>Day</rt>
  <rb>2003</rb><rt>Year</rt>
</ruby>
```

对象模型引用方式 [window.]document.getElementById(*elementID*)

元素专有属性 name

元素专有事件处理属性 无。

name

IE 5 NN n/a Moz n/a Saf n/a Op n/a HTML n/a

name="元素标识符" 可选

本属性用于为元素设置一个标识符。

值 该字符串区分大小写。

默认值 无。

<s>

IE 3 NN 3 Moz all Saf all Op 7 HTML 3.2

<s>...</s> HTML结束标签：必选

浏览器将以删除线的形式显示s元素中的文字内容。此元素与strike元素完全相同，采用该元素的主要原因是因为它的名称与其他单字符排版元素的名称更为相似，如b、i和u元素等。无论如何，HTML 4都不再推荐使用s和strike元素，而建议使用样式单属性text-decoration:line-through。

示例

```
<p>If at first you don't succeed, <s>do it over</s> try, try again.</p>
```

对象模型引用方式 [window.]document.getElementById(*elementID*)

元素专有属性 无。

元素专有事件处理属性 无。

<samp>

IE all NN all Moz all Saf all Op 7 HTML all

<samp>...</samp> HTML结束标签：必选

在HTML 4推荐标准中，有一类元素统称为**短语元素**（phrase element），samp元素也是其中之一。这些元素

为文档的特定部分指明了结构上的含义。samp元素作为一种容器，包含了一些来自于计算机程序或脚本的输出片段。它与代码范例不同，代码范例通常使用code元素进行封装。

浏览器可以自由决定如何（或是否）从其他body元素中区分出samp元素的内容。目前的主流浏览器使用等宽字体来显示这些的文本内容。可以使用任何合适的样式单来替换默认的显示形式。

示例

```
<p>When you press the Enter key, you will see <samp>Hello, world!</samp> on the screen.</p>
```

对象模型引用方式 [window.]document.getElementById(*elementID*)

元素专有属性 无。

元素专有事件处理属性 无。

<script>

IE *3* NN *2* Moz *all* Saf *all* Op *7* HTML *4*

```
<script>...</script>
```

HTML结束标签：必选

对于浏览器能够解释的任意脚本语言，script都为它们的脚本代码提供了一种容器。页面载入时，会执行所有未写入方法定义的脚本指令，而脚本方法则要通过用户或系统响应（事件）来显式调用。在同一个文档中，可以包含多个script元素，与此同时，这些脚本元素还可以使用不同的脚本语言进行编写。

伴随着HTML 4的出现，也为属性语法引入了一个重要的转换机制。为了明确指明script元素中的脚本语言类型，第一款支持脚本的浏览器就开始使用language属性对其进行说明。HTML 4不再推荐使用该属性，它建议使用能够表示MIME类型的type属性。除非能够确定页面访问者仅使用支持type属性的新型浏览器，否则在文档中最好同时使用这两个属性，以便为老式浏览器提供良好的兼容性。而且在过渡版DTD中，language属性也是有效的。

所有支持脚本的浏览器均允许通过外部文档引入脚本指令，此时可以使用src属性来说明外部文档的URL。

而那些老旧的，不支持脚本的浏览器，则无法识别script元素，它们可能将脚本指令以普通HTML内容的形式显示出来。为了防止这种现象发生，一贯的经验做法是将脚本指令放置在一个HTML注释标记之内。在结束注释标记（-->）之前，必须放置一个JavaScript注释标记（//），这样就可以避免出现脚本错误。

如果在XHTML中部署页面，那么就必须在一个XML CDATA分段中封装脚本指令，以便让XHTML校验器接受一些在XML中被视为非法字符的脚本符号（如，<和&）。与此同时，由于当前大多数正在使用的浏览器都无法识别CDATA标记，因此将CDATA标记与脚本注释符号混合在一起，是一种能够调和所有不同类型的浏览器和校验器的折中办法，示例如下。当然，也可以通过引入额外的脚本文件（.js）来规避这个问题。

示例

```
<script type="text/javascript" language="JavaScript">
// <![CDATA[
function howdy( ) {
   alert("Hello, HTML world!");
}
// ]]>
</script>

<script type="text/javascript" src="scripts/myscript.js"></script>
```

元素专有属性 charset、defer event、for、language、src、type、version、xml:space

元素专有事件处理属性 无。

charset

IE *n/a* NN *n/a* Moz *all* Saf *all* Op *7* HTML *4*

```
charset="字符集"
```

可选

此属性定义了src属性引用的文件中所使用的字符编码方式。

示例 `<script charset="csISO5427Cyrillic" src="moscow.js">... </script>`

值 是否区分大小写取决于字符集注册表，请参见*http://www.iana.org/assignments/character-sets*。

默认值 由浏览器决定。

defer

IE *4* NN *n/a* Moz *n/a* Saf *n/a* Op *n/a* HTML *4*

`defer`

可选

出现defer属性后，它会命令浏览器在加载页面时直接显示正常的HTML内容，而不必查找脚本以生成相关内容。对浏览器而言，这只是一个建议属性。由于浏览器可以在搜索document.write()指令时解析script元素的内容，因此它并不需要暂缓显示更多的HTML内容。在目前的浏览器中，仅有IE浏览器会响应defer属性。

示例 `<script type="text/javascript" language="JavaScript" defer>...</script>`

值 如果此属性已出现，就意味着其属性值为true。

默认值 false

对象模型引用方式 `[window.]document.getElementById(elementID).defer`

event

IE *4* NN *n/a* Moz *n/a* Saf *all* Op *7* HTML *4*

`event="事件名称"`

可选

在event和for属性的帮助下，IE浏览器的事件模型允许将对象事件与script元素进行绑定。加载页面时，浏览器会对每个script元素及其事件与对象绑定进行注册，这样一来，并不须要为对象编写事件处理程序或也不必将脚本指令放置在方法定义之中，当对象产生事件时，script元素中的脚本指令就可直接执行。

事件值的书写形式既可以是不带引号的事件名称，也可以是方法形式的带引号的事件名称（即事件名称后跟括号及可选参数名）。如有可能，请仅在IE浏览器中使用这种类型的脚本事件绑定方式。其他浏览器将在页面加载时就会执行脚本指令。为便于将来使用，过渡版HTML 4 DTD保留了这个属性，但XHTML 1.0并未保留此属性。

示例 `<script for="window" event="onresize( )">...</script>`

值 事件名称或置于引号之中的方法形式的事件名，均区分大小写。for属性中描述的对象必须支持event属性中命名的事件。

默认值 无。

对象模型引用方式 `[window.]document.getElementById(elementID).event`

for

IE *4* NN *n/a* Moz *n/a* Saf *all* Op *7* HTML *4*

`for="元素 ID"`

可选

在event和for属性的帮助下，IE浏览器的事件模型允许将对象事件与script元素进行绑定。加载页面时，浏览器会对每个script元素及其事件与对象绑定进行注册，这样一来，就不需要为对象编写事件处理程序或也不必将脚本指令放置在方法定义之中，当对象产生事件时，script元素中的脚本指令就可直接执行。此时，对于需要处理的事件，请使用对应元素的id属性值。需要注意的是，仅能在IE浏览器中使用这种类型的脚本事件绑定方式。其他浏览器将在页面加载时就会执行脚本指令。为便于将来使用，过渡版HTML 4 DTD保留了这个属性，但XHTML 1.0并未保留此属性。

示例 `<script for="firstNameEntry" event="onchange( )">...</script>`

值 事件发生元素的ID值，区分大小写。for属性中描述的对象必须支持在event属性中命名的事件。

默认值 无。

对象模型引用方式 `[window.]document.getElementById(elementID).htmlFor`

language

IE 3 NN 2 Moz all Saf all Op 7 HTML 4

```
language="脚本语言类型"
```

可选

此属性设置了元素中所包含的脚本指令的脚本语言类型。虽然HTML 4已不再推荐使用此属性，且建议使用type属性，但由于自脚本浏览器诞生之日起，language属性就被广泛地使用在页面上，因此在未来相当长的一段时间里，依然会支持该属性。此外，距离JavaScript一统天下的日子还遥遥无期（请参见第5章的Array对象），因此该属性还有存在的必要。

235

示例 `<script language="JavaScript">...</script>`

值 IE 4及后续版本的浏览器可识别5种不区分大小写的脚本语言：JavaScript、JScript、vbs、vbscript及ECMAScript。Navigator和Mozilla则仅能识别JavaScript。而Safari和Opera则可以识别JavaScript、Jscript和ECMAScript。

不同的浏览器也支持不同版本的JavaScript脚本。为了将属性值保持为单词汇的标识符，在“JavaScript”语言名称之后附加了其版本号。下表总结了每种JavaScript版本引入时对应的浏览器型号。

版本	首先引入的浏览器	版本	首先引入的浏览器
JavaScript	NN 2、IE 3	JavaScript 1.5	Mozilla 1.8
JavaScript 1.1	NN 3	JavaScript 1.6	Mozilla 1.8
JavaScript 1.2	NN 4、IE 4	JavaScript 1.7	Firefox 2.0

而对于目前的主流浏览器，IE 7仍然在使用JavaScript 1.3，而Safari和Opera则已经更新至JavaScript 1.5。

当script元素指定的版本比浏览器能够支持的版本更新时，旧版本的浏览器将忽略这些script元素。

默认值 JScript（IE）、JavaScript（其他浏览器）。

对象模型引用方式 `[window.]document.getElementById(elementID).language`

src

IE 4 NN 3 Moz all Saf all Op 7 HTML 4

```
src="URL"
```

可选

可以通过此属性从一个外部文件内导入脚本指令。外部指令一旦被加载，它们将与嵌入在主HTML文档中的指令有相同的地位。

从理论上说，对于一个加载了外部脚本库文件的script元素而言，可以向它添加其他脚本指令。但在实际使用中，会为每个外部库文件和文档内的脚本指令提供一个单独的script元素，这种做法往往更为可靠。

在当前的浏览器实现中，src属性仅限制于指向JavaScript类型的外部文件。这些文件必须拥有“*.js*”形式的文件扩展名，而且服务器在支持这种扩展名的同时，还必须设置为“application/x-javascript” MIME类型，以便提供对应的文件服务。

236

在XHTML文档中指定src属性时，浏览器也许不适应那种快捷形式的结束标签。在实际使用中，不要将这个标签视为一个空洞的元素，而应该将其看作一个引用外部资源的有用载体。最后提醒一点，编写代码时请明确使用</script>结束标签。

示例

```
<script language="JavaScript" type="text/javascript" src="stringParseLib.js"></script>
```

值 任何有效的URL。当前浏览器须要以“*.js*”作为结尾的文件扩展名形式。对那些使用了这项功能的早期浏览器而言，完整URL路径也许会解决一些潜在的问题。

默认值 无。

对象模型引用方式 `[window.]document.getElementById(elementID).src`

type

IE 4 NN n/a Moz all Saf all Op 7 HTML 4

```
type="MIME类型"
```

必选

本属性为脚本指令的类型给出了提示。这个类型会通知浏览器使用何种脚本引擎来解释脚本指令。type属性最终将会取代language属性，后者主要是用来定义编写元素指令时所使用的脚本语言种类。但为了兼容以前和将来的各种浏览器，最好在script元素中同时使用language和type属性。

示例 `<script type="text/javascript" language="JavaScript">...</script>`

值 MIME类型，且不区分大小写。属性值中包含的MIME类型必须已被浏览器所支持。在兼容ECMAScript的语言中，IE 4和后续版本的IE浏览器，以及其他各种支持脚本的浏览器均能接受“text/javascript”和“application/x-javascript”这两种MIME类型。IE浏览器还可以接受以下几种类型：text/ecmascript、text/jscript、text/vbs（IE/Windows），text/vbscript（IE/Windows）及text/xml（IE 5及后续版本）。

默认值 无。

对象模型引用方式 `[window.]document.getElementById(elementID).type`

version

IE n/a NN n/a Moz all Saf n/a Op n/a HTML n/a

```
version="x.y"
```

可选

由于基于Mozilla的浏览器在将来可能实现这一属性，因此也列出这个version。目前只有在Mozilla浏览器引擎中才有支持此属性的相关内容，但尚未将它们联系起来。

示例 `<script type="text/javascript" version="1.5">...</script>`

值 由主要与次要版本号来表示的语言版本，两个整数间使用句号进行分隔。

xml:space

IE n/a NN n/a Moz n/a Saf n/a Op n/a HTML X1.0

```
xml:space="preserve"
```

可选

在处理文档时，XHTML分析器会删除源代码中的全部空格。但这一做法也可能会损害到脚本。而通过引入XML名空间的space属性，就可以命令分析器完整地保留脚本元素内容中源代码的空格。

示例 `<script type="text/javascript" xml:space="preserve">...</script>`

值 常量：preserve。

默认值 无。

<select>

IE all NN all Moz all Saf all Op 7 HTML all

```
<select>...</select>
```

HTML结束标签：必选

通过滚动列表或弹出菜单的形式，select元素可在文档中显示其内嵌的option元素的相关信息。通常情况下，用户只能从列表中选择一项。而当其size属性值大于1，且设置了multiple属性时，用户即可在滚动列表中选择多项。对于已选中的option项，其value属性值会作为名称/值对中的值同表单一起传递至服务器端。当设置此元素以允许进行多项选择后，提交时多个名称/值对之间会以“&”作为分隔。在这些名称/值对中，名称往往相同，而值不同。在Navigator 4中，select元素必须放置在一个form元素内。

示例

```
<select name="chapters">
   <option value="chap1.html">Chapter 1</option>
   <option value="chap2.html">Chapter 2</option>
   <option value="chap3.html">Chapter 3</option>
```

```
   <option value="chap4.html">Chapter 4</option>
</select>
```

对象模型引用方式

```
[window.]document.formName.selectName
[window.]document.forms[i].elements[j]
[window.]document.getElementById(elementID)
```

元素专有属性 accesskey、align、autofocus、data、datafld、datasrc、isabled、form、multiple、name、size、tabindex

元素专有事件处理属性

238

处理程序	IE	NN	Opera	其他	HTML
onafterupdate	4	n/a	n/a	n/a	n/a
onbeforeupdate	4	n/a	n/a	n/a	n/a
onchange	3	2	all	all	4
onformchange	n/a	n/a	9	n/a	n/a
onforminput	n/a	n/a	9	n/a	n/a
oninvalid	n/a	n/a	9	n/a	n/a

accesskey

IE *4* NN *n/a* Moz *n/a* Saf *n/a* Op *n/a* HTML *n/a*

```
accesskey="字符"
```

可选

请参见本章前文，在共享属性中已对其进行了详细说明。

align

IE *4* NN *n/a* Moz *n/a* Saf *n/a* Op *n/a* HTML *n/a*

```
align="对齐常量"
```

可选

此属性决定了select元素在环绕四周的上下文中如何对齐，特别是当其size属性值大于1时的对齐特点。请参见本章前文中，“对齐常量”一节。须要注意的是，仅IE浏览器支持在select元素中使用align属性。

示例 `<select name="chapters" multiple align="baseline">...</select>`

值 不区分大小写的对齐常量。

默认值 bottom（IE/Windows）；absmiddle（IE/Macintosh）。

对象模型引用方式

```
[window.]document.formName.selectName.align
[window.]document.forms[i].elements[j].align
[window.]document.getElementById(elementID).align
```

autofocus

IE *n/a* NN *n/a* Moz *n/a* Saf *n/a* Op *n/a* HTML *n/a*

```
autofocus="自动聚焦"
```

可选

当页面加载完成后，这个Web Forms 2.0扩展会将焦点聚集在元素上。注意，每个页面上只能有一个表单控件元素能拥有此属性。

data

239

IE *n/a* NN *n/a* Moz *n/a* Saf *n/a* Op *n/a* HTML *n/a*

```
data="URI"
```

可选

这个Web Forms 2.0扩展允许表单从一个外部XML文件中获取表单控件的初始值。Web Forms 2.0规范为使用XML文件提供了详细的结构和命名空间说明。如须获取更多的信息请访问此链接：*http://www.whatwg.org*。

示例 `<select name="departments" data="form/departments.xml">...</select>`

值 统一资源标识符。

默认值 无。

datafld

IE 4 NN n/a Moz n/a Saf n/a Op n/a HTML |4|

```
datafld="列名称"
```

可选

此属性用于IE浏览器下的数据绑定，它将远程数据源的列名称与select元素的selectedIndex属性联系起来，例如，列表中当前选择的条目的序列值，请参见第2章中的select对象。这样一来，只要不指定multiple属性，就可在select元素中实现数据绑定。另外，元素中还必须设定datasrc属性。注意，只能在IE 5/Mac操作系统下实现文本文件数据源的绑定。

HTML 4中依然保留此属性，但它已被XHTML 1.0删除。

示例

```
<select name="chapters" datasrc="DBSRC3" datafld="chapterRequest">
    <option value="chap1.html">Chapter 1</option>
    <option value="chap2.html">Chapter 2</option>
    <option value="chap3.html">Chapter 3</option>
    <option value="chap4.html">Chapter 4</option>
</select>
```

值 区分大小写的标识符。

默认值 无。

对象模型引用方式

```
[window.]document.formName.selectName.dataFld
[window.]document.forms[i].elements[j].dataFld
[window.]document.getElementById(elementID).dataFld
```

datasrc

IE 4 NN n/a Moz n/a Saf n/a Op n/a HTML |4|

```
datasrc="数据源名称"
```

可选

为实现IE数据绑定功能，此属性用于指定页面内object元素的ID值，以便在进行远程数据访问时加载数据源对象。通过datafld属性可指定来自于数据源中的绑定内容。注意，只能在IE 5/Mac操作系统下实现文本文件数据源的绑定。

HTML 4中依然保留此属性，但它已被XHTML 1.0删除。

示例

```
<select name="chapters" datasrc="#DBSRC3" datafld="chapterRequest">
    <option value="chap1.html">Chapter 1</option>
    <option value="chap2.html">Chapter 2</option>
    <option value="chap3.html">Chapter 3</option>
    <option value="chap4.html">Chapter 4</option>
</select>
```

值 区分大小写的标识符。

默认值 无。

对象模型引用方式

```
[window.]document.formName.selectName.dataSrc
[window.]document.forms[i].elements[j].dataSrc
[window.]document.getElementById(elementID).dataSrc
```

disabled

IE 4 NN n/a Moz all Saf all Op 7 HTML 4

```
disabled
```

可选

出现此属性后，将禁用整个select元素及其内嵌的option元素。一旦元素被禁用，将无法接收任何事件。也可以通过option元素的disabled属性来禁用列表内的个别选项。

示例

```
<select name="chapters" disabled>
   <option value="chap1.html">Chapter 1</option>
   <option value="chap2.html">Chapter 2</option>
   <option value="chap3.html">Chapter 3</option>
   <option value="chap4.html">Chapter 4</option>
</select>
```

值 如果此属性已出现，就意味着其属性值为true。

默认值 false

对象模型引用方式

```
[window.]document.formName.selectName.disabled
[window.]document.forms[i].elements[j].disabled
[window.]document.getElementById(elementID).disabled
```

form

IE n/a NN n/a Moz n/a Saf n/a Op 9 HTML n/a

```
form=" formID [formID] ..."
```

可选

通过这个Web Forms 2.0扩展，无论表单是否嵌入了某控件，都可以将一个单独的表单控件元素与一个或多个表单联系在一起。由于Web Forms 2.0中的select元素并没有像form的子元素那样受到限制，因此它可以放在form元素之外的任何地方。通过form属性可以将select元素与页面上一个或多个表单元素联系起来。

input类型 所有的可显示类型。

示例

```
<select id="chapters" form="bookSearch"/>
```

值 页面上一个或多个form元素的ID值。此时，使用空格分隔多个ID值。

默认值 无。

multiple

IE all NN all Moz all Saf all Op 7 HTML all

```
multiple
```

可选

multiple元素出现后，它会命令浏览器将select显示成为一个列表框的形式，并允许用户在列表中选择多个选项。在默认情况下，size属性值与内嵌的option元素个数相当，但也可以修改其属性值设定。用户按住Shift键后，就可以选择分组中第一个和最后一个条目之间的连续项。如果要选择非连续项，那么Windows用户必须在选择每项时按住Ctrl键，而Mac用户则须要按下Command。当size设置为1，就只会显示一个弹出菜单，此时multiple属性将会失效。

示例

```
<select name="equipment" multiple>
   <option value="monitor">Video monitor</option>
   <option value="modem">Modem</option>
   <option value="printer">Printer</option>
   ...
</select>
```

值 如果此属性已出现，就意味着其属性值为true。

默认值 false

对象模型引用方式

```
[window.]document.formName.selectName.multiple
[window.]document.forms[i].elements[j].multiple
[window.]document.getElementById(elementID).multiple
[window.]document.formName.selectName.type
[window.]document.forms[i].elements[j].type
[window.]document.getElementById(elementID).type
```

name

IE *all* NN *all* Moz *all* Saf *all* Op *7* HTML *all*

`name="元素标识符"` 可选

此属性确定随表单一起提交的元素名称/值对中的名称部分。它的作用与input元素中的name属性相同。

示例

```
<select name="cpu" id="cpu">
    <option value="486">486</option>
    <option value="pentium">Pentium</option>
    <option value="pentium2">Pentium II</option>
    ...
</select>
```

值 区分大小写的标识符。

默认值 无。

对象模型引用方式

```
[window.]document.formName.selectName.name
[window.]document.forms[i].elements[j].name
[window.]document.getElementById(elementID).name
```

size

IE *all* NN *all* Moz *all* Saf *all* Op *7* HTML *all*

`size="行数量"` 可选

此属性可以控制出现在select元素中的option元素的数量。当size值为1时，select元素以弹出菜单的形式显示其内容，而当其值大于1时，则以列表框的形式显示选择项。根据内嵌的最宽的option元素文本长度，浏览器将自动控制select元素的宽度。

示例

```
<select name="equipment" size="3">
    <option value="monitor">Video monitor</option>
    <option value="modem">Modem</option>
    <option value="printer">Printer</option>
    ...
</select>
```

值 任意正整数。

默认值 1

对象模型引用方式

```
[window.]document.formName.selectName.size
[window.]document.forms[i].elements[j].size
[window.]document.getElementById(elementID).size
```

tabindex

IE *4* NN *n/a* Moz *all* Saf *all* Op *7* HTML *4*

`tabindex="整数值"` 可选

此属性可用来促进复杂表单控件的可操作性。一旦聚焦在select元素，那么用户就可以连续使用键盘来进行选择。请参见本章前述部分中关于此属性的相关讨论。

<small>

IE 3 NN 2 Moz all Saf all Op 7 HTML 3.2

243

```
<small>...</small>
```

HTML 结束标签：必选

根据元素之前文本的字体尺寸，small元素会以小一号的字体尺寸呈现其内容。考虑到font元素通过1–7的数字来表示字体尺寸，small元素在显示其内容时会比之前的文本尺寸小一号。通过指定<font size=-1>也可以达到同样的效果。

示例 `<p>Let's get really <small>small</small>.</p>`

对象模型引用方式 `[window.]document.getElementById(elementID)`

元素专有属性 无。

元素专有事件处理属性 无。

<spacer>

IE n/a NN 3 Moz n/a Saf n/a Op n/a HTML n/a

```
<spacer>
```

HTML 结束标签：禁用

为了在不使用“ ”、连续的<p>标签或透明图片的前提下就能够创建空白区域，Navigator 3引入了spacer元素。此元素可以在一行文本之内、行间创建空白区域，或直接作为一个空白矩形来使用。当然，也可以通过跨浏览器的样式单技术来实现它的部分功能。须要注意的是，仅在Navigator 3和4中支持这个元素。

示例

```
<p>This is one line of a paragraph.
<spacer type="vertical" size="36">
And this completes the paragraph with a three-line gap from the first line.</p>
```

元素专有属性 align、height、size、type、width

元素专有事件处理属性 无。

align

IE n/a NN 3 Moz n/a Saf n/a Op n/a HTML n/a

```
align="对齐常量"
```

可选

本属性决定了在环绕四周的上下文中，spacer元素块的对齐方式。请参考本章前述部分中“对齐常量”一节，以了解各种可能的对齐方式。

244

示例 `<spacer type="block" height="90" width="40" align="absmiddle">`

值 不区分大小写的对齐常量。

默认值 bottom

height, width

IE n/a NN 3 Moz n/a Saf n/a Op n/a HTML n/a

```
height="长度值" width="长度值"
```

必选

height和width这两个属性可以控制块状spacer元素在文档中所占据的空间大小。仅当type属性值为block时，此属性才能发挥作用。

示例 `<spacer type="block" height="150" width="250">`

值 正整数或百分比。

默认值 0

size

IE n/a NN 3 Moz n/a Saf n/a Op n/a HTML n/a

```
size="像素值"
```

可选

根据type属性值的不同，此属性决定了所插入的空白空间的像素尺寸。当type值为line时，它表示水平尺

寸，而vertical则对应竖直尺寸。如果type属性的值为block，则会忽略size属性。

示例 `<spacer type="line" size="40">`

值 任意正整数。

默认值 0

type

IE *n/a* NN *3* Moz *n/a* Saf *n/a* Op *n/a* HTML *n/a*

`type="spacer 类型"` 必选

此属性决定了spacer元素将采用3种类型中的哪一种分隔类型。当类型为line时，会在同一行的文本前添加一段空白区域；vertical（或vert）会在两行之间添加空白空间；而block则定义了一种同时向竖直和水平方向扩展的矩形空间。当类型定义为line或vertical时，必须指定size属性，否则就应该为block类型指定height和width属性。

示例 `<spacer type="line" size="40">`

值 4种常量之一：block | line | vertical | vert，不区分大小写。

默认值 line

width

参见height属性。

IE *3* NN *4* Moz *all* Saf *all* Op *7* HTML *4*

`<span>...</span>` HTML 结束标签：必选

span元素为文档中的任何行内内容提供了结构和上下文关系。与那些被赋予了特殊内涵的结构化元素（如p元素）不同，作者们可以依靠span元素的属性和内嵌的内容，自由地为每个span元素都赋予不同的含义。只须将相关的内容放入起始和结束标签之内，每个span元素都可以变成一个通用的容器。

如果须要使用一个样式单规则来控制细小的行内内容的样式，那么使用span元素进行包装是最合适的选择。例如，如果希望区分出一段文字内的部分词汇，那么可以使用span元素将这些词包装起来，并在span中通过其样式单定义所需的字体和文本样式。这个样式单既可以在span内嵌的style属性中定义，也可以通过class或id属性来指定，具体采取哪种方式取决于其余的文档结构。

如果想为块级内容选择一个更为合适的容器，则请使用div元素。

示例

```
<span style="font-size:10pt; text-transform:uppercase">30-day special offer</span>
```

对象模型引用方式 [window.]document.getElementById(*elementID*)

元素专有属性 datafld、dataformatas、datasrc

元素专有事件处理属性 无。

datafld

IE *4* NN *n/a* Moz *n/a* Saf *n/a* Op *n/a* HTML *4*

`datafld="列名称"` 可选

此属性用于IE中的数据绑定，以便将一个远程数据源的列名与span元素的HTML内容联系在一起。数据源的对应列必须是HTML（参见dataformatas部分），span元素中还必须设置datasrc和dataformatas属性。注意，只能在IE 5/Mac操作系统下实现文本文件数据源的绑定。

HTML 4中依然保留此属性，但它已被XHTML 1.0删除。

示例 `<span datasrc="DBSRC3" datafld="quote" dataformatas="HTML">...</span>`

值 区分大小写的标识符。

默认值 无。

对象模型引用方式 `[window.]document.getElementById(elementID).dataFld`

246

dataformatas

IE 4 NN n/a Moz n/a Saf n/a Op n/a HTML |4|

`dataformatas="数据类型"` 可选

本属性用于IE浏览器中的数据绑定。对于来自于数据源的原始资料，浏览器根据此属性值决定将它们视为纯文本亦或是HTML标签。span元素仅接收HTML格式的数据。

HTML 4中依然保留此属性，但它已被XHTML 1.0删除。

示例 `<span datasrc="DBSRC3" datafld="quote" dataformatas="HTML">...</span>`

值 两种可选常量值：`html | text`，不区分大小写。

默认值 `text`

对象模型引用方式 `[window.]document.getElementById(elementID).dataFormatAs`

datasrc

IE 4 NN n/a Moz n/a Saf n/a Op n/a HTML |4|

`datasrc="数据源名称"` 可选

为实现IE数据绑定功能，此属性用于指定页面内object元素的ID值，以便在进行远程数据访问时加载数据源对象。通过datafld属性指定来自于数据源中的绑定内容。注意，只能在IE 5/Mac操作系统下实现文本文件数据源的绑定。

HTML 4中依然保留此属性，但它已被XHTML 1.0删除。

示例 `<span datasrc="DBSRC3" datafld="quote" dataformatas="HTML">...</span>`

值 区分大小写的标识符。

默认值 无。

对象模型引用方式 `[window.]document.getElementById(elementID).dataSrc`

<strike>

IE 3 NN 3 Moz all Saf all Op 7 HTML 3.2

`<strike>...</strike>` HTML结束标签：必选

浏览器会以删除线的形式显示strike元素中的文本内容。此元素与s元素完全相同，采用该元素的主要原因是因为它的名称与其他单字符排版元素的名称更为相似，如b、i和u元素等。无论如何，HTML 4都不再推荐使用s和strike元素，而建议使用样式单属性text-decoration:line-through。另外，strike和s元素也无法通过严格的HTML 4或XHTML DTD的验证。

247

示例

```
<p>If at first you don't succeed, <strike>do it over</strike> try, try again.</p>
```

对象模型引用方式 `[window.]document.getElementById(elementID)`

元素专有属性 无。

元素专有事件处理属性 无。

IE *all* NN *all* Moz *all* Saf *all* Op *7* HTML *all*

`<strong>...</strong>`

HTML 结束标签：必选

在HTML 4推荐标准中，有一类元素统称为**短语元素**（phrase element），strong元素也是其中之一。这些元素为文档的特定部分指明了结构上的含义。作为一种强调文本内容的容器元素，strong所表达的强调意味比em元素更为强烈。em元素会以斜体的形式显示文本内容，而strong则以粗体的形式来显示文本。可以使用任何合适的样式单来替换默认的显示形式。

示例 `<p>Don't delay. <strong>Order today</strong> to get the maximum discount.</p>`

对象模型引用方式 `[window.]document.getElementById(elementID)`

元素专有属性 无。

元素专有事件处理属性 无。

<style>

IE *3* NN *4* Moz *all* Saf *all* Op *7* HTML *4*

`<style>...</style>`

HTML 结束标签：必选

style是一种用于存储样式单规则的容器元素。它只允许在head元素内使用。但是可以在一个head元素中使用多个style元素，请参考media属性。

那些非常古老的浏览器可能还会显示style元素中的内容。为了防止这种现象发生，一贯的经验做法是将样式单放置在HTML注释标记之内。请参见在线参考III以获取更多关于样式单规则的信息。

示例

```
<style type="text/css">
h1 {font-size: 18pt; text-transform: capitalize}
p  {font-size: 12pt}
</style>
```

对象模型引用方式

```
[window.]document.getElementsByTagName("style")[i]
[window.]document.getElementById(elementID)
```

元素专有属性 disabled、edia、ype

元素专有事件处理属性 无。

disabled

IE *4* NN *n/a* Moz *n/a* Saf *n/a* Op *n/a* HTML *n/a*

`disabled`

可选

此属性可以完全禁用整个style元素，使它犹如从未在文档中出现过。从IE 5起，IE/Mac才开始响应此属性。

由于disabled属性是布尔类型，因此如果出现此属性，即意味着它的值为为“true”。当然，也可以通过脚本在事后调整此属性值，请参见第2章中的style对象。

示例 `<style type="text/css" disabled>...</style>`

值 一旦此属性出现，即表示禁用该元素。

默认值 false

对象模型引用方式

```
[window.]document.getElementsByTagName("style")[i].disabled
[window.]document.getElementById(elementID).disabled
```

media

IE *4* NN *n/a* Moz *all* Saf *all* Op *7* HTML *4*

`media="描述符列表"`

可选

本属性用于设置元素内容所需要的输出设备。只有当浏览器能够将文档内容进行修正，以便适合不同的设备

（如手持电脑、文本-语音转换器或模糊电视设备等）时，media属性才会大有作用。目前，主要将其作为一种打印输出的替代方式来使用。HTML 4规范为预期设备定义了一系列的常量值，但这个列表尚未完全确定，将来的浏览器可以为其他的媒体或装置修改输出设备。

示例 `<style type="text/css" media="print">...</style>`

值 区分大小写的对齐常量。在用括号括起的字符串中，可以将多个值组合在一起，彼此之间使用逗号分隔。HTML 4定义了下列值：`all | aura | braille | handheld | print | projection | screen | tty | tv`。而目前的浏览器仅支持3个值：`all | print | screen`。

默认值 `all`

对象模型引用方式

```
[window.]document.getElementsByTagName("style")[i].media
[window.]document.getElementById(elementID).media
```

type

IE 4 NN 4 Moz all Saf all Op 7 HTML 4

`type="MIME类型"` 必选

type属性会告知浏览器使用哪种样式单语法来解释当前元素中定义的样式规则。

示例 `<style type="text/css">...</style>`

值 MIME类型，且不区分大小写。当前所有的主流浏览器均能接受“text/css”这种MIME类型。当使用JavaScript样式单语法时，Navigator 4还可以识别“text/javascript”这种类型。

默认值 `text/css`

对象模型引用方式

```
[window.]document.getElementsByTagName("style")[i].type
[window.]document.getElementById(elementID).type
```

<sub>

IE 3 NN 2 Moz all Saf all Op 7 HTML 3.2

`<sub>...</sub>` HTML结束标签：必选

作为一个排版元素，sub会命令浏览器以下标的形式显示其内容，而字体尺寸将与周围文本内容的字体大小相同。浏览器则倾向于使用较小的字体尺寸来显示下标内容。

示例

```
<p>A hydronium ion (H<sub>3</sub>O) has one more hydrogen atom than regular water.</p>
```

对象模型引用方式 `[window.]document.getElementById(elementID)`

元素专有属性 无。

元素专有事件处理属性 无。

<sup>

IE 3 NN 2 Moz all Saf all Op 7 HTML 3.2

`<sup>...</sup>` HTML结束标签：必选

作为一个排版元素，sup会命令浏览器以上标的形式显示其内容，而字体尺寸将与周围文本内容的字体大小相同。浏览器则倾向于使用较小的字体尺寸来显示这些上标内容。

示例 `<p>This book is published by O'Reilly<sup>&#153;</sup>.</p>`

对象模型引用方式 `[window.]document.getElementById(elementID)`

元素专有属性 无。

元素专有事件处理属性 无。

<table>

IE *all* NN *all* Moz *all* Saf *all* Op *7* HTML *3.2*

`<table>...</table>`

HTML 结束标签：必选

table元素用于容纳表格中其他表示具体内容的附加元素。每个表格都包含一定数量的行列内容。其他与table相关的元素包括：caption、col、colgroup、tbody、td、tfoot、th、thead和tr等。无论表格中包含多少行和列，均可以通过table元素来为整个表格定义一系列的可视属性。与此同时，针对给定的某行、列或单元格，还可以重新指定这些属性。表格中tr和td元素的组成结构完全决定了其行列数。如果须要通过脚本修改表格内容，那么在表格中引入tbody将是一个非常明智的决定。

长期以来，表格不仅用于组织行和列，还可以用于网页内容定位。使用不可见的边框后，就可以将表格的行与列设置为空白空间。但由于CSS 2已经广泛应用于各种浏览器之中，网页设计者们已经很少使用表格来进行排版。表格目前主要用来显示那些纯粹的表格式数据。这类信息将会以列表的形式进行组织并呈现在文档中。

需要注意的是，在某些浏览器中，嵌套表格可能会引起一些问题。在处理复杂表格时，如果表格的嵌套层数超过三层，那么Navigator 4将难于实现样式单的继承，而且显示效果也很差。在IE 5/Mac中，如果通过脚本创建或修改与表格相关的元素，有时会令人费解地突然增大表格单元的尺寸。因此，表格结构越简单，其跨浏览器的可靠性越好。如果使用可视化编辑工具反复地编辑表格框架，那么可能会在源代码中留下一些隐藏的复杂性，例如许多空表格单元。因此为了防止出现这些隐藏内容，可以为表格暂时设置细边框，以便能够看清额外的行列结构。某些网页开发浏览器还可以让用户看到表格在页面中所处的位置。

示例

```
<table cols="3">
<thead>
<tr>
<th>Time</th><th>Event</th><th>Location</th>
</tr>
</thead>
<tbody>
<tr>
<td>7:30am-5:00pm</td><td>Registration Open</td><td>Main Lobby</td>
</tr>
<tr>
<td>9:00am-12:00pm</td><td>Keynote Speakers</td><td>Cypress Room</td>
</tr>
</tbody>
</table>
```

对象模型引用方式 [window.]document.getElementById(*elementID*)

元素专有属性 align、background、bgcolor、border、bordercolor、bordercolordark、bordercolorlight、cellpadding、cellspacing、cols、datapagesize、datasrc、frame、height、hspace、layout、rules、summary、vspace、width

元素专有事件处理属性

处理程序	NN	IE	HTML	处理程序	NN	IE	HTML
onafterupdate	n/a	4	n/a	onrowenter	n/a	4	n/a
onbeforeupdate	n/a	4	n/a	onrowexit	n/a	4	n/a

align

IE *all* NN *all* Moz *all* Saf *all* Op *7* HTML *3.2*

`align="对齐位置"`

可选

此属性决定了表格在其临近的最外层容器中的对齐方式，这些容器往往是文档的body或html元素。HTML 4不再赞成使用此属性，而推荐使用样式单。

示例 `<table align="center">...</table>`

值 对齐常量：center | left | right。

默认值 left

252

对象模型引用方式 `[window.]document.getElementById(elementID).align`

background

IE 3 NN 4 Moz all Saf all Op 7 HTML n/a

`background="URL"` 可选

此属性可指定一个图像文件作为表格背景。与浏览器载入的其他普通图像不同的是，背景图会以未经缩放的原始大小进行加载，然后贴附并充满整个可用的表格空间。小图片虽然下载迅速，但总是会很明显地在背景中重复显示。也可以使用GIF动画作为背景图，但这实在是一件令读者抓狂的做法。在选择背景图时，一定要选择相对柔和的图片，以便使主体内容更加引人注目。如果一定要使用背景图，那么请选用非常淡雅的图片。

值得注意的是，Navigator 4在处理这个属性时，表现很奇特。它会将一个无背景图的表格放置在另一个指定了background属性的表格之中。

示例 `<table background="watermark.jpg">...</table>`

值 任何有效的图像文件URL地址，既可使用相对URL也可以使用绝对URL地址。

默认值 无（此时表格为透明状态）。

对象模型引用方式 `[window.]document.getElementById(elementID).background`

bgcolor

IE 3 NN 3 Moz all Saf all Op 7 HTML 3.2

`bgcolor="颜色三元组或颜色名"` 可选

此属性为整个表格设置填充色，此颜色位于文本和其他内容之下。如果同时使用bgcolor和background属性，那么背景色会透过背景图的任何透明部分。HTML 4并不赞成使用此属性，而建议使用CSS的backgroud-color属性。

示例 `<table bgcolor="tan">...</table>`

值 一个十六进制的三元组或明语颜色名称。不设置属性值就默认为黑色（#000000，black）。目前可用的明语颜色名称请参见附录A。

默认值 随着浏览器、浏览器版本和操作系统的不同而改变。

对象模型引用方式 `[window.]document.getElementById(elementID).bgColor`

253

border

IE all NN all Moz all Saf all Op 7 HTML 3.2

`border="像素值"` 可选

此属性决定了table元素四周边框的宽度（单位为像素）。一旦为border属性设置了属性值，那么在默认情况下，浏览器在显示表格时均会为表格内的每个单元格加上一定的边框。在显示table元素的边框时，其cellspacing属性决定了分隔单元格的内部边框的宽度。

如果仅仅使用了border属性且未设置具体的属性值，那么除非通过其他属性来改变此宽度值，否则浏览器会使用默认尺寸显示整个表格的边框宽度。

浏览器往往会以突出于单元格内平面内容的3D形式显示这些边框。还有其他很多属性也可以影响边框的外观，例如：bordercolor、bordercolordark、bordercolorlight、frame和rules。需要注意的是，表格边框的显示样式与使用样式单规则定义的边框有所不同。因此，最好使用table元素中的专用属性来控制边框的外观样式。

示例 `<table border="1">...</table>`

值 正整数。

默认值 0

对象模型引用方式 `[window.]document.getElementById(elementID).border`

bordercolor

IE *3* NN *4* Moz *all* Saf *all* Op *n/a* HTML *n/a*

`bordercolor="颜色三元组或颜色名"` 可选

此属性可以设置环绕在单元格和整个表格四周的边框的颜色。只有将border属性设置为非零值，才能显示出彩色的边框效果。如果在页面背景或默认颜色四周仔细地勾勒出明暗线条，还可以创造出3D效果的边框，请参见图1-4。标准颜色通常是逐渐变暗的灰色或白色，而具体的效果取决于浏览器。

在IE和其他支持此属性的浏览器中为表格边框指定颜色后，会有一些不同的效果。IE和Safari会将颜色应用于组成边框的所有线条之上。而这也会使边框丧失原有的3D效果，在IE中，请使用bordercolordark和bordercolorlight属性为边框设置颜色并保存3D效果。 而在Navigator和Mozilla中，针对组成边框的两种阴影，只会在较暗的阴影上使用着色。与此同时，浏览器会自动调整某些线条的暗度，以便加强边框的3D效果。

示例 `<table bordercolor="green" border="2">...</table>`

值 一个十六进制的三元组或明语颜色名称。不设置属性值就默认为黑色（#000000，black）。目前可用的明语颜色名称请参见附录A。

默认值 由浏览器和操作系统共同决定。

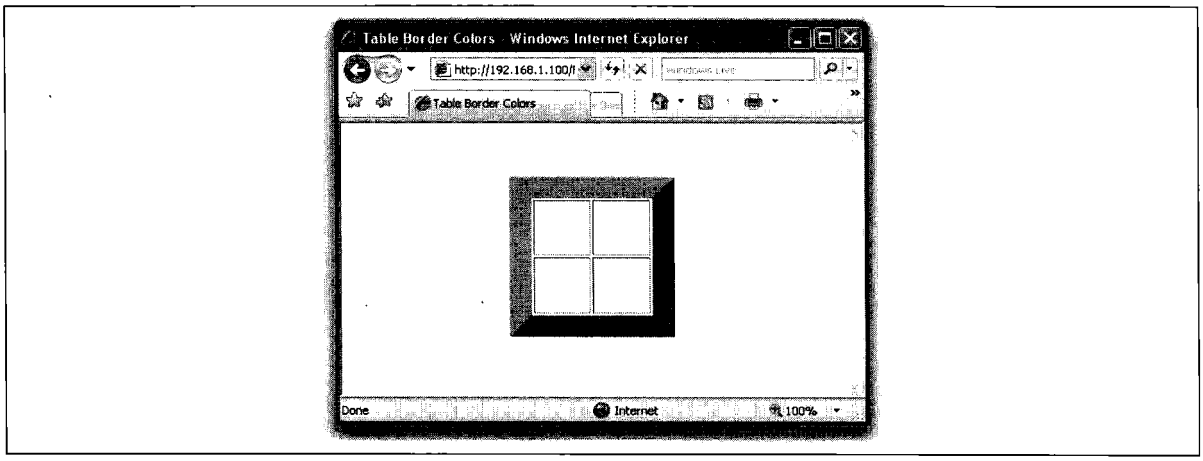


图 1-4：表格边框颜色的组成

对象模型引用方式 `[window.]document.getElementById(elementID).borderColor`

bordercolordark, bordercolorlight

IE *3* NN *n/a* Moz *n/a* Saf *n/a* Op *n/a* HTML *n/a*

`bordercolordark="颜色三元组或颜色名" bordercolorlight="颜色三元组或颜色名"` 可选

如果在页面背景或默认颜色四周仔细地勾勒出明暗线条，就可以在IE中创造出3D效果的表格边框（请参见图1-4）。通过为bordercolordark和bordercolorlight属性设置具体的颜色值，还可以独立地控制边框上明暗两种线条的颜色。此时，必须将border属性设置为非零值才能显示出彩色的边框效果。

此时，应该为这对属性设置互补的颜色。而对于具体的颜色设置并没有强制性要求，并不一定要为

bordercolordark属性设置暗色。这两个属性仅仅是为了使得边框线条轮廓分明，对它们进行控制就可以改变一些边框线条的颜色。

为了达到与支持表格边框颜色的浏览器（如IE）一样的显示效果，必须确定对应的互补颜色，使得其他浏览器通过设置bordercolor属性就可以获得同样的3D效果。然后，为bordercolordark和bordercolorlight属性设置对应的互补色。有时须要在同一个<table>标签中同时使用这3个颜色相关的属性。

示例

```
<table bordercolordark="darkred" bordercolorlight="salmon" border="3">...</table>
```

255

值 一个十六进制的三元组或明语颜色名称。不设置属性值就默认为黑色（#000000，black）。目前可用的明语颜色名称请参见附录A。

默认值 由操作系统决定。

对象模型引用方式

```
[window.]document.getElementById(elementID).borderColorDark
[window.]document.getElementById(elementID).borderColorLight
```

cellpadding

IE 3 NN all Moz all Saf all Op 7 HTML 3.2

```
cellpadding="长度值"
```

可选

此属性决定了表格单元格的边框与内容之间的空白空间大小。值得注意的是，此属性适用于单元格内的空白空间。如果不设置此属性，那么大多数浏览器会紧贴着单元格的左边缘开始显示文本内容。如果须要显示表格边框，那么应在边框边缘与内容之间增加一定的空间，从而使得这些内容更便于阅读。而在某些设计实例中，可能需要较大的空白空间。当不显示表格边框时，往往不易察觉此属性的存在，但依然可以通过它来调整单元格之间的间隔。

示例 `<table border="2" cellpadding="3">...</table>`

值 任意一个单位为像素的长度值，或可用空间的百分比。此处的百分比值是基于单元格在水平和竖直方向的全部可用空间。例如，当属性值为“10%”时，意味着左右两侧的间隔分别为单元格宽度的5%，而上下两端的间隔分别为单元格高度的5%。

默认值 0

对象模型引用方式 `[window.]document.getElementById(elementID).cellPadding`

cellspacing

IE 3 NN all Moz all Saf all Op 7 HTML 3.2

```
cellspacing="长度值"
```

可选

此属性决定了表格单元格的外部边框之间的空白空间大小。如果将table元素的border属性设置为任意的正整数值，那么设置cellspacing属性会定义单元格之间的边框的厚度。即使边框不可见，单元格间距也有利于增强表格的可读性。

示例 `<table border="2" cellspacing="10">...</table>`

值 任意正整数。

默认值 0（无表格边框），2（使用表格边框）。

对象模型引用方式 `[window.]document.getElementById(elementID).cellSpacing`

cols

256

IE 3 NN 4 Moz all Saf all Op n/a HTML n/a

```
cols="列数量"
```

可选

此属性决定了表格的列数。HTML规范从未采用过此属性。而在HTML 4中，由colgroup和col元素代替此属

性的对应功能。但与此同时，以前和现在的浏览器均能识别此属性，并辅助浏览器为表格显示做出准备。如果不使用此属性，那么浏览器只能依靠解析tr和td元素来决定表格的分割形式。

示例 `<table cols="4">...</table>`

值 任意正整数。

默认值 无。

对象模型引用方式 `[window.]document.getElementById(elementID).cols`

datapagesize

IE *4* NN *n/a* Moz *n/a* Saf *n/a* Op *n/a* HTML *n/a*

`datapagesize="记录数量"`

可选

本属性用于IE浏览器中的数据绑定。它可以建议浏览器显示一定的表格行来适应此属性所提供的数据源记录数。一种常见的用法是，使用一个表格单元格来显示一个input元素，而其datafld属性已绑定到数据源的一个特定列中（datasrc在table元素中指定）。如果datapagesize属性设置为5，那么浏览器必须显示表格中的5行。需要注意的时，此时对应行在HTML代码中只用指定一次。

示例

```
<table datasrc="DBSRC3" datapagesize="5">
<tr>
  <td><input type="text" datafld="stockNum"></td>
  <td><input type="text" datafld="qtyOnHand"></td>
</tr>
</table>
```

值 任意正整数。

默认值 无。

对象模型引用方式 `[window.]document.getElementById(elementID).dataPageSize`

datasrc

IE *4* NN *n/a* Moz *n/a* Saf *n/a* Op *n/a* HTML *n/a*

`datasrc="数据源名称"`

可选

为实现IE数据绑定功能，此属性用于指定页面内object元素的ID值，以便在进行远程数据访问时加载数据源对象。通过datafld属性指定来自于数据源中的绑定内容。

一旦在表格中设置了datapagesize属性，那么就可以在表格之中显示一系列连续的数据记录。注意，只能在IE 5/Mac操作系统下实现文本文件数据源的绑定。

示例 `<table datasrc="DBSRC3" datapagesize="5">...</table>`

值 区分大小写的标识符。

默认值 无。

对象模型引用方式 `[window.]document.getElementById(elementID).dataSrc`

frame

IE *3* NN *n/a* Moz *all* Saf *n/a* Op *7* HTML *4*

`frame="框架常量"`

可选

此属性决定了须要显示表格哪一侧的外部边框。但它并不会影响单元格之间的内部边框。如果仅使用了border属性，但未对frame属性设置任何值，那么其属性值等效于“border”。当然也可以使用CSS中边框相关的属性来取代这些设定。

示例 `<table border="3" frame="void">...</table>`

值

可使用以下任意一个框架常量，它们均不区分大小写。

`above`

仅显示表格顶部的边框。

`below`

仅显示表格底部的边框。

`border`

显示四周的所有边框（IE和NN中的默认值）。

`box`

与`border`相同，显示四周的所有边框。

`hsides`

仅显示表格顶部与底部的边框。

`lhs`

仅显示表格左侧的边框。

`rhs`

仅显示表格由侧的边框。

`void`

隐藏所有的边框（HTML 4中的默认值）。

`vsides`

仅显示表格左侧和右侧的边框。

默认值 Mozilla：`void`（当`border`为0时）；`border`（当`border`为其他任意值时）。Internet Explorer：`border`。

对象模型引用方式 `[window.]document.getElementById(elementID).frame`

height, width

IE *3* NN *all* Moz *all* Saf *all* Op *7 (width)* HTML *3.2 (width)*

`height="长度值" width="长度值"` 可选

这两个属性决定了表格的矩形尺寸，该尺寸可能与浏览器计算出的默认尺寸不同。如果其值小于显示表格单元格内容所需要的最小空间值，那么即使出现文本自动换行，浏览器依然会取代属性设置以便保证显示所有的内容。当然，也可以通过这两个属性扩展表格尺寸，使之大于浏览器计算得到的尺寸值。表格单元格内出现的额外的空白空间会体现不同设置之间的差异。如果仅设置了其中的一个属性值，那么浏览器会完成必要的计算，以便沿着另一轴向自动调整表格的尺寸。

须要注意的是，`height`属性并未出现在HTML规范之中。HTML 4假设，无论是使用默认设置还是使用属性指定的宽度值，通过浏览器计算所得的高度值都能够为显示单元格内容呈现最佳的效果。但由于不同操作系统中的不同浏览器会使用变化多样的字体尺寸显示文本内容，因此实际应用中很少通过浏览器计算表格的高度。

示例 `<table width="80%">...</table>`

值 任意一个单位为像素的长度值，或者可用空间的百分比。

默认值 由表格内容决定，但宽度值不要超过其临近的最外层容器宽度的100%。除Mozilla之外，所有的浏览器均能接受超过100%的百分比值，但这会导致在表格容器中出现滚动条，并打乱排版设计的版面完整性。

对象模型引用方式

```
[window.]document.getElementById(elementID).height
[window.]document.getElementById(elementID).width
```

hspace, vspace

IE *n/a* NN *n/a* Moz *all* Saf *all* Op *n/a* HTML *n/a*

```
hspace="像素值" vspace="像素值"
```

可选

通过这两个属性，可以在整个表格边缘的外部加入透明填充空间。尽管如此，还是请使用CSS中与填充相关的属性来代替它们。需要注意的是，Mozilla仅在“quirks”模式下才会响应table元素的这些属性，请参见DOCTYPE元素。

示例 `<table hspace="20" vspace="40">...</table>`

值 整数像素值。

默认值 0

layout

IE *n/a* NN *n/a* Moz *all* Saf *n/a* Op *n/a* HTML *n/a*

```
layout="布局类型"
```

可选

此属性可以控制表格布局显示算法。“fixed”属性值会命令浏览器根据明确的高度和宽度设定来排列表格及其单元格，而不必对内容尺寸进行最小化处理。此属性类似于CSS的“table-layout”属性。

示例 `<table layout="fixed" width="500">...</table>`

值 常量：auto | fixed。

默认值 auto

rules

IE *3* NN *n/a* Moz *1.0.1* Saf *n/a* Op *7* HTML *4*

```
rules="线条常量"
```

可选

如果存在单元格之间的内部边框，那么通过此属性可以定义它们的显示位置。使用此属性后，不仅可以通过绘制边框使得表格以阵列的形式显示各单元格，还可以通过边框仅仅分隔出一些行、列或任意其他类型的单元格分组（如thead、tbody、tfoot、colgroup或col）。为了显示单元格的边框，必须使用border属性，其属性值既可以是一个布尔变量也可以是一个特定的边框尺寸。但IE 5/Mac已删除了rules属性，这使得内部单元格之间均会存在一定的间隔。另外，在支持边框颜色的浏览器中，设定的边框颜色也同样会作用于这些线条之上。

示例 `<table border="3" rules="groups">...</table>`

值

可使用以下任意一个线条常量，它们均不区分大小写。

`all`

显示每个单元格四周的边框。

`cols`

仅显示表格列之间的边框。

`groups`

显示单元格分组之间的边框，这些分组可通过thead、tfoot、tbody、colgroup或col等元素定义。

`none`

隐藏所有内部边框。

`rows`

仅显示表格行之间的边框。

默认值 none（当border为0时）；all（当border为其他任意值时）。

对象模型引用方式 [window.]document.getElementById(*elementID*).rules

summary

IE *6* NN *n/a* Moz *all* Saf *all* Op *7* HTML *4*

summary="文本" 可选

此属性用于存储表格的文字说明，它可以包含使用此表格的目的及表格数据的组成方式，以便非可视化的浏览器使用，当然，也可以包含其他内容。，通过Mozilla中的上下文菜单中可以设置这个元素的特性，并为此属性指定的URL显示一个包含链接的小窗口。目前主流的浏览器尚未为该属性提供更多的功能。

示例 <table summary="Order form for entry of up to five products.">...</table>

值 使用引号括起来的任意字符串。

默认值 无。

vspace

参见hspace属性。

width

参见height属性。

<tbody>

IE *3* NN *n/a* Moz *all* Saf *all* Op *7* HTML *4*

<tbody>...</tbody> 可选

作为一种容器，tbody元素可以容纳一行或多行表格的单元格。在一个table元素内，可以定义多个tbody元素。使用这个元素，可以在表格内定义一些结构分段，使得它们可以拥有自己的样式或边框（关于边框样式请参见rules属性）。类似于作用于列的colgroup元素，tbody是一种面向行的元素。另外，tfoot和thead元素也可用于表格行分组，它们都不会与tbody元素发生重叠。

示例

```
<table cols="3">
<thead>
<tr>
<th>Time</th><th>Event</th><th>Location</th>
</tr>
</thead>
<tbody>
<tr>
<td>7:30am-5:00pm</td><td>Registration Open</td><td>Main Lobby</td>
</tr>
<tr>
<td>9:00am12:00pm</td><td>Keynote Speakers</td><td>Cypress Room</td>
</tr>
</tbody>
</table>
```

对象模型引用方式 [window.]document.getElementById(*elementID*)

元素专有属性 align、bgcolor、ch、char、charoff、choff、valign

元素专有事件处理属性 无。

align

IE 4 NN n/a Moz all Saf all Op 7 HTML 4

```
align="对齐常量"
```

可选

在tbody元素控制的行中，此属性确定它们的水平对齐方式。

示例 `<tbody align="center">`

值

HTML 4与各种浏览器分别实现了不同的属性值集合。

值	IE/Windows	其他	HTML 4	值	IE/Windows	其他	HTML 4
center	·	·	·	left	·	·	·
char	–	–	·	right	·	·	·
justify	–	·	·				

center、left和right这3个值不解自明。如果使用justify这种对齐方式，将会分散排列多行内容，使它们与左右边缘均对齐。而针对char这种对齐方式，在使用时必须为char属性指定一个字符，并将会以这个字符为中心进行对齐。在HTML 4规范的示例中，未包含字符的内容将会与同一列中其他行的文本内容右对齐。

需要牢记于心的是，align属性会作用于tbody元素内所有行中的每个单元格。如果希望调整起始行的样式，可以使用单独的align属性或text-align样式单属性为thead元素或th元素设定样式。

默认值 left

对象模型引用方式 [window.]document.getElementById(*elementID*).align

bgcolor

IE 4 NN n/a Moz all Saf all Op n/a HTML n/a

```
bgcolor="颜色三元组或颜色名"
```

可选

此属性为tbody元素包含的所有单元格设置填充色，此颜色将位于文本和其他内容之下。

示例 `<tbody bgcolor="tan">`

值 一个十六进制的三元组或明语颜色名称。不设置属性值就默认为黑色（#000000，black）。目前可用的明语颜色名称请参见附录A。

默认值 随着浏览器、浏览器版本和操作系统的不同而改变。

对象模型引用方式 [window.]document.getElementById(*elementID*).bgColor

char

IE n/a NN n/a Moz n/a Saf n/a Op n/a HTML 4

```
char="某字符"
```

可选

char属性用于指定一个字符，tbody元素中某一单元格内的文本将会以该字符作为对齐的参考点。只有当align属性设置为char时，此属性才有价值。而微软则提出了一个ch属性，它可以和标准的char属性相对应。可惜的是，浏览器无论如何都不会响应char或ch属性。

示例 `<tbody align="char" char=".">`

值 任意一个单独的字符。

默认值 无。

charoff

IE n/a NN n/a Moz n/a Saf n/a Op n/a HTML 4

```
charoff="长度值"
```

可选

charoff属性用于指定一个特定的偏移点，char属性所指定的字符将在单元格内的该位置上出现。提供此属

性主要是为了防止浏览器的默认定位不符合表格的设计目标。而微软则提出了一个choff属性，它可以和标准的charoff属性相对应。可惜的是，浏览器无论如何都不会响应char或ch属性。

示例 `<tbody align="char" char="." charoff="80%">`

值 任意一个单位为像素的长度值或单元格空间的百分比。

默认值 无。

choff

参见charoff。

valign

IE 4 NN n/a Moz all Saf all Op 7 HTML 4

`valign="对齐常量"` 可选

在tbody元素控制的列单元格中，此属性确定其竖直对齐的方式。可以为列中的任意一个特定单元格更换竖直对齐方式。

示例 `<tbody valign="bottom">`

值 目前有4个可用的常量值：top、middle、bottom和baseline。当属性值设定为top和bottom时，其内容将会齐平或紧贴表格单元格的顶部或底部显示。设置为middle（默认值）时，其内容会恰好悬浮在单元格的竖直中心处。如果单元格内的内容可能会依照通常的窗口宽度（假设为表格宽度）分割为多行，那么最好将valign属性设置为baseline。这样就可以保证单元格内文本的第一行的字符基线能够与行内其他单元格的文本对齐，而这是最符合审美观的编排方式。

默认值 middle

对象模型引用方式 `[window.]document.getElementById(elementID).vAlign`

<td>

IE all NN all Moz all Saf all Op 7 HTML 3.2

`<td>...</td>` HTML结束标签：可选

作为一个容器元素，td可以容纳那些显示在一个单元格内的相关内容。每个单元格实质上就是一个行与列的交点。与td元素相关的其他元素包括caption、col、colgroup、table、tbody、tfoot、th、thead和tr等。除了可以为单元格中的内容提供一个包装环境，通过td元素还可以为单个的单元格定义一系列的可视属性，这些属性往往会替代表格中一些嵌套元素的类似属性。

HTML 4规范引入了4种属性：abbr、axis、headers和scope以应用于非可视化的浏览器，使得这类浏览器能通过文本-语音转换技术将HTML页面中的内容以“语言呈现”的方式表达出来。尽管为了保证知识结构的完整性，这些属性已经简要地罗列在这里，但它们在非可视化浏览器中的具体应用细节实在是大大地超出了本书所涉及的动态HTML这一主体内容。因此，请参考HTML 4推荐标准以获取更多相关细节信息。

示例

```
<table cols="3">
<thead>
<tr>
<th>Time</th><th>Event</th><th>Location</th>
</tr>
</thead>
<tbody>
<tr>
<td>7:30am-5:00pm</td><td>Registration Open</td><td>Main Lobby</td>
</tr>
```

```
<tr>
<td>9:00am12:00pm</td><td>Keynote Speakers</td><td>Cypress Room</td>
</tr>
</tbody>
</table>
```

对象模型引用方式 [window.]document.getElementById(*elementID*)

元素专有属性 abbr、align、axis、background、bgcolor、bordercolor、bordercolordark、bordercolorlight、ch、char、charoff、choff、colspan、datafld、dir、headers、height、nowrap、rowspan、scope、valign、width

264

元素专有事件处理属性

处理程序	IE	其他	HTML	处理程序	IE	其他	HTML
onafterupdate	4	n/a	n/a	onrowenter	4	n/a	n/a
onbeforeupdate	4	n/a	n/a	onrowexit	4	n/a	n/a

abbr

IE *6* NN *n/a* Moz *all* Saf *all* Op *n/a* HTML *4*

abbr="文本" 可选

此属性可以为单元格的内容提供一段简短的说明文字。这往往是一个很简要的标签说明，使得非可视化浏览器能够将单元格想表达的内容朗读出来。可以Mozilla上下文菜单中设置这个元素的特性，从而为此属性指定的URL显示一个包含链接的小窗口。目前主流的浏览器尚未为该属性提供更多的功能。

示例 `<td abbr="Main Event">Keynote Speakers</td>`

值 任何使用引号括起的字符串。

默认值 无。

对象模型引用方式 [window.]document.getElementById(*elementID*).abbr

align

IE *all* NN *all* Moz *all* Saf *all* Op *7* HTML *3.2*

align="对齐常量" 可选

在td元素控制的单元格中，此属性确定对应内容的水平对齐方式。

示例 `<td align="center">`

值

HTML 4与各种浏览器各自实现了不同的属性值集合。

值	IE/Windows 与 NN 4	其他	HTML 4	值	IE/Windows 与 NN 4	其他	HTML 4
center	·	·	·	left	·	·	·
char	–	–	·	right	·	·	·
justify	–	·	·				

265

center、left和right这3个值不解自明。如果使用justify这种对齐方式，将会分散排列多行内容，使它们与左右边缘均对齐。而针对char这种对齐方式，在使用时必须为char属性指定一个字符，并将会以这个字符为中心进行对齐。在HTML 4规范的示例中，未包含字符的内容将会与同一列中其他行的文本内容右对齐。

默认值 left

对象模型引用方式 [window.]document.getElementById(*elementID*).align

axis

IE *6* NN *n/a* Moz *all* Saf *all* Op *n/a* HTML *4*

axis="文本" 可选

此属性可以为单元格的分类提供一段简短的说明文字。这往往是一个很简要的标签说明，使得非可视化浏览

<td>

器能够将单元格想表述的内容朗读出来。Mozilla中，通过在上下文菜单中设置这个元素的特性，可以为此属性指定的URL显示一个包含链接的小窗口。目前主流的浏览器尚未为该属性提供更多的功能。

示例 `<td axis="event">Keynote Speakers</td>`

值 任何使用引号括起的字符串。

默认值 无。

对象模型引用方式 `[window.]document.getElementById(elementID).axis`

background

IE *3* NN *4* Moz *all* Saf *all* Op *7* HTML *n/a*

`background="URL"` 可选

使用此属性，可指定一个图像文件作为单元格背景。与浏览器载入的其他普通图像不同的是，背景图会以未经缩放的原始大小进行加载，然后贴附并充满整个可用的单元格空间。小图片虽然下载迅速，但总是会很明显地在背景中重复显示。然而，值得注意的是，Navigator 4需要图片至少大于16像素×16像素。也可以使用GIF动画作为背景图，但这实在是一件令读者抓狂的做法。在选择背景图时，一定要选择相对柔和的图片，以便使主体内容更加引人注目。如果一定要使用背景图，那么请选用非常淡雅的图片。

示例 `<td background="watermark.jpg">`

值 任何有效的图像文件URL地址，既可使用相对URL也可以使用绝对URL地址。

默认值 无。

对象模型引用方式 `[window.]document.getElementById(elementID).background`

bgcolor

IE *3* NN *3* Moz *all* Saf *all* Op *7* HTML *3.2*

`bgcolor="颜色三元组或颜色名"` 可选

使用此属性可以为td元素定义的单元格设置填充色，该颜色将位于文本和其他内容之下。

示例 `<td bgcolor="yellow">`

值 一个十六进制的三元组或明语颜色名称。不设置属性值就默认为黑色（#000000，black）。目前可用的明语颜色名称请参见附录A。

默认值 随着浏览器、浏览器版本和操作系统的不同而改变。

对象模型引用方式 `[window.]document.getElementById(elementID).bgColor`

bordercolor

IE *3* NN *n/a* Moz *n/a* Saf *n/a* Op *n/a* HTML *n/a*

`bordercolor="颜色三元组或颜色名"` 可选

通过此属性，可以设置环绕在单元格和整个表格四周的边框的颜色。IE会将颜色应用于组成单元格边框的四条线段之上。因此，临近的单元格边框颜色并不会互相冲突。同时，单元格的边框颜色可以异于其所属表格的边框颜色。

示例 `<td bordercolor="green">`

值 一个十六进制的三元组或明语颜色名称。不设置属性值就默认为黑色（#000000，black）。目前可用的明语颜色名称请参见附录A。

默认值 由浏览器和操作系统共同决定。

对象模型引用方式 `[window.]document.getElementById(elementID).borderColor`

bordercolordark, bordercolorlight

IE *3* NN *n/a* Moz *n/a* Saf *n/a* Op *n/a* HTML *n/a*

`bordercolordark="颜色三元组或颜色名", bordercolorlight="颜色三元组或颜色名"` 可选

如果在页面背景或默认颜色四周仔细地勾勒出明暗线条，还可以在IE中创造出3D效果的表格边框（请参见本书

在探讨table元素时给出的图1-4)。通过为bordercolordark(单元格的左边缘与上边缘)和bordercolorlight（单元格的右边缘与下边缘）属性指定属性值，还可以独立地控制明暗两种线条的颜色。

此时，应该为这对属性设置互补的颜色。而对于具体的颜色设置并没有强制性要求，并不一定要为bordercolordark属性设置暗色。这两个属性仅仅是为了使得边框线条轮廓分明，对它们进行控制就可以改变一些边框线条的颜色。

示例 `<td bordercolordark="darkred" bordercolorlight="salmon">`

值 一个十六进制的三元组或明语颜色名称。不设置属性值就默认为黑色（#000000，black）。目前可用的明语颜色名称请参见附录A。

默认值 由操作系统决定。

对象模型引用方式

```
[window.]document.getElementById(elementID).borderColorDark
[window.]document.getElementById(elementID).borderColorLight
```

char

IE *n/a* NN *n/a* Moz *n/a* Saf *n/a* Op *n/a* HTML *n/a*

`char="某字符"` 可选

char属性用于指定一个字符，该字符会作为某一单元格中文本对齐的参考点。只有当align属性设置为char时，此属性才有价值。而微软则提出了一个ch属性，它可以和标准的char属性相对应。可惜的是，浏览器无论如何都不会响应char或ch属性。

示例 `<td align="char" char=".">203.00</td>`

值 任意一个单独的字符。

默认值 无。

charoff

IE *n/a* NN *n/a* Moz *n/a* Saf *n/a* Op *n/a* HTML *4*

`charoff="长度值"` 可选

charoff属性用于指定一个特定的偏移点，char属性所指定的字符将在单元格内的该位置上出现。提供此属性主要是为了防止浏览器的默认定位不符合表格的设计目标。而微软则提出了一个choff属性，它可以和标准的charoff属性相对应。可惜的是，浏览器无论如何都不会响应char或ch属性。

示例 `<td align="char" char="." charoff="80%">`

值 任意一个单位为像素的长度值或单元格空间的百分比。

默认值 无。

choff

参见charoff。

colspan

IE *all* NN *all* Moz *all* Saf *all* Op *7* HTML *3.2*

`colspan="列数"` 可选

此属性决定了表格单元格扩展时所横跨的列数。colspan的数值每增加一列，则对应的表格行中的td元素就应该减少一个。如果将align设置为“center”或“right”，那么对齐时将根据其横跨的列数来计算整个td元素的宽度。除非还为此单元格指定了一个rowspan属性，否则下一行将回复到正常列数。

示例 `<td colspan="2" align="center">`

<td>

值 任意正整数，但通常是等于2或更大的数值。

默认值 1

对象模型引用方式 [window.]document.getElementById(*elementID*).colSpan

datafld

IE *4* NN *n/a* Moz *n/a* Saf *n/a* Op *n/a* HTML *n/a*

`datafld="列名称"` **可选**

此属性用于IE中的数据绑定，以便将一个远程数据源的列名与单元格的内容联系在一起。在使用过程中，table元素内必须设置datasrc属性，并根据需要设置datapagesize属性。注意，只能在IE 5/Mac操作系统下实现文本文件数据源的绑定。

示例

```
<table datasrc="DBSRC3" datapagesize="5">
<tr>
  <td datafld="stockNum"></td>
  <td datafld="qtyOnHand"></td>
</tr>
</table>
```

值 区分大小写的标识符。

默认值 无。

headers

IE *6* NN *n/a* Moz *all* Saf *all* Op *n/a* HTML *4*

`headers="单元格 ID 列表"` **可选**

通过此属性，可以为当前单元格指明一个或多个作为行/列头的th或td元素。对应的th或td元素的id属性值列表将作为headers元素的属性值，并以空格分隔列表中的不同id。非可视化浏览器可以在朗读单元格内容之前首先读出表头单元格，以辅助收听者获知单元格内容的实质。尽管一些主流浏览器声称将会支持此属性，但到目前为止，尚未出现相关支持。

示例

```
<tr>
<th id="hdr1">Product Number</th>
<th id="hdr2">Description</th>
</tr>
<tr>
<td headers="hdr1">0392</td>
<td headers="hdr2">Round widget</td>
</tr>
```

值 以空格进行分隔的表头单元格ID值列表，请注意区分大小写。

默认值 无。

height, width

IE *all* NN *all* Moz *all* Saf *all* Op *7* HTML *3.2*

`height="长度值" width="长度值"` **可选**

这两个属性决定了单元格的矩形尺寸，该尺寸可能不同于浏览器计算出的默认尺寸。如果其值小于显示表格单元格内容所需要的最小空间值，那么即使出现文本自动换行，浏览器依然会取代属性设置以便保证显示所有的内容。当然，也可以通过这两个属性扩展表格尺寸，使之大于浏览器计算得到的尺寸值。表格单元格内出现的额外的空白空间会体现不同设置之间的差异。如果仅设置了其中的一个属性值，那么浏览器会完成必要的计算，以便沿着另一轴向自动调整表格的尺寸。单元格必须拥有一定的实际内容，否则它可能收缩到最小尺寸。

鉴于表格整齐划一的特性，一旦某行内一个单元格的高度值设定大于其他单元格的高度值，那么对应行将以该值作为行高。类似的，如果一个单元格的宽度值设定大于同一列中的其他单元格，也会使得该列内所有的单元格均采用这个最大值作为宽度值。

HTML 4已不再赞成使用height和width属性，而建议使用height和width样式单属性，但后者无法在Navigator 4中对表格单元格进行控制。

示例 `<td width="80%" height="30">`

值 任意一个单位为像素的长度值，或可用空间的百分比。

默认值 由单元格内实际内容的具体尺寸决定。

对象模型引用方式

```
[window.]document.getElementById(elementID).height
[window.]document.getElementById(elementID).width
```


nowrap

IE *all* NN *all* Moz *all* Saf *all* Op *7* HTML *3.2*

```
nowrap
```

可选

如果出现nowrap属性，那么浏览器会按照需要加宽单元格，以便将单元格内的文本不间断地显示在一整行上。当单元格内内容很多时，如果滥用此属性，那么为了显示所有内容，会在页面上出现一个非常不便于使用的横向滚动条。HTML 4.0已不再赞成使用nowrap属性，而推荐使用样式单属性white-space:nowrap。

示例 `<td nowrap>`

值 如果此属性已出现，就意味着其属性值为true。

默认值 `false`

对象模型引用方式 `[window.]document.getElementById(elementID).noWrap`

rowspan

IE *all* NN *all* Moz *all* Saf *all* Op *7* HTML *3.2*

```
rowspan="行数量"
```

可选

此属性决定了表格单元格扩展时所横跨的列数。rowspan中的数值每增加一个，则需要在该单元格所在行的下一行的同一位置减少一个td元素。

示例 `<td rowspan="2">`

值 任意正整数，但通常是等于2或更大的数值。

默认值 `1`

对象模型引用方式 `[window.]document.getElementById(elementID).rowSpan`

scope

IE *6* NN *n/a* Moz *all* Saf *all* Op *n/a* HTML *4*

```
scope="作用域常量"
```

可选

相对于td元素，此属性在th元素中使用得更多。当前单元格可以作为其他单元格的表头，而通过scope属性则可以设置这些单元格的具体范围。对那些结构十分规范的表格而言，即使不定义表头单元格的id属性，使用scope属性也可以很简单地完成header属性所具有的功能。尽管一些主流浏览器声称将会支持此属性，但到目前为止，尚未出现相关支持。

示例

```
<tr>
<th scope="col">Product Number</th>
<th scope="col">Description</th>
```

```
</tr>
<tr>
<td>0392</td>
<td>Round widget</td>
</tr>
```

值

可使用以下4个作用域常量之一。

`col`

当前单元格内的文本成为列内所有剩余单元格的标题文本。

`colgroup`

当前单元格内的文本成为`colgroup`元素内所有剩余单元格的标题文本。

`row`

当前单元格内的文本成为`tr`元素内所有剩余单元格的标题文本。

`rowgroup`

当前单元格内的文本成为`tbody`元素内所有剩余单元格的标题文本。

默认值 无。

valign

IE *all* NN *all* Moz *all* Saf *all* Op *7* HTML *3.2*

`valign="对齐常量"` 可选

此属性决定了`td`元素内所有内容的水平对齐方式。在对应的单元格内，此属性设定会取代其外部容器（如`tr`或`tbody`）的同名属性设定。

示例 `<td valign="bottom">`

值 目前有4个可用的常量值：`top`、`middle`、`bottom`和`baseline`。当属性值设定为`top`和`bottom`时，其内容将会齐平或紧贴表格单元格的顶部或底部显示。设置为`middle`（默认值）时，其内容会恰好悬浮在单元格的竖直中心处。如果单元格内的内容可能会依照通常的窗口宽度（假设为表格宽度）分割为多行，那么最好将`valign`属性设置为`baseline`。这样就可以保证单元格内文本的第一行的字符基线能够与行内其他单元格的文本对齐，而这是最符合审美观的编排方式。

默认值 `middle`

对象模型引用方式 `[window.]document.getElementById(elementID).vAlign`

width

参见height属性。

<textarea>

IE *all* NN *all* Moz *all* Saf *all* Op *7* HTML *all*

`<textarea>...</textarea>` HTML结束标签：必选

`textarea`是一种多行文本输入控件，它常见于`form`元素之内。与`input`元素这种文本输入控件不同，可以改变`textarea`元素的大小，以便接收多行文本。在当前的浏览器中，此元素还支持换行，而且用户也可以在文本框中输入回车符（在ASCII码中，它由十进制的13和10组成）。如果在提交表单时其中还包含一个`textarea`元素，那么对应的名称/值对也会一并提交，此时值为本文框中的所有内容，`name`属性则可以通过任意方式进行指定。而服务器端的CGI程序必须能够处理文本数据中的回车符。

如果需要在加载`textarea`元素之后显示文本信息，那么可以将这些文本放置在起始和结束标签之间，否则就

不要在这两个标签之间插入任何字符。在textarea元素之前或之后，最好放置一段提示文本。当然，从结构上来说，也可以直接在元素内嵌入一个label元素。

示例

```
<textarea rows="5" cols="60" name="notes">Use this area for extra notes. </textarea>
```

对象模型引用方式

```
[window.]document.formName.elementName
[window.]document.forms[i].elements[j]
[window.]document.getElementById(elementID)
```

元素专有属性 accept、autofocus、cols、datafld、datasrc、disabled、form、inputmode、maxlength、name、readonly、required、rows、wrap

元素专有事件处理属性

处理程序	IE	NN	Opera	其他	HTML
onafterupdate	4	n/a	n/a	n/a	n/a
onbeforeupdate	4	n/a	n/a	n/a	n/a
onchange	3	2	all	all	4
onformchange	n/a	n/a	9	n/a	n/a
onforminput	n/a	n/a	9	n/a	n/a
oninvalid	n/a	n/a	9	n/a	n/a
onscroll	3	n/a	n/a	n/a	4
onselect	3	2	all	all	4

accept

IE *n/a* NN *n/a* Moz *all* Saf *n/a* Op *n/a* HTML *n/a*

```
accept="MIME类型列表"
```

可选

在Web Forms 2.0扩展中，通过此属性可以设定哪种或哪几种MIME类型的数据可以输入到textarea元素之中。除了简单文本之外，如果浏览器还为其他类型的内容提供了输入编辑环境，那么通过设置此属性，textarea元素还可以为这些内容做好准备，并且在提交表单时对它们进行编码。

示例 `<textarea name="newsItem" accept="message/news"></textarea>`

值 MIME类型（内容类型）。如果使用多种类型，请在列表中使用逗号进行分隔。Web Forms 2.0允许使用通配符列出可用的MIME类型范围。

默认值 无。

autofocus

IE *n/a* NN *n/a* Moz *n/a* Saf *n/a* Op *n/a* HTML *n/a*

```
autofocus="自动聚焦"
```

可选

页面加载完成后，这个Web Forms 2.0扩展会将焦点聚集在元素上。注意，每个页面上只能有一个表单控件元素能拥有此属性。

cols

IE *all* NN *all* Moz *all* Saf *all* Op *7* HTML *all*

```
cols="列数量"
```

可选

此属性规定了textarea元素内文本编辑域的宽度。其属性值表示在文本编辑域的宽度内，可显示的单宽度字符的数量。对那些支持样式单字体尺寸的浏览器而言，实际宽度会根据具体的字体设定而发生改变。

示例 `<textarea cols="40">...</textarea>`

值 任意正整数。

默认值 由浏览器和操作系统共同决定。

对象模型引用方式

```
[window.]document.formName.elementName.cols
[window.]document.forms[i].elements[j].cols
[window.]document.getElementById(elementID).cols
```

datafld

IE 4 NN n/a Moz n/a Saf n/a Op n/a HTML n/a

```
datafld="列名称"
```

可选

此属性用于IE中的数据绑定，以便将一个远程数据源的列名同textarea元素的内容联系在一起。此时元素中还必须设定datasrc属性。注意，只能在IE 5/Mac操作系统下实现文本文件数据源的绑定。

示例 `<textarea name="summary" datasrc="DBSRC3" datafld="summary"></textarea>`

值 区分大小写的标识符。

默认值 无。

对象模型引用方式

```
[window.]document.formName.elementName.dataFld
[window.]document.forms[i].elements[j].dataFld
[window.]document.getElementById(elementID).dataFld
```

datasrc

IE 4 NN n/a Moz n/a Saf n/a Op n/a HTML n/a

```
datasrc="数据源名称"
```

可选

为实现IE数据绑定功能，此属性用于指定页面内object元素的ID值，以便在进行远程数据访问时加载数据源对象。另外还需要通过datafld属性指定来自于数据源中的绑定内容。注意，只能在IE 5/Mac操作系统下实现文本文件数据源的绑定。

示例 `<textarea name="summary" datasrc="DBSRC3" datafld="summary"></textarea>`

值 区分大小写的标识符。

默认值 无。

对象模型引用方式

```
[window.]document.formName.elementName.dataSrc
[window.]document.forms[i].elements[j].dataSrc
[window.]document.getElementById(elementID).dataSrc
```

disabled

IE 4 NN n/a Moz all Saf all Op 7 HTML 4

```
disabled
```

可选

如果textarea元素被禁用，那么用户将无法对其进行操作。被禁用的textarea控件将无法接收焦点，进行跳格选择时也无法被激活。同时，表单提交时也不会将已禁用的textarea元素的名称/值对发送至服务器。

由于disabled属性是布尔类型，因此一旦出现此属性就意味着它的值为为“true”。当然，也可以通过脚本在事后调整此属性值，请参见第2章中的textarea对象。

示例 `<textarea disabled></textarea>`

值 一旦此属性出现，即表示禁用该元素。

默认值 false

对象模型引用方式

```
[window.]document.formName.elementName.disabled
[window.]document.forms[i].elements[j].disabled
[window.]document.getElementById(elementID).disabled
```

form

IE *n/a* NN *n/a* Moz *n/a* Saf *n/a* Op *9* HTML *n/a*


```
form="formID [formID] ..."
```

可选

通过这个Web Forms 2.0扩展，无论表单是否嵌入了某控件，都可以将一个单独的表单控件元素与一个或多个表单联系在一起。由于Web Forms 2.0中的textarea元素并没有像form的子元素那样受到限制，因此它可以放在form元素之外的任何地方。通过页面上一个或多个表单元素，可以将form属性同textarea元素联系起来。

示例 `<textarea name="notes" form="orderForm"></textarea>`

值 页面上一个或多个form元素的ID值。此时，请使用空格分隔多个ID值。

默认值 无。

inputmode

IE *n/a* NN *n/a* Moz *n/a* Saf *n/a* Op *n/a* HTML *n/a*

```
inputmode="脚本标记 [修改量标记]"
```

可选

这个Web Forms 2.0 扩展（完全继承自W3C XForms 1.0规范，请参阅*http://www.w3.org/TR/xforms/sliceE.html*）会引导浏览器为某种书面语言显示合适的文本输入用户界面。请查阅W3C XForms 1.0文档，以获取更多详细信息。

示例 `<textarea name="notes_jp" inputmode="hiragana"></textarea>`

值 每种书面语言均伴随着一个可选择的修订标记。这些修订标记通常相当于Unicode脚本，请参见*http://www.unicode.org/unicode/reports/tr24/*。

默认值 无。

name

IE *all* NN *all* Moz *all* Saf *all* Op *7* HTML *all*

```
name="元素标识符"
```

可选

如果textarea元素是表单的一部分，并且其元素值会随表单一起被提交至服务器端，那么就必须使用name属性。新的DOM建议为id属性也指定相同的标识符，以便能够使用统一的脚本来引用元素对象。

示例 `<textarea name="comments" id="comments"></textarea>`

值 区分大小写的标识符。

默认值 无。

对象模型引用方式

```
[window.]document.formName.elementName.name
[window.]document.forms[i].elements[j].name
[window.]document.getElementById(elementID).name
```

readonly

IE *4* NN *n/a* Moz *all* Saf *all* Op *7* HTML *4*


```
readonly
```

可选

当textarea元素中出现readonly属性时，用户就不能在页面上编辑其文本域中的内容。此时只能通过脚本对这些内容进行修改。但此时用跳格选择还可以将焦点聚集在textarea元素之上。而且用户依然可以对这些只读内容进行选择与复制。

示例 `<textarea name="instructions" readonly></textarea>`

值 如果此属性已出现，就意味着其属性值为true。

默认值 false

对象模型引用方式

```
[window.]document.formName.elementName.readOnly
[window.]document.forms[i].elements[j].readOnly
[window.]document.getElementById(elementID).readOnly
```

required

IE *n/a* NN *n/a* Moz *n/a* Saf *n/a* Op *9* HTML *n/a*

```
required="required"
```

可选

这个Web Forms 2.0 扩展表示提交时该textarea属性值是否必须存在。当此值设置为true后，一旦元素未设定值，那么就会为ValidityState对象设置missingValue属性值。

rows

IE *all* NN *all* Moz *all* Saf *all* Op *7* HTML *all*

```
rows="行数量"
```

可选

根据此属性指定的文本行数量，可以设定元素textarea的高度，以便不使用滚动条就能完全显示元素内容。此属性值表示在滚动条出现之前，编辑域内所能显示的单间隔字符行的数量。对那些支持样式单字体尺寸的浏览器而言，实际宽度会根据具体的字体设定而改变。

示例 `<textarea rows="5" cols="40"></textarea>`

值 任意正整数。

默认值 由浏览器和操作系统共同决定。

对象模型引用方式

```
[window.]document.formName.elementName.rows
[window.]document.forms[i].elements[j].rows
[window.]document.getElementById(elementID).rows
```

wrap

IE *4* NN *2* Moz *all* Saf *all* Op *7* HTML *n/a*

```
wrap="换行的形式"
```

可选

wrap属性会通知浏览器是否对元素内的文本进行换行，以及当软换行转换为硬换行时，这些文本是否应该被提交至服务器端。几年以来，HTML规范未对此对象进行任何说明，而主流浏览器对它的处理则一直处于混乱之中。但在最近，主流浏览器逐渐形成了共识，它们都认可以下3种属性值：off、soft和hard。

如果属性值为soft，那么在用户输入时，文本会自动换行，但在表单提交过程中，回车换行符（CRLF）并不会随着文本一并提交。当属性值为hard时，也会出现换行，但此时CRLF会成为textarea值的一部分并被提交至服务器端（Safari除外）。在以前，virtual和physical是soft属性值的一种代名词。如果属性值为off，那么当输入内容超出了编辑域的右边缘时，会在textarea中出现水平方向的滚动条。此时，只有用户按下“回车”键才会在对应的位置出现新的一行。

示例 `<textarea name="comments" wrap="hard"></textarea>`

值 常量：hard、off或soft。

默认值 soft

对象模型引用方式

```
[window.]document.formName.elementName.wrap
[window.]document.forms[i].elements[j].wrap
[window.]document.getElementById(elementID).wrap
```

<tfoot>

IE *3* NN *n/a* Moz *all* Saf *all* Op *7* HTML *4*

```
<tfoot>...</tfoot>
```

HTML结束标签：可选

tfoot是一种具有特殊意义的容器，它可以包含一行或多行表格单元格并显示在表格的底部。通常当用户将页面滚动至底部时，tfoot元素可视为thead元素内容的一种对照。在一个table元素中，tfoot元素的数量不应超过一个，而且在源代码中，tfoot元素的位置应该**先于**同一表格内任意一个tbody元素的位置。与tbody

和thead元素相类似，tfoot元素也是一种行分组元素。需要注意的是，Navigator 4会忽略tfoot元素的标签，因此它会根据源代码的顺序将其内嵌的tr元素按照普通tr元素的方式进行显示。

示例

```
<table cols="3">
<thead>
<tr>
<th>Time</th><th>Event</th><th>Location</th>
</tr>
</thead>
<tfoot>
<tr>
<th>Time</th><th>Event</th><th>Location</th>
</tr>
</tfoot>
<tbody>
<tr>
<td>7:30am-5:00pm</td><td>Registration Open</td><td>Main Lobby</td>
</tr>
<tr>
<td>9:00am-12:00pm</td><td>Keynote Speakers</td><td>Cypress Room</td>
</tr>
</tbody>
</table>
```

对象模型引用方式 [window.]document.getElementById(*elementID*)

元素专有属性 align、bgcolor、ch、char、charoff、choff、valign

元素专有事件处理属性 无。

align

IE *4* NN *n/a* Moz *all* Saf *all* Op *7* HTML *4*

```
align="对齐常量"
```

可选

在tfoot元素控制的行中，此属性确定单元格内容的水平对齐方式。

示例 `<tfoot align="center">`

值

HTML 4与各种浏览器分别实现了不同的属性值集合。

值	IE/Windows	其他	HTML 4	值	IE/Windows	其他	HTML 4
center	·	·	·	left	·	·	·
char	–	–	·	right	·	·	·
justify	–	·	·				

center、left和right这3个值不解自明。如果使用justify这种对齐方式，将会分散排列多行内容，使它们与左右边缘均对齐。而针对char这种对齐方式，在使用时必须为char属性指定一个字符，并将会以这个字符为中心进行对齐。在HTML 4规范的示例中，未包含字符的内容将会与同一列中其他行的文本内容右对齐。

必须牢记的是，align属性将作用于tfoot元素所涉及的每一行中的全部单元格，其中还包括为表格指定的所有th元素。如果希望调整起始行的样式，可以使用单独的align属性或text-align样式单属性为th元素设定样式。

默认值 left

对象模型引用方式 [window.]document.getElementById(*elementID*).align

bgcolor

IE *4* NN *n/a* Moz *all* Saf *all* Op *7* HTML *n/a*

```
bgcolor="颜色三元组或颜色名"
```

可选

此属性为tfoot元素包含的所有单元格设置填充色，此颜色将位于文本和其他内容之下。

示例 `<tfoot bgcolor="tan">`

值 一个十六进制的三元组或明语颜色名称。不设置属性值就默认为黑色（#000000，black）。目前可用的明语颜色名称请参见附录A。

默认值 随着浏览器、浏览器版本和操作系统的不同而改变。

对象模型引用方式 `[window.]document.getElementById(elementID).bgColor`

char

IE *n/a* NN *n/a* Moz *n/a* Saf *n/a* Op *n/a* HTML *4*

`char="某字符"` 可选

char属性用于指定一个字符，tfoot元素中某一单元格内的文本将会以该字符作为对齐的参考点。只有当align属性设置为char时，此属性才有价值。而微软则提出了一个ch属性，它可以和标准的char属性相对应。可惜的是，浏览器无论如何都不会响应char或ch属性。

示例 `<tfoot align="char" char=".">`

值 任意一个单独的字符。

默认值 无。

charoff

IE *n/a* NN *n/a* Moz *n/a* Saf *n/a* Op *n/a* HTML *4*

`charoff="长度值"` 可选

charoff属性用于指定一个特定的偏移点，char属性所指定的字符将在单元格内的该位置上出现。提供此属性主要是为了防止浏览器的默认定位不符合表格的设计目标。而微软则提出了一个choff属性，它可以和标准的charoff属性相对应。可惜的是，浏览器无论如何都不会响应char或ch属性。

示例 `<tfoot align="char" char="." charoff="80%">`

值 任意一个单位为像素的长度值或单元格空间的百分比。

默认值 无。

choff

280

参见charoff。

valign

IE *4* NN *n/a* Moz *all* Saf *all* Op *7* HTML *4*

`valign="对齐常量"` 可选

在tfoot元素控制的列单元格中，此属性确定其竖直对齐的方式。可以为列中的任意一个特定单元格更换竖直对齐方式。

示例 `<tfoot valign="bottom">`

值 目前有4个可用的常量值：top、middle、bottom和baseline。当属性值设定为top和bottom时，其内容将会齐平或紧贴表格单元格的顶部或底部显示。设置为middle（默认值）时，其内容会恰好悬浮在单元格的竖直中心处。如果单元格内的内容可能会依照通常的窗口宽度（假设为表格宽度）分割为多行，那么最好将valign属性设置为baseline。这样就可以保证单元格内文本的第一行的字符基线能够与行内其他单元格的文本对齐，而这是最符合审美观的编排方式。

默认值 middle

对象模型引用方式 `[window.]document.getElementById(elementID).vAlign`

<th>

IE *all* NN *all* Moz *all* Saf *all* Op *7* HTML *3.2*

```
<th>...</th>
```

HTML 结束标签：可选

th元素是一种容器元素，在表格中，会按照标题的形式显示此元素内的所有内容。而大多数浏览器都会以粗体字的形式显示th元素中的内容。每个th单元格实质上就是一个行与列的交点。与th相关的其他元素包括caption、col、colgroup、table、tbody、td、tfoot、thead和tr等。th元素除了可以为单元格中的内容提供一个包装环境之外，通过它还可以为单个的单元格定义一系列的可视属性，这些属性往往会替代表格中一些嵌套元素的类似属性。

HTML 4规范引入了4种属性：abbr、axis、headers和scope，以应用于非可视化的浏览器，使得这类浏览器能通过文本-语音转换技术将HTML页面中的内容以“语言呈现”的方式表达出来。尽管为了保证知识结构的完整性，这些属性已经简要地罗列在这里，但它们在非可视化浏览器中的具体应用细节实在是大大地超出了本书所涉及的动态HTML这一主体内容。因此，请参考HTML 4推荐标准以获取更多相关的细节信息。

示例

```
<table cols="3">
<thead>
<tr>
<th>Time</th><th>Event</th><th>Location</th>
</tr>
</thead>
<tbody>
<tr>
<td>7:30am-5:00pm</td><td>Registration Open</td><td>Main Lobby</td>
</tr>
<tr>
<td>9:00am-12:00pm</td><td>Keynote Speakers</td><td>Cypress Room</td>
</tr>
</tbody>
</table>
```

对象模型引用方式 [window.]document.getElementById(*elementID*)

元素专有属性 abbr、align、axis、background、bgcolor、bordercolor、bordercolordark、bordercolorlight、ch、char、charoff、choff、colspan、datafld、headers、height、nowrap、rowspan、scope、valign、width

元素专有事件处理属性

处理程序	IE	其他	HTML	处理程序	IE	其他	HTML
onafterupdate	4	n/a	n/a	onrowenter	4	n/a	n/a
onbeforeupdate	4	n/a	n/a	onrowexit	4	n/a	n/a

abbr

IE *6* NN *n/a* Moz *all* Saf *all* Op *n/a* HTML *4*

```
abbr="文本"
```

可选

此属性可以为单元格的内容提供一段简短的说明文字。这往往是一个很简要的标签说明，使得非可视化浏览器能够将单元格想表述的内容朗读出来。Mozilla中，通过在上下文菜单中设置这个元素的特性，可以为此属性指定的URL显示一个包含链接的小窗口。目前主流的浏览器尚未为该属性提供更多的功能。

示例 `<th abbr="What">Event</th>`

值 任何使用引号括起的字符串。

默认值 无。

<th>

282 align

IE *all* NN *all* Moz *all* Saf *all* Op *7* HTML *3.2*

```
align="对齐常量"
```

可选

在th元素控制的单元格中，此属性确定其水平对齐的方式。

示例 `<th align="center">`

值

HTML 4与各种的浏览器分别实现了不同的属性值集合。

值	IE/Windows与NN 4	其他	HTML 4
center	·	·	·
char	—	—	·
justify	—	·	·
left	·	·	·
right	·	·	·

center、left和right这3个值不解自明。如果使用justify这种对齐方式，将会分散排列多行内容，使它们与左右边缘均对齐。而针对char这种对齐方式，在使用时必须为char属性指定一个字符，并将会以这个字符为中心进行对齐。在HTML 4规范的示例中，未包含字符的内容将会与同一列中其他行的文本内容右对齐。

默认值 left

对象模型引用方式 [window.]document.getElementById(*elementID*).align

axis

IE *6* NN *n/a* Moz *all* Saf *all* Op *n/a* HTML *4*

```
axis="文本"
```

可选

此属性可以为单元格的所属分类提供一段简短的说明文字。这往往是一个很简要的标签说明，使得非可视化浏览器能够将单元格想表述的内容朗读出来。Mozilla中，通过在上下文菜单中设置这个元素的特性，可以为此属性指定的URL显示一个包含链接的小窗口。目前主流的浏览器尚未为该属性提供更多的功能。

示例 `<th axis="event">Events</th>`

值 任何使用引号括起的字符串。

默认值 无。

对象模型引用方式 [window.]document.getElementById(*elementID*).axis

283 background

IE *3* NN *4* Moz *all* Saf *all* Op *7* HTML *n/a*

```
background="URL"
```

可选

此属性可指定一个图像文件作为单元格背景。与浏览器载入的其他普通图像不同的是，背景图会以未经缩放的原始大小进行加载，然后贴附并充满整个可用的单元格空间。小图片虽然下载迅速，但总是会很明显地在背景中重复显示。然而，值得注意的是，Navigator 4需要图片至少大于16像素×16像素。也可以使用GIF动画作为背景图，但这实在是一件令读者抓狂的做法。在选择背景图时，一定要选择相对柔和的图片，以便使主体内容更加引人注目。如果一定要使用背景图，那么请选用非常淡雅的图片。

示例 `<th background="watermark.jpg">`

值 任何有效的图像文件URL地址，既可使用相对URL也可以使用绝对URL地址。

默认值 无。

对象模型引用方式 [window.]document.getElementById(*elementID*).background

bgcolor

IE 3 NN 3 Moz all Saf all Op 7 HTML 3.2

```
bgcolor="颜色三元组或颜色名"
```

可选

使用此属性可以为th元素定义的单元格设置填充色，此颜色将位于文本和其他内容之下。

示例 `<th bgcolor="yellow">`

值 一个十六进制的三元组或明语颜色名称。不设置属性值就默认为黑色（#000000，black）。目前可用的明语颜色名称请参见附录A。

默认值 随着浏览器、浏览器版本和操作系统的不同而改变。

对象模型引用方式 `[window.]document.getElementById(elementID).bgColor`

bordercolor

IE 3 NN n/a Moz n/a Saf n/a Op n/a HTML n/a

```
bordercolor="颜色三元组或颜色名"
```

可选

通过此属性，可以设置环绕在单元格和整个表格四周的边框颜色。IE会将颜色应用于组成单元格边框的四条线段之上。因此，临近的单元格边框颜色并不会互相冲突。

示例 `<th bordercolor="green">`

值 一个十六进制的三元组或明语颜色名称。不设置属性值就默认为黑色（#000000，black）。目前可用的明语颜色名称请参见附录A。

默认值 由浏览器和操作系统共同决定。

对象模型引用方式 `[window.]document.getElementById(elementID).borderColor`

bordercolordark, bordercolorlight

IE 3 NN n/a Moz n/a Saf n/a Op n/a HTML n/a

```
bordercolordark="colorTripletOrName" , bordercolorlight="colorTripletOrName"
```

可选

如果在页面背景或默认颜色四周仔细地勾勒出明暗线条，还可以在IE中创造出3D效果的表格边框（请参见本书在探讨table元素时给出的图1-4）。通过为bordercolordark（单元格的左边缘与上边缘）和bordercolorlight（单元格的右边缘与下边缘）属性指定属性值，还可以独立地控制明暗两种线条的颜色。

此时，应该为这对属性设置互补的颜色。而对于具体的颜色设置并没有强制性要求，并不一定要为bordercolordark属性设置暗色。这两个属性仅仅是为了使得边框线条轮廓分明，对它们进行控制就可以改变一些边框线条的颜色。

示例 `<th bordercolordark="darkred" bordercolorlight="salmon">`

值 一个十六进制的三元组或明语颜色名称。不设置属性值就默认为黑色（#000000，black）。目前可用的明语颜色名称请参见附录A。

默认值 由操作系统决定。

对象模型引用方式

```
[window.]document.getElementById(elementID).borderColorDark
[window.]document.getElementById(elementID).borderColorLight
```

char

IE n/a NN n/a Moz n/a Saf n/a Op n/a HTML 4

```
char="某字符"
```

可选

char属性用于指定一个字符，该字符会作为某一单元格中文本对齐的参考点。只有当align属性设置为char时，此属性才有价值。而微软则提出了一个ch属性，它可以和标准的char属性相对应。可惜的是，浏览器无论如何都不会响应char或ch属性。

<th>

示例 `<th align="char" char=".">$325.10</th>`

值 任意一个单独的字符。

默认值 无。

charoff

IE *n/a* NN *n/a* Moz *n/a* Saf *n/a* Op *n/a* HTML *n/a* cn/a

285

`charoff="长度值"`

可选

charoff属性用于指定一个特定的偏移点，char属性所指定的字符将在单元格内的该位置上出现。提供此属性主要是为了防止浏览器的默认定位不符合表格的设计目标。而微软则提出了一个choff属性，它可以和标准的charoff属性相对应。可惜的是，浏览器无论如何都不会响应char或ch属性。

示例 `<th align="char" char="." charoff="80%">`

值 任意一个单位为像素的长度值或单元格空间的百分比。

默认值 无。

choff

参见charoff。

colspan

IE *all* NN *all* Moz *all* Saf *all* Op *7* HTML *3.2*

`colspan="列数"`

可选

此属性决定了表格单元格扩展时所横跨的列数。colspan的数值每增加一列，则对应的表格行中的th或td元素就应该减少一个。当align设置为center或right时，则对应的单元格将根据其横跨的列数来计算整个th元素的宽度以便进行对齐。除非还为此单元格指定了一个rowspan属性，否则下一行将回复到正常列数。

示例 `<th colspan="2" align="right">`

值 任意正整数，但通常取大于或等于2的数值。

默认值 1

对象模型引用方式 `[window.]document.getElementById(elementID).colSpan`

datafld

IE *4* NN *n/a* Moz *n/a* Saf *n/a* Op *n/a* HTML *n/a*

`datafld="列名称"`

可选

此属性用于IE中的数据绑定，以便将一个远程数据源的列名同表头单元格的内容联系在一起。在使用过程中，table元素内必须设置datasrc属性，并根据需要设置datapagesize属性。注意，只能在IE 5/Mac操作系统下实现文本文件数据源的绑定。

示例

```
<table datasrc="DBSRC3" datapagesize="5">
<tr>
  <th datafld="stockNum"></th>
  <th datafld="qtyOnHand"></th>
</tr>
</table>
```

286

值 区分大小写的标识符。

默认值 无。

headers

IE 6 NN n/a Moz all Saf all Op n/a HTML 4

```
headers="单元格 ID 列表"
```

可选

通过此属性，可以为当前单元格指明一个或多个作为行/列头的th或td元素。对应的th或td元素的id属性值列表将作为headers元素的属性值，并以空格分隔列表中的不同id。非可视化浏览器可以在朗读单元格内容之前首先读出表头单元格，以辅助收听者获知单元格内容的实质。尽管一些主流浏览器声称将会支持此属性，但到目前为止，尚未出现相关支持。

示例

```
<tr>
<th id="hdr1">Product Number</th>
<th id="hdr2">Description</th>
</tr>
<tr>
<th headers="hdr1">0392</th>
<th headers="hdr2">Round widget</th>
</tr>
```

值 以空格进行分隔的表头单元格ID值列表，请注意区分大小写。

默认值 无。

height, width

IE all NN all Moz all Saf all Op 7 HTML 3.2

```
height="长度值" width="长度值"
```

可选

这两个属性决定了单元格的矩形尺寸，该尺寸可能与浏览器计算出的默认尺寸不同。如果其值小于显示表格单元格内容所需要的最小空间值，那么即使出现文本自动换行，浏览器依然会取代属性设置以便保证显示所有的内容。当然，也可以通过这两个属性扩展表格尺寸，使之大于浏览器计算得到的尺寸值。表格单元格内出现的额外的空白空间会体现不同设置之间的差异。如果仅设置了其中的一个属性值，那么浏览器会完成必要的计算，以便沿着另一轴向自动调整表格的尺寸。

鉴于表格整齐划一的特性，一旦某行内某个单元格的高度设定值大于其他单元格的高度值，那么对应行将以该值作为行高。类似的，如果一个单元格的宽度值设定大于同一列中的其他单元格，也会使得该列内所有的单元格均采用这个最大值作为宽度值。

HTML 4已不再赞成使用height和width属性，并且建议使用height和width样式单属性，但后者却无法在Navigator 4中对表格单元格进行控制。

示例 `<th width="80%" height="30">`

值 任意一个单位为像素的长度值，或可用空间的百分比。

默认值 由单元格内实际内容的具体尺寸决定。

对象模型引用方式

```
[window.]document.getElementById(elementID).height
[window.]document.getElementById(elementID).width
```

nowrap

IE all NN all Moz all Saf all Op 7 HTML 3.2

```
nowrap
```

可选

如果出现nowrap属性，那么浏览器会根据需要加宽单元格，以便将单元格内的文本不间断地显示在一整行上。当单元格内内容很多时，如果滥用此属性，那么为了显示所有内容，会在页面上出现一个非常不便于使用的横向滚动条。HTML 4.0已不再赞成使用nowrap属性，而推荐使用样式单属性white-space:nowrap及对应的属性值。

示例 `<th nowrap>`

<th>

值 如果此属性已出现，就意味着其属性值为true。

默认值 false

对象模型引用方式 [window.]document.getElementById(*elementID*).noWrap

rowspan

IE *all* NN *all* Moz *all* Saf *all* Op *7* HTML *3.2*

```
rowspan="行数量"
```

可选

此属性决定了表格单元格扩展时所横跨的列数。rowspan中的数值每增加一个，则需要在该单元格所在行的下一行的同一位置减少一个th或td元素。

示例 <th rowspan="2">

值 任意正整数，但通常是等于2或更大的数值。

默认值 1

对象模型引用方式 [window.]document.getElementById(*elementID*).rowSpan

scope

IE *6* NN *n/a* Moz *all* Saf *all* Op *n/a* HTML *4*

```
scope="作用域常量"
```

可选

当前单元格可以作为其他单元格的表头，而通过scope属性则可以设置这些单元格的具体范围。对那些结构十分规范的表格而言，即使不定义表头单元格的id属性，使用scope属性也可以很简单地完成header属性所具有的功能。尽管一些主流浏览器声称将会支持此属性，但到目前为止，尚未出现相关支持。

示例

```
<tr>
<th scope="col">Product Number</th>
<th scope="col">Description</th>
</tr>
<tr>
<td>0392</td>
<td>Round widget</td>
</tr>
```

值

可使用以下4个作用域常量之一。

col

当前单元格内的文本成为列内所有剩余单元格的标题文本。

colgroup

当前单元格内的文本成为colgroup元素内所有剩余单元格的标题文本。

row

当前单元格内的文本成为tr元素内所有剩余单元格的标题文本。

rowgroup

当前单元格内的文本成为tbody元素内所有剩余单元格的标题文本。

默认值 无。

valign

IE *all* NN *all* Moz *all* Saf *all* Op *7* HTML *3.2*

```
valign="对齐常量"
```

可选

此属性决定了td元素内所有内容的水平对齐方式。在对应的单元格内，此属性设定会取代其外部容器（如tr或tbody）的同名属性设定。

示例 `<th valign="bottom">`

值 目前有4个可用的常量值：top、middle、bottom和baseline。当属性值设定为top和bottom时，其内容将会齐平或紧贴表格单元格的顶部或底部显示。设置为middle（默认值）时，其内容会恰好悬浮在单元格的竖直中心处。如果单元格内的内容可能会依照通常的窗口宽度（假设为表格宽度）分割为多行，那么最好将valign属性设置为baseline。这样就可以保证单元格内文本的第一行的字符基线能够与行内其他单元格的文本对齐，而这是最符合审美观的编排方式。

289

默认值 middle

对象模型引用方式 `[window.]document.getElementById(elementID).vAlign`

width

参见height属性。

<thead>

IE 3 NN n/a Moz all Saf all Op 7 HTML 4

`<thead>...</thead>` HTML结束标签：可选

thead是一种包含特殊意义的容器，它可以包含一行或多行表格单元格并显示在表格的底部。在一个table元素中，thead元素的数量不应多于一个，而且在源代码中，thead元素的位置紧接在table元素的起始标签之后。可以根据需要，在thead元素中使用任意的td、th元素组合。与tbody和tfoot元素相类似，thead元素也是一种行分组元素。Navigator 4会忽略thead元素的标签，因此它会根据源代码的顺序将其内嵌的tr元素按照普通tr元素的方式进行显示。

示例

```
<table cols="3">
<thead>
<tr>
<th>Time</th><th>Event</th><th>Location</th>
</tr>
</thead>
<tfoot>
<tr>
<th>Time</th><th>Event</th><th>Location</th>
</tr>
</tfoot>
<tbody>
<tr>
<td>7:30am-5:00pm</td><td>Registration Open</td><td>Main Lobby</td>
</tr>
<tr>
<td>9:00am-12:00pm</td><td>Keynote Speakers</td><td>Cypress Room</td>
</tr>
</tbody>
</table>
```

对象模型引用方式 `[window.]document.getElementById(elementID)`

290

元素专有属性 align、bgcolor、ch、char、charoff、choff、valign

元素专有事件处理属性 无。

align

IE 4 NN n/a Moz all Saf all Op 7 HTML 4

`align="对齐常量"` 可选

在thead元素控制的行中，此属性确定其单元格水平对齐的方式。

示例 `<thead align="center">`

值

HTML 4与各种的浏览器分别实现了不同的属性值集合。

值	IE/Windows	其他	HTML 4	值	IE/Windows	其他	HTML 4
center	·	·	·	left	·	·	·
char	–	–	·	right	·	·	·
justify	–	·	·				

center、left和right这3个值不解自明。如果使用justify这种对齐方式，将会分散排列多行内容，使它们与左右边缘均对齐。而针对char这种对齐方式，在使用时必须为char属性指定一个字符，并将会以这个字符为中心进行对齐。在HTML 4规范的示例中，未包含字符的内容将会与同一列中其他行的文本内容右对齐。

必须牢记的是，align属性将作用于thead元素所涉及的每一行中的所有单元格，其中还包括为表格指定的全部th元素。如果希望调整起始行的样式，可以使用单独的align属性或text-align样式单属性为th元素设定样式。

默认值 left

对象模型引用方式 `[window.]document.getElementById(elementID).align`

bgcolor

IE *4* NN *n/a* Moz *all* Saf *all* Op *n/a* HTML *n/a*

`bgcolor="颜色三元组或颜色名"` 可选

此属性为thead元素包含的所有单元格设置填充色，此颜色将位于文本和其他内容之下。

示例 `<thead bgcolor="tan">`

值 一个十六进制的三元组或明语颜色名称。不设置属性值就默认为黑色（#000000，black）。目前可用的明语颜色名称请参见附录A。

默认值 随着浏览器、浏览器版本和操作系统的不同而改变。

对象模型引用方式 `[window.]document.getElementById(elementID).bgColor`

char

IE *n/a* NN *n/a* Moz *n/a* Saf *n/a* Op *n/a* HTML *4*

`char="某字符"` 可选

char属性用于指定一个字符，thead元素中某一单元格内的文本将会以该字符作为对齐的参考点。只有当align属性设置为char时，此属性才有价值。而微软则提出了一个ch属性，它可以和标准的char属性相对应。可惜的是，浏览器无论如何都不会响应char或ch属性。

示例 `<thead align="char" char=".">`

值 任意一个单独的字符。

默认值 无。

charoff

IE *n/a* NN *n/a* Moz *n/a* Saf *n/a* Op *n/a* HTML *4*

`charoff="长度值"` 可选

charoff属性用于指定一个特定的偏移点，char属性所指定的字符将在单元格内的该位置上出现。提供此属性主要是为了防止浏览器的默认定位不符合表格的设计目标。而微软则提出了一个choff属性，它可以和标准的charoff属性相对应。可惜的是，浏览器无论如何都不会响应char或ch属性。

示例 `<thead align="char" char="." charoff="80%">`

值 任意一个单位为像素的长度值或单元格空间的百分比。

默认值 无。

choff

参见charoff。

valign

IE 4 NN n/a Moz all Saf all Op 7 HTML 4


```
valign="对齐常量"
```

可选

在thead元素控制的列单元格中，此属性确定它们竖直对齐的方式。可以为列中的任意一个特定单元格更换竖直对齐方式。

示例 `<thead valign="bottom">`

值 目前有4个可用的常量值：top、middle、bottom和baseline。当属性值设定为top和bottom时，其内容将会齐平或紧贴表格单元格的顶部或底部显示。设置为middle（默认值）时，其内容会恰好悬浮在单元格的竖直中心处。如果单元格内的内容可能会依照通常的窗口宽度（假设为表格宽度）分割为多行时，那么最好将valign属性设置为baseline。这样就可以保证单元格内文本的第一行的字符基线能够与行内其他单元格的文本对齐，而这是最符合审美观的编排方式。

默认值 middle

对象模型引用方式 `[window.]document.getElementById(elementID).vAlign`

<title>

IE all NN all Moz all Saf all Op 7 HTML all

```
<title>...</title>
```

HTML结束标签：必选

title元素指明了当前文档的大致内容。尽管其元素内容并不会显示在文档中，但浏览器会在窗口的标题栏上显示这些标题内容。在每个文档中仅允许使用一个title元素，并且它必须放置在head元素之内。由于不是所有的访问者都会依次访问Web站点上的文档，因此有必要为文档指定一个较为详细的标题。同时，它还可以为文档提供一定的上下文语境。

示例 `<title>Declaration of Independence</title>`

对象模型引用方式 `[window.]document.getElementById(elementID)`

元素专有属性 无。

元素专有事件处理属性 无。

<tr>

IE all NN all Moz all Saf all Op 7 HTML all


```
<tr>...</tr>
```

HTML结束标签：可选

tr是一种用于放置一行单元格的容器元素。行内的每个单元格都是一个th或td元素。每一行至少需要一个起始标签，以便命令浏览器在表格的下一行中开始显示随后的单元格元素。在其他行分组元素中，tfoot和thead元素带有一定的特殊功能，而tbody则是一种常用的分组元素。

示例

```
<table cols="3">
<thead>
<tr>
<th>Time</th><th>Event</th><th>Location</th>
</tr>
</thead>
```

<tr>

```
<tbody>
<tr>
<td>7:30am-5:00pm</td><td>Registration Open</td><td>Main Lobby</td>
</tr>
<tr>
<td>9:00am-12:00pm</td><td>Keynote Speakers</td><td>Cypress Room</td>
</tr>
</tbody>
</table>
```

对象模型引用方式 [window.]document.getElementById(*elementID*)

元素专有属性 align、background、bgcolor、bordercolor、bordercolordark、bordercolorlight、ch、char、charoff、choff、height、valign、width

元素专有事件处理属性 无。

align

IE *4* NN *n/a* Moz *all* Saf *all* Op *7* HTML *4*

`align="对齐常量"` 可选

此属性确定了tr元素所包含内容的水平对齐方式。

示例 `<tr align="center">`

值 HTML 4与各种的浏览器分别实现了不同的属性值集合。

值	IE/Windows	其他	HTML 4	值	IE/Windows	其他	HTML 4
center	·	·	·	left	·	·	·
char	–	–	·	right	·	·	·
justify	–	·	·				

center、left和right这3个值不解自明。如果使用justify这种对齐方式，将会分散排列多行内容，使它们与左右边缘均对齐。而针对char这种对齐方式，在使用时必须为char属性指定一个字符，并将会以这个字符为中心进行对齐。在HTML 4规范的示例中，未包含字符的内容将会与同一列中其他行的文本内容右对齐。

必须牢记的是，align属性将作用于tr元素包含的所有单元格，其中还包括为表格指定的全部th元素。如果希望调整起始行的样式，可以使用单独的align属性或text-align样式单属性为tr元素或th元素设定样式。

默认值 center

对象模型引用方式 [window.]document.getElementById(*elementID*).align

background

IE *n/a* NN *4* Moz *all* Saf *all* Op *7* HTML *n/a*

`background="URL"` 可选

此属性可指定一个图像文件作为对应行的背景。与浏览器载入的其他普通图像不同的是，背景图会以未经缩放的原始大小进行加载，然后贴附并充满整个可用的单元格空间。小图片虽然下载迅速，但总是会很明显地在背景中重复显示。然而，值得注意的是，Navigator 4需要图片至少大于16像素×16像素。也可以使用GIF动画作为背景图，但这实在是一件令读者抓狂的做法。在选择背景图时，一定要选择相对柔和的图片，以便使主体内容更加引人注目。如果一定要使用背景图，那么请选用非常淡雅的图片。

示例 `<tr background="watermark.jpg">`

值 任何有效的图像文件URL地址，既可使用相对URL也可以使用绝对URL地址。

默认值 无。

对象模型引用方式 [window.]document.getElementById(*elementID*).background

bgcolor

IE 4 NN 3 Moz all Saf all Op 7 HTML 4

295

```
bgcolor="颜色三元组或颜色名"
```

可选

此属性为tr元素包含的所有单元格设置填充色，此颜色将位于文本和其他内容之下。

示例 `<tr bgcolor="lavender">`

值 一个十六进制的三元组或明语颜色名称。不设置属性值就默认为黑色（#000000，black）。目前可用的明语颜色名称请参见附录A。

默认值 随着浏览器、浏览器版本和操作系统的不同而改变。

对象模型引用方式 `[window.]document.getElementById(elementID).bgColor`

bordercolor

IE 3 NN n/a Moz n/a Saf n/a Op n/a HTML n/a

```
bordercolor="颜色三元组或颜色名"
```

可选

此属性可以设置环绕在单元格和整个表格四周的边框的颜色。IE会将颜色应用于组成单元格边框的四条线段之上。因此，临近的单元格边框颜色并不会互相冲突。

示例 `<tr bordercolor="green">`

值 一个十六进制的三元组或明语颜色名称。不设置属性值就默认为黑色（#000000，black）。目前可用的明语颜色名称请参见附录A。

默认值 由浏览器和操作系统共同决定。

对象模型引用方式 `[window.]document.getElementById(elementID).borderColor`

bordercolordark, bordercolorlight

IE 3 NN n/a Moz n/a Saf n/a Op n/a HTML n/a

```
bordercolordark="颜色三元组或颜色名" , bordercolorlight="颜色三元组或颜色名"
```

可选

如果在页面背景或默认颜色四周仔细地勾勒出明暗线条，还可以在IE中创造出3D效果的表格边框（请参见本书在探讨table元素时给出的图1-4）。通过为bordercolordark（单元格的左边缘与上边缘）和bordercolorlight（单元格的右边缘与下边缘）属性指定属性值，可以独立地控制明暗两种线条的颜色。

此时，应该为这对属性设置互补的颜色。而对于具体的颜色设置并没有强制性要求，并不一定要为bordercolordark属性设置暗色。这两个属性仅仅是为了使得边框线条轮廓分明，对它们进行控制就可以改变一些边框线条的颜色。

示例 `<tr bordercolordark="darkred" bordercolorlight="salmon">`

296

值 一个十六进制的三元组或明语颜色名称。不设置属性值就默认为黑色（#000000，black）。目前可用的明语颜色名称请参见附录A。

默认值 由操作系统决定。

对象模型引用方式

```
[window.]document.getElementById(elementID).borderColorDark
[window.]document.getElementById(elementID).borderColorLight
```

char

IE n/a NN n/a Moz n/a Saf n/a Op n/a HTML 4

```
char="某字符"
```

可选

char属性用于指定一个字符，tr元素中某一单元格内的文本将会以该字符作为对齐的参考点。只有当align属性设置为char时，此属性才有价值。而微软则提出了一个ch属性，它可以和标准的char属性相对应。可惜的是，浏览器无论如何都不会响应char或ch属性。

<tt>

示例 `<tr align="char" char=".">`

值 任意一个单独的字符。

默认值 无。

charoff

IE *n/a* NN *n/a* Moz *n/a* Saf *n/a* Op *n/a* HTML *n/a*

`charoff="长度值"` 可选

charoff属性用于指定一个特定的偏移点，char属性所指定的字符将在单元格内的该位置上出现。提供此属性主要是为了防止浏览器的默认定位不符合表格的设计目标。而微软则提出了一个choff属性，它可以和标准的charoff属性相对应。可惜的是，浏览器无论如何都不会响应char或ch属性。

示例 `<tr align="char" char="." charoff="80%">`

值 任意一个单位为像素的长度值或单元格空间的百分比。

默认值 无。

choff

参见charoff。

valign

IE *4* NN *n/a* Moz *all* Saf *all* Op *7* HTML *4*

`valign="对齐常量"` 可选

在tr元素控制的列单元格中，此属性确定其竖直对齐的方式。可以为行中的任意一个特定单元格更换竖直对齐方式。

示例 `<tr valign="bottom">`

值 目前有4个可用的常量值：top、middle、bottom和baseline。当属性值设定为top和bottom时，其内容将会齐平或紧贴表格单元格的顶部或底部显示。设置为middle（默认值）时，其内容会恰好悬浮在单元格的竖直中心处。如果单元格内的内容可能会依照通常的窗口宽度（假设为表格宽度）分割为多行时，那么最好将valign属性设置为baseline。这样就可以保证单元格内文本的第一行的字符基线能够与行内其他单元格的文本对齐，而这是最符合审美观的编排方式。

默认值 middle

对象模型引用方式 `[window.]document.getElementById(elementID).vAlign`

<tt>

IE *all* NN *all* Moz *all* Saf *all* Op *7* HTML *all*

`<tt>...</tt>` HTML结束标签：必选

tt元素会以等宽字体显示其中的文本内容，这种效果类似于打印机输出。与上下文设置元素不同，tt仅仅是一种排版元素。针对计算机程序代码或输入的上下文设置，可以参考code、kbd和samp元素。与大多数字体相关的元素相同，最好使用样式单来控制元素的具体样式。

示例 `<p>The computer said, <tt>"That does not compute."</tt></p>`

对象模型引用方式 `[window.]document.getElementById(elementID)`

元素专有属性 无。

元素专有事件处理属性 无。

<u>

IE 3 NN 3 Moz all Saf all Op 7 HTML 3.2

<u>...</u>

HTML 结束标签：必选

浏览器将为u元素中的文字内容添加下划线。HTML 4并不赞成使用此属性，而建议使用样式单属性text-decoration:underline。在严格的HTML 4或XHTML DTD中，此元素无效，同时，用户也可能无法区分这些下划线文本与可点击的链接。

298

示例 `<p>You may already be a <u>winner</u>!</p>`

对象模型引用方式 `[window.]document.getElementById(elementID)`

元素专有属性 无。

元素专有事件处理属性 无。

IE all NN all Moz all Saf all Op 7 HTML all

...

HTML 结束标签：必选

ul元素是为无序条目列表提供的一种容器。无序列表中的每个条目在显示时，均会加上一个不代表具体顺序的前置符号（符号的具体样式由type属性或样式单的list-style-type属性决定），而显示顺序则由条目在列表中的具体位置决定。通过嵌套的li元素，可定义每一个列表条目内容。如果在ul元素上应用了样式单，那么其内嵌的li元素也会继承其样式（只有Navigator 4有时会出现一些莫名的问题）。

示例

```
<ul>
   <li>Africa</li>
   <li>Antarctica</li>
   <li>Asia</li>
   <li>Australia</li>
   <li>Europe</li>
   <li>North America</li>
   <li>South America</li>
</ul>
```

对象模型引用方式 `[window.]document.getElementById(elementID)`

元素专有属性 compact、type

元素专有事件处理属性 无。

compact

IE 4 NN n/a Moz all Saf all Op n/a HTML 3.2

299

compact

可选

该属性值是一个布尔变量，设计的初衷是让浏览器能够以另一种更为紧凑的样式显示列表，即与正常情况相比，使条目之间的行距更小。尽管为了顾及兼容性，依然支持compact属性，但它在主流浏览器上已失去了作用。因此，请使用样式单来控制元素尺寸大小与行间距。

示例 `<ul compact>...</ul>`

值 如果此属性已出现，就意味着其属性值为true。

默认值 false

type

IE all NN all Moz all Saf all Op 7 HTML 3.2

type="标注类型"

可选

type属性为如何在浏览器中显示前置符号或序列号提供了一定的灵活性。通过此属性，可以指定前置符号的

具体样式，如圆点、圆圈或方块。在这几种符号中，圆圈内的空白空间被充满后就是圆点符号。在早期的Macintosh浏览器中，方块符号还只是一个方形边框，而在Windows及所有浏览器下的现代浏览器中，它已经演变成了一种实心方块。HTML 4.0已不再赞成使用type属性，推荐使用样式单属性list-style-type。

示例 `<ul type="disc">...</ul>`

值 可使用的值为circle | disc | square。

默认值 disc

对象模型引用方式 `[window.]document.getElementById(elementID).type`

<var>

IE *all* NN *all* Moz *all* Saf *all* Op *7* HTML *all*

`<var>...</var>`

HTML 结束标签：必选

在HTML 4推荐标准中，有一类元素统称为**短语元素**（phrase element），var元素也是其中之一。这些元素为文档的特定部分指明了结构上的含义。var元素通常用于显示一个或多个表示计算机程序变量名的字符。

浏览器可以自由决定如何（或是否）从其他body元素中区分出var元素内容。当前的主流浏览器使用斜体字体来显示var元素中的内容。可以使用任何合适的样式单来替换默认的显示形式。

示例 `<p>The value of <var>offsetWidth</var> becomes 20.</p>`

对象模型引用方式 `[window.]document.getElementById(elementID)`

元素专有属性 无。

元素专有事件处理属性 无。

<wbr>

IE *all* NN *all* Moz *n/a* Saf *n/a* Op *n/a* HTML *n/a*

`<wbr>`

HTML 结束标签：禁用

如果使用nobr元素定义了不包含任何回车或换行的内容，那么可以使用wbr元素来建议浏览器根据其窗口宽度对其内容进行换行。在源代码中使用wbr元素，就可以标明这些临时换行的位置。从某种意义上说，nobr和wbr元素均给予作者控制网页内容换行的权利。但HTML规范并未包含这两个元素，而且仅有IE在继续推动wbr元素的使用。

示例

```
<nobr>This is a long line of text that could run on and on, <wbr>forcing
the browser to display the horizontal scrollbar after awhile.</nobr>
```

对象模型引用方式 `[window.]document.getElementById(elementID)`

元素专有属性 无。

元素专有事件处理属性 无。

<xml>

IE *5* NN *n/a* Moz *n/a* Saf *n/a* Op *n/a* HTML *n/a*

`<xml>...</xml>`

HTML 结束标签：必选

在Windows系统中，IE 5及后续版本均支持XML数据岛，它是HTML页面内一种完备的、非显示的XML数据块。在使用过程中，既可以作为HTML文档的一部分（嵌入在起始与结束标签之间）传输这些XML数据，也可以从外部资源中加载它们。一旦加载了这些XML数据，微软的XML DOM（在很多方面，它与W3C DOM的核心部分有些相似）就允许脚本读取这些数据以便进行定制显示。另外，目前通常使用XMLHttpRequest对象和JavaScript将这些XML数据融入HTML页面。详情请参见在线参考VII。

示例

```
<xml id="xmlData">
  <xmlresults>
    <!-- xml data here -->
  </xmlresults>
</xml>
```

对象模型引用方式 [window.]document.getElementById(*elementID*)

属性 src

事件处理程序属性 无。

src

IE *5* NN *n/a* Moz *n/a* Saf *n/a* Op *n/a* HTML *n/a*

`src="URI"` 可选

通过其属性值可指明将要加载至元素中的外部XML数据源。

示例 `<xml id="xmldata" src="http://www.magacorp.com/data/2003Forecast.xml"></xml>`

值 任意一个能够返回XML数据的有效URI。

默认值 无。

对象模型引用方式 [window.]document.getElementById(elementID).src

<xmp>

IE *all* NN *all* Moz *all* Saf *all* Op *7* HTML *<4*

`<xmp>...</xmp>` HTML 结束标签：必选

作为一个块级元素，xmp元素会使用单间距字体显示它所包含的内容，例如，使用80列宽显示的计算机代码清单。在大多数浏览器中，字体尺寸比默认尺寸要小。浏览器需要为元素内容中的回车和其他空白空间留下位置。很早以前HTML就不再推荐使用此元素，HTML 4规范也删除了这个元素。因此，建议使用pre元素代替listing元素。需要注意的是，在下文的示例中，只有Opera才会将实体转换为字符。

示例

```
<xmp>
&lt;script language="JavaScript"&gt;
  document.write("Hello, world.");
&lt;/script&gt;
</xmp>
```

对象模型引用方式 [window.]document.getElementById(*elementID*)

元素专有属性 无。

元素专有事件处理属性 无。

<!-- comment-->

IE *all* NN *all* Moz *all* Saf *all* Op *7* HTML *all*

`<!--comment-->` HTML 结束标签：禁用

comment元素是一个注释元素，它允许作者插入一些浏览器不会显示的内容，但任何阅读源代码的人均可以看到这些注释内容。尽管HTML标准允许在“comment”与右侧角括号之间插入一个空格，但大多数浏览器并不允许出现这个空格。

在其他非IE浏览器中，微软通过HTML注释的非显示特性来实现一种称为“条件注释”的系统指令。IE 5的Windows版本首先引入了这种“**条件注释**”，它提供了一种途径，使得特定的HTML内容仅在IE 5及更早的浏览器版本中显示，或者在IE 5之外的所有版本中显示，甚至只在指定的后续浏览器版本中显示。注释标签中的表达式可以控制这种条件注释，例如，判断当前浏览器是否为IE 6或后续版本。

<!-- comment-->

目前有两种类型的条件注释：*downlevel-hidden*及*downlevel-revealed*。使用前者时，可以从始于IE 5的某个版本开始支持HTML代码，而后者则使得HTML内容可以在IE 5或后续版本之外的其他浏览器中得以显示。这两种类型的注释语法稍有不同。

downlevel-hidden类型的条件注释语法如下：

```
<!--[if expression]> HTMLContent <![endif]-->
```

对于这些包含了if和endif指令的方括号，Windows下的IE 5及后续版本浏览器会将它们理解为条件注释，而且当表达式值为true时，这些浏览器还会显示位于*HTMLContent*中的内容。而其他浏览器只会将这些代码视为普通的HTML注释，而不会显示任何内容。

downlevel-revealed类型的条件注释语法如下：

```
<![if expression]> HTMLContent <![endif]>
```

注意，在这种注释形式中不存在连字符。因此在这种形式下，除IE 5及后续浏览器外，其他浏览器总会显示位于*HTMLContent*部分中的内容。如果希望IE 5及5.5显示这些内容，而IE 6及后续版本不显示，那么可以创建一个**表达式**，以便通知IE浏览器在对应的条件下才显示*HTMLContent*中的信息，代码如下：

```
<![if lt IE 6]
<script type="text/javascript" language="JavaScript">
    // 用于非 IE 浏览器或 IE6 之前的代码
</script>
<![endif]>
```

这个表达式包括以下3个部分：一个浏览器对象（目前仅支持IE），一个比较运算符（可以忽略“等于”运算符）及一个版本号（大于等于5）。目前可使用的比较操作符包括：lt（小于）、gt（大于）、lte（小于或等于）及gte（大于或等于）另外，还可以在浏览器对象前使用“!”号，以对表达式取反，例如，“<![if !IE 7]>”表示“如果IE版本号不为7”。

如要获取有关版本向量及其他提示的详细信息，其访问此网址：*http://msdn.microsoft.com/zh-cn/library/cc817577.aspx*。

303

示例

```
<!-- layout inspired by www.example.com/catalog.html -->
```

对象模型引用方式 通过相邻节点的引用或根据节点类型遍历文档树。

元素专有属性 无。

元素专有事件处理属性 无。

文档对象模型参考

Document Object Model Reference

本章主要着眼于文档中的各种对象。一旦浏览器加载了一个文档，那么它就会在其内存空间中维护这些可进行脚本操作的实体对象。在浏览器中解释文档内容的内嵌标签时，就会创建大部分对象。但是，出于增强脚本活动性的目的，很多对象往往独立存在，例如事件处理、窗口操作、使用新对象来创建并增补文档、获取客户端系统环境资源，以及使用AJAX实现客户端与服务器之间的XML数据交换等。

通过属性、方法、内嵌的项目集合及事件，可以完整地描述一个对象。动态HTML也指出，如果页面访问者所使用的浏览器能够提供相应的支持，那么依靠对象及其属性、方法和事件，就可以将开发者与文档联系起来。在早期的浏览器中，脚本对象模型还比较简陋，它只包含相对较少的对象，而且为每个对象所实现的属性、方法和事件也较少。然而，由于微软的IE 4为对象模型进行了大范围的扩展，而且W3C最近也在不断地增加一些全新的抽象对象模型，这些工作都使得当前的对象模型非常巨大。W3C DOM是当前最新的对象模型，目前所有的主流脚本化浏览器均不同程度地支持这一模型，因此，任何新的DHTML脚本操作均应以它作为目标。

为了帮助开发者在页面开发中选择合适的对象、属性、方法和事件处理程序，本章罗列了所有与动态HTML相关的客户端对象模型，它们已在微软的IE、早期的Mozilla Netscape（NN）、Mozilla（Moz）、Safari（Saf）、Opera（Op）等浏览器中得到了一定的应用，另外本章还对W3C（DOM）及DOM 3模型内一些接近完成的内容也进行了阐述。通过本章内容，开发者可以判断某个对象及其属性、方法或事件是否能够在目标应用环境中正常工作。

如果须要兼容不同的浏览器，那么请关注浏览器对每个应用项的支持能力及对应的浏览器版本号。此处的版本号表示一类浏览器支持某项的起始版本，一个特殊的情况是，Opera从版本7开始表示其兼容性。

305

当浏览器未实现相关支持时，本章中使用“n/a”来表示。当只有一个单独版本的浏览器支持时，则使用管道符将版本号括起来以进行标示，例如，|4|代表只支持版本4。对于Mozilla基金会浏览器的衍生版本，如Firefox、Netscape 6及后续版本、Camino等，请参见附录F。须要注意的是，在某些特定的浏览器品牌和版本下，有些项目并不能在所有的操作系统平台中均正常使用，例如IE 5就存在这种问题。如果在Win32和Macintosh中对这种差异性进行测试，就可以发现一些反常的现象，从而证明不同的操作系统对同一版本的浏览器的支持的确存在差别。例如，“IE 5”包括Windows和Mac两个操作系统下的版本，而“IE 5（Win）”则只代表Windows版，同时“IE 5.5”及后续版本均仅代表Windows版。

接下来的内容将会介绍所有HTML元素对象所共有的属性、方法和事件，本章按照对象类型的字母顺序来组织内容。HTML元素对象类型和它们对应的HTML元素标签名比较类似，它们均采用XHTML的小写字母形式。除非须要将标签名称字符串作为方法参数（例如`document.getElementsByTagName("h1")`），否则脚本通常不会使用它们的名称进行元素对象引用。脚本通常采用另外几种方法来完成这一工作，以便生成有效的元素对象引用。为元素的`id`属性指定一个标识符，然后使用这个标识符来创建引用是一种最常用的方法，对应的W3C DOM语法如下：

```
document.getElementById("elementID")
```

此时，输入参数是元素标识符字符串。创建引用之后，实质上可以通过有效的元素对象引用来调用对象模型的众多属性，并不须要每次都通过对象标识符来创建显式引用。例如，event对象就包含一个指向事件元素的有效引用。这样一来，脚本指令就可以根据需要使用这个引用来访问元素对象的属性或方法。

由于W3C DOM中所涉及的内容非常巨大，因此还不能将它们全部应用于当今的浏览器之中。例如，本章中列出了一个名为“div”的元素对象，而它在W3C DOM中的正式名称则为“HTMLDivElement”。为了让本书紧凑充实，本章并未将W3C DOM的HTML元素对象单独列出，但可以参考下文中的“静态W3C HTML DOM对象”以获得更多细节。

属性值类型

在DOM中，很多属性都共享类似的数据要求。因此为了使得参考列表简单明了，本节仅详细介绍了其中一部分常用属性值类型。当属性与这些属性值类型中的某个有关时，请参照本节中对此类型具体说明。

长度

长度值定义了文档空间状态的线性度量。其度量单位可以是任何能够在屏幕上标识位置或空间的适用单位。对于那些反映了HTML属性（attribute）的DOM属性（property），长度单位均为像素。但在其他场合中，如在层叠样式单（参见第4章）中设置这一属性时，度量单位可以用英寸（inche）、派卡（pica）、em或其他相关度量单位来表示。当它用于定义距离某个元素边缘的偏移量时，一个单一的数字值就可以表示一个长度。例如，坐标点（10,20）包含两个长度值，分别表示该点距离某个元素的左边缘和上边缘的像素值。

标识符

标识符是一个符合严格语法规则的名字。最重要的是，标识符是一个不允许出现空格的单词。如果须要使用多个单词来描述一个项目，既可以使用骆驼拼写法（内部单词首字母大写，如lowerCamelCase），也可以在单词间使用下划线进行间隔（如this_is_a_sample）。除所有数字和字母外，不允许在标识符中使用大多数的标点符号。脚本语言还不允许将数字作为标识符的第一个字符。

URI 与 URL

通用资源标识符（Universal Resource Identifier，简称URI（译注1））是一个用于网络内容地址的广义术语。而国际化资源标识符（Internationalized Resource Identifier，简称IRI）则是一个可以包含非ASCII字符的Unicode字符地址。通用资源定位器（Universal Resource Locator，简称URL（译注2））是URI中的一种。由于大多数浏览器只关注URL，所以对网站建设者而言，可以将它们视同一体。另外，URL既可以是完整路径（包括协议、主机、域等部分）也可以是相对于当前文档URL的相对路径。对后者而言，它意味着该URL可能由锚链、文件或路径名组成。须要注意的是，使用URL作为对象属性时，须要将该URL字符串用引号括起来。

语言代码

目前有一个详尽的标准代码列表用以标识世界上的口语与书面语。一种语言代码通常会包含一个主要语言代码，如“en”代表英语，而“zh”则代表中文。常用的两字母的主要语言代码收录于ISO639（如须查看代码摘录列表可以访问*http://www.ietf.org/rfc/rfc1766.txt*）。通常为区分某种语言在不同国家或地区的使用习惯，还可以用一个可选的子代码（通过一个连字符与主代码分开）来标识主语言的一种特定的实现。例如，虽然“en”代表英语，但“en-US”代表一种特定的英语——美式英语。因为它的含意关系到元素属性的具体取值，浏览器必须支持一种特定的语言代码。

译注1：W3C 中全称为 Uniform Resource Identifier。

译注2：W3C 中全称为 Uniform Resource Locator。

颜色

颜色值可以使用十六进制三元组或对应的明语（plain-language）来表示。十六进制3元组由三对十六进制数字组成，每个数字的取值范围为从00到FF，分别对应于红、绿、蓝3个颜色成分。将3对数字组合在一起，并在前面再加一个“#”号就可以表示一种颜色，如“#rrggbb”。因此在全红（#FF0000）中仅包含red值（FF），其他2个色调均为空（00），而纯蓝则由#0000FF表示。颜色的这种表示方法也可以使用小写字母。对于那些应用于样式单属性的颜色值，颜色可能须要使用RGB（red-green-blue，红-绿-蓝）格式来表示，以便与CSS规范保持一致（请参见第4章）。

这种数值表达方式可以导致数量巨大的可能组合，使用它们可以表示多达1 600余万种不同的颜色。而在Web出现初期，典型的PC显示设备受处理能力和内存的限制，只能显示256色，这意味着如果页面访问者使用这样的显示设备根本无法区分上千万种颜色之间细微差别。因此网页开发者往往使用一种称为**“网络安全色”**的颜色设置，它仅包含216种区别明显的颜色。尽管当今的计算机已经有充足的处理能力和内存，可以轻松适应这上千万种颜色，但一些页面设计者依然坚持使用更为有限的颜色集合以保证后向兼容性。一个可以适用于所有浏览器和计算机的颜色参考说明可以在此链接中获取：*http://www.w3school.com.cn/html/html_colors.asp*（译注3）。

HTML推荐标准指定了一组由16种颜色组成的基本颜色库，并辅以明语命名。注意，这些颜色名并不区分大小写。这些颜色名与对应的十六进制三元组列表如下：

Black	#000000	Maroon	#800000	Green	#008000	Navy	#000080
Silver	#C0C0C0	Red	#FF0000	Lime	#00FF00	Blue	#0000FF
Gray	#808080	Purple	#800080	Olive	#808000	Teal	#008080
White	#FFFFFF	Fuchsia	#FF00FF	Yellow	#FFFF00	Aqua	#00FFFF

换句话说，将颜色属性值设置为bgcolor="Aqua"与bgcolor="#00FFFF"会产生相同的颜色效果。

多年以前，Netscape还制定了一个更长的明语颜色对应表，该颜色表最初由X Windows 系统调色板采用，因此称为“X11颜色”。附录A列出了详细的X11颜色，目前主流版本的浏览器均可以识别这些颜色名称。（译注4）

关于 client-与 offset-属性集

在IE 4中，微软公司为那些充当正文内容的元素引入了一系列的尺寸控制与定位属性。这些属性可以辅助脚本来决定主体内容的位置和尺寸，这样一来，主体中放置的元素就可以根据这些固定的元素来移动。这些属性包括：clientHeight、clientLeft、clientTop、clientWidth、offsetHeight、offsetLeft、offsetTop、offsetWidth。

然而不幸的是，由于IE/Windows组合在常用的环境下很容易出现问题，而IE/Mac组合又采用了另外一种完全不同的实现思路，因此这些属性在实际中很难使用。之后，当DOCTYPE元素将浏览器设置为标准兼容（standards-compliant）模式时，通过修改浏览器窗口，微软尝试着在IE 6中修复这一错误。早期的Mozilla版本浏览器还实现了一些非W3C属性，以方便DHTML开发者，但是这种方式更类似于早期的IE/Windows模式，与IE 6的标准兼容模式差别较大。糊涂了吗？这还用说！

浏览器与兼容模式之间的主要测量误差往往与元素的填充与定位环境（对未定位的元素也如此）有关。元素与样式单组合所带来的新奇排列方式往往令人难于想象，因此这里使用一个常见的任务作为范例，来说明在不同的浏览器和兼容模式下可能存在的各种问题。此时，我们使用脚本将一个由CSS定位的元素内容直接放置在一个行内元素（inline element）的内容之上。由于本节中所描述的这些属性并不是W3C DOM（至Level 2

译注3：原书中提供的链接已无法访问。

译注4：颜色名称与真实颜色的对应关系的可参见此网页：*http://www.w3school.com.cn/html/html_colornames.asp*。

为止）的一部分，因此很难说哪种方式是“正确”的。一个更值得关注的问题是，如何使用这些属性在目标浏览器上完成预定的任务。

在这种情况下，如果将下列标签插入文档中的任意位置，那么元素的精确位置会随着浏览器、窗口尺寸及其他环境状态的不同而发生改变。

```
<img id="fixedImg" src="bkgnd.jpg" alt="Locator box" height="90"
    width="120" style="padding:2px; border:3px solid green; margin:5px">
```

由于图像尺寸不会随着浏览器大小而发生改变，因此此处使用它作为范例（例如，此时就不会受到不同字体显示效果的影响）。这个例子能够反映出不同的尺寸属性如何描述元素的全貌或内容尺寸。对于固定的元素而言，包括填充、边框和边距在内的一些设置均能明显地说明，这些样式单属性能够影响文档中元素的外形特征。表2-1列出了在几种浏览器和兼容模式下对应的尺寸值与定位属性。

309

表 2-1：120 像素 × 90 像素行内元素的属性值比较（单位：像素）

属性名	IE 7（Quirks 模式）	IE 7（CSS 模式）	FF 1.5（CSS 模式）	Saf 2	Op 9（CSS 模式）
clientLeft	3	3	n/a	n/a	3
clientTop	3	3	n/a	n/a	3
offsetLeft	15	5	13	5	13
offsetTop	243	228	251	215	229
clientWidth	120	124	124	124	124
clientHeight	90	94	94	94	94
offsetWidth	126	130	130	130	130
offsetHeight	96	100	100	100	100
naturalWidth	n/a	n/a	120	n/a	n/a
naturalHeight	n/a	n/a	90	n/a	n/a
width	120	124	120	120	120
height	90	94	90	90	90
offsetParent. clientLeft	2	0	n/a	n/a	0
offsetParent. clientTop	2	0	n/a	n/a	0
offsetParent. offsetLeft	0	10	0	8	0
offsetParent. offsetTop	0	15	0	8	0

通过表2-1的大量数据可以看出，不同的浏览器对定位与行内元素尺寸的描述也不尽相同。由于每个浏览器在显示元素周围的内容时彼此之间往往存在一定的差异，而且在各种情况下，任何依赖于位置的脚本均会读取实时数据，因此某些精确值往往并不一定是准确尺寸，如位置坐标等。但是，针对特殊的元素插入，仍然有几个值得注意的细节，例如：

- 在后向兼容（“quirks”）模式下，IE 6和IE 7在计算高度和宽度时与Windows下的IE 4、IE 5完全相同，这有利于保证后向兼容性；
- 在quirks模式下，IE 6和IE 7中的偏移量包括内容两侧的边框厚度；
- 在标准模式下，IE 6和IE 7在计算`offsetLeft`和`offsetTop`属性时会参考`offsetParent`元素，而该元素也拥有自己的偏移量。因此，计算实际位置时必须考虑到`offsetParent`元素的偏移量；
- 在标准模式下，填充区域的厚度还会影响IE 6和IE 7对宽度、高度属性的计算，但这些属性实际上应该影响元素的高度和宽度属性值，因此这种做法是错误的，在实际使用过程中，甚至`getAttribute()`都会返回错误的值；

310

- 标准模式下的IE 6和IE 7、Mozilla、Safari及Opera拥有完全相同的`offsetWidth`和`offsetHeight`值，这两个值中均会包含填充区及边框的厚度，但并不包括边距值。

理解这些问题的关键点在于，将绝对位置的元素与主文档中其他元素混合在一起时，固定项的client和offset属性既有帮助但又会带来麻烦，此时须要视DOCTYPE定义的标准兼容性级别及这些元素是否具有边框、边距与填充等不同情况而定。在标准兼容模式下，可能必须将固定元素的offsetParent坐标考虑在内。尽管在页面上几乎可以实现任何的元素组合，但要真正地达到设计意图，还需要很多的试验、纠错与跨平台测试。如果一个复杂的元素设计与精确定位任务涉及固定元素、已定位的元素和事件组合，那么请不要为它的实现难度感到诧异。

默认属性值

尽管很多属性列表都提供了明确的默认属性值，但这只是一种误导。在当前的浏览器中，如果源代码中并未明确设定一个对应于HTML属性的DOM属性，那么浏览器倾向于为这个元素对象属性返回一个空字符串。但由于即使未设置属性值，元素也会根据默认的规范行事，因此这个空字符串往往会带来误解。块级（block-level）元素的`align`属性就是一个很好的例子。此时，除非设置了其他的属性值，否则元素的行为将会与`align`属性值为“`left`”的情况相同，但由于在标注中并未显式设置`align`属性，因此在默认情况下会返回一个空字符串。

鉴于以上情况，本章罗列了元素对象所需的默认属性值。这样就为属性值的选择提供了一种便捷的方法，在面对一些可能的属性值时就不须要在第1章中查找对应属性以获取其默认值。如果列出的默认值为“无”，就意味着不存在默认的HTML行为，而默认值确实是一个空字符串。

事件

对于那些能够接收事件的对象而言，它们的主要信息中均包含事件列表。与之前的版本不同，本书使用事件类型，即以不包含字首“on”前缀的事件名称来编排事件。如果须要通过事件属性将一个事件处理函数与一个对象进行绑定，那么可以使用带“on”前缀的形式，在线参考VI中的“Binding Event Handlers to Elements”已对具体的绑定方法进行了说明。

如果须要从对象的事件列表中选择合适的事件，可以考虑以下两个因素。首先，大多数HTML 4.0元素均拥有与之关联的内置事件，本章也会列出这些事件，以及那些与HTML元素对应的对象。这样一来，那些在页面上并无任何视觉外观的元素反而拥有了键盘和鼠标事件，这看起来也的确比较怪异。尽管通过脚本使用这些事件的几率几乎为零，但本章依然会将这些事件罗列出来。

第二，IE和W3C DOM的事件冒泡模型（bubbling model）指出，来自于一个元素的事件可以依照元素的层次包含关系一直上升至根节点或根元素。这意味着，大多数内嵌元素中出现的每个事件从本质上来说也可以应用于包含关系链中的所有高级元素之中。换句话说，实质上每个容器元素均拥有与之相关联的冒泡事件类型。可以从第3章中获取更多有关事件类型特性的有关信息。

静态 W3C HTML DOM 模型

各种HTML元素对象实例都是派生自静态DOM对象，本章并未明确地对这类DOM对象进行详细说明。尽管在浏览器（Mozilla和最新版本的Safari和Opera）中进行调试时偶尔会突然出现它们的名称，但总的来说，并不须要使用脚本来直接操作这些静态对象。例如，如果在Mozilla中使用`alert()`方法来显示一个`p`元素的对象引用，那么该引用转换为字符串时其形式为：

```
[object HTMLParagraphElement]
```

很明显，这并不是对象的JavaScript类型（其JavaScript类型仅仅是object）。但是这表示上例中讨论的p元素是W3C DOM中HTMLParagraphElement对象的一个实例。如果浏览器支持对静态对象进行直接访问（如Mozilla、2.0.4版本之后的Safari及Opera 8及后续版本），就可以享受为这些对象的prototype属性添加属性和方法的乐趣。这样一来，通过脚本创建的任何新实例都将拥有这些新的内置属性和方法。请参考第5章中对Array.prototype的说明，以便通过示例了解这种JavaScript机制。

下文列出了W3C DOM Level 2模型中为HTML元素定义的静态对象。所有的这些对象均继承了HTMLElement对象的各种属性和方法。本章后续内容中，在对应的HTML元素对象名称下引入了这些对象的相关说明。

312

HTMLAnchorElement	HTMLAppletElement	HTMLAreaElement	HTMLBaseElement
HTMLBaseFontElement	HTMLBodyElement	HTMLBRElement	HTMLButtonElement
HTMLDirectoryElement	HTMLDivElement	HTMLDListElement	HTMLElement
HTMLFieldSetElement	HTMLFontElement	HTMLFormElement	HTMLFrameElement
HTMLFrameSetElement	HTMLHeadElement	HTMLHeadingElement	HTMLHRElement
HTMLHtmlElement	HTMLIFrameElement	HTMLImageElement	HTMLInputElement
HTMLIsIndexElement	HTMLLabelElement	HTMLLegendElement	HTMLLIElement
HTMLLinkElement	HTMLMapElement	HTMLMenuElement	HTMLMetaElement
HTMLModElement	HTMLObjectElement	HTMLOListElement	HTMLOptGroupElement
HTMLOptionElement	HTMLParagraphElement	HTMLParamElement	HTMLPreElement
HTMLQuoteElement	HTMLScriptElement	HTMLSelectElement	HTMLStyleElement
HTMLTableCaptionElement	HTMLTableCellElement	HTMLTableColElement	HTMLTableElement
HTMLTableRowElement	HTMLTableSectionElement	HTMLTextAreaElement	HTMLTitleElement
HTMLUListElement			

当浏览器须要实现一个既不属于HTML 4也不属于W3C DOM规范的元素时，它应该将HTMLDivElement或HTMLSpanElement对象的行为和/或样式进行扩展，以便给予该元素专有的特征。因此，Mozilla浏览器指出，marquee元素就是派生自HTMLSpanElement对象。

共享的对象属性、方法与事件

无论是IE的私有DOM还是W3C DOM，均公布了大量的属性（包括事件处理属性）及方法，这些属性和方法几乎完全覆盖了HTML元素所对应的全部对象。因此，为了避免在参考列表中重复描述这些条目，仅在此处对通用条目进行详细的说明。由于共享条目数量很多（共有95个属性、74个方法和72个事件类型），因此在本章的剩余内容里，这些共享属性、方法和事件并不会再出现在每一个对象的说明列表中，但实际上它们在每一HTML元素对象中均可使用（本章会标明对应的浏览器或标准文档的版本信息）。

很明显，由于一些可绘制元素对象不能使用在非绘制元素上，因此少数共享属性或方法可能没有任何意义。例如，由于样式元素不会在页面上显示其内容，因此在IE中调用一个样式元素对象的scrollIntoView()方法没有任何意义。由于这些条目都是DOM的一部分，而浏览器的内部架构利用了一种继承机制，这才这使得浏览器能够使用所有的元素对象及相同的基础属性、方法和事件，而不用考虑它们是否具有实际应用意义。因此，如果须要使用这些条目的特殊功能，那么必须首先好好了解这些共享条目及相应的对象特定条目，本章后续内容对它们均进行了说明。

313

在接下来的说明中，示例代码使用条目的*elementID*来指代元素的id属性中指定的标识符。在实际的脚本代码中，应该使用对象的真实ID来替代这个*elementID*。

共享的元素对象属性

ATTRIBUTE_NODE	CDATA_SECTION_NODE	COMMENT_NODE	DOCUMENT_FRAGMENT_NODE
DOCUMENT_NODE	DOCUMENT_POSITION_CONTAINED_BY	DOCUMENT_POSITION_CONTAINS	DOCUMENT_POSITION_DISCONNECTED
DOCUMENT_POSITION_FOLLOWING	DOCUMENT_POSITION_IMPLEMENTATION_SPECIFIC	DOCUMENT_POSITION_PRECEDING	DOCUMENT_TYPE_NODE
ELEMENT_NODE	ENTITY_NODE	ENTITY_REFERENCE_NODE	NOTATION_NODE
PROCESSING_INSTRUCTION_NODE	TEXT_NODE	accessKey	all[]
attributes[]	baseURI	behaviorUrns[]	canHaveChildren
canHaveHTML	childNodes[]	children	cite
className	clientHeight	clientLeft	clientTop
clientWidth	contentEditable	currentStyle	dateTime
dir	disabled	document	filters[]
firstChild	hideFocus	id	innerHTML
innerText	isContentEditable	isDisabled	isMultiLine
isTextEdit	lang	language	lastChild
localName	namespaceURI	nextSibling	nodeName
nodeType	nodeValue	offsetHeight	offsetLeft
offsetParent	offsetTop	offsetWidth	outerHTML
outerText	ownerDocument	parentElement	parentNode
parentTextEdit	prefix	previousSibling	readyState
recordNumber	repeatMax	repeatMin	repeatStart
repetitionBlocks[]	repetitionIndex	repetitionTemplate	repetitionType
runtimeStyle	scopeName	scrollHeight	scrollLeft
scrollTop	scrollWidth	sourceIndex	style
tabIndex	tagName	tagUrn	textContent
title	uniqueID	unselectable	

共享的元素对象方法

addBehavior()	addEventListener()	addRepetitionBlock()
addRepetitionBlockByIndex()	appendChild()	applyElement()
attachEvent()	blur()	clearAttributes()
click()	cloneNode()	compareDocumentPosition()
componentFromPoint()	contains()	createControlRange()
detachEvent()	dispatchEvent()	doScroll()
dragDrop()	fireEvent()	focus()
getAdjacentText()	getAttribute()	getAttributeNode()
getAttributeNodeNS()	getAttributeNS()	getBoundingClientRect()
getClientRects()	getElementsByTagName()	getElementsByTagNameNS()
getExpression()	getFeature()	getUserData()
hasAttribute()	hasAttributeNS()	hasAttributes()
hasChildNodes()	insertAdjacentElement()	insertAdjacentHTML()
insertAdjacentText()	insertBefore()	isDefaultNamespace()
isEqualNode()	isSameNode()	isSupported()
lookupNamespaceURI()	lookupPrefix()	mergeAttributes()
moveRepetitionBlock()	normalize()	releaseCapture()
removeAttribute()	removeAttributeNode()	removeAttributeNS()
removeBehavior()	removeChild()	removeEventListener()
removeExpression()	removeNode()	removeRepetitionBlock()

replaceAdjacentText()	replaceChild()	replaceNode()
scrollIntoView()	setActive()	setAttribute()
setAttributeNode()	setAttributeNodeNS()	setAttributeNS()
setCapture()	setExpression()	setUserData()
swapNode()	toString()	

共享的元素对象事件

事件	IE (Win)	IE (Mac)	Mozilla	Safari	Opera	W3CDOM
DOMActivate[a]	n/a	n/a	n/a	1.3/2	n/a	2
DOMAttrModified[a]	n/a	n/a	n/a	n/a	9	2
DOMCharacterDataModified[a]	n/a	n/a	n/a	n/a	n/a	2
DOMFocusIn[a]	n/a	n/a	n/a	1.3/2	9	2
DOMFocusOut[a]	n/a	n/a	n/a	1.3/2	9	2
DOMNodeInserted[a]	n/a	n/a	n/a	n/a	9	2
DOMNodeInsertedIntoDocument[a]	n/a	n/a	n/a	n/a	n/a	2
DOMNodeRemoved[a]	n/a	n/a	n/a	n/a	n/a	2
DOMNodeRemovedFromDocument[a]	n/a	n/a	n/a	n/a	n/a	2
DOMSubtreeModified[a]	n/a	n/a	n/a	1.3/2	n/a	2
activate	5.5	n/a	n/a	n/a	n/a	n/a
afterupdate[b]	4	5	n/a	n/a	n/a	n/a
beforeactivate	6	n/a	n/a	n/a	n/a	n/a
beforecopy	5	n/a	n/a	1.3/2	n/a	n/a
beforecut	5	n/a	n/a	1.3/2	n/a	n/a
beforedeactivate	5.5	n/a	n/a	n/a	n/a	n/a
beforeeditfocus	5	n/a	n/a	n/a	n/a	n/a
beforepaste	5	n/a	n/a	1.3/2	n/a	n/a
beforeupdate[b]	4	5	n/a	n/a	n/a	n/a
blur[c]	3	3.01	all	all	7	2
cellchange[d]	5	n/a	n/a	n/a	n/a	n/a
click[c]	3	3.01	all	all	all	2
contextmenu	5	n/a	all	all	n/a	n/a
controlselect	5.5	n/a	n/a	n/a	n/a	n/a
copy	5	n/a	n/a	1.3/2	n/a	n/a
cut	5	n/a	n/a	1.3/2	n/a	n/a
dataavailable[d]	4	n/a	n/a	n/a	n/a	n/a
datasetchanged[d]	4	n/a	n/a	n/a	n/a	n/a
datasetcompleted	4	n/a	n/a	n/a	n/a	n/a
dblclick[c]	4	4	all	n/a	all	n/a
deactivate	5.5	n/a	n/a	n/a	n/a	n/a
drag	5	n/a	n/a	1.3/2	n/a	n/a
dragend	5	n/a	n/a	1.3/2	n/a	n/a
dragenter	5	n/a	n/a	1.3/2	n/a	n/a
dragleave	5	n/a	n/a	1.3/2	n/a	n/a
dragover	5	n/a	n/a	1.3/2	n/a	n/a
dragstart	5	n/a	n/a	1.3/2	n/a	n/a
drop	5	n/a	n/a	1.3/2	n/a	n/a
errorupdate[b]	4	5	n/a	n/a	n/a	n/a
filterchange	4	n/a	n/a	n/a	n/a	n/a
focus[c]	3	3.01	all	all	all	2
focusin	6	n/a	n/a	n/a	n/a	n/a

续表

事件	IE (Win)	IE (Mac)	Mozilla	Safari	Opera	W3CDOM
focusout	6	n/a	n/a	n/a	n/a	n/a
help[c]	4	5	n/a	n/a	n/a	n/a
keydown[c]	4	4	all	all	all	3
keypress[c]	4	4	all	all	all	3
keyup[c]	4	4	all	all	all	3
layoutcomplete	5.5	n/a	n/a	n/a	n/a	n/a
losecapture	5	n/a	n/a	n/a	n/a	n/a
mousedown[c]	4	4	all	all	all	2
mouseenter	5.5	n/a	n/a	n/a	n/a	n/a
mouseleave	5.5	n/a	n/a	n/a	n/a	n/a
mousemove[c]	4	4	all	all	all	2
mouseout[c]	3	3.01	all	all	all	2
mouseover[c]	3	3.01	all	all	all	2
mouseup[c]	4	4	all	all	all	2
mousewheel	6	n/a	n/a	n/a	n/a	n/a
move	5.5	n/a	n/a	n/a	n/a	n/a
moveend	5.5	n/a	n/a	n/a	n/a	n/a
movestart	5.5	n/a	n/a	n/a	n/a	n/a
paste	5	n/a	n/a	1.3/2	n/a	n/a
propertychange	5	n/a	n/a	n/a	n/a	n/a
readystatechange[e]	4	n/a	1.0.1	1.2	8	n/a
resize[c]	4	4	all	all	all	2
resizeend	5.5	n/a	n/a	n/a	n/a	n/a
resizestart	5.5	n/a	n/a	n/a	n/a	n/a
rowenter[d]	4	n/a	n/a	n/a	n/a	n/a
rowexit[d]	4	n/a	n/a	n/a	n/a	n/a
rowsdelete[d]	5	n/a	n/a	n/a	n/a	n/a
rowsinserted[d]	5	n/a	n/a	n/a	n/a	n/a
scroll[f]	4	4	n/a	n/a	n/a	2
selectstart	4	4	n/a	1.3/2	n/a	n/a

a 对于任意节点，只能通过 addEventListener() 方法来指定事件类型。

b 尽管可作为 IE 6+内的所有元素的属性，但只能在支持数据绑定的元素中使用，这些元素包括：a、bdo、button、custom、div、frame、iframe、img、input (checkbox、hidden、password、radio、text)、label、legend、marquee、rt、ruby、select、span 及 textarea。

c 仅在新浏览器中的元素对象中才能使用这些事件，而早期的实现方式与兼容事件将与适用的对象一起列举在本章的其他部分之中。

d 尽管可作为 IE 6+内所有元素的属性，但只能在支持数据绑定的元素中使用，例如 applet、object 和 xml 元素集。

e 尽管可作为 IE 6+内的所有元素的属性，但是，除非将一种 HTML 行为附加到元素之上，否则只能应用于 applet、document、frame、frameSet、iframe、img、link、object、script 及 xml 元素集。

f 尽管可作为 IE 6+内的所有元素的属性，但只能应用于 applet、bdo、body、custom、div、embed、map、marquee、object、table 及 textarea 元素和 window 对象。

316

317

ATTRIBUTE_NODE, CDATA_SECTION_NODE, COMMENT_NODE, DOCUMENT_FRAGMENT_NODE, DOCUMENT_NODE, DOCUMENT_TYPE_NODE, ELEMENT_NODE, ENTITY_NODE, ENTITY_REFERENCE_NODE, NOTATION_NODE, PROCESSING_INSTRUCTION_NODE, TEXT_NODE

IE *n/a* NN *n/a* Moz *all* Saf *all* Op *7* DOM *1*

只读

这类常量属于W3C DOM的根节点对象，因此它们也会被所有文档级的节点和元素所继承，但是在Safari和

Opera中，它们仅是Node对象的属性。这里的每个属性都对应于一个整数值，该值与所有DOM节点的nodeType属性有关。使用这些属性可以更为直观地在脚本中指明节点类型。

示例

```
if (myObject.nodeType == Node.ELEMENT_NODE) {
    // 按照元素进行处理
}
```

值

DOM节点类型的对应整数值如下所示：

属性	nodeType 值	属性	nodeType 值
ELEMENT_NODE	1	PROCESSING_INSTRUCTION_NODE	7
ATTRIBUTE_NODE	2	COMMENT_NODE	8
TEXT_NODE	3	DOCUMENT_NODE	9
CDATA_SECTION_NODE	4	DOCUMENT_TYPE_NODE	10
ENTITY_REFERENCE_NODE	5	DOCUMENT_FRAGMENT_NODE	11
ENTITY_NODE	6	NOTATION_NODE	12

默认值 常量值（如上表所示）。

DOCUMENT_POSITION_CONTAINED_BY, DOCUMENT_POSITION_CONTAINS, DOCUMENT_POSITION_DISCONNECTED, DOCUMENT_POSITION_FOLLOWING, DOCUMENT_POSITION_IMPLEMENTATION_SPECIFIC, DOCUMENT_POSITION_PRECEDING

IE *n/a* NN *n/a* Moz *1.7* Saf *n/a* Op *n/a* DOM *3*

只读

这类常量属于W3C DOM的根节点对象，因此它们也会被所有文档级的节点和元素（包括属性节点）所继承。以上的每个属性均对应于一个用于计算节点位置对比的位掩码值。在使用时，请参考共享的compareDocumentPosition()方法，以获得更多详细信息。另外，尽管早在1.4版开始，Mozilla就已经实现了部分常量，但自从1.7版之后，又修改了这些常量的名称。

示例

```
if (document.body.compareDocumentPosition(document.body.parentNode) ==
   (Node.DOCUMENT_POSITION_PRECEDING | Node.DOCUMENT_POSITION_CONTAINS)) {
   // 脚本代码
}
```

值

对应的整数值如下所示：

属性	掩码值	整数值
DOCUMENT_POSITION_CONTAINED_BY	0×10	16
DOCUMENT_POSITION_CONTAINS	0×08	8
DOCUMENT_POSITION_DISCONNECTED	0×01	1
DOCUMENT_POSITION_FOLLOWING	0×04	4
DOCUMENT_POSITION_IMPLEMENTATION_SPECIFIC	0×20	32
DOCUMENT_POSITION_PRECEDING	0×02	2

默认值 常量值（如上表所示）。

accessKey

IE 4 NN n/a Moz all Saf all Op 7 DOM 1

可读/可写

此属性是一个字符键，它可以让一个元素获得焦点（在一些浏览器中），也可以激活一个表单控件或链接动作。浏览器和操作系统会决定用户在按下快捷键时是否还须要按下其他组合键（例如，Ctrl、Alt或Command）以激活链接。大多数Windows浏览器须要用户同时按下Alt键和指定的快捷键，而Mac中的浏览器则使用Ctrl键。另外，在Opera中则使用完全不同的组合序列Shift+Esc+快捷键。

虽然这里列出的访问快捷键是一个被广泛共享的属性，但它还不能完全应用在所有的程序实现之中。注意，在W3C DOM、Mozilla、Safari和Opera中，此属性只能用于下列元素：a、area、button、input、label、legend及textarea。在这一列表中，IE 4增加了applet、body、div、embed、isindex、marquee、object、select、span、table和td元素，但是删除了label和legend元素。IE 5则加入了其他所有可显示元素，但同时也给出了一条警告。除input和其他表单相关的元素外，必须为IE 5及更高版本中的元素分配一个tabindex属性（即使是简单地将其值设置为0），这样访问快捷键组合才可以让元素获得焦点。其他所有主流浏览器均支持在select元素中使用这一属性。

示例 `document.links[3].accessKey = "n";`

值 任意一个键盘字符，包括字母、数字和标点。

默认值 空字符串。

all[]

IE 4 NN n/a Moz 1.7 Saf n/a Op 7 DOM n/a

只读

返回当前元素拥有的所有HTML元素对象所组成的数组。须要注意的是，Mozilla中只能在document对象中使用此属性，请参见document.all以获取更多信息。数组中的条目会根据源代码顺序以0为基建立索引。由于这种方式会超出内嵌元素的层次限制，因此通过document.all[]就能够暴露出整个文档中的所有元素。请参见all对象，以获取此属性值自己的一套属性与方法。

对于这个整体集合，既可以使用传统的JavaScript数组语法（形如“[索引值]”）也可以使用IE的JScript语法规则（形如“(索引值)”）来进行索引。须要注意的是，如果希望实现跨平台部署，使得在多个平台下均可使用这个对象集合，那么请使用方括号的语法形式。

除非相关应用完全针对IE浏览器器进行开发，否则均应使用document.getElementById()来引用W3C DOM对象，IE 5和后续版本及所有其他主流浏览器中都已实现了这一方法。

语法

```
object.all(index).objectPropertyOrMethod
object.all[index].objectPropertyOrMethod
object.all("elementID").objectPropertyOrMethod
object.all["elementID"].objectPropertyOrMethod
object.all.elementID.objectPropertyOrMethod
```

示例 `var inpVal = document.all.first_name.value;`

值 按照HTML源代码属性排列的元素对象引用数组。

默认值 由当前文档模型决定。

attributes[]

IE 5 NN n/a Moz all Saf all Op 7 DOM 1

只读

返回一个命名节点图（named node map）对象（W3C DOM类型为NamedNodeMap），它类似于一个属性对象（W3C DOM类型为Attr）数组，但区别在于，它还包含一些自有的方法使得访问该组成员更为便利。此属

320 性继承自Node对象，但是除元素外，在所有其他节点类型中其值均为“null”（元素是唯一一种拥有attributes的节点类型）。IE的attributes数组包含的条目适用于元素内部DTD中的所有属性，也适用于那些显式设置在HTML源代码中的自定义属性（需要IE 6或后续版本）。但使用脚本改变的元素属性或对应的属性值并不会反映在这个队列之中。

对非IE浏览器而言，attributes数组仅包含那些显式定义在HTML源代码中的条目，其中也包括自定义属性。使用脚本改变的元素属性（增加或减少）或对应的属性值可以反映在attribute对象中，并且可以通过attributes数组来进行引用。

使用标准的JavaScript数组语法可以代替命名节点图对象方法来访问单独的属性对象。但从总体上说，无论是通过元素对象属性的映射，还是通过元素的getAttribute()和setAttribute()方法，在访问HTML元素属性方面都不及JavaScript数组方便有效。如果要获取更多关于W3C DOM的细节信息（这些对XML文档解析很有助益），请参见与有关对象的属性和方法相关的Attr及NameNodeMap对象。

示例

```
var ldAttr = document.getElementById("myImg").attributes.getNamedItem("longdesc");
```

值 属性对象参考的集合（IE）或命名节点图（其他浏览器），该集合将根据源代码（Mozilla）顺序、名称字母顺序（IE/Mac）或随机顺序（IE/Windows）进行排序。

默认值 由当前元素模型决定。

baseURI

IE *n/a* NN *n/a* Moz *all* Saf *n/a* Op *n/a* DOM *3*

只读

baseURI是Node对象（由DOM 3推荐）的一个属性，它可以显示节点来自于哪个基准URI（源文件的全路径）。例如，document.implementation对象可以加载一个XML文档，而每个复制自该XML文档的节点均可以揭示其baseURI，而且这个URI地址可以与当前显示的HTML页面地址完全不同。

示例 `var nodeSrc = myXMLDoc.firstChild.childNodes[14].baseURI;`

值 完整的URI字符串。

默认值 由当前节点的内在值决定。

behaviorUrns[]

IE *5(Win)* NN *n/a* Moz *n/a* Saf *n/a* Op *n/a* DOM *n/a*

只读

它为所有的外部行为（*.htc*文件）提供了一个统一资源名称（Uniform Resource Name）数组，而这些外部行为则可以通过样式单语法与元素联系起来。可能出于安全考虑，这个数值的字符串条目都为空值。

321 **示例** `var htcCount = document.getElementById(elementID).behaviorUrns.length;`

值 （空）字符串数组。

默认值 长度为0的数组。

canHaveChildren

IE *5(win)* NN *n/a* Moz *n/a* Saf *n/a* Op *n/a* DOM *n/a*

只读

它指明了当前元素是否能够作为其他元素的容器。其属性值基于Windows系统下IE浏览器的内置HTML DTD，它定义了几种不能插入子节点的元素，例如br元素。

示例

```
if (elementRef.canHaveChildren) {
    // 插入或附加子元素的脚本指令
}
```

值 布尔值：true或false。

默认值 由元素特性决定。

canHaveHTML

IE *5.5* NN *n/a* Moz *n/a* Saf *n/a* Op *n/a* DOM *n/a*

只读或可读/可写

它指明了当前元素是否能够作为一种容器，以包含其他含有HTML标记的内容。对普通HTML元素而言，此属性与canHaveChildren表达的信息完全相同。另外，对IE HTML组件（定义在一个基于XML的*.htc*文件之中）而言，此属性是可读可写的，它还可以指导浏览器如何处理由组件定义的自定义元素。

示例

```
if (elementRef.canHaveHTML) {
    // 插入 HTML 信息的脚本指令
}
```

值 布尔值：true | false。

默认值 由元素特性决定。

childNodes[]

IE *5* NN *n/a* Moz *all* Saf *all* Op *7* DOM *1*

只读

这是W3C DOM中Node对象的一个属性，它由一个数组组成，而该数组则对应于节点层次中下一层所有子节点的引用集合。如果要获取更深层次的内嵌节点，那么必须从当前节点入手，读取每个子节点的childNodes数组。对节点树的遍历而言，这是一个非常重要的属性。有关这种数组的属性与方法的相关信息，请参见NodeList对象。

示例

```
for (var i = 0; i < nodeRef.childNodes.length; i++) {
    if (nodeRef.childNodes[i].nodeType == document.ELEMENT_NODE) {
        // 操作一个元素
    }
}
```

值 节点对象引用数组。

默认值 长度为0的数组。

children[]

IE *4* NN *n/a* Moz *n/a* Saf *1.2* Op *7* DOM *n/a*

只读

使用此属性可以返回当前元素拥有的所有第一层HTML元素对象组成的数组。这个集合与all[]集合不同，它仅包含当前元素的直接子对象，而all[]则包含各级子对象。例如，document.body.children[]可能仅包含一个表单，但并不包含内嵌在表单中的其他表单元素引用。数组中的条目会根据源代码顺序以0为基建立索引。另外，与childNodes[]数组相比，此属性的作用域是元素而不是节点。请参看children对象。在W3C DOM中，childNodes属性与此属性等价。

示例

```
for (var i = 0; i < elementRef.children.length; i++) {
    if (elementRef.children[i].tagName == "FORM") {
        // 操作一个表单元素
    }
}
```

值 元素对象引用数组。

默认值 长度为0的数组。

cite

IE 6 NN n/a Moz all Saf all Op 7 DOM 1

可读/可写

在IE 6或后续浏览器中，所有短语元素对象均共享此属性（与dateTime一起），但从官方角度来说，它事实上只属于blockquote、q、del和ins这几个元素对象（关于这些元素对应属性的详细信息，请参见第1章）。由于以上4个对象共享了此属性，因此微软公司发现将此属性应用于其他相关的HTML元素对象可能会带来更多的便利。当然，这也可能仅仅是一个误会。实质上，无论原因如何，都不要期待在新的IE版本中出现的元素对象都能支持这一属性。对于兼容性而言，其他浏览器和DOM只会在未来的元素对象中才会应用这一属性。

值 一个指向网络上某文档的有效URL链接，既可使用相对URL，也可以使用绝对URL地址。

默认值 空字符串。

className

IE 4 NN n/a Moz all Saf all Op 7 DOM 1

可读/可写

通常用于将元素和为某个类选择器设定的样式单规则关联起来。然后就可以使用脚本来改变与元素相关联的类。如果在文档中包含一个可供替换的类选择器和样式规则，那么调整元素的className属性可以为改变样式属性提供一种快捷而迅速的方法。

示例
```
document.getElementById("elementID").className = "altHighlighted";
```

值 该字符串区分大小写。请使用空格分隔多个类名称。

默认值 空字符串。

clientHeight, clientWidth

IE 4 NN n/a Moz 1.0.1 Saf all Op 7 DOM n/a

只读

概括地说，它们可以为元素内容提供高度和宽度，但在不同操作系统内的IE浏览器和DOCTYPE声明控制的各种兼容模式里，元素四周的填充往往有较小的变化，因此具体的高度和宽度还是会受到一定的影响。在Macintosh系统的IE浏览器中，这两个属性所有元素类型均无效。对Mozilla浏览器而言，如果元素内容超出了可视区域，这时属性值表示可视区域的尺寸，否则对应的属性值总是0。例如，对document.body元素而言，浏览器窗口的内容区域。请参见本章前述部分中“关于client-和offset-属性集”一节。

示例
```
var midHeight = document.body.clientHeight/2;
```

值 整数像素值。

默认值 0

clientLeft, clientTop

IE 4 NN n/a Moz n/a Saf n/a Op n/a DOM n/a

只读

概括地说，它们可以为元素内容及周围填充所处的方框提供左、上坐标，但在不同操作系统内的IE浏览器里，具体数值会有一定的变化。在Macintosh系统的IE浏览器中，这两个属性所有元素类型均无效。请参见本章前述部分中“关于client-和offset-属性集”一节。对于行内元素的定位，offsetLeft和offsetTop属性通常可以提供更为有用的信息。而对于那些使用CSS进行定位（包括位置改变）的元素而言，请使用style对象的属性，如left和top，以及pixelLeft和pixelTop（这两个仅适用于IE浏览器）。

值 整数像素值。

默认值 0

contentEditable

IE 5.5 NN n/a Moz n/a Saf 1.2 Op 9 DOM n/a

可读/可写

此属性可以决定元素是否可编辑，如果可编辑，那么用户就可以使用浏览器内置的实时内容编辑功能。如果

服务器不进行干预，那么用户所做出的修改并不会保留在服务器上，通常可以使用客户端脚本捕获更改的内容并通过表单或XMLHTTPRequest提交至服务器端。在编辑状态下，应该使用脚本改变元素的外形（如边框、背景颜色等），以便向用户强调这种模式。此时Safari会自动进行处理，对应元素边框将会出现蓝色光芒。默认情况下，该元素的所有子元素会继承其编辑模式设定。请参见第3章中的move事件，它提供了一个扩展示例。

示例 `document.getElementById("elementID").contentEditable = "true";`

值 字符串常量：`false | inherit | true`。

默认值 `inherit`

currentStyle

IE *5* NN *n/a* Moz *n/a* Saf *n/a* Op *9* DOM *n/a*

只读

返回一个style对象，其各个属性反映了当前元素所采用的实际属性值。这个属性还考虑到了样式单规则，这些样式单既可以是定义在style元素之中，也可以来自于外部样式单文件，还可以是内联的样式属性。由于style属性仅仅能够反映内联的样式属性，所以在文档加载之后，读取样式初始值时currentStyle属性更有价值。如果要修改样式属性，也可以使用元素的style对象属性。另外，如果要使用W3C DOM语法实现类似的功能，请参考window.getComputedStyle()方法。

示例 `var currSize = document.getElementById("elementID").currentStyle.fontSize;`

值 style对象引用。

默认值 实际的style对象。

dateTime

IE *6* NN *n/a* Moz *all* Saf *all* Op *7* DOM *1*

可读/可写

在IE 6或后续浏览器中，所有短语元素对象均共享此属性（与cite一起），但从官方角度来说，它事实上只属于del和ins这两个元素对象（关于这些元素对应属性的详细信息，请参见第1章）。由于以上4个对象共享了此属性，因此微软公司发现将此属性应用于其他相关的HTML元素对象可能会带来更多的便利。当然，这也可能仅仅是一个误会。实质上，无论原因如何，都不要期待在IE 6中会出现的元素对象都能支持这一属性。对于兼容性而言，其他浏览器和DOM只会在未来的元素对象中才会应用这一属性。

值 日期字符串。

默认值 空字符串。

dir

IE *5* NN *n/a* Moz *all* Saf *all* Op *7* DOM *1*

可读/可写

如果元素内的字符显示方向不受Unicode标准和浏览器默认语言系统的控制，那么此属性可以指明元素内文本内容的字符显示方向。决定文字绘制方向是从左到右，还是从右到左。

值 `ltr | rtl`，不区分字符大小写。

默认值 `ltr`

disabled

IE *4* NN *n/a* Moz *all* Saf *all* Op *7* DOM *2*

可读/可写

此属性指明对应元素是否能与用户进行交互。如果将此属性设置为true，那么该元素将无法接受焦点或无法被用户修改，并且它在页面上的外形也会被灰化。在IE 5.5及后续版本的浏览器内，此属性对所有的HTML

元素对象均有效。但对IE 4和IE 5而言，它仅能应用于表单控件。Mozilla、Safari和Opera则可在表单控件及style元素对象中识别此属性。对一个被禁用的表单控件而言，并不会随表单一起提交其名称/值对。

示例 `document.getElementById("myButton").disabled = true;`

值 布尔值：true | false。

默认值 false

document

IE *4* NN *n/a* Moz *n/a* Saf *1.2* Op *7* DOM *n/a*

只读

此属性会返回包含当前元素的document对象的引用。对于那些作用于对象引用（可通过事件属性或ID字符串获得）的方法而言，此属性可能有一定的帮助。ownerDocument是W3C DOM的对应属性。

示例
```
var currDoc = document.getElementById("elementID").document;
```

值 document对象的引用。

默认值 当前document对象。

filters[]

IE *4(win)* NN *n/a* Moz *n/a* Saf *n/a* Op *n/a* DOM *n/a*

只读

使用此属性可以返回当前元素拥有的所有filter对象所组成的数组。它只用应用于下列元素对象之中：bdo、body、button、div、fieldset、img、input、marquee、rt、ruby、span、table、td、textarea和th。请参考filter对象以获取引用语法。

值 filter对象引用数组。

默认值 长度为0的数组。

firstChild, lastChild

IE *5* NN *n/a* Moz *all* Saf *all* Op *7* DOM *1*

只读

此属性会返回当前元素节点的第一个或最后一个子节点的引用。在通常情况下，这些子节点往往是嵌入在元素内的文本节点。如果一个简单元素仅包含一个文本节点，那么这两个属性均会返回同一个文本节点的引用。对于更为复杂的情况，例如tr元素，通常可以包含多个元素节点（例如一些td元素）作为其子节点，但是某些浏览器可能会将元素之间的源代码回车符也转变成文本节点。因此，在使用这两个属性返回的引用之前，有必要确定节点类型。

示例
```
if (document.getElementById("elementID").firstChild.nodeType == 3) {
    // 按照文本节点进行处理
}
```

值 Node对象（包括文本节点、HTML元素节点等）引用。

默认值 null

hideFocus

IE *5.5* NN *n/a* Moz *n/a* Saf *n/a* Op *n/a* DOM *n/a*

可读/可写

使用此属性可以设定浏览器是否应该在聚焦元素外围显示一个虚线聚焦矩形框。如果默认设定为可聚焦或设定了tabindex属性，则该元素能够获得焦点。当此属性设置为“true”时，在聚焦状态时不会出现任何可视效果。

示例 `document.getElementById("elementID").hideFocus = true;`

值 布尔值：true | false。

默认值 false

id

IE 4 NN n/a Moz all Saf all Op 7 DOM 1

可读/可写

此属性可指定一个独一无二的标识符，使所属元素区别于文档中所有的其他元素。此属性值常用于获取元素的引用，但也可以遍历所有元素以检查是否存在一个匹配的id值。如果元素已经存在于文档树之中，那么改变其id属性值并不是一个明智的做法。但是如果通过脚本创建一个新的元素对象（例如，通过document.createElement()方法），则可以使用它为该对象的id属性设置一个唯一的标识符，然后再将该元素添加至文档树之中。

327

示例 `var headID = document.getElementsByTagName("head")[0].id;`

值 字符串。

默认值 空字符串。

innerHTML

IE 4 NN n/a Moz all Saf all Op 7 DOM n/a

只读或可读/可写

此属性可指出当前元素的开始和结束标签之间应显示的文本和HTML标签（例如，所有的源代码）。修改此属性值后，其中包含的HTML标签会通过HTML解析器得以显示，这些新值看起来就像是初始源代码的一部分。须要注意的是，应该在完全加载和显示文档树之后再改变此属性。但在查看浏览器中的源代码时，会发现innerHTML元素的相关修改并不会反映在对应的源代码中。对于IE浏览器中的col、colgroup、frameset、html、style、table、tbody、tfoot、thead、title和tr等元素对象，此属性为只读状态。

尽管W3C DOM并不支持此属性，但为了方便，所有的现代浏览器均支持innerHTML。如果为innerHTML属性指定一个不包含HTML标签的字符串，那么其效果就等同于为innerText属性（仅适用于IE浏览器）指定一个字符串。另外，在IE中还可以通过outerHTML属性读、写包含在元素标签之中的源代码。

示例 `document.getElementById("elementID").innerHTML = "How <i>now</i> brown cow?";`

值 字符串（可以不包含任何HTML标签）。

默认值 空字符串。

innerText

IE 4 NN n/a Moz n/a Saf all Op 7 DOM n/a

可读/可写

此属性指出元素内可显示的文本（不包含任何标签）。如果不但希望显示文本，而且还须要显示内嵌HTML标签，那么请参见innerHTML。并不会通过HTML解析器显示此属性值的修改内容，这意味着修改的内容中所包含的任何HTML标签只会作为普通的可显示文本进行处理。须要注意的是，应该在完全加载和显示文档树之后再改变此属性。但在查看浏览器中的源代码时，会发现innerText元素的相关修改并不会反映在对应的源代码中。在最新版的Mozilla和Opera浏览器中，等价的W3C DOM 3属性是textContent。

示例 `document.getElementById("elementID").innerText = "How now brown cow?";`

值 字符串。

默认值 空字符串。

isContentEditable

IE 5.5 NN n/a Moz n/a Saf 1.2 Op 9 DOM n/a

只读

328

它指明了当前元素是否能让用户进行编辑。对于实际的编辑状态，既可以在元素中显示设置（使用元素属性或脚本属性），也可以从父级树中继承。

示例

```
if (document.getElementById("elementID").isContentEditable) {
    // 处理可编辑元素
}
```

值 布尔值：true | false。

默认值 false

isDisabled

IE 5.5 NN n/a Moz n/a Saf n/a Op n/a DOM n/a

只读

它指明了当前元素是否处于禁用状态。对于实际的禁用状态，既可以在元素中显示设置（使用元素属性或脚本属性），也可以从父级树中继承。

示例

```
if (document.getElementById("elementID").isDisabled) {
    // 处理已禁用的元素
}
```

值 布尔值：true | false。

默认值 false

isMultiLine

IE 5.5 NN n/a Moz n/a Saf n/a Op n/a DOM n/a

只读

此属性指定了当前元素是否能将其内容扩展至多行之上。大多数文本容器均允许使用多行，但其他类型的元素往往并非如此，例如文本input元素就将内容显示限制在单行之内。

示例

```
if (document.getElementById("elementID").isMultiLine) {
    // 将该元素作为多行元素进行处理
}
```

值 布尔值：true | false。

默认值 由元素默认设定决定。

isTextEdit

IE 4 NN n/a Moz n/a Saf n/a Op n/a DOM n/a

只读

此属性指定了对应元素是否能够被用于创建一个IE/Windows的TextRange对象（通过createTextRange()方法）。目前只有body、button和文本输入时使用的input和textarea元素能够为其内容创建文本域。

示例

```
if (document.getElementById("elementID").isTextEdit) {
    var rng = document.getElementById("elementID").createTextRange();
}
```

值 布尔值：true | false。

默认值 由元素默认设定决定。

lang

IE 4 NN n/a Moz all Saf all Op 7 DOM 1

可读/可写

此属性指定了元素属性与属性值中所使用的语言种类。其他应用程序和搜索引擎也可以使用这些信息，以辅助进行拼写检查时字典的选择和索引的创建。

示例

```
document.getElementById("elementID").lang = "de";
```

值 不区分大小写的语言代码。
默认值 浏览器的默认值。

language

IE *4* NN *n/a* Moz *n/a* Saf *n/a* Op *n/a* DOM *n/a*

可读/可写

此属性指明了元素中所包含的脚本指令的脚本语言类型。

示例
```
document.getElementById("elementID").language = "vbscript";
```
值 不区分字符大小写的脚本语言名称字符串：javascript | jscript | vbs | vbscript。
默认值 jscript

lastChild

参见firstChild。

localName,namespaceURI,prefix

IE *4* NN *n/a* Moz *all* Saf *all* Op *7* DOM *2*

只读

这3个属性主要应用于XML文档元素，而这些元素一般是在XML命名空间的帮助之下进行定义。一个简单的文档范例如下所示：

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<results xmlns:libBook="http://catalog.umv.edu/schema">
<libBook:title libBook:rareBooks="true">De Principia</libBook:title>
</results>
```

这些属性能够反映元素命名特征的相关细节。其中，localName等价于元素的nodeName属性，它们都可以表示整个文档范围内的标签名，即使该标签名被另一个来自于其他命名空间的元素重用了也不会有影响。prefix则将元素与一个名称前缀联系起来，该名称通常由XML文档中一个容器的xmlns属性定义。这样就可以辅助脚本来识别与元素相关联的命名空间。而namespaceURI属性则可以反映更深层次的联系，它能返回容器元素中xmlns属性的URI字符串。尽管这三个属性均属于Node对象，但除了元素和属性节点之外的节点类型，它们的值均为null（但在Mozilla中其值为空字符串）。

示例
```
var allTitles = document.getElementsByTagName("title");
for (var i = 0; i < allTitles.length; i++) {
    if (allTitles[i].prefix == "libBook" &&
        allTitles[i].namespaceURI.indexOf("catalog.umv.edu") != -1) {
        // 在对应的命名空间中处理标题元素
    }
}
```
值 字符串。
默认值 对localName而言，默认值为元素标签名。对其他两个，均为空字符串。

namespaceURI

参见localNmae。

nextSibling, previousSibling

IE *5* NN *n/a* Moz *all* Saf *all* Op *7* DOM *1*

只读

它们将分别返回与当前节点处于文档树中同一内嵌层的下一个或前一个节点的引用。如果在属性名所指定的

位置上不存在节点，那么返回值为null。如果元素节点中仅包含一个孤立的文本节点，那么这两个值均会返回null。源代码属性将决定节点的初始排列顺序，但脚本可以改变文档树中的节点顺序，在这种情况下，这两个属性的返回值也会随之发生改变。

示例 `var nextNode = document.getElementById("elementID").nextSibling;`

值 Node对象（包括文本节点、HTML元素节点等）引用。

默认值 null

nodeName

IE 5 NN n/a Moz all Saf all Op 7 DOM 1

只读

该属性会返回一个表示节点名称的字符串，其返回值受节点类型的影响。对于元素和属性节点类型，此属性会分别返回标签名和属性名。而对于其他诸多不同类型的节点，一般都没有相关联的固有标题，此时nodeName属性会返回一个固定的字符串以指明其节点类型。例如，一个文本节点的返回值为“#text”，而根文档节点则会返回“#document”。对于元素，此属性会返回与元素对象的tagName属性值相同的字符串。值得注意的是，无论源代码的样式或DOCTYPE规范的具体类型，支持这一属性的浏览器均会以大写字母的形式返回元素标签字符串。

示例

```
if (document.getElementById("elementID").nextSibling.nodeName == "#text") {
   // 按照文本节点进行处理
}
```

值 对于节点，返回值为固定字符串，包括：#cdata-section、#document、#document-fragment和#text，而对于属性、元素和其他节点类型，将返回不同的字符串变量。

默认值 由节点特性决定。

nodeType

IE 5 NN n/a Moz all Saf all Op 7 DOM 1

只读

它会返回一个对应于节点类型的整数值，该节点类型由W3C DOM指定。如果要检查节点对象类型，那么这个属性是最好的一种选择（而不是nodeName属性值）。尽管每个节点类型都拥有一个对应的整数值，但对于支持nodeType属性的浏览器而言，并非每个浏览器都能支持全部的节点类型。这些整数值还拥有对应的常量字符串，这些常量看起来更为清晰详细，在处理节点类型时脚本代码也会比较利于阅读。须要注意的是，通过nodeType无法分辨元素类型（例如，根Element节点就对应于一个HTMLElement节点）。与此同时，IE 6和IE 7还会错误地将DOCTYPE元素当作注释节点类型。

示例

```
if (document.getElementById("elementID").firstChild.nodeType == 1) {
   // 按照元素进行处理
}
```

值

整数值，如下表所示：

值	节点类型	值	节点类型
1	ELEMENT_NODE	7	PROCESSING_INSTRUCTION_NODE
2	ATTRIBUTE_NODE	8	COMMENT_NODE
3	TEXT_NODE	9	DOCUMENT_NODE
4	CDATA_SECTION_NODE	10	DOCUMENT_TYPE_NODE
5	ENTITY_REFERENCE_NODE	11	DOCUMENT_FRAGMENT_NODE
6	ENTITY_NODE	12	NOTATION_NODE

默认值 由节点特性决定。

nodeValue

IE 5 NN n/a Moz all Saf all Op 7 DOM 1

可读/可写

尽管nodeValue属性属于所有的节点类型，但由于它可以为文本节点中的实际可绘内容提供读写功能，因此它对文本节点作用最大。假设脚本可以寻址到一个元素的firstChild节点的nodeValue属性，那么该属性就可以提供W3C DOM正则表达式来读取或修改元素中内嵌的文本节点。但对于元素节点，该属性只能返回null，因此不能将它视作innerText或innerHTML属性的简单替代物。如果将在一个属性节点上使用这个属性，那么它会返回该属性的属性值。请参考textContent属性。

示例
```
document.getElementById("elementID").firstChild.nodeValue = "New Text!";
```

值 字符串，但如果元素节点的值包含一个数字量，那么Windows下的IE浏览器会返回一个数字类型的值。但无论具体情况如何，均应该为此属性指定一个字符串。

默认值 空字符串。

offsetHeight, offsetWidth

IE 4 NN n/a Moz all Saf all Op 7 DOM n/a

只读

概括地说，它们可以为元素内容提供高度和宽度，但在不同操作系统内的IE浏览器和DOCTYPE声明控制的各种兼容模式里，元素的边框和四周的填充往往有较小的变化，因此具体的高度和宽度还是会受到一定的影响。目前倾向于在属性值中包含这些边框和填充值，但不考虑对应的边距。尽管它们并不是W3C DOM的一部分，但为了方便起见，目前所有的主流浏览器均实现了这两个属性。请参看本章前述部分中“关于client-和offset-属性集”一节。

示例
```
var midpoint = document.getElementById("elementID").offsetWidth/2;
```

值 整数像素值。

默认值 由元素特性决定。

offsetLeft, offsetTop

IE 4 NN n/a Moz all Saf all Op 7 DOM n/a

只读

概括地说，它们可以为元素所在方框提供左上角坐标值，但在不同操作系统内的IE浏览器和DOCTYPE声明控制的各种兼容模式里，由于坐标系统环境的不同（与offsetParent元素相比）具体值也会略有不同。尽管它们并不是W3C DOM的一部分，但为了方便起见，目前所有的主流浏览器均实现了这两个属性。请参看本章前述部分中“关于client-和offset-属性集”。对于已定位的元素，更需要元素的style属性来控制元素在文档或浏览器可视空间中的具体位置。

示例
```
if (document.getElementById("elementID").offsetLeft <= 20 &&
document.getElementById("elementID").offsetTop <=40) {
    ...
}
```

值 整数像素值。

默认值 由元素特性决定。

offsetParent

IE 4 NN n/a Moz all Saf all Op 7 DOM n/a

只读

此属性会返回一个对象引用，该对象包含当前元素的偏移定位环境。对于大多数IE页面内的元素，以及所有在

Mozilla、Safari和Opera页面内的元素，这个对象均是body对象。但是在IE中，被div元素包围的元素，或表格中的单元格则有其他类型的父级对象。另外，对于更为复杂的内嵌元素，会发现这个属性所返回的对象会根据浏览器版本的不同而偶有变化。例如，在IE 4中td元素的offsetParent属性是其临近的最外层tr元素，而在后续版本的IE浏览器及新的主流浏览器中，则会把它所在的table元素作为其偏移父级元素。请参看本章前述部分中“关于client-和offset-属性集”一节，在该节中阐述了一个使用此属性计算行内元素准确位置的范例。

示例 `var containerLeft = document.getElementById("elementID").offsetParent.offsetLeft;`

值 对象引用。

默认值 body对象。

outerHTML

IE 4 NN n/a Moz n/a Saf 1.3/2 Op 7 DOM n/a

可读/可写

此属性可指出当前元素的开始和结束标签之间应显示的文本和HTML标签（例如，所有的源代码）。如果仅需要已显示的文本内容，请参看outerText。另外，如果须要删除当前元素标签之外的源代码内容，请参看innerHTML。修改此属性值后，其中包含的HTML标签会通过HTML解析器得以显示，这些新值看起来就像是初始源代码的一部分。最好在文档树加载完成并得以显示后再修改此属性，此时甚至可以修改元素的类型或直接使用文本内容替换对应的元素。但在查看浏览器中的源代码时，会发现outerHTML元素的相关修改并不会反映在对应的源代码中。如果要为已存在的HTML添加内容，请参看insertAdjacentHTML()方法。W3C DOM中的等价方法则需要更多的节点对象操作，请参看在线参考IV。

示例

```
document.getElementById("elementID").outerHTML =
   "<acronym id="quotes">NI<i>M</i>BY</acronym>";
```

值 字符串（可以不包含任何HTML标签）。

默认值 空字符串。

outerText

IE 4 NN n/a Moz n/a Saf 1.3/2 Op 7 DOM n/a

可读/可写

此属性指出元素内可显示的文本（不包含任何标签）。如果要获取已显示的文本及元素的HTML标签，请参看outerHTML。并不会通过HTML解析器显示此属性值的修改内容，这意味着修改的内容中所包含的任何HTML标签仅会作为普通的可显示文本进行处理。须要注意的是，应该在文档被完全加载之后再改变此属性。但在查看浏览器中的源代码时，会发现outerText元素的相关修改并不会反映在对应的源代码中。

示例 `document.getElementById("elementID").outerText = "UNESCO";`

值 字符串。

默认值 空字符串。

ownerDocument

IE 5(Mac)/6(Win) NN n/a Moz all Saf all Op 7 DOM 1

只读

此属性会返回包含当前节点的document对象的引用。对于那些作用于对象引用（可通过事件属性或ID字符串获得）的方法而言，此属性可能有一定的帮助。此外，它也等同于IE中的document元素。

示例 `var currDoc = document.getElementById("elementID").ownerDocument;`

值 document对象的引用。

默认值 当前document对象。

parentElement

IE 4 NN n/a Moz n/a Saf 1.2 Op 7 DOM n/a

只读

此属性返回HTML包含层次中临近的最外层元素的引用。一个元素的父级元素对象并不一定同offsetParent属性返回的对象相同。对parentElement而言，它主要关注于源代码的包含关系。而临近的最外层元素会作为度量当前元素位置的坐标系统，这才是offsetParent属性返回的元素对象。例如，如果主文档中包含一个内嵌em元素的p元素，那么em元素就拥有两个父级元素对象。由于HTML源代码的包含关系，parentElement会将p元素作为其返回值，而考虑到坐标空间的包含关系，offsetParent则会返回body元素。

通过级联的parentElement属性就可以越过多个父级层次，如下所示：

```
document.getElementById("elementID").parentElement.parentElement;
```

这样一来，就可以使用对象引用来访问父级元素的属性或方法。

W3C DOM中的等价属性是parentNode。

示例 `document.getElementById("elementID").parentElement.style.fontSize = "14pt";`

值 元素对象的引用。

默认值 由元素特性决定。

parentNode

IE 5 NN n/a Moz all Saf all Op 7 DOM 1

只读

此属性会返回临近最外层节点（通常是一个元素）的一个引用，在文档树中，这个外层节点充当当前节点的容器。当前节点与其父节点之间只存在结构上的关系，而不涉及定位方面的关系。父节点是一个完全包裹住当前节点的节点，请不要将它与同辈节点混为一谈，同辈节点往往位于当前节点的两侧。此时，针对IE中parentElement元素的级联技巧也可以使用在这里。但由于这两个结果一个来自于以元素为中心的IE属性，另一个则来自于以W3C DOM节点为中心的属性，因此尽管目前最新版本的IE浏览器能够同时支持这两个视图，但这两个结果依然很难完全相等。

示例

```
if (document.getElementById("elementID").parentNode.nodeType == 1) {
    document.getElementById("elementID").parentNode.style.fontSize = "14pt";
}
```

值 元素对象的引用。

默认值 由节点特性决定。

parentTextEdit

IE 4 NN n/a Moz n/a Saf n/a Op n/a DOM n/a

只读

此属性会返回HTML包含层次中临近的最高层元素的一个引用，在这个类型的元素中允许创建一个Textrange对象。这个属性可能须要透过很多层才能找到一个合适的对象。在Macintosh系统中的IE浏览器中，由于缺乏对文本域的有效支持，此属性总是返回null。

示例

```
var rangeElement = document.getElementById("elementID").parentTextEdit;
var rng = rangeElement.createTextRange();
```

值 元素对象的引用。

默认值 body对象。

prefix

参见localName。

previousSibling

参见nextSibling。

readyState

IE 4 NN n/a Moz n/a Saf n/a Op n/a DOM n/a

只读

此属性将返回对象内容的当前下载状态。在加载文档的过程中，如果一段脚本（特别是由用户事件启动的脚本）能够执行某些动作，但在整个页面加载完成之前又必须避免其他动作，那么使用这个属性就可以获得一些与加载进程相关的中间信息。因此，可以在状态测试中使用这一属性值。这个属性的属性值会随着加载状态的改变而改变，而每次属性值改变均会激发一个readyStateChange事件。

在将这个属性引入IE 4时，它只对document、embed、img、link、object、script和style对象有效。而IE 5则将这个覆盖范围扩大至所有的HTML元素对象。

示例

```
if (document.readyState == "loading") {
   // "loading"状态下所采用的脚本指令
}
```

值 除object元素之外，所有的元素均可能返回以下字符串之一：uninitialized | loading | loaded | interactive | complete。有些元素可能会允许用户与部分元素内容进行交互，在这种情况下，此属性可能会返回interactive直到所有加载全部完成。但在加载的过程中，并不是所有的元素类型均会按顺序一次返回所有的属性值。比较特殊的是，object元素会根据这5个状态返回对应的数字值。初始状态是0，直到最终状态4。

默认值 无。

recordNumber

IE 4 NN n/a Moz n/a Saf n/a Op n/a DOM n/a

只读

在IE数据绑定时会使用此属性，它会返回一个整数值以表示数据集中一个元素的记录，例如，一个元素可根据该记录通过数据绑定填充其内容。此属性的属性值可以被用于从一个动态数据对象（Active Data Object，简称ADO）数据集中提取一条特定的记录。尽管此属性是为所有IE元素对象而定义的，但与数据绑定相关的其他属性则仅仅隶属于其中的一个元素子集。

示例

```
<script for="tableTemplate" event="onclick">
   myDataCollection.recordset.absoluteposition = this.recordNumber;
   ...
</script>
```

值 整数值。

默认值 null

repeatMax, repeatMin

IE n/a NN n/a Moz n/a Saf n/a Op 9 DOM n/a

可读/可写

这两个属性用于Web Forms 2.0，它们分别表示一个重复块中所允许的最大和最小数量的元素实例。它们实际上是repeat-max和repeat-min属性的脚本表示方式。如果指定了一个超出当前状态的新值（例如，设定一

个比当前重复项数量更小的最大值），那么浏览器会尝试着恢复到最好的状态（例如，禁用“添加”按钮），但浏览器并不会清除多出的重复项。请参见第1章中的repeat属性。

示例 `document.getElementById("order").repeatMax = 10;`

值 整数值。

默认值 repeatMax为JavaScript中的最大可用整数值，repeatMin为0。

repeatStart

IE *n/a* NN *n/a* Moz *n/a* Saf *n/a* Op *9* DOM *n/a*

可读/可写

此属性用于Web Forms 2.0，当浏览器将文档内的重复项实例集合在一起时，它可以使用一个整数表示表单控件元素名称所附加的起始索引编号。

示例 `document.getElementById("order").repeatStart = 1;`

值 整数值。

默认值 Opera 9使用0作为默认值，而Web Forms 2.0标准建议使用1作为默认值。

repetitionBlocks

IE *n/a* NN *n/a* Moz *n/a* Saf *n/a* Op *9* DOM *n/a*

只读

此属性使用在Web Forms 2.0中，它返回与重复模板相关的当前重复块数组（HTML元素对象）。例如，如果从同一个重复模板中生成三个表格行，那么作为模板的元素将包含一个拥有三个子条目的repetitionBlocks属性。每个条目（本例中就是一个tr元素对象）均有一个包含重复属性（repetitionIndex）值的重复块。读取这个数组的length属性后，就可以知晓用户或脚本已经添加了多少行。

示例

```
if (document.getElementById("order").repetitionBlocks > 10) {
    // 处理表单中重复次数大于 10 的重数块
}
```

值 数组。

默认值 长度为0的数组。

repetitionIndex

IE *n/a* NN *n/a* Moz *n/a* Saf *n/a* Op *9* DOM *n/a*

可读/可写

此属性使用在Web Forms 2.0中，它返回一个整数值以表示一个重复元素的重复属性。浏览器还会将这个值赋值给此属性，以便将它作为一个索引值来填充表单控件的索引值占位符。改变一个重复项中的repetitionIndex属性并不会通知表单控件，也不会改变与同一个模板相关的其他重复项的repetitionIndex属性。

示例

```
if (document.getElementById("order").repetitionBlocks > 10) {
    // 处理表单中重复次数大于 10 的重复块
}
```

值 数组。

默认值 长度为0的数组。

repetitionTemplate

IE *n/a* NN *n/a* Moz *n/a* Saf *n/a* Op *9* DOM *n/a*

只读

此属性使用在Web Forms 2.0中，针对当前重复元素的模板对象，它将返回对应HTML元素对象的引用。例如，

一个重复表格行的repetitionTemplate属性指向源代码中的tr元素，它就包含有关重复状态的信息。如果拥有一个来自repetitionBlocks属性的重复块引用，并且希望获取一个控制重复状态的元素对象引用，那么这个属性将大有用处。

示例

339

```
var templateRow = document.forms["orderform"].
elements["row0.quantity").parentNode.parentNode.repetitionTemplate;
```

值 HTML元素对象的引用。

默认值 无。

repetitionType

IE *n/a* NN *n/a* Moz *n/a* Saf *n/a* Op *9* DOM *n/a*

可读/可写

此属性使用在Web Forms 2.0中，它返回一个整数以指明该元素是否一个重复块、重复模板，或者表示元素没有任何重复特性。这个整数与一个已定义的RepetitionElement对象类型相对应，如下表所示：

代码	常量	代码	常量	代码	常量
0	REPETITION_NONE	1	REPETITION_TEMPLATE	2	REPETITION_BLOCK

示例

```
if (document.getElementById("orders").repetitionType == 1) {
   // 处理重复模板
}
```

值 整数值：0（无）、1（重复模板）或2（重复块）

默认值 0

runtimeStyle

IE *5(win)* NN *n/a* Moz *n/a* Saf *n/a* Op *n/a* DOM *n/a*

可读/可写

返回一个style对象，除非使用样式单显式地进行修改，否则该对象的样式属性一般不会发生改变。与currentStyle对象有所不同的是，系统的默认样式单属性并不会受到影响。也可以为这个runtimeStyle对象设置独立的样式属性，但这样做会超越（扰乱）普通的级联优先顺序。对于控制元素的样式属性而言，通过脚本设置的任意一个属性均会取代其他所有设定，其中还包括为元素标签指定的style属性。例如，如果将元素的style.color属性设置为red，并且将同一元素的runtimeStyle.color属性设置为green，那么尽管此时style.color的属性值仍然为red，但元素的文本将使用绿色进行显示。由于绿色为此时控制元素的有效样式，因此在这种情况下元素的currentStyle.color属性返回值为“green”。

通过为runtimeStyle.cssText属性重新指定一个CSS语法规则，就可以使用runtimeStyle对象来指定多个样式属性。如果为cssText属性指定一个空字符串来移除所有的属性值，那么就可以允许常规的样式单级联控制元素的有效样式。

340

示例

```
document.getElementById("elementID").runtimeStyle.cssText = "border: 5px blue solid";
```

值 style对象引用。

默认值 实际的style对象及其显式定义的样式属性值。

scopeName, tagUrn

IE *5(win)* NN *n/a* Moz *n/a* Saf *n/a* Op *n/a* DOM *n/a*

只读

对于使用XML命名空间的自定义元素而言，scopeName属性返回将名空间与标签名联系起来的标识符，而

xmlns属性将该名空间定义在文档中的任意位置。所有的简单HTML元素均会为此属性返回一个HTML值。而tagUrn属性则返回为名空间指定的URI地址。W3C DOM中的对应属性为prefix和namespaceURI。

示例

```
var allTitles = document.getElementsByTagName("title");
for (var i = 0; i < allTitles.length; i++) {
    if (allTitles[i].scopeName == "libBook" &&
        allTitles[i].tagUrn.indexOf("catalog.umv.edu") != -1) {
        // 在对应的命名空间中处理标题元素
    }
}
```

值 字符串。

默认值 scopeName的默认值为HTML，tagUrn的默认值为一个空字符串。

scrollHeight, scrollWidth

IE 4 NN n/a Moz 1.0.1 Saf all Op 7 DOM n/a

只读

它们最初出现于IE 4，应用在那些影响元素滚动的元素中，例如body、button、caption、div、fieldset、legend、marquee和textarea。这两个属性返回一个元素的像素尺寸，即使元素比浏览器窗口的可视空间还要大，也可以使用它们。与它们不同的是，可滚动元素的clientHeight和clientWidth属性只会返回元素可视部分的尺寸。但是在Macintosh系统下的IE中，对滚动属性的解释却与其他环境有所不同，此时会返回元素可视部分的尺寸。

从IE 5/Windows开始，所有的HTML元素均拥有这些属性，对于不可滚动的元素而言，其属性值与offsetHeight和offsetWidth一致。而Mozilla、Safari和Opera浏览器为所有的元素均实现了这些属性，在这些浏览器中，无论元素是否在显示空间中，均会返回元素的高度和宽度。比较重要的一点是，对于关键元素，如body，这些属性的含义不同，而且在跨操作系统的操作中，还会带来一些问题。

示例 `var midPoint = document.body.scrollHeight/2;`

值 任何大于或等于0的整数。

默认值 无。

scrollLeft, scrollTop

IE 4 NN n/a Moz 1.0.1 Saf all Op 7 DOM n/a

可读/可写

提供元素物理内容的实际左（上）边缘与内容可视部分左（上）边缘之间的像素距离。通过设置这两个属性，可以使用脚本调整可滚动容器中的滚动内容，例如，一个textarea元素中的文本，或者浏览器窗口、框架中的完整文档。如果对应的内容不需要滚动，这两个属性值均为0。将scrollTop属性值设置为15后，会使窗口中的文档向上滚动15像素，但是如果不明确设置，则scrollLeft属性不会受到影响。当用户调整滚动条时，属性值会随之变化。在Windows系统下的IE中，由于事件偏移测量的坐标系统与浏览器窗口中页面的可视区域有关，因此这种处理方式对一些事件驱动的定位任务很重要。此外，必须添加document.body滚动因子，以便由主体内容位置来调整事件坐标，请查看在线参考VI中的元素拖拽示例。从IE 5/Windows开始，scrollLeft和scrollTop适用于所有HTML元素对象，但对于不可滚动的元素，它们的属性值为0。

示例 `document.body.scrollTop = 40;`

值 任何大于或等于0的整数。

默认值 0

sourceIndex

IE 4 NN n/a Moz n/a Saf n/a Op 7 DOM n/a

只读

根据文档内所有元素的排列顺序，返回从零开始计数的元素序数值。浏览器会根据元素源代码顺序对它们进

行编号，对于第一个元素，其sourceIndex值为0。

示例 `var whichElement = document.getElementById("elementID").sourceIndex;`

值 任何大于或等于0的整数。

默认值 由元素特性决定。

style

IE 4 NN n/a Moz all Saf all Op 7 DOM 1

可读/可写

指明与元素相关的style对象，这种处理方式与明确在标签中为元素指定style属性值基本一致。在W3C DOM对象术语中，该对象的一个更为特殊的称呼是CSSStyleDeclaration。这个属性是读写元素样式单属性设置的入口点。如果要读取控制元素的有效样式单属性（包括已引入的样式单属性），请查看本章中的currentStyle属性（针对IE），以及AbstractView.getComputedStyle()方法（针对W3C DOM浏览器）。

示例 `document.getElementById("elementID").style.fontSize = "14pt";`

值 style对象。

默认值 无。

tabIndex

IE 4 NN n/a Moz all Saf all Op 7 DOM 1

可读/可写

此属性为一个整数值，表示此元素在文档内所有可聚焦元素中进行跳格（tab）选择时的序列号。跳格选择的顺序依照一个严格的规则。当用户在一个页面内进行跳格选择时，那些tabIndex属性值不为0的元素将排列在前面。无论元素在页面或文档中所处的物理位置在哪，总是从tabIndex属性值最小的元素开始聚焦，然后随着该属性值的增长依次获得焦点。如果两个元素的tabIndex属性值相同，则较早出现在文档中的元素能够首先获得焦点。此后，那些未设置tabIndex属性或其属性值为0的元素才能获得焦点。对这些元素而言，它们获得焦点的次序与其在文档中出现的先后顺序相同。

W3C DOM和大多数当代浏览器将tabIndex的应用限制于a、area、button、input、object、select和textarea等元素对象。另外，IE 4添加了applet、body、div、embed、isindex、marquee、span、table和td。而IE 5则能支持所有其他的可渲染元素。在IE浏览器和Mozilla 1.8版及更新版本中，如果将tabindex属性值设置为负数，则该元素将会从跳格选择序列中完全去除。

示例 `document.getElementById("link3").tabIndex = 6;`

值 整数值。

默认值 0

tabName

IE 4 NN n/a Moz all Saf all Op 7 DOM 1

只读

返回当前元素的标签名。为了便于字符串比对，无论源代码中标签名的样式或DOCTYPE声明是怎样的，通常均会使用小写字母的形式返回标签名。

示例 `var theTag = document.getElementById("elementID").tagName;`

值 字符串。

默认值 由元素特性决定。

tagUrn

参见scopeName。

textContent

IE *n/a* NN *n/a* Moz *1.7* Saf *n/a* Op *9* DOM *3* 343

可读/可写

反映一个节点的文本字符串，其中还包括元素内的组合字符串节点。例如，如果一个p元素包含一个内嵌的em元素，则该p元素的textContent属性包含这两个元素的所有文本，看起来就像em元素并未出现过一样。如果为此属性指定一个字符串，那么元素内的所有内嵌内容将被新文本所期待，形成一个文本节点。就这一点而言，可以将W3C DOM的textContent属性看作是IE内innerText属性的等价物。

示例 `document.getElementById("elementID").textContent = "This is some new text.";`

值 字符串。

默认值 空字符串。

title

IE *4* NN *n/a* Moz *all* Saf *all* Op *7* DOM *1*

可读/可写

提供一个元素的提示信息。如果对应元素将显示在页面之上，那么当光标在元素上方停留片刻时，浏览器会将此属性提供的信息以悬浮文本提示的形式显示出来。但这个悬浮提示所使用的字体、尺寸及颜色并不受脚本控制。

示例 `document.getElementById("elementID").title = "Hot stuff!";`

值 字符串。

默认值 空字符串。

uniqueID

IE *5(Win)* NN *n/a* Moz *n/a* Saf *n/a* Op *n/a* DOM *n/a*

只读

返回一个与页面上所有其他对象标识符均不同的标识符字符串。如果并不介意浏览器使用其命名机制对标识符进行命名，那么可以使用这种方式为一个新建的元素指定标识符。它通常作为document对象的一个元素来使用，但通过一个已存在的元素对象引用也可以对这个属性进行访问。当某个方法需要它作为元素ID的字符串参数时，这个标识符也是完全适用的。

示例

```
var newElem = document.createElement("p");
newElem.id = document.uniqueID;
```

值 字符串。

默认值 由浏览器生成。

unselectable

IE *5.5* NN *n/a* Moz *n/a* Saf *n/a* Op *9* DOM *n/a* 344

可读/可写

控制用户是否能够选择当前元素的内容。子元素并不一定继承这个设定。

示例 `document.getElementById("elementID").unselectable = "on";`

值 字符串常量：on | off。

默认值 空字符串。

addBehavior( )

IE *5(Win)* NN *n/a* Moz *n/a* Saf *n/a* Op *n/a* DOM *n/a*

左对齐

为当前元素附加一个内部或外部IE行为。一旦通过脚本附加了行为，那么元素将对该行为所定义的事件进行响应，并且为该行为相关的属性和方法提供接口。须要注意的是，必须使用与当前页面相同的域和协议提供

外部行为文件。如须获取更多适用于IE/Windows的行为，请访问*http://msdn.microsoft.com/en-us/library/ms531078.aspx*。

返回值 可作为removeBehavior()方法参数的整形序列号。

参数

URL

对于外部行为，参数为一个指向服务器内.*htc*文件的相对或绝对URL。对于内部行为，参数为下文中描述的特定格式。

#default#behaviorName

此处，*behaviorName*是下列内置行为之一：anchorClick | anim | clientCaps | download | homePage | httpFolder、mediaBar | saveFavorite | saveHistory | saveSnapshot | userData。

addEventListener( )

IE *n/a* NN *n/a* Moz *all* Saf *all* Op *7* DOM *2*

```
addEventListener("eventType", listenerFunction, useCapture)
```

此方法将一个事件处理方法绑定到当前节点，这样一来，无论是将节点作为事件目标，还是在事件传播过程中，一旦某个特定类型的事件到达节点就会执行已绑定的方法。须要注意的是，W3C DOM事件会通过文本节点及元素节点传播。根据方法中第3个布尔参数的设置，节点可以通过事件捕获或事件上升传播监听事件。可以在同一个节点上多次调用这个方法，通过不同的参数值就可以根据需要指定很多事件处理行为，但是对于同一个事件和传播类型只会调用一个监听方法。如果事件监听器被添加到一个临时基类之上，那么应该使用removeEventListener()方法进行移除。

返回值 无

参数

eventType

浏览器对象模型中一个已知事件类型的字符串（不含“on”前缀）。W3C DOM能够理解的事件类型如：abort、blur、change、click、DOMActivate、DOMAttrModified、DOMCharacterDataModified、DOMFocusIn、DOMFocusOut、DOMNodeInserted、DOMNodeInsertedIntoDocument、DOMNodeRemoved、DOMNodeRemovedFromDocument、DOMSubtreeModified、error、focus、load、mousedown、mousemove、mouseout、mouseover、mouseup、reset、resize、scroll、select、submit、unload。

listenerFunction

方法引用，当节点在特定的传播模式中监听到事件类型时，将执行该方法。由于这是一个方法对象的引用，因此不要将方法名放置在引号中，也不要包含括号。匿名方法引用也适用于此处。在执行过程中，浏览器会自动将当前的事件对象作为一个参数传递到监听程序。

useCapture

一个布尔值。如果为true，那么只有当节点朝着目标节点（即事件捕获节点）传播时节点才会监听事件类型。如果为false，那么只有当事件从事件目标向上传播时，节点才会进行监听。如果当前节点即为事件的目标节点，那么这两个布尔值均可使用。

addRepetitionBlock( )

IE *n/a* NN *n/a* Moz *n/a* Saf *n/a* Op *9* DOM *n/a*

```
addRepetitionBlock(blockReferenceOrNull)
```

假设当前元素为一个重复模板，此方法会在与当前元素相关的一组分块中插入一个新的重复块。调用此方法等价于用户直接点击一个添加类型的输入元素。

返回值 被插入的元素的引用。

参数

blockReferenceOrNull

如果将“null”作为参数（并非参数为空），那么新元素将作为最后一个重复块附加至分组中。另外，也可以将分组中某个特定重复块的引用（可从repetitionBlocks数组中获取）作为参数进行传递，那么新的块将插入到该重复块之后。

addRepetitionBlockByIndex( )

IE *n/a* NN *n/a* Moz *n/a* Saf *n/a* Op *9* DOM *n/a*


```
addRepetitionBlockByIndex(blockReferenceOrNull, newRepeatIndex)
```

假设当前元素为一个重复模板，此方法会向与当前元素相关的一组分块中插入一个新的重复块。调用此方法等价于用户直接点击一个“添加”类型的输入元素。通过第2个参数，还可以为最新插入的重复块指定一个自定义的索引号。

返回值 被插入的元素的引用。

参数

blockReferenceOrNull

如果将“null”作为参数（并非参数为空），那么新元素将作为最后一个重复块附加至分组中。另外，也可以将分组中某个特定重复块的引用（可从repetitionBlocks数组中获取）作为参数进行传递，那么新的块将插入到该重复块之后。

newRepeatIndex

整数值。

appendChild( )

IE *5* NN *n/a* Moz *all* Saf *all* Op *7* DOM *1*

```
appendChild(nodeObject)
```

在当前节点对象的最后一个子节点之后插入一个新的节点。当前节点对象必须能够包含子节点，否则此方法会抛出一个异常。这是添加动态创建对象的最常用的一种方法，这些动态对象包括元素、文本节点，以及一个已存在元素或节点的文档框架，例如，可以使用一段脚本为一个文档增加一大块新内容。但是，如果appendChild()的参数指向一个已经在文档树中存在的节点，那么首先将从树中移除节点，然后再将它添加至当前对象子节点列表中的最后一位。如果须要将一个节点从原有位置移至容器末尾，那么这实际上是一种非常快捷的方法。

另外，为一个已存在的文本节点添加一个同辈文本节点，并不会将它们结合在一起。如果要将所有的同辈文本节点整合成一个大的文本节点，可以调用其父对象的normalize()方法。

返回值 被添加节点的引用。

参数

nodeObject

任意节点对象的引用，只要它们能够成为当前对象的子节点。该对象可能来自于动态生成的内容，也可能是现有文档树中的一个节点。

applyElement( )

IE *5(win)* NN *n/a* Moz *n/a* Saf *n/a* Op *n/a* DOM *n/a*

```
applyElement(elementObject[, type])
```

根据方法的第2个参数值的不同，此方法将插入一个新的元素，以作为当前对象的子元素，或作为其新的父对象。默认的做法是将当前对象包含在新元素内。当然，通过不同的参数，也可以将插入的新元素作为一个子元素。在这种情况下，如果当前对象位于文档树之中，并且已经内嵌了一些子元素，那么插入新元素后，

原有的子对象将成为这个元素的子对象，即成为当前对象的孙级对象。在IE元素的插入方法中，这种包含做法非常独特，它对文档树的影响也非常大。因此，使用时请倍加小心。

返回值 新增加的元素对象的引用。

参数

elementObject

任何一个动态生成的对象的引用，或文档树中已存在的元素对象的引用。

type

可选用的字符串值，包括：inside（新元素成为底部对象，即当前对象的第1个子节点）；outside（新元素成为当前对象的父节点）。默认值为outside。

attachEvent( )

IE *5(win)* NN *n/a* Moz *n/a* Saf *n/a* Op *7* DOM *n/a*

```
attachEvent("eventName", functionReference)
```

为一个元素对象绑定一个事件处理程序，以处理一种特定类型的事件。它的处理方式与W3C DOM的addEventListener()方法比较类似。如果使用attachEvent()方法绑定事件处理程序，那么应该使用detachEvent()方法解除绑定。

返回值 如果绑定成功，则返回布尔值true。

参数

eventName

字符串形式的事件名称，其中包括“on”前缀。尽管此字符串并不区分大小写，但推荐使用小写字母。

functionReference

一个方法的引用，无论是元素作为事件目标，还是在事件广播过程中，一旦元素接收到事件，都将执行该方法。由于这是一个方法对象的引用，因此不要将方法名放置在引号中，也不要包含括号。此时，没有任何参数将传递至此函数之中。

blur( )

IE *3* NN *2* Moz *all* Saf *all* Op *7* DOM *1*

当对象的blur事件激发时，去掉当前对像上的焦点。须要注意的是，除较新版本的IE浏览器之外，几乎所有的浏览器均对能够使用聚焦和模糊（包括事件和方法）的元素范围有所限制，请参见第1章中的tabindex属性。因此，为了尽可能地保证后向兼容性，请将blur()方法应用于可聚焦元素，诸如input和textarea元素。例如，可为input元素指定onfocus="this.blur();"属性，这是一种自然而有效的方法，可用来禁用一大片不提供元素禁用的区域，并且保证后向兼容性。

在同一个页面上，请适度使用blur()和focus()方法。有时，如果在处理时调用了警告对话框，在不经意之间可能会陷入无尽循环的“模糊化”和“聚焦”轮回之中，因此需要特别注意。另外，还须要了解的是，如果在一个对象上调用blur()方法，那么某些其他对象（可能是window对象）也有可能接收到focus事件。

返回值 无。

参数 无。

clearAttributes

IE *5(win)* NN *n/a* Moz *n/a* Saf *n/a* Op *n/a* DOM *n/a*

从当前元素内去掉除id和name属性之外的所有属性。对于那些受元素属性影响的显式特性而言，一旦去掉了对应属性，它们也不会再作用于元素。

返回值 无。

参数 无。

click( )

IE 4 NN 2 Moz all Saf all Op 7 DOM 1

在元素上模拟用户的点击操作，并且激发一个点击事件。但是对于所有能够在真实点击时改变外观状态的元素，不要期待它们也会在脚本模拟点击的过程产生同样的效果。例如，在某些Macintosh浏览器版本，当使用脚本在复选框元素上调用click()方法时，它们无法改变复选框的选择状态。在这种情况下，如果要调用click()方法，那么不仅须要点击事件处理程序执行某些代码，还须要为复选框的checked属性设置合适属性值。

返回值 无。

参数 无。

cloneNode( )

IE 5 NN n/a Moz all Saf all Op 7 DOM 1

```
cloneNode(deepBoolean)
```

将当前节点复制至内存，并返回节点副本的引用（它不是节点树的一部分）。由于该副本是一个完整节点，因此在向文档树中插入节点之前，可以在副本上执行相关的操作。然而须要注意的是，副本元素的id属性与原节点相同。因此在将副本插入文档树之前，须要改变这些元素的id属性。

方法中的布尔参数决定副本在包含当前节点的同时，是否还包含所有内嵌节点。因此，如果复制一个简单的元素容器并且将此参数设置为false，那么元素内的文本节点并不会成为副本的一部分。

返回值 内存中文档片段的引用。

参数

deepBoolean

布尔值，控制副本是否包含所有的内嵌节点（true）或仅包含节点本身（false）。在IE中此参数是可选参数，其默认值为false。

compareDocumentPosition( )

IE n/a NN n/a Moz 1.4 Saf n/a Op n/a DOM 3

```
compareDocumentPosition(nodeReference)
```

返回一个整数，表示与调用此方法的元素相比，*nodeReference*参数在节点树位置上的相对位置。由于此方法属于节点对象，因此包括HTML元素节点和属性节点在内的所有类型的节点均会继承此方法。

根据下文所示的6个不同状态，返回值由其十六进制位掩码值计算决定。假定表达式为NodeA.compareDocumentPosition(NodeB)，则六组值、常量名和对应的含义如下：

值	常量	描述
0x01	DOCUMENT_POSITION_DISCONNECTED	两个节点不相连
0x02	DOCUMENT_POSITION_PRECEDING	节点B位于节点A之前
0x04	DOCUMENT_POSITION_FOLLOWING	节点B位于节点A之后
0x08	DOCUMENT_POSITION_CONTAINS	节点B包含节点A(因此节点B位于节点A之前)
0x10	DOCUMENT_POSITION_CONTAINED_BY	节点A包含节点B(因此节点B位于节点A之后)
0x20	DOCUMENT_POSITION_IMPLEMENTATION_SPECIFIC	浏览器决定相对位置

由于一次比较可能复合多个状态，因此此处使用位掩码。例如，考虑如下表达式：

```
document.body.compareDocumentPosition(document.body.parentNode)
```

在一个典型的HTML文档中，body的父节点为html元素。由于html元素位于body元素之前，因此得到掩码0x02，而html元素又包含body元素，又可得掩码0x08，因此该表达式的返回值为10，还可以使用常量值表示两个状态，此时常量值之间使用或运算符（|）进行分隔，如下所示：

```
if (document.body.compareDocumentPosition(document.body.parentNode) ==
    (Node.DOCUMENT_POSITION_PRECEDING | Node.DOCUMENT_POSITION_CONTAINS)) {
    // 脚本指令
}
```

当用于比较的两个引用指向同一个节点时，此方法返回0。

返回值 整数值。

参数

nodeReference

当前文档或外部的任意节点的引用。

componentFromPoint( )

IE *5(Win)* NN *n/a* Moz *n/a* Saf *n/a* Op *n/a* DOM *n/a*

```
componentFromPoint(x, y)
```

返回一个字符串，表示坐标点在元素中的位置。对于显示滚动条的元素而言，返回值可以精确地表示滚动条的哪一部分在坐标区域内。如果使用微软的文档编辑模式，则附加块（如可拖拽的尺寸控制柄）也由返回值表示。对于未显示滚动条或编辑控制柄的元素区域，可以判断对应的坐标是位于元素之内还是之外，这样就可以简便快捷地应用于事件坐标与元素位置之间的碰撞检测。

event对象是坐标参数值最常见的来源，明确地说，就是event.clientX和event.clientY属性。可直接使用这些参数值，如下所示：

```
var where = event.srcElement.componentFromPoint(event.clientX, event.clientY);
```

返回值

下表中的任意一个字符串：

返回的字符串	描述
空字符串	在元素内容区域之内
outside	在元素内容区域之外
handleBottom	编辑样式尺寸调整控制柄在底部
handleBottomLeft	编辑样式尺寸调整控制柄在左下部
handleBottomRight	编辑样式尺寸调整控制柄在右下部
handleLeft	编辑样式尺寸调整控制柄在左侧
handleRight	编辑样式尺寸调整控制柄在右侧
handleTop	编辑样式尺寸调整控制柄在顶部
handleTopLeft	编辑样式尺寸调整控制柄在左上部
handleTopRight	编辑样式尺寸调整控制柄在右上部
scrollbarDown	滚动条的向下箭头
scrollbarHThumb	滚动条的水平滑块控件
scrollbarLeft	滚动条的左箭头
scrollbarPageDown	滚动条的向下翻页区域
scrollbarPageLeft	滚动条的向左翻页区域
scrollbarPageRight	滚动条的向右翻页区域
scrollbarPageUp	滚动条的向上翻页区域
scrollbarRight	滚动条的向右箭头
scrollbarUp	滚动条的向上箭头
scrollbarVThumb	滚动条的竖直滑块控件

参数

x

与屏幕上端相对的正或负像素值。

y

与屏幕左边缘相对的正或负像素值。

contains( )

IE 4 NN n/a Moz n/a Saf n/a Op 7 DOM n/a

```
contains(elementReference)
```

返回的布尔值表示当前元素是否包含指定的元素。

返回值 布尔值：`true | false`。

参数

elementReference

一个完整的元素对象参考，如`document.getElementById("myDIV")`。

createControlRange( )

IE 5(Win) NN n/a Moz n/a Saf n/a Op n/a DOM n/a

尽管该方法已在很多HTML元素对象中得以实现，但它只应该与选择对象一起使用。参见选择对象的`createControlRange()`以获取更多详细信息。

detachEvent( )

IE 5(Win) NN n/a Moz n/a Saf n/a Op n/a DOM n/a

```
detachEvent("eventName", functionReference)
```

如果元素对象已经绑定了针对某个特定事件类型的处理方法，那么此方法可以移除绑定方法。它的处理方式与W3C DOM的`removeEventListener()`方法比较类似。`detachEvent()`方法的等价事件属性会将事件属性设置为`null`。

返回值 无。

参数

eventName

字符串形式的事件名称，其中包括“on”前缀。可用的事件类型包含IE/Windows共享事件类型列表中的任意一个事件。尽管此字符串并不区分大小写，但推荐使用小写字母。

functionReference

一个方法的引用，无论是元素作为事件目标，还是在事件广播过程中，一旦元素接收到事件，就将执行该方法。由于这是一个方法对象的引用，因此不要将方法名放置在引号中，也不要包含括号。

dispatchEvent( )

IE n/a NN n/a Moz all Saf all Op 7 DOM 2


```
dispatchEvent(eventObjectReference)
```

指向当前节点激发的一个事件。此方法主要使用在通过脚本人工创建事件的情况下，调用此方法后会将事件发送至一个节点，从而执行其事件监听方法。作为参数传递的事件对象必须首先指定一个事件类型，但也可能在事件对象创建初始化的时候已经设置了该对象的其他属性。例如，下述这个脚本片段就创建了一个通配的鼠标事件，然后将其初始化为一个`mousedown`类型，即可以向上传递并可以被取消，然后再将这个事件发送至一个ID为“myNode”的元素：

```
var newEvt = document.createEvent("MouseEvents");
newEvt.initEvent("mousedown", true, true);
document.getElementById("myNode").dispatchEvent(newEvt);
```

参见W3C DOM的Event、MouseEvent和UIEvent对象以获取更多详细信息。IE/Windows中的对应方法是fireEvent()。

返回值 如果响应dispatchEvent()方法的任意事件监听方法还会调用event.preventDefault()方法，那么W3C DOM规范会使此方法返回的布尔值为false。否则此方法返回true。

参数

eventObjectReference

一个事件对象的引用。通常使用脚本指令创建并初始化这个对象。

doScroll( )

IE *5(Win)* NN *n/a* Moz *n/a* Saf *n/a* Op *n/a* DOM *n/a*

```
doScroll(["scrollAction"])
```

控制能够显示滚动条的任意元素的滚动情况。由于大多数HTML元素能够使用样式单来固定高度和宽度，如果将overflow样式属性设置为scroll，则doScroll()方法就能适用于这些元素。

与直接滚动至某个坐标位置不同的是，根据相关的参数设置，doScroll()方法能够在滚动条控件或区域上模拟点击。每次调用这个方法均会为此元素触发onscroll事件。如果脚本代码带来的滚动还会引起页面重新排版，那么在setTimeout()调用的独立的方法中应该使用此方法。

返回值 无。

参数

scrollAction

一个滚动条区域的字符串名称。如果忽略这个参数，则默认的参数值为scrollbarDown。大多数区域拥有可互换的长、短两种名称，如下所示：

scrollbarDown（或down）
scrollbarHThumb
scrollbarLeft（或left）
scrollbarPageDown（或pageDown）
scrollbarPageLeft（或pageLeft）
scrollbarPageRight（或pageRight）
scrollbarPageUp（或up）
scrollbarVThumb

dragDrop( )

IE *5.5* NN *n/a* Moz *n/a* Saf *n/a* Op *n/a* DOM *n/a*

为当前元素触发一个ondragstart事件，在用户真正开始拖拽元素之前，它允许一个鼠标事件处理方法启动一段与拖拽事件相关的脚本过程。在实际的用户拖拽操作完成后，当用户释放鼠标按键时返回一个布尔变量true。

返回值 布尔值：true | false。

参数 无。

fireEvent( )

IE *5.5* NN *n/a* Moz *n/a* Saf *n/a* Op *n/a* DOM n/a

```
fireEvent("eventType"[, eventObjectReference])
```

指向当前元素激发的一个事件。它主要使用在通过脚本人工创建事件的情况下，调用此方法后会将事件发送至一个元素，从而执行其事件处理方法。也可以根据需要发送任意类型的事件。对这个简单的事件对象，会自动为其4个属性赋值，如下所示：

```
cancelBubble = false;
returnValue = true;
srcElement = reference-to-current-element;
type = event-type-specified-as-the-parameter;
```

另外，也可以传递一个包含更多详细信息的event对象，这些详细信息包括事件位置等。下述的脚本片段创建了一个通配的event对象，然后为它指定了某些属性，最好将这个事件发送至一个ID为“myElem”的元素：

```
var newEvt = document.createEventObject();
newEvt.clientX = 50;
newEvt.clientY = 300;
newEvt.cancelBubble = true;
document.getElementById("myElem").fireEvent("onclick", newEvt);
```

请参见IE的event对象以获取更多详细信息。对应的W3C DOM方法是dispatchEvent()。

返回值 布尔值，true | false。指明事件是否触发成功。

参数

eventType

带有“on”前缀的事件名称字符串，例如“onmousedown”。

eventObjectReference

一个可选的事件对象引用，该对象通常会被重建。

focus( )

IE *3* NN *2* Moz *all* Saf *all* Op *7* DOM *1*

当对象的focus事件激发时，此方法可为当前对像赋予焦点。为了尽可能地保证后向兼容性，请将focus()方法应用于可聚焦元素，诸如input和textarea元素。

如果须要为文本框赋予焦点，并且选择框内的所有文本，请在此元素上依次使用focus()和select()方法。另外，如果是在窗口改变之后（例如，在一个警告对话框关闭之后）才让以上功能执行，那么应该将以上的方法放置在一个单独的方法中，然后在alert()方法显示对话框之后，由setTimeout()方法调用这个单独的方法。这样一来，就可以让IE/Windows正确地按照顺序执行指令。

返回值 无。

参数 无。

getAdjacentText( )

IE *5(Win)* NN *n/a* Moz *n/a* Saf *n/a* Op *n/a* DOM *n/a*

```
getAdjacentText("where")
```

根据4种不同的参数值所指明的方向，返回当前元素之内和四周的文本（不包括HTML标签和属性）。在出现临近元素的开始或结束标签之前，所有的文本段均视为元素四周的文本。请考虑如下HTML代码：

```
<p>This is a very <span id="mySpan">short</span> paragraph.</p>
```

然后就可以在span元素上调用getAdjacentText()方法，并分别采用4种不同的参数值，如下所示：

```
document.getElementById("mySpan").getAdjacentText("beforeBegin")
    // 返回: "This is a very"
document.getElementById("mySpan").getAdjacentText("afterBegin")
    // 返回: "short"
document.getElementById("mySpan").getAdjacentText("beforeEnd")
    // 返回: "short"
document.getElementById("mySpan").getAdjacentText("afterEnd")
    // 返回: "paragraph."
```

在这种情况下，由于span元素内未包含其他元素，因此使用“afterBegin”和“beforeEnd”参数时返回值相同。如果在外部的p元素上调用此方法，并且使用“afterBegin”作为参数，则返回值为span元素起始标签之前的文本。在某些文档树结构中，这个方法的返回值与W3C DOM的子节点属性nodeValue等价。

返回值 字符串，根据元素内部文本块的结构，可能还会包含头部和尾部的空格。

参数

where

4个常量字符串之一，不区分大小写。

常量	描述
beforeBegin	前一个起始或结束标签与当前元素起始标签之间的文本
afterBegin	当前元素起始标签和下一个起始或结束标签之间的文本
beforeEnd	前一个起始或结束标签与当前元素结束标签之间的文本
afterEnd	当前元素结束标签和下一个起始或结束标签之间的文本

355

getAttribute( )

IE *4* NN *n/a* Moz *all* Saf *all* Op *7* DOM *1*

```
getAttribute(attributeName), getAttribute(attributeName[,caseSensitivity])
```

返回当前元素内指定属性的值。如果此属性反映了对象模型中的一个属性，那么此方法在读取对象的对应属性后会返回相同的值。这是在W3C DOM下读取一个元素对象属性值的很好方式。

在当前的各种浏览器中，作为参数传递的属性名称并不区分大小写。然而，IE还提供了一个可选的第二参数，它可以强制限制属性名称必须区分大小写。这样可能鼓励重用名称相同而大小写不同的属性名，但这可不是一种值得推荐的好习惯。

请参见setAttribute()方法以指定属性值并创建新的属性/值对。

返回值 W3C DOM（已在Mozilla、Safari和Opera中实现）指出属性值均应以字符串数据类型来表示。但是IE也可能以字符串、数字或布尔变量的形式返回属性值。

参数

attributeName

HTML标签中所使用的属性名称（不包括“=”），默认情况下不区分大小写。但IE浏览器可以选择是否区分大小写，而其他浏览器则完全不介意字符的大小写，即使将DOCTYPE设置为XHTML变量时也是如此。考虑到XHTML区分大小写，因此最好形成良好的习惯，注意区分大小写。

caseSensitivity

仅适用于IE的一个可选整数值。默认值为0，即不区分大小写。如果值为1，则为了返回对应的属性值，须要让*attributeName*参数与HTML标签中的属性名称完全一致。

getAttributeNode( )

IE *6* NN *n/a* Moz *all* Saf *all* Op *7* DOM *1*

```
getAttributeNode(attributeName)
```

返回与输入参数相关的属性节点（Attr对象）的一个引用。尽管节点类型与元素attributes属性所返回的数组中的节点类型一致，但使用getAttributeNode()方法可以直接通过属性名称访问Attr节点对象。在XML文档中，由于属性能够传达与元素相关的重要数据，所以此方法对XML文档更有帮助。请参见Attr对象以获取更多关于节点类型的详细信息。

返回值 Attr对象的一个引用。

参数

356

attributeName

标签中使用的属性名称，不包括“=”。浏览器并不要求区分大小写。但是考虑到XHTML区分大小写，因此最好形成良好的习惯，注意区分大小写。

getAttributeNodeNS( )

IE *n/a* NN *n/a* Moz *all* Saf *all* Op *7* DOM *2*

```
getAttributeNodeNS("namespaceURI", "localName")
```

根据当前元素内与输入参数相匹配的命名空间URI，返回一个本地命名的Attr对象的引用。这个方法的工作方

式与getAttributeNS()相类似，但它更适用于XML文档中由命名空间规范指定的属性。

返回值 Attr对象的一个引用。

参数

namespaceURI

URI字符串，它与文档中指定给某个标注的URI相对应。

localName

属性的本地名称。

getAttributeNS( )

IE *n/a* NN *n/a* Moz *all* Saf *all* Op *7* DOM *2*

```
getAttributeNS("namespaceURI", "localName")
```

根据当前元素内与输入参数相匹配的命名空间URI，返回一个本地命名属性的值。这个方法的工作方式与getAttribute()相似，但它更适用于XML文档中由命名空间规范指定的属性。在下面的这段示例XML文档中，针对libBook:title元素的一个属性就使用了一个命名空间：

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<results xmlns:libBook="http://catalog.umv.edu/schema">
<libBook:title libBook:rareBooks="true">De Principia</libBook:title>
</results>
```

如果要获取libBook:rareBooks属性值，那么针对该元素的方法应该包含getAttributeNS()方法及如下的参数：

```
getAttributeNS("http://catalog.umv.edu/schema", "rareBooks")
```

返回值 W3C DOM、Netscape、Safari和Opera中均以字符串数据类型来返回属性值。

参数

namespaceURI

URI字符串，它与文档中指定给某个标注的URI相对应。

localName

属性的本地名称。

getBoundingClientRect( )

IE *5(Win)* NN *n/a* Moz *n/a* Saf *n/a* Op *n/a* DOM *n/a*

返回一个IE中的TextRectangle对象，它描述了当前元素（包括非文本元素，如图像等）所占的矩形空间。这个矩形（拥有4个关于坐标的属性：top、right、bottom、left）与内容中最宽的部分（如，在段落中最长的一行）等宽，与所有内容的高度和等高。如果须要获取文本元素中单独行的矩形空间尺寸，请参考getClientRects()方法。

返回值 TexTRectangle对象。

参数 无。

getClientRects( )

IE *5(Win)* NN *n/a* Moz *n/a* Saf *n/a* Op *n/a* DOM *n/a*

返回一个TextRectangle对象数组。该数组中的每一项都是一个TextRectangle对象，这个对象则对应于一个多行文本元素中的一行。对于那些拥有不同字体尺寸或行高的不同行，将使用不同高度的方块进行包装。请参见TextRectangle对象以了解其属性。如果要获取整个元素的TextRectangle对象，请使用getBoundingClientRect()方法。

返回值 TextRectangle对象数组。

参数 无。

getElementsByTagName( )

IE 5 NN n/a Moz all Saf all Op 7 DOM 1

```
getElementsByTagName("tagName")
```

如果元素的标签名与方法参数相匹配，那么返回该元素的子元素数组。数组中的元素包括其子级、孙级元素等，它们均按照源代码顺序确定其层次等级。而此数组内并不包含当前元素。当无法找到与方法参数相匹配的元素时，将返回一个长度为零的数组。

IE 5/Macintosh、IE 6/Windows及其他所有能够提供支持的浏览器均允许使用星号匹配符作为参数，这样就可以不必进行标签名称匹配而直接返回一个由所有子孙元素组成的数组。还须要注意的是，由于不同浏览器的文档树结构可能存在细微的差别，因此当使用通配符作为参数时返回的数组长度也可能不完全相同。

返回值 由零个或多个元素引用组成的数组。

参数

tagName

目标元素的标签名，默认情况下不区分大小写。当然，也可以使用“*”作为通配符以指代所有的标签名称。

getElementsByTagNameNS( )

IE n/a NN n/a Moz all Saf all Op 7 DOM 2

```
getElementsByTagNameNS("namespaceURI", "localName")
```

如果元素的本地名称与方法的第二个参数相匹配，并且有一个命名空间的URI（该URI作为一个命名空间声明出现在文档中的任意位置）与第一个参数相同，那么会以数组的形式返回该元素的所有子孙元素。数组中的元素包括其子级、孙级元素等，它们均会按照源代码顺序确定其层次等级。此数组内并不包含当前元素。当无法找到与方法参数相匹配的元素时，将返回一个长度为零的数组。这个方法主要应用于XML文档。

返回值 由零个或多个元素引用组成的数组。

参数

namespaceURI

URI字符串，它与文档中指定给某个标注的URI相对应。

localName

标签名称的本地名称部分。

getExpression( )

IE 5(Win) NN n/a Moz n/a Saf n/a Op n/a DOM n/a

```
getExpression("attributeName")
```

返回字符串形式的脚本表达式，该表达式使用在`setExpression()`方法之中，通过它可以访问当前元素的某个属性。`setExpression()`方法指定了一段脚本表达式，以便计算该属性的属性值。为响应某些事件类型及`document.recalc()`方法，会自动计算这个表达式。为读取属性的当前值，必须使用`eval()`方法来解析并计算`getExpression()`方法返回的字符串。请参看本章后文中的`setExpression()`方法。

返回值 字符串。

参数

attributeName

当前元素中某个属性的名称，使用此方法前应该已经使用`setExpression()`方法为该属性指定了一个表达式。

getFeature( )

IE n/a NN n/a Moz 1.7.2 Saf n/a Op n/a DOM 3

```
getFeature("feature", "version")
```

W3C DOM规范指出，根据参数传递的功能部件/版本组合，这个方法返回已实现对应API集的对象。到Mozilla 1.8.1为止，该方法在Mozilla浏览器中均会返回一个未将自身暴露给脚本的对象。目前看来，除了检查参数的

数量，这个方法并不会对参数值进行合法性检查。请参见implementation.hasFeature()方法。

返回值 对象。

参数

feature

与DOM模型对应的DOM功能部件名称，如Core、HTML或Events。

version

由DOM等级组成的字符串，如“1.0”或“3.0”。

getUserData( )

IE *n/a* NN *n/a* Moz *(1.7.2)* Saf *n/a* Op *n/a* DOM *3*

```
getUserData("key")
```

W3C DOM规范指出，该方法返回一个任意类型的对象，而这个对象已经通过setUserData()方法与当前节点联系在一起。一个节点也可能与多个对象联系在一起，并且可以通过它们的关键字（标注）来区分它们。尽管最新版本的Mozilla浏览器已经一定程度上实现了这个方法（例如，浏览器会检查是否输入了参数），但直到Mozilla 1.8.1为止，在Web页面中调用这个方法依然会抛出一个NS_DOM_NOT_IMPLEMENTED错误。

返回值 任意JavaScript类型的对象。

参数

key

指定标注的字符串，已经通过setUserData()方法以该标注为关键字进行了数据保存。

hasAttribute( )

IE *n/a* NN *n/a* Moz *all* Saf *all* Op *7* DOM *1*

```
hasAttribute("attributeName")
```

如果当前元素拥有一个与方法参数相匹配的属性，则返回布尔值true。

返回值 布尔值：true | false。

参数

attributeName

须要搜索的属性名称，注意区分大小写。

hasAttributeNS( )

IE *n/a* NN *n/a* Moz *all* Saf *all* Op *7* DOM *2*

```
hasAttributeNS("namespaceURI", "localName")
```

如果当前元素中某个属性的本地名称与方法的第2个参数相匹配，并且有一个命名空间的URI（该URI作为一个命名空间声明出现在文档中的任意位置）与第1个参数相同，那么返回一个布尔值true。

返回值 布尔值：true | false。

参数

namespaceURI

URI字符串，它与文档中指定给某个标注的URI相对应。

localName

属性名的本地名称。

hasAttributes( )

IE *n/a* NN *n/a* Moz *all* Saf *all* Op *7* DOM *1*

如果在当前元素的标签内明确指定了任意一个属性，那么返回一个布尔值true。

返回值 布尔值：true | false。

参数 无。

hasChildNodes( )

IE 5 NN n/a Moz all Saf all Op 7 DOM 1

如果当前节点包含一个或多个子节点，那么返回一个布尔值true。

返回值 布尔值：true | false。

参数 无。

insertAdjacentElement( )

IE 5(Win) NN n/a Moz n/a Saf n/a Op 7 DOM n/a

```
insertAdjacentElement("where", elementObjectReference)
```

此方法可将一个元素对象插入某个与当前元素相关的指定位置。通常会单独创建（例如，通过document.createElement()）这个将要被插入的对象，或者它就是文档树中某个已存在的对象的引用，而且这个方法会在insertAdjacentElement()方法的辅助之下将这个对象移动至新的位置。

具体的目标位置将由第一个参数控制，它决定了在何处进行插入，第1个参数所使用的4个常量值如下所示：

位置	插入新元素
BeforeBegin	放置在当前元素的起始标签之前，作为一个前置的同辈元素
AfterBegin	放置在当前元素的起始标签之后，作为其第一个子元素
BeforeEnd	放置在当前元素的结束标签之前，作为其最后一个子元素
AfterEnd	放置在当前元素的结束标签之后，作为其下一个同辈元素

尽管文档树中不同的插入效果已经得到了很好的定义，但随着当前元素与插入元素所组成的行内和块级元素组合的不同，实际的显示结果也会存在差异。如果插入一个块级元素（例如一个div或p元素），就会使得该元素显示在下一行，且位于块级元素定位环境（如body或td元素）的左边缘处。另外，在元素上使用W3C DOM的appendChild()方法等效于使用本方法并以“beforeEnd”作为第1个参数。

返回值 被插入的元素对象的引用。

参数

where

以下字符串常量之一：BeforeBegin | AfterBegin | BeforeEnd | AfterEnd，不区分大小写。第1个和最后1个插入位置位于当前元素的HTML标签之外，而另外两个则位于元素内容与标签之间。

elementObjectReference

任意有效的元素对象引用，该对象既可以存在于文档树中，也可以是动态创建的对象。

insertAdjacentHTML( )

IE 4 NN n/a Moz n/a Saf n/a Op 7 DOM n/a

```
insertAdjacentHTML("where", HTMLText)
```

此方法可将一个文本字符串插入元素现有HTML代码的指定位置。如果HTML标签是待插入文本的一部分，那么浏览器会解释这些标签并且按照预期进行显示。

返回值 无。

参数

where

以下字符串常量之一：BeforeBegin | AfterBegin | BeforeEnd | AfterEnd。第1个和最后1个插入位置位于当前元素的HTML标签之外，而另外两个则位于元素内容与标签之间。

HTMLText

将要插入到指定位置的文本和/或HTML代码字符串。

insertAdjacentText( )

IE *4* NN *n/a* Moz *n/a* Saf *n/a* Op *7* DOM *n/a*

```
insertAdjacentText("where", text)
```

将文本插入元素现有HTML代码的指定位置。如果HTML标签是待插入文本的一部分，那么这些标签会直接显示在页面上。

返回值 无。

参数

where

以下字符串常量之一：`BeforeBegin`、`AfterBegin`、`BeforeEnd`或`AfterEnd`。第1个和最后1个插入位置位于当前元素的HTML标签之外，而另外两个则位于元素内容与标签之间。

HTMLText

将要插入到指定位置的文本字符串。

insertBefore( )

IE *5* NN *n/a* Moz *all* Saf *all* Op *7* DOM *1*

```
insertBefore(newChildNode, referenceChildNodeOrNull)
```

在当前元素的某个子节点之前，插入一个节点作为其子节点，通常情况下当前节点是一个元素。新的子节点可以是文档树中的某个节点的引用，在这种情况下如果调用此方法，就会将它从原始位置移走。当然，也可以重新创建一个有效的DOM节点作为子节点，其中包括文档片段（它可能会含有HTML标签）或`Attr`，但只有IE 6及其后续版本，以及部分浏览器能够支持后者。

第2个参数用于从现有子节点中指定一个引用点，新的子节点将插入到该点之前。另外，如果将第2个参数设置为`null`（或在IE中忽略该参数），那么新节点作为当前节点的最后一个子节点，此时就与`appendChild()`方法的执行结果一致。

返回值 被插入的节点对象的引用。

参数

newChildNode

任意一个能够作为子节点的有效节点对象。

referenceChildNodeOrNull

当前节点的任意一个子节点，或`null`。

isDefaultNamespace( )

IE *n/a* NN *n/a* Moz *1.7.2* Saf *n/a* Op *9* DOM *3*

```
isDefaultNamespace("namespaceURI")
```

如果当前节点的默认命名空间与参数相匹配，那么返回一个布尔值`true`。

返回值 布尔值：`true | false`。

参数

namespaceURI

用于测试的一个命名空间的URI字符串。

isEqualNode( )

IE *n/a* NN *n/a* Moz *1.7.2* Saf *n/a* Op *n/a* DOM *3*

```
isEqualNode(nodeReference)
```

如果根据节点相等的判断规则，当前节点与参数传入的节点相等，那么将返回一个布尔值`true`。这两个节点

的部分属性值必须完全相等，如localName、namespaceURI、nodeName、nodeType、nodeValue和prefix，而且它们的attributes和childNodes属性也必须相对。而在比较相等性时，并未考虑它们在文档树中的相对位置。如果这两个节点通过isSameNode()方法判断时完全一致，那么它们也是相等的节点。

返回值 布尔值：true | false。

参数

nodeReference

一个不同的节点的引用，将检测它是否与当前节点相等。

isSameNode( )

IE *n/a* NN *n/a* Moz *1.7.2* Saf *n/a* Op *n/a* DOM *3*

```
isSameNode(nodeReference)
```

如果当前节点与参数传入的节点完全相同，那么返回布尔值true。

返回值 布尔值：true | false。

参数

nodeReference

另外一个节点的引用，将检测它是否与当前节点相同。

isSupported( )

IE *n/a* NN *n/a* Moz *all* Saf *all* Op *7* DOM *2*

```
isSupported("feature", "version")
```

如果当前节点支持一个指定的W3C DOM模型和版本，那么就会返回一个布尔值true。如果document.implementation对象的hasFeature()方法也完成了相同的检测，那么在整个浏览器应用程序中均是如此。而isSupported()方法是在一个单独的节点上完成检测的，以便检查当前节点类型是否能为某些功能提供支持。isSupported()方法与document. implementation.hasFeature()方法所使用的参数值相同。

在浏览器为特定的DOM模型返回true之前，浏览器会首先验证浏览器中实现的DOM能否符合该模型。但此时并不强求这些实现完美无缺没有漏洞，也不要求它与其他实现保持一致。因此，这不过是个资格记录而已。

从理论上说，可以在访问一个属性或调用一个方法之前使用这个方法来检查所用的模型是否得到了支持。在如下的示例代码中，假设myElem是一个元素节点的引用：

```
if (myElem.isSupported("CSS", "2.0")) {
    myElem.style.color = "green";
}
```

但由于W3C DOM所支持的这种报告功能并未在所有的主流浏览器中得以实现，而且也不能完全保证后向兼容，因此实际上对对象进行检查是一种更好的解决方式。

返回值 布尔值：true | false。

参数

feature

至W3C DOM Level 2为止，可使用的模型名称字符串（区分大小写）如下：Core、XML、HTML、Views、StyleSheets、CSS、CSS2、Events、UIEvents、MouseEvents、MutationEvents、HTMLEvents、Range和traversal。

version

字符串表达式，使用主、次两个数字表示第一个参数所引用的DOM模型的版本号。即使DOM模型支持另一种拥有版本计数系统的W3C标准，对W3C DOM Level 2而言，版本号依然是“2.0”。因此，就算HTML版本为4.x，也只须检查对2.0版的HTML DOM模型是否支持。

lookupNamespaceURI( )

IE *all* NN *all* Moz *1.7.2* Saf *all* Op *9* DOM *3*

364

```
lookupNamespaceURI("prefix")
```

如果参数传入的前缀与预定义的命名空间相匹配，那么就为节点返回对应的URI字符串。这个方法主要应用于XML文档之中。

返回值 URI字符串。

参数

prefix

一个由命名空间前缀组成的字符串。

lookupPrefix( )

IE *all* NN *all* Moz *1.7.2* Saf *all* Op *9* DOM *3*

```
lookupPrefix("namespaceURI")
```

如果参数传入的命名空间前缀与预定义的命名空间相匹配，那么就为节点返回对应的前缀。这个方法主要应用于XML文档中。某些早期版本的Mozilla浏览器已经实现了这个方法，不过其名称为`lookupNamespacePrefix()`。

返回值 前缀名字符串。

参数

namespaceURI

一个命名空间的URI字符串。

mergeAttributes( )

IE *5(Win)* NN *n/a* Moz *n/a* Saf *n/a* Op *n/a* DOM *n/a*

```
mergeAttributes(modelElementReference[,preserveIDs])
```

这个方法从作为参数的元素处复制属性的名称/值对到当前元素。如果须要从一个已存在的元素处复制大量属性到一个新创建的元素，这个方法将很有用处。在默认情况下，为了便于脚本和表单操作，在复制时不会包含`id`或`name`属性，因此复制双方将保持独立的标识符。但自从IE 5.5开始，可以将一个布尔变量作为可选的第2个参数，当它设置为`false`时，也会复制`id`和`name`属性。

返回值 无。

参数

modelElementReference

一个现有元素的引用，该元素会作为模板提供属性的名称/值对以复制到当前元素。

preserveIDs

一个可选的布尔值。如果其值为`false`，那么模板元素的`id`和`name`属性会复制到当前元素。此参数的默认值为`true`。

moveRepetitionBlock( )

IE *n/a* NN *n/a* Moz *n/a* Saf *n/a* Op *9* DOM *n/a*

365

```
moveRepetitionBlock(distance)
```

与上移、下移输入元素类型等价，这个方法可以将一个重复块向上或向下移动。方法中的整形参数既指定了移动的方向也说明了当前块须要移动的距离。当参数为负值时向上移动指定的距离，而正值则向下移动。

返回值 无。

参数

distance

一个正整数或负整数，指出当前元素须要在其同辈节点中移动多远。负值向上移动，而正值向下移动。

normalize( )

IE 5(Mac)/6(Win) NN n/a Moz all Saf 1.2 Op 7 DOM 2

将当前元素/节点内的所有同辈文本节点压缩至一个文本节点内。向一个元素内插入或删除子节点后可能须要调用这个方法，特别是当节点遍历脚本以便让一个文本节点包含获取连续的文本时，更需要这个方法。W3C DOM认为，只有当文本节点没有其他同辈文本节点时，文档树才处于正常状态。须要注意的是，在IE 6中，使用这个方法有崩溃的危险。

返回值 无。

参数 无。

releaseCapture( )

IE 5(Win) NN n/a Moz n/a Saf n/a Op n/a DOM n/a

关闭由`setCapture()`方法设定的鼠标事件捕获模式。在IE的事件模型中，只是临时使用鼠标事件捕获，例如，在一个自定义上下文菜单被激发时处理鼠标事件。在以下几种情况下，会自动释放IE的事件捕获：聚焦于另一个窗口、框架或浏览器的地址栏；滚动一个窗口；显示一个系统对话框；显示真实的上下文菜单。

返回值 无。

参数 无。

removeAttribute( )

IE 4 NN n/a Moz all Saf all Op 7 DOM 1

```
removeAttribute("attributeName"), removeAttribute("attributeName"[, caseSensitivity])
```

从当前元素上删除指定的属性。如果使用`setAttribute()`方法为元素添加了属性，那么IE 4将无法删除这些属性，但自IE 5开始属性的删除并不会受此限制。虽然从浏览器查看源代码时会发现删除属性并不会修改源代码，但这的确会影响浏览器对元素的显示。一旦属性被删除，那么也就不能再使用其属性值或节点。

返回值 在IE中，如果删除成功则返回布尔值`true`；如果属性不存在，将返回`false`。其他支持这一方法的浏览器（或 W3C DOM规范）无返回值。

参数

attributeName

HTML标签中所使用的属性名称（不包括“=”），默认情况下不区分大小写。IE浏览器可以选择是否区分大小写，而Netscape 6则不考虑这个因素。但是考虑到XHTML区分大小写，因此最好形成良好的习惯，注意区分大小写。

caseSensitivity

仅适用于IE的一个可选整数值。默认值为0，即不区分大小写。如果值为1，则为了返回对应的属性值，须要让*attributeName*参数与HTML标签中的属性名称完全一致。

removeAttributeNode( )

IE 6 NN n/a Moz all Saf all Op 7 DOM 1

```
removeAttributeNode(attrObjectReference)
```

根据参数传递的`Attr`节点对象引用，从当前元素上删除属性。如果脚本仅拥有一个`Attr`节点对象的引用，而不是具体的属性名称，那么这个方法就可用于从元素上删除一个属性。虽然从浏览器查看源代码时会发现删除属性节点并不会修改源代码，但这的确会影响浏览器对元素的显示。一旦属性被删除，那么也就不能再使用其属性值或节点。

返回值 被删除的`Attr`对象的引用，它将不再是文档树的一部分，但可以被插入到文档树的其他位置。

参数

attrObjectReference

与当前元素相关的一个`Attr`节点对象的引用。

removeAttributeNS( )

IE *n/a* NN *n/a* Moz *all* Saf *all* Op *7* DOM *2*

```
removeAttributeNS("namespaceURI", "localName")
```

从当前元素内删除一个本地命名的属性，该属性与参数传递的命名空间URI相匹配的。这个方法的工作方式与removeAttribute()相类似，但它更适用于XML文档中由命名空间规范指定的属性。在下面的这段示例XML文档中，针对libBook:title元素的一个属性使用一个命名空间：

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<results xmlns:libBook="http://catalog.umv.edu/schema">
<libBook:title libBook:rareBooks="true">De Principia</libBook:title>
</results>
```

如果要删除libBook:rareBooks属性值，那么针对该元素的方法应该包含removeAttribute()方法及如下的参数：

```
removeAttributeNS("http://catalog.umv.edu/schema", "rareBooks")
```

返回值 无。

参数

namespaceURI

URI字符串，它与文档中指定给某个标注的URI相对应。

localName

属性的本地名称部分。

removeBehavior( )

IE *5(Win)* NN *n/a* Moz *n/a* Saf *n/a* Op *n/a* DOM *n/a*

```
removeBehavior(behaviorID)
```

如果曾经使用addBehavior()方法为元素增加了一个行为，那么removeBehavior()方法可以断开这个行为和当前元素之间的联系。此方法需要的参数是之前调用的addBehavior()方法返回的整数值，因此在完成addBehavior()方法的调用之后必须保存该返回值。

返回值 如果删除成功，返回布尔值true，否则返回false。

参数

behaviorID

针对当前元素和行为类型，最初由addBehavior()方法生成的整形序列号。

removeChild( )

IE *5* NN *n/a* Moz *all* Saf *all* Op *7* DOM *1*

```
removeChild(childNodeReference)
```

从当前节点上删除一个子节点。传入的参数必须是一个内嵌在当前节点之内现存子节点的引用。一旦删除该子节点，它就不再是文档树的一部分，但依然会将它保存在内存之中。此方法会返回这个子节点的引用，因此可以对它进行修改并放置在文档树中的任意位置。须要注意的是，可以命名一个节点删除其子节点，但不能命令它删除其自身，请参看IE的removeNode()方法。

返回值 被删除的节点的引用。

参数

childNodeReference

一个现存子节点的引用。

removeEventListener( )

IE *n/a* NN *n/a* Moz *all* Saf *all* Op *7* DOM *2*

```
removeEventListener("eventType", listenerFunction, useCapture)
```

断开当前节点与一个事件处理方法之间已建立的事件绑定。这个方法假设之前已经为节点添加了一个事件监

听器。为了确保正确地删除对应的事件监听器，removeEventListener()方法中所使用的3个参数应该与addEventListener()方法的参数完全相同。可以在同一个节点上多次调用这个方法，只要参数不同就不会干扰同一个节点上分配的其他事件监听器。须要注意的是，只有当关闭特定的事件处理能够改善用户与节点的交互性时，才调用这个方法。

368

返回值 无。

参数

eventType

浏览器对象模型中的一个已知事件类型的字符串（不含“on”前缀）。W3C DOM能够理解的事件类型有：abort、blur、change、click、DOMActivate、DOMAttrModified、DOMCharacterDataModified、DOMFocusIn、DOMFocusOut、DOMNodeInserted、DOMNodeInsertedIntoDocument、DOMNodeRemoved、DOMNodeRemovedFromDocument、DOMSubtreeModified、error、focus、load、mousedown、mousemove、mouseout、mouseover、mouseup、reset、resize、scroll、select、submit、unload。

listenerFunction

方法引用，当节点在特定的传播模式中监听到事件类型时，将执行该方法。由于这是一个方法对象的引用，因此不要将方法名放置在引号中，也不要包含括号。在执行过程中，浏览器会自动将当前的事件对象作为一个参数传递到监听程序。

useCapture

一个布尔值。如果为true，那么只有当节点朝着目标节点（即事件捕获节点）传播时节点才会监听事件类型。如果为false，那么只有当事件从事件目标向上传播时，节点才会进行监听。如果当前节点即为事件的目标节点，那么这两个布尔值均可使用。

removeExpression( )

IE *5(Win)* NN *n/a* Moz *n/a* Saf *n/a* Op *n/a* DOM *n/a*

```
removeExpression("attributeName")
```

解除一个为元素属性指定的表达式（通过setExpression()方法指定）。自动的表达式重算功能能够根据用户动作修改属性值（也可以使用document.recalc()方法进行显式的重算），但调用removeExpression()方法后，会关闭这种重算功能。但是，就算表达式已经删除，最近一次的计算结果依然会作用于该属性。

返回值 如果删除成功，返回布尔值true，否则返回false。

参数

369

attributeName

当前元素某个属性的名称，使用此方法前应该已经使用setExpression()方法为该属性指定了一个表达式。

removeNode( )

IE *5(Win)* NN *n/a* Moz *n/a* Saf *n/a* Op *7* DOM *n/a*

```
removeNode([childrenFlag])
```

从文档树中删除当前节点。此方法会返回这个子节点的引用，因此可以对它进行修改并放置在文档树中的任意位置。默认情况下，此方法只会删除当前节点，但不会删除其子节点。但是，如果仅仅只删除容器节点而保留其子节点会给文档树带来很大的破坏，当容器节点是复杂元素时尤其如此，例如表格容器。

返回值 被删除的节点的引用。

参数

childrenFlag

可选用的布尔值，包括：false（默认值），仅删除当前节点；true，删除当前节点与其所有内嵌的子节点。

removeRepetitionBlock( )

IE *n/a* NN *n/a* Moz *n/a* Saf *n/a* Op *9* DOM *n/a*

从其父容器内删除当前重复块。这个脚本代码等价于“删除”类型的input元素。

返回值 无。

参数 无。

replaceAdjacentText( )

IE *5(Win)* NN *n/a* Moz *n/a* Saf *n/a* Op *n/a* DOM *n/a*

```
replaceAdjacentText("where", "newText")
```

使用新文本替换与当前元素临近的连续文本块。这个方法只能作用于已显示的文本字符，对HTML标签没有影响。

须要删除的文本位置和新文本的插入点由第一个属性控制，它决定了在何处进行插入，这4个常量值如下所示：

位置	替换
BeforeBegin	前一个起始或结束标签与当前元素起始标签之间的文本
AfterBegin	当前元素起始标签和下一个起始或结束标签之间的文本
BeforeEnd	前一个起始或结束标签与当前元素结束标签之间的文本
AfterEnd	当前元素结束标签和下一个起始或结束标签之间的文本

由于替换文本仅仅是改变了文本节点的值，因此并不会影响文档树的结构。

返回值 已删除的文本字符串。

参数

where

以下字符串常量之一：BeforeBegin | AfterBegin | BeforeEnd | AfterEnd，不区分大小写。第一个和最后一个插入位置位于当前元素的HTML标签之外，而另外两个则位于元素内容与标签之间。

newText

用于替换旧文本的文本字符串。如果文本内包含HTML标签，那么会直接显示这些标签字符。

replaceChild( )

IE *5* NN *n/a* Moz *all* Saf *all* Op *7* DOM *1*

```
replaceChild(newChildNodeReference, oldChildNodeReference)
```

使用一个新的子节点替换当前节点的一个子节点。通常此方法与元素节点一同使用，但是它也可以用于Attr节点对象。方法所需的两个参数分别指向输入与输出子节点。新的子节点既可以是重新创建的节点，也可以是已存在于文档树中的现有节点的引用。在后一种情况中，调用replaceChild()方法会将该节点从它在文档树的原有位置上移除，并且根据第2个参数所指定的节点，插入到其子节点所在的位置。此方法会返回这个子节点的引用，因此可以对它进行修改并放置在文档树中的任意位置。须要注意的是，可以命名一个节点替换其子节点，但不能命令它替换其自身，但请参看IE的replaceNode()和swapNode()方法。

返回值 被删除的节点的引用。

参数

newChildNodeReference

一个节点引用，它会替换一个已存在的节点。

oldChildNodeReference

一个已存在的子节点引用，它将要被另一个节点所替换。

replaceNode( )

IE 5(Win) NN n/a Moz n/a Saf n/a Op n/a DOM n/a

```
replaceNode(newNodeObjectReference)
```

使用一个新节点替换当前节点。新的节点既可以是重新创建的节点（例如，一个文本节点或元素），也可以是已存在于文档树中的现有节点的引用。在后一种情况中，调用replaceNode()方法会将该节点从它在文档树的原有位置上移除，插入到当前节点所在的位置。

返回值 被删除的节点的引用。

参数

newNodeObjectReference

一个节点引用，它会替换当前节点。

scrollIntoView( )

IE 4 NN n/a Moz n/a Saf 2.02 Op 7 DOM n/a

```
scrollIntoView([showAtTop])
```

将包含当前元素的内容滚动至视图之中。在默认情况下，显示元素时会将元素的顶部与滚动空间的顶部平齐。当然，如果须要也可以将元素放置在滚动空间的底部。

返回值 无。

参数

showAtTop

一个可选的布尔值。如果值为true（默认值），那么会将元素放置在滚动空间的顶部；如果值为false，元素内容的底部会与滚动空间的底部平齐。

setActive( )

IE 5.5 NN n/a Moz n/a Saf n/a op n/a DOM n/a

不必将当前元素滚动至页面的可视区域就可让当前元素成为活跃元素。如果在窗口对象范围内调用这个方法，这个方法不会改变窗口或框架之间的聚焦状态。但调用这个方法后，元素会收到一个onfocus事件通知。

返回值 无。

参数 无。

setAttribute( )

IE 4 NN n/a Moz all Saf all Op 7 DOM 1

```
setAttribute("attributeName", value) setAttribute("attributeName", value[, caseSensitivity])
```

设定当前元素内指定属性的值。如果此属性反映了对象模型中的一个属性，那么此方法还为对象属性设置相同的值。W3C DOM指出，setAttribute()方法是调整一个属性值的最优方法，而getAttribute()则对应于属性值读取。

如果属性并不存在于元素之中，那么setAttribute()方法会将这个属性以名称/值对的形式添加至元素，但在IE 4至IE 5.5中，新增加的属性并不会成为元素attributes集合的一部分。

IE会将属性名视为节点属性的名称。因此，当两者之间出现差异时，IE需要节点属性名称。为IE或其他浏览器的元素指定class属性值时，如果对应的浏览器类型并未向Mozilla、Safari和Opera元素增加一个className属性，那么就须要增加一部分代码来调用这个方法。为了便于对象检测，支持W3C DOM的浏览器会为元素的getAttribute("class")方法返回一个字符串类型的值。

对于非IE类型的浏览器，必须使用字符串来为属性指定属性值。IE浏览器允许使用另外几种数据类型，例如数字量和布尔量，此时如果使用字符串形式的数字值来指定属性值，就会进行数据类型转换，这样一来getAttribute()会根据IE需要的数据类型返回属性值。

W3C浏览器并不区分属性名称的大小写，但是应该形成使用小写属性名称的良好习惯，这是XHTML发展的趋势。但在默认情况下，IE浏览器在处理此方法时会区分属性名称的大小写。因此IE提供了一个可选的第3个参数，以便对如何处理属性名称的大小写风格问题进行控制。要避免由大小写拼写问题而带来的属性名称混乱的现象，例如，两个不同的属性有相同的属性名，只不过大小写拼写方式不同。如果在代码中完全使用小写的属性名称，那么可以忽略IE提供的第3个属性，这样就可以保持对W3C DOM的兼容性。

如果在IE中使用`setAttribute()`指定元素的属性值时，须要加载外部内容，这时此方法有可能会失效。因此，将属性值赋给元素对象的对应属性后，须要对赋值情况进行检查。

返回值 无。

参数

`attributeName`

HTML标签中所使用的属性名称（在IE中会有一些例外，如上文所述）。

`value`

对Mozilla、Safari和Opera而言，属性值均为字符串数据类型。而在IE中，属性值可以是字符串、数字或布尔型，由属性的数据类型决定。对所有的值而言，字符串都是安全类型，IE会根据需要在其内部进行数据类型转换。

`caseSensitivity`

仅适用于IE的一个可选整数值。如果值为`1`（默认值），那么HTML标签中的属性必须与*`attributeName`*参数完全匹配才能进行属性值设置。这在允许拼写相同而大小写不同的多个属性名共存的情况下尤其适用。如果值为`0`，那么*`attributeName`*会找到第一个与它同名的属性并进行赋值，此时不考虑字符大小写。

setAttributeNode( )

IE *6* NN *n/a* Moz *all* Saf *all* Op *7* DOM *1*

`setAttributeNode(`*`attrObjectReference`*`)`

插入或替换当前元素内的一个属性。此方法的参数是一个`Attr`节点对象的引用，它既可以是一个新建的对象引用，也可以是文档树中另一个元素的引用。在调用`setAttributeNode()`方法时，浏览器首先会检查新属性的名称是否能够与现存属性名称相匹配。如果存在匹配，那么新属性将替换原有属性，否则会在元素中添加这个新属性。虽然从浏览器查看源代码时会发现添加属性节点并不会修改源代码，但如果该属性会影响元素的外观，那么这就会影响浏览器对元素的显示。通过`getAttribute()`方法可得到新属性的值。

返回值 属性替换时返回被替换的`Attr`对象的引用，如果是插入则返回`null`。

参数

`attrObjectReference`

由`document.createAttribute()`方法创建的`Attr`节点对象的引用，或者来自于文档树中另一个元素的`Attr`节点的引用。

setAttributeNodeNS( )

IE *n/a* NN *n/a* Moz *all* Saf *all* Op *7* DOM *2*

`setAttributeNodeNS(`*`attrObjectReference`*`)`

插入或替换当前元素内的一个属性。此方法的参数是一个`Attr`节点对象的引用，它既可以是一个新建的对象引用，也可以是文档树中另一个元素的引用。在调用`setAttributeNodeNS()`方法时，浏览器首先会检查能否找到一个与新属性的本地名称和命名空间URI组合相匹配的现存属性本地名称和命名空间URI组合。如果存在匹配，那么新属性将替换原有属性，否则会在元素中添加这个新属性。虽然从浏览器查看源代码时会发现添加属性节点并不会修改源代码，但如果该属性会影响元素的外观，那么这就会影响浏览器对元素的显示。通过`getAttributeNS()`方法得到新属性的值。

返回值 属性替换时返回被替换的Attr对象的引用，如果是插入则返回null。

参数

attrObjectReference

由document.createAttributeNS()方法创建的Attr节点对象的引用，或者来自于文档树中另一个元素的Attr节点的引用。

setAttributeNS( )

IE *n/a* NN *n/a* Moz *all* Saf *all* Op *7* DOM *2*

```
setAttributeNS("namespaceURI", "qualifiedName", " value")
```

插入或替换当前元素内的一个属性。如果根据传入的参数，能够在元素中找到一个完全符合命名空间URI和限定名的属性，那么会将该新值赋予这个现有属性。如果不存在匹配项，则在元素中加入新的属性。

返回值 无。

参数

namespaceURI

URI字符串，它与文档中指定给某个标注的URI相对应。

qualifiedName

属性的完整名称，包括本地名称前缀、一个冒号，以及本地名称。

value

字符串形式的属性值。

setCapture( )

IE *5(Win)* NN *n/a* Moz *n/a* Saf *n/a* Op *n/a* DOM *n/a*

```
setCapture([containerFlag])
```

为所有与点击相关的鼠标事件启动IE捕获模式，这些事件包括onclick、ondblclick、onmousedown、onmousemove、onmouseout、onmouseover和onmouseup。此时，无论实际的事件目标是什么，均会将这些事件的事件处理发送到当前元素。在处理自定义上下文菜单或拖拽时，这个方法对鼠标状态控制很有帮助。在鼠标事件捕获过程中，每个事件的event.srcElement属性均会保持一个元素引用，鼠标事件会发送给这个元素，但只有处于捕获模式的元素才会调用其事件处理程序实际处理这些事件。一旦不再需要这种捕获模式，可以通过releaseCapture()方法来进行解除。在以下几种情况中，会自动释放IE的事件捕获：聚焦于另一个窗口、框架或浏览器的地址栏；滚动一个窗口；显示一个系统对话框；显示真实的上下文菜单。

从IE 5.5开始，当用户在一个元素上触发鼠标事件时，如果该元素的父对象处于捕获模式，那么可以使用一个可选的参数来为事件传播提供更多的控制功能。在默认情况下（忽略这个参数或将参数设置为true），正如所料，父容器会截取这个事件。一旦将这个参数设置为false，那么会直接将事件传递至事件目标，即捕获模式元素的子元素。然后，这些事件可以正常地向上传递，而其他鼠标事件会发送至捕获模式的元素中。例如，如果在一个包含可点击表单控件的表单元素中调用setCapture()方法，那么通常会将参数设置为false，这样一来，就算打开捕获模式，也可以让鼠标动作到达它们期待的目标对象。否则，子元素不会响应鼠标动作，而表单控件就会如同被禁用一样。

返回值 无。

参数

containerFlag

布尔值，如果参数为true（默认值），那么会让当前元素（如果是一个容器元素）捕获所有的鼠标事件，如果设置为false，则会让鼠标事件在向上传递之前到达其真正的目标对象。

setExpression( )

IE 5(Win) NN n/a Moz n/a Saf n/a Op n/a DOM n/a

```
setExpression("propertyName", "expression", ["scriptLanguage"])
```

为元素对象的属性指定一个脚本表达式，以便动态计算该属性值。这个方法能够对元素对象的属性及它们的style对象起作用。对于大多数面向用户的事件，会自动重算该表达式，当然，在任意时刻都可以使用recalc()方法强制进行重算。

为元素属性指定一个表示后，可能会取代某些事件处理，例如，当用户改变浏览器窗口的尺寸时，保持元素之间的位置关系。为了让元素一直在浏览器窗口中水平居中，可以使用以下几种方法之一来为元素的style.left属性设置表达式：第一种方式将表达式作为元素的一个内嵌属性，语法如下：

```
<div id="heading" style="position:absolute; left:expression(
  document.body.clientWidth/2-document.getElementById("heading").offsetWidth/2);
```

另外，在装载期运行的方法可以包含如下的指令：

```
document.getElementById("heading").style.setExpression("left", "document.body.
clientWidth/2-document.getElementById('heading').offsetWidth/2;", "JScript");
```

在这两种方法中，使用了相同的表达式来计算元素左边缘相对于body元素当前可视宽度的坐标位置。由于这个表达式依赖于body元素的尺寸属性，因此当窗口大小改变从而引起body尺寸发生变化时，浏览器会自动地重算表达式。

此时须要保证表达式的计算结构值与属性所需数据类型的一致性。将表达式应用于setExpression()方法之前，最好对表达式进行一些测试，以保证其正确性。否则，一旦部署完毕，排错工作将会更加困难。

如果要使用一个表达式为属性指定一个固定值，请使用removeExpression()方法防止再对该属性值进行进一步的重算工作。

返回值 无。

参数

propertyName

由表达式控制的属性名称，它区分大小写，例如，须要使用className代替对应的class属性名称。

expression

一个包含脚本表达式的字符串。针对第一个参数所指定的属性，此表达式必须计算出一个适用于它的值，因此不允许使用由分号分隔的多条指令。与此同时，对其他元素的引用应该是完整引用。在早期的实现版本中，避免使用包含数组的引用形式。

scriptLanguage

以下3个常量字符串之一：JScript | JavaScript | VBScript。默认值为JScript。

setUserData( )

IE n/a NN n/a Moz (1.7.2) Saf n/a Op n/a DOM 3

```
setUserData("key", object, userDataHandler)
```

W3C DOM规范指出，这个方法可以将任意类型的一个对象与当前节点联系起来，而且之后可以通过getUserData()方法获取数据值。一个节点可能与多个对象联系在一起，并且可以通过它们的关键字（标注）来区分它们。这个对象可以是任意一种有效的JavaScript对象类型。当节点被吸纳（document.adoptNode()）、复制（cloneNode()）、删除、导入（document.importNode()）或重命名（document.renameNode()）时，可以根据第3个参数传入的处理方法引用来调用该方法。尽管最新版本的Mozilla浏览器已经一定程度上实现了这个方法（例如，浏览器会检查是否输入了参数），但直到Mozilla 1.8.1为止，在Web页面中调用这个方法依然会抛出一个NS_DOM_NOT_IMPLEMENTED错误。

返回值 任意类型的JavaScript对象，它将作为第2个参数被传入方法之中。

参数

376

key

指定标注的字符串，已经以该标注为关键字进行了数据保存。

object

任意类型的JavaScript对象，或者使用null表示删除之前设置的数据。

userDataHandler

当节点被吸纳、复制、删除、导入或重命名（document.renameNode()）时，可以根据这个方法引用来调用该方法。调用此方法时需要5个参数，如下所示：一个表示操作类型的整数值（复制=1、导入=2、删除=3、重命名=4、吸纳=5）；一个带有关键字的字符串；预存的数据对象；操作对象节点的引用；操作后新创建的节点的引用（如果未创建，则为null）。

swapNode( )

IE *5(Win)* NN *n/a* Moz *n/a* Saf *n/a* Op *n/a* DOM *n/a*

```
swapNode(otherNodeObject)
```

将当前节点与参数传入的节点进行交换。另一个节点可以重新创建或直接使用文档树中其他节点的引用。在第2种情况下，结果与双向交换相同，其本质就是两个节点交换位置。如果这两个节点属于不同的节点类型或元素显示类型（例如一个行内元素和一个块级元素），那么文档的显示效果可能会发生显著的改变。

返回值 调用该方法的节点的引用。

参数

otherNodeObject

任意节点对象的引用，通常是文档树中的另一个节点。

toString( )

IE *4* NN *4* Moz *all* Saf *all* Op *7* DOM *n/a*

返回一个代表当前元素对象的字符串。针对某些元素，从Mozilla、Safari和Opera浏览器中得到的字符串可能存在一定的相似性，但无法确保此现象一定出现。例如，这3个浏览器均会为p元素返回“[HTMLParagraphElement]”，而针对em元素，Mozilla将返回“[HTMLSpanElement]”，而Safari和Opera则返回“[HTMLElement]”。针对所有类型的元素，IE则会返回“[object]”。

返回值 由浏览器决定的字符串。

参数 无。

对象参考（按字母顺序编排）

377

a

IE *3* NN *2* Moz *all* Saf *all* Op *7* DOM *1*

a对象对应于一个a元素，此时这个a元素既可以是一个链接（link），也可以是一个定位标记（anchor）。早期的Netscape Navigator和IE浏览器只是将这个对象视为文档内links[]和anchors[]数组的一个成员。从IE 4及其他支持W3C DOM的浏览器开始，可以使用元素对象引用语法访问这个对象，例如，IE下可以使用document.all[]集合，而IE 5及后续版本或其他W3C浏览器还可以使用document.getElementById()。

等价HTML元素 <a>

对象模型引用方式

```
[window.]document.links[i]
[window.]document.anchors[i]
[window.]document.getElementById("elementID")
```

对象特定属性 charset、coords、dataFld、dataFormatAs、dataSrc、hash、host、hostname、href、hreflang、Methods、mimeType、name、nameProp、pathname、port、protocol、protocolLong、rel、rev、search、shape、target、text、type、urn

对象特定方法 无。

对象特定事件

事件	IE	Mozilla	Safari	Opera	W3C DOM
blur	·	·	—	·	—
click	·	·	·	·	·
dblclick	·	·	—	·	—
focus	·	·	—	·	—
help	·	—	—	—	—
mousedown	·	·	·	·	·
mousemove	·	·	·	·	·
mouseout	·	·	·	·	·
mouseover	·	·	·	·	·
mouseup	·	·	·	·	·

从第4版开始，Navigator中仅作为锚链的a对象就不再拥有任何事件。

charset

IE *6* NN *n/a* Moz *all* Saf *all* Op *7* DOM *1*

可读/可写

文档内容中所使用的字符编码。

示例

```
if (document.getElementById("myAnchor").charset == "csISO5427Cyrillic") {
    // 处理西里尔(Cyrillic)字符集
}
```

值 是否区分大小写取决于字符集注册表，请访问以下链接获取更多详细信息：*http://www.iana.org/assignments/character-sets*。

默认值 由浏览器决定。

coords

IE *6* NN *n/a* Moz *all* Saf *all* Op *7* DOM *1*

可读/可写

定义一个区域的外观，该区域与一个特定链接或脚本动作相关。此属性是a对象的内部成员之一，但由于area对象继承了a对象的属性，因此它实际上属于area。其坐标值是一系列由逗号分隔的数值列表。如果两个区域重叠在一起，则优先使用在代码中定义较早的区域。

示例

```
document.getElementById("mapArea2").coords = "25, 5, 50, 70";
```

值 每个坐标值都是一个长度值，但坐标的数值与它们的顺序都依赖于shape属性指定的元素形状。如果shape="rect"，则有4个坐标（左，上，右，下）；如果shape="circle"，则有3个坐标（圆心-x，圆心-y，半径）；当shape="poly"时，则有一组坐标值对，每对坐标代表多边形的一个顶点（x1，y1，x2，y2，x3，y3，…xN，yN）。

默认值 无。

dataFld

IE *4* NN *n/a* Moz *n/a* Saf *n/a* Op *n/a* DOM *n/a*

可读/可写

此属性用于实现IE浏览器中的数据绑定功能，可以使用远程数据源中的某一列的值代替元素中的href属性。

此时，元素中还必须设定datasrc属性。如果dataFld和dataSrc属性均设置为空字符串，那么就会打破元素与数据源之间的绑定关系。注意，只能在IE 5/Mac操作系统下实现文本文件数据源的绑定。

示例 `document.getElementById("hotlink").dataFld = "linkURL";`

值 数据源数据列内区分大小写的标识符。

默认值 无。

dataFormatAs

IE *4* NN *n/a* Moz *n/a* Saf *n/a* Op *n/a* DOM *n/a*

可读/可写

本属性用于IE浏览器中的数据绑定。对于来自于数据源的原始资料，浏览器根据此属性值决定将它们视为纯文本亦或是HTML标签。

示例 `document.getElementById("hotlink").dataFormatAs = "HTML";`

值 IE浏览器可识别两种有效的设定值：`text | html`。

默认值 `text`

dataSrc

IE *4* NN *n/a* Moz *n/a* Saf *n/a* Op *n/a* DOM *n/a*

可读/可写

为实现IE数据绑定功能，此属性用于指定页面内object元素的ID值，以便在进行远程数据访问时加载数据源对象。插入a元素的数据源内容由datafld属性决定，见上文。如果dataFld和dataSrc属性均设置为空字符串，那么就会打破元素与数据源之间的绑定关系。注意，只能在IE 5/Mac操作系统下实现文本文件数据源的绑定。

示例 `document.all.hotlink.dataSrc = "#DBSRC3";`

值 数据源内区分大小写的标识符。

默认值 无。

hash

IE *3* NN *2* Moz *all* Saf *all* Op *7* DOM *1*

可读/可写

提供href属性的URL中的锚链部分（即“#”之后的部分），它指向文档中某个锚链位置。在设置属性值时，请不要包含“#”。

示例

```
document.getElementById("myLink").hash = "section3";
document.links[2].hash = "section3";
```

值 字符串。

默认值 无。

host

IE 3 NN *2* Moz *all* Saf *all* Op *7* DOM *1*

可读/可写

对于链接而言，这个组合表示目标文档所在服务器的主机名与端口。如果端口是URL的一部分，那么主机和端口之间使用冒号进行分隔，这与它们在URL中的形式一样。如果在针对IE的HTTP URL中并未明确指定端口号，则会自动返回默认端口号，即80。

示例

```
document.getElementById("myLink").host = "www.megacorp.com:80";
document.links[2].host = "www.megacorp.com:80";
```

值 主机名字符串，可在其后加上冒号和端口号。

默认值 由服务器决定。

hostname

IE 3 NN 2 Moz all Saf all Op 7 DOM 1

可读/可写

对链接而言，这是目标文档所在服务器的主机名。须要注意的是，hostname属性并不包括端口号。

示例

```
document.getElementById("myLink").hostname = "www.megacorp.com";
document.links[2].hostname = "www.megacorp.com";
```

值 主机名字符串（主机与域）。

默认值 由服务器决定。

href

IE 3 NN 2 Moz all Saf all Op 7 DOM 1

可读/可写

提供由元素的href属性所指定的URL。

示例

```
document.getElementById("myLink").href = "http://www.megacorp.com";
document.links[2].href = "http://www.megacorp.com";
```

值 由完整或相对URL路径组成的字符串。

默认值 无。

hreflang

IE 6 NN n/a Moz all Saf all Op 7 DOM 1

可读/可写

380

提供了链接目标文档所采用的语言代码。使用此属性时，须要首先设定对应元素或对象的href属性。

示例 `document.getElementById("myLink").hreflang = "DE";`

值 不区分大小写的语言代码。

默认值 无。

Methods

IE 4 NN n/a Moz n/a Saf n/a Op n/a DOM n/a

可读/可写

为链接目标的功能的提供一个提示性的属性。浏览器获取此信息后，可以根据链接目标的相关动作来显示特定的颜色或图像，但IE浏览器对它似乎并不会做出任何处理。

示例 `document.links[1].Methods = "post";`

值 以字符串形式给出的任意的有效HTTP方法。

默认值 无。

381

mimeType

IE 4 NN n/a Moz n/a Saf n/a Op n/a DOM n/a

只读

返回明语版本的目标文档MIME类型，该目标文档位于href属性所指定的链接端。根据其返回信息，可以设置鼠标经过它时的对应样式。不要将这个属性与navigator.mimeTypes[]数组混为一谈，与此同时，它与Netscape Navigator中独立的mimeType对象也不相同。另外还要注意本属性无法在IE 4/Macintosh组合中使用。

示例

```
if (document.getElementById("myLink").mimeType == "GIF Image") {
    ...
}
```

值 以字符串形式给出的MIME类型明语参考。

默认值 无。

name

IE *3* NN *2* Moz *all* Saf *all* Op *7* DOM *1*

可读/可写

这是与一个元素相关的标识符，它将该元素转换成一个锚链。使用时也可以将该名称作为对象引用的一部分。

示例

```
if (document.links[12].name == "section3") {
    ...
}
```

值 区分大小写的标识符，必须遵循标识符命名规则：不包含空白、不能以数字作为起始字符、除下划线外不允许使用其他标点符号。

默认值 无。

nameProp

IE *4* NN *n/a* Moz *n/a* Saf *n/a* Op *n/a* DOM *n/a*

只读

仅返回元素内href属性的文件名，而不是整个URL。

示例

```
if (document.getElementById("myLink").nameProp == "logo2.gif") {
    ...
}
```

值 字符串。

默认值 无。

pathname

IE *3* NN *2* Moz *all* Saf *all* Op *7* DOM *1*

可读/可写

根据元素中href属性所对应的URL，提供其中的路径名部分。它由URL域名末字符之后的全部有用信息组成，其中包括位于首位的斜线符号。

示例

```
document.getElementById("myLink").pathname = "/images/logoHiRes.gif";
document.links[2].pathname = "/images/logoHiRes.gif";
```

值 字符串。

默认值 无。

port

IE *3* NN *2* Moz *all* Saf *all* Op *7* DOM *1*

可读/可写

根据元素的href属性所对应的URL，提供其中的端口号部分。它包含URL域名末字符之后紧接着冒号的全部有用信息。注意，冒号并不是port属性值的一部分。

示例

```
document.getElementById("myLink").port = "80";
document.links[2].port = "80";
```

值 字符串形式的数字值。

默认值 无。

protocol

IE 3 NN 2 Moz all Saf all Op 7 DOM 1

可读/可写

根据元素内href属性所对应的URL，此属性提供其中的协议内容。它包括URL内从起始部分到第一个冒号为止的所有信息。几种典型值包括："http:"、"file:"、"ftp:"和"mailto:"。

示例
```
document.getElementById("secureLink").protocol = "https:";
```
值 字符串。

默认值 无。

protocolLong

IE 4 NN n/a Moz n/a Saf n/a Op n/a DOM n/a

只读

383

根据元素或对象内href属性对应的URL，为其中所暗含的协议提供一个详细的说明。目前已不再赞成使用此属性。

示例
```
if (document.getElementById("myLink").protocolLong == "HyperText Transfer Protocol") {
    // 将此文档作为服务器文件进行处理
}
```
值 字符串。

默认值 无。

rel

IE 4 NN n/a Moz all Saf all Op 7 DOM 1

可读/可写

此属性用于定义当前元素与链接目标之间的关系，它也可叫做前向链接。在使用中无论如何也不要将该属性与href定义的链接目标文档相混淆。尽管主流浏览器并不会在a元素上使用这个属性，但可以将它看作一种参数，并可以在脚本控制之下对其进行检查或修改。请参见第1章内关于a元素的rel属性的讨论，这一部分简单讲述了未来将如何使用此属性。

值 不区分大小写的单个字符串，但其选择范围仅限于HTML 4.0认可的标准链接类型：alternate、appendix、bookmark、chapter、contents、copyright、glossary、help、index、next、prev、section、start、stylesheet、subsection

默认值 无。

rev

IE 4 NN n/a Moz all Saf all Op 7 DOM 1

可读/可写

此属性用于定义当前元素与链接目标之间的关系，通常也称之为“反向链接”。尽管主流浏览器并未支持这个属性，但可以将它看作一种参数并在脚本控制之下对其进行检查或修改。请参看第1章内关于a元素的rev属性的讨论，这一部分简单讲述了未来将如何使用此属性。

值 不区分大小写的单个字符串，但其选择范围仅限于HTML 4.0认可的标准链接类型。HTML 4认可链接类型请参考上文中的rel属性。

384

默认值 无。

search

IE 3 NN 2 Moz all Saf all Op 7 DOM 1

可读/可写

根据href属性对应的URL，提供其中以“?”开头的编码部分。作为查询结果的文档可能也包含对应的查询

部分，并将它作为window.location属性的一部分。在使用中也可以通过脚本来修改这个属性。这样一来，就可以将URL和搜索标准同时发送至服务器端。但是必须预先知晓服务器所需的数据格式（通常是一些名称/值对），才能正确地完成这一功能。

示例

```
document.getElementById("searchLink").search="?p=Tony+Blair&d=y&g=0&s=a&w=s&m=25";
document.links[1].search="?p=Tony+Blair&d=y&g=0&s=a&w=s&m=25";
```

值 以“?”开头的字符串。

默认值 无。

shape

IE *6* NN *n/a* Moz *all* Saf *all* Op *7* DOM *1*

可读/可写

此属性定义了服务器端图像映射区域的外形，具体的坐标由COORDS属性指定。通常由继承了a对象相关属性的area对象使用这个属性。

示例 `document.getElementById("myLink").shape = "circle";`

值 不区分大小写的外形名称常量字符串，包括：`default | rect | rectangle | circle | poly | polygon`。

默认值 `rect`

target

IE *3* NN *2* Moz *all* Saf *all* Op *7* DOM *1*

可读/可写

提供某个窗口或框架的名称，以便使得窗口或框架接收链接操作后返回的结果内容。这些名称会通过frame元素的name属性分配给对应框架，对子窗口而言，则会通过window.open()方法的第2个参数来指定其名称。如果须要使用target属性来指定链接的打开页面，并且还须要使用标记来通过严格HTML或XHTML DTD的验证，那么可以忽略源代码中的target属性，但必须在页面加载之后使用脚本为a元素的target属性设置所需的值。

示例

```
document.getElementById("homeLink").target = "_top";
document.links[3].target = "_top";
```

值 窗口或框架的名称字符串，或者下述字符串常量之一：_parent、_self、_top、_blank。_parent以当前文档所在的框架集作为目标；_self以当前窗口作为目标；_top以浏览器主窗口为目标，因此会删除所有的其他框架；而_blank会以默认尺寸创建一个新的窗口。

默认值 无。

text

IE *n/a* NN *4* Moz *all* Saf *all* Op *7* DOM *n/a*

只读

返回a元素起始与结束标签之间的文本。这个属性比W3C DOM还要历史悠久，尽管Mozilla、Safar和Opera均支持该属性，但只应该在Navigator 4中使用这个属性。

值 字符串。

默认值 无。

type

IE *6* NN *n/a* Moz *all* Saf *all* Op *7* DOM *1*

可读/可写

这是明语版本的目标文档MIME类型，该目标文档位于href属性所指定的链接端。此信息可以帮助浏览器为须要使用多媒体播放器或插件的资源提前做好准备。

示例

```
if (document.getElementById("myLink").type == "image/jpeg") {
    ...
}
```

值 MIME类型，且不区分大小写。MIME类型目录可从此处获得：*http://www.iana.org/assignments/media-types/*。

默认值 无。

urn

IE *4* NN *n/a* Moz *n/a* Saf *n/a* Op *n/a* DOM *n/a*

可读/可写

表示href属性指定的目标文档的URN（ Uniform Resource Name，统一资源名称）版本号。使用此属性主要是为将来URN格式的URI提供支持，IEFT一直在讨论这个不断发展的推荐标准，请参见RFC 2141。尽管IE浏览器支持这个属性，但它依然未能取代href属性的位置。

示例 `document.getElementById("link3").urn = "http://www.megacorp.com";`

值 完整或相对的统一资源名称。

默认值 无。

abbr, acronym, cite, code, dfn, em, kbd, samp, strong, var

IE *4* NN *n/a* Moz *all* Saf *all* Op *7* DOM *1*

所有这些对象都对应于一个同名的HTML短语元素。这些短语元素均为行内内容提供了一个上下文环境。在显示其中一部分元素时，会采取各种方法将它们与周围的文本区分开来。请参看第1章中对这些元素的相关描述，以获取更多详细信息。从脚本操作的角度看，所有的短语元素对象共享相同的属性、方法和事件集合。

等价HTML元素 <abbr>、<acronym>、<cite>、<code>、<dfn>、、<kbd>、<samp>、、<var>

对象模型引用方式 `[window.]document.getElementById("elementID")`

对象特定属性 无。

对象特定方法 无。

对象特定事件 无。

AbstractView

请参见window对象。

address

IE *4* NN *n/a* Moz *all* Saf *all* Op *7* DOM *1*

address对象对应于address元素。

等价HTML元素 <address>

对象模型引用方式 `[window.]document.getElementById("elementID")`

对象特定属性 无。

对象特定方法 无。

对象特定事件 无。

all

IE 4 NN n/a Moz 1.7 Saf n/a Op 7 DOM n/a

表示当前元素内嵌的元素集合，但目前只有Mozilla浏览器部分支持document.all。document.all的引用会返回文档中所包含的所有元素组成的对象集合，其中还包括文档的第一层元素内嵌的元素。这个集合会根据元素标签的源代码顺序进行排序。可以通过以下5种语法从元素ID入手获取该元素的引用：

```
document.all.elementID
document.all["elementID"]
document.all["elementID"]
document.all.item("elementID")
document.all.namedItem("elementID")
```

W3C DOM内只有document对象拥有与之等价的操作（即document.getElementById()方法），使用它可以在整个文档范围内获取元素引用。须要注意的是，Mozilla仅在quirks模式下才支持document.all操作，并且每次使用时均会在JavaScript错误控制台中显示一条警告信息。此外，如果在一个“if”条件下测试document.all的存在性，浏览器的返回数值如同document.all并不存在，因此，请不要将IE浏览器中使用的document.all的脚本转移到Mozilla下执行。

对象模型引用方式 *elementReference*.all

对象特定属性 length

对象特定方法 item()、namedItem()、tags()、urns()

对象特定事件 无。

length

IE 4 NN n/a Moz n/a Saf n/a Op 7 DOM n/a

只读

返回集合中元素的数量。

示例

```
var howMany = document.all.length;
```

值 整数值。

item( )

IE 4 NN n/a Moz n/a Saf n/a Op 7 DOM n/a

```
item(index[, subindex])
```

根据符合索引值（或任意索引及子索引值）的元素对象，返回一个单独的对象或一组对象集合。

返回值 一个对象或一个对象集合。如果并不存在与参数相匹配的对象，则返回null。

参数

index

当输入参数是一个从零开始计数的整数时，返回值是源代码内与某个具体项相对应的独立元素（内嵌在当前元素之内），而如果参数为一个字符串，则返回id或name属性与该字符串相符的元素集合。

subindex

如果为第1个参数指定一个字符串值，那么可以将第2个参数指定为一个从零开始计数的索引值，这样就可以从集合中找到一个id或name属性与第1个参数的字符串相匹配的具体元素。

namedItem( )

IE 4 NN n/a Moz n/a Saf n/a Op 7 DOM n/a

```
namedItem(IDOrName)
```

返回单个对象或对象集合，它们对应于符合参数字符串值的元素。

返回值 一个对象或一个对象集合。如果并不存在与参数相匹配的对象，则返回null。

参数

IDOrName

与目标元素的id或name属性包含相同值及拼写方式的字符串。

tags( )

IE 4 NN n/a Moz n/a Saf n/a Op 7 DOM n/a

```
tags(tagName)
```

从当前元素的全部内嵌对象中，返回标签与*tagName*参数相匹配的对象组成的集合。

返回值 对象集合。如果与参数相匹配的对象不存在，则返回一个长度为零的数组。

参数

tagName

仅包含元素标签名称的字符串（不包含尖括号），不区分大小写，例如document.all. tags("p")。

389

urns( )

IE 4 NN n/a Moz n/a Saf n/a Op n/a DOM n/a

```
urns(URN)
```

返回内嵌元素对象的集合，这些对象不仅附加了一些行为，而且其URN也符合输入的*URN*参数。

返回值 对象集合。如果与参数相匹配的对象不存在，则返回一个长度为零的数组。

参数

URN

一个包含本地或外部行为文件的统一资源命名的字符串。

anchors

IE 3 NN 2 Moz all Saf all Op 7 DOM 1

所有作为锚链使用的a元素的集合。此集合中的成员按照它们在源代码中的次序进行排序。所有的浏览器均允许使用数组符号来访问集合中的一个锚链，例如document.anchors[0]或document.anchors["section3"]。IE浏览器还允许将索引值放置在一个圆括号之中，例如document.anchors(0)。如果须要将锚链的名称作为索引值，则一定要使用name属性的属性值，而不要使用id属性。另外，如果须要通过锚链引用的id属性来访问对象，那么此时可以使用document.all.elementID（仅适用于IE浏览器）或document.getElementById("elementID")。目前的浏览器还包含一些额外的方法以提供基于名称或索引的访问操作。

对象模型引用方式 document.anchors

对象特定属性 length

对象特定方法 item()、namedItem()、tags()、urns()

对象特定事件 无。

length

IE 3 NN 2 Moz all Saf all Op 7 DOM 1

只读

390

返回集合中元素的数量。

示例

```
var howMany = document.anchors.length;
```

值 整数值。

item( )

IE 4 NN n/a Moz all Saf all Op 7 DOM 1

```
item(index[, subindex]) item(index)
```

根据符合索引值（IE中也可以使用index或subindex值）的元素对象，返回一个单独的锚链对象或一组锚链对象集合。

返回值 一个锚链对象或一个锚链对象集合。如果并不存在与参数相匹配的对象，则返回null。

参数

index

当输入参数是一个从零开始计数的整数（根据W3C规范）时，返回值是源代码内与某个具体项相对应的独立元素。如果参数为一个字符串，则返回id或name属性与该字符串相符的元素集合。

subindex

在IE中，如果为第1个参数指定一个字符串值，那么可以将第2个参数指定为一个从零开始计数的索引值，这样就可以从集合中找到一个id或name属性与第1个参数的字符串相匹配的具体元素。

namedItem( )

IE *6* NN *n/a* Moz *all* Saf *all* Op *7* DOM *1*

```
namedItem(IDOrName)
```

返回单个锚链对象或锚链对象集合，它们对应于符合参数字符串值的元素。

返回值 一个锚链对象或一个锚链对象集合。如果并不存在与参数相匹配的对象，则返回null。

参数

IDOrName

与目标元素的id或name属性包含相同值的字符串。

tags( )

IE *4* NN *n/a* Moz *n/a* Saf *all* Op *7* DOM *n/a*

```
tags(tagName)
```

从当前集合的全部内嵌对象中，返回标签与*tagName*参数相匹配的对象集合。虽然在所有的IE、Safari和Opera集合中均已实现此方法（请参见all.tags()方法），但对同一类型的元素集合而言它完全是一种冗余。

urns( )

IE *5(Win)* NN *n/a* Moz *n/a* Saf *n/a* Op *n/a* DOM *n/a*

```
urns(URN)
```

请参见all.urns()方法。

applet

IE *4* NN *3* Moz *all* Saf *all* Op *7* DOM *1*

applet对象对应于applet元素。须要注意的是，尽管此处并未罗列出对象特定方法，但applet通过LiveConnect为脚本暴露了其公共的属性和方法。如果在基于Mozilla的浏览器中遍历applet的属性和方法，那么就会发现加载至applet元素中的小程序拥有一长列的公共方法。

等价HTML元素 <applet>

对象模型引用方式

```
[window.]document.appletName
[window.]document.getElementById("elementID")
```

对象特定属性 align、alt、altHTML、archive、code、codeBase、dataFld、dataSrc、height、hspace、name、object、vspace、width

对象特定方法 无。

对象特定事件属性 无。

align

IE *4* NN *n/a* Moz *all* Saf *all* Op *7* DOM *1*

可读/可写

定义元素在其容器内的对齐方式。请参见第1章起始部分中“对齐常量”一节，该节对不同对齐常量所包含的不同含义给出了解释。

示例 `document.getElementById("myApplet").align = "center";`

值 任意一个对齐常量，包括：absbottom | absmiddle | baseline | bottom | left | middle | right | exttop | top。

默认值 left

alt

IE *6* NN *n/a* Moz *all* Saf *all* Op *7* DOM *1*

可读/可写

当对象或小程序加载失败时，会显示它提供的文本信息。但是在一个已经存在的applet对象上设置这个属性并不会产生任何可见的效果。

示例 `document.myApplet.alt= "Image Editor Applet";`

值 放置在引号之中的任意字符串，但浏览器并不会解析其中所包含的HTML标签。

默认值 无。

altHTML

IE *4* NN *n/a* Moz *n/a* Saf *n/a* Op *n/a* DOM *n/a*

可读/可写

当对象或小程序加载失败时，会显示它提供的HTML内容。它可以是一个文本消息、静态图片或任意其他适合于该场景的HTML内容。但是在一个已经存在的applet对象上设置这个属性并不会产生任何可见的效果。

示例 `document.myApplet.altHTML = "<img src='appletAlt.gif'>";`

值 放置在引号之中的任意字符串，可以包含HTML标签。

默认值 无。

archive

IE *6* NN *n/a* Moz *all* Saf *all* Op *7* DOM *1*

只读

此属性反映了applet元素的archive属性。目前浏览器仅实现了该属性的部分功能。请参见第1章中关于archive属性的讨论。

示例

```
if (document.applets["clock"].archive == "myClock.zip") {
    // 处理已获取的类文件
}
```

值 字符串形式的URI，注意区分大小写。

默认值 无。

code

IE *4* NN *n/a* Moz *all* Saf *all* Op *7* DOM *1*

只读

提供code属性对应的Java小程序类的文件名。

示例

```
if (document.applets["clock"].code == "XMAScounter.class") {
    // 处理已获取的类文件
}
```

值 小程序类的文件名称字符串，注意区分大小写。

默认值 无。

393

codeBase

IE 4 NN n/a Moz all Saf all Op 7 DOM 1

只读

提供code属性对应的Java小程序类文件的保存路径。要注意一点，codebase属性并不包含类文件的名称，它仅仅说明其所在路径。

示例

```
if (document.applets["clock"].codeBase == "classes") {
    // 处理已获取的类文件目录
}
```

值 区分大小写的路径名称，它往往与保存当前HTML文档的文件夹相关。

默认值 无。

dataFld

IE 4 NN n/a Moz n/a Saf n/a Op n/a DOM n/a

可读/可写

由于dataFld和dataSrc属性应用于单独的param元素，因此并不清楚如何在一个applet对像上使用这个属性。

值 数据源数据列内区分大小写的标识符。

默认值 无。

dataSrc

IE 4 NN n/a Moz n/a Saf n/a Op n/a DOM n/a

可读/可写

由于dataFld和dataSrc属性应用于单独的param元素，因此并不清楚如何在一个applet对像上使用这个属性。

值 数据源内区分大小写的标识符。

默认值 无。

height, width

IE 4 NN n/a Moz all Saf all Op 7 DOM 1

可读/可写

指出标签属性中所设置的元素高度与宽度，单位为像素。小程序被加载后，改变这两个值无法有效地改变其实际所占矩形空间的大小。

示例

```
var appletHeight = document.myApplet.height;
```

值 整数值。

默认值 无。

394

hspace, vspace

IE 4 NN n/a Moz all Saf all Op 7 DOM 1

可读/可写

指出一个小程序四周在水平和垂直方向上的边距值，单位为像素。hspace属性对元素左右两侧施加相等的影响，而vspace对元素上下两端的影响也是相同的。尽管这些边距与样式单设置的边距并不相同，但它们的实际效果却是一致的。

示例

```
document.getElementById("myApplet").hspace = 5;
document.getElementById("myApplet").vspace = 8;
```

值 整数像素值。

默认值 0

name

IE 4 NN n/a Moz all Saf all Op 7 DOM 1

只读

这是一个与小程序相关的标识符。如果要通过document.*appletName*的形式指代对象，就请使用这个名称。

值 区分大小写的标识符，命名时必须遵循标识符命名规则：不包含空白、不能以数字作为起始字符、除下划线外不允许使用其他标点符号。

默认值 无。

object

IE 4 NN n/a Moz all Saf all Op 7 DOM n/a

只读

返回一个小程序对象的引用，这样就可以使用脚本访问这个小程序的某个属性或方法，该属性或方法的名称与一个applet元素对象中某个属性或方法名称完全一致。但是这个功能只使用于IE浏览器，其他浏览器只会简单地返回一个空字符串。

值 IE浏览器将返回一个小程序对象（并非applet元素对象）引用，在其他提供支持的浏览器中，仅返回一个空字符串。

默认值 无。

vspace

参见hspace属性。

width

参见height属性。

applets

IE 3 NN 2 Moz all Saf all Op 7 DOM 1

当前文档内所有Java小程序的集合，这个集合会根据源代码顺序对这些对象进行排序。Navigator和Internet Explorer浏览器均允许使用数组符号来访问集合中的一个小程序对象，例如document.applets[0]、document.applets["clockApplet"]。IE浏览器还允许将索引值放置在一个圆括号之中，例如document.applets(0)。如果须要将小程序的名称作为索引值，那么一定要使用name属性的属性值，而不要使用id属性。如果须要通过小程序引用的id属性来访问对象，那么此时可以使用document.all.*elementID*（仅适用于IE浏览器）或document.getElementById("*elementID*")。

对象模型引用方式 document.applets[i]

对象特定属性 length

对象特定方法 item()、namedItem()、tags()

length

IE 3 NN 2 Moz all Saf all Op 7 DOM 1

只读

返回集合中元素的数量。

示例

```
var howMany = document.applets.length;
```

值 整数值。

item( )

IE 4 NN n/a Moz all Saf all Op 7 DOM 1

```
item(index[, subindex]) item(index)
```

根据符合索引值（IE中也可以使用*index*或*subindex*值）的元素对象，返回一个单独的小程序对象或一组小程序对象集合。

返回值 一个小程序对象或一个小程序对象集合。如果并不存在与参数相匹配的对象，则返回null。

参数

index

当输入参数是一个从零开始计数的整数时，返回值是源代码内与某个具体项相对应的独立元素（内嵌在当前元素之内），而如果参数为一个字符串，则返回id或name属性与该字符串相符的元素集合。

subindex

在IE中，如果为第1个参数指定一个字符串值，那么可以将第2个参数指定为一个从零开始计数的索引值，这样就可以从集合中找到一个id或name属性与第1个参数的字符串相匹配的具体元素。

namedItem( )

396

IE 6 NN n/a Moz all Saf all Op 7 DOM 1

```
namedItem(IDOrName)
```

返回单个小程序对象或小程序对象集合，它们对应于符合参数字符串值的元素。

返回值 一个小程序对象或一个小程序对象集合。如果并不存在与参数相匹配的对象，则返回null。

参数

IDOrName

与目标元素的id或name属性包含相同值的字符串。

tags( )

IE 4 NN n/a Moz n/a Saf all Op 7 DOM n/a

```
tags(tagName)
```

从当前集合的全部内嵌对象中，返回标签与*tagName*参数相匹配的对象集合。在所有的IE、Safari和Opera集合中均已实现此方法（请参见all.tags()方法），但对同一类型的元素集合而言它完全是一种冗余。

area

IE 4 NN n/a Moz all Saf all Op 7 DOM 1

area对象对应于area元素，它定义了一个客户端图像映射可点击区域的外形、坐标位置，以及链接目标。Navigator和Internet Explorer将area对象视为links集合中的一员，因此area对象的行为与链接很类似，只不过它是作用于一个图片分块。

等价HTML元素 <area>

对象模型引用方式

```
[window.]document.links[i]
[window.]document.getElementById("elementID")
```

对象特定属性 alt、coords、hash、host、hostname、href、noHref、pathname、port、protocol、search、shape、target

对象特定方法 无。

对象特定事件 无。

alt

397

IE 4 NN n/a Moz all Saf all Op 7 DOM 1

可读/可写

未来非图形化的浏览器能借助alt属性来显示图像热点的扼要说明。

示例 `document.getElementById("elementID").alt = "To Next Page";`

值 放置在引号之中的任意字符串。

默认值 无。

coords

IE 4 NN n/a Moz all Saf all Op 7 DOM 1

可读/可写

定义一个与特定链接或脚本动作相关的区域的外观。其坐标值是一系列由逗号分隔的数值列表。如果两个区域重叠在一起，则优先使用在代码中定义较早的区域。

示例 `document.getElementById("mapArea2").coords ="25, 5, 50, 70";`

值 每个坐标值都是一个单位为像素的长度值，但坐标的数值与它们的顺序都依赖于shape属性指定的形状，而这个形状可以与area元素任意组合。如果shape="rect"，则有4个坐标（左，上，右，下）；如果shape="circle"，则有3个坐标（圆心-x,圆心-y,半径）；如果shape="poly"，则有一组坐标值对，每对代表多边形的一个顶点（x1, y1, x2, y2, x3, y3,...xN, yN）。

默认值 无。

hash

IE 3 NN 2 Moz all Saf all Op 7 DOM 1

可读/可写

这是href属性的URL中的锚链部分（即“#”之后的部分），它指向文档中某个锚链位置。在设置属性值时，请不要包含“#”。

示例 `document.getElementById("mapArea2").hash = "section3";`

值 字符串。

默认值 无。

host

IE 3 NN 2 Moz all Saf all Op 7 DOM 1

可读/可写

对于区域链接而言，这个组合提供目标文档所在服务器的主机名与端口。如果端口是URL的一部分，那么主机和端口之间使用冒号进行分隔，这与它们在URL中的形式一样。如果在针对IE的HTTP URL中并未明确指定端口号，那么会自动返回默认端口号，即80。

示例 `document.getElementById("mapArea2").host = "www.megacorp.com:80";`

值 主机名字符串，可在其后加上冒号和端口号。

默认值 由服务器决定。

hostname

IE 3 NN 2 Moz all Saf all Op 7 DOM 1

可读/可写

对链接而言，这是目标文档所在服务器的主机名。须要注意的是，hostname属性并不包括端口号。

示例 `document.areas[2].hostname = "www.megacorp.com";`

值 主机名字符串（主机与域）。

默认值 由服务器决定。

href

IE 3 NN 2 Moz all Saf all Op 7 DOM 1

可读/可写

这是由元素的href属性所指定的URL。

示例 `document.areas[2].href = "http://www.megacorp.com";`
值 由完整或相对URL路径组成的字符串。
默认值 无。

noHref

IE *4* NN *n/a* Moz *all* Saf *all* Op *7* DOM *1*

可读/可写

它指明由坐标定义的区域是否拥有一个相关的链接。一旦将此属性设置为`true`，则支持脚本的浏览器就不会将此`area`元素视为一个链接。

示例 `document.areas[4].noHref = "true";`
值 布尔值：`true | false`。
默认值 `false`

pathname

IE *3* NN *2* Moz *all* Saf *all* Op *7* DOM *1*

可读/可写

根据元素中`href`属性所对应的URL，提供其中的路径名部分。它由URL域名末字符之后的全部有用信息组成，其中包括位于首位的斜线符号。

示例 `document.getElementById("myLink").pathname = "/images/logoHiRes.gif";`
值 字符串。
默认值 无。

port

IE *3* NN *2* Moz *all* Saf *all* Op *7* DOM *1*

可读/可写

根据元素的`href`属性所对应的URL，提供其中的端口号部分。它包含URL域名末字符之后紧接着冒号的全部有用信息。冒号并不是`port`属性值的一部分。

示例 `document.getElementById("myLink").port = "80";`
值 字符串形式的数字值。
默认值 无。

protocol

IE *3* NN *2* Moz *all* Saf *all* Op *7* DOM *1*

可读/可写

根据元素内`href`属性所对应的URL，此属性提供其中的协议内容。它包括URL内从起始部分到第一个冒号为止的所有信息。典型的协议形式包括`"http:"`、`"file:"`、`"ftp:"`和`"mailto:"`。

示例 `document.getElementById("secureLink").protocol = "https:";`
值 字符串。
默认值 无。

search

IE *3* NN *2* Moz *all* Saf *all* Op *7* DOM *1*

可读/可写

根据`href`属性对应的URL，提供其中以“`?`”开头的编码部分。作为查询结果的文档可能也包含对应的查询部分，并将它作为`window.location`属性的一部分。在使用中也可以通过脚本来修改这个属性。这样一来，

就可以将URL和搜索标准同时发送至服务器端。但是，必须预先知晓服务器所需的数据格式（通常是一些名称/值对），才能正确地实现这一功能。

示例 `document.getElementById("searchLink").search="?p=Tony+Blair&d=y&g=0&s=a&w=s&m=25";`

值 以“?”开头的字符串。

默认值 无。

shape

IE *4* NN *n/a* Moz *all* Saf *all* Op *7* DOM *1*

400

可读/可写

此属性定义了服务器端图像映射区域的外形，具体的坐标由coords属性指定。

示例 `document.getElementById("area51").shape = "circle";`

值 不区分大小写的外形名称常量字符串，包括：default、rect、rectangle、circle、poly、polygon。

默认值 RECT（IE）；空字符，但仍表示rect（Mozilla）。

target

IE *3* NN *2* Moz *all* Saf *all* Op *7* DOM *1*

可读/可写

提供某个窗口或框架的名称，该窗口或框架将接收区域链接操作后返回的结果内容。这些名称会通过frame元素的name属性分配给对应框架，对子窗口而言，则会通过window.open()方法的第2个参数来指定其名称。如果须要使用target属性来指定链接的打开页面，并且还须要使用标记来通过严格HTML或XHTML DTD的验证，那么可以忽略源代码中的target属性，但必须在页面加载之后使用脚本为area元素的target属性设置所需的值。

示例 `document.getElementById("homeArea").target = "_blank";`

值 窗口或框架的名称字符串，或者下述字符串常量之一：_parent、_self、_top、_blank。_parent以当前文档所在的框架集作为目标；_self以当前窗口作为目标；_top以浏览器主窗口为目标，因此会删除所有的其他框架；而_blank会以默认尺寸创建一个新的窗口。

默认值 无。

areas

IE *4* NN *n/a* Moz *all* Saf *all* Op *7* DOM *1*

与一个map元素的相关所有area元素组成的集合。须要注意的是，areas集合中的每一项同时也是文档范围内links集合（document.links[]数组）的成员之一。它们的区别在于，areas集合中的每一项均位于同一个map元素之中。

对象模型引用方式 `document.getElementById("mapElementID").areas`

对象特定属性 `length`

对象特定方法 `item()、namedItem()、tags()、urns()`

length

IE *4* NN *n/a* Moz *all* Saf *all* Op *7* DOM *1*

401

只读

返回集合中元素的数量。

示例 `var howMany = document.areas.length;`

值 整数值。

item( )

IE 4 NN n/a Moz all Saf all Op 7 DOM 1

```
item(index[, subindex]) item(index)
```

根据符合索引值（IE中也可以使用index或subindex值）的元素对象，返回一个单独的area对象或一组area对象集合。

返回值 一个area对象或一个area对象集合。如果并不存在与参数相匹配的对象，则返回null。

参数

index

当输入参数是一个从零开始计数的整数时，返回值是源代码内与某个具体项相对应的独立元素（内嵌在当前元素之内），而如果参数为一个字符串，则返回id或name属性与该字符串相符的元素集合。

subindex

在IE中，如果为第1个参数指定一个字符串值，那么可以将第2个参数指定为一个从零开始计数的索引值，这样就可以从集合中找到一个id或name属性与第一个参数的字符串相匹配的具体元素。

namedItem( )

IE 6 NN n/a Moz all Saf all Op 7 DOM 1

```
namedItem(IDOrName)
```

返回单个area对象或area对象集合，它们对应于符合参数字符串值的元素。

返回值 一个area对象或一个area对象集合。如果并不存在与参数相匹配的对象，则返回null。

参数

IDOrName

与目标元素的id或name属性包含相同值的字符串。

tags( )

IE 4 NN n/a Moz n/a Saf all Op 7 DOM 1

```
tags(tagName)
```

从当前集合的全部内嵌对象中，返回标签与*tagName*参数相匹配的对象集合。在所有的IE、Safari和Opera集合中均已实现此方法（请参见all.tags()方法），但对同一类型的元素集合而言它完全是一种冗余。

urns( )

IE 5(Win) NN n/a Moz n/a Saf n/a Op n/a DOM n/a

```
urns(URN)
```

请参见all.urns()方法。

Attr, attribute

IE 4 NN n/a Moz all Saf all Op 7 DOM 1

一个元素属性的名称/值对可以表示为一个对象，在W3C DOM中，这个对象称之为Attr对象，而在IE的术语中，称之为attribute对象。它们只不过是针对同一个对象的不同名称而已。在这两个环境下，均可以通过document.createAttribute()方法来创建属性对象，然后就可以将这个属性对象的引用作为元素setAttributeNode()方法的一个参数，从而将这个属性对象插入元素之中。例如：

```
var newAttr = document.createAttribute("author");
newAttr.value = "William Shakespeare";
document.getElementById("hamlet").setAttributeNode(newAttr);
```

某些W3C DOM元素方法会返回属性对象（最常用的是getAttributeNode()方法），此时可以按照处理脚本对象的方法来访问这些对象的属性。

在W3C DOM的抽象模式中，Attr对象继承了Node对象所有的属性与方法。然而直到IE 6/Windows为止，attribute对象仍未继承Node对象的某些属性，但IE 5却已经在元素和文本节点中实现了这些属性。

等价HTML元素 起始标签内的任意名称/值对。

对象模型引用方式

```
[window.]document.getElementById("elementID").attributes[i]
[window.]document.getElementById("elementID").attributes.item(i)
[window.]document.getElementById("elementID").attributes.getNamedItem[attrName]
```

对象特定属性 expando、name、ownerElement、specified、value

对象特定方法 无。

对象特定事件 无。

expando

IE 6 NN n/a Moz n/a Saf n/a Op n/a DOM n/a

只读

如果被插入到元素中的某个属性并不是该元素的原生属性，那么返回布尔值true。对于一个使用document.createAttribute()方法创建的属性而言，在该属性被加入到元素之前，expando属性的值为false。加入到元素之后，会在元素原生属性的上下文环境内重新评估expando的值。

示例

```
var isCustomAttr = document.getElementById("book3928").getAttributeNode("author").expando;
```

值 布尔值：true | false。

默认值 false

name

IE 5 NN n/a Moz all Saf all Op 7 DOM 1

只读

这是属性的名称/值对中的名称部分。它等价于Attr对象的nodeName属性。由于与此属性的相关内容较多，因此请不要使用脚本随意修改一个属性的名称，否则会使文档树变得混乱。如果的确要对属性进行修改，可以使用一个新创建的属性代替原有属性，此时新属性的名称将作为document.createAttribute()方法的一个必要的参数。

示例

```
if (myAttr.name == "author") {
    // 处理"author"属性
}
```

值 字符串。

默认值 尽管创建一个新属性需要一个名称，但默认值仍然为空字符串。

ownerElement

IE n/a NN n/a Moz all Saf all Op 7 DOM 2

只读

指向包含当前属性对象的元素。在向元素插入一个新创建的属性之前，它的值为null。

示例

```
if (myAttr.ownerElement.tagName == "fred") {
    // 处理<fred>元素的属性
}
```

值 元素节点的引用。

默认值 null

specified

IE 5 NN n/a Moz all Saf all Op 7 DOM 1

只读

如果在源代码中明确指定了属性值，或使用脚本调整了属性值，那么返回布尔值true。即使使用文档的DTD定义了属性的默认值，但如果浏览器（IE）未发现明确设置的属性值，那么该属性所对应的specified值仍然为false。W3C DOM Level 2建议将新创建的Attr对象的specified属性设置为true，但IE 6+及Mozilla会将它设置为false，直到该属性被被插入到一个元素之中。

404

示例

```
if (myAttr.specified) {
   // 如果属性值不等于 DTD 默认值，则进行处理
}
```

值 布尔值：true | false。

默认值 false

value

IE 6 NN n/a Moz all Saf all Op 7 DOM 1

可读/可写

提供属性的名称/值对中的属性值部分。它和Attr节点的nodeValue属性完全一致，与元素的getAttribute()和setAttribute()方法一样，这是一种更为直接的数据操作方式。如果创建了一个新的属性对象，那么在将此属性插入元素之前，都可以通过value指定这个属性的值。属性节点的值通常为字符串形式，但IE不仅可以使用字符串，还可以使用数字或布尔量作为这些属性的数据类型。

示例

```
document.getElementById("hamlet").getAttributeNode("author").value = "Shakespeare";
```

值 字符串。

默认值 空字符串。但在IE/Windows中，会返回“undefined”（并不存在名为“undefined”的类型）。

attributes, NamedNodeMap

IE 5 NN n/a Moz all Saf all Op 7 DOM 1

每个W3C DOM元素对象的attributes属性均会返回一个由元素内Attr对象引用组成的集合对象。属性节点总有一个与之相关的名称，使得节点的诸多方法可以直接通过名称访问这些属性，从而避免为找到相匹配的节点名称而遍历整个集合。在W3C DOM架构中，有一个名为NamedNodeMap的对象抽象地表示了由已命名的节点组成的数组，这个NamedNodeMap还共享了一些属于IE attributes对象的属性和方法。由于这两个对象均指向文档树中的同一部分，因此此处对它们等同视之。另外两种W3C DOM集合也是NamedNodeMap对象的变异形式，但在HTML文档中主要使用的NamedNodeMap却是一个Attr对象集合。另外，此集合中的成员按照它们在源代码中的次序进行排序。

当然，还有更为直接的方式来访问元素中的属性，例如getAttribute()或getAttributeNode()方法。然而，此处列出的属性与方法均假设已通过脚本获取了一个独立于宿主元素的属性集合，并且据此执行其他操作。因此对这些属性与方法进行说明还是很有必要的。

405

对象模型引用方式 *elementReference*.attributes

对象特定属性 length

对象特定方法 getNamedItem()、getNamedItemNS()、item()、removeNamedItem()、removeNamedItemNS()、setNamedItem()、setNamedItemNS()

对象特定事件 无。

length

IE 5 NN n/a Moz all Saf all Op 7 DOM 1

只读

返回集合中元素的数量。

示例 `var howMany = document.getElementById("myTable").attributes.length;`

值 整数值。

getNamedItem( )

IE 6 NN n/a Moz all Saf all Op 7 DOM 1

`getNamedItem("attributeName")`

根据参数值获取节点名称与之相符的属性，并返回对应的Attr对象。

返回值 一个Attr对象的引用。如果并不存在与参数相匹配的对象，则返回null。

参数

attributeName

与属性的名称/值对中名称部分相对应的字符串。

getNamedItemNS( )

IE n/a NN n/a Moz all Saf all Op 7 DOM 2

`getNamedItemNS("namespaceURI", "localName")`

返回一个与输入参数拥有相同的本地名称和命名空间URI的Attr对象。

返回值 一个Attr对象的引用。如果并不存在与参数相匹配的对象，则返回null。

参数

namespaceURI

URI字符串，它能够与文档中指定给某个标注的URI相匹配。

localName

属性的本地名称部分。

item( )

IE 5 NN n/a Moz all Saf all Op 7 DOM 1

`item(index)`

返回一个Attr对象，它对应于一个与索引值相匹配的元素。

返回值 一个Attr对象的引用。如果与索引值相匹配的对象不存在，则返回null。与IE中某些集合不同，不允许在attributes对象中使用字符串形式的索引值。

参数

index

一个从零开始计数的整数，依照源代码次序，对应于集合中的某个特定项。

removeNamedItem( )

IE 6 NN n/a Moz all Saf all Op 7 DOM 1

`removeNamedItem("attributeName")`

根据参数值获取节点名称与之相符的属性，然后从集合中删除对应的Attr对象。

返回值 被删除的Attr对象的引用。如果并不存在与参数相匹配的对象，则返回null。

参数

attributeName

与属性的名称/值对中名称部分相对应的字符串。

removeNamedItemNS( )

IE *n/a* NN *n/a* Moz *all* Saf *all* Op *7* DOM *2*

```
removeNamedItemNS("namespaceURI", "localName")
```

根据参数值获取本地名称和命名空间均与之匹配的Attr对象，并从集合中删除该对象。

返回值 被删除的Attr对象的引用。如果与参数相匹配的对象不存在，此方法会产生一个错误。

参数

namespaceURI

URI字符串，它与文档中指定给某个标注的URI相对应。

localName

属性的本地名称部分。

setNamedItem( )

IE *6* NN *n/a* Moz *all* Saf *all* Op *7* DOM *1*

```
setNamedItem(attrObjectReference)
```

将一个Attr对象插入当前属性集合。如果插入的目标是一个已存在的元素，那么也可以在该元素上使用setAttributeNode()方法来插入这个Attr对象。在调用setNamedItem()方法时，浏览器会首先检查新属性的名称是否能够与现存属性名称相匹配。如果存在匹配项，那么新属性将替换原有属性，否则直接在集合中添加这个新属性。

返回值 一个Attr对象的引用，它既可以来自于一个新建的对象，也可以来自于文档树中其他的对象。

参数

attrObjectReference

由document.createAttribute()方法创建的Attr节点对象的引用，或者来自于文档树中另一个元素的Attr节点的引用。

setNamedItemNS( )

IE *n/a* NN *n/a* Moz *all* Saf *all* Op *7* DOM *2*

```
setNamedItemNS(attrObjectReference)
```

将一个Attr对象插入当前属性集合之中。如果插入的目标是一个已存在的元素，那么也可以在该元素上使用setAttributeNodeNS()方法来插入这个Attr对象。在调用setNamedItemNS()方法时，浏览器会首先检查能否在集合中找到一个与新属性的本地名称和命名空间URI组合相匹配的现存属性组合。如果存在匹配组合，那么新属性将替换原有属性，否则直接在集合中添加这个新属性。

返回值 一个Attr对象的引用，它既可以来自于一个新建的对象，也可以来自于文档树中其他的对象。

参数

attrObjectReference

由document.createAttributeNS()方法创建的Attr节点对象的引用，或来自于文档树中另一个元素的Attr节点的引用。

Audio

IE *n/a* NN *n/a* Moz *n/a* Saf *n/a* Op *9* DOM *n/a*

Audio对象是一个仅适用于脚本的控件，WHATWG控制着这个对象的标准规范。它的目的是为脚本提供一种标准途径，以便加载、播放并停止背景音乐，而Opera 9则是首个支持它的主流浏览器。由于浏览器不会自动显示插件控制面板，因此须要页面开发者提供相关的控制。

如果要使用这个对象，请先调用构造方法来创建一个实例，并传入唯一的一个参数，即须要加载的声音文件的URL。Opera 9目前支持.wav文件格式，如下所示：

```
var aud = new Audio("sample.wav");
```

然后，可以使用load事件来触发play()方法。

对象模型引用方式 new Audio("*audioFile*").

对象特定属性 无。

对象特定方法 loop()、play()、stop()

对象特定事件

事件	IE	Mozilla	Safari	Opera	W3C DOM
error	—	—	—	·	—
load	—	—	—	·	—

loop( )

IE *n/a* NN *n/a* Moz *n/a* Saf *n/a* Op *9* DOM *n/a*

```
loop([count])
```

调用play()方法之后，这个属性可以控制已加载的声音文件的播放次数。如果参数为空，就会一直循环播放这个声音文件。

返回值 无。

参数

count

一个控制声音文件播放次数的整数。

play( ), stop( )

IE *n/a* NN *n/a* Moz *n/a* Saf *n/a* Op *9* DOM *n/a*

在Audio对象实例的生存期内，控制播放的开始与结束。在播放开始之前，必须完整加载声音文件。停止播放之后，播放计数被重置为零，并且播放点也被设置到文件的初始位置。

返回值 无。

参数 无。

b, big, i, s, small, strike, tt, u

IE *4* NN *n/a* Moz *all* Saf *all* Op *7* DOM *1*

所有这些对象都分别对应于一个与HTML字体样式相关的同名元素。这些元素中的每一个都指定了一种显示样式以控制一段行内内容，否则这些行内内容将由每个浏览器默认的内部样式单规则控制。HTML 4不再赞成使用这些元素，而推荐使用样式单属性。请参看第1章中对它们的相关说明，以获取更多详细信息。从脚本操作的角度看，这些与字体相关的元素对象共享着相同的属性、方法、事件处理程序和集合。

等价HTML元素 <b>、<big>、<i>、<s>、<small>、<strike>、<tt>、<u>

对象模型引用方式 [window.]document.getElementById("*elementID*")

对象特定属性 无。

对象特定方法 无。

对象特定事件 无。

base

IE *4* NN *n/a* Moz *all* Saf *all* Op *7* DOM *1*

base对象可对当前文档的URL路径进行控制。对于文档中用于指定不同的src和href属性的相对URL而言，此路径是它们的基准路径。

等价HTML元素 <base>
对象模型引用方式 [window.]document.getElementById("*elementID*")
对象特定属性 href、target
对象特定方法 无。
对象特定事件 无。

href

IE 4 NN n/a Moz all Saf all Op 7 DOM 1

可读/可写

提供一个文档的URL路径，该URL的服务器路径会作为当前文档中所有相对路径的基准路径。一般来说，其属性值就是当前文档的URL路径，但为了便于组织文档和文件夹，它也可以使用其他路径。

示例 document.getElementById("myBase").href = "http://www.megacorp.com";
值 由完整或相对URL路径组成的字符串。
默认值 当前文档的路径名。

target

IE 4 NN n/a Moz all Saf all Op 7 DOM 1

可读/可写

提供某个窗口或框架的名称，该窗口或框架将接收链接操作后返回的结果内容，或者加载页面上其他操作所引入的文档。这些名称会通过frame元素的name属性分配给对应框架，对子窗口而言，则会通过window.open()方法的第2个参数来指定其名称。如果须要使用target属性来指定链接的打开页面，并且还须要使用标记来通过严格HTML或XHTML DTD的验证，那么可以忽略源代码中的target属性，但必须在页面加载之后使用脚本为base元素的target属性设置所需的值。

示例 document.getElementById("myBase").target = "_blank";
值 窗口或框架的名称字符串或下述字符串常量之一：_parent | _self | _top | _blank。_parent以当前文档所在的框架集作为目标；_self以当前窗口作为目标；_top以浏览器主窗口为目标，因此会删除所有的其他框架；而_blank会以默认尺寸创建一个新的窗口。
默认值 _self

basefont

IE 4 NN n/a Moz all Saf all Op 7 DOM 1

basefont元素用于向浏览器提示一些字体信息，它们将会作为当前页面内basefont元素之后所有文本渲染的依据。basefont元素替换了浏览器的用户首选项中定义的默认字体设置。

如果要通过脚本改变这个属性，只能通过本书中列出的属性或使用其他W3C DOM兼容的文档树操作。其他方法可能影响文档的正常显示，或者不允许在其他浏览器中使用。

等价HTML元素 <basefont>
对象模型引用方式 [window.]document.getElementById("*elementID*")
对象特定属性 color、face、size
对象特定方法 无。
对象特定事件 无。

color

IE 4 NN n/a Moz all Saf all Op 7 DOM 1

可读/可写

此属性能够为basefont元素之后的文本设置字体颜色。

示例 `document.getElementsByTagName("basefont")[0].color = "#c0c0c0";`

值 一个十六进制的三元组或明语颜色名称字符串，均不区分大小写。目前可用的明语颜色名称请参见附录A。

默认值 浏览器的默认值。

face

IE *4* NN *n/a* Moz *all* Saf *all* Op *7* DOM *1*

可读/可写

对于那些受basefont元素约束的部分，可以使用此属性为它们设置一组字体作为默认字体。这组字体名称使用逗号进行分隔，浏览器从它们中的第1个开始，逐个与客户端系统中的字体进行比对，以便设置默认字体，如果最终仍然找不到对应字体就会使用浏览器的默认字体。因此face中指定的字体名称必须与系统中的字体名称完全匹配。

示例 `document.getElementById("myBaseFont").face = "Bookman, Times Roman, serif";`

值 包含一个或多个字体名称的字符串，字体间使用逗号进行分隔。可以使用真实的字体名称或可识别的通用字体：`serif | sans-serif | cursive | fantasy | monospace`。

默认值 浏览器的默认值。

size

IE *4* NN *n/a* Moz *all* Saf *all* Op *7* DOM *1*

可读/可写

提供字体大小，它的取值范围来自于浏览器定义的相对尺寸级别（1–7）。

示例 `document.getElementById("myBaseFont").size = "+1";`

值 既可以是一个整数（使用引号括起来一个字符串），也可以是一个由“+”或“-”和整数组成的相对数值（必须用引号括起来）。

默认值 3

bdo

IE *6* NN *n/a* Moz *all* Saf *all* Op *7* DOM *1*

如果文本处理过程中使用了不同的转换方式，必须明确地关闭双向算法，那么bdo元素就适用于这种情况。这个对象的主要属性为dir，所有其他类型的元素对象均共享这个属性。

等价HTML元素 <bdo>

对象特定属性 无。

对象特定方法 无。

对象特定事件 无。

bgsound

IE *4* NN *n/a* Moz *n/a* Saf *n/a* Op *n/a* DOM *n/a*

bgsound元素定义了一个声音文件，当用户访问这个页面时会在后台播放这段录音。对其属性进行设置，就可以在加载声音文件后控制音量和重播次数。Windows版本暴露了少量属性，例如innerHTML和innerText，但是如果未使用结束标签，那么就无法在元素中应用这些属性。

等价HTML元素 <bgsound>

对象模型引用方式 [window.]document.getElementById("*elementID*")

对象特定属性 balance、loop、src、volume

对象特定方法 无。

对象特定事件 无。

balance

IE *4* NN *n/a* Moz *n/a* Saf *n/a* Op *n/a* DOM *n/a*

只读

指明如何在左右两个声道的上进行声音输出。一旦对此属性赋值，则不能通过脚本修改其属性值。

示例
```
var currBal = document.getElementsByTagName("bgsound")[0].balance;
```

值 介于—10 000和+10 000之间的带符号整数。如果取值为0，就代表两边的声音输出达到平衡。而负值表示左侧输出占主导地位，而正值则相反，表示右侧占主导地位。

默认值 0

loop

IE *4* NN *n/a* Moz *n/a* Saf *n/a* Op *n/a* DOM *n/a*

可读/可写

指明了声音的播放次数。如果其值等于－1，则意味着该音乐会一直播放，直到页面被卸载。

示例
```
document.getElementById("mySound").loop = 3;
```

值 整数值。

默认值 1

src

IE *4* NN *n/a* Moz *n/a* Saf *n/a* Op *n/a* DOM n/a

可读/可写

提供将要播放的声音文件的URL地址。指定一个新的URL给此属性后，会改变正在播放的曲调。新曲子的播放仍然受loop属性设定的控制。

示例
```
document.getElementById("tunes").src = "sounds/blues.aif";
```

值 字符串形式的完整或相对URL。

默认值 无。

volume

IE *4* NN *n/a* Moz *n/a* Saf *n/a* Op *n/a* DOM *n/a*

可读/可写

指明了背景声音播放时的相对音量，具体的声音大小与客户计算机中用户选择的最大声音输出等级有关。此处最大的音量值为0，但即使设置为0，播放出的声音也只能和**声音**控制面板里设置的音量一样大。此属性值最小可以调节到－10 000，但大多数用户设置的音量均高于此值。

示例
```
var currVolume = document.getElementById("themeSong").volume;
```

值 整数值。

默认值 随操作系统和声音设置的不同而发生改变。

big

参见b。

blockquote

IE *4* NN *n/a* Moz *all* Saf *all* Op *7* DOM *1*

blockquote对象对应于blockquote元素，该元素主要用于在文档内设置一个较长的块级引述内容。

等价HTML元素 <blockquote>
对象模型引用方式 [window.]document.getElementById("*elementID*")
对象特定属性 cite
对象特定方法 无。
对象特定事件 无。

cite

IE *5(Mac)/6(Win)* NN *n/a* Moz *all* Saf *all* Op *7* DOM *1*

可读/可写

提供一个在线原文档的URL地址，引用语就出自此文档。值得注意是，设置此属性根本无法从另一个文档中复制或摘录任何内容。而且目前主流浏览器也未对这一信息做出任何特殊操作。

值 一个指向网络上某文档的有效URL链接，既可使用相对URL，也可以使用绝对URL地址。
默认值 无。

body

IE *4* NN *n/a* Moz *all* Saf *all* Op *7* DOM *1*

body对象对应于body元素，它与document对象截然不同。body对象仅涉及元素自身及其内嵌内容。在一个HTML页面中，只允许存在一个body元素，因此IE和W3C DOM均为该对象引用的获取提供了一种快捷的途径，即document.body。此处列出的事件处理程序看起来就像是<body>标签中的属性，但它们实际上是文档级的事件（最好以document.*eventName*的形式来引用它们）。尽管在IE/Mac中并不是所有的元素对象均共享了客户端及滚动相关的属性集合，但body对象则定义了这类属性。

为了在IE 6/Windows中建立标准兼容模式，Microsoft用body元素的clientHeight和clientWidth属性来获取Netscape中等价的window.innerHeight和window.innerWidth属性。在标准兼容性模式中（此时document.compatMode == "CSS1Compat"），必须使用html元素的clientHeight和clientWidth来获取这些值。可以使用下列有效的快捷引用方式：

```
document.body.parentNode.clientHeight
document.body.parentNode.clientWidth
```

虽然body元素中大多数独有的属性均用于影响外观样式，但可以通过应用于body选择器的CSS规则来进行更好的控制。

等价HTML元素 <body>
对象模型引用方式 [window.]document.body
对象特定属性 aLink、background、bgColor、bgProperties、bottomMargin、leftMargin、link、noWrap、rightMargin、scroll、text、topMargin、vLink
对象特定方法 createControlRange()、createTextRange()
对象特定事件

事件	IE (Win)	Mozilla	Safari	Opera	W3C DOM
afterprint	·	—	—	—	—
beforeprint	·	—	—	—	—
beforeunload	·	—	—	—	—
load	·	·	·	·	·
select	—	·	·	—	—
unload	·	·	·	·	·

aLink

IE 4 NN n/a Moz all Saf all Op 7 DOM 1

可读/可写

指定一个超文本链接的颜色，链接被点击时就会显示该颜色。设置此属性后，链接文字、图像的边框或嵌入a元素之中的对象均会应用这个颜色。请参看link和vLink属性以了解未访问和已访问链接的颜色。这个属性的后向兼容版本是document对象的alinkColor属性，但并不建议使用alinkColor属性。在当前的实际应用中，它已经被:active伪类所取代。

示例 `document.body.aLink = "green";`

值 一个十六进制的三元组或明语颜色名称。目前可用的明语颜色名称请参见附录A。

默认值 `#0000FF`

416

background

IE 4 NN n/a Moz all Saf all Op 7 DOM 1

可读/可写

提供文档背景图片的URL地址。如果同时为元素设置了bgColor，那么当图片加载失败时才会出现背景色，否则图片将覆盖背景色。

示例 `document.body.background = "images/watermark.jpg";`

值 背景图片文件的完整或相对URL。

默认值 无。

bgColor

IE 4 NN n/a Moz all Saf all Op 7 DOM 1

可读/可写

为元素提供背景色。须要注意的是，即使使用明语颜色名称设置了元素或对象的bgcolor属性，其返回值总是一个十六进制三元组。

示例 `document.body.bgColor = "yellow";`

值 一个十六进制的三元组或明语颜色名称。目前可用的明语颜色名称请参见附录A。

默认值 由浏览器和操作系统共同决定。

bgProperties

IE 4 NN n/a Moz n/a Saf n/a Op n/a DOM n/a

可读/可写

指定了背景图片是保持在一个固定位置还是在用户滚动页面时随之一起滚动。将背景图保持在一个固定位置后，此时的屏幕显示非常有趣，滚动的内容滑过背景图时就像电影的致谢画面在背景图上滚动播放。

示例 `document.body.bgProperties = "fixed";`

值 字符串（表示随页面一起滚动），或字符串常量“`fixed`”。

默认值 字符串。

bottomMargin

IE 4 NN n/a Moz n/a Saf n/a Op n/a DOM n/a

可读/可写

指明了文档内容的最底端与可滚动页面底部之间的空白空间大小。当页面内容的长度或窗口大小使得窗口并不须要滚动时，设置这个属性并不会带来任何可视效果。使用默认值时，内容的底部会充满页面的底部，但在Macintosh版本的IE浏览器中，即使就此属性值设置为0，两者之间也存在着大约10像素的距离。一般说来，此值越大效果越明显。这个属性为设置body元素对象的marginBottom样式单属性提供了一种较为快捷的替代方法。

417

示例 `document.body.bottomMargin = 20;`
值 一个整数值（大于等于零），表示页面底部所需空白区域的像素值。
默认值 `0`

leftMargin

IE *4* NN *n/a* Moz *n/a* Saf *n/a* Op *n/a* DOM *n/a*

可读/可写

提供浏览器窗口或框架内body元素左边距的宽度，单位为像素。在默认情况下，浏览器会插入一小段空白以防止元素内容紧贴着窗口的左边缘。如果将属性值设置为一个空字符串，就等效于将它设置为0。

示例 `document.body.leftMargin = 16;`
值 整数像素值。
默认值 `10`（Windows）；`8`（Macintosh）。

link

IE *4* NN *n/a* Moz *all* Saf *all* Op *7* DOM *1*

可读/可写

此元素设定超文本链接尚未被访问时的颜色，此时链接的URL地址尚不在浏览器的缓存之中。链接存在三种状态：未访问（unvisited）、激活（active）和已访问（visited），这个状态是其中之一。设置这个属性后，链接文字、图像的边框，以及嵌入a元素之中的对象均会应用这个颜色。设置document对象的linkColor属性与这个属性具有相同的效果。在当前的实际应用中，它已经被:link伪类所取代。

示例 `document.body.link = "#00FF00";`
值 一个十六进制的三元组或明语颜色名称。目前可用的明语颜色名称请参见附录A。
默认值 `#0000FF`

noWrap

IE *4* NN *n/a* Moz *n/a* Saf *n/a* Op *n/a* DOM *n/a*

可读/可写

指明浏览器是否应该按照需要加宽主体内容的宽度，以便将一行不间断的文本显示在一整行上。当单元格内内容很多时，如果滥用此属性，那么为了显示所有内容，会在页面上出现一个非常不便于使用的横向滚动条。

示例 `document.body.noWrap = "true";`
值 布尔值：`true | false`。
默认值 `false`

rightMargin

IE *4* NN *n/a* Moz *n/a* Saf *n/a* Op *n/a* DOM *n/a*

可读/可写

提供浏览器窗口或框架内body元素右边距的宽度，单位为像素。在默认情况下，浏览器会插入一小段空白以防止元素内容紧贴着窗口的右边缘（除Macintosh之外）。如果将属性值设置为一个空字符串，就等效于将它设置为0。

示例 `document.body.leftMargin = 16;`
值 整数像素值。
默认值 `10`（Windows）；`0`（Macintosh）。

scroll

IE *4* NN *n/a* Moz *n/a* Saf *n/a* Op *n/a* DOM *n/a*

可读/可写

当内容空间超过当前窗口（或框架）大小时，控制是否在窗口（或框架）上显示滚动条。如果文档指定了一

个标准兼容的DOCTYPE定义，那么scroll属性就不会为body元素而对其属性值的改变做出响应。实际情况并非像微软开发者文档所声称的那样，html元素对象此时并不能从这个属性上获得多少收益。

示例 `document.body.scroll = "no";`

值 并非是一个完全的布尔值。需要以下字符串值之一：`yes | no | auto`。

默认值 `yes`

text

IE *4* NN *n/a* Moz *all* Saf *all* Op *7* DOM *1*

可读/可写

为整个文档主体指定文本颜色。它与前景色等价，但在实际应用中已经逐渐被CSS的color属性所取代。

示例 `document.body.text = "darkred";`

值 一个十六进制的三元组或明语颜色名称。目前可用的明语颜色名称请参见附录A。

默认值 由浏览器决定默认值，也可以按用户需要进行修改。

topMargin

IE *4* NN *n/a* Moz *n/a* Saf *n/a* Op *n/a* DOM *n/a*

可读/可写

提供浏览器窗口或框架内body元素上边距的宽度，单位为像素。在默认情况下，浏览器会插入一小段空白以防止元素内容紧贴着窗口的上边缘。如果将属性值设置为一个空字符串，就等效于将它设置为0。

示例 `document.body.topMargin = 16;`

值 整数像素值。

默认值 15（Windows）；8（Macintosh）。

vLink

IE *4* NN *n/a* Moz *all* Saf *all* Op *7* DOM *1*

可读/可写

为刚被访问的超文本链接指定颜色。设置这个属性后，链接文字、图像的边框，以及嵌入a元素之中的对象均会应用这个颜色。请参看link和aLink属性以了解未访问和已访问链接的颜色。这个属性的后向兼容版本是document对象的vlinkColor属性，不建议使用这个属性。在当前的实际应用中，它已经被:vlink伪类所取代。

示例 `document.body.vLink = "gold";`

值 一个十六进制的三元组或明语颜色名称。目前可用的明语颜色名称请参见附录A。

默认值 由浏览器和操作系统共同决定。

createControlRange( )

IE *5(Win)* NN *n/a* Moz *n/a* Saf *n/a* Op *n/a* DOM *n/a*

当元素内容处于用户编辑状态时，此方法会创建一个controlRange集合，即当前选中的所有相邻元素的集合。

返回值 controlRange集合。

参数 无。

createTextRange( )

IE *4* NN *n/a* Moz *n/a* Saf *n/a* Op *n/a* DOM *n/a*

从当前元素的显示文本内容中创建一个TextRange对象。请参见TextRange对象以获取更多详细信息。

返回值 TextRange对象。

参数 无。

br

IE 4 NN n/a Moz all Saf all Op 7 DOM 1

br对象对应于br元素。

等价HTML元素

对象模型引用方式 [window.]document.getElementById("*elementID*")

对象特定属性 clear

对象特定方法 无。

对象特定事件 无。

clear

IE 4 NN n/a Moz all Saf all Op 7 DOM 1

可读/可写

如果当前文本环绕在一个浮动图像或其他对象周围，那么它就可以告知浏览器如何处理br元素之后的下一行文本。其属性值既取决于嵌入了一个或多个图像的页面，也取决于下一行文本相对于这些图像的期望位置。

示例 document.getElementById("specialBreak").clear = "all";

值 以下字符串常量之一，不区分大小写：all | left | none | right。

默认值 none

button

IE 4 NN n/a Moz all Saf all Op 7 DOM 1

button对象对应于button元素。请参看第1章中关于button元素的讨论，以了解它与button类型的input元素之间的区别。为支持Web Forms 2.0应用，Opera 9在这个对象上实现了大量属性与方法。

等价HTML元素 <button>

对象模型引用方式 [window.]document.getElementById("*elementID*")

对象特定属性 action、autofocus、dataFld、dataFormatAs、dataSrc、enctype、form、forms、htmlTemplate、labels、method、name、replace、status、target、type、validationMessage、validity、value、willValidate

对象特定方法 checkValidity()、createTextRange()、dispatchChange()、dispatchFormChange()、setCustomValidity()

对象特定事件属性 无。

action

IE n/a NN n/a Moz n/a Saf n/a Op 9 DOM n/a

可读/可写

当button元素（须要将其type属性设置为"submit"）被点击时，这个Web Forms 2.0扩展可以将封闭表单提交至另一个URI，而该URI可以与常规的表单提交地址不同。

示例 document.getElementById("myButton").action = "redirect.php";

值 URI字符串。

默认值 无。

autofocus

IE n/a NN n/a Moz n/a Saf n/a Op 9 DOM n/a

可读/可写

页面加载完成后，这个Web Forms 2.0扩展将会把焦点聚集在元素上。每个页面上只能有一个表单控件元素拥有此属性。按钮的type属性必须设置为submit。

示例 `document.getElementById("myButton").autofocus = true;`
值 布尔值：`true | false`。
默认值 `false`

dataFld

IE *4* NN *n/a* Moz *n/a* Saf *n/a* Op *n/a* DOM *n/a*

可读/可写

此属性用于IE中的数据绑定，以便将一个远程数据源的列名与按钮对象的标注联系在一起。此时，元素中还必须设定`datasrc`属性。如果`dataFld`和`dataSrc`属性均设置为空字符串，那么就会打破元素与数据源之间的绑定关系。

示例 `document.getElementById("myButton").dataFld = "linkURL";`
值 数据源数据列内区分大小写的标识符。
默认值 无。

dataFormatAs

IE *4* NN *n/a* Moz *n/a* Saf *n/a* Op *n/a* DOM *n/a*

可读/可写

本属性用于IE浏览器中的数据绑定。对于来自于数据源的原始资料，浏览器根据此属性值决定将它们视为纯文本亦或是HTML标签。

示例 `document.getElementById("myButton").dataFormatAs = "html";`
值 字符串常量：`text | html`。
默认值 `text`

dataSrc

IE *4* NN *n/a* Moz *n/a* Saf *n/a* Op *n/a* DOM *n/a*

可读/可写

为实现IE数据绑定功能，此属性用于指定页面内`object`元素的ID值，以便在进行远程数据访问时加载数据源对象。通过`button`元素中的`datafld`属性指定来自于数据源中的绑定内容。如果`dataFld`和`dataSrc`属性均设置为空字符串，那么就会打破元素与数据源之间的绑定关系。

示例 `document.getElementById("myButton").dataSrc = "DBSRC3";`
值 `object`元素的标识符，区分大小写。
默认值 无。

enctype

IE *n/a* NN *n/a* Moz *n/a* Saf *n/a* Op *9* DOM *n/a*

可读/可写

通过`enctype`与其他属性（如`action`）的协助，点击`button`元素（其`type`属性为`submit`）以提交封闭表单时，这个Web Forms 2.0扩展可以使得提交目标URI和附件的MIME类型均与常规的表单不同。

示例 `document.getElementById("myButton").enctype = "text/plain";`
值 MIMEtype字符串。
默认值 无。

form

IE *4* NN *n/a* Moz *all* Saf *all* Op *7* DOM *1*

只读

返回包含当前元素的`form`元素的引用。在Web Forms 2.0环境下，可以通过`setAttribute()`方法将一个按钮与另一个不同的表单关联起来。

示例 `var theForm = event.srcElement.form;`
值 对象引用。
默认值 无。

forms

IE *n/a* NN *n/a* Moz *n/a* Saf *n/a* Op *9* DOM *n/a*

只读

这个Web Forms 2.0扩展返回一个对象引用数组（NodeList），与当前button元素相关联的form对象组成了该数组。

示例 `var formList = document.getElementById("myButton").forms;`
值 数组。
默认值 包含一个封闭表单元素的引用。

htmlTemplate

IE *n/a* NN *n/a* Moz *n/a* Saf *n/a* Op *9* DOM *n/a*

只读

这个Web Forms 2.0扩展返回元素对象（RepetitionElement对象）的一个引用，该对象的ID与当前button元素的template属性（不能设置为null）相匹配。

示例 `var repeatTemplate = document.getElementById("myButton").htmlTemplate;`
值 元素对象的引用。
默认值 `null`

labels

IE *n/a* NN *n/a* Moz *n/a* Saf *n/a* Op *9* DOM *n/a*

只读

这个Web Forms 2.0扩展返回一个对象引用数组（HTML集合），与当前button元素相关联的label元素组成了该数组。

示例 `var allLabels = document.getElementById("myButton").labels;`
值 label元素对象引用组成的数组。
默认值 空数组。

method

IE *n/a* NN *n/a* Moz *n/a* Saf *n/a* Op *9* DOM *n/a*

可读/可写

当点击button元素（须要将其type属性设置为"submit"）以提交封闭表单时，这个Web Forms 2.0扩展允许其提交目标URI和事件方法与常规的表单不同。

示例 `document.getElementById("myButton").method = "get";`
值 方法类型字符串。
默认值 `get`

name

IE *4* NN *n/a* Moz *all* Saf *all* Op *7* DOM *1*

可读/可写

当某个元素作为一个表单控件时，这是与该元素相关的标识符。当表单被提交时，此属性的值将作为名称/值对的一部分而被提交至服务器。由于根据控件类型可以通过其他手段指定控件标注，因此从用户的角度来看，控件名称是被隐藏的。与此同时，脚本也可以使用控件名称来引用对象。

示例 `document.forms[0].compName.name = "company";`

值 区分大小写的标识符，必须遵循标识符命名规则：不包含空白、不能以数字作为起始字符、除下划线外不允许使用其他标点符号。

默认值 无。

replace

IE *n/a* NN *n/a* Moz *n/a* Saf *n/a* Op *9* DOM *n/a*

可读/可写

这个Web Forms 2.0扩展将一些指令与表单提交控件联系在一起，用以在表单提交后处理服务器返回数据的相关指令。目前有两种处理方式，默认方式是使用服务器的响应数据替换原文档。如果已为form元素的data属性指定了一个URL，那么浏览器会将返回的数值填充到表单中，而不是重新载入表单的初始化数据，这是第二种处理方式。

示例 `document.getElementById("myButton").replace = "values";`

值 两个常量值之一：`document | values`。

默认值 `document`

status

IE *4* NN *n/a* Moz *n/a* saf *n/a* Op *n/a* DOM *n/a*

可读/可写

与其他类型表单控件的status属性不同，此属性对按钮控件并没有任何外形或功能方面的影响。

值 布尔值：`true | false`；或者`null`。

默认值 `null`

target

IE *n/a* NN *n/a* Moz *n/a* Saf *n/a* Op *9* DOM *n/a*

可读/可写

在button元素（必须将其type设置为submit）被点击后，这个Web Forms 2.0扩展允许其内嵌表单的返回点与常规表单有所不同，使得其返回页面出现在另一个窗口或框架之中。

示例 `document.getElementById("myButton").target = "_blank";`

值 字符串形式的框架或窗口名称。

默认值 无（表示当前窗口或框架）。

type

IE *4* NN *n/a* Moz *all* Saf *all* Op *7* DOM *1*

只读

指明button元素所指定的按钮形式，如button、reset，还是submit。Web Forms 2.0还添加了其他的按钮类型，以便促进对重复块的控制。

示例

```
if (evt.target.type == "button") {
    // 处理 button 元素
}
```

值 对所有支持它的浏览器而言，可以使用如下3个常量字符串之一：`button | reset | submit`。在支持Web Forms 2.0的浏览器中还可以使用这些附加的字符串常量：`add | move-down | move-up | remove`。

默认值 `button`

validationMessage

IE *n/a* NN *n/a* Moz *n/a* Saf *n/a* Op *9* DOM *n/a*

只读

如果表单控件在Web Forms 2.0规范下校验失败，那么这个Web Forms 2.0的扩展属性会返回一个由浏览器生成的消息。由于返回空字符串就表示校验正常，这使得这个属性对面向文本的表单控件而言更有意义。对一个button元素而言，这个属性总会返回一个空字符串。

值 空字符串。

默认值 空字符串。

validity

IE *n/a* NN *n/a* Moz *n/a* Saf *n/a* Op *9* DOM *n/a*

只读

这个Web Forms 2.0扩展返回一个ValidityState对象。对一个button元素而言，在返回的对象中所有成员的值均为false，仅有valid属性是ture。请参看ValidityState对象。

值 ValidityState对象。

默认值 ValidityState对象。

value

IE *4* NN *n/a* Moz *all* Saf *all* Op *7* DOM *1*

只读

提供与表单控件相关的当前值，该值将与元素的名称/值对一起被提交至服务器端。与按钮类型的input元素对象不同，用户无法看到这个value属性的值，并由元素内容设置对应的标注（通过innerHTML属性或内嵌的节点）。

示例

```
var val = document.getElementById("myButton").value;
```

值 字符串。

默认值 无。

willValidate

IE *n/a* NN *n/a* Moz *n/a* Saf *n/a* Op *9* DOM *n/a*

只读

这个Web Forms 2.0扩展返回一个布尔值，以表明表单控件是否能够达到Web Forms 2.0机制下的校验标准。由于button元素并不是一个正确的类型，因此总是会返回false。

值 布尔值：true | false。

默认值 false

checkValidity( )

IE *n/a* NN *n/a* Moz *n/a* Saf *n/a* Op *9* DOM n/a

这个Web Forms 2.0方法会返回一个布尔值，以表明表单控件是否能够达到其校验标准，归根到底，就是判断validity.valid属性是否为true。由于对button而言，该值一直为true，因此checkValidity()方法此时总会返回true。

返回值 布尔值：true | false。

参数 无。

createTextRange( )

IE *4* NN *n/a* Moz *all* Saf *all* Op *7* DOM *1*

创建一个包含按钮标注文本的TextRange对象。请参见TextRange对象。

返回值 TextRange对象。
参数 无。

dispatchChange( ), dispatchFormChange( )

IE *n/a* NN *n/a* Moz *n/a* Saf *n/a* Op *9* DOM *n/a*

这两个Web Forms 2.0方法为当前元素触发了change和formchange事件。因此，让onclick事件处理程序调用这两个方法之一，就可以将一个click事件转换成为一个change或formchange事件。

返回值 无。
参数 无。

setCustomValidity( )

IE *n/a* NN *n/a* Moz *n/a* Saf *n/a* Op *9* DOM *n/a*

```
setCustomValidity([errorString])
```

这个Web Forms 2.0方法可以设置validity属性的customError布尔值。但它对button元素无效，调用后反而会抛出一个NOT_SUPPORTED_ERR错误。

返回值 无。
参数

errorString

如果此参数为null或空字符串，将重新设置validity对象的customError属性，即表示表单控件无效。在当前会话期内，由于浏览器会记录一条错误信息，因此验证失败后会显示这条信息。

canvas

IE *n/a* NN *n/a* Moz *1.8* Saf *2* Op *9* DOM *n/a*

canvas对象对应于canvas元素，WHATWG在网络应用程序1.0规范之中定义了canvas元素。这个对象仅仅是文档树内的DOM元素。为了在canvas元素空间内完成图形操作，首先要获取图像显示环境对象（通过getContext()），然后调用该对象的方法，请参见CanvasRenderingContext2D对象。

等价HTML元素 <canvas>
对象模型引用方式 [window.]document.getElementById("*elementID*")
对象特定属性 height、width
对象特定方法 getContext()、toDataURL()
对象特定事件 无。

height, width

IE *n/a* NN *n/a* Moz *1.8* Saf *2* Op *9* DOM *n/a*

可读/可写

决定元素的图像绘制空间的尺寸，单位为像素。即使使用元素属性对其尺寸进行了设定，也可以使用脚本改变元素尺寸的大小。

示例
```
document.getElementById("myCanvas").width = 450;
```
值 整数像素值。
默认值 0

getContext( )

IE *n/a* NN *n/a* Moz *1.8* Saf *2* Op *n/a* DOM *n/a*

```
getContext("contextID")
```

返回元素内绘图空间（称之为“环境“）的引用。请参见CanvasRenderingContext2D对象，以便了解在canvas元素中指定并绘制线条与各种图形所需的方法。

返回值 绘图环境（canvas context）对象（CanvasRenderingContext2D）。

参数

contextID

一个字符串，它指出须要从canvas元素中引用哪个绘图环境。早期出现的实现方式仅包含两维空间的绘制环境，名为2D。

toDataURL( )

IE *n/a* NN *n/a* Moz *1.8.1* Saf *n/a* Op *n/a* DOM *n/a*

```
toDataURL(["MIMEType"])
```

返回一个URI形式的字符串，该URI（以data:协议作为起始部分）包含元素中图像的二进制数据。默认的MIME类型是"image/png"，也可以指定浏览器支持的其他类型，例如，Mozilla还支持"image/jpeg"。

返回值 URI字符串。

参数

MIMEType

一个表示此方法返回数据的MIME类型的字符串。

CanvasRenderingContext2D

IE *n/a* NN *n/a* Moz *1.8* Saf *2* Op *n/a* DOM *n/a*

canvas元素对象的getContext()方法将返回这个对象，它可以为可显示元素内的空间提供绘图机制。这个对象的属性和方法控制着绘图操作，而脚本也只有在canvas元素中才能执行这些操作。虽然最终还可能出现其他类型的绘制环境，比如三维绘制空间。但目前的这个环境仅限于二维空间，即只能沿着x和y轴绘制图形。Safari 2称这个对象为Context2D。

作为一个典型的简单编码实例，下面这个方法使用两种不同的粗细的弧形绘制了一张笑脸：

```
function draw() {
    var canvas = document.getElementById("mycanvas");
    var ctxt = canvas.getContext("2d");
    ctxt.fillStyle = "rgb(255,225,0)";
    ctxt.beginPath();
    ctxt.lineWidth = 2;
    ctxt.arc(75,75,50,0,Math.PI*2,true);
    ctxt.fill();
    ctxt.moveTo(65,65);
    ctxt.arc(60,65,5,0,Math.PI*2,true);
    ctxt.moveTo(95,65);
    ctxt.arc(90,65,5,0,Math.PI*2,true);
    ctxt.stroke(); // 绘制所有的圆弧
    ctxt.beginPath();
    ctxt.moveTo(110,75);
    ctxt.lineWidth = 4;
    ctxt.arc(75,75,35,0,Math.PI,false);
    ctxt.stroke(); // 绘制更为密集的圆弧
}
```

绘图环境（canvas context）中包含着大量用于绘制图像和动画的属性和方法。然而只有对颜色梯度、几何学，乃至插图技巧等有所研究，才能更为有效地使用这些方法，可惜的是，这些内容已经超出了本书所探讨的范围。如果想了解如何更好地应用这些属性和方法，不妨访问以下链接：*https://developer.mozilla.org/en/Canvas_tutorial*。

对象模型引用方式

```
[window.]document.getElementById("canvasElementID").getContext("2D")
```

对象特定属性 canvas、fillStyle、globalAlpha、globalCompositeOperation、lineCap、lineJoin、lineWidth、miterLimit、shadowBlur、shadowColor、shadowOffsetX、shadowOffsetY、strokeStyle

对象特定方法 arc()、arcTo()、beginPath()、bezierCurveTo()、clearRect()、clip()、closePath()、createLinearGradient()、createPattern()、createRadialGradient()、drawImage()、fill()、fillRect()、lineTo()、moveTo()、quadraticCurveTo()、rect()、restore()、rotate()、save()、scale()、stroke()、strokeRect()、translate()

对象特定事件属性 无。

canvas

IE *n/a* NN *n/a* Moz *1.8* Saf *2* Op *n/a* DOM *n/a*

只读

一个canvas元素的引用，当前绘图环境即来自于该元素。这属性可以给开发者带来方便，它可以让一段可重用的脚本重新获取原始元素对象，从而调整元素的height或width属性。

示例 `myContext.canvas.width = 450;`

值 HTML元素引用。

默认值 包含canvas元素引用。

fillStyle

IE *n/a* NN *n/a* Moz *1.8* Saf *2* Op *n/a* DOM *n/a*

可读/可写

指明填充一个图形时所使用的颜色、渐进或图案等。

示例 `myContext.fillStyle = "rgb(255, 0, 0)";`

值 对颜色而言，可使用任何字符串形式的有效的CSS3颜色值，其中包括颜色名称、RGB数字或RGBA（Red Green Blue Alpha，alpha表示透明度）数字。对于渐变和图案，则可以使用任意一个由绘图环境对象的渐变或图案生成方法所创建的对象。

默认值 `rgb(0, 0, 0)`，即黑色。

globalAlpha

IE *n/a* NN *n/a* Moz *1.8* Saf *2* Op *n/a* DOM *n/a*

可读/可写

在显示之前，须要为所有的图形或图像指定其透明度。

示例 `myContext.globalAlpha = 0.5;`

值 从0到1.0的浮点数，0表示完全透明，而1.0表示完全不透明。

默认值 1.0（不透明）。

globalCompositeOperation

IE *n/a* NN *n/a* Moz *1.8* Saf *2* Op *n/a* DOM *n/a*

可读/可写

指定新图形（设置此属性之后就会显示）与绘图环境内已有内容之间的遮盖类型。

示例 `myContext.globalCompositeOperation = "destination-in";`

值 以下11个常量字符串之一：`copy | darker | destination-atop | destination-in | destinationout | destination-over | lighter | source-atop | source-in | source-out | source-over`。

默认值 `source-over`，新图形不进行任何裁剪，直接在已有内容之上绘制。

lineCap

IE *n/a* NN *n/a* Moz *1.8* Saf *2* Op *n/a* DOM *n/a*

可读/可写

指定线段端点的样式。类似于lineTo()的方法可以指定一条线段的端点坐标，在默认情况下该坐标上的端点是一个空心方块。当然也可以改变线段的绘制形式，这样一来可以使用实心半圆（其半径为线宽的一半）或方形（其高度是线宽的一半）来显示端点。在后面的两种图形中，由于顶端有了额外的图形，会使得线段略微超出其端点坐标。

431

示例 `myContext.lineCap = "round";`

值 以下3个常量字符串之一：butt | round | square。

默认值 butt

lineJoin

IE *n/a* NN *n/a* Moz *1.8* Saf *2* Op *n/a* DOM *n/a*

可读/可写

指定两条线段交点的显示样式。在默认情况下，两条线段结合处形成一个斜角接头，即线段的外边缘交会于一个点。但也可以使用圆角接头或斜削接头表示两条线的交点。

示例 `myContext.lineJoin = "bevel";`

值 以下3个常量字符串之一：bevel | miter | round。

默认值 miter

lineWidth

IE *n/a* NN *n/a* Moz *1.8* Saf *2* Op *n/a* DOM *n/a*

可读/可写

指定绘制图形时所用线条的宽度。其单位是图像坐标空间单位，通常使用像素来表示。须要注意的是，当根据坐标来绘制线条时，实际上是该线条的中线对应于所需的坐标位置，这可能会使得较粗的线条出现一点定位偏差，例如，使用fillRect()方法绘制一个矩形时，实际的图像位置就可能与预期中的位置存在微小的差别。

示例 `myContext.lineWidth = 5;`

值 正浮点数。

默认值 1.0

miterLimit

IE *n/a* NN *n/a* Moz *1.8* Saf *2* Op *n/a* DOM *n/a*

可读/可写

指定斜接长度系数，即内接点与外接点之间的距离。

示例 `myContext.miterLimit = 17.5;`

值 正浮点数。

默认值 10.0

shadowBlur

IE *n/a* NN *n/a* Moz *n/a* Saf *2* Op *n/a* DOM *n/a*

432

可读/可写

指定图形旁被阴影覆盖的坐标空间大小。

示例 `myContext.shadowBlur = 3;`

值 正浮点数。

默认值 1.0

shadowColor

IE *n/a* NN *n/a* Moz *n/a* Saf *2* Op *n/a* DOM *n/a*

可读/可写

指定图形阴影的颜色。

示例 `myContext.shadowColor = "rgb(90, 90, 90)";`

值 符合CSS3兼容色规范的任意一种颜色。

默认值 `black`

shadowOffsetX, shadowOffsetY

IE *n/a* NN *n/a* Moz *n/a* Saf *2* Op *n/a* DOM *n/a*

可读/可写

指明阴影与图形之间在水平（x）和竖直（y）方向上的坐标空间相对距离。正值表示向右、上移动，而负值表示向左、下移动。

示例 `myContext.shadowOffsetX = 5;`

值 正、负浮点数。

默认值 `0`

strokeStyle

IE *n/a* NN *n/a* Moz *1.8* Saf *2* Op *n/a* DOM *n/a*

可读/可写

指明绘制线条或图形轮廓时使用的颜色、渐进或图案等。

示例 `myContext.strokeStyle = "rgb(255, 0, 0)";`

值 对颜色而言，可使用任何字符串形式的有效的CSS3颜色值，其中包括颜色名称、RGB数字或RGBA（Red Green Blue Alpha，alpha表示透明度）数字。对于渐变和图案，可以使用任意一个由绘图环境对象的渐变或图案生成方法所创建的对象。

默认值 `rgb(0, 0, 0)`，即黑色。

arc( )

IE *n/a* NN *n/a* Moz *1.8* Saf *2* Op *n/a* DOM *n/a*

`arc(x, y, radius, startAngle, endAngle, counterClockwiseFlag)`

说明如何以当前点为圆心在绘图环境内绘制一个圆弧。这个圆弧以x、y参数为圆心，由两个端点角度（单位为弧度）来定义弧长。与其他所有路径相关的方法一样，调用这个方法时须要将它放置在`beginPath()`和`stroke()`方法之间。

返回值 无。

参数

x, y

由`arc()`定义的圆形的圆心坐标，x表示横坐标，y表示纵坐标，所有坐标均以绘图坐标单元（canvas coordinate unit）为单位。

radius

长度值，表示圆弧所在圆的半径长度，以绘图坐标单元为单位。

startAngle, endAngle

圆弧在圆周上的起始和结束点，使用弧度值表示。

counterClockwiseFlag

布尔值，表示圆弧从起始到结束的绘制方向：`true`表示逆时针，而`false`表示顺时针。

arcTo( )

IE *n/a* NN *n/a* Moz *1.8* Saf *2* Op *n/a* DOM *n/a*

```
arc(x1, y1, x2, y2, radius)
```

说明如何以当前点为圆心在绘图环境内绘制一个圆弧。此时绘制的圆弧半径为*radius*，且与两条直线相切，其中一条直线从当前点到点（*x1*, *y1*），另一条直线则是从（*x1*, *y1*）到（*x2*, *y2*）。

返回值 无。

参数

x1, *y1*

圆弧的起始点坐标，x表示横坐标，y表示纵坐标，所有坐标均以绘图坐标单元为单位。

x2, *y2*

圆弧的结束点坐标，x表示横坐标，y表示纵坐标，所有坐标均以绘图坐标单元为单位。

radius

长度，这是一个图像坐标单位数值，表示圆弧所在圆的半径长度。

beginPath( )

IE *n/a* NN *n/a* Moz *1.8* Saf *2* Op *n/a* DOM *n/a*

将绘图环境的路径列表清空，并且将绘图点移动到（0,0），以准备接收来自于另一个图形的新路径说明。首次创建绘图环境时，都会自动调用这个方法。

返回值 无。

参数 无。

bezierCurveTo( )

IE *n/a* NN *n/a* Moz *1.8.1* Saf *2* Op *n/a* DOM *n/a*

```
bezierCurveTo(cp1x, cp1y, cp2x, cp2y, x, y)
```

指明一条从当前点出发到（x,y）结束的三次方贝塞尔曲线，并绘制在绘图环境之内。参数中指定的两个坐标点控制着这条曲线的具体形状，起始点为（cp1x, cp1y），而结束点为（cp2x, cp2y）。

返回值 无。

参数

cp1x, *cp1y*

在起始点影响曲线形状的控制点，x表示横坐标，y表示纵坐标，所有坐标均以绘图坐标单元为单位。

cp2x, *cp2y*

在结束点影响曲线形状的控制点，x表示横坐标，y表示纵坐标，所有坐标均以绘图坐标单元为单位。

x, *y*

曲线的端点，x表示横坐标，y表示纵坐标，所有坐标均以绘图坐标单元为单位。

clearRect( )

IE *n/a* NN *n/a* Moz *1.8.1* Saf *2* Op *n/a* DOM *n/a*

```
clearRect(x, y, width, height)
```

从绘图环境内擦除由参数指定的区域。使用这个方法既可以有选择地擦除多个矩形区域，也可以清理整个绘图环境。此时只须使用一段普通的脚本就可以通过canvas.height和canvas.width属性获取绘图环境的尺寸，然后将这个尺寸作为参数，即可清除整个绘图环境。

返回值 无。

参数

x, *y*

待擦除矩形的左上角坐标，x表示横坐标，y表示纵坐标，所有坐标均以绘图坐标单元为单位。

width, height

待擦除矩形的宽度和高度，所有坐标均以绘图坐标单元为单位。这两个值均不能为0。

435

clip( )

IE *n/a* NN *n/a* Moz *1.8.1* Saf *2* Op *n/a* DOM *n/a*

通过其他方法创建了一个线条之后，可以使用clip()方法为绘图环境创建一个剪切区域。此后新图形只会出现在剪切区域内。可以用不同的路径方法创建任意形状的图形作为剪切区域。

返回值 无。

参数 无。

closePath( )

IE *n/a* NN *n/a* Moz *1.8.1* Saf *2* Op *n/a* DOM *n/a*

在当前路径（current path）点和当前路径的起始点直接按绘制一条直线。

返回值 无。

参数 无。

createLinearGradient( )

IE *n/a* NN *n/a* Moz *1.8.1* Saf *2* Op *n/a* DOM n/a

```
createLinearGradient(x0, y0, x1, y1)
```

为直线渐变创建并返回一个CanvasGradient对象，它可以指定给绘图环境的fillStyle和strokeStyle属性。根据方法的输入参数，绘图空间内的一个线性渐变将拥有一个起始点和一个结束点。一旦拥有了渐变对象，就可以通过渐变对象的addColorStop()方法为渐变区域内的不同位置设置具体的颜色。例如下面这段代码就创建了一个线性渐变，它将从绘图环境的左上角一直向右下方延伸200像素，而且将从白色过渡到红色：

```
var ctxt = document.getElementById("mycanvas").getContext("2d");
var gradient = ctxt.createLinearGradient(0, 0, 200, 200);
gradient.addColorStop(0, "white");
gradient.addColorStop(1, "red");
ctxt.fillStyle = gradient;
ctxt.fillRect(0, 0, 200, 200);
```

addColorStop()方法的第1个参数是一个介于0（渐变的起始点）和1（渐变的结束点）之间的浮点数。当然，也可以在渐变方法上指定多个中间色。

返回值 CanvasGradient对象。

参数

x0, y0

颜色渐变的起始点坐标，x表示横坐标，y表示纵坐标，所有坐标均以绘图坐标单元为单位。

x1, y1

颜色渐变的结束点坐标，x表示横坐标，y表示纵坐标，所有坐标均以绘图坐标单元为单位。

436

createPattern( )

IE *n/a* NN *n/a* Moz *1.8.1* Saf *2* Op *n/a* DOM *n/a*

```
createPattern(imageObjectRef, "repeatStyle")
```

创建并返回一个CanvasPattern对象，它的原始图像将作为一个图形填充模式，或者作为一个笔画模式。可以通过熟知的DOM技术创建一个新的图像对象，并作为第一个参数传入此方法，代码如下：

```
var img = new Image();
img.src = "myImage.png";
var pattern = ctxt.createPattern(img, "repeat");
```

创建后可以将结果对象指定给绘图环境的fillStyle或strokeStyle属性。

返回值 CanvasPattern对象。

参数

imageObjectRef

一个图像对象的引用，其src属性指向一个图像URL。如果从canvas元素的dataToURL()方法获取一个图像对象，那么某些浏览器也可能支持使用该对象。

repeatStyle

以下4个常量字符串之一：no-repeat | repeat | repeat-x | repeat-y。

createRadialGradient( )

IE *n/a* NN *n/a* Moz *1.8.1* Saf *2* Op *n/a* DOM *n/a*

```
createRadialGradient(x0, y0, r0, x1, y1, r1)
```

创建并返回一个用于颜色渐变的CanvasGradient对象，该渐变以某个圆或点为核心向外扩张一段距离，具体的扩展范围由半径值决定。根据方法的输入参数，绘图空间内的一个径向渐变拥有一个起始点和一个结束点。一旦拥有了渐变对象，就可以通过渐变对象的addColorStop()方法为渐变区域内的不同位置设置具体的颜色。

返回值 CanvasGradient对象。

参数

x0, y0

颜色渐变的起始点坐标，x表示横坐标，y表示纵坐标，所有坐标均以绘图坐标单元为单位。

x1, y1

颜色渐变的结束点坐标，x表示横坐标，y表示纵坐标，所有坐标均以绘图坐标单元为单位。

r0, r1

从中心点到起始点（r0）和结束点（r1）的半径距离，所有坐标均以绘图坐标单元为单位。

drawImage( )

IE *n/a* NN *n/a* Moz *1.8.1* Saf *2* Op *n/a* DOM *n/a*


```
drawImage(imageOrCanvas, dx, dy) drawImage(imageOrCanvas, dx, dy, dw, dh)
drawImage(imageOrCanvas, sx, sy, sw, sh, dx, dy, dw, dh)
```

向绘图环境内的某个指定位置复制一个导入的（或生成的）图像。根据所指定的参数数目，这个方法有3种不同的版本。

返回值 无。

参数

dx, dy

绘图环境内图像左上角坐标，x表示横坐标，y表示纵坐标，所有坐标均以绘图坐标单元为单位。

dw, dh

图像的宽度和高度，所有坐标均以绘图坐标单元为单位。

sx, sy

在原始图像空间内缩放图像的左上角坐标，x表示横坐标，y表示纵坐标，所有坐标均以绘图坐标单元为单位。

dw, dh

图像缩放后的宽度和高度，所有坐标均以绘图坐标单元为单位。

fill( )

IE *n/a* NN *n/a* Moz *1.8.1* Saf *2* Op *n/a* DOM n/a

使用fillStyle属性所对应的颜色、渐变或样式，填充当前路径指定的空间。

返回值 无。

参数 无。

fillRect( )

IE *n/a* NN *n/a* Moz *1.8.1* Saf *2* Op *n/a* DOM *n/a*

```
fillRect(x, y, width, height)
```

使用fillStyle属性所对应的颜色、渐变或样式，绘制由参数指定的区域。

返回值 无。

参数

x, y

待绘制矩形的左上角坐标，x表示横坐标，y表示纵坐标，所有坐标均以绘图坐标单元为单位。

width, height

待绘制矩形的宽度和高度，所有坐标均以绘图坐标单元为单位。这两个值均不能为0。

lineTo( )

IE *n/a* NN *n/a* Moz *1.8.1* Saf *2* Op *n/a* DOM *n/a*

```
lineTo(x, y)
```

在当前路径点和参数指定的坐标点之间绘制一条线段，然后将最后经过的坐标点作为当前点。

返回值 无。

参数

x, y

线段的端点，x表示横坐标，y表示纵坐标，所有坐标均以绘图坐标单元为单位。

moveTo( )

IE *n/a* NN *n/a* Moz *1.8.1* Saf *2* Op *n/a* DOM *n/a*

```
moveTo(x, y)
```

将当前点移动至参数指定的坐标位置，同时以该点作为起始位置创建一条子路径。

返回值 无。

参数

x, y

新位置坐标，x表示横坐标，y表示纵坐标，所有坐标均以绘图坐标单元为单位。

quadraticCurveTo( )

IE *n/a* NN *n/a* Moz *1.8.1* Saf *2.02* Op *n/a* DOM *n/a*

```
quadraticCurveTo(cp1x, cp1y, x, y)
```

指明一条从当前点出发，到（x,y）结束的二次贝塞尔曲线，并绘制在绘图环境之内。这条曲线的具体形状由坐标点（cp1x, cp1y）控制。

返回值 无。

参数

cp1x, cp1y

影响曲线形状的控制点，x表示横坐标，y表示纵坐标，所有坐标均以绘图坐标单元为单位。二次贝塞尔曲线的形状与其起始和结束点均有关。

x, y

曲线的端点，x表示横坐标，y表示纵坐标，所有坐标均以绘图坐标单元为单位。

rect( )

IE *n/a* NN *n/a* Moz *1.8.1* Saf *2* Op *n/a* DOM *n/a*

```
rect(x, y, width, height)
```

创建一个单独的子路径，它根据指定的尺寸在指定的位置定义一个矩形。然后此方法会将当前点设置为(0,0)。

返回值 无。

参数

x, y

矩形路径的左上角坐标，x表示横坐标，y表示纵坐标，所有坐标均以绘图坐标单元为单位。

width, height

矩形路径的宽度和高度，所有坐标均以绘图坐标单元为单位。

restore( )

IE *n/a* NN *n/a* Moz *1.8.1* Saf *2* Op *n/a* DOM *n/a*

从内部绘图状态栈中弹出顶端的绘图状态。如果要向栈内推送数据请使用`save()`方法。

返回值 无。

参数 无。

rotate( )

IE *n/a* NN *n/a* Moz *1.8.1* Saf *2* Op *n/a* DOM *n/a*

```
rotate(angle)
```

此方法可以有效地旋转绘图环境（此时可称之为变换矩阵，transformation matrix），如果须要进行变换那么应该在绘制之前调用这个方法。

返回值 无。

参数

angle

顺时针旋转的弧度值。

save( )

IE *n/a* NN *n/a* Moz *1.8.1* Saf *2* Op *n/a* DOM *n/a*

将当前绘图状态压入内部绘图状态栈中。如果要从栈内弹出数据请使用`restore()`方法。

返回值 无。

参数 无。

scale( )

IE *n/a* NN *n/a* Moz *1.8.1* Saf *2* Op *n/a* DOM *n/a*

```
scale(x, y)
```

此方法通过将调整因子与水平轴和纵轴相乘来调整绘图环境的比例尺。

返回值 无。

参数

x, y

浮点数，表示调整水平（x）和竖直（y）轴上绘图环境的乘法因子。

stroke( )

IE *n/a* NN *n/a* Moz *1.8.1* Saf *2* Op *n/a* DOM *n/a*

在绘图环境内绘制当前路径，请使用`lineJoin`、`lineWidth`、`miterLimit`及`strokeStyle`属性设置。

返回值 无。

参数 无。

strokeRect( )

IE *n/a* NN *n/a* Moz *1.8.1* Saf *2* Op *n/a* DOM *n/a*

```
strokeRect(x, y, width, height)
```

依据lineJoin、lineWidth、miterLimit和strokeStyle的属性值，按照参数指定的位置和尺寸绘制一个矩形轮廓。

返回值 无。

参数

x, y

待绘制矩形的左上角坐标，x表示横坐标，y表示纵坐标，所有坐标均以绘图坐标单元为单位。

width, height

待绘制矩形的宽度和高度，所有坐标均以绘图坐标单元为单位。

translate( )

IE *n/a* NN *n/a* Moz *1.8.1* Saf *2* Op *n/a* DOM *n/a*

```
translate(x, y)
```

将绘图环境的原点移动到参数所指定的坐标上。

返回值 无。

参数

x, y

浮点数，表示绘图环境的新原点相对旧原点的偏移量，x表示横轴距离，y表示纵轴距离。

caption

IE *4* NN *n/a* Moz *all* Saf *all* Op *7* DOM *1*

caption对象对应于caption元素，它必须内嵌在一个table元素中。IE/Mac为此对象实现了客户端和滚动相关的属性集。

等价HTML元素 <caption>

对象模型引用方式 [window.]document.getElementById("*elementID*")

对象特定属性 align、vAlign

对象特定方法 无。

对象特定事件 无。

align

IE *4* NN *n/a* Moz *all* Saf *all* Op *7* DOM *1*

可读/可写

决定标题在表格中的位置。请参见第1章中caption元素的align属性，以便了解IE/Windows环境下align和vAlign之间的相互作用。W3C DOM主要使用align属性来决定标题是放置在表格之上还是在表格之下。

示例 document.getElementById("myCaption").align = "bottom";

值 任意一个字符串常量：bottom | left | right | top。

默认值 top

vAlign

IE *4* NN *n/a* Moz *n/a* Saf *n/a* Op *n/a* DOM *n/a*

可读/可写

指定表格标题是出现在表格上方还是下方。

示例 document.getElementById("tabCaption").vAlign = "bottom"
值 不区分大小写的字符串常量：bottom | top。
默认值 top

cells

IE 4 NN n/a Moz all Saf all Op 7 DOM 1

某个tr元素包含的所有td元素的集合。此集合中的成员按照它们在源代码中的次序进行排序。

对象模型引用方式 document.getElementById("*rowID*").cells
对象特定属性 length
对象特定方法 item()、namedItem()、tags()、urns()
对象特定事件 无。

442

length

IE 4 NN n/a Moz all Saf all Op 7 DOM 1

只读

返回集合中元素的数量。

示例 var howMany = document.getElementById("myTable").rows[0].cells.length;
值 整数值。

item( )

IE 4 NN n/a Moz all Saf all Op 7 DOM 1

item(*index*[, *subindex*]) item(*index*)

根据符合索引值（IE中也可以使用index或subindex值）的元素对象，返回一个单独的td对象或一组td对象集合。

返回值 一个td对象或一个td对象集合。如果并不存在与参数相匹配的对象，则返回null。

参数

index

当输入参数是一个从零开始计数的整数时，返回值是源代码内与某个具体项相对应的独立元素（内嵌在当前元素之内），而如果参数为一个字符串（仅适用于IE），则返回id属性与该字符串相符的元素集合。

subindex

在IE中，如果为第1个参数指定一个字符串值，那么可以将第2个参数指定为一个从零开始计数的索引值，这样就可以从集合中找到一个id属性与第1个参数的字符串相匹配的具体元素。

namedItem( )

IE 6 NN n/a Moz all Saf all Op 7 DOM 1

namedItem("*ID*")

返回单个td对象或td对象集合，它们对应于符合参数字符串值的元素。

返回值 一个td对象或一个td对象集合。如果并不存在与参数相匹配的对象，则返回null。

参数

ID

与目标元素的id属性包含相同值的字符串。

443

tags( )

IE 4 NN n/a Moz n/a Saf n/a Op n/a DOM n/a

tags("*tagName*")

从当前集合的全部内嵌对象中，返回标签与*tagName*参数相匹配的对象集合。尽管在所有的IE集合中均已实现了这个方法（请参见all.tags()方法），但对同一类型的元素集合而言它完全是一种冗余。

urns( )

IE *5(Win)* NN *n/a* Moz *n/a* Saf *n/a* Op *n/a* DOM *n/a*

`urns(URN)`

请参见`all.urns()`方法。

center

IE *4* NN *n/a* Moz *all* Saf *all* Op *7* DOM *1*

`center`对象对应于`center`元素。W3C DOM并不支持HTML 4所反对的`center`元素。为了保证后向兼容性，Mozilla在面对这个元素时，其处理方式与早期浏览器相类似，但它会将这个元素视作一个`span`对象，并且将其`text-align`样式设置为`center`。

等价HTML元素 `<center>`
对象模型引用方式 `[window.]document.getElementById("elementID")`
对象特定属性 无。
对象特定方法 无。
对象特定事件 无。

checkbox

请参见input。

CharacterData

请参见Text。

childNodes, NodeList

IE *4* NN *n/a* Moz *all* Saf *all* Op *7* DOM *1*

若干个W3C DOM对象的`childNodes`属性所返回的对象均为`Node`对象引用的集合，这些`Node`对象都是当前节点对象的直接子节点。在W3C DOM架构中，有一个名为`NodeList`的对象抽象地表示了这个数组，与此同时，`NodeList`还共享了一些属于`childNodes`（IE）对象的属性和方法。这两个对象均指向文档树中的同一部分，此处对它们等同视之。此集合中的成员按照它们在源代码中的次序进行排序。

对象模型引用方式 *nodeReference*`.childNodes`
对象特定属性 `length`
对象特定方法 `item()`、`urns()`
对象特定事件 无。

length

IE *4* NN *n/a* Moz *all* Saf *all* Op *7* DOM *1*

只读

返回集合中节点的数量。

示例
```
var howMany = document.getElementById("myTable").attributes.length;
```
值 数值。

item( )

IE *5* NN *n/a* Moz *all* Saf *all* Op *7* DOM *1*

`item(index)`

返回一个`Node`对象，它对应于一个与索引值相匹配的元素。

返回值 一个Node对象的引用。如果与索引值相匹配的对象不存在，则返回null。与IE中某些集合不同，对childNodes对象不允许使用字符串形式的索引值。

参数

index

一个从零开始计数的整数，依照源代码次序，对应于集合中的某个特定项（内嵌在当前节点之中）。

urns( )

IE *5(Win)* NN *n/a* Moz *n/a* Saf *n/a* Op *n/a* DOM *n/a*


```
urns(URN)
```

请参见all.urns()方法。

children

IE *4* NN *n/a* Moz *n/a* Saf *all* Op *7* DOM *n/a*

当前元素所包含的所有元素的集合。须要注意一点，与childNodes集合不同，children集合仅关注元素而不包含文本节点。此集合中的成员按照它们在源代码中的次序进行排序。Internet Explorer浏览器允许使用数组符号来访问集合中的一个元素。

对象模型引用方式

```
document.getElementById("elementID").children(i)
document.getElementById("elementID").children[i]
```

对象特定属性 length

对象特定方法 item()、namedItem()、tags()、urns()

对象特定事件 无。

length

IE *4* NN *n/a* Moz *n/a* Saf *all* Op *7* DOM *n/a*

只读

返回集合中元素的数量。

示例

```
var howMany = document.body.children.length;
```

值 整数值。

item( )

IE *4* NN *n/a* Moz *n/a* Saf *all* Op *7* DOM *n/a*

```
item(index)
```

返回一个元素对象，其对应元素的源代码次序可与参数传入的索引值相匹配。

返回值 元素对象的一个引用。如果并不存在与参数相匹配的对象，则返回null。

参数

index

一个从零开始计数的整数，依照源代码次序，对应于集合中的某个特定项（内嵌在当前元素之中）。

namedItem( )

IE *6* NN *n/a* Moz *n/a* Saf *all* Op *7* DOM *n/a*

```
namedItem(IDOrName)
```

返回一个元素对象或对象集合，它们对应于符合参数字符串值的元素。

返回值 一个元素对象或一个元素对象集合。如果并不存在与参数相匹配的对象，则返回null。

参数

IDOrName

与目标元素的id或name属性包含相同值的字符串。

tags( )

IE 4 NN n/a Moz n/a Saf all Op 7 DOM n/a

```
tags(tagName)
```

从当前集合的全部内嵌对象中，返回标签与*tagName*参数相匹配的对象集合。尽管在所有的IE集合中均已实现了这个方法（请参见all.tags()方法），但对同一类型的元素集合而言它完全是一种冗余。

urns( )

IE 5(Win) NN n/a Moz n/a Saf n/a Op n/a DOM n/a

```
urns(URN)
```

请参见all.urns()方法。

cite

请参见abbr。

clientInformation

请参见navigator。

clipboardData

IE 5(Win) NN n/a Moz n/a Saf n/a Op n/a DOM n/a

clipboardData对象允许脚本访问Windows系统的剪贴板，并将它作为一个临时容器。这样一来，IE 5及后续版本的IE浏览器在Windows系统下就可以使用这个临时容器来转移文本数据，一些脚本控制的操作尤其需要这个功能，例如模拟剪切、复制和粘贴，以及控制拖拽等。脚本还可以控制存储在clipboardData对象中的具体数据，如一个元素内的文本、整个元素的HTML代码或一个图像的URL。例如，一个针对小朋友的页面可以显示一些不同动物的小图标。如果用户开始拖拽小狗图标，那么img元素的onDragStart事件处理程序启动的一段脚本就会将该元素的一个自定义属性值（可能是一张小狗图片的URL）存储到clipboardData对象中。当用户将图片拖至指定区域时，onDrop事件处理程序的方法会读取clipboardData对象的数据，并且在页面上的指定位置加载对应的照片图像。

属于同一域名和协议的其他页面也可以访问这个对象中储存的数据。因此，不须要借助cookie或location.search字符串的帮助，就可以通过这个对象在页面间传递文本数据。如果使用Array.join()方法将数组转换为字符串，也可以对这个数组进行传递。

使用这个对象及event.dataTransfer对象来传递数据的更多信息，请访问以下链接：*http://msdn.microsoft.com/en-us/library/ms537658(VS.85).aspx*。

等价HTML元素	无。
对象模型引用方式	[window.]clipboardData
对象特定属性	dropEffect、effectAllowed
对象特定方法	clearData()、getData()、setData()
对象特定事件	无。

dropEffect, effectAllowed

IE 5(Win) NN n/a Moz n/a Saf n/a Op n/a DOM n/a

可读/可写

这两个属性均属于clipboardData对象，而clipboardData则继承自dataTransfer对象，后者才是它们真正发挥作用的位置。因此对clipboardData而言，可以忽略这两个属性。

clearData( )

IE *5(Win)* NN *n/a* Moz *n/a* Saf *n/a* Op *n/a* DOM *n/a*

```
clearData([dataFormat])
```

从clipboardData对象内删除数据。

返回值 无。

参数

dataFormat

一个可选的字符串参数，它指定须要删除哪种格式的数据。曾经计划在clipboardData对象中保存多种类型的数据，不过从目前看来，这个计划已经失败。到IE 6为止，只有一种可靠的格式，即“text”。如果忽略这个参数，那么将删除所有类型的数据。

getData( )

IE *5(Win)* NN *n/a* Moz *n/a* Saf *n/a* Op *n/a* DOM *n/a*

```
getData(dataFormat)
```

返回一份来自于clipboardData对象的数据。而clipboardData中的内容将保持不变，以便后来的脚本指令能够继续读取这些数据。

返回值 字符串。

参数

dataFormat

一个字符串参数，指明须要保存的数据类型。曾经计划在clipboardData对象中保存多种类型的数据，不过从目前看来，这个计划已经失败。到IE 6为止，只有一种可靠的格式，即“text”。

setData( )

IE *5(Win)* NN *n/a* Moz *n/a* Saf *n/a* Op *n/a* DOM *n/a*

```
setData(dataFormat, stringData)
```

在clipboardData对象中保存字符串数据。如果赋值成功，那么返回布尔值true。

返回值 布尔值：true | false。

参数

dataFormat

一个字符串参数，指明须要保存的数据类型。曾经计划在clipboardData对象中保存多种类型的数据，不过从目前看来，这个计划已经失败了。到IE 6为止，只有一种可靠的格式，即“text”。尽管此方法可以将URL作为一种输入类型，但是却无法成功读取这种类型的数据。

stringData

任意字符串，可以包含HTML标签。

code

请参见abbr。

col

IE *4* NN *n/a* Moz *all* Saf *all* Op *7* DOM *1*

col对象对应于col元素。使用这个元素可以将一个或多个相邻列分配在一个组内，以便为表内的多个列指定样式、宽度并进行其他可视化处理。

等价HTML元素 <col>

449

对象模型引用方式 [window.]document.getElementById("*elementID*")
对象特定属性 align、ch、chOff、span、vAlign、width
对象特定方法 无。
对象特定事件 无。

align

IE *4* NN *n/a* Moz *all* Saf *all* Op *7* DOM *1*

可读/可写

在col元素控制的单元格中，此属性确定其水平对齐的方式。

示例 document.getElementById("myCol").align = "center";
值 以下4个对齐常量之一：center | char | left | right。
默认值 left

ch

IE *6* NN *n/a* Moz *all* Saf *all* Op *n/a* DOM *1*

可读/可写

ch对应于char属性，它将一个文本字符定义为对齐参照点，列或列分组内的文本均依照该字符进行对齐。注意，只有将align属性设置为char，此属性才有价值。但事实上目前所有的浏览器均不会响应这个属性。

示例 document.getElementById("myCol").ch = ".";
值 单个字符组成的字符串。
默认值 无。

chOff

IE *6* NN *n/a* Moz *all* Saf *all* Op *n/a* DOM *1*

可读/可写

定义一个偏移点，char属性所指定的字符将出现在单元格中偏移点所在位置。事实上目前所有的浏览器均不会响应这个属性。

示例 document.getElementById("myCol").chOff = "80%";
值 字符串形式的像素值或单元格宽度的百分比。
默认值 无。

450

span

IE *4* NN *n/a* Moz *all* Saf *all* Op *7* DOM *1*

可读/可写

提供相邻的列数量，元素的属性和样式设置均会应用于这些列。

示例 document.getElementById("myCol").span = 2;
值 整数值。
默认值 1

vAlign

IE *4* NN *n/a* Moz *all* Saf *all* Op *7* DOM *1*

可读/可写

提供列分组内单元格的文本水平对齐的风格。

示例 document.getElementById("myCol").vAlign = "baseline";
值 不区分大小写的字符串常量：baseline | bottom | middle | top。
默认值 middle

width

IE 4 NN n/a Moz all Saf all Op 7 DOM 1

可读/可写

提供列分组内每一列的宽度，单位为像素。改变这些值后，会立刻反应在页面的内容排列上。

示例 `document.getElementById("myCol").width = 150;`

值 整数值。

默认值 无。

colgroup

IE 4 NN n/a Moz all Saf n/a Op 7 DOM 1

colgroup对应于colgroup元素，它可以将多个相邻列分配在一个组内，以便指定样式、宽度并进行其他可视化处理。

等价HTML元素 `<colgroup>`

对象模型引用方式 `[window.]document.getElementById("elementID")`

对象特定属性 align、ch、chOff、span、vAlign、width

对象特定方法 无。

对象特定事件 无。

align

IE 4 NN n/a Moz all Saf n/a Op 7 DOM 1

可读/可写

在colgroup元素控制的单元格中，此属性确定其水平对齐的风格。

示例 `document.getElementById("myColgroup").align = "center";`

值 以下3个对齐常量之一：center | char | left | right。

默认值 left

ch

IE 4 NN n/a Moz all Saf n/a Op n/a DOM 1

可读/可写

ch对应于char属性，它将一个文本字符定义为对齐参照点，列或列分组内的文本均依照该字符进行对齐。注意，只有将align属性设置为char，此属性才有价值。事实上目前所有的浏览器均不会响应这个属性。

示例 `document.getElementById("myColgroup").ch = ".";`

值 单个字符组成的字符串。

默认值 无。

chOff

IE 4 NN n/a Moz all Saf n/a Op n/a DOM 1

可读/可写

定义一个偏移点，char属性所指定的字符将出现在单元格中偏移点所在位置。事实上目前所有的浏览器均不会响应这个属性。

示例 `document.getElementById("myColgroup").chOff = "80%";`

值 字符串形式的像素值或单元格宽度的百分比。

默认值 无。

span

IE 4 NN n/a Moz all Saf n/a Op 7 DOM 1

可读/可写

提供相邻的列数量，元素的属性和样式设置均会应用于这些列。

示例 `document.getElementById("myColgroup").span = 2;`

值 整数值。

默认值 1

vAlign

IE 4 NN n/a Moz all Saf n/a Op 7 DOM 1

可读/可写

提供列分组内单元格的文本水平对齐的风格。

示例 `document.getElementById("myColgroup").vAlign = "baseline";`

值 不区分大小写的字符串常量：`baseline | bottom | middle | top`。

默认值 `middle`

width

IE 4 NN n/a Moz all Saf all Op 7 DOM 1

可读/可写

提供列分组内每一列的宽度，单位为像素。改变这些值后，会立刻在反应在页面的内容排列上。

示例 `document.getElementById("myColgroup").width = 150;`

值 整数值。

默认值 无。

comment, Comment

IE 4 NN n/a Moz all Saf all Op 7 DOM 1

comment对象对应于HTML文档中的“!”元素。 但在W3C DOM环境中，例如Mozilla内，这个对象并不是W3C DOM抽象模型中一个真实的元素。相反，这个对象只是一种特殊的节点，其nodeType值为8，表示它是一个Comment节点。在DOM抽象模型中，一个Comment节点拥有如下的遗传链：Node->CharacterData->Comment。然而在Comment节点中，会自动为它的某些属性设置特殊的值（例如，nodeValue为8），而且其属性和方法均完全继承自Node和CharacterData对象，不会包含任何其他的属性和方法。本章已经在共享项中讨论了Node对象的属性和方法，另外，Text对象也继承自CharacterData，它不仅涵盖了CharacterData对象的属性和方法，并且更易于实现脚本操作。

如果要引用一个注释元素，请使用相关的元素或节点属性。尽管根据其继承模型，IE为它提供了一个id属性，但依然不能通过id为这个元素指定标识符。然而比较奇怪的是，在IE中这样的一个元素的确拥有一个值为“!”的标签名称。因此，可以通过document.all.tags("!")方法返回的元素集合引用一个IE的HTML注释元素。但在Mozilla中，则只能引用位于body元素内的注释节点，而无法引用head元素内的注释。更有甚者，Safari内只能引用由脚本添加的注释节点，例如，通过document.createComment()或其他节点插入方法增加的注释。

等价HTML元素 *<!--comment text-->*

对象模型引用方式 *nodeReference*

对象特定属性 data、length、text

对象特定方法 appendData()、deleteData()、insertData()、replaceData()、tringData()

对象特定事件 无。

data

IE 6 NN n/a Moz all Saf all Op 7 DOM 1

可读/可写

提供注释内的文本内容。请参见Text.data。

length

IE 6 NN n/a Moz all Saf all Op 7 DOM 1

只读

提供注释信息中的字符数。请参见Text.length。

text

IE 4 NN n/a Moz n/a Saf n/a Op 7 DOM n/a

可读/可写

提供元素内的文本内容。根据IE中这种元素的本质特性，text的属性值同innerHTML和outerHTML的属性值完全相等。如果从浏览器查看文档的源代码，会发现修改这个属性并不会影响注释中的文本内容。但在IE 4/Macintosh组合中，本属性无法使用。因此，如果要实现跨浏览器访问，请使用data属性。

示例 `document.all.tags("!")[4].text = "Replaced comment, but no one will know.";`

值 字符串。

默认值 无。

appendData( ), deleteData( ), insertData( ), eplaceData( ), substringData( )

IE 6 NN n/a Moz all Saf all Op 7 DOM 1

提供操纵注释文本的一系列方法。请在Text对象中查看这些方法。

Context2D

请参见CanvasRenderingContext2D。

controlRange

IE 5(Win) NN n/a Moz n/a Saf n/a Op n/a DOM n/a

controlRange对象是一个元素对象集合，这些元素对象被document.body.controlRange()或document.selection.createRange()方法所创建，它们位于一个可编辑的容器中，而且处于被选中状态。在一个典型的编辑应用程序中，用户可以在编辑器的允许下从容器中选择任意的连续元素。然后脚本可以遍历controlRange集合（可能会过滤部分元素，例如img元素），并在集合中的元素上执行相关操作（其中包括通过execCommand()方法提供的指令）。微软为这一功能提供了一个示例程序，请访问此链接：*http://samples.msdn.microsoft.com/workshop/samples/author/dhtml/collections/controlrange.htm*。

对象模型引用方式 *controlRangeReference*

对象特定属性 length

对象特定方法 add()、addElement()、execCommand()、item()、queryCommandEnabled()、queryCommandIndeterm()、queryCommandState()、queryCommandSupported()、queryCommandValue()、remove()、scrollIntoView()、select()

对象特定事件 无。

add( )

IE 5 NN n/a Moz n/a Saf all Op n/a DOM n/a

```
add(elementRef[, index])
```

将一个由createElement()方法创建的元素添加至当前集合。

返回值 无。

参数

elementRef

一个完整的元素对象的引用，通常由createElement()方法生成。

index

一个可选的整数，表示新元素应该放置在集合中的哪个位置。

addElement( )

IE 5(Win) NN n/a Moz n/a Saf all Op n/a DOM n/a

```
add(elementRef)
```

将一个元素添加至当前集合。这是为controlRange集合而特设的一个方法。如果要调用execCommand()或select()方法，那么新元素应该紧邻集合中已存在的元素。

返回值 无。

参数

elementRef

一个完整的元素对象的引用。

execCommand( )

请参考Textrange.execCommand方法以获取更多详细信息。

item( )

IE 5(Win) NN n/a Moz n/a Saf all Op n/a DOM n/a

```
item(index)
```

从集合中返回一个单独的对象。

返回值 一个对象引用。如果并不存在与参数相匹配的对象，则返回null。

参数

index

一个从零开始计数的整数。

queryCommandEnabled( ), queryCommandIndeterm( ), queryCommandState( ), queryCommandSupported( ), queryCommandValue( )

请参考TexTRange对象中的对应方法，以获取更多详细信息。

remove( )

IE 5(Win) NN n/a Moz n/a Saf all Op n/a DOM n/a

```
remove(index)
```

从当前集合中删除一个元素。只须简单地设置一个从零开始计数的索引值，就可以从集合中删除对应的元素。

返回值 无。

参数

index

一个从零开始计数的整数，指明集合中的哪一项应该被删除。

scrollIntoView( )

IE 5(Win) NN n/a Moz n/a Saf all Op n/a DOM n/a

请参考本章起始部分，在“共享方法”内分析了scrollIntoView()方法。

select( )

IE 5(Win) NN n/a Moz n/a Saf all Op n/a DOM n/a

选择controlRange集合内的所有元素。

返回值 无。

参数 无。

CSSCharsetRule, CSSFontFaceRule, CSSImportRule, CSSMediaRule, CSSPageRule, CSSStyleRule, CSSUnknownRule

请参看CSSRule。

cssRule, CSSRule, rule

IE 5 NN n/a Moz all Saf all Op 9 DOM 2

样式单规则对象是文档内styleSheet对象集合的一个成员。IE和W3C DOM采用了不同的语法来引用这些规则对象。在IE中，通过rules集合引用rule对象,而在W3C中，IE 5/Macintosh、Mozilla、Safari和Opera 9则通过cssRules集合来引用cssRule对象。须要注意的是，在Windows系统下IE浏览器直到第7版依然不支持cssRule对象。另外，受IE 6/7内置安全机制的限制，脚本无法访问styleSheet对象的rules集合。因此尽管下文中罗列的某些属性和方法受IE支持，但实际上它们无法真正地得到使用，否则会引起“拒绝访问”这种安全性错误。

在W3C DOM中，cssRule的抽象对象称为CSSRule对象，但是只有在Mozilla中修改CSSRule对象的原型属性和方法时，脚本才需要这种形式的对象名称。W3C DOM进一步地为每种规则类型定义了不同类型的CSSRule对象，例如CSSImportRule、CSSMediaRule等，请参见type属性。cssRules集合中的成员对象属于这些类型中的任意一种，可以通过其type属性来识别它们的具体类型。每种不同的cssRule类型均包含其特有的属性和方法集合。因此，在下文中列出的属性和方法中，请注意它们具体适用于哪一种或几种类型。但总的来说，脚本中使用的规则通常都属于CSSStyleRule类型，它们拥有指向文档中不同元素的选择器。

但在实际使用过程中，如果要通过脚本访问rule或cssRule对象，仍须小心谨慎。如果修改了一个规则的选择器或样式定义，那么就会影响到整个文档，此外，如果错标了一个冒号，还会破坏文档中的其他规则。如果要在一个元素、类或元素类型上使用两个或多个样式风格，那么使用其他适用于多规则（切换元素的className属性）或多样式单（激活或禁用styleSheet对象）的技术会更为可靠而高效。但为了保证对象模型的完整性，如果确实须要访问样式单规则中的各个细节，W3C DOM也特别提供了完整的访问方法。

对象模型引用方式

```
document.styleSheets[i].rules[j]      // IE 6之前
document.styleSheets[i].cssRules[j]
```

对象特定属性 cssRules、cssText、encoding、href、media、parentRule、parentStyleSheet、readOnly、selectorText、style、styleSheet、type

对象特定方法 deleteRule()、insertRule()

对象特定事件 无。

cssRules

IE n/a NN n/a Moz all Saf all Op 9 DOM 2

只读

返回一个“@media”规则中内嵌的cssRule对象集合。

W3C DOM CSSRule类型 CSSMediaRule
值 一个cssRules集合对象的引用。
默认值 长度为0的数组。

cssText

IE *5(Mac)* NN *n/a* Moz *all* Saf *all* Op *9* DOM *2*

可读/可写

指明样式单规则中的完整文本，其中包括选择器和大括号内的名称/值对。IE 6/Windows并未提供等价的属性。在支持这一属性的浏览器中，改变这一属性并不会对对象产生影响，也不会改变显示效果。
W3C DOM CSSRule类型 所有类型。
示例

```
document.styleSheets[0].cssRules[2].cssText = "td {text-align:center}";
```

值 字符串。
默认值 无。

encoding

IE *n/a* NN *n/a* Moz *all* Saf *all* Op *9* DOM *2*

只读

返回与“@charset”规则相关的字符集编码，例如ISO-8859-1或UTF-8。
W3C DOM CSSRule类型 CSSCharsetRule
值 字符串。
默认值 无。

href

IE *n/a* NN *n/a* Moz *all* Saf *all* Op *9* DOM *2*

只读

返回一个URI，通过@import规则可导入该URI所指向的外部样式单文件。
W3C DOM CSSRule类型 CSSImportRule
值 字符串。
默认值 无。

media

IE *n/a* NN *n/a* Moz *all* Saf *all* Op *9* DOM *2*

只读

返回@import或@media规则所对应的媒体类型。
W3C DOM CSSRule类型 CSSImportRule、CSSMediaRule
值 字符串常量，表示浏览器所支持的媒体类型，例如“screen”或“print”。
默认值 all

parentRule

IE *n/a* NN *n/a* Moz *all* Saf *all* Op *9* DOM *2*

只读

包含当前cssRule对象的一个cssRule对象的引用，例如，一个样式规则内嵌在一个@rule中，则会返回“@rule”对象的引用。
W3C DOM CSSRule类型 所有类型。
示例

```
var superRule = document.styleSheets[0].cssRules[1].parentRule;
```

值 cssRule对象的引用。
默认值 null

parentStyleSheet

IE *5(Mac)* NN *n/a* Moz *all* Saf *all* Op *9* DOM *2*

459

只读

指向包含当前cssRule对象的styleSheet对象。如果将一个cssRule对象引用作为参数传入某方法，那么就可以获取这个cssRule对象父容器的styleSheet对象，这样也许可以更深入地了解样式单的其他信息。

W3C DOM CSSRule类型 所有类型。

示例
```
var ss = document.styleSheets[0].cssRules[3].parentStyleSheet;
```

值 styleSheet对象的引用。

默认值 当前对象。

readOnly

IE *5* NN *n/a* Moz *n/a* Saf *n/a* Op *n/a* DOM *n/a*

只读

如果使用@imprt规则或link元素导入规则，那么返回布尔值true。虽然可能无法通过脚本改变这些规则，但由于元素的样式修改只与其style属性相关而不会作用于rule对象，因此对于这些规则控制的元素而言，可以单独修改它们的样式属性。

示例
```
if (!document.styleSheets[2].cssRules[0].readOnly) {
    // 并非为只读状态，因此可以进行修改
}
```

值 布尔值：true、false。

默认值 随规则类型而变化。

selectorText

IE *5* NN *n/a* Moz *all* Saf *all* Op *9* DOM *2*

可读/可写

表示样式单规则中的选择器部分。尽管这个属性可以被读、写（IE 5/Mac除外），但它不会影响对应的对象或显示效果。

W3C DOM CSSRule类型 CSSPageRule、CSSStyleRule

示例
```
document.styleSheets[0].cssRules[2].selectorText = "td.leftHeaders";
```

值 字符串。

默认值 无。

style

IE *5* NN *n/a* Moz *all* Saf *all* Op *9* DOM *2*

460

可读/可写

返回一个style对象，其属性表示当前规则的属性设置。这个style对象与文档内元素的style对象属于同一种类型（对应于W3C DOM的CSSStyleDeclaration对象）。如果必须在规则层修改样式单设置，请通过rule或cssRule对象的style属性进行修改。相关的修改会立即对其自身进行注册，对应的元素也会根据修改内容相应地调整其显示效果。

W3C DOM CSSRule类型 CSSFontRule、CSSPageRule、CSSStyleRule

示例
```
var oneRule;
if (document.styleSheets) {
    if (document.styleSheets[0].cssRules) {
        oneRule = document.styleSheets[2].cssRules[1];
    } else if (document.styleSheets[0].rules) {
```

```
        oneRule = document.styleSheets[2].rules[1];
    }
}
if (oneRule) {
    oneRule.style.color = "red";
    oneRule.style.fontWeight = "bold";
}
```

值 style（W3C的CSSStyleDeclaration）对象的引用。

默认值 当前style对象。

styleSheet

IE *n/a* NN *n/a* Moz *all* Saf *all* Op *9* DOM *2*

只读

返回一个styleSheet对象引用，它包含在导入的样式单中。在这里就可以检查该styleSheet对象的cssRule对象，这实际上就是向下再多搜索一层，以到达远端样式单文件的styleSheet对象结构。

W3C DOM CSSRule类型 CSSImportRule

值 styleSheet对象的引用。

默认值 无。

type

IE *n/a* NN *n/a* Moz *all* Saf *all* Op *9* DOM *2*

只读

返回一个整数，它对应于W3C DOM定义的7个cssRule类型之一。在Mozilla中，每个cssRule对象均拥有明语常量属性，以表示其规则类型，如下所示。

常量	等价整数值	常量	等价整数值
cssRuleReference.UNKNOWN_RULE	0	*cssRuleReference*.MEDIA_RULE	4
cssRuleReference.STYLE_RULE	1	*cssRuleReference*.FONT_FACE_RULE	5
cssRuleReference.CHARSET_RULE	2	*cssRuleReference*.PAGE_RULE	6
cssRuleReference.IMPORT_RULE	3		

W3C DOM CSSRule类型 所有类型。

示例

```
var oneRule = document.styleSheets[2].cssRules[1];
if (oneRule.type == oneRule.IMPORT_RULE) {
    // 处理@import 规则
}
```

值 整数值。

默认值 1

deleteRule( )

IE *n/a* NN *n/a* Moz *all* Saf *all* Op *9* DOM *2*

deleteRule(*index*)

从当前@media规则中，删除某个从零开始计数的索引号所标示的规则。

W3C DOM CSSRule类型 CSSMediaRule

返回值 无。

参数

index

一个从零开始计数的整数，依照源代码次序，对应于集合中的某个特定项。

insertRule( )

IE *n/a* NN *n/a* Moz *all* Saf *all* Op *9* DOM *2*

```
insertRule("rule", index)
```

根据第2个参数所指明的位置，在当前@media规则中插入一个新的规则（选择器文本及样式单属性）。

W3C DOM CSSRule类型 CSSMediaRule

返回值 整数，表示插入的位置。

参数

rule

字符串，内含选择器及在大括号内包含待插入规则的样式属性。

index

一个从零开始计数的整数，依照源代码次序，对应于集合中的某个特定项。

cssRules, CSSRuleList, rules

IE *4* NN *n/a* Moz *all* Saf *all* Op *9* DOM *2*

一个cssRule或rule对象的集合，这3个对象均为同一个styleSheet对象的一份子。W3C DOM中，这个集合的抽象表示称为CSSRuleList对象。虽然只能通过其整数索引号直接访问集合中的成员，但也可以遍历整个集合并检查每个规则对象的属性（例如，selectorText属性）以便对它们进行区分。

对象模型引用方式

IE（Windows, IE 6之前）

```
document.styleSheets[i].rules
```

其他

```
document.styleSheets[i].cssRules
```

对象特定属性 length

对象特定方法 item()

对象特定事件 无。

length

IE *4* NN *n/a* Moz *all* Saf *all* Op *9* DOM *2*

只读

返回集合中元素的数量，其中包括@规则。

示例

```
var howMany = document.styleSheets[1].cssRules.length;
```

值 数值。

item( )

IE *4* NN *n/a* Moz *all* Saf *all* Op *9* DOM *2*

```
item(index)
```

返回一个样式单规则对象，其对应规则的源代码次序可与参数传入的索引值相匹配。

返回值 据具体的对象模型，返回一个cssRule或rule对象的引用。如果并不存在与参数相匹配的对象，则返回null。

参数

index

一个从零开始计数的整数，依照源代码次序，对应于集合中的某个特定项（内嵌在当前styleSheet对象之中）。

CSSStyleDeclaration

IE *4* NN *n/a* Moz *all* Saf *all* Op *7* DOM *1*

参见style。

CSSStyleSheet

IE *4* NN *n/a* Moz *all* Saf *all* Op *7* DOM *1*

参见styleSheet。

currentStyle

IE *4* NN *n/a* Moz *n/a* Saf *n/a* Op *9* DOM *n/a*

currentStyle对象为当前元素的有效（层叠的）样式属性提供了只读的访问方法，其中包含那些受链接、导入文件及显式样式单设置影响的样式属性。这个对象是所有可显示HTML元素对象共有的一个属性，而且它与元素的sytle对象有很大区别，对于已明确设置的内嵌style属性，它不仅可以对其进行描述还可以作出修改。

对象模型引用方式 `[window.]document.getElementById("elementID").currentStyle`

对象特定属性 请参看style对象。

对象特定方法 请参看style对象。

对象特定事件 无。

custom, HTMLUnknownElement

IE *5* NN *n/a* Moz *all* Saf *all* Op *7* DOM *1*

为开发者定义的元素提供脚本访问方法。这些元素共享普通HTML元素对象所拥有的属性、方法和事件处理程序，并且通常还拥有自定义的属性。Internet Explorer会将这些自定义属性（attribute）暴露为元素对象的属性（property）。

等价HTML元素 *<user-defined-tag>*

对象模型引用方式 `[window.]document.getElementById("elementID")`

对象特定属性 无。

对象特定方法 无。

对象特定事件 无。

dataTransfer

IE *5(Win)* NN *n/a* Moz *n/a* Saf *2* Op *n/a* DOM *n/a*

dataTransfer对象（可作为event对象的一个属性进行访问）是一个临时容器。Safari 2和Windows系统下IE 5及后续版本的IE浏览器可以使用这个临时容器来转移文本数据，一些脚本控制的操作尤其需要这个功能，例如模拟剪切、复制和粘贴，以及控制拖拽等。脚本可以控制存储在dataTransfer对象中的具体数据，如一个元素内的文本、整个元素的HTML代码或一个图像的URL。例如，一个针对小朋友的页面可以显示一些不同动物的小图标。如果用户开始拖拽小狗图标，那么img元素的onDragStart事件处理程序启动的一段脚本就会将该元素的一个自定义属性值（可能是一张小狗图片的URL）存储到dataTransfer对象中。当用户将图片拖至指定区域时，onDrop事件处理程序的方法会读取dataTransfer对象的数据，并且在页面上的指定位置加载对应的照片图像。

虽然在每个新的事件产生时，事件对象会改变其属性，但dataTransfer对象会在事件间保留其数据，直到脚本将数据从对象中删除或其他数据被存储到对象中。dataTransfer对象的一系列属性体现了它与clipboardData对象的区别。通过设置dropEffect和effectAllowed属性，可以让脚本控制拖拽时鼠标指针

图标的类型。示例2-1展示了如何将dataTransfer对象的属性和方法串接起来处理拖拽事件，在这个示例中，当鼠标滚动到指定的区域时，鼠标指针会变成“复制”样式。

示例 2-1：使用 dataTransfer 对象

```
<html>
<head>
<title>dataTransfer Demo</title>
<style type="text/css">
td {text-align:center}
th {text-decoration:underline}
.cyan {color:cyan}
.yellow {color:yellow}
.magenta {color:magenta}
#blank1 {text-decoration:underline}
</style>
<script type="text/javascript">
// 在拖拽一个起始元素的进行此阶段的相关信息设置
function setupDrag(evt) {
    evt = (evt) ? evt : window.event;
    var elem = (evt.target) ? evt.target : evt.srcElement;
    if (elem.nodeType == 3) {elem = elem.parentNode;}
    if (elem.tagName != "TD") {
        // 不允许拖拽任何其他元素
        evt.returnValue = false;
    if (evt.preventDefault) {evt.preventDefault();}
    } else {
        // 让指针看起来和复制动作一样
        evt.dataTransfer.effectAllowed = "copy";
        // 在 dataTransfer 对象中存储拖拽的单元格文本
        evt.dataTransfer.setData("Text", elem.innerHTML);
    }
}
// 执行设置操作
function handleDrop( evt) {
    evt = (evt) ? evt : window.event;
    var elem = (evt.target) ? evt.target : evt.srcElement;
    var passedData = event.dataTransfer.getData("Text");
    if (passedData) {
        // 显示放置目标指针
        event.dataTransfer.dropEffect = "copy";
        // 在放置目标上应用之前获得的数据
        elem.innerHTML = passedData;
        elem.className = passedData;
        if (document.selection) {
            document.selection.empty();
        }
    }
}
// 禁止在此事件上实现拖拽/复制操作
function cancelDefault(evt) {
    evt = (evt) ? evt : window.event;
    // 将指针设置成"No"
    evt.dataTransfer.dropEffect = "copy";
    evt.returnValue = false;
    if (evt.preventDefault) {evt.preventDefault();}
}
</script>
<head>
<body ondragstart="setupDrag(event);">
```

```
<table cellpadding="5">
<tr><th>Select and Drag Your Favorite Color</th></tr>
<tr><td>cyan</td></tr>
<tr><td>yellow</td></tr>
<tr><td>magenta</td></tr>
</table>

<p>My favorite color is <span id="blank1" ondragenter="cancelDefault(event);"
ondragover="cancelDefault(event);" ondrop="handleDrop(event);">
     </span> .</p>
</body>
</html>
```

使用这个对象及clipboardData对象来传递数据的更多信息，请访问以下链接：*http://msdn.microsoft.com/en-us/library/ms537658(VS.85).aspx*。

等价HTML元素 无。

对象模型引用方式 [window.]event.dataTransfer

对象特定属性 dropEffect、effectAllowed

对象特定方法 clearData() | getdata() | setData()

对象特定事件 无。

dropEffect, effectAllowed

IE *5(Win)* NN *n/a* Moz *n/a* Saf *2* Op *n/a* DOM *n/a*

可读/可写

这两个属性彼此协同工作，但它们分别在dataTransfer对象调用的拖拽操作的不同时期发挥着作用。它们均能够控制鼠标指针在拖拉过程中的外观。在拖动操作开始时，可以通过ondragstart事件和effectAllowed属性指定一个鼠标指针样式。然后，放置目标（drop target）的ondragover和ondragenter事件处理程序应该将dropEffect属性设置为预期的鼠标指针样式，同时还会将event.returnValue属性设置为false。这使得ondrop事件处理程序不仅可以通过dropEffect属性设置鼠标指针样式，还可以处理放置动作。请参看示例2-1，它简单地示范了以上这些事件和属性彼此间如何交互作用。

示例 event.dataTransfer.dropEffect= "copy";

值 不区分大小写的字符串常量：copy | link | move | none。

默认值 none

clearData( )

IE *5(Win)* NN *n/a* Moz *n/a* Saf *2* Op *n/a* DOM *n/a*

clearData([*dataFormat*])

从dataTransfer对象内删除数据。

返回值 无。

参数

dataFormat

一个可选的字符串，指定须要删除哪种格式的数据。曾经计划在clipboardData对象中保存多种类型的数据，不过从目前看来，这个计划已经失败了。Text是唯一一种可靠的数据类型。如果忽略这个参数，那么将删除所有类型的数据。

getData( )

IE *5(Win)* NN *n/a* Moz *n/a* Saf *2* Op *n/a* DOM *n/a*

getData(*dataFormat*)

返回一份来自于dataTransfer对象的数据。而dataTransfer中的内容将保持不变，以便后来的脚本指令能够继续读取这些数据。

返回值 字符串。

参数

dataFormat

一个字符串，指明须要保存的数据类型。曾经计划在clipboardData对象中保存多种类型的数据，不过从目前看来，这个计划已经失败了。Text是唯一一种可靠的数据类型。

setData( )

IE *5(Win)* NN *n/a* Moz *n/a* Saf *2* Op *n/a* DOM *n/a*

```
setData(dataFormat, stringData)
```

在dataTransfer对象中保存字符串数据。如果赋值成功，那么返回布尔值true。

返回值 布尔值：true | false。

参数

dataFormat

一个字符串，指明须要保存的数据类型。曾经计划在clipboardData对象中保存多种类型的数据，不过从目前看来，这个计划已经失败了。Text是目前唯一一种可靠的数据类型。尽管此方法可以将URL作为一种输入类型，但是却无法成功读取这种类型的数据。

stringData

任意字符串，可以包含HTML标签。

dd

IE *4* NN *n/a* Moz *all* Saf *all* Op *7* DOM *1*

dd对象对应于dd元素。

等价HTML元素 <dd>

对象模型引用方式 [window.]document.getElementById("*elementID*")

对象特定属性 noWrap

对象特定方法 无。

对象特定事件 无。

noWrap

IE *4* NN *n/a* Moz *n/a* Saf *n/a* Op *n/a* DOM *n/a*

可读/可写

指明浏览器是否应该按照需要加宽元素的宽度，以便将一行不间断的文本显示在一整行上。当单元格内内容很多时，如果滥用此属性，那么为了显示所有内容会在页面上出现一个非常不便于使用的横向滚动条。

示例 document.getElementById("wideBody").noWrap = "true";

值 布尔值：true | false。

默认值 false

del

IE *4* NN *n/a* Moz *all* Saf *all* Op *7* DOM *1*

del对象对应于del元素。

等价HTML元素 <del>

对象模型引用方式 [window.]document.getElementById("*elementID*")

对象特定属性 cite | dateTime

对象特定方法 无。

对象特定事件 无。

469 cite, dateTime IE 4 NN n/a Moz all Saf all Op 7 DOM 1

可读/可写

由于IE 6和IE 7实现了这两个古怪的属性，因此本章还是将它们罗列在共享属性之内。而IE 5/Macintosh、Mozilla、Safari和Opera则只能在del和ins对象中按照W3C DOM的说明正确地实现这两个属性，但主流浏览器均未给它们赋予任何特殊的功能。请参见cite和dateTime属性。

dfn

请参见abbr。

Dialog Helper

IE 6 NN n/a Moz n/a Saf n/a Op n/a DOM n/a

Dialog Helper是同IE 6和IE 7一同发布的一个ActiveX控件，它用于提供系统及文档的信息组合，而这个方法也可以用于显示一个颜色选择对话框，以便脚本从中获取用户的颜色选择。通常在使用脚本操作IE的编辑模式时会使用这个对象，此时用户须要选择颜色、字体和元素。另外，在传统的浏览器文档设置中，也能够发现其属性和方法的用处。

在页面中加载这个对象须要使用<object>标签，如下所示：

```
<object id="dlgHelper" classid="clsid:3050f819-98b5-11cf-bb82-00aa00bdce0b"
width="0px" height="0px"></object>
```

由于页面并不会显示这个对象，因此可以将它的标签放置在文档开头部分。还可以根据需要设置id属性的标识符。一旦加载这个对象，可以将它作为窗口中的全局对象进行引用。

等价HTML元素 无。

对象模型引用方式 [window.]document.getElementById("*elementID*")

对象特定属性 blockFormats、fonts

对象特定方法 ChooseColorDlg()、getCharset()

对象特定事件 无。

470 blockFormats IE 6 NN n/a Moz n/a Saf n/a Op n/a DOM n/a

只读

返回由浏览器所支持的全部块级元素的明语名称所组成的集合。与其他IE集合不同，如果要读取集合中成员项的数目，必须访问其Count属性而不是length属性。集合中每一项的名称都是字符串形式，例如“Heading 1”和“Numbered List”，它们分别对应于h1和ol元素。请通过集合的item()方法访问集合中的每一项。

示例

```
var blockList = dlgHelper.blockFormats;
var blockNames = new Array();
for (var i = 0; i < blockList.Count; i++) {
   blockNames[blockNames.length]= blockList.item(i);
}
```

值 字符串数组。

默认值 依赖于浏览器的具体实现。

fonts

IE 6 NN n/a Moz n/a Saf n/a Op n/a DOM n/a

只读

返回由系统字体明语名称组成的集合。与其他IE集合不同，如果要读取集合中成员项的数目，必须访问其

Count属性而不是length属性。集合中每一项的名称都是字符串形式，例如“MS Sans Serif”和“Verdana”。可以通过集合的item()方法访问集合中的每一项。

示例

```
var fontList = dlgHelper.fonts;
var fontNames = new Array();
for (var i = 0; i < fontList.Count; i++) {
   fontNames [fontNames .length]= fontList .item(i);
}
```

值　　字符串数组。

默认值　　依赖于浏览器的具体实现。

ChooseColorDlg( )

IE *6* NN *n/a* Moz *n/a* Saf *n/a* Op *n/a* DOM *n/a*

```
ChooseColorDlg([initialHexColor])
```

显示一个颜色选择对话框，并根据用户的颜色选择返回一个十进制数字。如果要将用户选择的颜色应用于样式或其他颜色属性设置，必须将十进制的数值转换成“#RRGGBB”格式的十六进制三元组。在下面这段代码中，首先获取颜色，然后按照预期的设定将十进制数字转换成另一种形式的数字值，最后将它赋值给一个样式属性：

```
var colorChoice = dlgHelper.ChooseColorDlg();
var hexColor = colorChoice.toString(16);
while (hexColor.length < 6) {hexColor = "0" + hexColor;}
document.body.style.color = "#" + hexColor;
```

虽然用户在对话框中可以选择一个自定义颜色，并将它添加至一个小型的快捷调用框中，但当对话框第二次显示时，之前选择的颜色并不会出现在快捷调用框中。但如果将某个自定义颜色的十六进制值作为参数传输给此方法，那么依然可以预先选定这个颜色。

返回值　　用户选定颜色所对应的十进制整数（从0到客户端所能支持的颜色数）。

参数

initialHexColor

可选的十六进制数字，它表示对话框中初始状态下预先选择的颜色。

getCharset( )

IE *6* NN *n/a* Moz *n/a* Saf *n/a* Op *n/a* DOM *n/a*

```
getCharset("fontName")
```

返回一个对应于某常量的整数，该常量与操作系统已知的某个字符集相关。针对安装在拉丁文系统上的字符集，返回值一般是0（针对简单的ANSI字符集）和2（针对符号集合）。此方法的必选参数是一个字体的名称，方法会检索它的字符集。这些名称可以从Dialog Helper对象的字体属性中获得，其代码如下所示：

```
var setID = dlgHelper.getCharset(dlgHelper.fonts.item(4));
```

但并不是所有的Windows版本均安装了同样的字符集组合。

返回值　　整数值。

参数

fontName

已安装的系统字体的字符串名称。

dir

IE *4* NN *n/a* Moz *all* Saf *all* Op *7* DOM *1*

dir对象对应于dir元素。最初是想将这个元素作为一个多栏列表格式，但目前它与ul元素的处理方式一致。

等价HTML元素 <dir>
对象模型引用方式 [window.]document.getElementById("*elementID*")
对象特定属性 compact
对象特定方法 无。
对象特定事件 无。

compact

IE *5(Mac)/6(Win)* NN *n/a* Moz *all* Saf *all* Op *7* DOM *1*

可读/可写

出于保证W3C DOM规范兼容性的需要，为dir元素提供了这个属性。然而主流浏览器并不会受这个属性的影响。

值 布尔值：true | false。
默认值 false

directories, locationbar, menubar, personalbar, scrollbars,statusbar, toolbar

IE *n/a* NN *4* Moz *all* Saf *n/a* Op *n/a* DOM *n/a*

这些对象属于window对象，它们描述了环绕在浏览器窗口内容区域四周的“装饰”部分。Mozilla还添加了directories对象。通过Navigator 4或Mozilla中的签名脚本（signed script）及用户许可，可以在浏览器窗口中动态隐藏或显示这些属性。还可使用window.open()方法的第3个参数来关闭这些功能部件，但这个方法只能在创建新窗口时才能使用。如果要在一个已经存在的窗口中改变它们的可见性，必须使用签名脚本。对已经打开的窗口，其他浏览器并未提供等价功能来改变它们的可见性。

对象模型引用方式

```
[window.]directories
[window.]locationbar
[window.]menubar
[window.]personalbar
[window.]scrollbars
[window.]statusbar
[window.]toolbar
```

对象特定属性 visible
对象特定方法 无。
对象特定事件 无。

visible

IE *n/a* NN *4* Moz *all* Saf *n/a* Op *n/a* DOM *n/a*

可读/可写

只能通过Navigator 4或Mozilla中的签名脚本才能访问，它决定了是否显示窗口的装饰功能部件。

示例

```
netscape.security.PrivilegeManager.enablePrivilege("UniversalBrowserWrite");
window.statusbar.visible = "false";
netscape.security.PrivilegeManager.revertPrivilege("UniversalBrowserWrite");
```

值 布尔值：true | false。
默认值 true

div

IE *4* NN *n/a* Moz *all* Saf *all* Op *7* DOM *1*

div对象对应于div元素。这个元素创建一个块级元素，它通常用于元素定位或几个相关元素的包含关系分组。在IE/Macintosh中，其客户端和滚动属性是有效的。

等价HTML元素
对象模型引用方式 [window.]document.getElementById("*elementID*")
对象特定属性 align、dataFld、dataFormatAs、dataSrc、noWrap
对象特定方法 无。
对象特定事件 无。

align

IE *4* NN *n/a* Moz *all* Saf *all* Op *7* DOM *1*

可读/可写

定义了元素盒中相关内容的水平对齐方式。除非加以控制，否则元素盒的宽度会与其临近最外层定位环境（通常是body元素）的宽度一致。

示例 document.getElementById("myDIV").align = "center";
值 以下3个对齐常量之一：center | left | right。
默认值 left

dataFld

IE *4* NN *n/a* Moz *n/a* Saf *n/a* Op *n/a* DOM *n/a*

可读/可写

此属性用于IE中的数据绑定，以便将一个远程数据源的列名与一个div元素的内容联系在一起。此时元素中还必须设定datasrc属性。如果dataFld和dataSrc属性均设置为空字符串，那么就会打破元素与数据源之间的绑定关系。须要注意的是，这一功能只能在IE 5/Mac组合的文件数据源中正常工作。

示例 document.getElementById("myDiv").dataFld = "comment";
值 数据源数据列内区分大小写的标识符。
默认值 无。

dataFormatAs

IE *4* NN *n/a* Moz *n/a* Saf *n/a* Op *n/a* DOM *n/a*

可读/可写

本属性用于IE浏览器中的数据绑定。对于来自于数据源的原始资料，浏览器根据此属性值决定是将它们视为纯文本亦或是HTML标签。

示例 document.getElementById("myDiv").dataFormatAs = "text";
值 字符串常量：text | html。
默认值 text

dataSrc

IE *4* NN *n/a* Moz *n/a* Saf *n/a* Op *n/a* DOM *n/a*

可读/可写

为实现IE数据绑定功能，此属性用于指定页面内object元素的ID值，以便在进行远程数据访问时加载数据源对象。通过datafld属性指定来自于数据源中的绑定内容。如果dataFld和dataSrc属性均设置为空字符串，那么就会打破元素与数据源之间的绑定关系。

示例 document.getElementById("myDiv").dataSrc = "DBSRC3";
值 数据源内区分大小写的标识符。
默认值 无。

noWrap

IE 4 NN n/a Moz n/a Saf n/a Op n/a DOM n/a

可读/可写

指明浏览器是否应该按照需要加宽元素的宽度，以便将一行不间断的文本显示在一整行上。当单元格内内容很多时，如果滥用此属性，那么为了显示所有内容，会在页面上出现一个非常不便于使用的横向滚动条。不建议使用与之对应的元素属性。

示例 `document.getElementById("wideDiv").noWrap = "true";`

值 布尔值：`true | false`。

默认值 `false`

dl

IE 4 NN n/a Moz all Saf all Op 7 DOM 1

dl对象对应于dl元素。这个元素是一个定义列表分组的包装容器。

等价HTML元素 `<dl>`

对象模型引用方式 `[window.]document.getElementById("elementID")`

对象特定属性 `compact`

对象特定方法 无。

对象特定事件 无。

compact

IE 4 NN n/a Moz all Saf all Op 7 DOM 1

可读/可写

如果将compact属性设置为true，它会命令浏览器在空间允许的情况下在同一行显示相关的dt和dd元素对。但这种紧凑的格式完全可以用于仅包含少量几个字符的dt元素。

示例 `document.getElementById("maindl").compact = true;`

值 布尔值：`true | false`。

默认值 `false`

document

IE 3 NN 2 Moz all Saf all Op 7 DOM 1

document对象描述了浏览器窗口或框架中已显示的内容，以及从HTML文件中加载的其他内容。因此，文档中head部分的所有信息也是document对象的一部分。除“document”之外，document对象并没有其他的别名。

如果一个浏览器的内部架构忠实于W3C DOM，那么其document对象将代表HTMLDocument对象，而这个HTMLDocument对象则是一种特殊的核心模块的Document对象，它适合于保持HTML文档。换句话说，HTMLDocument对象继承了核心Document和Node对象的属性和方法（使用XML文档共享相关功能），并且还获得了一些仅适用于HTML文档的额外属性和方法。当然，XHTML文档从本质上来说是一个XML文档，因此HTMLDocument能否适用于XHTML文档在概念上还存一定的不定因素。但在实际应用中，并不存在这个问题，为了实现脚本操作，XHTML文档可以转换成HTML文档，从而使得所有的HTMLDocument属性和方法均可应用于该文档。

对W3C DOM实现而言，还有更为重要的实际的一面，即document对象内部实现了来自于其他DOM模型的文档级属性和方法，如视图、事件和样式等。Mozilla、Safari和Opera等浏览器均证明了这一情况的真实性。每个模型均定义了一个对象，如DocumentEvent、DocumentRange、DocumentStyle和DocumentView等，而

且这些对象在HTMLDocument和附加的模型功能之间建立了动态的联系。因此，可进行脚本操作的document对象正是使用W3C DOM中DocumentStyle对象的styleSheets属性来获取styleSheet对象及其规则。与此同时，DocumentEvent对象也与其createEvent()方法联系在一起，这使得document对象可以在用户/系统所创建的常规事件之外生成其他事件。这样一来，在Mozilla和其他以W3C DOM为中心的浏览器中，所有这些功能均被纳入到document对象之中。因此对脚本创建者而言，某个特定功能的准确来源模型已不是那么重要，只须要明确它们在document对象中对应的属性和方法即可。

对象模型引用方式 [window.]document

对象特定属性 activeElement、alinkColor、all[]、anchors[]、applets[]、bgColor、body、charset、characterSet、compatMode、contentType、cookie、defaultCharset、defaultView、designMode、doctype、documentElement、documentURI、domain、domConfig、embeds[]、expando、fgColor、fileCreatedDate、fileModifiedDate、fileSize、fileUpdatedDate、forms[]、frames[]、height、ids[]、images[]、implementation、inputEncoding、lastModified、layers[]、linkColor、links[]、location、media、mimeType、nameProp、namespaces[]、parentWindow、plugins[]、preferredstylesheetSet、protocol、readyState、referrer、scripts[]、security、selectedstylesheetSet、selection、strictErrorChecking、styleSheets[]、tags[]、title、URL、URLUnencoded、vlinkColor、width、xmlEncoding、xmlStandalone、xmlVersion

对象特定方法 addBinding()、adoptNode()、captureEvents()、clear()、close()、createAttribute()、createAttributeNS()、createCDATASection()、createComment()、createDocumentFragment()、createElement()、createElementNS()、createEntityReference()、createEvent()、createEventObject()、createNodeIterator()、createProcessingInstruction()、createRange()、createStyleSheet()、createTextNode()、createTreeWalker()、elementFromPoint()、execCommand()、getAnonymousElementByAttribute()、getAnonymousNodes()、getBindingParent()、getElementById()、getElementsByName()、getOverrideStyle()、getSelection()、handleEvent()、hasFocus()、importNode()、loadBindingDocument()、normalizeDocument()、open()、postMessage()、queryCommandEnabled()、queryCommandIndeterm()、queryCommandState()、queryCommandSupported()、queryCommandText()、queryCommandValue()、recalc()、releaseEvents()、removeBinding()、renameNode()、routeEvent()、write()、writeln()

对象特定事件

事件	IE	其他	DOM
selectionchange	4	n/a	n/a
stop	4	n/a	n/a

activeElement

IE *4* NN *n/a* Moz *n/a* Saf *n/a* Op *7* DOM *n/a*

只读

指向文档内正处于活跃状态的元素对象。如果要更为详细地了解返回对象的信息，须要检查该对象的tagName、id或其他属性。由于在所有的操作系统平台上，都可能无法在按钮和其他部分元素上聚焦，因此

这个属性的返回值可能随着操作系统的不同而发生改变。须要注意的是，如果某个元素获得了焦点，那么该元素就会成为活跃元素，但在新IE版本中，其他相关设定会使得一个已激活的元素并未获得焦点。请参见共享的setActive()方法。

示例 `var currObj = document.activeElement;`
值 元素对象的引用。
默认值 无。

alinkColor

IE 3 NN 2 Moz all Saf all Op 7 DOM n/a

可读/可写

指定一个超文本链接的颜色，当它被点击时会显示该颜色。设置这个属性后，文档中链接文字、图像的边框或嵌入a元素之中的对象均会应用这个颜色。请参考linkColor和vlinkColor属性以了解未访问和已访问链接的颜色。在W3C DOM中，它已被body对象的aLink属性所代替，而在CSS中，则使用:active伪类代替这个属性。在Mozilla之前的Navigator浏览器中，动态改变alinkColor的值并不会直接体现在页面上。

示例 `document.alinkColor = "green";`
值 一个十六进制的三元组或明语颜色名称。目前可用的明语颜色名称请参见附录A。
默认值 `#0000FF`

all[ ]

IE 4 NN n/a Moz 1.7 Saf all Op 7 DOM n/a

只读

从IE 4和Opera 7开始，all属性是这些浏览器中所有容器中的一员，但在Mozilla 1.7及后续版本中，它只是document对象的一员。由于旧脚本代码会直接使用document.all进行引用，而不会检查是否存在all属性，因此Mozilla添加了这个属性以便旧代码进行调用。但就这一点而言，即使使用if条件判断表达式发现并不存在有效的all属性，Mozilla也能在标记为“quirks”模式（请参见第1章中的DOCTYPE元素）的页面上识别document.all。在IE中可以使用document.all.*elementID*进行元素引用，当然，这种使用ID来引用单个元素的方法也适用于“quirks”模式下的Mozilla浏览器。请参见本章中已介绍过的all集合对象。

值 文档中元素对象的集合。
默认值 当前集合。

anchors[ ]

IE 3 NN 2 Moz all Saf all Op 7 DOM 1

返回当前文档中所有anchor对象组成的一个数组。此时无论a元素是作为锚链、组合锚链还是链接，均会包含在这个组合中。数组中的条目会根据源代码顺序从0开始建立索引。

示例 `var aCount = document.anchors.length;`
值 anchor元素对像数组。
默认值 长度为0的数组。

applets[ ]

IE 4 NN 3 Moz all Saf all Op 7 DOM 1

返回当前文档中所有Java applet对象组成的一个数组。小程序必须被启动并处于运行状态才能成为一个对象。数组中的条目会根据源代码顺序从0开始建立索引。

示例 `var appletCount = document.applets.length`
值 applet元素对像数组。
默认值 长度为0的数组。

bgColor

IE 3 NN 2 Moz all Saf all Op 7 DOM n/a

可读/可写

提供文档的背景色。即使使用明语颜色名称设置元素或对象的bgcolor属性，其返回值仍然是十六进制三元组。

通过CSS控制body元素的background属性来管理文档背景色是目前较为流行的一种方式。

示例 `document.bgColor = "yellow";`

值 一个十六进制的三元组或明语颜色名称。目前可用的明语颜色名称请参见附录A。

默认值 由浏览器和操作系统共同决定。

body

IE 4 NN n/a Moz all Saf all Op 7 DOM 1

只读

返回由文档内body元素定义的body对象的引用。实际使用时往往将这个属性作为body对象属性的一个入口点。

示例 `document.body.style.marginLeft = "15px";`

值 对象引用。

默认值 当前body对象。

charset

IE 4 NN n/a Moz n/a Saf n/a Op 9 DOM n/a

可读/可写

指明文档内容中所使用的字符编码方式。在Opera之外的其他浏览器中，请使用document.characterSet属性。

示例

```
if (document.charset == "csISO5427Cyrillic") {
    // 处理西里尔(Cyrillic)字符集
}
```

值 是否区分大小写取决于字符集注册表，请访问以下链接获取更多详细信息：*http://www.iana.org/assignments/character-sets*。

默认值 由浏览器决定。

characterSet

IE n/a NN n/a Moz all Saf all Op 9 DOM n/a

只读

指明文档内容中所使用的字符编码方式。

示例

```
if (document.characterSet == "ISO-8859-1") {
    // 处理标准的拉丁文字符集
}
```

值 是否区分大小写取决于字符集注册表，请访问以下链接获取更多详细信息：*http://www.iana.org/assignments/character-sets*。

默认值 由浏览器决定。

compatMode

IE 6 NN n/a Moz 1.0.1 Saf n/a Op 7 DOM 1

只读

为文档返回兼容模式，DOCTYPE元素的具体内容控制着它的兼容模式。请参见第1章中有关DOCTYPE元素的讨论，那里详细讲述了如何迫使浏览器按后向兼容或标准兼容模式处理文档，以完成元素定位和其他的实现细

节。由于不同模式的选择会影响某些样式属性的结构，因此可以使用这个属性区分共享库中的两种计算方式，以便浏览器在各种模式下均能正确地处理当前文档。

示例

```
if (document.compatMode == "BackCompat") {
    // 按照"quirks"模式进行处理
}
```

值　　字符串常量：BackCompat、CSS1Compat。

默认值　　BackCompat

contentType

IE *n/a* NN *n/a* Moz *1.0.1* Saf *n/a* Op *n/a* DOM *n/a*

只读

根据服务器传达的信息，为文档返回其MIME类型，

示例

```
if (document.contentType != "text/html") {
  // 对替代的MIME类型进行处理
}
```

值　　字符串。

默认值　　text/html。

cookie

IE *3* NN *2* Moz *all* Saf *all* Op *7* DOM *1*

可读/可写

指明与文档域相关并已保存在客户机上的HTTP cookie。Mozilla浏览器将所有的cookie数据组合在一个文件内，而IE则为每个域的cookie数据分别创建了一个独立的文件。

读、写cookie属性并不是并行的操作。读取一个cookie属性时，会返回一个由分号分隔的名称/值对列表，其格式如下：

```
name=value
```

在cookie属性中，可以为一个给定的域存储多达20个名称/值对（不必考虑在那个Web站点中HTML文档的数量）。在cookie中可以存储4 000个字符，但是最好还是将每个名称/值对的长度保持在2 000个字符以下为佳。通常由脚本代码来决定如何解析cookie属性值以获取一个单独的已命名的cookie值。

记录cookie属性值时允许将多个可选的数据对与一个单独的名称/值对关联起来。cookie数据必须是一个字符串，但可以使用Array.join()方法将一个数组转换成一个字符串以便记录cookie值，在读取cookie数据之后，可以使用String.split()方法来重建数组。其格式如下所示：

```
document.cookie = "name=value
  [; expires=timeInGMT]
  [; path=pathName]
  [; domain=domainName]
  [; secure]";
```

须要注意的是，无论在cookie中设置了多少个可选的子属性，始终只能检索到名称/值对。此外，所有写入cookie属性的cookie数据均保存在浏览器的内存中，直到浏览器退出。在浏览器退出时，如果cookie数据包含截止日期并且当前时间并未超出该日期，那么会将对应的cookie数据保存到真实的cookie文件中，否则这些数据将被直接丢弃。当浏览器再次启动时，会自动删除超时的cookie数据。

示例

```
var exp = new Date();
```

```
var nowPlusOneWeek = exp.getTime() + (7 * 24 * 60 * 60 * 1000);
exp.setTime(nowPlusOneWeek);
document.cookie = "userName=visitor; expires=" + exp.toGMTString();
```

值 字符串形式的cookie数据。参见上文中的详细说明。

默认值 无。

defaultCharset

IE *4* NN *n/a* Moz *n/a* Saf *n/a* Op *n/a* DOM *n/a*

只读

指明文档内容中所使用的字符编码。

示例
```
var cset = document.defaultCharset;
```

值 是否区分大小写取决于字符集注册表，请访问以下链接获取更多详细信息：*http://www.iana.org/assignments/character-sets*。

默认值 由浏览器决定。

defaultView

IE *n/a* NN *n/a* Moz *all* Saf *all* Op *7* DOM *2*

只读

返回“查看器”的W3C DOM抽象描述的引用。这种“查看器”对象显示了当前文档，其正式名称是AbstractView。在Mozilla和Opera中，这个对象等价于包含当前文档的window或frame对象。在这些浏览器中，任意一个可以访问document对象的脚本方法（例如，使用元素对象的ownerDocument属性）都能够通过defaultView属性获得文档所处窗口的一个有效引用。从另一方面来看，Safari也会返回AbstractView对象实例的一个引用。这种文档视图（或对象引用体系）包含每个元素所使用的层叠样式规则的相关信息。请参看在线参考V，它使用一个示例说明了如何借助于document.defaultView属性和getComputedStyle()方法来获取元素的有效样式。

示例
```
var elem = document.getElementById("myDiv");
var vw = document.defaultView;
var currStyle = vw.getComputedStyle(elem, "");
var elemLeft = currStyle.getPropertyValue("left");
```

值 在Mozilla和Opera内，返回一个window对象的引用，而在应用了W3C DOM的ViewCSS对象的Safari浏览器内，则返回AbstractView对象实例的引用。

默认值 文档的窗口或视图。

designMode

IE *5.5* NN *n/a* Moz *1.3* Saf *1.3/2* Op *9* DOM *n/a*

可读/可写

对用户是否能够编辑文档进行控制。由于这个属性能够控制整个文档，因此更为谨慎的一种做法是在iframe元素内的主页面上创建一个单独的可以编辑的部分，并设置该iframe内文档的designMode属性。

开启designMode后，支持这一功能的浏览器还会添加一些额外的文档方法以便脚本调用，特别是执行命令方法（例如，execCommand()）可以提供字体和其他文本样式，从而修改用户在可编辑文档中选择的文本。

如要获取Mozilla环境下使用designMode和相关功能的详细说明，请访问如下链接：*https://developer.mozilla.org/en/Rich-Text_Editing_in_Mozilla*。

示例
```
document.getElementById("editMe").designMode = "on";
```

值 字符串常量：off | on。IE浏览器还允许使用Inherit。

默认值 off（W3C）；Inherit（IE）。

doctype

IE 5(Mac) NN n/a Moz all Saf n/a Op 7 DOM 1

只读

返回一个DOCTYPE元素对象（与W3C DOM抽象DocumentType节点对象相同）的引用。只有在文档中指定DOCTYPE后，此属性才会返回一个引用，否则它将返回null。请参考DocumentType对象，以便了解在不同浏览器中可用的对象属性。在一个纯W3C DOM的环境内，doctype属性继承自核心的document对象，因此它也可用于XML文档。

示例 `var docsType = document.doctype;`

值 节点引用。

默认值 无。

documentElement

IE 5 NN n/a Moz all Saf all Op 7 DOM 1

只读

返回文档根元素节点的一个引用。在HTML文档中，该引用是html元素，它包围着文档的head和body元素。在一个纯W3C DOM的环境内，documentElement属性继承自核心的document对象，因此它也可用于XML文档。

示例 `var rootElem = document.documentElement;`

值 元素节点的引用。

默认值 当前html元素对象。

documentURI

IE n/a NN n/a Moz 1.7 Saf all Op 9 DOM 3

可读/可写

指明当前文档的URI。此值与location.href相同，它还可以导航到另一个不同的URI。

示例 `document.documentURI = "http://www.megacorp.com";`

值 完整或相对的统一资源标识符字符串。

默认值 当前文档的URI。

domain

IE 4 NN 3 Moz all Saf all Op 7 DOM 1

可读/可写

提供支持当前文档的服务器的主机名。如果同一域中来自于不同服务器的文档必须在彼此之间交换内容，那么必须将交换双方的domain属性设置为同一个域，以免出现安全约束问题。通常情况下，如果主机并不匹配，那么基于安全考虑，浏览器将不允许访问其他文档表单的数据。使用该属性就可以允许以上情况发生，例如，它可以使一个来自于“www”服务器的页面与一个由安全服务器支持的页面进行通信。

示例 `document.domain = "megaCorp.com";`

值 两个文档共有的域名字符串，此时不考虑服务器名称。

默认值 无。

domConfig

IE n/a NN n/a Moz n/a Saf n/a Op n/a DOM 3

只读

返回一个DOMConfiguration对象，它能够显示当前文档的很多配置特征。在与XML文档一起协同工作时，这些特征往往更为有用。

示例 `var paramList = document.domConfig.parameterNames;`

值 DOMConfiguration对象。
默认值 当前DOMConfiguration对象。

embeds[]

IE 4 NN 3 Moz all Saf all Op 7 DOM n/a

返回当前文档中所有嵌入式对象（embed元素）组成的一个数组。数组中的条目会根据源代码顺序从0开始建立索引。

示例
```
var embedCount = document.embeds.length;
```
值 嵌入式对象引用数组。
默认值 长度为0的数组。

expando

IE 4 NN n/a Moz n/a Saf n/a Op n/a DOM n/a

可读/可写

指明当前文档中的脚本是否允许创建并使用document对象的自定义属性。只须要为document对象指定一个值，JavaScript的扩展特性就允许脚本创建一个新的对象属性，例如document.stooge = "Curly"。这也意味着文档会接受拼写错误的属性赋值，例如，如果未将一个长属性名称的中间字母设置为大写也会对属性进行设置。尽管浏览器会毫无疑问地接受这类赋值，但这样肯定无法达到预期的设置效果。如果并不希望创建自定义属性，那么在编辑页面时可以考虑在一段开放的脚本指令中将document.expando设置为false，这样就可以避免由拼写错误而带来的隐患。这个设置只会影响当前文档内的脚本代码。

示例
```
document.expando = false;
```
值 布尔值：true | false。
默认值 true

fgColor

IE 3 NN 2 Moz all Saf all Op 7 DOM n/a

可读/可写

为文档提供提供前景（文本）色。在Navigator 6之前的版本中，修改这个属性无法自动地改变文本颜色。当前的各种浏览器依然支持这一属性，而且可以通过document.body.text属性或CSS的color属性设定来更好地对它进行控制。

示例
```
document.fgColor = "darkred";
```
值 一个十六进制的三元组或明语颜色名称。目前可用的明语颜色名称请参见附录A。
默认值 由浏览器决定默认值，通常为“black”。

fileCreatedDate

IE 4(Win) NN n/a Moz n/a Saf n/a Op n/a DOM n/a

只读

返回一个日期（而不是时间）字符串，服务器（或本地文件系统）以此表示当前加载文件的创建日期。从IE 5开始，日期信息的格式是“mm/dd/yyyy”。一旦服务器提供的日期不符合IE的要求，那么这个值也可能会出错。

示例
```
var dateObj = new Date(document.fileCreatedDate);
```
值 日期字符串。
默认值 无。

fileModifiedDate

IE 4(Win) NN n/a Moz n/a Saf n/a Op n/a DOM n/a

只读

返回一个日期（而不是时间）字符串，服务器（或本地文件系统）以此表示当前加载的文件的最近一次修改

日期。从IE 5开始，日期信息的格式是“mm/dd/yyyy”。一旦服务器提供的日期格式不符合IE的要求，那么这个值可能会出错。

示例 `var dateObj = new Date(document.fileModifiedDate);`

值 日期字符串。

默认值 无。

fileSize

IE *4* NN *n/a* Moz *n/a* Saf *n/a* Op *n/a* DOM *n/a*

只读

返回当前已加载文档大小的字节数。IE/Windows会返回一个字符串值，而IE/Macintosh则会返回一个数字，如果要对这个值进行数学计算，那么这两者的区别是巨大的。

示例 `var byteCount = parseInt(document.fileSize, 10);`

值 字符串形式的整数（Windows）或数字（Mac）。

默认值 无。

fileUpdatedDate

IE *5.5* NN *n/a* Moz *n/a* Saf *n/a* Op *n/a* DOM *n/a*

只读

返回一个空字符串。官方显然还未支持这个属性。

forms[ ]

IE *3* NN *2* Moz *all* Saf *all* Op *7* DOM *1*

只读

返回由当前文档中所有`form`对象（`form`元素）组成的一个数组。数组中的各项均以它们在源代码中的顺序进行排序（从零开始计数），但是也可以使用表单的名称作为一个字符串索引值来访问数组中的对象。

示例 `var elemCount = document.forms[0].elements.length;`

值 `form`对象数组。

默认值 长度为0的数组。

frames[ ]

IE *4* NN *n/a* Moz *n/a* Saf *n/a* Op *n/a* DOM *n/a*

只读

返回由当前文档中所有`iframe`对象（`iframe`元素）组成的一个数组。须要注意的是，不要将`iframe`与类似于窗口的`frame`对象混为一谈。数组中的条目会根据源代码顺序从0开始建立索引。为了在现代浏览器中实现跨平台的兼容性，请使用`document.getElementsByTagName("iframe")`方法。

示例 `var iframeCount = document.frames.length;`

值 `iframe`对象数组。

默认值 长度为0的数组。

height, width

IE *n/a* NN *4* Moz *all* Saf *all* Op *n/a* DOM *n/a*

只读

返回整个已显示的文档的像素尺寸。这两个值与`document.body.parentNode`（例如，`html`元素）对象的`offsetHeight`和`offsetWidth`属性值一致。然而，以上所述的两种属性对均未被W3C DOM批准，但在实际应用中最好还是使用“offset”类的属性，它们至少还是跨平台兼容的属性。

示例 `var howTall = document.height;`
值 像素值。
默认值 当前文档的尺寸。

ids[ ]

IE *n/a* NN *|4|* Moz *n/a* Saf *n/a* Op *n/a* DOM *n/a*

只读

这是一种只能在Navigator 4中使用的样式单JavaScript语法，对单个ID而言，`ids[]`集合是其引用的一部分，而通过`[document.]ids.idName.stylePropertyName`这种语法格式可以为它指定样式属性。如果要获取相关属性列表，请参见本章中列出的`tags`对象。

images[ ]

IE *4* NN *3* Moz *all* Saf *all* Op *7* DOM *1*

只读

返回当前文档节点树内所有`img`元素对象（通过`new Image()`构造方法加载的预缓存图像除外）组成的一个数组。数组中的各项均以它们在源代码中的顺序进行排序（从零开始计数），并且可以通过索引数字或字符串名称来进行访问。一旦出现这个属性，就表明该浏览器支持活动图像和可交换图像。

示例 `document.images["home"].src = "images/homeHilite.jpg";`
值 `img`元素对像数组。
默认值 长度为0的数组。

implementation

IE *5(Mac)/6(Win)* NN *n/a* Moz *all* Saf *all* Op *7* DOM *1*

只读

返回W3C的`DOMImplementation`对象的一个引用，在一定限度内，它表示组成文档容器的外界环境，对我们而言，这个环境就是浏览器。这个对象的方法能够给出浏览器声明支持的DOM模型。通过它还可以在当前文档树之外创建虚拟的W3C `Document`和`DocumentType`对象。因此，在Mozilla中可以将`document.implementation`作为一个出发点，以便为外部XML文档生成一个非显示的文档。请参见`DOMImplementation`对象，以便了解其对象方法及浏览器对它的相关支持等详细信息。

示例 `var xDoc = document.implementation.createDocument("", "theXdoc", null);`
值 `DOMImplementation`对象的引用。
默认值 当前`DOMImplementation`对象。

inputEncoding

IE *n/a* NN *n/a* Moz *1.8* Saf *n/a* Op *n/a* DOM *3*

只读

返回文档中指定的字符编码类型。此值会受到文档内编码规范的影响，例如一个指定了字符集的`meta`元素。

示例

```
if (document.inputEncoding == "ISO-8859-5") {
    // 处理 ISO-8859-5 字符编码
}
```

值 字符串形式的字符编码类型，如果是在当前文档节点树之外创建的文档，那么返回值为`null`。
默认值 `UTF-8`

lastModified

IE *3* NN *2* Moz *all* Saf *all* Op *7* DOM *n/a*

只读

提供最后一次修改文档文件的日期和时间字符串。某些服务器无法全部或正确地支持这一信息。只有在最新

的浏览器中才能将字符串的日期形式作为Date对象构造方法的一个参数。

示例 `document.write(document.lastModified);`

值 日期和时间的字符串表示形式。

默认值 无。

layers[ ]

IE *n/a* NN *|4|* Moz *n/a* Saf *n/a* Op *n/a* DOM *n/a*

只读

返回Navigator 4中当前文档内所有layer元素对象组成的一个数组。由样式单控制着相对或绝对位置的其他HTML元素引用也包含在这个数组中（在这种情况下，Navigator 4将这些元素视作layer对象）。数组中的各项均以它们在源代码中的顺序进行排序（从零开始计数），并且可以通过索引数字或字符串名称来进行访问。作为只能应用于Navigator 4的一个功能，一旦出现这个属性就表明对Netscape层提供了唯一性引用需求的支持。

示例

```
if (document.layers) {
    // 使用 document.layers[]语法进行引用
}
```

值 layer对象数组或其等价物。

默认值 长度为0的数组。

linkColor

IE *3* NN *2* Moz *all* Saf *all* Op *7* DOM *n/a*

可读/可写

此元素设定超文本链接尚未被访问时的颜色，此时链接的URL地址尚不在浏览器的缓存之中。链接存在三种状态：未访问（unvisited）、激活（active）和已访问（visited）。设置这个属性后，链接文字、图像的边框，以及嵌入a元素之中的对象均会应用这个颜色。但在Navigator 4或之前的版本中，修改这个属性并不会动态改变链接的颜色。在现代浏览器中，应该使用W3C DOM中的替代物document.body.link，当然，如果能使用CSS伪类:link则更好。

示例 `document.link Color= "#00FF00";`

值 一个十六进制的三元组或明语颜色名称。目前可用的明语颜色名称请参见附录A。

默认值 `#0000FF`

links[ ]

IE *3* NN *2* Moz *all* Saf *all* Op *all* DOM *1*

只读

返回由所有area元素对象和指定了href属性的a元素对象（此时它们为可点击的链接）组成的数组。数组中的各项均以它们在源代码中的顺序进行排序。

示例

```
for (var i = 0; i < document.links.length; i++) {
    // 遍历 document.links[]数组
}
```

值 area和a（作为链接使用）元素对象数组。

默认值 长度为0的数组。

location

IE *3* NN *2* Moz *all* Saf *all* Op *7* DOM *n/a*

可读/可写

指出当前文档的URL路径。虽然目前浏览器支持这个属性，但由于它会和window.location发生混淆，因此

不建议使用这个属性。在实际应用中，请使用实现更为广泛的document.URL属性，当然，使用window.location.href更佳。

示例 document.location = "products/widget33.html";
值 字符串形式的完整或相对URL。
默认值 文档URL。

media

IE 5.5 NN n/a Moz n/a Saf n/a Op n/a DOM n/a

可读/可写

返回一个字符串，文档内容按照字符串所指明的输出媒体进行格式化。但直到IE 6为止，这个属性仍然只返回一个空字符串，而且一旦为其设置了一个可接受的字符串值（all、print或screen），甚至还会抛出一个安全性错误。因此请避免在document对象上使用这个属性。

mimeType

IE 5.5 NN n/a Moz n/a Saf n/a Op n/a DOM n/a

只读

返回一个指明基本文档类型的字符串，但字符串中的内容并不遵循MIME格式。从IE 5.5开始，HTML文档所返回的字符串值变为“HTML Document”。不要将这个document对象属性与Netscape、Mozilla和IE/Mac中的navigator.mimeTypes属性混为一谈，后者是完全不同的一种事物。

示例 var what = document.mimeType;
值 字符串。
默认值 HTML Document

nameProp

IE 6 NN n/a Moz n/a Saf n/a Op n/a DOM n/a

只读

返回一个字符串，它包含的数据与document.title的内容相同，如果在文档内并不存在title元素，那么就会返回一个空字符串。而IE可能并未对document对象提供有关此属性的官方支持。

值 字符串。
默认值 空字符串。

namespaces[ ]

IE 5.5 NN n/a Moz n/a Saf n/a Op n/a DOM n/a

只读

返回当前文档内已实现的namespace对象所组成的一个集合。namespace对象是加载外部行为的一个入口点。如须获取更多详细信息，请访问如下链接：*http://msdn.microsoft.com/ workshop/author/behaviors/overview/elementb_ovw.asp*。

示例 var IENSCount = document.namespaces.length;
值 namespace对象引用数组。
默认值 长度为0的数组。

parentWindow

IE 4 NN n/a Moz n/a Saf n/a Op 7 DOM n/a

只读

返回包含当前文档的window对象的引用，这个window对象可能是框架集中的一个框架。由于这个属性的返回值与文档中window对象的引用完全相同，因此可以使用这个引用直接访问窗口的属性和方法。

示例 `var siblingCount = document.parentWindow.frames.length;`
值 `window`或`frame`对象引用。
默认值 `window`对象。

plugins[ ]

IE *4* NN *4* Moz *all* Saf *n/a* Op *7* DOM *n/a*

只读

返回当前文档中所有嵌入式对象（`embed`元素）组成的一个数组。数组中的条目会根据源代码顺序从0开始建立索引。不要将这个集合与`navigator.plugins`集合混为一谈。

示例 `var embedCount = document.plugins.length;`
值 `embed`对象引用数组。
默认值 长度为0的数组。

preferredStylesheetSet

IE *n/a* NN *n/a* Moz *1.7.5* Saf *1.3/2* Op *n/a* DOM *n/a*

只读

根据HTML 4规范中`link`元素的定义，返回导入的样式单的`title`属性值，文档作者会将该样式单标示为其首选样式单。

示例

```
if (document.preferredStylesheetSet == "main") {
    // 对'main'样式单规则进行处理
}
```

值 字符串。
默认值 空字符串。

protocol

IE *4* NN *n/a* Moz *n/a* Saf *n/a* Op *n/a* DOM *n/a*

只读

返回一个明语字符串，以描述加载当前文档时所使用的协议。与`location.protocol`属性的字母描述（如，“`http:`”或“`file:`”）不同，`document.protocol`则提供了可供人阅读的相关信息，例如，“`Hypertext Transfer Protocol`”或“`File Protocol`”。

示例

```
if (document.protocol == "File Protocol") {
    // 在 IE 中进行文件访问操作
}
```

值 明语字符串。
默认值 当前文档的协议类型。

readyState

IE *4* NN *n/a* Moz *n/a* Saf *all* Op *7* DOM *n/a*

只读

此属性将返回文档内容的当前下载状态。在加载文档的过程中，如果一段脚本（特别是由用户事件启动的脚本）能够执行某些动作，但在整个页面加载完成之前又必须避免其他动作，那么使用这个属性就可以获得一些与加载进程相关的中间信息。因此，可以在状态测试中使用其属性值。这个属性的属性值会随着加载状态的改变而发生变化。而每当属性值发生变化时均会激发一个`readystatechange`事件。

示例

```
if (document.readyState == "loading") {
    // 脚本代码
}
```

值 下列字符串之一：complete | interactive | loading | uninitialized。有些元素可能会允许用户与部分元素内容进行交互，在这种情况下，此属性可能会返回interactive直到所有加载全部完成。

默认值 无。

referrer

IE 3 NN 2 Moz all Saf all Op 7 DOM 1

只读

返回一个页面的URL字符串，假设原始页面拥有一个指向当前页面的链接，则当前页面就来自于该URL指定的页面。很多服务器日志也会记录这个信息。通过脚本可以查看访问者是否从特定的初始页面链接至当前文档，并且也可以根据不同的初始页面对当前内容进行细微的调整。如果访问者通过其他方式到达当前页面，例如在浏览器中输入文档的URL或选择一个书签，那么referrer属性会返回一个空字符串。Windows下的很多IE版本不仅无法报告正确的引用URL，而且还会错误地将当前页面的URL作为引用URL。

示例

```
if (document.referrer) {
   document.write("<p>Thanks for following the link to our web site.</p>");
}
```

值 字符串。

默认值 无。

scripts[ ]

IE 4 NN n/a Moz n/a Saf all Op 7 DOM n/a

只读

返回当前文档中所有脚本对象（script元素）组成的一个数组。每个script对象可能包含任意数量的方法。scripts[]集合计算了文档中实际<script>标签的数量。而数组中的条目则会根据源代码顺序从0开始建立索引。

示例 `var scriptCount = document.scripts.length;`

值 script元素引用组成的数组。

默认值 长度为0的数组。

security

IE 5.5 NN n/a Moz n/a Saf n/a Op n/a DOM n/a

只读

返回一个描述当前文档中已生效的安全策略的字符串。

示例 `var secPolicy = document.security;`

值 字符串。

默认值 “这种类型的文档没有安全证书。”

selectedStylesheetSet

IE n/a NN n/a Moz n/a Saf 1.3/2 Op n/a DOM n/a

可读/可写

返回已导入的样式单的title属性值，文档作者将该样式单标示为其首选样式单并且已选择该样式。它似乎是一个实验性的属性，请参见*http://hixie.ch/specs/css/dom/altss/altss-1.0-pre7*。

值 字符串。

默认值 空字符串。

selection

IE 4 NN n/a Moz n/a Saf n/a Op 8 DOM n/a

只读

返回一个selection对象。为了利用由用户或脚本所选中的文本，必须将这些内容转换成一个TextRange对象。而通过window.getSelection()方法则可以访问Mozilla或Safari中选中的文本内容。

示例 `var range = document.selection.createRange();`

值 对象引用。

默认值 无。

strictErrorChecking

IE n/a NN n/a Moz 1.8 Saf n/a Op n/a DOM 3

可读/可写

布尔值，它控制浏览器是否为DOM操作抛出异常。

示例 `document.strictErrorChecking = false;`

值 布尔值：true | false。

默认值 true

styleSheets[ ]

IE 4 NN n/a Moz all Saf all Op 7 DOM 2

只读

返回当前文档中所有styleSheet对象组成的一个数组。每个样式单对象可能包含任意数量的样式单规则。styleSheets[]集合计算了文档中实际<style>标签的数量，以及那些加载了外部样式单文件的<link>标签的数量。数组中的条目会根据源代码顺序从0开始建立索引。通过styleSheet对象的cssRule.styleSheet属性也可以访问@import样式单对象。请参见styleSheet对象。

示例

```
for (var i = 0; i < document.styleSheets.length; i++) {
   // 遍历 styleSheet 对象
}
```

值 styleSheet对象引用数组。

默认值 长度为0的数组。

tags[ ]

IE n/a NN 4 Moz n/a Saf n/a Op n/a DOM n/a

只读

这是一种只能在Navigator 4中使用的样式单JavaScript语法，对单个标签类型及指定给它的样式属性而言，tags[]集合是其引用的一部分。如果要获取属性列表，请参见本章中列出的tags对象。此外，不要将Navigator中tags[]集合的用法与IE中tags[]集合的用法相混淆，后者实际上属于all集合。

示例 `document.tags.H1.color= "red";`

值 由Navigator 4 JavaScript 样式单tag对象引用组成的数组。

默认值 长度为0的数组。

title

IE 3 NN 2 Moz all Saf all Op 7 DOM 1

可读/可写

与其他能够反映HTML元素的对象title属性不同，document.title属性对应于文档head部分中定义的title元素的内容。这个标题内容会显示在浏览器的标题栏中，以帮助访问者识别文档。如果要将页面加入

书签列表，那么其标题内容也会进入书签。尽管此属性是可读、可写的，但当某个浏览器版本出于安全考虑而无法改变窗口的标题栏时，也不要因此而感到吃惊。

示例 `document.title = "Fred\'s Home Page";`

值 字符串。

默认值 无。

URL

IE *4* NN *3* Moz *all* Saf *all* Op *7* DOM *1*

可读/可写

提供当前文档的URL路径。此属性值与`location.href`的值相同。`document.URL`属性已经逐渐发展成为`document.location`属性的替代物，这样就可以避免混淆`location`对象和`document.location`属性（脚本编写者和JavaScript解释引擎均可能发生这种混淆错误）。

示例 `document.URL = "http://www.megacorp.com";`

值 字符串形式的完整或相对URL。

默认值 当前文档的URL。

URLUnencoded

IE *5.5* NN *n/a* Moz *n/a* Saf *n/a* Op *n/a* DOM *n/a*

只读

返回当前文档的URL，但会将所有的URL编码字符转换成它们的明语版本，例如“`%20`”会转化成一个空格字符。如果将JavaScript的`decodeURI()`方法应用于`document.URL`，会发现其处理结果与本属性的返回值相同。

示例 `var straightPath = document.URLUnencoded;`

值 字符串形式的完整或相对URL。

默认值 当前文档的URL。

vlinkColor

IE *3* NN *2* Moz *all* Saf *all* Op *7* DOM *n/a*

可读/可写

指明已被访问的超文本链接的颜色。设置这个属性后，链接文字、图像的边框，以及嵌入`a`元素之中的对象均会应用这个颜色。请参看`alinkColor`和`linkColor`属性以了解已点击和未访问链接的颜色。在Navigator 4或之前的版本中，修改这个属性并不会动态改变链接的颜色。在现代浏览器中，应该使用W3C DOM中的替代物`document.body.vlink`，当然，使用CSS伪类`:visited`是一种更好的选择。

示例 `document.vlinkColor = "gold";`

值 一个十六进制的三元组或明语颜色名称。目前可用的明语颜色名称请参见附录A。

默认值 由浏览器和操作系统共同决定。

width

参见height属性。

xmlEncoding, xmlStandalone, xmlVersion

IE *n/a* NN *n/a* Moz *1.8* Saf *n/a* Op *n/a* DOM *3*

见下文

这3个属性反映了一个XML文档内XML声明的相关信息。例如，有如下一个声明：

```
<?xml version="1.0" standalone="yes"?>
```

xmlVersion属性会报告1.0，而xmlStandalone则会返回true。由于这段代码中并未声明字符编码，因此xmlEncoding属性将返回一个空字符串。须要注意的是，xmlVersion是只读属性而另外两个均可读可写。

尽管Mozilla 1.8和1.8.1支持这些属性，但它们均未得以完全实现，而且还可能返回错误的值。

值　字符串（xmlVersion和xmlEncoding），布尔值（xmlStandalone）。

默认值　分别为null、false和null，而在Mozilla 1.8和1.8.1中则分别是空字符串、false和空字符串。

addBinding( ), getAnonymousElementByAttribute( ), getAnonymousNodes( ), getBindingParent( ), loadBindingDocument( ), removeBinding( )

IE *n/a* NN *n/a* Moz *all* Saf *n/a* Op *n/a* DOM *n/a*

在Mozilla中，这一系列的document对象方法是可扩展绑定语言（Extensible Binding Language，简称为XBL）的浏览器编程功能部件的一部分，该语言附属于一种基于XML的机制，浏览器可以使用这种机制生成用户界面皮肤。如果要获悉更多关于XBL的信息，请访问此链接：*https://developer.mozilla.org/en/XBL/XBL_1.0_Reference*。

adoptNode( )

IE *n/a* NN *n/a* Moz *1.7.8* Saf *n/a* Op *7* DOM *3*

```
adoptNode(nodeReference)
```

从另一个文档中删除一个节点并且将它带至当前文档的环境中。然后就可以使用当前文档的其他方法将它插入文档树中的指定位置。

返回值　指向被吸收的节点的引用。

参数

nodeReference

一个节点（通常在另一个文档中）的引用，此方法会将该节点吸收至当前文档。

captureEvents( )

IE *n/a* NN *4* Moz *all* Saf *all* Op *8* DOM *n/a*

```
captureEvents(eventTypeList)
```

命令浏览器拦截一个特定类型的事件，以防止它们到达其预定目标对象。而调用这个方法的对象必须拥有针对该事件类型的事件处理程序，以便处理捕获的事件。尽管这个方法最初只是Navigator 4事件模型的一部分，但Mozilla、Safari和Opera 8及其后续浏览器陆续为它提供了相关支持，它实际上已经为document对象创建了一种与W3C DOM捕获模式事件监听器相同的等价物。尽管这个方法可能比较简便，但还是应该按照W3C DOM事件监听器语法编写新代码，请参照在线参考VI。

返回值　无。

参数

eventTypeList

一个使用逗号分隔的事件类型列表，对应的事件类型来自于可用的静态Event对象常量，如Event.CLICK或Event.MOUSEMOVE，注意区分字符大小写。

clear( )

IE *3* NN *2* Moz *all* Saf *all* Op *7* DOM *n/a*

从窗口或框架内移除当前文档，通常在为写入新内容而开启一个新的数据流时使用这个方法。document.write()和document.writeln()方法会自动调用这个方法。早期的浏览器曾经深受document.clear()方法中众多bug的折磨。甚至直到今天，最好还是让文档自动地编写方法以处理这项任务。W3C DOM已明确地省略了这个方法。

返回值 无。
参数 无。

close( )

IE 3 NN 2 Moz all Saf all Op 7 DOM 1

关闭文档向窗口或框架的写入流。如果脚本使用document.write()或document.writeln()来为一个窗口或框架生成所有的新内容（这会使得新写入的内容取代整个当前文档），那边必须附加一个document.close()方法以保证全部内容均被写入文档。如果忽略这个方法，就可能会造成部分内容无法被正常写入。另外，这个方法也能为下一个文档写入方法写入崭新的内容准备好窗口或框架。但从服务器端加载内容时，如果使用文档写入方法动态地将内容写入页面，那么就不要使用document.close()。

返回值 无。
参数 无。

createAttribute( )

IE 5(Mac)/6(Win) NN n/a Moz all Saf all Op 7 DOM 1

```
createAttribute("attributeName ")
```

在内存中生成一个属性节点（Attr对象）的实例。一个典型的流程是，首先创建属性，然后通过其nodeValue属性指定一个值，最后使用元素的setAttributeNode()方法将Attr节点插入该元素的属性列表。

返回值 Attr节点对象引用。
参数

attributeName

属性名称字符串，需要区分大小写。

createAttributeNS( )

IE n/a NN n/a Moz all Saf all Op 8 DOM 2

```
createAttributeNS("namespaceURI ", "qualifiedName")
```

在内存中生成一个属性节点（Attr对象）的实例，该对象的名称定义在一个外部命名空间之中。一个典型的流程是，首先创建属性，然后通过其nodeValue属性指定一个值，最后通过元素的setAttributeNodeNS()方法将Attr节点插入该元素的属性列表。

返回值 Attr节点对象引用。
参数

namespaceURI

URI字符串，它与文档中指定给某个标注的URI相对应，而该属性最终被加入这个文档。

qualifiedName

属性的完整名称，包括本地名称前缀、一个冒号，以及本地名称。

createCDATASection( )

IE 5(Mac) NN n/a Moz all Saf all Op 8 DOM 1

```
createCDATASection("data")
```

在内存中生成XML（包括XHTML）文档内某个字符数据分块（character data section）节点（CDATASection对象）的实例。

返回值 CDATASection节点对象引用。
参数

data

字符串数据，其中包含分块中的文本内容。

createComment( )

IE 5(Mac)/6(Win) NN n/a Moz all Saf all Op 7 DOM 1

```
createComment("commentText")
```

499 在内存中生成一个注释节点（comment对象，且nodeValue为8）的实例。一个典型流程是，首先创建Comment节点，然后通过节点的appendChild()或insertBefore()方法将它插入文档树中的预期位置。不过，仅有IE 5/Mac部分实现了这个方法。

返回值 Comment节点对象引用。

参数

commentText

包含注释数据的字符串。

createDocumentFragment( )

IE 5(Mac)/6(Win) NN n/a Moz all Saf all Op 7 DOM 1

在内存中生成一个空的文档片段节点（DocumentFragment对象）的实例。那些最终将附加或插入到节点树的一系列节点将全部集合到这个DocumentFragment节点之中。请参见DocumentFragment对象以获取更多详细信息。

返回值 DocumentFragment节点对象引用。

参数 无。

createElement( )

IE 4 NN n/a Moz all Saf all Op 7 DOM 1

```
createElement("tagName")
```

在内存中生成一个元素对象的实例，而这个元素对象与方法参数传入的标签关联在一起。在IE 4中，这个方法被限制在area、img和option元素，但在其他支持这一方法的浏览器中，则允许所有的元素使用这一方法。在一个新创建的元素中，其属性并未被指定具体的属性值（除非根据DTD指定默认的属性值），而这个元素也不是文档树的一部分。然后须要为它分配属性（例如，为input元素指定type类型，或者为任意元素指定id属性），并将它附加或插入当前文档树中。这一顺序正是W3C DOM创建新内容的常用步骤（它代替了许多现代浏览器中实现的innerHTML属性）。

返回值 元素对象的引用。

参数

tagName

新元素的标签名字符串：document.createElement("option")。IE还允许使用一个完整的开始标签字符串，连同其尖括号和名称/值对。但W3C DOM规范只支持纯粹的标签名形式。

createElementNS( )

IE n/a NN n/a Moz all Saf all Op 8 DOM 2

```
createElementNS("namespaceURI ", "qualifiedName ")
```

在内存中生成一个元素对象的实例，而这个元素对象与方法参数传入的命名空间、标注和标签关联在一起。在一个新创建的元素中，其属性并未被指定具体的属性值（除非根据DTD指定默认的属性值），而这个元素也不是文档树的一部分。然后须要为它分配属性（例如，为input元素指定type类型，或者为任意元素指定id属性），并将它附加或插入当前文档树中。

500

返回值 元素对象的引用。

参数

namespaceURI

URI字符串，它与文档中指定给某个标注的URI相对应，而该属性最终被加入这个文档。

qualifiedName
属性的完整名称，包括本地名称前缀、一个冒号，以及本地名称。

createEntityReference( )

IE *5(Mac)* NN *n/a* Moz *all* Saf *all* Op *n/a* DOM *1*

```
createEntityReference("entityName")
```

在内存中生成一个XML文档实体引用节点对象的实例。不过，目前仅有IE 5/Mac部分支持这个方法。

返回值 实体引用节点对象的引用。

参数

entityName
字符串。

createEvent( )

IE *n/a* NN *n/a* Moz *all* Saf *all* Op *7* DOM *2*

```
createEvent("eventType ")
```

在内存中生成一个W3C DOM的Event对象的实例，该对象属于某个特定的事件类别。在创建普通事件后，必须通过初始化方法将它初始化为一个特定的事件类型，并初始化那些属于该事件类别的对应属性。下面这段代码创建了一个mousedown事件并将它发送至一个元素：

```
var evt = document.createEvent("MouseEvents");
evt.initEvent("mousedown", true, true);
document.getElementById("myElement").dispatchEvent(evt);
```

然后这个事件可能被传递给另一个元素，这样一来，就像用户点击鼠标按钮而生成的事件一样，该元素的事件监听器就能够处理这个事件。

返回值 Event对象引用。

参数

eventType
在所有支持这一方法的浏览器中，可以使用以下任意一个对应于特定事件类型的字符串常量：HTMLEvents、KeyEvents（W3C DOM Level 2并未指定）、MouseEvents、MutationEvents或UIEvents。DOM Level 3根据所有的新事件类型（从Mozilla 1.7.5和Opera 8开始支持）重新调整了这个列表，它包括：Event（早期的HTMLEvents）、KeyboardEvent、MouseEvent、MutationEvent、MutationNameEvent、TextEvent或UIEvent。

createEventObject( )

IE *5.5* NN *n/a* Moz *n/a* Saf *n/a* Op *n/a* DOM *n/a*

```
createEventObject([existingEventObject])
```

在内存中生成一个空的IE DOM event节点对象的实例。创建普通事件之后，可以使用合适的值填充其属性以辅助事件处理。然后就可以将这个事件作为元素fireEvent()方法的输入参数，从这一点上来看，事件类型是和事件关联在一起的。下面这段代码创建了一个mousedown事件并将它发送至一个元素：

```
var evt = document.createEventObject();
document.getElementById("myElement").fireEvent("onmousedown", evt);
```

当然，使用脚本生成事件时，也可以将一个已存在的事件对象作为其模板。这时可以将当前event对象作为参数传入createEventObject()方法，然后根据需要修改新对象的属性。

返回值 event对象引用。

参数

existingEventObject
一个event对象的引用，该对象既可以由用户激发也可以由脚本生成。新事件会拥有已存在事件对象的所有属性。

createNodeIterator( )

IE *n/a* NN *n/a* Moz *n/a* Saf *1.3/2* Op *8* DOM *2*

```
createNodeIterator(rootNode, whatToShow, filterFunction, entityRefExpansion)
```

在内存中生成一个NodeIterator对象的实例。这个方法的输入参数与createTreeWalker()方法相同。

返回值 NodeIterator对象引用。

参数

rootNode

文档树中某个节点的引用，该节点将成为NodeIterator对象的节点列表中的第一个节点。

whatToShow

整数值，对应于某个内置过滤器，这种过滤器可以使得方法只返回NodeIterator对象中包含的某一种类型的节点。对于这些整数值，NodeFilter对象包含一些等价的明语常量，如：NodeFilter.SHOW_ALL、NodeFilter.SHOW_ATTRIBUTE、NodeFilter.SHOW_CDATA_SECTION、NodeFilter.SHOW_COMMENT、NodeFilter.SHOW_DOCUMENT、NodeFilter.SHOW_DOCUMENT_FRAGMENT、NodeFilter.SHOW_DOCUMENT_TYPE、NodeFilter.SHOW_ELEMENT、NodeFilter.SHOW_ENTITY、NodeFilter.SHOW_ENTITY_REFERENCE、NodeFilter.SHOW_NOTATION、NodeFilter.SHOW_PROCESSING_INSTRUCTION、NodeFilter.SHOW_TEXT。

filterFunction

一个用户方法的引用，它可以进一步过滤NodeIterator对象中包含的节点。这个方法只有一个参数，即一个用于检测的节点引用，NodeIterator对象会自动对其进行调用。此方法的返回值决定了这个被检测节点是否会包含在节点列表中。返回值均为整形，但NodeFilter对象提供了与之等价的3个明语常量：NodeFilter.FILTER_ACCEPT、NodeFilter.FILTER_REJECT、NodeFilter.FILTER_SKIP。

由于NodeIterator对象不会将这个节点列表保存在对象内，因此NodeFilter.FILTER_REJECT和NodeFilter.FILTER_SKIP会跳过某个节点而不会对子节点产生任何影响。如须获得此类方法的范例，请参见treeWalker对象。

entityRefExpansion

布尔值，它控制着是否应该将实体引用节点（常用于XML文档中）的内容视为层次节点，true表示是，false则表示否。

createProcessingInstruction( )

IE *5(Mac)* NN *n/a* Moz *all* Saf *all* Op *8* DOM *1*

```
createProcessingInstruction("target", " data")
```

在内存中为XML文档中的处理指令节点对象生成一个实例。但目前仅有IE 5/Mac能部分支持这个方法。

返回值 处理指令节点对象的引用。

参数

target

字符串。

data

字符串。

createRange( )

IE *n/a* NN *n/a* Moz *all* Saf *1.2* Op *n/a* DOM *2*

创建一个空的Range对象，在出现第1个主体文本字符之前，该对象将缩小为一个点。此方法将返回该Range对象的引用，然后通过这个引用就可以调整它的边界点或调用其方法等。请参见Range对象以获取关于其语言特性的更多细节内容。

返回值 W3C DOM的Range对象引用。

参数 无。

createStyleSheet( )

IE 4 NN n/a Moz n/a Saf n/a Op n/a DOM n/a


```
createStyleSheet(["url"[, index]])
```

在Windows和Macintosh平台下的IE浏览器中，虽然这个方法会执行同样的操作，但在这两个平台的返回值有所不同。此外，它在文档树中的特定操作还依赖于方法中传入的参数。如果并未传入参数，那么此方法会将一个空style元素插入文档树。但在这种情况下，在为这个空style元素对象增加一个或多个样式规则之前，它并不会反映在document.styleSheets集合中。但如果将一个指向外部.*css*文件的URL作为第一个参数，那么此方法会创建一个link元素，并将它插入到文档的head部分，从而立即激活这个外部样式规则。

IE/Windows总是返回一个styleSheet对象的引用，而IE/Macintosh则会返回一个指向新插入元素的引用，根据参数的组成形式，这个元素可能是一个style元素也可能是link元素。由于此时并不能引用相关的styleSheet对象,因此这个已插入的style元素的引用对添加规则毫无帮助。为实现跨操作系统的兼容性，最好只针对外部样式单使用这个方法。

返回值 styleSheet对象引用（Windows），style或link元素对象引用（IE 5及后续版本/Macintosh）。

参数

url

一个外部.*css*样式单定义文件的URL字符串。

index

一个可选的从零开始计数的整数，它指出新的样式单应该插入到styleSheets[]集合中的哪个位置。默认操作是将它附加到集合末尾，但这可能会影响文档的层叠样式规则。详情请参见在线参考III。

createTextNode( )

IE 5 NN n/a Moz all Saf all Op 7 DOM 1

```
createTextNode("text")
```

在内存中生成一个文本节点（W3C DOM的Text对象）的实例，其nodeValue由参数传入的无标签的文本内容所组成。这个新创建的文本节点还不是文档树的一部分。此方法后，须调用要将这个节点附加或插入到文档树中，也可以将它放入文档片段中以便以后进行文档插入工作。这一操作顺序正是W3C DOM创建新内容的常用步骤（它代替了某些浏览器中实现的innerText属性）。

返回值 文本节点对象引用。

参数

text

字符串，在被插入文档树之后，会显示该字符串的内容。

createTreeWalker( )

IE n/a NN n/a Moz 1.0.1 Saf 1.3/2 Op 8 DOM 2


```
createTreeWalker(rootNode, whatToShow, filterFunction, entityRefExpansion)
```

在内存中生成一个treeWalker对象的实例。

返回值 treeWalker对象引用。

参数

rootNode

文档树中某个节点的引用，该节点将成为treeWalker对象的节点列表中的第一个节点。

whatToShow

整数值，对应于某个内置过滤器，这种过滤器可以使得方法只返回treeWalker对象中包含的某一种类型的节点。对于这些整数值，NodeFilter对象包含一些等价的明语常量，如：NodeFilter.SHOW_ALL、

NodeFilter.SHOW_ATTRIBUTE、NodeFilter.SHOW_CDATA_SECTION、NodeFilter.SHOW_COMMENT、NodeFilter.SHOW_DOCUMENT、NodeFilter.SHOW_DOCUMENT_FRAGMENT、NodeFilter.SHOW_DOCUMENT_TYPE、NodeFilter.SHOW_ELEMENT、NodeFilter.SHOW_ENTITY、NodeFilter.SHOW_ENTITY_REFERENCE、NodeFilter.SHOW_NOTATION、NodeFilter.SHOW_PROCESSING_INSTRUCTION、NodeFilter.SHOW_TEXT。

filterFunction

一个用户方法的引用，它可以进一步过滤treeWalker对象中包含的节点。这个方法只有一个参数，即一个用于检测的节点引用，treeWalker对象会自动对其进行调用。此方法的返回值决定了这个被检测节点是否会包含在节点列表中。返回值均为整形，但NodeFilter对象提供了与之等价的三个明语常量：NodeFilter.FILTER_ACCEPT、NodeFilter.FILTER_REJECT、NodeFilter.FILTER_SKIP。

如果返回值为NodeFilter.FILTER_SKIP，那么已跳过的节点的子孙节点仍然可能作为TreeWalker节点列表的一员（假设它们未被其他过滤器过滤）。而当返回值为NodeFilter.FILTER_REJECT时，检测节点及其子孙节点均会被从treeWalker对象的节点列表中移除。如须获得此类方法的范例，请参见treeWalker对象。

entityRefExpansion

布尔值，它控制着是否应该将实体引用节点（常用于XML文档中）的内容视为层次节点，true表示是，false则表示否。

505

elementFromPoint( )

IE *4* NN *n/a* Moz *all* Saf *2.01* Op *7* DOM *n/a*

```
elementFromPoint(x, y)
```

返回一个对象引用，该对象就位于参数所指定的像素坐标之下，其中x表示横坐标，y表示纵坐标。对于一个能够被识别的元素而言，它必须能响应鼠标事件。另外，如果在同一个位置存在多个元素，那么将被返回拥有最大zIndex值的元素，或zIndex值相等，但在源代码顺序中位于最后的元素。

返回值 元素对象的引用。

参数

x

到窗口或框架左边缘的水平距离的像素值。

y

到窗口或框架上边缘竖直距离的像素值。

execCommand( )

IE *4(Win)* NN *n/a* Moz *1.3* Saf *1.3/2* Op *9* DOM *n/a*

```
execCommand("commandName"[, UIFlag[, value]])
```

执行已命名的命令。IE中的大多数命令须要为插入点首先创建一个TextRange对象，而在其他浏览器中，这些命令既可以作用于当前文本插入点，也可以影响文本选择。请参考附录D（译注5）以获取命令列表。

返回值 布尔值：如果命令执行成功，返回true，否则返回false。

参数

commandName

命令名称的字符串，不需要区分大小写。请参见附录D。

UIFlag

可选用的布尔值，包括：true，显示被命令触发的任何用户界面，false，则防止显示这些界面。

value

命令所需的参数值。

译注 5：此部分内容请读者访问 *http://www.china-pub.com/195581* 进行在线阅读。

getAnonymousElementByAttribute( )

参见addBinding()。

getAnonymousNodes( )

参见addBinding()。

getBindingParent( )

参见addBinding()。

getElementById( )

IE 5 NN n/a Moz all Saf all Op 7 DOM 1

```
getElementById("elementID ")
```

返回文档树中某个元素节点的引用，该节点的id属性与参数传入的值相匹配。如果不存在相匹配的节点，则返回null。尽管在观察节点名称时这个方法只不过是一个例行的工作，但是在W3C DOM浏览器中，它实际上是脚本通向元素对象的一扇大门。

返回值 一个元素节点对象的引用。

参数

elementID

目标元素的ID字符串。

getElementsByName( )

IE 5 NN n/a Moz all Saf all Op 7 DOM 1

```
getElementsByName("elementName")
```

根据参数传入的值从文档树中查询并返回name属性与之相匹配的所有元素。如果不存在相匹配的节点，则返回null。如果一个元素同时支持name和id属性，那么在Windows和Opera中的IE浏览器内，即使仅有id属性与参数值相同，也会在返回数组中包含这个元素。在其他支持这一方法的浏览器中，只有当元素的name属性明确设置为参数值时才会返回该元素，这中处理方式与W3C DOM规范的规定一致。

返回值 元素节点对象引用组成的一个数组。

参数

elementName

目标元素的名称字符串。

getOverrideStyle( )

IE n/a NN n/a Moz n/a Saf n/a Op n/a DOM 2

```
getOverrideStyle(elementReference, pseudoElement)
```

针对参数传入的元素，返回一个样式对象（CSSStyleDeclaration类型）。这个方法允许脚本获取XML文档的样式单属性，但这些XML文档并不具备HTML文档所具有的简便的样式功能。Safari 1.3/2在执行这一方法时，返回值为空。

返回值 类型为CSSStyleDeclaration的样式对象。

参数

elementReference

文档中某个元素对象的引用。

pseudoElement

一个表示伪元素的字符串，否则请使用null。

507

getSelection( )

IE *n/a* NN *4* Moz *all* Saf *n/a* Op *7* DOM *n/a*

在Navigator 4中，这个方法可以捕获文档中当前的文本选择。在Mozilla中不建议使用这个方法，而推荐使用window.getSelection()方法，后者会返回一个精巧的selection对象，而不是简单的文本。在Mozilla的document对象上调用这个方法还会导致JavaScript/错误控制台窗口输出一条警告，但此时不会抛出完整的异常。IE中它则等价于读取selection属性。

返回值 字符串。

参数 无。

handleEvent( )

IE *n/a* NN *4* Moz *n/a* Saf *n/a* Op *n/a* DOM *n/a*

```
handleEvent(event)
```

命令document对象接收并处理一个仅适用于Navigator 4的事件，该事件的描述将作为参数传入此方法。此时document对象必须拥有一个事件处理程序来处理这种类型的事件。

返回值 无。

参数

event

一个Navigator 4的event对象。

hasFocus( )

IE *6* NN *n/a* Moz *n/a* Saf *n/a* Op *n/a* DOM *n/a*

如果文档或文档中的任意一个元素获得了焦点，那么返回布尔值true。后台处理程序（例如一个由setTimeout()调用的方法）通过这个返回值就能够获知文档窗口是否当前桌面上的前端窗口。

返回值 布尔值：true | false。

参数 无。

importNode( )

IE *n/a* NN *n/a* Moz *all* Saf *all* Op *7* DOM *2*

```
importNode(nodeReference, deepBoolean)
```

从另一个已加载的文档中导入一个节点对象到当前文档，但并不将该节点放入文档树之中。从很多方面上来看，importNode()的工作方式和cloneNode()比较类似，但importNode()会假设源节点可能存在于一个完全不同的文档树环境中，例如一个XML文档环境。这个方法的W3C DOM规则控制着源节点的属性转移以及节点到达当前文档前的处理工作。例如，将一个Attr节点从一个文档中的某个元素中导入到新文档时，它会处于一种类似于文档片段的状态，在将它添加至新文档中的某个元素之前，它会失去其ownerElement（其值变为null）。须要注意的是，无法导入Document和DocumentType这两种类型的节点。

由于importNode()方法并不会承担文档间的持续责任，所以此时一个JavaScript变量就可以在这里发挥作用。同cloneNode()一样，importNode()方法并不会打乱源代码，而adoptNode()则会从初始位置移除对应的代码。

508

返回值 已导入的节点对象副本的引用。

参数

nodeReference

位于另一个已加载文档中的某个节点的引用，该文档还包括通过document.implementation.createDocument()方法加载至浏览器的未显示的文档。

deepBoolean

布尔值，用以控制副本是否包含所有的内嵌节点（true）或仅包含节点本身（false）。这是一个必选参数，它主要适用于元素节点。

loadBindingDocument( )

参见addBinding()。

normalizeDocument( )

IE *n/a* NN *n/a* Moz *n/a* Saf *n/a* Op *n/a* DOM *3*

与其他动作相比，此方法与调用normalize()方法的作用相同，它们均会立即作用于文档中的所有元素。而其他方法可能受当前文档DOMConfiguration对象的特定设置的影响。

返回值 无。

参数 无。

open( )

IE *3* NN *2* Moz *all* Saf *all* Op *7* DOM *1*

```
open(["MIMEType"][, "replace"])
```

开启一个输出流以便向当前窗口或框架进行写入。如果尚未调用document.clear()方法，那么会自动调用这个方法以响应document.open()。如果正在写入HTML文本，那么可以忽略这个方法，此时可以使用document.write()或document.writeln()开始正常写入。

返回值 无。

参数

MIMEType

将后续指令中将要写入的数据的MIME类型通知浏览器。Navigator支持“text/html”、“text/plain”、“image/ gif”、“image/jpeg”、“image/xbm”和“plugIn”。而IE仅支持“text/html”。

replace

一旦出现这个参数，为了改变当前文档会命令浏览器用即将被写入的文档代替历史列表内的入口文档。

postMessage( )

IE *n/a* NN *n/a* Moz *n/a* Saf *n/a* Op *8* DOM *n/a*

```
postMessage("messageText ")
```

即使两个文档处于不同的域，那么此方法也允许从其中一个文档向另一个文档传递文本数据。前提是，首先这两个文档必须已经完成加载，其次目的文档必须包含一个合适的引用，才能让这个方法完成工作。例如，如果主文档包含一个iframe元素，而该iframe则包含一个来自于另一个域的文档，那么主文档就能够引用另一个文档，并向它发送一个文本消息，代码如下所示：

```
var destDoc = document.getElementById("myIframe").contentDocument;
destDoc.postMessage("blue");
```

而目标文档中绑定的事件处理程序则会处理这个事件，并改变iframe元素内文档的背景色，代码如下所示：

```
document.addEventListener("message", receiver, false);
function receiver(evt) {
    if (evt.domain == "example-friendly-place.com") {
        document.body.style.backgroundColor = evt.data;
    }
}
```

另外还可以明确地给目的文档中的其他脚本变量指定字符串数据，其中包括能够转换成数组或对象（使用JSON，JavaScript Object Notation）的数据。这一机制来自于WHATWG的Web应用1.0规范，请参见：*http://whatwg.org/specs/web-apps/current-work*。

返回值 无。

参数

messageText

将要传递到另一个文档的字符串值。

queryCommandEnabled( )

IE *4(Win)* NN *n/a* Moz *n/a* Saf *1.3/2* Op *9* DOM *n/a*

```
queryCommandEnabled("commandName")
```

指定能否在文档或选择的当前状态下调用对应的命令。

返回值 布尔值：如果可以调用，则返回true，否则返回false。

参数

commandName

命令名称的字符串，不须要区分大小写。请参见附录D。

queryCommandIndeterm( )

IE *4(Win)* NN *n/a* Moz *n/a* Saf *1.3/2* Op *n/a* DOM *n/a*

```
queryCommandIndeterm("commandName")
```

指明命令是否处于一个不确定的状态。

返回值 布尔值：true | false。

参数

commandName

命令名称的字符串，不须要区分大小写。请参见附录D。

queryCommandState( )

IE *4(Win)* NN *n/a* Moz *n/a* Saf *1.3/2* Op *9* DOM *n/a*

```
queryCommandState("commandName")
```

决定指定命令的当前状态。

返回值 如果命令已完成，返回true；如果未完成，返回false；如果无法准确判断其状态，则返回null。

参数

commandName

命令名称的字符串，不须要区分大小写。请参见附录D。

queryCommandSupported( )

IE *4(Win)* NN *n/a* Moz *n/a* Saf *1.3/2* Op *9* DOM *n/a*

```
queryCommandSupported("commandName")
```

决定文档对象是否支持对应的命令。

返回值 布尔值：true | false。

参数

commandName

命令名称的字符串，不须要区分大小写。请参见附录D。

queryCommandText( )

IE *4(Win)* NN *n/a* Moz *n/a* Saf *1.3/2* Op *9* DOM *n/a*

```
queryCommandText("commandName")
```

返回与命令相关的文本。

返回值 字符串。

参数

commandName

命令名称的字符串，不须要区分大小写。请参见附录D。

queryCommandValue( )

IE *4(Win)* NN *n/a* Moz *n/a* Saf *1.3/2* Op *9* DOM *n/a*

```
queryCommandValue("commandName")
```

返回与命令相关的值，例如选中部分的字体名称。

返回值 由具体命令决定。

参数

commandName

命令名称的字符串，不须要区分大小写。请参见附录D。

recalc( )

IE *5(Win)* NN *n/a* Moz *n/a* Saf *n/a* Op *n/a* DOM *n/a*


```
recalc([allBoolean])
```

强制对表达式进行重算工作，该表达式已通过setExpression()方法指定给元素属性。只有当用户动作并未激发重算工作，而受影响的值可能发生改变的情况才须要调用这个方法。

返回值 无。

参数

allBoolean

设置为true时，命令文档内所有的动态属性表达式进行重算。而默认值false则命令浏览器决定自最后一次手动或自动重算后哪些表达式因改变而受到了影响，从而须要重算。

releaseEvents( )

IE *n/a* NN *4* Moz *all* Saf *all* Op *8* DOM *n/a*

```
releaseEvents(eventTypeList)
```

此方法与document.captureEvents()相对，它根据参数列表中指定的特定事件名称，在文档级关闭对应的事件捕获。尽管这个方法最初只是Navigator 4事件模型的一部分，但它之后陆续被Mozilla、Safari和Opera 8及其后续浏览器所支持，它实际上为document对象提供了一种与W3C DOM事件监听器消除方法等价的方法。尽管这个方法可能比较简便，但还是应该按照W3C DOM事件监听器语法编写新代码，请参照在线参考VI。

返回值 无。

参数

eventTypeList

一个使用逗号分隔的事件类型列表，对应的事件类型来自于可用的Event对象常量，如Event.CLICK或Event.MOUSEMOVE，注意区分字符大小写。

removeBinding( )

参见addBinding()。

renameNode( )

IE *n/a* NN *n/a* Moz *n/a* Saf *n/a* Op *n/a* DOM *3*

```
renameNode(nodeReference, "namespaceURI", "qualifiedName")
```

重命名当前文档中的一个元素或属性节点。在这个语境中，元素的名称就是其标签名。

返回值 如果最近的重命名生效，则返回被重命名的节点的引用。

参数

nodeReference

节点引用，其名称即将被修改。

namespaceURI

命名空间URI字符串，如果与命名空间无关则使用一个空字符串。

512

qualifiedName

元素或属性的全称，其中包括命名空间前缀。

routeEvent( )

IE *n/a* NN *4* Moz *all* Saf *n/a* Op *n/a* DOM *n/a*

```
routeEvent(event)
```

通常在事件处理方法中使用这个方法，它会命令Navigator 4让事件传递至其预期目标对象。尽管这个方法在Mozilla内不会导致错误，但它也不会产生任何的作用。

返回值 无。

参数

event

一个Navigator 4的event对象。

write( ), writeln( )

IE *3* NN *2* Moz *all* Saf *all* Op *7* DOM *n/a*

```
write("string 1"[, ..." string N"]) writeln("string 1"[, ..." string N"])
```

在页面加载时调用这两个方法，它们能够动态地向页面添加内容。如果在页面加载完成后再调用这两者中的一个方法，那么它会清除当前文档的内容，然后打开一个新的输出流并向窗口或框架内写入内容。之后必须使用一个document.close()方法进行关闭。由于第一个document.write()或document.writeln()方法销毁了当前文档，因此不要使用多条写入命名来创建一个新文档。此时请将内容加载至一个变量，并且将该变量作为参数传递至一个单独的document.write()或document.writeln()方法。

很多早期浏览器会将结束脚本标签解释为脚本写入结束，因此在这些浏览器中为<script>标签使用document.write()所出现的情况会比较复杂。 然而如果在字符串中分割结束脚本标签，那么就应该能够获得成功：

```
document.write("<" + "/script>");
```

另外，如果还包含了“隐藏脚本”注释的策略，那么请以这种方式书写代码：

```
document.write("//" + ">");
```

这两种方法的区别在于，document.writeln()会在写入文档时向源代码中加入一个回车。虽然这并不会反映在显示的内容中，但可以使得那些支持动态内容读取的浏览器能够更简便地读取动态源代码。

返回值 无。

参数

string

任意字符串，包含HTML标签。

513

Document

IE *n/a* NN *n/a* Moz *all* Saf *all* Op *8* DOM *1*

之前在W3C DOM架构中描述的document对象是一种更为特别的HTMLDocument节点，它是标准HTML模型中的一员。HTMLDocument节点从Document节点中继承了属性和方法（此处使用大写的“D”进行描述，它定义在W3C DOM核心模型中）。这是一个纯粹的抽象Document节点，而且它所须要包含的就是一个非显示的XML文档。

Mozilla和Opera 9对这个节点进行了扩展，并引入了一个load()方法，这个方法允许脚本将XML文档加载至一个简单的（且不可见的）Document节点。可以通过document.implementation.createDocument()方法来创建这类节点。然后脚本就可以通过W3C DOM文档树的属性和方法访问文档中的XML数据。然而XMLHttpRequest对象依然是将外部XML数据加载至浏览器的首选方式。

为了加强对document对象的继承特性的认识，对于这个通常使用脚本来访问的对象（即每个窗口文档均包含的HTMLDocument节点实例），下文列出了核心Document对象的属性和方法。关于这些属性和方法（除未被继承的load()方法之外）完整说明，请参见document对象，本章已对它进行了介绍。

对象模型引用方式 *documentNodeReference*

对象特定属性 doctype、documentElement、documentURI、domConfig、implementation、inputEncoding、strictErrorChecking、XMLEncoding、XMLStandalone、XMLVersion

对象特定方法 adoptNode()、createAttribute()、createAttributeNS()、createCDATASection()、createComment()、createDocumentFragment()、createElement()、createElementNS()、createEntityReference()、createProcessingInstruction()、createTextNode()、getElementById()、getElementsByTagName()、getElementsByTagNameNS()、importNode()、load()、normalizeDocument()、rename()

对象特定事件属性 无。

load( )

IE *n/a* NN *n/a* Moz *all* Saf *n/a* Op *9* DOM *n/a*


```
load("URI")
```

将一个XML文件加载至当前Document对象。如果试图加载其他类型的文件，如HTML文件，将会抛出一个异常。此外还必须对服务器进行设置，使之按照“text/xml”的内容形式发送文件。

返回值 无。

参数

URI

一个指向外部XML文件的URI字符串。

DocumentEvent

参见document。

DocumentFragment

IE *6* NN *n/a* Moz *all* Saf *all* Op *7* DOM *1*

从本质上来说，W3C DOM的DocumentFragment对象是其他DOM节点与上下文无关的容器。即可以在不受DocumentFragment提供的包含关系的限制下，使用所有的节点属性与方法在文档树外汇聚一系列的元素或文本节点。如果再向文档树中附加或插入DocumentFragment节点，那么DocumentFragment容器会随之消失，而它的节点内容则会根据它们的相对位置独立地出现在文档树中。由于元素节点既可以充当文档树之外的临时容器，也可以作为插入文档树之后的容器，因此由元素节点封装的内容并不一定须要通过DocumentFragment才能聚集在一起。但是，如果内容分块的一端或两端都在一个文本节点中结束，那么DocumentFragment节点就可以提供一个透明的存储区域以便将节点的字符串内容保存在一起，直到将它们融入到文档之中。

在脚本代码中可以通过document.createDocumentFragment()方法来创建一个空的DocumentFragment容器。DocumentFragment类型的节点从Node对象继承了全部的属性和方法，以便插入或附加其他新创建的节点，与此同时，除了能够存储其他节点，它并未加入其他自有的功能。须要注意的是，不要将DocumentFragment节点与为元素innerHTML属性而指定的标记文本混为一谈。W3C DOM直到Level 3为止，仍然未提供这种字符串到节点层次的转换。

对象模型引用方式 *documentFragmentNodeReference*

对象特定属性 无。
对象特定方法 无。
对象特定事件 无。

515

DocumentRange

参见document。

DocumentStyle

参见document。

DocumentTraversal

IE *n/a* NN *n/a* Moz *1.0.1* Saf *1.3/2* Op *8* DOM *2*

DocumentTraversal对象定义在W3C DOM的遍历(Traversal)模型中，这个模型还定义了createNodeIterator()和createTreeWalker()方法。这些方法（以及其他隐式DocumentTraversal接口）已经融入了document对象，因此脚本可以通过document.createTreeWalker()和createNodeIterator()来访问它们。

DocumentType

IE *5(Mac)* NN *n/a* Moz *all* Saf *n/a* Op *7* DOM *1*

如果有DOCTYPE元素作为文档流的一部分进入浏览器，那么DocumentType即可反映这个DOCTYPE元素。在W3C DOM中，DocumentType对象就是其自身的节点类型，并且正如它在文档的源代码中所处的位置一样，该对象也处于文档树的内容部分之外。在支持这一对象的浏览器中，可以通过document.doctype属性对其进行访问。如果在文档文件中不存在DOCTYPE元素，那么此属性将返回null。

DocumentType对象的属性暴露了DOCTYPE标签中的部分数据块，而这个DOCTYPE标签的结构则由SGML（Standard Generalized Markup Language，标准通用标记语言）确定。W3C DOM Level 2规范为这些数据块提供了占位符属性，并且Mozilla和Opera均不同程度地实现了大部分的占位符。但是从DOM规范可以清晰地看到，规范和实现尚未完全结合起来。

对象模型引用方式 *documentTypeNodeReference*
对象特定属性 entities、internalSubset、name、notations、publicId、systemId
对象特定方法 无。
对象特定事件 无。

516

entities

IE *n/a* NN *n/a* Moz *all* Saf *n/a* Op *7* DOM *1*

只读

返回由嵌套在DOCTYPE元素之内的Entity节点组成的数组。可以根据如下语法来格式化每个Entity（它会出现在DOCTYPE元素的尖括号之内）：

```
[<!ENTITY publicID "systemID">]
```

这个属性主要适用于XML文档。

值 Entity节点对象引用组成的数组（从技术上说，它是NamedNodeMap数据类型）。
默认值 null

internalSubset

IE *n/a* NN *n/a* Moz *all* Saf *n/a* Op *7* DOM *2*

只读

返回元素内部子集部分的字符串。

值 字符串。

默认值 空字符串。

name

IE *5(Mac)* NN *n/a* Moz *all* Saf *n/a* Op *7* DOM *1*

只读

返回元素名称部分的字符串。该名称是是紧随DOCTYPE元素标签名之后的第1个词。在这本书的主题范围内，这个值在所有的HTML和XHTML文档中均为“html”。须要注意的是，尽管IE 5/Macintosh已实现了这个对象及属性，但该浏览器会返回DOCTYPE元素的整个内部字符串，只不过以“html”开头而已。

值 字符串。

默认值 html

notations

IE *n/a* NN *n/a* Moz *all* Saf *n/a* Op *7* DOM *1*

只读

返回由DOCTYPE元素内Notation节点引用组成的数组。

值 Notation节点对象引用组成的数组（从技术上说，它是NamedNodeMap数据类型）。

默认值 null

publicId

IE *n/a* NN *n/a* Moz *all* Saf *n/a* Op *7* DOM *2*

只读

返回元素公共标识符部分的字符串内容。这个数据显示了DTD的类型，如“-//W3C//DTD XHTML 1.0 Strict//EN”。

值 字符串。

默认值 空字符串。

systemId

IE *n/a* NN *n/a* Moz *all* Saf *n/a* Op *7* DOM *2*

只读

返回元素系统标识符部分的字符串内容。这个数据显示了DTD的URI，如：“*http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd*”。

值 字符串。

默认值 空字符串。

DocumentView

参见document。

DOMException

IE *n/a* NN *n/a* Moz *all* Saf *all* Op *7* DOM *1*

某些作用于W3C DOM对象上的操作会触发错误，或者用JavaScript 1.5的术语来说，即：如果出现错误就会

抛出异常。因此W3C DOM定义了一个对象，用以传达对应于异常列表的代码号。例如，如果试图将一个文本节点附加至另一个文本节点，以作为其子节点，那么这个操作的appendChild()方法就会抛出一个代码号为3的异常。这个数字对应的异常表示：尝试在DOM层次关系上执行非法或不符合逻辑的操作。此时它表示文本节点无法包含任何子节点。

将DOM异常信息传达至脚本编写者的工作则由托管环境来完成，而不是DOM本身。自从JavaScript 1.5开始，由于JavaScript已经拥有了一个异常处理机制，因此将DOMException系统与JavaScript异常处理混合在一起的任务首次落在了Mozilla身上。这个新的机制允许不同类型的错误对象在异常处理操作中传播，因此在对这个机制进行扩展，以适应W3C DOM的DOMException对象及某些浏览器的特定错误之后，还能保持原始系统的完整性。对所有类型的异常处理均在try/catch结构中的catch块中完成，并且与异常相关的所有信息都会以一个对象的形式通过一个参数传递至catch块。

Mozilla的DOM异常对象（它体现了W3C的DOMException对象）在到达catch块时附带了很多相关的属性和方法，它们远比其他原因导致的异常（例如，试图使用一个未被初始化的JavaScript变量）所带来的属性和方法要多得多。code属性正是DOMException对象所独有的属性，而且并未出现在其他类型的对象中。此外，任意一个介于1~17之间的代码均指出了一种异常类型，DOM正式规范的3个级别均能识别这些异常类型。而Mozilla则将1000作为其浏览器特定异常列表的起始代码号。

如果希望沿着异常发生的执行路径来处理真实的W3C DOM异常，那么可以使用与下文类似的代码结构，在这段代码里，允许DOMException列表在迭代中增长到999：

```
try {
    // 此处可放置与 DOM 相关的脚本代码
}
catch(e) {
    if (typeof e.code == "number") {
        if (e.code < 1000) {
            // 处理 DOMException 对象
        } else {
            // 处理 Netscape 的 DOM 异常对象
        }
    } else {
        // 其他异常处理代码
    }
}
```

当然，那些异常细节很可能并不会使用户受益，但它们对开发而言却有着巨大的价值。如果想进一步了解异常处理，请参考第5章中的error对象。

对象模型引用方式 *errorObjectReference*
对象特定属性 code
对象特定方法 无。
对象特定事件 无。

code

IE *n/a* NN *n/a* Moz *all* Saf *all* Op *7* DOM *1*

只读

提供一个对应于已定义的DOM错误类型的整数。下表罗列了所有的代码值、对应的等价常量，并示范了什么样的问题会抛出该异常。

代码	常量	常见原因
1	INDEX_SIZE_ERR	偏移量整数超出了目标对象的范围
2	DOMSTRING_SIZE_ERR	对托管的语言而言，属性字符串过长

续表

代码	常量	常见原因
3	HIERARCHY_REQUEST_ERR	将一个节点附加至一个无法接收子节点的节点上
4	WRONG_DOCUMENT_ERR	未通过导入过程进行传递就插入了来不同文档的节点
5	INVALID_CHARACTER_ERR	使用非法字符为标识符赋值
6	NO_DATA_ALLOWED_ERR	向不允许拥有数据的节点分配数据
7	NO_MODIFICATION_ALLOWED_ERR	为只读属性赋值
8	NOT_FOUND_ERR	方法参数指向一个在对象作用域内不存在的节点
9	NOT_SUPPORTED_ERR	在HTML文档内调用一个仅适用于XML文档的方法
10	INUSE_ATTRIBUTE_ERR	方法参数所使用的Attr节点已属于另一个元素(未首先进行Attr复制)
11	INVALID_STATE_ERR	引用了一个不可读或不可写的节点
12	SYNTAX_ERR	键盘输入错误
13	INVALID_MODIFICATION_ERR	修改节点类型
14	NAMESPACE_ERR	名空间不匹配或名称错误
15	INVALID_ACCESS_ERR	进行了错误的访问
16	VALIDATION_ERR	一个尝试中的节点操作会导致节点无效
17	TYPE_MISMATCH_ERR	对象类型错误

示例

```
if (e.code == e.INVALID_CHARACTER_ERR) {
    // 处理非法的标识符字符
}
```

值 整数值。

默认值 由错误决定。

DOMImplementation

参见implementation。

DOMParser

IE *n/a* NN *n/a* Moz *1.0.1* Saf *1.2* Op *8* DOM *n/a*

DOMParser对象可以将一个XML标记字符串转换成一个XML的document对象（节点类型为9）。浏览器在内部使用这个对象，但如果其他任务也需要它，则脚本也可以使用这个对象。Mozilla暴露了该对象的parseFromBuffer()和parseFromStream()方法，但对Web页面相关的客户端脚本而言，可能无法应用某些必需的参数所使用的数据类型。

对象模型引用方式 new DOMParser()

对象属性 无。

对象方法 parseFromString()

对象特定事件 无。

parseFromString( )

IE *n/a* NN *n/a* Moz *1.0.1* Saf *1.2* Op *8* DOM *n/a*

```
parseFromString("string", " MIMEType")
```

如果解析成功就返回一个document对象，其内容由参数传入的字符串指定的节点树组成。

返回值 document对象。

参数

string

字符串形式的XML标记，它会被转换成为一个完整的XML文档。

MIMEType
一个字符串形式的内容类型，例如“text/xml”、“application/xml”或“application/xhtml+xml”。

dt

IE *4* NN *n/a* Moz *all* Saf *all* Op *7* DOM *1*

dt对象对应于dt元素。

等价HTML元素 <dt>
对象模型引用方式 [window.]document.getElementById("*elementID*")
对象特定属性 noWrap
对象特定方法 无。
对象特定事件 无。

noWrap

IE *4* NN *n/a* Moz *n/a* Saf *n/a* Op *n/a* DOM *n/a*

可读/可写

指明浏览器是否应该按照须要加宽元素的宽度，以便将一行不间断的文本显示在一整行上。当单元格内内容很多时，如果滥用此属性，那么为了显示所有内容会在页面上出现一个非常不便于使用的横向滚动条。

示例 document.getElementById("wideItem").noWrap = "true";
值 布尔值：true | false。
默认值 false

Element

IE *4* NN *n/a* Moz *all* Saf *all* Op *7* DOM *1*

W3C DOM的Element对象来自于其核心模型，它表示在真实的XML文档内发现的一类元素对象。这个节点类型从根Node对象继承了属性和方法，并且增加了一些功能，使它能够成为其他节点的容器。HTML文档内的元素是HTMLDocument类型，而它就继承自这个Element对象。正如本章开头部分所述，所有的HTML元素对象均会共享Element对象的全部属性和方法。DOM Level 3还包含某些额外的属性和方法，但可能所有的浏览器都未实现这些属性和方法。

对象模型引用方式 elementNodeReference
对象特定属性 tagName
对象特定方法 getAttribute()、getAttributeNode()、getAttributeNodeNS()、getAttributeNS()、hasAttribute()、hasAttributeNS()、removeAttribute()、removeAttributeNode()、removeAttributeNS()、setAttribute()、setAttributeNode()、setAttributeNodeNS()、setAttributeNS()
对象特定事件属性 无。

ElementCSSInlineStyle

IE *6* NN *n/a* Moz *all* Saf *all* Op *7* DOM *2*

W3C DOM的ElementCSSInlineStyle对象来自于StyleSheets模型，它表示通过style属性为一个元素指定的样式设置。由于HTMLElement对象（即HTML文档中的所有元素）与ElementCSSInlineStyle对象建立了联系，从而获得了它的style属性。而ElementCSSInlineStyle对象的唯一一个属性就是style，HTML元素正是获得了这个属性。尽管脚本并不会触及到这个对象，但W3C DOM中的动态样式在抽象模式下并不能脱离这个对象而独立存在。

对象模型引用方式 无。

对象特定属性 无。
对象特定方法 无。
对象特定事件 无。

elements

IE 3 NN 2 Moz all Saf all Op 7 DOM 1

一个表单内所包含的全部元素的集合。此集合中的成员按照它们在源代码中的次序进行排序。由于每个表单元素均包含一个type属性，因此脚本可以遍历所有的元素以便找到特定类型的元素，例如，所有的复选框元素。

对象模型引用方式

```
document.forms[i].elements
document.formName.elements
```

对象特定属性 length
对象特定方法 item()、namedItem()、tags()
对象特定事件 无。

length

IE 3 NN 2 Moz all Saf all Op 7 DOM 1

只读

返回集合中表单控件元素的数量。

示例

```
var howMany = document.forms[0].elements.length;
```

值 整数值。

item( )

IE 4 NN n/a Moz all Saf all Op 7 DOM 1

```
item(index [,subindex]), item(index)
```

根据符合索引值（IE中也可以使用index或subindex值）的元素对象，返回一个单独的表单控件对象或一组表单控件对象集合。

返回值 一个表单控件对象或一个表单控件对象集合（数组）。如果并不存在与参数相匹配的对象，则返回null。

参数

index

当输入参数是一个从零开始计数的整数（根据W3C规范）时，返回值是源代码内与某个具体项相对应的独立元素。如果参数为一个字符串，则返回id或name属性与该字符串相符的元素集合。

subindex

如果在IE中为第1个参数指定一个字符串值，那么可以将第2个参数指定为一个从零开始计数的索引值，这样就可以从集合中找到一个id或name属性与第一个参数的字符串相匹配的具体元素。

namedItem( )

IE 6 NN n/a Moz all Saf all Op 7 DOM 1

```
namedItem(IDOrName)
```

返回单个表单控件对象或表单控件对象集合，它们对应于符合参数字符串值的元素。

返回值 一个表单控件对象或一个表单控件对象集合（数组）。如果并不存在与参数相匹配的对象，则返回null。

参数

IDOrName

与目标元素的id或name属性包含相同值的字符串。

tags( )

IE 4 NN n/a Moz n/a Saf all Op 7 DOM n/a

```
tags(tagName)
```

从当前集合的全部内嵌对象中，返回标签与*tagName*参数相匹配的对象集合。在所有的IE、Safari和Opera集合中均已实现此方法（请参见`all.tags()`方法），但对同一类型的元素集合而言完全它是一种冗余。

em

请参见abbr。

embed

IE 4 NN 3 Moz all Saf all Op 7 DOM n/a

`embed`对象对应于`embed`元素。尽管W3C标准使用`object`元素加载外部内容（能够触发诸如媒体播放器之类的事物），但Mozilla浏览器依然中意于使用`embed`元素处理这类任务。通过`embed`对象还可以调用由播放器暴露的属性和方法，这样一来，脚本在处理`embed`对象时可以认为已经扩展了这个对象的脚本操作能力。此处列出了元素对象暴露给脚本的属性，它们均来自于元素本身而不是任何外部控制器。

须要注意的是，IE 5/Macintosh对`object`对象和`embed`对象的处理方式相同，而且它暴露的属性更接近于一个`object`或`applet`对象，但与`embed`对象有一定的差别。然而由于IE/Mac并不允许脚本与外部播放器或控制器进行通信，因此这也不会存在什么问题。

等价HTML元素 `<embed>`

对象模型引用方式

```
[window.]document.getElementById("elementID")
[window.]document.embeds[i]
```

对象特定属性 `align`、`height`、`hidden`、`name`、`palette`、`pluginspage`、`src`、`type`、`units`、`width`

对象特定方法 `getSVGDocument()`

对象特定事件 无。

align

IE 5(Mac) NN n/a Moz all Saf n/a Op n/a DOM n/a

可读/可写

定义元素在其容器内的对齐方式。请参见第1章起始部分中对齐常量一节，该节对不同值所包含的不同含义给出了解释。

示例 `document.getElementById("audioPlayer").align = "center";`

值 任意一个对齐常量，包括：`absbottom | absmiddle | baseline | bottom | left | middle | right | texttop | top`。

默认值 `left`

height, width

IE 4 NN n/a Moz all Saf all Op n/a DOM n/a

可读/可写

给出标签属性中所设置的元素高度与宽度，单位为像素。小程序被加载后，改变这两个值无法有效地改变其实际矩形大小。

示例 `var controllerHeight = document.embeds["audioPlayer"].height;`

值 整数值。

默认值 `0`

hidden

IE *4* NN *n/a* Moz *n/a* Saf *n/a* Op *n/a* DOM *n/a* 525

可读/可写

指定是否允许在屏幕上显示内嵌数据的插件控制面板。改变这个属性后，会强制页面重新绘制页面内容，以便为插件控制面板腾出空间，或者关闭已隐藏的面板所占据的空间。

示例 `document.embeds["jukebox"].hidden = true;`
值 布尔值：`true | false`。
默认值 `false`

name

IE *4* NN *n/a* Moz *all* Saf *n/a* Op *n/a* DOM *n/a*

可读/可写(IE)

反映了元素标签的`name`属性值。

示例 `document.embeds["myEmbed"].name = "tunes";`
值 区分大小写的标识符，必须遵循标识符命名规则：不包含空白、不能以数字作为起始字符、除下划线外不允许使用其他标点符号。
默认值 无。

palette

IE *4(Win)* NN *n/a* Moz *n/a* Saf *n/a* Op *n/a* DOM *n/a*

只读

返回`embed`元素对象的`palette`属性设置。

值 字符串。
默认值 无。

pluginspage

IE *4(Win)* NN *n/a* Moz *n/a* Saf *n/a* Op *n/a* DOM *n/a*

只读

为相关插件的下载和安装提供URL链接，以便运行当前对象的内嵌数据。

值 字符串形式的完整或相对URL。
默认值 无返回值，但IE拥有默认的URL插件信息。

src

IE *4* NN *n/a* Moz *all* Saf *all* Op *7* DOM *1* 526

可读/可写

指定与对象相关的外部内容文件的URL地址。尽管某些控制器可能会响应此属性的修改，但更可靠的方法还是使用它提供的加载方法或属性重新向控制器内加载一个不同的文件。

示例 `document.embeds["myEmbed"].src = "tunes/dannyboy.wav";`
值 字符串形式的完整或相对URL。
默认值 无。

type

IE *n/a* NN *n/a* Moz *all* Saf *n/a* Op *n/a* DOM *n/a*

只读

指明已分配给元素`type`属性的外部数据的MIME类型。

示例 `var dataMIME = document.embeds["myEmbed"].type;`
值 任何一个有效的MIME类型名称字符串，它包含类型和子类型，两者之间使用正斜杠（/）分隔。整个字符串使用引号括起来。

默认值 无。

units

IE *4(Win)* NN *n/a* Moz *n/a* Saf *n/a* Op *n/a* DOM *n/a*

可读/可写

指定度量元素高度和宽度时所使用的单位。IE会按照像素单位对这些设定进行处理。

示例
```
document.getElementById("myEmbed").units = "ems";
```
值 任意一个字符串常量，不需要区分大写：`pixels | px | em`。

默认值 `pixels`

width

参见height属性。

getSVGDocument( )

IE *6* NN *n/a* Moz *1.8* Saf *1.3/2* Op *9* DOM *n/a*

返回一个`document`对象的引用，它在*.svg*文档内包含了XML数据。然后就可以在这个对象上调用W3C DOM核心模型的方法。

返回值 `embed`元素加载的*.svg*文档对象的引用。

参数 无。

embeds

IE *4* NN *n/a* Moz *all* Saf *all* Op *7* DOM *1*

当前元素所包含的所有`embed`元素的集合。此集合中的成员按照它们在源代码中的次序进行排序。Internet Explorer浏览器允许使用数组符号来访问集合中的一个元素。

对象模型引用方式 `document.embeds`

对象特定属性 `length`

对象特定方法 `item()`、`namedItem()`、`tags()`

length

IE *4* NN *n/a* Moz *all* Saf *all* Op *7* DOM *1*

只读

返回集合中元素的数量。

示例
```
var howMany = document.embeds.length;
```
值 整数值。

item( )

IE *4* NN *n/a* Moz *all* Saf *all* Op *7* DOM *1*

`item(`*index* `[,` *subindex*`])`, `item(`*index*`)`

根据符合索引值(IE中也可以使用`index`或`subindex`值)的元素，返回一个单独的`embed`元素对象或一组`embed`元素对象集合。

返回值 一个`embed`元素对象或一个`embed`元素对象集合。如果并不存在与参数相匹配的对象，则返回`null`。

参数

index

当输入参数是一个从零开始计数的整数（根据W3C规范）时，返回值是源代码内与某个具体项相对应的独立元素。如果参数为一个字符串，则返回`id`或`name`属性与该字符串相符的元素集合。

subindex

如果在IE中为第1个参数指定一个字符串值，那么可以将第2个参数指定为一个从零开始计数的索引值，这样就可以从集合中找到一个id或name属性与第1个参数的字符串相匹配的具体元素。

namedItem( )

IE *6* NN *n/a* Moz *all* Saf *all* Op *7* DOM *1*


```
namedItem(IDOrName)
```

返回单个embed元素对象或embed元素对象集合，它们对应于符合参数字符串值的元素。

返回值 一个表单控件对象或一个表单控件对象集合（数组）。如果并不存在与参数相匹配的对象，则返回null。

参数

IDOrName

与目标元素的id或name属性包含相同值的字符串。

tags( )

IE *4* NN *n/a* Moz *n/a* Saf *all* Op *7* DOM *n/a*

```
tags(tagNam)
```

从当前集合的全部内嵌对象中，返回标签名与*tagName*参数相匹配的对象集合。在所有的IE、Safari和Opera集合中均已实现此方法（请参见all.tags()方法），但对同一类型的元素集合而言完全它是一种冗余。

Entity, EntityReference

IE *4* NN *n/a* Moz *all* Saf *all* Op *7* DOM *1*

在W3C DOM中，Entity对象（节点类型之一）是一个元素的抽象描述，而在XML文档中，这类元素就是一个存储单元。某些实体还定义了一个名称，这样其他元素就可以将它作为一个引用来调用存储在实体中的信息。而在W3C DOM中，这个引用使用EntityReference节点类型来描述。在DTD文档中就可以看到很多Entity元素的范例。关于引用Entity元素的更多详细信息，请参考XML规范：*http://www.w3.org/TR/REC-xml/*。

event

IE *4* NN *4* Moz *all* Saf *all* Op *7* DOM *2*

由用户或系统生成的事件的有关信息就包含在event对象中。但目前存在着3种不同类型的event对象，在下列浏览器中各自实现了其中一种事件对象模型：Windows下的IE、Navigator 4和W3C DOM（已在Mozilla、Safari和Opera中实现）。而IE 5/Macintosh则实现了一种由IE/Windows版本和W3C DOM版本混合而成的事件对象模型。请参见在线参考VI，以便了解如何在跨浏览器环境下处理事件。由于event对象的属性很少能够适用于多个事件模型，因此要特别注意浏览器对每个属性的兼容性。

由于W3C DOM的Event对象框架暗含着面向对象的本质，因此从这个方面来看，这种事件对象是三者中较为复杂的一种。与那种完全封闭式的对象（如IE的event对象）不同，W3C DOM的event对象根据事件分类分别暴露了不同的属性和方法集。而所有的事件类均能共享W3C DOM的根Event对象的属性和方法。但实际上，真正的事件对象实例均属于某种Event对象子类别，甚至是次级子类别（sub-subclass）。这些子类别分别称为UIEvent（即所谓的用户界面事件，如DOMFocusIn）、MouseEvent（包含鼠标事件）、MutationEvent（由脚本引起的文档节点结构变化所激发的事件），以及DOM Level 3中的TextEvent（与输入相关的事件）和KeyboardEvent（键盘输入相关的事件）。在DOM Level 3中与键盘操作相关的事件定稿之前，Mozilla和其他W3C DOM浏览器已实现了一个名为KeyEvent的临时键盘事件分类。尽管这个分类借用了一些MouseEvent属性以便脚本操作，但它仍然属于UIEvent的子集。

从总体上说，由于特定事件类型的监听程序会查找与该事件相关的属性，因此针对事件对象的这种功能性划分并不会对W3C DOM事件处理产生影响。目前对事件类关注还比较少。另外，从面向对象的框架结构的角度来看，审视事件对象的属性和方法如何形成层次关联也是很有意义的。下表清楚地说明了W3C DOM事件

类中的属性分配情况（某些属性为Mozilla所独有）：

事件属性	Event	UI	Mouse	Keyboard	Text	Mutation
Event properties						
bubbles	•	•	•	•	•	•
cancelable	•	•	•	•	•	•
cancelBubble[a]		•	•	•	•	
currentTarget	•	•	•	•	•	•
eventPhase	•	•	•	•	•	•
originalTarget	•	•	•	•		•
target	•	•	•	•	•	•
timeStamp	•	•	•	•	•	•
type	•	•	•	•	•	•
UIEvent properties						
detail		•	•	•	•	
view		•	•	•	•	
MouseEvent properties						
altKey			•	•	•	
button			•			
clientX			•			
clientY			•			
ctrlKey			•	•	•	
metaKey			•	•	•	
relatedTarget			•			
screenX			•			
screenY			•			
shiftKey			•	•	•	
KeyboardEvent properties						
charCode[b]				•		
isChar[b]				•		
keyCode[b]				•		
rangeOffset[b]				•		
rangeParent[b]				•		
TextEvent property						
data					•	
MutationEvent properties						
attrChange						•
attrName						•
newValue						•
prevValue						•
relatedNode						•

a 在Mozilla中实现的IE属性，以方便跨平台部署。

b 实现在Mozilla和Safari中，以代替W3C DOM Level 3的键盘事件规范。

而下表则阐明了W3C DOM事件类中方法的分配情况：

	Event	UI	Mouse	Keyboard	Text	Mutation
Event methods						
initEvent()	•	•	•	•	•	•
getPreventDefault()	•	•	•	•		•

续表

	Event	UI	Mouse	Keyboard	Text	Mutation
preventDefault()	·	·	·	·	·	·
stopPropagation()	·	·	·	·	·	·
UIEvent methods						
initUIEvent()		·				
MouseEvent methods						
initMouseEvent()			·			
KeyboardEvent methods						
initKeyEvent()				·		
TextEvent methods						
initTextEvent()					·	
MutationEvent methods						
initMutationEvent()						·

Mozilla中的event对象还实现了Navigator 4中静态Event对象的属性，并且它还从W3C DOM的TextEvent对象中继承了大量的常量，这些常量可用来表示非字符或数字的键盘上的键代码，例如*eventObject*.DOM_VK_PAGE_UP。然而于此同时，这些键盘常量却被推荐的DOM Level 3事件模型所删除。因此如果在Mozilla内探索一个事件对象的属性，会发现这些常量仍然存在。在下文完成事件实例说明之后，会在有关Event对象的内容中继续讨论Navigator 4的静态Event对象的相关属性。

正如在线参考VI所述，在IE和W3C DOM的事件模型中，必须使用不同的脚本技术来获取一个event对象的引用。但是一旦获得这个引用，完全可以在不同浏览器中使用相同的事件处理方式。因此本章之后的示例代码片段均假设已通过脚本指令获得了浏览器特定的event对象引用，并且示例代码中使用变量evt来指代这个引用。

对象模型引用方式

*NN*和*W3C DOM*

eventObj

IE

[window.]event

对象特定属性

altKey、altLeft、attrChange、attrName、behaviorCookie、behaviorPart、bookmarks、boundElements、bubbles、button、cancelable、cancelBubble、charCode、clientX、clientY、contentOverflow、ctrlKey、ctrlLeft、currentTarget、data、dataFld、dataTransfer、detail、eventPhase、explicitOriginalTarget、fromElement、isChar、isTrusted、keyCode、layerX、layerY、metaKey、modifiers、newValue、nextPage、offsetX、offsetY、originalTarget、pageX、pageY、prevValue、propertyName、qualifier、rangeOffset、rangeParent、reason、recordset、relatedNode、relatedTarget、repeat、returnValue、screenX、screenY、shiftKey、shiftLeft、srcElement、srcFilter、srcUrn、target、timeStamp、toElement、type、view、wheelDelta、which、x、y

对象特定方法

getPreventDefault()、initEvent()、initKeyEvent()、initMouseEvent()、initMutationEvent()、initUIEvent()、preventDefault()、stopPropagation()

对象特定事件属性

无。

altKey

IE *4* NN *n/a* Moz *all* Saf *all* Op *7* DOM *2*

只读

如果在事件激发时按下了左侧或右侧的Alt键，那么返回true。

示例

```
if (evt.altKey) {
    // 处理 Alt 键被按下的情况
}
```

值 布尔值：true、false。

默认值 false

altLeft

IE *5.5* NN *n/a* Moz *n/a* Saf *n/a* Op *n/a* DOM *n/a*

只读

当事件激发时，如果按下了左侧的Alt键，那么返回true。

示例

```
if (evt.altLeft) {
    // 处理左侧 Alt 键被按下的情况
}
```

值 布尔值：true | false。

默认值 false

attrChange

IE *n/a* NN *n/a* Moz *all* Saf *all* Op *8* DOM *2*

只读

返回一个对应于Attr节点变化类型的整数代码，W3C DOM转变事件中的DOMAttrModified类型通常会引发这种变化。每个转变事件对象还拥有3个对应于该整数值的常量，虽然在DOMAttrModified事件处理中使用这些常量可能使得代码变得冗长，但却提高了脚本的可读性。下表罗列了整数值和对应的常量。尽管Safari也实现了这个属性，但它并未支持能触发这一属性的事件类型。但如果人工创建转变事件还是会包含这个属性。

值	常量	描述
1	*evtObj*.MODIFICATION	改变已存在的 Attr 节点的值
2	*evtObj*.ADDITION	Attr 节点已被添加至文档树
3	*evtObj*.REMOVAL	Attr 节点已从文档树中删除

示例

```
if (evt.attrChange == evt.MODIFICATION) {
    // 对属性值的修改进行后期处理
}
```

值 整数值：1 | 2 | 3。

默认值 无。

attrName

IE *n/a* NN *n/a* Moz *all* Saf *all* Op *8* DOM *2*

只读

返回Attr节点的名称字符串，该节点受到W3C DOM转变事件中DOMAttrModified事件类型的影响。尽管Safari也实现了这个属性，但它并不支持能触发这一属性的事件类型。但是如果人工创建转变事件就会包含这个属性。

示例

```
var changedAttr = evt.attrName;
```

值 字符串。

默认值 空字符串。

behaviorCookie, behaviorPart, bookmarks, boundElements

IE *6* NN *n/a* Moz *n/a* Saf *n/a* Op *n/a* DOM *n/a*

只读

这些属性由IE 6和IE 7中的event对象返回，返回值分别为0、0、null和空数组，但微软并未在相关文献中记录它们。可能会在以后的IE版本中最终支持并实现这些属性。

bubbles

IE *n/a* NN *n/a* Moz *all* Saf *all* Op *7* DOM *2*

只读

如果允许事件通过元素层次关系向上传递是事件的默认行为，那么返回布尔值true。

示例

```
if (evt.bubbles) {
    // 对事件冒泡进行处理
}
```

值 布尔值：true、false。

默认值 由事件的特定类型决定。

button

IE *4* NN *n/a* Moz *all* Saf *all* Op *7* DOM *2*

只读

指明按下了哪个鼠标按钮从而激发了鼠标事件。须要注意的是，在典型的Macintosh系统中，鼠标仅有一个按键。此外，如果要在IE/Windows中拦截右键单击上下文菜单，那么请使用oncontextmenu事件处理程序。

就这个属性的返回数值而言，IE和W3C DOM实现之间存在着明显的差异。W3C DOM（实现在Mozilla和其他浏览器中）使用0表示左（主）按钮被点击。而IE/Windows还为鼠标按钮组合增加了一些额外的值。

示例

```
if (evt.button == 2) {
    // 处理鼠标右键单击事件
}
```

值

整数值，如下表所示：

按钮	IE	W3C DOM	按钮	IE	W3C DOM
无按钮	0	null	左+右	3	n/a
左（主）	1	0	左+中	5	n/a
中	4	1	右+中	6	n/a
右	2	2	左+中+右	7	n/a

默认值 0

cancelable

IE *n/a* NN *n/a* Moz *all* Saf *all* Op *7* DOM *2*

只读

如果在目标元素上能够通过preventDefault()方法取消元素的默认行为，那么将为这种事件类型返回布尔值true。

示例

```
if (evt.cancelable ) {
    evt.preventDefault();
}
```

值 布尔值：true | false。
默认值 由事件的特定类型决定。

cancelBubble

IE 4 NN n/a Moz all Saf all Op 7 DOM n/a

可读/可写

指明事件是否应该向上传递至元素容器层。通常应该将这个属性设置为true，以替代默认设置并防止事件传播得过远。为了提供一定的方便，其他浏览器也大多实现了这个IE属性。W3C DOM中的等价物是event对象的stopPropagation()方法。

示例
```
evt.cancelBubble = true;
```
值 布尔值：true | false。
默认值 false

charCode

IE n/a NN n/a Moz all Saf all Op n/a DOM n/a

只读

返回一个整数，该整数对应于触发事件的键所生成的字符的Unicode值。这里的字符码与键码并不相同，字符码分为大写字母和小写字母（例如，97是“a”而65是“A”），但keyCode与该键相同，而与它所创建的字符无关。通常只有在onkeypress事件下这个属性才会包含一个值，而在keydown和keyup事件中，这个值是0。对于IE中的等价物，请参考keyCode属性。

W3C DOM Level 3事件模型并未一个提供完全一致的替代品，但它提出使用data属性来传达字符的Unicode值，例如，字符串“U+0041”表示大写字母A。

示例
```
if (evt.charCode > 96 && evt.charCode < 123) {
   evt.target.value += String.fromCharCode(evt.charCode - 32);
   evt.preventDefault();
}
```
值 整数值。
默认值 由事件特性决定。

clientX, clientY

IE 4 NN n/a Moz all Saf all Op 7 DOM 2

只读

指明当前事件触发时坐标所处的水平（x）和竖直（y）坐标。这些坐标与浏览器窗口或框架的可视文档区域相关。因此如果在IE中要将这些相对坐标转换为文档的绝对坐标，须要加入body元素的滚动条尺寸，而在IE 6或IE 7的标准兼容模式下，则须要加入html元素的滚动条尺寸。对Mozilla、Safari和Opera而言，由pageX和pageY属性提供文档空间内的坐标值。

示例
```
if ((evt.clientX >= 10 && evt.clientX <= 20) &&
(evt.clientY >= 50 && evt.clientY <= 100)) {
   // 在(10,50)和(20,100)限定的热点区域内处理点击事件
}
```
值 整数像素值。
默认值 无。

contentOverflow

IE 5.5 NN n/a Moz n/a Saf n/a Op n/a DOM n/a

只读

如果尚未显示的内容需要一个新排版框以处理多余的内容，则返回布尔值true。在部署自定义打印模板或打

印预览模板时，这个属性只能应用于onlayoutcomplete事件。针对这种模板所需要的更多C++编程信息，请访问以下链接：*http://msdn.microsoft.com/en-us/library/bb250460(VS.85).aspx*。

值 布尔值：true | false。

默认值 false

ctrlKey

IE *4* NN *n/a* Moz *all* Saf *all* Op *7* DOM *2*

只读

当事件激发时，如果左侧或右侧Control键被按下，那么返回true。测试这个键所需的跨浏览器事件处理代码请见在线参考VI。

示例

```
if (evt.ctrlKey) {
    // 处理 ctrl 键被按下的情况
}
```

值 布尔值：true | false。

默认值 false

ctrlLeft

IE *5.5* NN *n/a* Moz *n/a* Saf *n/a* Op *n/a* DOM *n/a*

只读

当事件激发时，如果按下了左侧Control键，则返回true。

示例

```
if (evt.ctrlLeft) {
    // 处理左侧 ctrl 键被按下的情况
}
```

值 布尔值：true | false。

默认值 false

currentTarget

IE *n/a* NN *n/a* Moz *all* Saf *all* Op *7* DOM *2*

只读

如果节点的事件监听器正在处理事件，那么就返回该节点的引用。这样就可以使得方法获知它是被真实的目标节点调用还是被事件传播过程中其他节点所调用。

示例

```
if (evt.currentTarget.nodeType == 1) {
    // 在元素层处理可能的文本节点
}
```

值 事件传播层次结构中的某个节点的引用。

默认值 事件目标对象的引用。

data

IE *|4|* NN *n/a* Moz *n/a* Saf *n/a* Op *n/a* DOM *n/a*

只读

为dragdrop事件（仅适用于Navigator 4）提供相关的附属数据。data属性会返回对应项的URL值，而该对应项将会被置入窗口或框架中。W3C DOM Level 3事件模型中的TextEvent可能会采用这个属性名称。

示例 `var srcDoc = evtObj.data;`

值 字符串。

默认值 无。

dataFld

IE *5(Win)* NN *n/a* Moz *n/a* Saf *n/a* Op *n/a* DOM *n/a*

可读/可写

用于IE的数据绑定，dataFld属性保存着与HTML表格列相关的数据源对象字段的名称。通过事件绑定在表格内生成一个oncellchange事件后，这个属性就会包含一个具体的值。

值 字符串。

默认值 空字符串。

dataTransfer

IE *5(Win)* NN *n/a* Moz *n/a* Saf *1.3/2* Op *n/a* DOM *n/a*

只读

返回dataTransfer对象的引用，以便在拖放操作过程中促进自定义的数据在来源元素与目的元素之间的移动。关于具体用法请参见dataTransfer对象。

值 dataTransfer对象的引用。

默认值 无。

detail

IE *n/a* NN *n/a* Moz *all* Saf *all* Op *7* DOM *2*

只读

返回一个整数，以转达事件类型特定的附加信息。针对鼠标按键事件，这个数字反映了用户在同一区域位置连续点击的次数。如果移动了光标，那么就会将此数重置为0，以准备下一次的鼠标点击。对于DOMActivate事件类型的detail属性，一个简单的用户动作（如点击鼠标或使用tab键）将返回1，而一个复杂的事件（如双击）则会返回2。

示例

```
if (evt.type == "click" && evt.detail > 5) {
   alert("Relax, dude!");
}
```

值 整数值。

默认值 由事件的特定类型决定。

eventPhase

IE *n/a* NN *n/a* Moz *all* Saf *all* Op *7* DOM *2*

只读

返回一个整数，以表示事件监听器正在哪个具体阶段处理事件，如，在捕获阶段、在事件目标之上，或者在向上传递阶段。W3C DOM事件对象还实现了3个对应的明语常量。

示例

```
if (evt.eventPhase == evt.AT_TARGET) {
  // 处理事件目标上的事件监听器
}
```

值

整数值，如下表所示。

值	常量
1	*eventObjectReference*.CAPTURING_PHASE
2	*eventObjectReference*.AT_TARGET
3	*eventObjectReference*.BUBBLING_PHASE

默认值 2

explicitOriginalTarget

IE *n/a* NN *n/a* Moz *1.4* Saf *n/a* Op *n/a* DOM *n/a*

只读

返回接收到实际事件的节点的引用。由于target属性在Mozilla 1.4中发生了改变，并指向一个元素对象（而不是一个元素所包含的文本节点），因此须要使用explicitOriginalTarget属性来访问接收到事件的具体节点。关于什么是匿名内容，在explicitOriginalTarget和originalTarget之间存在着技术上的差异，这也给在Mozilla浏览器下进行XUL开发带来了一定的问题。但在一个HTML文档中，这两个属性将引用同一个节点。

539

示例

```
if (evt.explicitOriginalTarget !== evt.target) {
    ...
}
```

值 元素或节点对象引用。

默认值 无。

fromElement

IE *4* NN *n/a* Moz *n/a* Saf *n/a* Op *n/a* DOM *n/a*

只读

返回一个对象引用，在mouseover或mouseout事件之前，光标正好位于该对象的上方。

示例

```
if (evt.fromElement.id == "lowerLevel") {
    ...
}
```

值 元素对象的引用。

默认值 无。

isChar

IE *n/a* NN *n/a* Moz *all* Saf *n/a* Op *n/a* DOM *n/a*

只读

如果键盘事件来自于一个字符按键，则返回true。但在实际情况下，包括功能键在内，Mozilla对所有的键均会返回false。请使用onkeydown或onkeyup事件处理程序来处理非字符键。

值 布尔值：true | false。

默认值 true

isTrusted

IE *n/a* NN *n/a* Moz *1.7.5* Saf *n/a* Op *n/a* DOM *n/a*

只读

如果事件是用户操作的一个结果，那么返回true。对于那些由人工脚本生成的事件，这个event对象属性则为false。

值 布尔值：true | false。

默认值 true

keyCode

IE *4* NN *n/a* Moz *all* Saf *all* Op *7* DOM *n/a*

540

可读/可写

针对按键事件，主流浏览器对这个属性的处理方式并不相同，但对于onkeydown和onkeyup事件处理程序，所有的浏览器均会采取同样的方式来处理keyCode属性。对于这些事件，无论键盘按键产生了什么字符，keyCode属性均会返回与该按键相关的代码。例如，在一个典型的拉丁文字符集键盘上，“A”键会产生代码65。而组合键在按下和释放时则会产生对应的事件和代码。

对于onkeypress事件，IE、Safari和Opera将返回一个有效值，它对应于通过字符输入而显示在文本框内的一个实际字符的ASCII值，例如，“A”为65而“a”为97。在Mozilla中的charCode属性与keypress事件等价。关于如何处理键盘事件，请见在线参考VI。

示例

```
if (evt.keyCode == 65) {
    ...
}
```

值 整数值。

默认值 无。

layerX, layerY

IE *n/a* NN *4* Moz *all* Saf *all* Op *n/a* DOM *n/a*

只读

提供当前事件触发时坐标所处的水平（x）和竖直（y）坐标。这些坐标与容器层有关。如果并未定义层或可定位元素，那么就以基础文档的默认层作为引用点，此时它们等价于pageX和pageY属性。在Mozilla中，对于输入文本和密码的input元素，textarea元素，以及select元素，会根据元素自身的矩形空间来测量这些属性。

示例

```
if ((evt.layerX >= 10 && evt.layerX <= 20) &&
(evt.layerY >= 50 && evt.layerY <= 100)) {
    // 在(10,50)和(20,100)限定的热点区域内处理点击事件
}
```

值 整数像素值。

默认值 无。

metaKey

IE *n/a* NN *n/a* Moz *all* Saf *all* Op *7* DOM *n/a*

只读

如果在事件触发时按下了键盘的Meta键（在Macintosh键盘中为Command键），则返回true。

示例

```
if (evt.metaKey) {
    // 处理 meta 键被按下的情况
}
```

值 布尔值：true | false。

默认值 false

modifiers

IE *n/a* NN *4* Moz *n/a* Saf *n/a* Op *n/a* DOM *n/a*

只读

提供一个整数，它表示Navigator 4的事件被触发时正按住哪些键盘组合键。可以使用“&”运算符与一系列的静态Event对象常量来判断某个特定的组合键是否已经被按下。详情请参见在线参考VI。

示例

```
var altKeyPressed = evt.modifiers & Event.ALT_MASK;
```

值 整数值。

默认值 0

newValue, prevValue

IE *n/a* NN *n/a* Moz *all* Saf *all* Op *8* DOM *2*

只读

这两个属性分别返回与W3C DOM转变事件类的DOMAttrModified和DOMCharacterDataModified事件类型

相关的新数据值与前一数据值。为修改元素属性或CharacterData节点内容而创建撤销操作的缓存时，这些信息将比较有用。

示例 `undoAttrBuffer = {attrNode:evt.relatedNode, oldVal:evt.prevValue};`

值 字符串。

默认值 空字符串。

nextPage

IE *5.5* NN *n/a* Moz *n/a* Saf *n/a* Op *n/a* DOM *n/a*

只读

返回一个字符串，它指明了自定义打印模板的下一页将出现在页面的左侧还是右侧。有关这种模板的更多C++编程信息，请访问以下链接：*http://msdn.microsoft.com/en-us/library/bb250460(VS.85).aspx*。

值 字符串常量：`left | right |`（空字符串）。

默认值 空字符串。

offsetX, offsetY

IE *4* NN *n/a* Moz *n/a* Saf *n/a* Op *n/a* DOM *n/a*

只读

提供事件触发时鼠标指针相对于容器元素（不包括填充、边框或边距）的左上坐标位置。访问这个两个属性前可以使用offsetParent属性来确定其容器元素。请参看本章前述部分中关于“client-和offset-属性集”，以了解IE中关于偏移量测量的一些异常情况。

示例

```
if (evt.offsetX <= 20 && evt.offsetY <=40) {
    ...
}
```

值 整数像素值。

默认值 无。

originalTarget

IE *n/a* NN *n/a* Moz *all* Saf *n/a* Op *n/a* DOM *n/a*

只读

返回一个节点引用，在Mozilla内部会将它作为事件的第一个目标。总体上来说，这一信息已深入到某些元素的内部构造之中，因此它对DHTML的脚本编程并不能提供多少帮助。例如，在早期版本的Mozilla中，在input元素中内嵌着一个div元素，但DOM节点树并不会将div元素视为input元素的一个子节点。针对很多事件和事件对象，target和originalTarget属性均引用了同一节点，但当元素包含一个文本节点时，其target属性会被设置为文本节点的父对象，即容器元素。

值 节点对象引用。

默认值 由元素特性决定。

pageX, pageY

IE *n/a* NN *n/a* Moz *all* Saf *all* Op *7* DOM *n/a*

只读

提供事件触发时元素内容相对于页面区域左上角的左、上坐标。在进行尺寸测量时会忽略页面上的所有滚动。

示例

```
if (evt.pageX <= 20 && evt.pageY <=40) {
    ...
}
```

值 整数像素值。

默认值 无。

prevValue

请参见newValue。

propertyName

IE 5(Win) NN n/a Moz n/a Saf n/a Op n/a DOM n/a

可读/可写

返回一个字符串，该字符串中包含一个随着onpropertychange事件而发生改变的对象属性名称。对于其他事件类型而言，返回值为一个空字符串。如果这个被改变的属性本身就来自于某个属性（例如，一个元素的style属性中的某个属性），那么返回值会包含一个圆点，表示子属性，例如“style.color”。

示例

```
if (evt.propertyName.indexOf("style") == 0) {
   // 在一个已改变的样式上执行进一步的操作
}
```

值 属性名称字符串。

默认值 空字符串。

qualifier

IE 5(Win) NN n/a Moz n/a Saf n/a Op n/a DOM n/a

可读/可写

此属性通常与IE数据绑定事件一同使用，例如ondatasetcomplete。它返回一个字符串以表明数据源成员，此后可以将这个字符串作为一个参数以访问数据源中一个已命名的记录集。请查阅微软文档中与数据源对象（Data Source Object）相关的信息，以便了解它是否提供限定数据。

值 字符串。

默认值 空字符串。

rangeOffset

IE n/a NN n/a Moz all Saf n/a Op n/a DOM n/a

只读

返回节点内字符的偏移量，Mozilla DOM将该节点视为一个潜在的范围域端点。在与之相关的event对象的rangeParent属性中可以找到这个节点的引用。可以将这两个值作为参数传递至W3C DOM中Range对象的方法，以便设置它的起始或结束端点。因此mousedown事件的监听程序能够建立一个范围的起始点，而mouseup事件监听程序方法则可以设置其终点，这两个方法就可以为Range对象的方法提供rangeParent及rangeOffset属性值。

示例

```
var rng;
function processMouseDown(evt) {
   rng = document.createRange();
   rng.setStart(evt.rangeParent, evt.rangeOffset);
}
```

值 整数值。

默认值 0

rangeParent

IE n/a NN n/a Moz all Saf n/a Op n/a DOM n/a

只读

返回一个文档树节点的引用，这个节点是W3C文本域的一个合适起点或终点。它通常配合rangeOffset属性一同使用。

示例

```
function processMouseUp(evt) {
   rng.setEnd(evt.rangeParent, evt.rangeOffset);
}
```

值 指向节点的一个引用。

默认值 无。

reason

IE *4* NN *n/a* Moz *n/a* Saf *n/a* Op *n/a* DOM *n/a*

可读/可写

返回一个与ondatasetcomplete事件相关的代码，它指明了IE数据绑定时的数据传输是否成功，如果未完成传输，则指明传输是因一个错误而停止还是被客户端或用户中止。在处理ondatasetcomplete事件的事件处理程序中，必须检查这个属性。在IE 4中，这是一个只读属性。尽管IE 5/Mac在event对象中包含了这个属性，但它并未实现相关事件。

示例

```
if (evt.reason == 2) {
   alert("An error occurred during the most recent update.");
}
```

值 以下3个可能值之一：

0	传输成功
1	传输中止
2	传输因错误而停止

默认值 无。

recordset

IE *5(Win)* NN *n/a* Moz *n/a* Saf *n/a* Op *n/a* DOM *n/a*

可读/可写

返回一个IE数据绑定记录集对象的引用。

值 对象引用。

默认值 无。

relatedNode

IE *n/a* NN *n/a* Moz *all* Saf *all* Op *7* DOM *2*

只读

返回一个节点的引用，激发某些W3C DOM转变事件的动作会影响这个节点。这样就为引用受到事件影响的节点提供了一条更为直接的途径，它们的对应关系如下表所示：

转变事件的类型	事件对象.相关节点引用
DOMNodeInserted	被插入节点的父节点
DOMNodeRemoved	已删除节点的原父节点
DOMAttrModified	Attr 节点

对于其他类型的转变事件，此属性将返回null，而对其他的事件类，则会返回undefined。

示例 `var newParent = evt.relatedNode;`

值 节点引用，null或undefined。

默认值 无。

relatedTarget

IE *n/a* NN *n/a* Moz *all* Saf *all* Op *7* DOM *2*

只读

返回文档树中一个已显示的节点的引用，根据事件类型，该节点对应事件的前一个或下一个目标。对一个mouseover事件类型而言，relatedTarget属性指出鼠标指针来自于哪个节点，而在mouseout事件中，relatedTarget属性则针对鼠标指针将要前往的节点。而在IE的事件对象中，fromElement和toElement属性拥有与之相同的功能。

示例 `var beenThere = evt.relatedTarget;`

值 指向节点的一个引用。

默认值 无。

repeat

IE *5(Win)* NN *n/a* Moz *n/a* Saf *n/a* Op *n/a* DOM *n/a*

可读/可写

此属性只适用于onkeydown事件，当按键按下的时间长度足够进入自动重复模式时，返回布尔值true。使用下面这段示例代码可以防止自动重复的字符输入某个文本域中。

示例

```
function handleKeyDown() {
    if (evt.repeat) {
        evt.returnValue = false;
    }
}
```

值 布尔值：true | false。

默认值 false

returnValue

IE *4* NN *n/a* Moz *n/a* Saf *1.2* Op *7* DOM *n/a*

可读/可写

此属性为事件来源元素提供一个返回值，以便运行或禁止与该事件有关的元素默认动作。如果将event.returnValue设置为false，那么元素就不会实现其正常操作，例如导航到一个链接或提交表单。但这个值并不会影响事件处理程序的实际返回值。

示例 `evt.returnValue = false;`

值 布尔值：true | false，或者undefined。

默认值 undefined

screenX, screenY

IE *4* NN *4* Moz *all* Saf *all* Op *7* DOM *2*

只读

提供事件发生时鼠标指针在屏幕上的水平及竖直坐标位置。此时屏幕的左上角坐标点为（0,0）。除非已经对浏览器的窗口或文档位置进行了设置，并且能够知晓当前窗口区域在屏幕中的相对位置，否则无法确定浏览器内窗口或文档的确切坐标。

示例

```
if (evt.screenX < 5 || evt.screenY < 5) {
    alert("You\'re too close to the edge!");
}
```

值 任何大于或等于0的整数。

默认值 0

shiftKey

IE 4 NN n/a Moz all Saf all Op 7 DOM 2

只读

当事件激发时，如果按下了左侧或右侧的Shift键，那么返回true。

示例

```
if (evt.shiftKey) {
    // 处理 shift 键被按下的情况
}
```

值 布尔值：true | false。

默认值 false

shiftLeft

IE 5.5(Win) NN n/a Moz n/a Saf n/a Op n/a DOM n/a

只读

当事件激发时，如果按下了左侧的Shift键，那么返回true。

示例

```
if (evt.shiftLeft) {
    // 处理左侧 shift 键被按下的情况
}
```

值 布尔值：true | false。

默认值 false

srcElement

IE 4 NN n/a Moz n/a Saf 1.2 Op n/a DOM n/a

只读

指向接收到当前事件的第一个元素对象。如果须要使用一个事件处理方法来为许多不同的元素处理同一个事件类型，那么这个属性可以很方便地来转换处理结构。W3C DOM中的等价属性是target。

示例

```
switch (evt.srcElement.id) {
    case "myDIV":
        ...
    ...
}
```

值 元素对象的引用。

默认值 无。

srcFilter

IE 4(Win) NN n/a Moz n/a Saf n/a Op n/a DOM n/a

只读

指向一个触发onfilterchange事件的滤镜对象。

值 滤镜对象引用。

默认值 无。

srcUrn

IE 5(Win) NN n/a Moz n/a Saf n/a Op n/a DOM n/a

只读

触发事件的附加行为的URN字符串。

值 字符串。

默认值 null

target

IE *n/a* NN *n/a* Moz *all* Saf *all* Op *7* DOM *2*

只读

指向当前事件的预期目标对象。在早期的Mozilla浏览器，以及Safari 2.02之前，即使是为围绕在文本节点四周的元素定义事件处理程序，target属性也能够指向该文本节点。在这种情况下进行事件处理时，必须将nodeType考虑在内，以便获取IE和Navigator中环绕元素的引用。但从Mozilla 1.4开始，其event对象会自动重定向到文本节点的容器元素。因此如果须要在Mozilla中获得接收事件的文本节点的引用，请使用originalTarget或explicitOriginalTarget。

示例 `var elem = (evt.target) ? evt.target : evt.srcElement;`
值 节点对象引用。
默认值 无。

timeStamp

IE *n/a* NN *n/a* Moz *all* Saf *all* Op *7* DOM *2*

只读

提供一个表示毫秒数的整数值，可以使用它作为事件发生时刻的相对指示器。尽管W3C DOM建议使用从1970年1月1日至今的毫秒数，但在实际使用中请不要完全相信这一设定。当然，也可以通过比较两个事件的timeStamp属性来获取它们之间的时间间隔。

示例 `var clickTime = evt.timeStamp;`
值 整数值。
默认值 当前时间戳。

toElement

IE *4* NN *n/a* Moz *n/a* Saf *n/a* Op *n/a* DOM *n/a*

只读

返回一个元素对象的引用，由于鼠标指针移动到该元素之上从而触发了onmouseout事件。

示例

```
if (evt.toElement.id == "upperLevel") {
    ...
}
```

值 元素对象的引用。
默认值 无。

type

IE *4* NN *4* Moz *all* Saf *all* Op *7* DOM *2*

只读

指明当前事件的类型，其中不包含前缀“on”。注意，所有的属性值均为小写形式。

示例

```
if (evt.type == "change") {
    ...
}
```

值 事件名称字符串，其中不包含前缀“on”。
默认值 无。

view

IE *n/a* NN *n/a* Moz *all* Saf *all* Op *7* DOM *2*

只读

返回事件发生时W3C DOM视图的引用，例如，Mozilla实现中的window或frame对象，以及Safari和Opera中的document.defaultView。

示例 `var whichWin = evt.view;`
值 对象窗口类型的引用。
默认值 当前窗口。

wheelDelta

IE *5.5* NN *n/a* Moz *n/a* Saf *n/a* Op *n/a* DOM *n/a*

只读

当鼠标配备了一个滚轮时，它会返回一个整数值，以表示在`onmousewheel`事件发生过程中用户滚动鼠标滚轮的方向。正整数表示用户朝着窗口的方向滚动滚轮，而负值则表示与之相反的方向。

示例

```
if (evt.wheelDelta > 0) {
    ...
}
```

值 整数，通常是`120`或`-120`。
默认值 无。

which

IE *n/a* NN *4* Moz *all* Saf *all* Op *7* DOM *n/a*

只读

返回一个与事件类型相关的值。对于鼠标事件而言，这个属性值是一个整数，它表明刚刚使用了哪个鼠标按键，`1`表示左按键，`3`则表示右按键。而在键盘事件中，这个属性值表示键盘字符的ASCII码。Mozilla和其他地方还残留着这个属性，以作为Navigator 4事件模型的一种保留。如果不再须要支持Navigator 4，那么请使用`button`、`charCode`和`keyCode`属性来代替这个属性。

示例

```
if (evt.which == 65) {
    ...
}
```

值 整数值。
默认值 无。

x, y

IE *4* NN *n/a* Moz *n/a* Saf *all* Op *7* DOM *n/a*

只读

返回事件发生时鼠标指针所在的水平和垂直坐标位置，单位为像素。所有的相对位置元素均会以`body`元素作为坐标系（在标准兼容模式下，IE 6及之后的版本以`html`元素作为坐标系）。如果事件发生在一个相对位置元素的矩形框内，那么坐标系就限制在该元素的空间之内，这时该元素的左上角坐标就变为（0,0）。如果鼠标指针此时位于浏览器窗口的文档区域之外，那么将返回`-1`。

示例

```
if (evt.x < 20 && evt.y < 30) {
    ...
}
```

值 整数。
默认值 无。

getPreventDefault( )

IE *n/a* NN *n/a* Moz *all* Saf *n/a* Op *n/a* DOM *n/a*

如果已经为当前事件对象调用了`preventDefault()`方法，那么将返回布尔值`true`。实质上这可以让脚本查询“禁用默认”的状态。这个方法是Mozilla针对W3C DOM事件模型的扩展。

返回值 布尔值：true | false。
参数 无。

initEvent( )

IE *n/a* NN *n/a* Moz *all* Saf *all* Op *7* DOM *2*

```
initEvent("eventType", bubblesFlag, cancelableFlag)
```

在document.createEvent()方法生成的事件对象上完成最小的必要初始化工作。初始化由脚本生成的事件之后，该事件可以作为一个参数传递给节点的dispatchEvent()方法。

返回值 无。
参数

eventType
事件类型的字符串标识符，例如“click”、“mousedown”、“keypress”和“DOMAttrModified”等。应该限制输入的类型，以便它们能够适用于对应的事件类型分类，例如，为document.createEvent("MouseEvent")使用“click”事件类型。

bubblesFlag
布尔值（true | 或false），用以确定事件的默认传播方式是否为冒泡方式，即向上传播。

cancelableFlag
布尔值（true | false），用以确定能否通过preventDefault()方法禁用事件的默认动作。

initKeyEvent( )

IE *n/a* NN *n/a* Moz *all* Saf *all* Op *n/a* DOM *3*

```
initKeyEvent("eventType", bubblesFlag, cancelableFlag, view, ctrlKeyFlag, altKeyFlag,
shiftKeyFlag, metaKeyFlag, keyCode, charCode)
```

使用一个与键盘事件相关的完整属性值集合来初始化一个新创建的事件对象。由于W3C DOM并未采用键盘事件，因此Mozilla实现了其独有的键盘事件初始化方法。现在Safari也采用了这一方法。必须在这个方法中输入所有的参数，而且如果某些值对事件类型并不是特别重要，那么必须将它们设置为默认值，例如，将布尔型的键标识设置为false，或者将keyCode设置为0。

返回值 无。
参数

eventType
事件类型的字符串标识符：keydown、keypress、keyup。

bubblesFlag
布尔值（true | false），用以确定事件的默认传播方式是否为冒泡方式，即向上传播。

cancelableFlag
布尔值（true | false），用以确定能否通过preventDefault()方法禁用事件的默认动作。

view
窗口、框架对象（Mozilla）或document.defaultView（Safari）的引用，事件应该能发生在这个视图中。

ctrlKeyFlag
布尔值（true | false），表示此事件的Control键状态。

altKeyFlag
布尔值（true | false），表示此事件的Alt键状态。

shiftKeyFlag
布尔值（true | false），表示此事件的Shift键状态。

metaKeyFlag
布尔值（true | false），表示此事件的Meta键（例如，Macintosh的Command键）状态。

keyCode
整数，表示这一事件的键代码。

charCode

整数，表示这一事件的字符代码。

initKeyboardEvent( )

IE *n/a* NN *n/a* Moz *n/a* Saf *n/a* Op *n/a* DOM *3*

```
initKeyEvent("eventType", bubblesFlag, cancelableFlag, view, keyIdentifier, keyLocation,
modifierList)
```

使用一个与键盘事件相关的完整属性值集合来初始化一个新创建的事件对象。由于目前W3C DOM Level 3的事件模型尚未定稿，因此根据其写法不同，这个方法的参数形式也会发生变化。必须在这个方法中输入所有的参数，而且如果某些值对事件类型并不是特别重要，那么必须将它们设置为默认值，例如，将布尔型的键标识设置为false，或者将keyCode设置为0。

返回值 无。

参数

eventType

事件类型的字符串标识符：keydown、keypress、keyup。

bubblesFlag

布尔值（true | false），用以确定事件的默认传播方式是否为冒泡方式，即向上传播。

cancelableFlag

布尔值（true | false），用以确定能否通过preventDefault()方法禁用事件的默认动作。

view

窗口、框架对象的引用，事件应该能发生在这个视图中。

keyIdentifier

字符串值，用以表示事件模拟了哪个键。这些值来自于常量字符串列表（例如，“F1”、“Home”），以及Unicode字符引用列表（例如，U+0041）完整列表可以从这个链接中获取：*http://www.w3.org/TR/DOM-Level-3-Events/events.html#Events-KeyboardEvent- keyIdentifier*。

keyLocation

整数，表示这个事件模拟键盘的哪一部分：0（标准）、1（左侧）、2（右侧），以及（数字小键盘）。

modifierList

由组合键组成的字符串，键之间使用空格进行分隔，在事件过程中会对这些按键进行模拟。常用值包括：Alt、AltGraph、CapsLock、Control、Meta、NumLock、Scroll及Shift。

initMouseEvent( )

IE *n/a* NN *n/a* Moz *all* Saf *all* Op *7* DOM *2*

```
initMouseEvent("eventType", bubblesFlag, cancelableFlag, view, detailVal, screenX, screenY,
clientX, clientY, ctrlKeyFlag, altKeyFlag, shiftKeyFlag, metaKeyFlag, buttonCode,
relatedTargetNodeRef)
```

使用一个与鼠标事件相关的完整属性值集合来初始化一个新创建的事件对象。必须在这个方法中输入所有的参数，而且如果某些值对事件类型并不是特别重要，那么必须将它们设置为默认值，例如，将布尔型的键标识设置为false，将整数值设置为0或将节点引用设置为null。

返回值 无。

参数

eventType

事件类型的字符串标识符，例如click、mousedown、mousemove、mouseout、mouseover和mouseup等。

bubblesFlag

布尔值（true | false），用以确定事件的默认传播方式是否为冒泡方式，即向上传播。

cancelableFlag

布尔值（true | false），用以确定能否通过preventDefault()方法禁用事件的默认动作。

view
窗口、框架对象的引用，事件应该能发生在这个视图中。
detailVal
整数代码，以表示与事件相关的详细数据。
screenX
整数，表示水平屏幕坐标。
screenY
554 整数，表示竖直屏幕坐标。
clientX
整数，表示水平浏览器窗口坐标。
clientY
整数，表示竖直浏览器窗口坐标。
ctrlKeyFlag
布尔值（true | false），表示此事件的Control键状态。
altKeyFlag
布尔值（true | false），表示此事件的Alt键状态。
shiftKeyFlag
布尔值（true | false），表示此事件的Shift键状态。
metaKeyFlag
布尔值（true | false），表示此事件的Meta键（例如，Macintosh的Command键）状态。
buttonCode
整数，表示这一事件的按钮代码。
relatedTargetNodeRef
节点引用，该节点接收到了前一个或下一个鼠标事件。

initMutationEvent( )

IE *n/a* NN *n/a* Moz *all* Saf *all* Op *8* DOM *2*

```
initMutationEvent("eventType", bubblesFlag, cancelableFlag, relatedNodeRef, prevValue,
newValue, attrName, attrChangeCode)
```

使用一个与转变事件相关的完整属性值集合来初始化一个新创建的事件对象。在这个方法中，必须输入所有的参数，而且如果某些值对事件类型并不是特别重要，那么必须将它们设置为默认值，例如，将布尔型的键标识设置为false，或将keyCode设置为0。

返回值 无。

参数

eventType
事件类型的字符串标识符：DOMAttrModified、DOMCharacterDataModified、DOMNodeInserted、DOMNodeInsertedIntoDocument、DOMNodeRemoved、DOMNodeRemovedFromDocument、DOMSubtreeModified。
bubblesFlag
布尔值（true | false），用以确定事件的默认传播方式是否为冒泡方式，即向上传播。
cancelableFlag
布尔值（true | false），用以确定能否通过preventDefault()方法禁用事件的默认动作。
relatedNode
与事件相关的节点引用。仅适用于DOMNodeInserted、DOMNodeRemoved和DOMAttrModified这3种事件类型。
prevValue
555 字符串，表示Attr或CharacterData节点的前一值。仅适用于DOMAttrModified和DOMCharacterDataModified这两种事件类型。

newValue

字符串，表示Attr或CharacterData节点的新值。仅适用于DOMAttrModified和DOMCharacterDataModified这两种事件类型。

attrName

字符串，Attr节点的名称。仅适用于DOMAttrModified这种事件类型。

attrChangeCode

整数，对应于改变事件模拟的类型的代码。仅适用于DOMAttrModified这种事件类型。

initUIEvent( )

IE *n/a* NN *n/a* Moz *all* Saf *all* Op *7* DOM *2*

```
initUIEvent("eventType", bubblesFlag, cancelableFlag, view, detailVal)
```

使用一个与UI事件相关的完整属性值集合来初始化一个新创建的事件对象。必须在这个方法中输入所有的参数，而且如果某些值对事件类型并不是特别重要，那么必须将它们设置为默认值，例如，将布尔型的键标识设置为false，或将整数值设置为0。

返回值 无。

参数

eventType

表示事件类型的字符串标识符，例如DOMFocusIn、DOMFocusOut和DOMActivate。

bubblesFlag

布尔值（true | false），用以确定事件的默认传播方式是否为冒泡方式，即向上传播。

cancelableFlag

布尔值（true | false），用以确定能否通过preventDefault()方法禁用事件的默认动作。

view

窗口、框架对象的引用，事件应该能发生在这个视图中。

detailVal

整数代码，以表示与事件相关的详细数据。

preventDefault( )

IE *n/a* NN *n/a* Moz *all* Saf *all* Op *7* DOM *2*

它命令当前事件忽略对节点的正常操作。一旦进行了设置，那么就无法为当前事件恢复原有设置。下面这段代码中实现了一个W3C DOM事件监听方法，它监听keypress事件并且只允许在文本域中输入数字：

```
function numsOnly(evt) {
   if (evt.charCode < 48 || evt.charCode > 57) {
      evt.preventDefault();
   }
}
```

这个方法等价于将IE的event.returnValue属性设置为false，或者使用一个事件处理程序来进行判断并返回false。

返回值 无。

参数 无。

stopPropagation( )

IE *n/a* NN *n/a* Moz *all* Saf *all* Op *7* DOM *2*

防止当前处理节点之外其他节点的捕获事件或当前事件在上传层次中继续传播。针对向上传播的情况，将事件对象的cancelBubble属性设置为false与这个方法的效果相同。

返回值 无。

参数 无。

Event

IE *4* NN *|4|* Moz *all* Saf *all* Op *n/a* DOM *n/a*

Event对象是一个静态对象，它包含大量区分大小写的常量值，可以使用它们测试用户或系统生成的事件的键盘修改类型及事件类型，请参见event对象的modifiers和type属性。而这些常量值与对应的整数值相等。为兼容Navigator 4的事件捕获语法，Mozilla和Safari一直都支持这个对象。

对象模型引用方式 Event

对象特定属性 ABORT、ALT_MASK、BACK、BLUR、CHANGE、CLICK、CONTROL_MASK、DBLCLICK、DRAGDROP、ERROR、FOCUS、FORWARD、HELP、KEYDOWN、KEYPRESS、KEYUP、LOAD、LOCATE、META_MASK、MOUSEDOWN、MOUSEDRAG、MOUSEMOVE、MOUSEOUT、MOUSEOVER、MOUSEUP、MOVE、RESET、RESIZE、SCROLL、SELECT、SHIFT_MASK、SUBMIT、UNLOAD、XFER_DONE

Event

IE *n/a* NN *n/a* Moz *all* Saf *all* Op *8* DOM *2*

W3C DOM的Event是一个抽象对象，它包含每个W3C DOM事件实例所共享的属性和方法。这种对象类型也是由document.createEvent()方法创建的一种通用事件对象。请参见本章前文中有关event对象的讨论，以了解支持这一对象的属性和方法，以及其他事件类型如何继承这些属性与方法。

对象模型引用方式 Event

对象特定属性 AT_TARGET、bubbles、BUBBLING_PHASE、cancelable、CAPTURING_PHASE、currentTarget、eventPhase、target、timeStamp、type

对象特定方法 initEvent()、preventDefault()、stopPropagation()

EventListener

IE *n/a* NN *n/a* Moz *all* Saf *all* Op *7* DOM *2*

W3C DOM的EventListener对象只不过是一个脚本方法的引用，节点可以调用这个方法以响应一个事件。通过这一对象，可以方便地为nodeObject.addEventListener()和nodeObject.removeEventListener()方法所需的第2个参数指定数据类型，本章前文中已进行了相关的说明。

EventTarget

IE *n/a* NN *n/a* Moz *all* Saf *all* Op *7* DOM *2*

W3C DOM的EventTarget对象是事件模型与实际接收事件的节点之间的联系纽带。由于所有的节点对象（尤其是HTML文档树中的文本及元素对象）均实现了EventTarget对象，因此也为那些节点赋予了原本属于EventTarget对象的3个方法：addEventListener()、dispatchEvent()和emoveEventListener()。换句话说，文档中的每个节点同样也是一个潜在的EventTarget对象。

external

IE *4(Win)* NN *n/a* Moz *n/a* Saf *n/a* Op *n/a* DOM *n/a*

如果开发者将IE作为应用系统的一部分并且须要访问文档对象模型的自定义扩展，那么他们通常会使用external对象。但是可能还须要为IE/Windows用户在应用系统中添加一个或两个对象方法。

下面这段代码示例请求用户将一个脚本控制的书签加入浏览器的收藏夹中：

```
external.AddFavorite("URL", "Favorites List Label");
```

而这段代码则演示了如何将另一个URL加载至目标窗口或框架，然后为第3个参数字符串执行一个文本查询工作：

```
external.NavigateAndFind("URL", "searchString", "targetFrameName");
```

如须获取更多详细信息，请访问如下链接：*http://msdn.microsoft.com/en-us/library/ms536641 (VS.85).aspx*。

fieldset

IE 4 NN n/a Moz all Saf all Op 7 DOM 1

fieldset对象对应于fieldset元素。IE 5/Mac为此对象实现了客户端和滚动相关的属性集。

等价HTML元素 <fieldset>
对象模型引用方式 [window.]document.getElementById("*elementID*")
对象特定属性 align、form
对象特定方法 无。
对象特定事件 无。

align

IE 4 NN n/a Moz n/a Saf n/a Op n/a DOM n/a

可读/可写

定义元素在其容器内的水平对齐方式。实际使用时，在IE 4中这个属性对fieldset对象影响很小。而它在IE 6/Windows中的行为并不确定，只有在IE 5/Macintosh中，它的反应才与预期相同。

示例 document.getElementById("myFieldset").align = "center";
值 以下3个对齐常量之一：center | left | right。
默认值 left

form

IE 5(Mac)/6(Win) NN n/a Moz all Saf all Op 7 DOM 1

只读

返回文档树中临近最外层form元素对象的一个引用。同一个form元素中的多个fieldset元素对象将引用相同的form元素对象。

示例 var theForm = document.getElementById("myFieldset").form;
值 form元素对象的引用。
默认值 无。

fileUpload

请参见input。

filters[]

IE 4(Win) NN n/a Moz n/a Saf n/a Op n/a DOM n/a

提供与当前元素相关的所有滤镜的集合。Internet Explorer浏览器允许使用数组符号来访问集合中的一个元素。但在IE/Macintosh中无法使用滤镜。

对象模型引用方式 document.getElementById("*elementID*").filters
对象特定属性 length
对象特定方法 item()、namedItem()

length

IE 4(Win) NN n/a Moz n/a Saf n/a Op n/a DOM n/a

只读

返回集合中过滤器的数量。

示例 var howMany = document.body.filters.length;
值 整数值。

item( )

IE 4(Win) NN n/a Moz n/a Saf n/a Op n/a DOM n/a

```
item(index)
```

返回一个过滤器对象，它对应的源代码次序可与参数传入的索引值相匹配。

返回值 过滤器对象的引用。如果并不存在与参数相匹配的对象，则返回null。

参数

index

一个以零为基的整数，依照源代码次序，对应于集合中的某个特定项。

namedItem( )

IE 4(Win) NN n/a Moz n/a Saf n/a Op n/a DOM n/a

```
namedItem(IDOrName)
```

返回滤镜对象或对象集合，它们对应于符合参数字符串值的所有滤镜。

返回值 一个滤镜对象或一个滤镜对象集合。如果并不存在与参数相匹配的对象，则返回null。

参数

IDOrName

包含目标滤镜名称的字符串。

font

IE 4 NN n/a Moz all Saf all Op 7 DOM 1

font对象对应于font元素。

等价HTML元素 <font>

对象模型引用方式 [window.]document.getElementById("*elementID*")

对象特定属性 color、face、size

对象特定方法 无。

对象特定事件 无。

color

IE 4 NN n/a Moz all Saf all Op 7 DOM 1

可读/可写

为font元素包含的所有文本设置字体颜色。

示例

```
document.getElementById("myFont").color = "red";
```

值 一个十六进制的三元组或明语颜色名称字符串，均不区分大小写。目前可用的明语颜色名称请参见附录A。

默认值 由浏览器决定默认值。

face

IE 4 NN n/a Moz all Saf all Op 7 DOM 1

可读/可写

为当前font对象包围的内容提供一个多层次的可用字体。这一组字体名称使用逗号进行分割，浏览器从它们中的第一个开始，逐个与客户端系统中的字体进行比对，以设置默认字体，如果最终仍然找不到对应字体就使用浏览器的默认字体。比对时face中的字体名称必须与系统中的字体名称完全匹配。

示例

```
document.getElementById("myFont").face = "Bookman, Times Roman, serif";
```

值 包含一个或多个字体名称的字符串，字体间使用逗号进行分隔。可以使用真实的字体名称或可识别的通用字体：serif | sans-serif | monospace。

默认值 由浏览器决定默认值。

size

IE 4 NN n/a Moz all Saf all Op 7 DOM 1

可读/可写

提供字体大小，其范围介于浏览器定义的相对尺寸级别（1~7）之中。如果要设定更精确的字体尺寸，请参见本章下文中的style.fontSize属性。

示例 `document.getElementById("fontSpec2").size = "+1";`

值 既可以是一个整数（使用引号括起来一个字符串），也可以是一个由“+”或“-”和整数组成的相对数值（必须用引号括起来）。

默认值 3

fonts

有关这个IE/Windows对象的详细内容，请参见本章前文中“Dialog Helper”对象的fonts属性。

form

IE 3 NN 2 Moz all Saf all Op 7 DOM 1

form对象对应于form元素。在所有的脚本浏览器中，均可以通过表单对象的name属性值或文档所包含的表单数组的索引来获取form对象的引用。在支持id属性的浏览器（包括IE 4和后续版本，以及W3C DOM浏览器）中，还可以使用元素的ID来引用元素对象。为了获取内嵌表单控件对象（如input和textare元素对象）的引用，可以将form对象作为引用的一部分来反复使用后向兼容引用（如document.formName.controlName）。在更现代的浏览器中，还可以通过其ID来直接引用这些控件元素。

等价HTML元素 <form>

对象模型引用方式

```
[window.]document.formName
[window.]document.forms[i]
[window.]document.form["formName"]
[window.]document.getElementById("elementID")
```

对象特定属性 accept、acceptCharset、action、autocomplete、data、elements[]、encoding、enctype、length、method、name、replace、target、templateElements

对象特定方法 checkValidity()、dispatchFormChange()、dispatchFormInput()、handleEvent()、reset()、resetFromData()、submit()

对象特定事件

事件	IE	Mozilla	Safari	Opera	W3C DOM
formchange[a]	—	—	—	•	—
forminput[a]	—	—	—	•	—
reset	•	•	•	•	•
submit	•	•	•	•	•

a 实现在 Opera 9 中的 Web Forms 2.0 事件。

accept

IE n/a NN n/a Moz n/a Saf n/a Op 9 DOM n/a

可读/可写

控制当前表格中文件输入元素可以上传的文件类型。

示例 `document.entryForm.accept= "image/gif, image/jpeg";`

值 MIME类型字符串。如果使用多个MIME类型，应该使用逗号进行分隔。
默认值 空字符串。

acceptCharset

IE *5* NN *n/a* Moz *all* Saf *all* Op *7* DOM *1*

可读/可写

客户端表单的字符集信息将通过acceptcharset属性告知服务器，以便服务器为解析表单信息做准备。

示例
```
document.entryForm.acceptCharset= "it, es";
```
值 来自于字符注册表的不区分大小写的字符串，请访问以下链接获取更多详细信息：*http://www.iana.org/assignments/character-sets*。可以使用逗号分隔多个字符集。
默认值 空字符串，但在IE/Windows中则为“UNKNOWN”。

action

IE *3* NN *2* Moz *all* Saf *all* Op *7* DOM *1*

可读/可写

指定表单提交时须要访问的URL。通过脚本对这个属性的控制，可以根据用户对表单剩余部分的操作让表单提交到不同的服务器处理进程。

示例
```
document.entryForm.action = "http://www.megacorp.com/cgi-bin/altEntry";
```
值 完整或相对URL路径。
默认值 无。

autocomplete

IE *5(Win)* NN *n/a* Moz *n/a* Saf *n/a* Op *n/a* DOM *n/a*

可读/可写

当IE用户设置了表单自动完成选项，并且页面作者也设置了表单元素的自动完成属性时，这个属性可以控制是否启用自动完成功能。关于HTML表单中如何实现自动完成的更多细节，请访问此链接：*http://msdn.microsoft.com/en-us/library/ms533032(VS.85).aspx*。

示例
```
document.entryForm.autocomplete = "off";
```
值 字符串：on | off。
默认值 无。

data

IE *n/a* NN *n/a* Moz *n/a* Saf *n/a* Op *9* DOM *n/a*

可读/可写

这个Web Forms 2.0扩展允许表单从一个外部XML文件中获取表单控件的初始值。Web Forms 2.0规范为使用XML文件提供了详细的结构和命名空间说明。如须获取更多信息请访问此链接：*http://www.whatwg.org*。

示例
```
document.entryForm.autocomplete = "off";
```
值 URL字符串。
默认值 空字符串。

elements[]

IE *3* NN *2* Moz *all* Saf *all* Op *7* DOM *1*

只读

使用此属性可以返回由当前表单拥有的所有表单控件对象组成的数组。

示例
```
for (var i = 0; i < document.entryForm.elements.length; i++) {
    if (document.entryForm.elements[i].type == "text") {
        document.entryForm.elements[i].value = "";
```

```
    }
}
```

值 元素对象引用数组。

默认值 长度为0的数组。

encoding

IE 3 NN 2 Moz all Saf n/a Op 7 DOM n/a

可读/可写

指明了随同表单一起提交至服务器的数据的MIME类型。一般情况下（method属性值为“post”时），其默认值就是当前内容的类型。但如果要使用脚本改变表单的action属性，那么就要考虑是否须要为达到这一目的设定一个自定义的编码方式。请参见encType属性。

示例
```
document.orderForm.encoding = "text/plain";
```

值 不区分大小写的MIME类型（内容类型字符串）。如果使用多种类型，请在字符串中使用逗号对类型进行分隔。

默认值 在IE和Opera中为“application/x-www-form-urlencoded”，而在Mozilla中则为空字符串。

enctype

IE 5(Mac)/6(Win) NN n/a Moz all Saf all Op 7 DOM 1

可读/可写

为早期DOM实现的encoding属性提供W3C DOM属性名称。目前的浏览器同时支持这两种属性名称。请参见encoding属性。

示例
```
document.orderForm.enctype = "text/plain";
```

值 不区分大小写的MIME类型（内容类型字符串）。如果使用多种类型，请在字符串中使用逗号对类型进行分隔。

默认值 在IE和Opera中为“application/x-www-form-urlencoded”。而在Mozilla和Safari中则为空字符串。

564

length

IE 3 NN 2 Moz all Saf all Op 7 DOM 1

只读

指明表单中表单控件元素的数量。可以使用这个属性代替表单内元素数组的长度。

示例
```
for (var i = 0; i < document.forms[0].length; i++)
    ...
}
```

值 整数值。

默认值 0

method

IE 3 NN 2 Moz all Saf all Op 7 DOM 1

可读/可写

可以通过“get”和“post”这两种HTTP方法提交表单。这两种方法决定了表单元素的数据提交方式，使用“get”方法时，这些数据会附加在action属性指定的URL之后，而使用“post”方法时，则会作为事务消息的正文进行发送。事实上，如果在form元素中并未设置action和method属性，那么表单提交时会直接重载同一个文档，并将表单控件都恢复到初始值。

565

示例
```
document.entryForm.method = "post";
```

值 可使用以下两个常量值之一：get | post。

默认值 get

name

IE 3 NN 2 Moz all Saf all Op 7 DOM 1

可读/可写

这是一个与表单相关的标识符。虽然这个信息并不会随着表单一同提交，但表单的名称可以用于引用表单及内嵌的表单元素。虽然新标准倾向于使用id属性，但很多浏览器依然须要为表单设置name属性，以便允许表单被正常提交。

示例 `var firstFormName = document.forms[0].name;`

值 区分大小写的标识符，必须遵循标识符命名规则：不包含空白、不能以数字作为起始字符、除下划线外不允许使用其他标点符号。

默认值 无。

replace

IE n/a NN n/a Moz n/a Saf n/a Op 9 DOM n/a

可读/可写

这个Web Forms 2.0扩展将一些指令与表单提交控件联系在一起，用以在表单提交后处理服务器返回数据的相关指令。目前有两种处理方式，默认方式是使用服务器的响应数据替换原文档。如果已为form元素的data属性指定了一个URL，那么浏览器会将返回的数值填充到表单中，而不是重新载入表单的初始化数据，这是第二种处理方式。

示例 `document.getElementById("myform").replace = "values";`

值 两个常量值之一：document | values。

默认值 document

target

IE 3 NN 2 Moz all Saf all Op 7 DOM 1

可读/可写

窗口或框架的名称，该对象将接收表单提交后服务器返回的内容。这些名称会通过frame元素的name属性分配给对应框架，对子窗口而言，则会通过window.open()方法的第2个参数来指定其名称。由于严格的HTML或XHTML校验程序并未承认对应的target属性，因此可以省去这个属性以便通过校验，但是在页面加载之后或在表单提交时须要使用脚本为表单的target属性指定属性值以便将表单结果定向到另一个窗口。

示例 `document.getElementById("myForm").target = "_blank";`

值 窗口或框架的名称字符串，或下述字符串常量之一：_parent、_self、_top、_blank。_parent以当前文档所在的框架集作为目标；_self以当前窗口作为目标；_top以浏览器主窗口为目标，因此会删除所有的其他框架；而_blank（或其他未定义的标识符）会以默认尺寸创建一个新的窗口。

默认值 无（暗指当前窗口或框架）。

templateElements

IE n/a NN n/a Moz n/a Saf n/a Op 9 DOM n/a

只读

这个Web Forms 2.0扩展返回当前表单中表单控件元素的数组，这些控件元素进而组成了重复模板。

值 HTML元素对象引用组成的数组。

默认值 空数组。

checkValidity( )

IE n/a NN n/a Moz n/a Saf n/a Op 9 DOM n/a

这个Web Forms 2.0扩展会强制检查所有已设置动态校验的表单控件。如果根据控件说明，它们均能通过校验，那么就返回布尔值true。

返回值 布尔值：`true | false`。
参数 无。

dispatchFormChange( )

IE *n/a* NN *n/a* Moz *n/a* Saf *n/a* Op *9* DOM *n/a*

这个Web Forms 2.0扩展强制在表单内的所有控件上触发`formchange`事件，因此这也会触发每个元素上`formchange`事件的事件处理程序。

返回值 无。
参数 无。

dispatchFormInput( )

IE *n/a* NN *n/a* Moz *n/a* Saf *n/a* Op *9* DOM *n/a*

这个Web Forms 2.0扩展强制在表单内的所有控件上触发`forminput`事件，因此这也会触发每个元素中`forminput`事件的事件处理程序。

返回值 无。
参数 无。

handleEvent( )

IE *n/a* NN *|4|* Moz *n/a* Saf *n/a* Op *n/a* DOM *n/a*

`handleEvent(event)`

命令对象接受并处理一个事件，该事件的描述将作为参数传入此方法。此时`document`对象必须拥有一个事件处理程序来处理这种类型的事件。

返回值 无。
参数

`event`
一个Navigator 4的`event`对象。

reset( )

IE *4* NN *3* Moz *all* Saf *all* Op *7* DOM *1*

这一方法与点击“重置”类型的输入元素的效果一样。调用这个方法后，所有表单控件会回复到其默认值。

返回值 无。
参数 无。

resetFromData( )

IE *n/a* NN *n/a* Moz *n/a* Saf *n/a* Op *9* DOM *n/a*

`resetFromData(`*`XMLDocument`*`)`

根据作为参数传入的XML文档对象中的说明，这个Web Forms 2.0扩展为当前表单中的控件元素设置对应的初始值。在Web Forms 2.0规范中包含这种文档类型的特定结构。如须获取更多的信息请访问此链接：*http://www.whatwg.org*。

返回值 无。
参数

`XMLDocument`
同一结构中的一个XML文档对象，它将从`data`属性的URL中加载外部文件。

submit( )

IE *4* NN *3* Moz *all* Saf *all* Op *7* DOM *1*

这一方法与点击“提交”类型的提交元素的效果一样。但是，这个方法并不会触发`onSubmit`事件处理程序。

返回值 无。
参数 无。

forms

568

IE 3 NN 2 Moz all Saf all Op 7 DOM 1

文档中所有form对象组成的集合。

对象模型引用方式 document.forms
对象特定属性 length
对象特定方法 item()、namedItem()、tags()、urns()
对象特定事件 无。

length

IE 3 NN 2 Moz all Saf all Op 7 DOM 1

只读

返回集合中元素的数量。

示例
```
var howMany = document.forms.length;
```
值 整数值。

item( )

IE 4 NN n/a Moz all Saf all Op 7 DOM 1

```
item(index[, subindex]) item(index)
```

根据符合索引值（或IE/Windows中的任意索引及子索引值）的元素对象，返回一个单独的对象或一组对象集合。

返回值 一个对象或一个对象集合。如果并不存在与参数相匹配的对象，则返回null。

参数

index

当输入参数是一个从零开始计数的整数（根据W3C规范的需要）时，返回值是源代码内与某个具体项相对应的独立元素（内嵌在当前元素内）。如果参数为一个字符串（IE/Windows允许这种使用方式），则返回id或name属性与该字符串相符的元素组成的集合。

subindex

如果在IE/Windows中为第1个参数指定一个字符串值，那么可以将第2个参数指定为一个从零开始计数的索引值，这样就可以从集合中找到一个id或name属性与第1个参数的字符串相匹配的具体元素。

namedItem( )

569

IE 6 NN n/a Moz all Saf all Op 7 DOM 1

```
namedItem(IDOrName)
```

返回单个W3C DOM对象或对象集合，它们对应于符合参数字符串值的元素。

返回值 一个W3C DOM对象或对象集合。如果并不存在与参数相匹配的对象，则返回null。

参数

IDOrName

与目标元素的id或name属性包含相同值的字符串。

tags( )

IE 4 NN n/a Moz n/a Saf n/a Op n/a DOM n/a

```
tags(tagName)
```

从当前集合内的全部对象中返回标签与tagName参数相匹配的对象集合。此时由于所有的元素均拥有相同的“form”标签，因此这是一个完全多余的方法。

返回值 对象集合。如果与参数相匹配的对象不存在，则返回一个长度为零的数组。

参数

tagName

包含元素标签的一个字符串，例如document.forms.tags("form")。

urns( )

IE *5(Win)* NN *n/a* Moz *n/a* Saf *n/a* Op *n/a* DOM *n/a*

```
urns(URN)
```

返回内嵌元素对象的集合，这些对象附加了一些行为，并且其URN符合输入的*URN*参数。

返回值 对象集合。如果与参数相匹配的对象不存在，则返回一个长度为零的数组。

参数

URN

一个包含本地或外部行为文件的统一资源命名的字符串。

frame

IE *4* NN *n/a* Moz *all* Saf *all* Op *7* DOM *1*

frame对象对应于frame元素，它只能在一个frameset元素中生成。此时要明确区分frame元素对象与window对象之间区别。frame元素对象的属性和方法有助于反映HTML元素及其属性相关的各个方面。框架中的内容是一个窗口（W3C DOM术语则称之为视图），可以对它进行脚本操作，而其内部实质上包含一个文档。即使为frame元素的id和name属性设置了相同的标识符，也可以通过其ID引用一个frame元素对象。例如，可以通过下面这段代码，从驻留在框架文档中的脚本访问frame元素对象：

```
parent.document.getElementById("TOCFrame")
```

但作为一个窗口，如果要获取同样的框架对象并访问其脚本和文档，那么须要在脚本中使用下面两种代码之一来得到所需的对象引用：

```
parent.TOCFrame
parent.frames["TOCFrame"]
```

当使用脚本处理frame元素对象的引用时，如果假定文档包含的脚本及目标框架的文档均保存在同一服务器的相同域中，那么可以通过下文中列出的contentDocument及contentWindow属性来建立元素对象及其内容之间的联系。但如果要访问另一个框架中的内容则会存在很多安全隐患。特别在IE浏览器中，由于开发者需要着重控制安全性而不是仅仅为脚本编写者提供便利，因此这种跨框架访问往往会导致一个“拒绝访问”的错误。

等价HTML元素 <frame>

对象模型引用方式 [windowRef.]document.getElementById("frameID")

对象特定属性 allowTransparency、borderColor、contentDocument、contentWindow、dataFld、dataSrc、frameBorder、height、longDesc、marginHeight、marginWidth、name、noResize、scrolling、src、width

对象特定方法 无。

对象特定事件 无。

allowTransparency

IE *6* NN *n/a* Moz *n/a* Saf *n/a* Op *n/a* DOM *n/a*

可读/可写

指定框架是否使用透明背景。由于位于框架之下的框架集并没有背景色或背景图，因此这个属性对框架而言并无实际用处。它实际上应用于相关的iframe元素对象。

值 布尔值：true | false。

默认值 false

571 borderColor

IE 4 NN n/a Moz n/a Saf n/a Op n/a DOM n/a

可读/可写

用以表示框架边框的颜色。更准确地说，这个属性应该属于frameset元素，但IE DOM将它作为frame对象的一个属性。

示例 `parent.document.getElementById("myFrame").borderColor = "salmon";`

值 一个十六进制的三元组或明语颜色名称。不设置属性值就默认为黑色（#000000，black）。目前可用的明语颜色名称请参见附录A。

默认值 由操作系统决定。

contentDocument

IE n/a NN n/a Moz all Saf all Op 7 DOM 2

只读

此属性会返回frame元素对象所加载的document对象的引用。通过这个document对象，可以使用getElementById()方法访问文档内的某个元素，也可以使用文档的defaultView属性访问其容器window对象。对IE/Windows而言，还可以使用frame元素对象的contentWindow属性来获取其包含的内容。

示例

```
var frameElem = parent.document.getElementById("myFrame");
var doc = frameElem.contentDocument;
```

值 指向document节点的一个引用。

默认值 当前document节点。

contentWindow

IE 5.5 NN n/a Moz 1.0.1 Saf all Op 7 DOM n/a

只读

此属性会返回frame元素生成的window对象的引用。通过这个window对象，可以访问其生成的document对象，以及该文档中的任意元素。使用W3C DOM提供的contentDocument属性则可以从frame元素对象直接访问其内容。但如果要访问框架中的脚本变量或方法，则须要通过contentWindow（或contentDocument.defaultView）来访问对应的脚本内容。

示例

```
var frameElem = parent.document.getElementById("myFrame");
var win = frameElem.contentWindow;
```

值 window对象的引用。

默认值 当前window对象。

572 dataFld

IE 4 NN n/a Moz n/a Saf n/a Op n/a DOM n/a

可读/可写

此属性用于IE中的数据绑定，以便将一个远程数据源的列名与框架的src属性联系在一起。此时元素中还必须设定datasrc属性。如果dataFld和dataSrc属性均设置为空字符串，那么就会打破元素与数据源之间的绑定关系。注意，只能在IE 5/Mac操作系统下实现文本文件数据源的绑定。

示例 `parent.document.getElementById("myFrame").dataFld = "srcURL";`

值 数据源数据列内区分大小写的标识符。

默认值 无。

dataSrc

IE 4 NN n/a Moz n/a Saf n/a Op n/a DOM n/a

可读/可写

为实现IE数据绑定功能，此属性用于指定页面内object元素的ID值，以便在进行远程数据访问时加载数据源

对象。如果dataFld和dataSrc属性均设置为空字符串，那么就会打破元素与数据源之间的绑定关系。注意，只能在IE 5/Mac操作系统下实现文本文件数据源的绑定。

示例 `parent.document.getElementById("myFrame").dataSrc = "DBSRC3";`

值 数据源内区分大小写的标识符。

默认值 无。

frameBorder

IE 4 NN n/a Moz all Saf all Op 7 DOM 1

可读/可写

此属性用于控制是否显示框架集内的一个独立框架的边框。在大部分浏览器和操作系统中，仅控制一个单独框架的边框会带来问题。如果周边其他框架均显示其边框，那么仅仅隐藏一个框架的可能毫无作用。虽然可以大胆试验隐藏和显示边框的不同效果，但一定要保证访问者通过其操作系统和浏览器浏览页面时的显示效果。此时，如果使用frame元素的frameborder属性或整个frameset对象的frameBorder属性将更为可靠。

示例 `parent.document.getElementById("frame2").frameBorder = "no";`

值 字符串值，“1”表示有边框，“0”表示无边框，也可以使用“yes”和“no”。

默认值 yes

height, width

IE 4 NN n/a Moz n/a Saf n/a Op n/a DOM n/a

只读

返回框架的高度和宽度，单位为像素。这个尺寸应该包含框架的外围装饰内容，如滚动条等。如果需要调整框架的尺寸，可以通过frameset对象的rows及cols属性。

示例 `var frHeight = parent.document.getElementById("myFrame").height;`

值 整数值。

默认值 当前高度和宽度。

longDesc

IE 5(Mac)/6(Win) NN n/a Moz all Saf all Op 7 DOM 1

可读/可写

反映frame元素的longDesc属性。目前主流的浏览器尚未为该属性提供有效的功能。

值 URL字符串。

默认值 空字符串。

marginHeight, marginWidth

IE 4 NN n/a Moz all Saf all Op 7 DOM 1

可读/可写

这两个属性值用于表示框架的内边框与显示内容之间空白距离的像素值。marginheight属性控制框架的上下边距高度，而marginwidth则控制左右边距宽度。

如果没有任何其他提示，浏览器会在框架内自动插入一小块空白空间，根据浏览器和操作系统的不同，这个边距值介于8~14像素之间。但如果想尝试改变默认状态，一定要注意的是，改变其中的任何一个可能会导致另一个值变为0。因此最好同时设置这两个值，以防止框架中的内容完全充满整个框架空间。

示例

```
parent.document.getElementById("myFrame").marginHeight = 14;
parent.document.getElementById("myFrame").marginWidth = 5;
```

值 任何大于或等于0的整数。

默认值 由浏览器和操作系统共同决定。

name

IE 4 NN n/a Moz all Saf all Op 7 DOM 1

可读/可写

这是一个框架的标识符，可以将它指定给target属性，脚本也可以使用它引用对应的框架。通常直接在frame元素的name属性上指定对应值，但也可以通过脚本对其进行修改。

示例 `parent.document.getElementById("myFrame").name = "results";`

值 区分大小写的标识符，必须遵循标识符命名规则：不包含空白、不能以数字作为起始字符、除下划线外不允许使用其他标点符号。

默认值 无。

574

noResize

IE 4 NN n/a Moz all Saf all Op 7 DOM 1

可读/可写

指明对用户是否能够调整框架的大小。此时受影响的框架元素的所有边缘都无法移动，这意味着伸展到框架集中其他框架的边缘也无法移动。

示例 `parent.document.getElementById("myFrame").noResize = "true";`

值 布尔值：`true | false`。

默认值 `false`

scrolling

IE 4 NN n/a Moz all Saf all Op 7 DOM 1

可读/可写

当框架中的内容超出了其可视区域时，它可以控制框架的滚动条的处理方式。通过这一属性，可以强制框架在任意时刻均显示或隐藏滚动条。当然，也可以让浏览器决定是否需要滚动条。但在很多支持这一属性的浏览器中，改变其属性值可能没有任何实际作用。

示例 `parent.document.getElementById("mainFrame").scrolling = "yes";`

值 字符串值，“1”表示有滚动条，“0”表示无滚动条，也可以使用“yes”、“no”和“auto”。

默认值 `auto`

src

IE 4 NN n/a Moz all Saf all Op 7 DOM 1

可读/可写

提供了一个URL，框架加载了对应的外部文件中的内容。如果要改变框架中的内容，直接指定一个新的URL即可。在跨平台的应用中，也可以使用窗口相关的引用来设定框架的location.href属性，以便将不同的文档加载至框架中，例如parent.frameName.location. href = "newDoc.html"。

示例 `parent.document.getElementById("myFrame").src = "images/altNavBar.jpg";`

值 字符串形式的完整或相对URL。

默认值 无。

width

参见height。

frames

IE 3 NN 2 Moz all Saf all Op 7 DOM n/a

窗口或框架中定义的全部frame对象所组成的集合。此时，只有第一层框架会暴露给frameset对象。因此，如果要进一步获取内嵌的框架，则须要深入探究内嵌frameset对象中的frames集合。此外，这个集合还包括窗口文档中定义的iframe元素。

575

对象模型引用方式 [windowRef.]frames
对象特定属性 length
对象特定方法 item()、namedItem()

length

IE 3 NN 2 Moz all Saf all Op 7 DOM n/a

只读

返回定义在某个框架集中的所有子框架的数量，使用时请从该框架集的窗口对象开始引用。这个属性在Mozilla 1.0之前的某些子版本中无法正常工作。

示例
```
var howMany = parent.frames.length;
```
值 整数值。

item( )

IE 4 NN n/a Moz n/a Saf all Op n/a DOM n/a

```
item(index)
```

返回一个window对象，其对应元素的源代码次序可与参数传入的索引值相匹配。

返回值 window对象。

参数

index

一个从零开始计数的整数。

namedItem( )

IE 4 NN n/a Moz n/a Saf n/a Op n/a DOM n/a

```
namedItem(IDOrName)
```

返回单个对象或对象集合，它们对应于符合参数字符串值的元素。

返回值 一个对象或一个对象集合。如果并不存在与参数相匹配的对象，则返回null。

参数

IDOrName

与目标元素的id或name属性包含相同值的字符串。

frameset

IE 4 NN n/a Moz all Saf all Op 7 DOM 1

frameset对象对应于frameset元素。此时，要明确区分frameset元素对象与window对象之间区别。frameset元素对象的属性和方法有助于反映HTML元素及其属性相关的各个方面。frameset中的内容是一个window（W3C DOM术语则称之为视图），可以对它进行脚本操作，而其内部实质上包含一个文档（尽管除frame元素之外不存在其他任何可显示元素）。在脚本编写中，可以通过其ID来引用一个frameset元素对象。例如，可以通过下面这段代码，从驻留在框架文档中的脚本访问frameset元素对象：

```
parent.document.getElementById("myFrameset")
```

但如果要获取框架集的窗口，那么可以在脚本中使用下面两种代码之一来得到所需的对象引用：parent、top。

当脚本在处理frameset元素对象的引用时，可以通过元素的ownerDocument属性来建立元素对象及其内容之间的联系，请参见本章前文中共享属性的相关讨论。

等价HTML元素 <frameset>
对象模型引用方式 [windowRef.]document.getElementById("*framesetID*")
对象特定属性 border、borderColor、cols、frameBorder、frameSpacing、rows
对象特定方法 无。

对象特定事件

事件	IE	Mozilla	Safari	Opera	W3C DOM
afterprint	•	—	—	—	—
beforeprint	•	—	—	—	—
beforeunload	•	—	—	—	—
load	•	•	•	•	•
resize	•	•	•	•	—
unload	•	•	•	•	•

border

IE 4 NN n/a Moz n/a Saf n/a Op n/a DOM n/a

可读/可写

框架集内框架之间间隔的距离，单位为像素。但在嵌套的框架集中，只有最外层的frameset元素才会响应border的属性设定。须要注意的是，IE浏览器对Windows和Macintosh系统采用了不同的默认厚度值，因此同样的设定在不同的操作系统平台下实际效果可能有所不同。

示例 `top.document.getElementById("myFrameset").border = 4;`

值 整数值。设置为0时则完全不显示边框。尽管此值用于指定框架集中边框厚度的精确像素值，但在不同的操作系统或浏览器中，具体的显示效果可能并不完全遵从此属性值。

默认值 6（IE/Windows）；1（IE/Mac）。

borderColor

IE 4 NN n/a Moz n/a Saf n/a Op n/a DOM n/a

可读/可写

此属性可以用于指定框架集内框架边框的颜色。须要注意的是，框架对borderColor属性的设置会替代frameset对象的原有设置。

示例 `parent.document.getElementById("myFrameset").borderColor = "salmon";`

值 一个十六进制的三元组或明语颜色名称。不设置属性值就默认为黑色（#000000，black）。目前可用的明语颜色名称请参见附录A。

默认值 由操作系统决定。

cols

IE 4 NN n/a Moz all Saf all Op 7 DOM 1

可读/可写

此属性定义了框架集中各框架所占据的列尺寸或比例。有如下3种方式定义列尺寸：

- 绝对的像素尺寸；
- 相对于整个框架集宽度的百分比；
- 当其他列使用像素或百分比确定了宽度后，可使用通配符（*）表示剩余的全部可用空间。

在使用脚本改变这一属性时要特别小心。实时改变框架集的组成可能打乱框架之间的脚本联系。例如，减少列数目可能销毁一些文档，而这些文档的脚本或对象可能恰好被其他框架或框架集所使用。因此最安全的做法是保持列数目不变，而使用这一属性来调整现有列的宽度。如果脚本使用了框架集的onresize事件处理程序，那么应该尽可能在各种平台下均进行测试，以便确保脚本在改变这类属性时会触发对应的事件。

示例 `parent.document.getElementById("framesetter").cols = "40%,60%";`

值 使用逗号分隔的数值列字符串，数值可以使用像素值、百分比或通配符（*）这3种形式。

默认值 100%

frameBorder

IE 4 NN n/a Moz n/a Saf n/a Op n/a DOM n/a

可读/可写

对框架集是否显示框架之间的边框进行控制。在IE/Windows中，调整这一属性并不会动态改变边框的可见性。

示例 `parent.document.getElementById("framesetter").frameBorder = "no";`
值 IE 4能接受字符串形式的值，_1表示有边框，0表示无边框，也可以使用yes和no。
默认值 `yes`

frameSpacing

IE 4 NN n/a Moz n/a Saf n/a Op n/a DOM n/a

可读/可写

框架集内框架之间间隔的像素值。在IE/Windows中，调整这一属性并不会动态改变框架间的间距。

示例 `parent.document.getElementById("framesetter").frameSpacing = 5;`
值 整数值。
默认值 `2`

rows

IE 4 NN n/a Moz all Saf all Op 7 DOM 1

可读/可写

指明了框架集中各框架所占据的行尺寸或比例。请参见cols属性以便了解如何为rows选择合适的属性值。

示例 `document.getElementById("myFrameset").rows = "20%, 300, *";`
值 使用逗号分隔的数值列字符串，可以使用像素值、百分比或通配符（*）这3种形式。
默认值 无。

h1, h2, h3, h4, h5, h6

IE 4 NN n/a Moz all Saf all Op 7 DOM 1

这些对象对应于HTML中的同名标头（header）元素。请参见第1章对这些元素的说明，以便了解不同的浏览器如何显示每种标头尺寸。

等价HTML元素 <h1>、<h2>、<h3>、<h4>、<h5>、<h6>
对象模型引用方式 `[window.]document.getElementById("elementID")`
对象特定属性 `align`
对象特定方法 无。
对象特定事件 无。

align

IE 4 NN n/a Moz all Saf all Op 7 DOM 1

可读/可写

定义元素在其容器内的水平对齐方式。

示例 `document.getElementById("myHeader").align = "center";`
值 以下3个对齐常量之一：`center | left | right`。
默认值 `left`

head

IE 4 NN n/a Moz all Saf all Op 7 DOM 1

head对象对应于head元素。在IE 4/Windows中如果通过其ID引用这个对象可能无法正常工作。此时请使用document.all.tags[]集合来获取对应的引用。而在第5版之后的IE浏览器及其他W3C DOM浏览器中，则可

以使用ID、document.getElementsByTagName("head")[0]数组及document.body.previousSibling来引用这个元素对象。

等价HTML元素 <head>

对象模型引用方式

```
[window.]document.getElementById("elementID")
[window.]document.getElementsByTagName("head")[0]
```

对象特定属性 profile

对象特定方法 无。

对象特定事件 无。

580

profile

IE 4 NN n/a Moz all Saf all Op 7 DOM 1

可读/可写

返回为元素的profile属性所指定的URL字符串。在现代的主流浏览器中，为这个属性设定某个值并不会产生任何的特殊结果。

值 URL字符串。

默认值 空字符串。

hidden

请参见input。

history

IE 3 NN 2 Moz all Saf all Op 7 DOM n/a

在一个浏览器会话期间，浏览器可以使用history对象来保存用户已访问的URL列表。浏览器会以数组的形式存储这个列表，并使用它辅助“Back”和“Forward”按钮的导航功能。鉴于history对象中存储的私人信息的敏感性，因此并未将很多信息暴露给脚本，同时也无法捕获这些信息并将它们暗中提交至服务器。而在较新的浏览器版本中，每个窗口均维护着它自身的history对象。

能否阻塞或禁用“Back”按钮的相关动作，是大家常问的一个问题，而它的答案则是否定的。最多也只能在用户点击页面的链接时，防止当前页面被放入浏览器的访问历史列表。此时可以通过location.replace()这一导航方法使用脚本操作链接以达到上述目的。而Navigator 4和Mozilla在使用签名脚本并获得用户的明确认可后，可以从浏览器窗口上移除工具栏，请参见locationbar对象中的讨论。另外，可以打开一个不包含工具栏的新窗口也是一种折中的办法，请参见window.open()方法。

对象模型引用方式 [window.]history

对象特定属性 current、length、navigationMode、next、previous

对象特定方法 back()、forward()、go()、item()

对象特定事件 无。

581

current, next, previous

IE n/a NN 4 Moz all Saf n/a Op (详见下文) DOM n/a

只读

它们分别表示history数组中当前、下一个，以及前一个URL地址。Opera仅支持其中的current属性。而在Navigator 4及Mozilla中，next和previous属性是私有信息，只有使用签名脚本并获得用户批准后才能获取它们。

示例

```
netscape.security.PrivilegeManager.enablePrivilege("UniversalBrowserRead");
```

```
var prevURL = parent.otherFrame.history.previous;
netscape.security.PrivilegeManager.revertPrivilege("UniversalBrowserRead");
```

值 URL字符串。

默认值 无。

length

IE *3* NN *2* Moz *all* Saf *all* Op *7* DOM *n/a*

只读

历史列表中的条目数。除非通过Navigator 4或Mozilla中的签名脚本并获得了用户许可，否则即使获得了这个信息，也不允许进一步获取任何一条指定的历史信息。

示例

```
if (history.length > 4) {
   ...
}
```

值 整数值。

默认值 无。

navigationMode

IE *n/a* NN *n/a* Moz *n/a* Saf *n/a* Op *9* DOM *n/a*

可读/可写

通过这一属性，Opera 9使得用户和页面作者能够控制浏览器处理历史导航的具体方式，即采取“快速”模式（不会触发load和unload事件，以便尽快显示历史页面），还是兼容方式（将触发load和unload事件）或者根据页面间的关系自动决定采用何种处理模式。

示例 `history.navigationMode = "fast";`

值 字符串：`automatic` | `compatible` | `fast`。

默认值 `automatic`

back( )

IE *3* NN *2* Moz *all* Saf *all* Op *7* DOM *n/a*

此方法的基本动作是将窗口导航至上一个已浏览文档，这与点击浏览器的“Back”按钮的效果相同。另外也可以让back()方法定向到一个特定的窗口或框架上，这样就可以改变“Back”按钮的默认行为。例如，如果反复调用parent.otherFrame.history.back()，总会超出该框架的历史列表，最终就会停止下来而没有任何反应。另一方面，如果反复调用top.history.back()，其效果就和不断点击“Back”按钮相同，显然只要当前文档不是这个浏览器会话中的第一个文档，那么整个框架集就会不断后退到第一个页面。

返回值 无。

参数 无。

forward( )

IE *3* NN *2* Moz *all* Saf *all* Op *7* DOM *n/a*

这个方法的基本动作和浏览器的“Forward”按钮相同，它们均会导航到同一个URL页面上。它与history.back()方法有类似的使用建议，请参见上文。

返回值 无。

参数 无。

go( )

IE *3* NN *2* Moz *all* Saf *all* Op *7* DOM *n/a*

```
go(stepCount | "URL")
```

导航到历史列表中的特定位置。

返回值 无。

参数

stepCount

一个整数，表示浏览器应该根据当前列表中的哪一条历史信息进行导航。如果此值为0则会导致重载当前页面，当值为-1时，等同于back()，而值为-2则表示根据当前历史信息后退两步，并载入对应的URL。

URL

URL（仅适用于Navigator）或历史列表中的存储的某个文档标题。

item( )

IE *n/a* NN *4* Moz *all* Saf *n/a* Op *n/a* DOM *n/a*

```
item(itemNumber)
```

返回历史列表中某个特定位置的URL地址。此时需要Netscape/Mozilla的签名脚本及用户的明确许可才能获取这些私有信息。

返回值 URL字符串。

参数

itemNumber

整数，表示历史列表中某条信息的所在位置。它的取值范围从0到history.length-1。

hr

IE *4* NN *n/a* Moz *all* Saf *all* Op *7* DOM *1*

hr对象对应于hr元素。

等价HTML元素 <hr>

对象模型引用方式 [window.]document.getElementById("*elementID*")

对象特定属性 align、color、noShade、size、width

对象特定方法 无。

对象特定事件 无。

align

IE *4* NN *n/a* Moz *all* Saf *all* Op *7* DOM *1*

可读/可写

定义元素在其容器内的水平对齐方式。

示例

```
document.getElementById("myHR").align = "center";
```

值 以下3个对齐常量字符串之一：center | left | right。

默认值 center

color

IE *4* NN *n/a* Moz *1.7* Saf *n/a* Op *n/a* DOM *n/a*

可读/可写

设置水平线的颜色方案。如果以3D的形式显示这条直线，那么会为阴影区域自动指定互补色。

示例

```
document.getElementById("myHR").color = "red";
```

值 一个十六进制的三元组或明语颜色名称字符串，均不区分大小写。目前可用的明语颜色名称请参见附录A。

默认值 由浏览器决定默认值。

noShade

IE *4* NN *n/a* Moz *all* Saf *all* Op *7* DOM *1*

可读/可写

指明浏览器是否应该以平整直线（非3D）的形式显示该线条。本属性仅用于IE浏览器，一旦设置了color属

性，那么浏览器就会将线条的默认样式转换为无阴影的样式。一旦在IE中将noShade设置为true，那么就不能再恢复阴影设置。

示例 document.getElementById("bar2").noShade = "true";
值 布尔值：true | false。
默认值 false

584

size

IE 4 NN n/a Moz all Saf all Op 7 DOM 1

可读/可写

提供水平线的厚度，单位为像素。

示例 document.getElementById("rule2").size = 3;
值 正整数。
默认值 2

width

IE 4 NN n/a Moz all Saf all Op 7 DOM 1

可读/可写

设置线条宽度，此时既可以用像素作为度量单位也可以用临近最外层块级容器的宽度的百分比字符串。

示例 document.getElementById("bar3").width = "70%";
值 整数（像素）或字符串（像素或百分比）。
默认值 100%

html

IE 4 NN n/a Moz all Saf all Op 7 DOM 1

html对象对应于html元素。

等价HTML元素 <html>
对象模型引用方式

```
[window.]document.getElementById("elementID")
[window.]document.body.parentNode
[window.]document.documentElement
```

对象特定属性 version
对象特定方法 无。
对象特定事件 无。

version

IE 5(Mac)/6(Win) NN n/a Moz all Saf all Op 7 DOM 1

585

可读/可写

对应于html元素中一个已被弃用的version属性。尽管它出现在现代浏览器中，但并不具备任何功能。请参见DocumentType对象。

值 字符串。
默认值 空字符串。

HTMLCollection

IE 5(Mac)/6(Win) NN n/a Moz all Saf all Op 7 DOM 1

HTMLCollection对象是W3C DOM中对任意HTML元素对象集合的一种抽象描述，这些HTML元素存在于同一个文档树之中，并且拥有相同的标签。例如，从W3C DOM的角度来看，document.images数组就是一个HTMLCollection对象。这个数组中的每一项都是img元素对象引用。JavaScript将这种集合视为数组，这样在

HTMLCollection的特有属性length的帮助下，可以通过数组符号直接访问集合中某一项。另外，也可以使用item()和namedItem()这两个方法之一来引用集合中的某一项。HTMLCollection对象的所有实例均从这个抽象对象中继承了其属性和方法，如下文所示。请参见本章中列出的每种实例，包括anchors、applets、areas、cells、elements、forms、images、links、options、rows和tBodies，以及all、children、embeds和frames这些非W3C DOM元素集合。

对象特定属性 length
对象特定方法 item()、namedItem()
对象特定事件 无。

length

IE *5(Mac)/6(Win)* NN *n/a* Moz *all* Saf *all* Op *7* DOM *1*

只读

返回集合中元素的数量。

示例
```
var howMany = document.myForm.elements.length;
```
值 整数值。

586

item( )

IE *5(Mac)/6(Win)* NN *n/a* Moz *all* Saf *all* Op *7* DOM *1*

```
item(index)
```

从集合中返回一个对象，该对象的源代码次序可与参数传入的索引值相匹配。而IE还实现了另一个变种方法，该方法包含一个可选的次要参数以应用于某些（而不是全部）集合。

返回值 元素对象的一个引用。如果并不存在与参数相匹配的对象，则返回null。
参数

index

一个从零开始计数的整数，依照源代码次序，对应于集合中的某个特定项。

namedItem( )

IE *5(Mac)/6(Win)* NN *n/a* Moz *all* Saf *all* Op *7* DOM *1*

```
namedItem(IDOrName)
```

从集合中返回一个能够与参数传入的字符串值相匹配的对象。

返回值 元素对象的一个引用。如果并不存在与参数相匹配的对象，则返回null。
参数

IDOrName

包含目标对象名称的字符串。

HTMLDocument

IE *4* NN *n/a* Moz *all* Saf *all* Op *7* DOM *1*

在W3C DOM中，HTMLDocument对象是针对代表HTML文档树的document节点的一种抽象描述。脚本可以通过document对象来引用这个HTMLDocument对象。

这个对象从W3C DOM核心模式中的根Node对象和Document对象中继承了属性和方法。HTMLDocument对象还向这些属性和方法集中加入了一些特别适用于HTML文档的附加内容，例如referrer和body属性，write()方法。此外，各种浏览器实现也加入了很多额外的专有属性和方法。请参考本章前文中关于document对象的讨论。

对象特定属性 anchors[ ]、aplets[ ]、body、cookie、domain、forms[ ]、images[ ]、links[ ]、referrer、title、URL

587

对象特定方法 close()、getElementsByName()、open()、write()、writeln()

HTMLElement

IE 4 NN n/a Moz all Saf all Op 7 DOM 1

从根本上来说，现代浏览器中可进行脚本操作的每种元素对象都是W3C DOM的HTMLElement抽象对象的后代。而HTMLElement对象本身则从核心DOM模型的Node及Element对象中继承了方法和属性。并且HTMLElement还将一些应用于HTML元素的属性添加到这个继承的属性集中，其中包括className、dir、id、lang和title。所有独立的HTML元素对象则从HTMLElement对象中获得它们的特性，例如HTMLBodyElement和HTMLFormElement对象。这也是本章开头列出的共享属性和方法之所以如此之长的一个主要原因，在这个列表中包含了从Node到Element直到HTMLElement这一系列对象中继承的内容。

获知DOM抽象对象名称的术语并不是使用脚本操纵这些元素对象的必要前提。这就意味着抽象对象名称几乎不会出现在脚本中，脚本会通过这些对象的标识符或属性来引用这类HTML对象的实例，例如eventObject.target。只有在调试的过程中才可能会看到这些抽象名称，例如在使用alert()方法或其他工具来检查对象引用变量时就可以看到这类名称。而Mozilla则会将这类对象引用作为一个特定的HTML元素类的实例，例如HTMLParagraphElement或HTMLInputElement。从本质上来说，这种信息比IE所提供的"[object]"要有用得多。

对象特定属性 className、dir、id、lang、title

对象特定方法 无。

i

参见b。

iframe

IE 4 NN n/a Moz all Saf all Op 7 DOM 1

iframe对象对应于iframe元素。须要注意的是，在IE中为这个对象定义的很多属性对对象本身既没有什么作用也未设置任何默认值，但由于对应元素会与使用这些属性的其他元素共享其内部构造，因此仍然实现了这些属性。

等价HTML元素 <iframe>

对象模型引用方式 [window.]document.getElementById("*elementID*")

对象特定属性 align、allowTransparency、border、contentDocument、contentWindow、dataFld、dataSrc、frameBorder、frameSpacing、height、hspace、location、longDesc、marginHeight、marginWidth、name、noResize、scrolling、src、vspace、width

对象特定方法 无。

对象特定事件 无。

align

IE 4 NN n/a Moz all Saf all Op 7 DOM 1

可读/可写

它决定了相对于四周的文本内容iframe元素如何对齐。大多数属性值设定了元素与四周文本之间的竖直关系。例如，如果要将元素的底部与四周文本的基线对齐，那么align属性值应该是baseline。元素也能够沿着容器的左或右边缘"漂浮"，以便使得四周的文本能够完全包围住该元素。

示例 document.getElementById("myIframe").align = "absmiddle";

值 以下任意一个字符串对齐常量：absbottom | absmiddle | baseline | bottom | right | left | none | texttop | top。

默认值 bottom

allowTransparency

IE 6 NN n/a Moz n/a Saf n/a Op n/a DOM n/a

可读/可写

指定iframe是否使用透明背景。为了使得主文档的背景能透过iframe及其文档内容而显示出来，必须将文档的background-color样式属性设置为transparent。

示例 document.getElementById("myIframe").allowTransparency = true;

值 布尔值：true | false。

默认值 false

589

border

IE 4(Win) NN n/a Moz n/a Saf n/a Op n/a DOM n/a

可读/可写

尽管在IE/Windows中为iframe元素对象定义了border属性，但这个属性并没有属性值，即使为它指定了一个值也不会影响元素的外观。

contentDocument

IE n/a NN n/a Moz all Saf all Op 7 DOM 2

只读

此属性会返回iframe元素对象所加载的document对象的引用。通过这个document对象，可以使用getElementById()方法访问文档内的某个元素，也可以使用文档的defaultView属性访问其容器window对象。对IE/Windows而言，可以使用iframe元素对象的contentWindow属性来获取其包含的内容。

示例

```
var iframeElem = parent.document.getElementById("myIframe");
var doc = iframeElem.contentDocument;
```

值 指向document节点的一个引用。

默认值 当前document节点。

contentWindow

IE 5.5 NN n/a Moz 1.0.1 Saf all Op 7 DOM n/a

只读

此属性会返回iframe元素生成的window对象的引用。通过这个window对象，可以访问其生成的document对象，以及该文档中的任意元素。使用W3C DOM提供的contentDocument属性可以从iframe元素对象直接访问其内容。但如果要访问框架中的脚本变量或方法，则须要通过contentWindow（或contentDocument.defaultView）来访问对应的脚本内容。

示例

```
var iframeElem = parent.document.getElementById("myIframe");
var win = iframeElem.contentWindow;
```

值 指向window节点的一个引用。

默认值 当前window节点。

dataFld

IE 4 NN n/a Moz n/a Saf n/a Op n/a DOM n/a

可读/可写

此属性用于IE中的数据绑定，以便将一个远程数据源的列名同src属性值联系在一起。此时元素中还必须设定datasrc属性。如果dataFld和dataSrc属性均设置为空字符串，那么就会打破元素与数据源之间的绑定关系。注意，只能在IE 5/Mac操作系统下实现文本文件数据源的绑定。

590

示例 document.getElementById("myIframe").dataFld = "frameURL";

值 数据源数据列内区分大小写的标识符。

默认值 无。

dataSrc

IE *4* NN *n/a* Moz *n/a* Saf *n/a* Op *n/a* DOM *n/a*

可读/可写

为实现IE数据绑定功能，此属性用于指定页面内object元素的ID值，以便在进行远程数据访问时加载数据源对象。如果dataFld和dataSrc属性均设置为空字符串，那么就会打破元素与数据源之间的绑定关系。注意，只能在IE 5/Mac操作系统下实现文本文件数据源的绑定。

示例
```
document.getElementById("myIframe").dataSrc = "DBSRC3";
```

值 数据源内区分大小写的标识符。

默认值 无。

frameBorder

IE *4* NN *n/a* Moz *all* Saf *all* Op *7* DOM *1*

可读/可写

用于控制iframe元素是否显示边框。

示例
```
document.getElementById("myIframe").frameBorder = "0";
```

值 字符串：1 (on) | 2 (off) 。

默认值 空字符串或1。

frameSpacing

IE *4* NN *n/a* Moz *n/a* Saf *n/a* Op *n/a* DOM *n/a*

可读/可写

框架集内相邻框架的间隔的像素值。在IE中，该属性对内联框架没有任何影响。

height, width

IE *4* NN *n/a* Moz *all* Saf *all* Op *7* DOM *1*

可读/可写

提供iframe元素的高度和宽度的像素值或百分比度量值。

示例
```
document.getElementById("myIframe").height = "200";
document.getElementById("myIframe").width = "500";
```

值 长度字符串。

默认值 宽度为300，高度为150。

hspace, vspace

IE *4* NN *n/a* Moz *n/a* Saf *n/a* Op *n/a* DOM *n/a*

可读/可写

指出内联框架四周在水平和垂直方向上的边距值，单位为像素。hspace属性对元素左右两侧施加相等的影响，而vspace对元素上下两端的影响也是相同的。尽管这些边距与样式单设置的边距并不相同，但它们的实际效果却是一致的。

示例
```
document.getElementById("myIframe").hspace = 5;
document.getElementById("myIframe").vspace = 8;
```

值 整数像素值。

默认值 0

location

IE *n/a* NN *n/a* Moz *n/a* Saf *n/a* Op *7* DOM *n/a*

可读/可写

这个属性会借用iframe元素所创建的window对象的location属性。虽然它是一个完整的location对象，但如果希望将一个新的URL加载至框架内，那么请使用跨浏览器的src属性。

longDesc

IE *5(Mac)/6(Win)* NN *n/a* Moz *all* Saf *all* Op *7* DOM *1*

可读/可写

反映iframe元素的longDesc属性。主流浏览器尚未为该属性提供有效的功能。

值 URL字符串。

默认值 空字符串。

marginHeight, marginWidth

IE *4* NN *n/a* Moz *all* Saf *all* Op *7* DOM *1*

可读/可写

它们用于控制框架的内边框与显示内容之间的空白距离的像素值。除去浏览器提供的默认空白空间，在调整其中一个属性的同时，会将另一个属性设置为0。

值 任何大于或等于0的整数。

默认值 由浏览器和操作系统共同决定。

name

IE *4* NN *n/a* Moz *all* Saf *all* Op *7* DOM *1*

可读/可写

这是一个iframe的标识符，可以将它指定给target属性，脚本也可以使用它引用对应的框架。通常可以直接在frame元素的name属性上指定对应值，但也可以通过脚本对其进行修改。

值 区分大小写的标识符字符串，必须遵循标识符命名规则：不包含空白、不能以数字作为起始字符、除下划线外不允许使用其他标点符号。

默认值 无。

noResize

IE *4* NN *n/a* Moz *n/a* Saf *1.2* Op *n/a* DOM *n/a*

可读/可写

指明用户是否能够调整框架的大小。但这个属性实际上并不能用于iframe元素。

scrolling

IE *4* NN *n/a* Moz *all* Saf *all* Op *7* DOM *1*

可读/可写

当框架中的内容超出了其可视区域时，它可以指明iframe的滚动条的处理方式。通过这一属性，可以强制iframe在任意时刻均显示或隐藏滚动条。当然也可以让浏览器决定是否需要滚动条。由于浏览器通常会忽略脚本对这一属性的修改，因此请通过元素的scorlling属性来作出选择。

示例

```
document.getElementById("myIframe").scrolling = "no";
```

值 下面3个字符串常量之一：auto | no | yes。

默认值 auto

src

IE 4 NN n/a Moz all Saf all Op 7 DOM 1

可读/可写

指明了一个URL，当前元素加载了对应的外部文件中的内容。如果要改变框架中的内容，直接指定一个新的URL即可。

示例 `document.getElementById("myIframe").src = "section2.html";`
值 字符串形式的完整或相对URL。
默认值 无。

vspace

参见请hspace属性。

width

请参见height属性。

ilayer

请参见layer。

Image

IE 4 NN 3 Moz all Saf all Op 7 DOM n/a

通过构造方法（new Image()），静态Image对象可以在浏览器内存中创建一个非显示的img对象。接着为这个实例的src属性指定一个图像文件的URL，就可以命令浏览器为这个图像分配缓存。然后可以将这个非显示实例的src属性指定到文档树中某个img对象的src属性上，这样就可以用缓存中的图像替换已存在的图像。Image对象的实例实际上就是一个DOM img元素，通过该Image对象可以描述img元素的属性、方法和事件。

对象模型引用方式 Image
对象特定属性 prototype
对象特定方法 无。

prototype

IE 4 NN 3 Moz all Saf all Op 7 DOM n/a

可读/可写

这是静态Image对象的一个属性。可以使用这个属性为当前文档中将要创建的图像实例指定新的属性和方法。请参见第5章中的Array.prototype属性，以便了解它的具体用法。

示例 `Image.prototype.exif = "";`
值 任意数据，包括方法的引用。

images

IE 4 NN 3 Moz all Saf all Op 7 DOM 1

文档包含的全部img对象所组成的集合（数组）。只有那些将图像视为对象的浏览器才会实现这一对象。因此可以将这个数组对象的存在性作为一个条件判断并环绕在进行图像替换或预加载的指令四周：

```
if (document.images) {
  // 处理 img 元素对象
}
```

对象模型引用方式 document.images

对象特定属性 length
对象特定方法 item()、namedItem()、tags()、urns()

length

IE 4 NN 3 Moz all Saf all Op 7 DOM 1

只读

返回集合中元素的数量。

示例 var howMany = document.images.length;
值 整数值。

item( )

IE 4 NN n/a Moz all Saf all Op 7 DOM 1

item(*index*[, *subindex*]) item(*index*)

根据符合索引值(IE中也可以使用index或subindex值)的元素对象,返回一个单独的image对象或一组image对象集合。

返回值 一个对象或一个对象集合。如果并不存在与参数相匹配的对象，则返回null。
参数

index

当输入参数是一个从零开始计数的整数（根据W3C规范的需要）时，返回值是源代码内与某个具体项相对应的独立元素（内嵌在当前元素内）。如果参数为一个字符串，则返回id或name属性与该字符串相符的元素集合。

subindex

如果在IE中为第1个参数指定一个字符串值，那么可以将第2个参数指定为一个从零开始计数的索引值，这样就可以从集合中找到一个id或name属性与第1个参数的字符串相匹配的具体元素。

namedItem( )

IE 6 NN n/a Moz all Saf all Op 7 DOM 2

namedItem(*IDOrName*)

返回一个Attr对象，它对应于一个与输入的参数字符串相匹配的元素。

返回值 一个对象引用。如果并不存在与参数相匹配的对象，则返回null。
参数

IDOrName

与目标元素的id或name属性包含相同值的字符串。

tags( )

IE 4 NN n/a Moz n/a Saf n/a Op n/a DOM n/a

tags(*tagName*)

从当前集合内的全部对象中，返回标签与tagName参数相匹配的对象集合。此时由于所有的元素均拥有相同的img标签，因此这完全是一个多余的方法。

返回值 对象集合。如果与参数相匹配的对象不存在，则返回一个长度为零的数组。
参数

tagName

包含元素标签的一个字符串，例如document.images.tags("img")。

urns( )

IE 5(Win) NN n/a Moz n/a Saf n/a Op n/a DOM n/a

urns(*URN*)

返回内嵌元素对象的集合，这些对象附加了一些行为，并且其URN符合输入的*URN*参数。

返回值 对象集合。如果与参数相匹配的对象不存在，则返回一个长度为零的数组。

参数

URN

一个包含本地或外部行为文件的统一资源命名的字符串。

img

IE *4* NN *n/a* Moz *all* Saf *all* Op *7* DOM *1*

img对象对应于img元素。 前文中已经介绍过，不必在页面上显示图像就可以使用静态Image对象在内存中预先缓存图像，而img对象则与静态Image对象共享相同的属性。

等价HTML元素 <img>

对象模型引用方式

```
[window.]document.imageName
[window.]document.images[i]
[window.]document.images["imageName"]
[window.]document.getElementById("elementID")
```

对象特定属性 align、alt、border、complete、dataFld、dataFormatAs、dataSrc、dynsrc、fileCreatedDate、fileModifiedDate、fileSize、fileUpdatedDate、height、href、hspace、isMap、longDesc、loop、lowsrc、lowSrc、mimeType、name、nameProp、naturalHeight、naturalWidth、protocol、src、start、useMap、vspace、width、x、y

对象特定方法 无。

对象特定事件

事件	IE	Mozilla	Safari	Opera	W3C DOM
abort	•	•	•	•	•
error	•	•	•	•	•
load	•	•	•	•	—

align

IE *4* NN *n/a* Moz *all* Saf *all* Op *7* DOM *1*

可读/可写

它决定了在四周的文本内容中iframe元素如何对齐。这个属性的大多数属性值设定了元素与四周文本之间的竖直关系。例如，如果要将元素的底部与四周文本的基线对齐，那么align属性值应该是baseline。元素也能够沿着左或右边缘漂浮，以便使得四周的文本能够完全包围住该元素。

示例 document.logoImg.align = "absmiddle";

值 以下任意一个字符串对齐常量：absbottom | absmiddle | baseline | bottom | right | left | none | texttop | top。

默认值 bottom

alt

IE *4* NN *n/a* Moz *all* Saf *all* Op *7* DOM *1*

可读/可写

它指定了当浏览器无法下载图形或正在等待图像下载时，显示在页面中img元素所在位置的文字内容。这些文字通常简明地描述了图像的内容。须要注意的是，页面上图像所在的区域大小可能限制了可显示的文本数量。因此必须确保这些说明文字的能够被完全显示出来。

示例 document.corpLogo.alt = "MegaCorp Logo";

值 字符串。

默认值 无。

597

border

IE 4 NN 3 Moz all Saf all Op 7 DOM 1

可读/可写

提供了元素四周边框的厚度，单位为像素。在Navigator中，直到第4版为止，它仍然只是一个只读属性。

示例 `document.logoImage.border = 4;`

值 整数值。设置为0时则完全不显示边框。

默认值 0

complete

IE 4 NN 3 Moz all Saf n/a Op 7 DOM n/a

只读

用以表示img元素的src或lowsrc图像文件是否已经被完全加载。须要注意的是，在图像全部加载完成之前，Navigator 4会提供不正确的值——true。

示例

```
if (document.logo.complete) {
    // 对图像对象进行可靠的处理
}
```

值 布尔值：true、false。

默认值 false

dataFld

IE 4 NN n/a Moz n/a Saf n/a Op n/a DOM n/a

可读/可写

此属性用于IE中的数据绑定，以便将一个远程数据源的列名同img对象的src属性联系在一起。此时元素中还必须设定datasrc属性。如果dataFld和dataSrc属性均设置为空字符串，那么就会打破元素与数据源之间的绑定关系。注意，只能在IE 5/Mac操作系统下实现文本文件数据源的绑定。

示例 `document.myImage.dataFld = "logoURL";`

值 数据源数据列内区分大小写的标识符。

默认值 无。

dataFormatAs

IE 4 NN n/a Moz n/a Saf n/a Op n/a DOM n/a

可读/可写

虽然这个属性是IE中img元素对象的一部分，但由于数据绑定值被链接到了src属性而不是已显示的元素，因此并未将它应用于img对象。

598

dataSrc

IE 4 NN n/a Moz n/a Saf n/a Op n/a DOM n/a

可读/可写

为实现IE数据绑定功能，此属性用于指定页面内object元素的ID值，以便在进行远程数据访问时加载数据源对象。如果dataFld和dataSrc属性均设置为空字符串，那么就会打破元素与数据源之间的绑定关系。注意，只能在IE 5/Mac操作系统下实现文本文件数据源的绑定。

示例 `document.myImage.dataSrc = "DBSRC3";`

值 数据源内区分大小写的标识符。

默认值 无。

dynsrc

IE 4 NN n/a Moz n/a Saf n/a Op n/a DOM n/a

可读/可写

一段将要通过img元素来显示的视频剪辑的URL地址。改变这个属性会将一段新的视频剪辑加载至图像对象之中。同时请参考loop属性以便控制视频剪辑的播放次数。

示例 `document.images[3].dynsrc = "snowman.avi";`

值 字符串形式的完整或相对URL。

默认值 无。

fileCreatedDate

IE 4(Win)/5(Mac) NN n/a Moz n/a Saf n/a Op n/a DOM n/a

只读

返回一个日期（而不是时间）字符串，服务器（或本地文件系统）以此表示当前加载文件的创建日期。通过Image对象加载一个图像后，可以使用一段脚本确定该图像的日期，但是fileUpdatedDate属性可以提供更为准确的数据。IE 4的值是一个长日期格式，但从IE 5开始，日期信息的格式就变为“mm/dd/yyyy”。如果服务器提供的日期格式不符合IE的要求，那么这个值可能会出错。此外，尽管IE 5/Mac也实现了这一属性，但其值为空。

示例 `var dateObj = new Date(document.logoImg.fileCreatedDate);`

值 日期字符串。

默认值 无。

fileModifiedDate

IE 4(Win)/5(Mac) NN n/a Moz n/a Saf n/a Op n/a DOM n/a

只读

返回一个日期（而不是时间）字符串，服务器（或本地文件系统）以此表示当前加载的文件的最近一次修改日期。IE 4的值是一个长日期格式，但从IE 5开始，日期信息的格式就变为“mm/dd/yyyy”。如果服务器提供的日期格式不符合IE的要求，那么这个值可能会出错。此外，尽管IE 5/Mac也实现了这一属性，但其值为空。

示例 `var dateObj = new Date(document.logoImg.fileModifiedDate);`

值 日期字符串。

默认值 无。

fileSize

IE 4(Win)/5(Mac) NN n/a Moz n/a Saf n/a Op n/a DOM n/a

只读

返回当前已加载图像大小的字节数。IE/Windows会返回字符串形式的数值，而IE/Macintosh则会返回一个数字值，但该值为0。

示例 `var byteCount = parseInt(document.fileSize, 10);`

值 字符串形式的整数（Windows）或数字（Mac）。

默认值 无。

fileUpdatedDate

IE 4(Win)/5(Mac) NN n/a Moz n/a Saf n/a Op n/a DOM n/a

只读

对于一个从服务器获取的图像文件，这个属性可以比其他相关属性更为准确地反映该文件最后一次上传至服务器的日期。而本地文件通常会返回一个空字符串。此外，尽管IE 5/Mac也实现了这一属性，但其值为空。

示例 `var dateObj = new Date(document.logoImg.fileUpdatedDate);`

值 日期字符串。
默认值 无。

height, width

IE 4 NN 3 Moz all Saf all Op 7 DOM 1

可读/可写

为显示在img元素中的图像提供了高度和宽度，单位为像素。除NN 4及早期版本之外，在所有支持这两个属性的浏览器中，这些值的修改操作会立刻反应在页面的内容排列上。但要注意对图像进行缩放以适应新的空间大小。

示例
```
document.prettyPicture.height = 250;
```
值 整数值。
默认值 无。

href

IE 4 NN n/a Moz n/a Saf n/a Op n/a DOM n/a

可读/可写

它对应于由元素的src属性所指定的URL地址。虽然它与src属性完全相同，但并不推荐使用这一属性。

示例
```
document.logoImage.href = "images/fancyLogo.gif";
```
值 由完整或相对URL路径组成的字符串。
默认值 无。

hspace, vspace

IE 3 NN 4 Moz all Saf all Op 7 DOM 1

可读/可写

指出一个图像对象四周在水平和垂直方向上的边距值，单位为像素。hspace属性对元素左右两侧施加相等的影响，而vspace对元素上下两端的影响也是相同的。尽管这些边距与样式单设置的边距并不相同，但它们的实际效果却是一致的。在NN 4及早期版本中，它们均为只读属性。

示例
```
document.logo.hspace = 5;
document.logo.vspace = 8;
```
值 整数像素值。
默认值 0（IE、Safari、Opera），-1（Mozilla）。

isMap

IE 4 NN n/a Moz all Saf all Op 7 DOM 1

可读/可写

指明该img元素是否作为一个服务器端图像映射。为了将一个图像转化为一个服务器端图像映射，首先须要将img元素内置于一个a元素中，同时该a元素的href属性应指向一个CGI程序的URL地址，用以解析点击时的坐标信息。就像使用get方法提交表单时须要将表单数据附加在action属性的URL之后一样，浏览器也会将点击发生点的坐标信息附加在URL后。

当然，许多新型浏览器也允许使用客户端图像映射（请参见useMap属性），由于不须要与服务器之间通信以检查点击时的坐标位置，用户使用这种图像映射时会更为快捷。

示例
```
document.navMap.isMap = true;
```
值 布尔值：true | false。
默认值 false

longDesc

IE *5(Mac)/6(Win)* NN *n/a* Moz *all* Saf *all* Op *7* DOM *1*

可读/可写

反映img元素的longDesc属性。主流浏览器尚未为该属性提供有效的功能。

值 URL字符串。
默认值 空字符串。

loop

IE *4* NN *n/a* Moz *n/a* Saf *n/a* Op *n/a* DOM *n/a*

可读/可写

如果用dynsrc属性指定了一段视频剪辑，那么就可通过loop属性控制该剪辑的播放次数。如果将这个值设置为-1，即表示不断地反复播放。

示例
```
document.movieImg.loop = 3;
```
值 整数值。
默认值 1

lowsrc

IE *4* NN *3* Moz *all* Saf *n/a* Op *7* DOM *n/a*

可读/可写

如果src属性指定的图像需要很长时间才能下载，那么可以使用这个属性指定一个低分辨率（或替代）图像的URL，以便文档空间快速加载。注意，这个低清晰度的图像应该与src指定的原始图像拥有相同的像素尺寸。如果将要准备修改src属性，那么此时改变lowsrc才具有一定的意义。在这种情况下，必须首先修改lowsrc属性，这样就可以让浏览器知道如何处理src图像加载时间过长的问题。

须要注意的是，早期的Mozilla浏览器还为这个属性实现了另一种不同形式的拼写方法，即lowSrc。不过这两种形式都未出现在W3C DOM规范之中。

示例
```
document.productImage.lowsrc = "images/widget43LoRes.jpg";
```
值 字符串形式的完整或相对URL。
默认值 无。

mimeType

IE *6* NN *n/a* Moz *n/a* Saf *n/a* Op *n/a* DOM *n/a*

只读

为该图像返回明语形式的文件MIME类型描述。虽然微软并未为这个属性提供官方支持，但它的确能够正确地反映一些来自于本地磁盘或服务器的典型文件类型。

示例
```
if (document.productImage.mimeType.indexOf("JPEG") != -1) {
    // 对 JPEG 图像进行处理
}
```
值 字符串，例如JPEG或GIF。
默认值 无。

name

IE *3* NN *2* Moz *all* Saf *all* Op *7* DOM *1*

可读/可写

这是一个与图像对象相关的标识符，脚本可以使用它来引用对象，例如，可通过document.images数组来引用对象。

示例
```
var imgName = document.images[3].name;
```

值 区分大小写的标识符字符串，必须遵循标识符命名规则：不包含空白、不能以数字作为起始字符、除下划线外不允许使用其他标点符号。

默认值 无。

nameProp

IE *5(Win)* NN *n/a* Moz *n/a* Saf *n/a* Op *n/a* DOM *n/a*

只读

返回当前图像的文件名（不包括URL路径中的其他内容）。这样可以简化对当前图像内容的检查工作。

示例

```
if (document.images[3].nameProp == "menuOn.jpg") {
   document.image[3].src = "../images/menuOff.jpg";
}
```

值 区分大小写的文件名及扩展名字符串。

默认值 无。

naturalHeight, naturalWidth

IE *n/a* NN *n/a* Moz *all* Saf *n/a* Op *n/a* DOM *n/a*

只读

返回图像的真实高度和宽度，单位为像素。使用脚本既可以获得原始图像的真实尺寸，也可以得到那些改变图像大小的元素属性。

示例

```
document.logoImg.height = document.logoImg.naturalHeight;
document.logoImg.width = document.logoImg.naturalWidth;
```

值 整数值。

默认值 无。

protocol

IE *4(Win)/5(Mac)* NN *n/a* Moz *n/a* Saf *n/a* Op *n/a* DOM *n/a*

只读

返回URL中的协议部分。Windows和Macintosh平台下的IE浏览器会返回不同的协议形式。在Windows版本中，返回值是以扩展的明语形式表示，如“`File Protocol`”或“`HyperText Transfer Protocol`”，而在Macintosh版本中，则类似于`location.protocol`的值，如“`file:`”或“`http:`”。

值 字符串。

默认值 无。

src

IE *4* NN *3* Moz *all* Saf *all* Op *7* DOM *1*

可读/可写

提供图像文件的相对或绝对URL地址，`img`元素已加载或将要加载该图像。除Netscape 3、Netscape 4及IE 3/Macintosh之外，如果要在一个已存在的`img`元素中加载另一个不同尺寸的图像，均会迫使该元素重新适应新图像的尺寸大小。另外，无论之前是以哪种URL形式进行属性赋值，都应该将这个属性的返回值视为一个完整的URL。

示例

```
document.image[3].src = "../images/menuOff.jpg";
```

值 URL字符串。

默认值 无。

start

IE *4* NN *n/a* Moz *n/a* Saf *n/a* Op *n/a* DOM *n/a*

可读/可写

如果用`dynsrc`属性指定了一段视频剪辑，那么`start`属性可以控制启动视频播放所需的具体动作。

示例 `document.movieImg.start = "mouseover";`
值 字符串常量：`fileopen`、`mouseover`。
默认值 `fileopen`

useMap

IE *4* NN *n/a* Moz *all* Saf *all* Op *7* DOM *1*

可读/可写

提供同一文档内某个map元素的URL地址，这个元素包含着客户端图像映射热点区域及链接。属性值由一个“#”及img元素的usemap属性中的映射名称组成。

示例 `document.images[0].useMap = "#altMap";`
值 一个以“#”开头，后跟map元素名称的字符串。
默认值 无。

vspace

参见hspace。

width

请参见height。

x, y

IE *n/a* NN *4* Moz *all* Saf *all* Op *n/a* DOM *n/a*

提供图像左上角相对于页面的水平坐标及竖直坐标，单位为像素。这两个属性只适用于Navigator，IE中的对应属性是offsetLeft和offsetTop。

示例 `var imageFromTop = document.logoImg.y;`
值 整数值。
默认值 无。

implementation

IE *5(Mac)/6(Win)* NN *n/a* Moz *all* Saf *all* Op *7* DOM *1*

implementation对象在一定程度上可以（针对W3C DOMImplementation 对象的JavaScript引用）表示组成文档容器的外界环境，对我们而言，这个环境就是浏览器。可以通过document.implementation属性访问这个对象。

这个对象的方法能够给出浏览器声明支持的DOM模型。在W3C DOM浏览器中，这个对象也可以视为一个入口，通过它可以在当前文档树之外创建虚拟的W3C Document和DocumentType对象。因此在支持W3C DOM的浏览器中，可以将document.implementation作为一个出发点，从而为外部XML文档生成一个非显示的文档。

Opera则实现了另外几个相关的方法，如createLSInput()、createLSOutput()、createLSParser()及createLSSerializer()，它们均定义在W3C DOM Level 3的加载和保存模块中。如果要获取更多详细信息，请参考W3C推荐标准：*http://www.w3.org/TR/DOM-Level-3-LS/load-save.html*。

对象模型引用方式 `document.implementation`
对象特定属性 无。
对象特定方法 `createCSSStyleSheet()`、`createDocument()`、`createDocumentType()`、`createHTMLDocument()`、`hasFeature()`
对象特定事件 无。

605

createCSSStyleSheet( )

IE *n/a* NN *n/a* Moz *n/a* Saf *1.2* Op *n/a* DOM *2*

```
createCSSStyleSheet("title", "mediaList")
```

返回一个新创建的虚拟styleSheet对象的引用，该对象尚未被分配相关规则。然后可以通过styleSheet.addRule()方法添加所需的规则。但不幸的是，根据W3C DOM推荐标准，目前还无法将一个以这种形式创建的styleSheet对象与实际应用中的某个文档联系起来。因此，为了插入样式单设定，可以针对文档树中一个已存的styleSheet对象使用addRule()方法。

返回值 一个styleSheet对象的引用，该对象尚未与一个Document对象关联起来。

参数

title

一个包含标题的字符串。

mediaList

一个以逗号分隔的输入媒介字符串，如screen、pringt，样式单将使用在对应媒介上。

createDocument( )

IE *n/a* NN *n/a* Moz *all* Saf *all* Op *7* DOM *2*

```
createDocument("namespaceURI", "qualifiedName", docTypeReference)
```

返回一个新创建的虚拟W3C DOM Document对象(注意,并不是HTML文档的document节点)的引用。Mozilla为这个Document对象增加了一个load()方法，通过这个方法就能够在浏览器内存中加载XML文档，目前已经用XMLHttpRequest对象代替了这种做法。

返回值 一个空Document节点对象的引用。

参数

namespaceURI

新XML文档元素命名空间的URI字符串。

qualifiedName

新文档元素的限定名字符串标识符。

docTypeReference

一个DocumentType节点的引用，可以通过document.implementation.createDocumentType()方法来生成这个节点。

createDocumentType( )

IE *n/a* NN *n/a* Moz *all* Saf *all* Op *7* DOM *1*

```
createDocumentType("qualifiedName", "publicID", "systemID")
```

返回一个新创建的虚拟W3C DOM DocumentType对象的引用。可以将这个方法返回的对象作为document.implementation.createDocument()方法的一个参数。

返回值 一个DocumentType对象的引用，该对象尚未与一个Document对象关联起来。

606

参数

qualifiedName

新文档元素的限定名字符串标识符。

publicID

针对DOCTYPE的公共标识符字符串。

systemID

针对DOCTYPE的系统标识符字符串，通常就是DTD文件的URI。

createHTMLDocument( )

IE *n/a* NN *n/a* Moz *n/a* Saf *1.2* Op *7* DOM *2*

```
createHTMLDocument("title")
```

返回一个新创建的虚拟W3C DOM HTMLDocument对象（注意，并不是HTML文档的document节点）的引用。

其实，这个文档就是一个框架文档（如，<HTML><HEAD><TITLE>*titleParameterText*</TITLE></HEAD></HTML>），可以向它附加一些自己创建的元素。

返回值 一个空HTMLDocument节点对象的引用。

参数

title

文档标题字符串，已经将它作为实体内容插入到title元素中。

hasFeature( )

IE *5(Mac)/6(Win)* NN *n/a* Moz *all* Saf *all* Op *7* DOM *1*

```
hasFeature("feature", "version")
```

如果某个浏览器支持一个指定的W3C DOM模型和版本，那么返回一个布尔值true。而与它密切相关的isSupported()方法则是在一个单独的节点上完成检测工作，以便检查当前节点类型是否能为某些功能提供支持。但这两个方法所需的参数值是完全一致的。

在浏览器为特定的DOM模型返回true之前，浏览器应该证其中实现的DOM能够符合该模型。但此时并不强求这些实现完美无缺没有漏洞，也不要求它与其他实现保持一致。因此这不过是个资格记录而已。

从理论上说，可以在访问一个属性或调用一个方法之前使用这个方法来检查所用的模型是否得到了支持。下面这段来自于文档头部的脚本片段会根据浏览器是否支持CSS2而动态设定不同的外部样式单文件：

```
var cssFile;
if (document.implementation.hasFeature("CSS", "2.0")) {
   cssFile = "styles/corpStyle2.css";
} else {
   cssFile = "styles/corpStyle1.css";
}
document.write("<link rel='stylesheet' type='text/css' href='" + cssFile + "'>");
```

相对于元素特定方法而言，更多的浏览器能够支持这个浏览器范围的方法，这使得大多数开发者能够更加快速地部署应用。

返回值 布尔值：true | false。

参数

feature

到W3C DOM Level 2为止，可使用的模型名称字符串（区分大小写）如下：Core、XML、HTML、Views、StyleSheets、CSS、CSS2、Events、UIEvents、MouseEvents、MutationEvents、HTMLEvents、Range、Traversal和Views。Level 3则增加了更多的功能特性，例如LS（即加载与保存模块）、BasicEvents、TextEvents及KeyboardEvents。

version

字符串表达式，使用主、次两个数字表示第1个参数所引用的DOM模型的版本号。即使DOM模型支持另一种拥有版本计数系统的W3C标准，W3C DOM Level 2的版本号总是“2.0”。因此就算HTML版本为4.x，也只须要检查对2.0版的HTML DOM模型是否支持。

imports

IE *4* NN *n/a* Moz *n/a* Saf *n/a* Op *n/a* DOM *n/a*

已通过@import规则导入到一个styleSheet对象之中的styleSheet对象组成的集合（数组）。换句话说，一个使用了@import的styleSheet对象会将导入的样式单视为内嵌对象，但这个导入的样式单本身也是一个完整的styleSheet对象。

因此，可以访问每个导入样式单内的rule对象。下面这段代码演示了如何遍历文档的所有styleSheet对象以查找导入的样式单：

```
for (var i = 0; i < document.styleSheets.length; i++) {
  for (var j = 0; j < document.styleSheets[i].imports.length; j++) {
```

```
    // 处理document.styleSheets[i].imports[j]所引用的已导入的样式单
  }
}
```

对象模型引用方式 document.styleSheets[i].imports
对象特定属性 length
对象特定方法 item()

length

IE *4* NN *n/a* Moz *n/a* Saf *n/a* Op *n/a* DOM *n/a*

只读

返回集合中对象的数量。

示例

```
var howMany = document.styleSheets[i].imports.length;
```

值 整数值。

item( )

IE *4* NN *n/a* Moz *n/a* Saf *n/a* Op *n/a* DOM *n/a*

```
item(index)
```

返回一个已导入的styleSheet对象，该对象在@import规则的源代码中所处的索引位置与参数输入的索引值相等。注意，IE 5/Macintosh会错误地返回null。

返回值 一个已导入的styleSheet对象的引用。如果并不存在与参数相匹配的对象，则返回null。

参数

index

一个从零开始的整数。

input

IE *4* NN *n/a* Moz *all* Saf *all* Op *7* DOM *1*

input对象对应于input元素。自从第一代脚本浏览器问世以来，就能够通过脚本访问大多数输入元素。而IE浏览器在第4版之后就和Mozilla、Safari和Opera一样，在过去的几年间都支持脚本访问很多相关属性。

在W3C DOM规范中，即使某些属性并非完全适用，所有的input元素对象仍共享着相同的属性，例如，文本输入类型的input元素也拥有checked属性。因此为减少这种情况带来的潜在的混淆，本书为每种属性和方法列举了它们真正适用的输入类型，这一做法和第1章中HTML输入元素的说明方式比较类似。

而在对象特定项列表中并未出现的一些属性和方法也值得大家关注。尽管IE DOM（尤其在Windows系统中）将某些属性划归给每个HTML元素，例如accessKey、disabled和tabIndex，但W3C DOM在为元素分配这些属性时却显得更为谨慎。但无论是上面哪种情况，这些属性都存在于input元素之中，并且在所有的现代浏览器中均可以找到这些input元素的身影。对于blur()、click()和focus()等方法而言，情况也比较类似，本章前文中的共享项中也对它们进行了一些介绍。

在这些输入元素中，radio是其中一个比较复杂的输入元素类型。如果希望一组radio输入元素能够实现互斥功能（即永远保持最多只选中其中一个），那么就须要将所有对应元素的name属性指定为相同的标识符。这意味着所有同名的radio对象会组成一个单选框对象集合（数组）。因此有必要将一个单独的单选按钮作为数组的一员进行引用。当然，这个数组也包含一个length属性，它为遍历组内的radio对象提供了一定的帮助，从而找到已被选中的单选框并获取其对象的值，示例代码如下所示：

```
var radioGrp = document.forms[0].myRadio;
for (var i = 0; i < radioGrp.length; i++) {
```

```
    if (radioGrp[i].checked) {
      alert("The value of the chosen button is " + radioGrp [i].value);
    }
  }
```

下面列出的很多属性和方法由Web Forms 2.0所独有，它们首先实现于Opera 9浏览器之中。因此，会在相关描述中明确指出它们是否适用于Web Forms 2.0。可以参考第1章中的input元素，以了解在Web Forms 2.0中如何对input元素的进行设置。

另外，此处列出的所有事件类型均适用于全部脚本浏览器。当然，所有的可显示的HTML元素均共享鼠标和键盘事件，本章前文中已经进行了一些说明。请参考第9章（译注6），以便了解各种input元素定制事件类型能够被哪种输入类型所识别。

对象模型引用方式

```
[window.]document.formName.elementName
[window.]document.forms[i].elements[i]
[window.]document.getElementById("elementID")
```

对象特定属性 accept、action、alt、autocomplete、autofocus、checked、complete、dataFld、dataSrc、defaultChecked、defaultValue、dynsrc、enctype、form、forms[]、height、hspace、htmlTemplate、indeterminate、inputmode、labels[]、list、loop、lowsrc、max、maxLength、method、min、name、pattern、readOnly、replace、required、selectedOption、selectionEnd、selectionStart、size、src、start、status、step、target、type、useMap、validationMessage、validity、value、valueAsDate、valueAsNumber、width、willValidate

对象特定方法 checkValidity()、createTextRange()、dispatchChange()、dispatchFormChange()、handleEvent()、select()、setCustomValidity()

对象特定事件

事件	事件	事件
change	invalid (Web Forms 2.0)	select

accept

IE *4* NN *n/a* Moz *all* Saf *all* Op *7* DOM *1*

可读/可写

这个属性提供了一个参考建议，其字符串值由一个或多个使用逗号进行分隔的MIME文件类型组成，以表示哪些文件可以被上传。在当前的浏览器中这些值对input元素没有任何影响。

input类型 file

示例

```
document.entryForm.myFileUpload.accept = "image/gif";
```

值 字符串。

默认值 无。

action

IE *n/a* NN *n/a* Moz *n/a* Saf *n/a* Op *9* DOM *n/a*

可读/可写

点击button元素（须要将其type属性设置为“submit”）后，这个Web Forms 2.0扩展可以将封闭表单提交至另一个URI，而该URI可以与常规的表单提交地址不同。

input类型 image、submit

示例

```
document.getElementById("redirSubmit").action = "redirect.php";
```

译注6：此部分内容请读者访问 *http://www.china-pub.com/195581* 进行在线阅读。

值 URI字符串。
默认值 无。

alt

IE *4* NN *n/a* Moz *all* Saf *all* Op *7* DOM *1*

可读/可写

它提供了当浏览器无法下载图形或正在等待图像下载时，显示在页面中图像类型的input元素所在位置的文字内容。请参考img对象的alt属性以获得更多详细信息。

input类型 image

autocomplete

IE *n/a* NN *n/a* Moz *n/a* Saf *n/a* Op *9* DOM *n/a*

可读/可写

这个Web Forms 2.0扩展说明了是否采用浏览器的文本域自动填写功能。在Opera 9中，尽管这个属性是文本相关的输入元素的成员，但改变这个值并不会对浏览器的运转产生任何影响。

input类型 Web Forms 2.0中与文本相关的输入类型。
值 字符串值：on | off。
默认值 on

autofocus

IE *n/a* NN *n/a* Moz *n/a* Saf *n/a* Op *9* DOM *n/a*

可读/可写

当页面加载完成后，这个Web Forms 2.0 扩展属性会将焦点聚集在元素上。每个页面上只能有一个表单控件元素拥有此属性。

input类型 Web Forms 2.0中与文本相关的所有可显示的输入类型。
值 布尔值：true | false。
默认值 false

checked

IE *3* NN *2* Moz *all* Saf *all* Op *7* DOM *1*

可读/可写

决定某个按键是否被用户或脚本所选中或开启。表单中各复选框的操作彼此独立。只有将radio或checkbox对象的checked属性设置为true时，它们的名称/值对才会随表单一起提交至服务器。在页面加载时如果要获知是否已选中了某个表单元素，请查看defaultChecked属性。另外，即使已经禁用了某个元素，也可以使用脚本改变其属性值。

input类型 checkbox、radio
示例

```
if (document.choiceForm.myRadio[0].checked) {
  // 处理第 1 个单选按钮
}
```

值 布尔值：true | false。
默认值 false

complete

IE *4* NN *n/a* Moz *n/a* Saf *n/a* Op *n/a* DOM *n/a*

只读

用以表示图像类型的input元素的src或lowsrc图像文件是否已经被完全加载。请参考img对象的complete属性以获得更多详细信息。

input类型 image

dataFld

IE 4 NN n/a Moz n/a Saf n/a Op n/a DOM n/a

可读/可写

此属性用于IE中的数据绑定，以便将一个远程数据源的列名与button对象的value属性联系在一起。此时元素中还必须设定datasrc属性。如果dataFld和dataSrc属性均设置为空字符串，那么就会打破元素与数据源之间的绑定关系。注意，只能在IE 5/Mac操作系统下实现文本文件数据源的绑定。

input类型 button、checkbox、hidden、password、radio、text
示例 document.myForm.myButton.dataFld = "linkURL";
值 数据源数据列内区分大小写的标识符。
默认值 无。

dataSrc

IE 4 NN n/a Moz n/a Saf n/a Op n/a DOM n/a

可读/可写

为实现IE数据绑定功能，此属性用于指定页面内object元素的ID值，以便在进行远程数据访问时加载数据源对象。通过datafld属性指定来自于数据源中的绑定内容。如果dataFld和dataSrc属性均设置为空字符串，那么就会打破元素与数据源之间的绑定关系。注意，只能在IE 5/Mac操作系统下实现文本文件数据源的绑定。

input类型 button、checkbox、hidden、password、radio、text
示例 document.myForm.myButton.dataSrc = "DBSRC3";
值 数据源内区分大小写的标识符。
默认值 无。

defaultChecked

IE 3 NN 2 Moz all Saf all Op 7 DOM 1

可读/可写

指明元素标签中是否设置了checked属性。比较当前的checked与defaultChecked这两个属性值后，就可以知道控件的状态是否在文档加载之后发生了改变。改变这个属性并不会影响当前的checked状态。

input类型 checkbox、radio
示例

```
var cBox = document.forms[0].checkbox1
if (cBox.checked != cBox.defaultChecked) {
  // 对状态改变作出响应
}
```

值 布尔值：true | false。
默认值 由HTML标签属性决定。

defaultValue

IE 4 NN 4 Moz all Saf all Op 7 DOM n/a

可读/可写

返回源代码中元素value属性指定的属性值字符串。但在IE/Windows中，其返回值总是一个空字符串。由于用户必须在file类型中手动选择一个待上传的文件，因此预先设置或试图改变这个值完全没有必要。

input类型 file、hidden、password、text
示例 var initVal = document.entryForm.myFileUpload.defaultValue;
值 字符串。
默认值 无。

613

dynsrc

IE 4 NN n/a Moz n/a Saf n/a Op n/a DOM n/a

可读/可写

提供一个视频剪辑的URL地址，以便在图像类型的input元素的空间中播放。请参考img对象的dynsrc属性以获得更多详细信息。

input类型 image

enctype

IE n/a NN n/a Moz n/a Saf n/a Op 9 DOM n/a

可读/可写

通过enctype与其他属性（如action）的协助，在点击submit按钮或图像以提交封闭表单时，这个Web Forms 2.0 扩展可以使得提交目标URI与附件的MIME类型均与常规的表单不同。

input类型 image、submit
示例 document.getElementById("mySubmit").enctype = "text/plain";
值 MIMEtype字符串。
默认值 无。

form

IE 3 NN 2 Moz all Saf all Op 7 DOM 1

只读

返回包含当前元素的form元素的引用。在处理一个来自于该元素的事件时，对应的事件处理方法可以通过事件对象的target或srcElement属性来自动访问对应的input元素。而通过读取form属性，脚本可以轻易地访问同一个表单中的其他控件。在Web Forms 2.0环境中，表单控件可能会和文档树中某个并不包含该控件的表单元素联系在一起。因此这个属性可以指向任意一个与控件相关的表单元素。

input类型 所有类型。
示例 var theForm = evt.srcElement.form;
值 form元素对象的引用。
默认值 无。

forms

IE n/a NN n/a Moz n/a Saf n/a Op 9 DOM n/a

只读

这个Web Forms 2.0扩展返回一个与当前input元素相关联的form对象引用所组成的数组（NodeList）。

示例 var formList = document.getElementById("myButton").forms;
值 数组。
默认值 包含一个封闭表单元素的引用。

614

height, width

IE 4 NN n/a Moz all Saf all Op 7 DOM 1

可读/可写

指明了显示在input元素中的图像的高度和宽度，单位为像素。请参考img对象的height和width属性以获得更多详细信息。

input类型 image

hspace, vspace

IE 4 NN n/a Moz all Saf all Op 7 DOM 1

可读/可写

指出在一个图像类型的input对象四周水平和垂直方向上的边距值，单位为像素。更多详细信息请参考img对象的hspace和vspace属性。

input类型 image

htmlTemplate

IE *n/a* NN *n/a* Moz *n/a* Saf *n/a* Op *9* DOM *n/a*

只读

这个Web Forms 2.0扩展返回元素对象（RepetitionElement对象）的一个引用，该对象的ID与当前表单控件的template属性（必须设置非null值）相匹配。

input类型 add

示例
```
var repeatTemplate = document.getElementById("myButton").htmlTemplate;
```

值 元素对象的引用。

默认值 null

indeterminate

IE *4* NN *n/a* Moz *n/a* Saf *n/a* Op *n/a* DOM *n/a*

可读/可写

指明一个仍处于活跃状态下的复选框看起来是否处于一种介于选中和未选中的中间状态。在不同的操作系统中，这种中间状态的显示效果也有所不同。在Windows中，复选框会被灰化，但依然让它处于活跃状态。于此同时，如果之前已将它打上了钩，那么这个钩也不会消失。而在Macintosh中，会在复选框的方框中显示一个连字符（“-”）。这种模棱两可的状态往往表示页面中的其他修改可能会影响到这个复选框的设定，此时须要由用户来检查复选框的设定，以确保其正确性。而这个不确定的复选框也会随表单一起被提交至服务器。

input类型 checkbox

示例
```
document.orderForm.2DayAir.indeterminate = true;
```

值 布尔值：true | false。

默认值 false

inputmode

IE *n/a* NN *n/a* Moz *n/a* Saf *n/a* Op *9* DOM *n/a*

可读/可写

这个Web Forms 2.0 扩展（完全继承自W3C XForms 1.0规范，请参阅*http://www.w3.org/ TR/xforms/sliceE.html*）会根据这一属性引导浏览器为某种书面语言显示合适的文本输入用户界面。请查阅W3C XForms 1.0文档，以获取更多详细信息。

input类型 email、password、text、url

示例
```
document.orderForm.searchText.inputmode = "hiragana";
```

值 每种书面语言均伴随着一个可选择的修订标记。这些修订标记通常相当于Unicode脚本，请参见*http://www.unicode.org/unicode/reports/tr24/*。

默认值 无。

labels

IE *n/a* NN *n/a* Moz *n/a* Saf *n/a* Op *9* DOM *n/a*

只读

这个Web Forms 2.0扩展返回一个与当前表单控件元素相关联的label元素对象引用所组成的数组（HTMLCollection）。

input类型 所有的可显示类型。

示例
```
var textboxLabels = document.getElementById("searchBox").labels;
```

值 label元素对象引用组成的数组。

默认值 空数组。

list

IE *n/a* NN *n/a* Moz *n/a* Saf *n/a* Op *9* DOM *n/a*

只读

这个Web Forms 2.0 扩展为输入类型的元素定制了一些预定义项，当然在input元素中也可以使用那些未包含在此列表中的文本项。这些预定义条目编写在option元素中，并包含在Web Forms 2.0的datalist元素之内。而list属性值就是包含这些预定义项的datalist元素的ID值。在Opera中，当input元素被选中时，这些预定义项会以选择列表的形式显示在元素下方。

input类型 date、datetime、datetime-local、email、month、number、range、text、time、url、week
示例 `var whichList = document.getElementById("opsystems").list;`
值 相关的datalist元素对象的引用。
默认值 无。

loop

IE *4* NN *n/a* Moz *n/a* Saf *n/a* Op *n/a* DOM *n/a*

可读/可写

如果用dynsrc属性指定了一段视频剪辑，那么就可通过loop属性控制该剪辑的播放次数。请参考img对象的loop属性以获得更多详细信息。

input类型 image

lowsrc

IE *4* NN *n/a* Moz *n/a* Saf *n/a* Op *n/a* DOM *n/a*

可读/可写

如果src属性指定的图像需要很长时间才能下载，那么这个属性可以提供一个低分辨率（或替代）图像的URL，以便文档空间进行加载。请参考img对象的lowsrc属性以获得更多详细信息。

input类型 image

max, min

IE *n/a* NN *n/a* Moz *n/a* Saf *n/a* Op *9* DOM *n/a*

可读/可写

这个Web Forms 2.0扩展允许为某些类型的input元素指定取值范围，这些数据类型包括：数字、日期、时间及文件上传等。如果用户输入的数据超出了这个范围，那么浏览器会设置ValidityState对象的rangeUnderflow或rangeOverflow属性，这些脚本可能会向用户提供更进一步的错误提示信息。

input类型 date、datetime、datetime-local、file、month、number、range、time、week
示例

```
document.getElementById("apptTime").min = "09:00";
document.getElementById("apptTime").max = "17:00";
```

值 number和range类型，整数或浮点数；data类型，根据ISO 8601指定的格式编写日期值，如2007-03-15；时间与日期的组合类型，根据ISO 8601指定的格式编写时间日期值，如2007-03-15T08:00:00；month类型，ISO格式的月份值，如2007-03；week类型，ISO 8601格式的日期值，如2007W3；file类型，正整数值，表明可随表单一同上传的文件数量。
默认值 无。

maxlength

IE *4* NN *n/a* Moz *all* Saf *all* Op *7* DOM *1*

可读/可写

此属性定义了在input元素的文本域中可输入字符的最大数量。在实际使用时，当用户输入的字符数量超过maxlength定义的值时，浏览器会发出蜂鸣声或给出其他提示。注意，maxlength和size属性之间并没有任

何内在的联系。如果maxlength允许的最大字符数超过了元素宽度可显示的字符数，那么浏览器会在元素中使用水平滚动（尽管对很多用户而言这显得比较笨拙），以便输入或修改文本内容。

input类型 password、text
示例 document.getElementById("ZIP").maxlength=10;
值 正整数。
默认值 无限制。

method

IE *n/a* NN *n/a* Moz *n/a* Saf *n/a* Op *9* DOM *n/a*

可读/可写

当点击此元素（须要将其type属性设置为submit）以提交封闭表单时，这个Web Forms 2.0扩展允许其提交目标URI与事件方法均与常规的表单不同。

input类型 image、submit
示例 document.getElementById("submitHere").method = "get";
值 方法类型字符串。
默认值 get

name

IE *4* NN *n/a* Moz *all* Saf *all* Op *7* DOM *1*

可读/可写

这是一个与表单控件相关的标识符。当表单被提交时，此属性的值将作为名称/值对的一部分而被提交至服务器。由于根据控件类型可以通过其他手段指定控件标注，因此从用户的角度来看，控件名称是被隐藏的。与此同时，脚本也可以使用控件名称来引用对象。虽然新标准倾向于使用id属性，但很多浏览器依然须要为表单控件设置name属性，以便允许表单控件的值被正常提交。

示例 document.orderForm.myButton.name = "Win32";
值 区分大小写的标识符字符串，必须遵循标识符命名规则：不包含空白、不能以数字作为起始字符、除下划线外不允许使用其他标点符号。
默认值 无。

pattern

IE *n/a* NN *n/a* Moz *n/a* Saf *n/a* Op *9* DOM *n/a*

可读/可写

当必须对用户输入进行验证时，这个Web Forms 2.0扩展允许指定一个正则表达式来进行判断。

input类型 email、password、text、url
示例 document.getElementById("partNum").pattern = "[A-Z][0-9]{7}"
值 正则表达式（与JavaScript中的正则表达式相同，不能放置在斜线符号中）。
默认值 无。

readOnly

IE *4* NN *n/a* Moz *all* Saf *all* Op *7* DOM *1*

只读

指明用户能否在页面上编辑表单元素。如果一个表单控件的readOnly属性被设置为true，那么脚本仍然可以修改其内容，但用户也许不能再做出任何修改。

input类型 password、text
示例

```
if (document.forms[0].myText.readOnly) {
  ...
}
```

值 布尔值：true | false。
默认值 false

replace

IE *n/a* NN *n/a* Moz *n/a* Saf *n/a* Op *9* DOM *n/a*

可读/可写

这个Web Forms 2.0扩展将一些指令与表单提交控件联系在一起，用以在表单提交后处理服务器返回数据的相关指令。目前有两种处理方式，默认方式是使用服务器的响应数据替换原文档。如果已为form元素的data属性指定了一个URL，那么浏览器会将返回的数值填充到表单中，而不是重新载入表单的初始化数据，这是第2种处理方式。

input类型 image、submit
示例 document.getElementById("submitMe").replace = "values";
值 两个常量值之一：document | values。
默认值 document

required

IE *n/a* NN *n/a* Moz *n/a* Saf *n/a* Op *9* DOM *n/a*

可读/可写

这个Web Forms 2.0 扩展属性表示提交时该input元素的值是否必须存在。如果元素未接收到任何值，那么就会将ValidityState对象的missingValue属性值设置为true。

input类型 checkbox、date、datetime、datetime-local、email、file、month、number、password、radio、range、text、time、url、week
值 布尔值：true | false。
默认值 false

selectedOption

IE *n/a* NN *n/a* Moz *n/a* Saf *n/a* Op *9* DOM *n/a*

只读

这个Web Forms 2.0扩展返回用户当前已选中的option元素（位于datalist元素之内）对象的引用。虽然Opera 9实现了这个属性，但它总是返回null。

input类型 date、datetime、datetime-local、email、month、number、range、text、time、url、week
示例 var optionElem = document.getElementById("opsystems").selectedOption;
值 被选中的option元素对象的引用。
默认值 无。

selectionEnd, selectionStart

IE *n/a* NN *n/a* Moz *all* Saf *1.3/2* Op *8* DOM *n/a*

可读/可写

这两个属性使得脚本能够在文本相关的输入元素内方便地获取和设置文本选择域的端点。它们的值均是从零开始的整数，对应于文本域中已输入的某个文本字符的位置。当这两个属性拥有相同的值时，其效果看起来就像一个文本插入点。例如，如果要将光标放在文本输入框的末尾，那么可以将这个两个值都设置为元素内文本的长度值，请参见textLength属性。在IE中，首先须要在元素内创建一个IE文本范围，然后调整这个范围的端点并选择这个文本范围才能达到相同的效果。

input类型 password、text
示例

```
var elem = document.forms[0].myPassword;
elem.selectionEnd = elem.textLength;
elem.selectionStart = elem.textLength;
```

值 正整数。
默认值 无。

size

IE 4 NN n/a Moz all Saf all Op 7 DOM 1 620

可读/可写

它表示输入框应该调整到一定的宽度以容纳此属性值所指定的字符数。在实际使用中，即使输入完全一样的字符，浏览器也无法总是准确预计合适的宽度，例如，密码框就属于这种情况。请参见第1章中有关input元素size属性的详细信息。对input对象而言，size和maxLength属性并不会相互影响。

input类型 password、text
示例 document.forms[0].myPassword.size = 12;
值 整数值。
默认值 20

src

IE 4 NN n/a Moz all Saf all Op 7 DOM 1

可读/可写

提供图像文件的相对或绝对URL地址，图像类型的输入元素已加载或将要加载该图像。请参考img对象的src属性以获得更多详细信息。

input类型 image

start

IE 4 NN n/a Moz n/a Saf n/a Op n/a DOM n/a

可读/可写

如果用dynsrc属性指定了一段视频剪辑，那么start属性可以控制启动视频播放所需的具体动作。请参考img对象的start属性以获得更多详细信息。

input类型 image

status

IE 4 NN n/a Moz n/a Saf n/a Op n/a DOM n/a

可读/可写

指明是否已选中或高亮显示对应元素。它与checked属性完全相同。

input类型 checkbox、radio
示例

```
if (document.forms[0].56KbpsBox.status) {
  ...
}
```

值 布尔值：true | false。
默认值 无。

step

IE n/a NN n/a Moz n/a Saf n/a Op 9 DOM n/a 621

可读/可写

这个Web Forms 2.0扩展可为input元素指定允许使用的递增值。如果设置了min或max属性，这两个属性就限制了用户输入数据的范围，否则递增值并不会受到限制。需要注意的是，对日期相关的元素而言，起始点是“1970-01-01T00:00:00.0Z”。

input类型 date、datetime、datetime-local、file、month、number、range、time、week
示例 document.getElementById(""apptTime").step = "900";

值 number和range类型，字符串形式的整数或浮点数；date、week和month类型，表示天、周或月的整数值；时间与日期的组合类型，则使用字符串形式的秒数。

默认值 60（datetime、datetime-local、time）；1（其他类型）。

target

IE *n/a* NN *n/a* Moz *n/a* Saf *n/a* Op *9* DOM *n/a*

可读/可写

当提交类型的输入元素被点击后，这个Web Forms 2.0扩展允许其内嵌表单的返回点与常规表单有所不同，使得其返回页面出现在另一个窗口或框架之中。

input类型 image、submit

示例
```
document.getElementById("submitMe").target = "_blank";
```

值 字符串形式的框架或窗口名称。

默认值 无（表示当前窗口或框架）。

type

IE *4* NN *3* Moz *all* Saf *all* Op *7* DOM *1*

只读

返回表单控件元素的类型。而返回值中的字符均为小写形式。有时可能须要遍历所有的表单元素，以找到特定的类型执行某些操作，例如，在将所有text类型的表单控件清空的同时不改变其他控件。请参见第1章中有关input元素type属性的说明，以便进一步详细了解每一种类型。

input类型 所有类型。

示例
```
if (document.forms[0].elements[3].type == "button") {
  // 处理“按钮”类型的输入元素
}
```

值 任意一个字符串常量：button | checkbox | file | hidden | image | password | radio | reset | select-multiple | select-one | submit | text | textarea（Mozilla之前的Netscape Navigator浏览器中无法访问image类型）。Web Forms 2.0浏览器（如Opera 9）增加了如下的附加类型：add | date | datetime | datetime-local | email | month | move-down | move-up | number | range | remove | time | url | week。

默认值 text

useMap

IE *4* NN *n/a* Moz *all* Saf *all* Op *7* DOM *1*

可读/可写

提供同一文档内某个map元素的URL地址，这个元素包含着将要应用于该图像的客户端图像映射热点区域及链接。请参考img对象的useMap属性以获得更多详细信息。

input类型 image

validationMessage

IE *n/a* NN *n/a* Moz *n/a* Saf *n/a* Op *9* DOM *n/a*

只读

如果表单控件在Web Forms 2.0规范下校验失败，那么这个Web Forms 2.0的扩展属性会返回一个由浏览器生成的消息。由于返回空字符串就表示校验正常，这使得这个属性对面向文本的表单控件而言更有意义。对一个button元素而言，这个属性总会返回一个空字符串。

input类型 所有类型，包括特殊的Web Forms 2.0类型。

值 空字符串。

默认值 空字符串。

validity

IE *n/a* NN *n/a* Moz *n/a* Saf *n/a* Op *9* DOM n/a

只读

这个Web Forms 2.0扩展返回一个ValidityState对象。对一个button元素而言，返回对象的所有成员的值均为false，仅有valid属性是ture。请参看ValidityState对象。

input类型 所有类型，包括特殊的Web Forms 2.0类型。

值 ValidityState对象。

默认值 ValidityState对象。

value

IE *3* NN *2* Moz *all* Saf *all* Op *7* DOM *1*

623

可读/可写

指明与表单控件相关的当前值，该值将与元素的名称/值对一起被提交至服务器端。所有的值均为字符串，并且脚本也是通过这个值将文本插入文本域中。浏览器会返回用户实际输入的全部字符（包括密码），因此可以在提交（或存储在cookie中）之前获取用户输入的密码以便进一步处理。对一个按钮类型的输入元素而言，value属性控制着这个表单控件的标注，即出现在按钮上的文字。但按钮输入元素的值并不会随表单一并提交。

input类型 除image外的所有类型。

示例

```
document.forms[0].myButton.value = "Undo";
```

值 字符串。

默认值 无。

valueAsDate

IE *n/a* NN *n/a* Moz *n/a* Saf *n/a* Op *9* DOM *n/a*

只读

这个Web Forms 2.0扩展按照标准形式返回表单控件的值，如下文所示。

input类型 datetime、date、time、week、month

值 字符串：对于datetime类型，表单中选定的值类似于JavaScript的date对象执行toString()方法之后的返回值，如“Sat, 26 Aug 2006 07:06:00 GMT-0700”；对于date类型，使用从UTC时间00:00开始的日期；对于time类型，使用自1970年1月1日开始的时间；对于week类型，使用UTC时间00:00星期一开始的星期数；对于month类型，使用选中月在UTC时间00:00里的第一天。对于所有的其他类型，均返回null。

默认值 null

624

valueAsNumber

IE *n/a* NN *n/a* Moz *n/a* Saf *n/a* Op *9* DOM *n/a*

只读

这个Web Forms 2.0扩展返回一个数字以表示输入的数据。日期和时间类型都被转换成从1970年1月1日UTC时间00:00开始的毫秒数。

input类型 所有文本输入类型。

值 数字或NaN。

默认值 NaN

width

请参见height。

willValidate

IE *n/a* NN *n/a* Moz *n/a* Saf *n/a* Op *9* DOM *n/a*

只读

这个Web Forms 2.0扩展返回一个布尔值，以表明表单控件是否能够达到Web Forms 2.0机制下的校验标准。但由于button元素并不是一个正确的类型，因此总是会返回false。

input类型 所有类型。
值 布尔值：true | false。
默认值 false

checkValidity( )

IE *n/a* NN *n/a* Moz *n/a* Saf *n/a* Op *9* DOM *n/a*

这个Web Forms 2.0方法会返回一个布尔值，以表明表单控件是否能够达到其校验标准，归根到底，就是判断validity.valid属性是否为true。

input类型 所有类型。
返回值 布尔值：true | false。按钮类型的元素总会返回true。
参数 无。

createTextRange( )

IE *4(Win)* NN *n/a* Moz *n/a* Saf *n/a* Op *n/a* DOM *n/a*

创建一个包含按钮标注文本的TextRange对象。请参见TextRange对象以获取更多详细信息。

input类型 password、text
返回值 TextRange对象。
参数 无。

dispatchChange( ), dispatchFormChange( )

IE *n/a* NN *n/a* Moz *n/a* Saf *n/a* Op *9* DOM *n/a*

这两个Web Forms 2.0方法为当前元素触发了change和formchange事件。因此，让onclick事件处理程序调用这两个方法之一，就可以将一个click事件转换成为一个change或formchange事件。

input类型 所有类型。
返回值 无。
参数 无。

handleEvent( )

IE *n/a* NN |4| Moz *n/a* Saf *n/a* Op *n/a* DOM *n/a*

```
handleEvent(event)
```

命令对象接受并处理一个事件，该事件的描述将作为参数传入此方法。此时document对象必须拥有一个事件处理程序来处理这种类型的事件。这个方法仅适用于Navigator 4。

input类型 button、checkbox、file、password、radio、reset、select-multiple、select-one、submit、text、textarea
返回值 无。
参数

event
一个Navigator 4的event对象。

select( )

IE *3* NN *2* Moz *all* Saf *all* Op *7* DOM *1*

选择所有显示在表单元素中的文本。在调用这个方法之前，通常首先须要在对应元素上进行聚焦。

input类型 password、text
返回值 无。
参数 无。

setCustomValidity( )

IE *n/a* NN *n/a* Moz *n/a* Saf *n/a* Op *9* DOM *n/a*

```
setCustomValidity([errorString])
```

这个Web Forms 2.0方法可以设置validity属性（它本身就是一个ValidityState对象）的customError布尔值。如果这个方法不支持某个元素类型，那么它将抛出一个NOT_SUPPORTED_ERR错误。

input类型 所有类型。
返回值 无。
参数

errorString

如果此参数为null或空字符串，将重新设置validity对象的customError属性，即表示表单控件无效。在当前会话期内，由于浏览器会记录一条错误信息，因此验证失败后会显示这条信息。

ins

IE *4* NN *n/a* Moz *all* Saf *all* Op *7* DOM *1*

ins对象对应于ins元素。

等价HTML元素 <ins>
对象模型引用方式 [window.]document.getElementById("*elementID*")
对象特定属性 cite、dateTime
对象特定方法 无。
对象特定事件 无。

cite, dateTime

IE *5(Mac)/6(Win)* NN *n/a* Moz *all* Saf *all* Op *7* DOM *1*

可读/可写

由于这两个属性已在IE内得以实现，因此本章已将它们罗列在共享属性之中。IE 5/Macintosh和W3C DOM浏览器只能在del和ins对象中按照W3C DOM的说明正确地实现这两个属性，但主流浏览器均未给它们赋予任何特殊的功能。请参见本章前文中有关共享属性cite和dateTime的说明。

isindex

IE *5* NN *n/a* Moz *all* Saf *all* Op *7* DOM *1*

isindex对象对应于一个古老的HTML元素——isindex。IE在其DOM中会将这个元素转换为一个文本类型的输入对象，甚至还会在它周围创建一个表单元素。在实际应用中请避免使用这个元素。

等价HTML元素 <isindex>
对象特定属性 prompt

prompt

IE *5(Mac)* NN *n/a* Moz *all* Saf *all* Op *7* DOM *1*

可读/可写

为文本输入域提供提示信息。

值 字符串。
默认值 无。

kbd

请参见abbr。

627

label

IE 4 NN n/a Moz all Saf all Op 7 DOM 1

label对象对应于label元素。

等价HTML元素 <label>
对象模型引用方式 [window.]document.getElementById("*elementID*")
对象特定属性 control、dataFld、dataFormatAs、dataSrc、form、forms、htmlFor
对象特定方法 无。
对象特定事件 无。

control

IE n/a NN n/a Moz n/a Saf n/a Op 9 DOM n/a

只读

这个Web Forms 2.0扩展返回一个与当前label元素相关的input元素的引用。

示例 var inpElem = document.getElementById("myLabel").control;
值 input元素对象的引用。
默认值 无。

dataFld

IE 4 NN n/a Moz n/a Saf n/a Op n/a DOM n/a

可读/可写

此属性用于IE中的数据绑定，以便将一个远程数据源的列名与输入元素的标注文本联系在一起。此时元素中还必须设定datasrc属性。如果dataFld和dataSrc属性均设置为空字符串，那么就会打破元素与数据源之间的绑定关系。须要注意的是，这一功能只能在IE 5/Mac组合的文件数据源中正常工作。

示例 document.getElementById("myLabel").dataFld = "labelText";
值 数据源数据列内区分大小写的标识符字符串。
默认值 无。

628

dataFormatAs

IE 4 NN n/a Moz n/a Saf n/a Op n/a DOM n/a

可读/可写

本属性用于IE浏览器中的数据绑定。对于来自于数据源的原始资料，浏览器根据此属性值决定将它们视为纯文本亦或是HTML标签。须要注意的是，这一功能只能在IE 5/Mac组合的文件数据源中正常工作。

示例 document.forms[0].myLabel.dataFormatAs = "html";
值 IE 4浏览器可识别两种有效的设定值：text | html。
默认值 text

dataSrc

IE 4 NN n/a Moz n/a Saf n/a Op n/a DOM n/a

可读/可写

为实现IE数据绑定功能，此属性用于指定页面内object元素的ID值，以便在进行远程数据访问时加载数据源对象。如果dataFld和dataSrc属性均设置为空字符串，那么就会打破元素与数据源之间的绑定关系。须要注意的是，这一功能只能在IE 5/Mac组合的文件数据源中正常工作。

示例 document.getElementById("myLabel").dataSrc = "DBSRC3";

值 数据源内区分大小写的标识符。
默认值 无。

form

IE *5(Mac)/6(Win)* NN *n/a* Moz *all* Saf *all* Op *7* DOM *1*

只读

返回文档树中临近的最外层form元素对象的一个引用。同一form元素中的多个label元素对象将引用相同的form元素对象。

示例
```
var theForm = document.getElementById("myLabel").form;
```
值 form元素对象的引用。
默认值 无。

forms

IE *n/a* NN *n/a* Moz *n/a* Saf *n/a* Op *9* DOM *n/a*

只读

针对与当前label元素所包含的输入元素相关的form元素，这个Web Forms 2.0扩展属性将返回一个由这些表单元素引用组成的数组（数组类型为NodeList）。

示例
```
var inpForms = document.getElementById("myLabel").forms;
```
值 input元素的引用。
默认值 无。

htmlFor

IE *4* NN *n/a* Moz *all* Saf *all* Op *7* DOM *1*

可读/可写

提供与标注相关联的input元素的id值，即for属性的值。它可以将label元素与一个特定的input元素进行绑定。

示例
```
document.getElementById("label3").htmlFor = "chkbox3";
```
值 字符串。
默认值 无。

layer

IE *n/a* NN *4* Moz *n/a* Saf *n/a* Op *n/a* DOM *n/a*

layer对象对应于layer和ilayer元素。但只能在以前的Navigator 4浏览器中找到这个对象。在Navigator 4中，任何将样式单属性position设置为absolute或relative的元素都会被转换为layer对象。

等价HTML元素 <ilayer>、<layer>
对象模型引用方式 [window.]document.*layerName*
对象特定属性 above、background、below、bgColor、clip、hidden、left、name、pageX、pageY、parentLayer、siblingAbove、siblingBelow、src、top、visibility、zIndex
对象特定方法 captureEvents()、handleEvent()、load()、moveAbove()、moveBelow()、moveBy()、moveTo()、moveToAbsolute()、releaseEvents()、resizeBy()、resizeTo()、routeEvent()

对象特定事件

处理程序	NN	其他	DOM	处理程序	NN	其他	DOM
blur	4	n/a	n/a	mouseout	4	n/a	n/a
focus	4	n/a	n/a	mouseover	4	n/a	n/a
load	4	n/a	n/a	mouseup	4	n/a	n/a

630

above, below

IE *n/a* NN *|4|* Moz *n/a* Saf *n/a* Op *n/a* DOM *n/a*

只读

它们分别返回在Z顺序叠放次序中位于当前元素之上或之下的可定位元素的引用。这两个属性运作在文档内所有可定位元素的上下文环境之中。如果当前元素在这个环境中位于堆栈的顶部，那么above属性会返回null。如果要将检验限定于一个单独的层空间内，则可以通过siblingAbove和siblingBelow来获取临近的高位和低位元素。关于如何调整特定对象的叠放次序，请参考moveAbove()和moveBelow()方法。

示例 `var nextHigher = document.myILayer.above;`
值 对象引用或null。
默认值 无。

background

IE *n/a* NN *|4|* Moz *n/a* Saf *n/a* Op *n/a* DOM *n/a*

可读/可写

这个属性保存了一个图像对象，该对象的src属性所指定的图像可以作为图层的背景图。换句话说，如果要改变背景图，就必须设定图层的background对象的src属性。

示例 `document.myIlayer.background.src = "images/newlogo.gif";`
值 一个图像对象的属性，例如src。
默认值 无。

bgColor

IE *n/a* NN *|4|* Moz *n/a* Saf *n/a* Op *n/a* DOM *n/a*

可读/可写

提供元素的背景色。此时既可以使用十六进制三元组也可以使用明语颜色名称来设置这个背景色，但此属性只会返回十六进制RGB颜色对应的十进制值。默认状态是将bgColor属性设置为null，从而使得其背景透明。

示例 `document.myIlayer.bgColor = "yellow";`
值 一个十六进制的三元组或明语颜色名称。目前可用的明语颜色名称请参见附录A。返回的值是与十六进制值等价的十进制值。将此值设置为null则背景完全透明。
默认值 null（透明）。

clip

IE *n/a* NN *|4|* Moz *n/a* Saf *n/a* Op *n/a* DOM *n/a*

可读/可写

为可定位元素定义一个裁剪区域。从它本身来看，这个属性更像是一个对象，可通过以下6个属性来调整其值：clip.top、clip.left、clip.bottom、clip.right、clip.width和clip.height。这样一来，可以根据需要调整各边或整体尺寸。此外，这里的每个值均以像素为单位。

631

示例 `document.myIlayer.clip.width = 150;`
值 整数值。
默认值 无。

hidden

IE *n/a* NN *|4|* Moz *n/a* Saf *n/a* Op *n/a* DOM *n/a*

可读/可写

指定对象在页面上是否可见。将对象隐藏后，它周围的内容并不会填满这个元素留下的空隙。

示例 `document.myIlayer.hidden = false;`
值 布尔值：true | false。
默认值 false

left

IE *n/a* NN *|4|* Moz *n/a* Saf *n/a* Op *n/a* DOM *n/a*

可读/可写

对可定位的元素而言，它定义了元素（包括元素内容及左侧填充、边框以及边距）的左边缘相对于临近的最外层块级容器的左边缘的偏移量。对于相对定位的层而言，会根据元素所在位置的左边缘来决定偏移量。

示例 `document.myIlayer.left = 45;`

值 整数值。

默认值 0

name

IE *n/a* NN *|4|* Moz *n/a* Saf *n/a* Op *n/a* DOM *n/a*

只读

这是一个层的标识符，可以将它指定给target属性，脚本也可以使用它引用对应的层。如果并未明确设置id属性值，那么Navigator会自动将name属性的值复制到id属性上。

示例

```
if (document.layers[2].name == "main") {
  ...
}
```

值 区分大小写的标识符，必须遵循标识符命名规则：不包含空白、不能以数字作为起始字符、除下划线外不允许使用其他标点符号。

默认值 无。

pageX, pageY

IE *n/a* NN *|4|* Moz *n/a* Saf *n/a* Op *n/a* DOM *n/a*

可读/可写

提供图层对象相对于整个文档左、上边缘的水平和竖直位置。

示例 `document.myIlayer.pageX = 400;`

值 整数值。

默认值 无。

parentLayer

IE *n/a* NN *|4|* Moz *n/a* Saf *n/a* Op *n/a* DOM *n/a*

只读

此属性返回HTML包含层次中临近的最外层图层的引用。如果该文档中只包含一个图层，那么它的parentLayer就是window对象。

示例

```
if (parentLayer != window) {
  ...
}
```

值 一个layer或window对象引用。

默认值 window

siblingAbove, siblingBelow

IE *n/a* NN *|4|* Moz *n/a* Saf *n/a* Op *n/a* DOM *n/a*

只读

在同一个parentLayer环境中，它们分别返回叠放次序中位于当前元素之上或之下的可定位元素的引用。如果在这个环境中当前元素位于堆栈的顶部，那么siblingAbove属性会返回null。如果要将临近元素的检测区域扩展到文档级上下文环境，那么请参见above和below。关于调整特定对象的叠放次序，请参考moveAbove()和moveBelow()方法。

示例 `var nextHigher = document.myILayer.siblingAbove;`

值 对象引用或null。

默认值 无。

src

IE *n/a* NN *|4|* Moz *n/a* Saf *n/a* Op *n/a* DOM *n/a*

可读/可写

此属性指明了一个外部文件的URL，当前元素已加载了该文件中的相关内容。如果要改变框架中的内容，直接指定一个新的URL即可。

在Navigator 4中，为这个属性指定一个新的URL地址并不会对内嵌图层（ilayer元素）起作用。相反，这会删除当前源文档，并遮盖其他页面的元素。因此，要避免在内嵌层中设置这个属性。load()方法能够起到相同的作用。

示例 `document.myIlayer.src = "swap2.html";`

值 字符串形式的完整或相对URL。

默认值 无。

top

IE *n/a* NN *|4|* Moz *n/a* Saf *n/a* Op *n/a* DOM *n/a*

可读/可写

对可定位的元素而言，它定义了元素（包括元素内容及顶部填充、边框及边距）的上边缘相对于临近的最外层块级容器的上边缘的偏移量。此时均以像素作为度量单位。如果当前元素是一个相对定位的内嵌层，那么就根据元素出现位置的上边缘来决定偏移量。

示例 `document.myIlayer.top = 50;`

值 整数值。

默认值 0

visibility

IE *n/a* NN *|4|* Moz *n/a* Saf *n/a* Op *n/a* DOM *n/a*

可读/可写

此属性指明已定位元素的可见性的具体状态。当元素的visibility属性被设置为hide（或通过CSS设置为hidden）时，它四周的内容并不会填满该元素留下的空白空间。如果将这个属性设置为CSS语法值（hidden、visible），那么在内部会将它们转换为JavaScript形式的值，并以将转换后的值作为返回值。

示例 `document.myIlayer.visibility = "hide";`

值 下面3个字符串常量之一：`hide | inherit | show`。

默认值 `inherit`

zIndex

IE *n/a* NN *|4|* Moz *n/a* Saf *n/a* Op *n/a* DOM *n/a*

可读/可写

对一个已定位的元素而言，它决定了在同一个父级容器内，这个元素相对于其他元素的叠放次序。请参见在线参考IV，它讲述了在多个容器内元素摆放层次之间的相互关系。

示例 `document.myIlayer.zIndex = 3;`

值 整数值。

默认值 0

captureEvents( )

IE *n/a* NN *|4|* Moz *n/a* Saf *n/a* Op *n/a* DOM *n/a* 634

```
captureEvents(eventTypeList)
```

命令浏览器拦截一个特定类型的事件，以防止它们到达其预定目标对象。而调用这个方法的对象必须拥有能够处理该事件类型的事件处理程序。

返回值 无。

参数

eventTypeList

一个使用逗号分隔的事件类型列表，对应的事件类型来自于可用的Event对象常量，如Event.CLICK或Event.MOUSEMOVE，请注意区分字符大小写。

handleEvent( )

IE *n/a* NN *|4|* Moz *n/a* Saf *n/a* Op *n/a* DOM *n/a*

```
handleEvent(event)
```

命令对象接受并处理一个事件，该事件的描述将作为参数传入此方法。此时，document对象必须拥有一个事件处理程序来处理这种类型的事件。

返回值 无。

参数

event

一个Navigator 4的event对象。

load( )

IE *n/a* NN *|4|* Moz *n/a* Saf *n/a* Op *n/a* DOM *n/a*

```
load("URL", newLayerWidth)
```

这个方法可以让一个layer对象加载新文档。在Navigator 4中，它无法正常工作在ilayer元素中。这时，图层会卸载已存在的文档，但它并不会按照预期加载新文档。因此，除非将它转换成一个layer元素，否则无法获取令人满意的处理结果。

返回值 布尔值：如果成功加载指定文档，则返回true。

参数

URL

某个文档的相对或绝对URL字符串，该图层将根据这个URL加载该文档。

newLayerWidth

整数值，表示该图层为适应新内容所需的图层宽度。

moveAbove( ), moveBelow( )

IE *n/a* NN *|4|* Moz *n/a* Saf *n/a* Op *n/a* DOM *n/a* 635

```
moveAbove(layerObject) moveBelow(layerObject)
```

这两个方法可以将当前图层所在的叠放位置改变到另一个与其他兄弟图层有关的特定位置。如果脚本无法准确获取当前图层叠放次序参考点的zIndex值，那么这两个方法就比较有用。此时可以使用moveAbove()方法将当前元素放置在参数传入的layer对象之上。

返回值 无。

参数

layerObject

指向另一个layer对象，它与当前图层拥有相同的父级容器。

moveBy( )

IE *n/a* NN |4| Moz *n/a* Saf *n/a* Op *n/a* DOM *n/a*

```
moveBy(deltaX, deltaY)
```

通过指定x、y两个轴向上的偏移量（单位为像素），这个方法可以很方便地改变当前元素的具体位置。如果只沿着其中一条轴线移动，那么请将另一个值设置为0。此时如果*deltaX*为正值，会让元素向右侧移动，而负值则是向左侧移动。如果*deltaY*为正值，会让元素向下移动，而负值则是向上移动。在setInterval()或setTimeout()方法的控制之下，这个方法能够十分方便地制造出一种路径动画，即根据时间的推移，让元素沿着一条直线路径不断移动。

返回值 无。

参数

deltaX

元素在水平方向上的位置改变值，单位为像素。

deltaY

元素在竖直方向上的位置改变值，单位为像素。

moveTo( ), moveToAbsolute( )

IE *n/a* NN |4| Moz *n/a* Saf *n/a* Op *n/a* DOM *n/a*

```
moveTo(x, y) moveToAbsolute(x, y)
```

这个方法可以很方便地将当前元素移动到一个特定的坐标点上。如果元素内嵌在另一个已定位的容器（例如，内嵌在layer元素之内的一个layer元素）之内，这个两种方法的差别才会显现出来。此时，moveTo()方法将使用父级容器的坐标系，而moveToAbsolute()则会使用整个页面的坐标系统。因此对于仅包含一个图层的页面而言，这两个方法的处理结果完全相同。

返回值 无。

参数

x

相对于参考容器（对moveTo而言是临近的最外层图层，而对moveToAbsolute而言则是页面）上边缘的正/负像素值。

y

相对于参考容器（对moveTo而言是临近的最外层图层，而对moveToAbsolute而言则是页面）左边缘的正/负像素值。

releaseEvents( )

IE *n/a* NN |4| Moz *n/a* Saf *n/a* Op *n/a* DOM *n/a*

```
releaseEvents(eventTypeList)
```

此方法与layerObj.captureEvents()相对，它根据参数列表中指定的特定事件名称，在文档级关闭对应的事件捕获。详情请参见在线参考VI。

返回值 无。

参数

eventTypeList

一个使用逗号分隔的事件类型列表，对应的事件类型来自于可用的Event对象常量，如Event.CLICK或Event.MOUSEMOVE，注意区分字符大小写。

resizeBy( )

IE *n/a* NN |4| Moz *n/a* Saf *n/a* Op *n/a* DOM *n/a*

```
resizeBy(deltaX, deltaY)
```

根据参数指定的像素值，这个方法可以很方便地改变当前元素的宽度和高度。如果只改变其中一个轴向数值，那么请将另一个值设置为0。*deltaX*为正值时会加宽元素宽度，而负值则会使元素变窄。与之类似，*deltaY*

为正值时会使元素更高，而负值则会缩短元素。调用此方法后，左、上边缘会固定不动，只会根据设定移动右、下边缘。

返回值 无。

参数

deltaX

元素在水平方向上的位置改变值，单位为像素。

deltaY

元素在竖直方向上的位置改变值，单位为像素。

resizeTo( )

IE *n/a* NN *4* Moz *n/a* Saf *n/a* Op *n/a* DOM *n/a*

```
resizeTo(x, y)
```

根据参数指定的像素尺寸，这个方法可以很方便地调整当前元素的宽度和高度。调用此方法后，左、上边缘会固定不动，只会根据设定移动右、下边缘。

返回值 无。

参数

x

元素的宽度，单位为像素。

y

元素的高度，单位为像素。

routeEvent( )

IE *n/a* NN *4* Moz *n/a* Saf *n/a* Op *n/a* DOM *n/a*


```
routeEvent(event)
```

通常在事件处理方法中使用这个方法，它会命令Navigator将事件传递至其预期目标对象。

返回值 无。

参数

event

一个Navigator 4的event对象。

legend

IE *4* NN *n/a* Moz *all* Saf *all* Op *7* DOM *1*

legend对象对应于legend元素。在使用时，legend元素必须内嵌在一个或一组表单控件相关的fieldset元素之中。

等价HTML元素 <legend>

对象模型引用方式 [window.]document.getElementById("*elementID*")

对象特定属性 align、form、forms

对象特定方法 无。

对象特定事件 无。

align

IE *4* NN *n/a* Moz *all* Saf *all* Op *7* DOM *1*

可读/可写

此属性用于控制fieldset元素内legend元素的对齐状态。但是在IE 4中的实际运行效果往往与预期有所差异。因此此时必须检查程序部署的客户机，以保证操作系统的设定与预期相同。

示例 document.getElementById("myLegend").align = "center";

值 以下字符串常量之一：bottom | center | left | right | top。

默认值 left

form

IE *5(Mac)/6(Win)* NN *n/a* Moz *all* Saf *all* Op *7* DOM *1*

只读

638

返回文档树中临近的最外层form元素对象的一个引用。同一个form元素中的多个legend元素对象将引用相同的form元素对象。

示例 `var theForm = document.getElementById("myLegend").form;`

值 form元素对象的引用。

默认值 无。

forms

IE *n/a* NN *n/a* Moz *n/a* Saf *n/a* Op *9* DOM *n/a*

只读

根据与当前legend元素所包含的输入元素相关的form元素，这个Web Forms 2.0扩展返回一个由这些表单元素引用所组成的数组（数组类型为NodeList）。

示例 `var inpForms = document.getElementById("myLegend").forms;`

值 input元素的引用。

默认值 无。

li

IE *4* NN *n/a* Moz *all* Saf *all* Op *7* DOM *1*

li对象对应于那些内嵌于ol或ul之内的li元素。

等价HTML元素

对象模型引用方式 [window.]document.getElementById("elementID")

对象特定属性 type、value

对象特定方法 无。

对象特定事件 无。

type

IE *4* NN *n/a* Moz *all* Saf *all* Op *7* DOM *1*

可读/可写

639

此属性指明了列表中每个条目的前导点号、数字或字符的样式。当li元素内嵌在ul元素中时，浏览器会显示点号，而当其内嵌在ol元素中时，则会显示数字或字符。值得注意的是，一旦使用脚本修改了列表中某个li对象的显示类型，那么这一改变会影响该列表中的所有li元素。

示例

```
document.getElementById("instruxListItem3").type = "a";
document.getElementById("point4").type = "square";
```

值

对于一个ol样式的列表而言，存在如下可选值：A、a、I、i、1。浏览器会自动完成排序工作，如下表所示。

类型	示例	类型	示例
A	A, B, C, ...	i	i, ii, iii, ...
a	a, b, c, ...	1	1, 2, 3, ...
I	I, II, III, ...		

对于一个ul样式的列表而言，则有如下可选值：circle、disc、square。

默认值 1和disc。注意，首先须要指定容器元素对应属性的值。

value

IE 4 NN n/a Moz all Saf all Op 7 DOM 1

可读/可写

指明了有序列表内某条目的序号。只有当li元素内嵌在一个ol元素中并且HTML代码也明确指定了ol的对应属性值时，才能在li元素中使用这个属性。在IE、Safari和Opera浏览器中，默认值为0，而在Mozilla中则为-1。如果为列表内某项设定了value值，那么后续的条目会以该值为起点逐步递增。但是一旦列表已经被完全显示在页面上，那么修改这个属性值也不会改变对应的序号。

示例

```
if (document.getElementById("step5").value > 0) {
  ...
}
```

值 整数值。

默认值 0或-1。

link

IE 4 NN n/a Moz all Saf all Op 7 DOM 1

link对象对应于link元素。首先须要注意一点，对于这里列出的很多对象属性，只有在HTML标签中显示初始化它们的对应元素属性后才能在脚本中进行访问。此外，由于在加载文档时会将那些元素属性作为一种输入指令，因此使用脚本为对应的对象属性指定新值往往没有任何效果，即使这些属性是可写的也不例外。它们包括href、rel、rev和type。还要提醒一点，在所有支持disabled属性的浏览器中，这个属性可以打开或关闭一个已链接的样式单。

等价HTML元素 <link>

对象模型引用方式 [window.]document.getElementById("*elementID*")

对象特定属性 charset、href、hreflang、media、rel、rev、sheet、styleSheet、target、type

对象特定方法 无。

对象特定事件

Event	IE	Opera 9	其他	DOM
error	•	•	–	–
load	•	•	–	–

charset

IE 4 NN n/a Moz all Saf all Op 7 DOM 1

可读/可写

指明链接另一端文本内容的编码方式。

示例

```
var charCoding = document.getElementById("myLink").charset;
```

值 是否区分大小写取决于字符集注册表，请访问以下链接获取更多详细信息：*http://www.iana.org/assignments/character-sets*。

默认值 无。

href

IE 4 NN n/a Moz all Saf all Op 7 DOM 1

可读/可写

提供元素的href属性所指定的URL。可以在IE/Windows中为这个属性指定一个新的URL地址，以便在事后加载一个替换的样式单。

示例

```
document.getElementById("styleLink").href = "altStyles.css";
```

值 由完整或相对URL路径组成的字符串。
默认值 无。

641

hreflang

IE 6 NN n/a Moz all Saf all Op 7 DOM 1

可读/可写

指明了链接目标文档所采用的语言代码。使用此属性时，需要首先设定对应元素或对象的href属性。

示例
```
document.getElementById("myLink").hreflang = "DE";
```
值 语言代码，不区分大小写。
默认值 无。

media

IE 4 NN n/a Moz all Saf all Op 7 DOM 1

可读/可写

指出href属性指定的目标文档内容所需要的输出设备。当浏览器能够对文档内容进行修正，以便适合不同的设备（如手持电脑、文本-语音转换器或模糊电视设备等）时，media属性才会大有作用。

示例
```
if (document.getElementById("link3").media == "print") {
    // 处理"pring"输出类型
}
```
值 以下字符串常量之一：all | print | screen。
默认值 all

rel

IE 4 NN n/a Moz all Saf all Op 7 DOM 1

可读/可写

此属性用于定义当前元素与外部链接目标之间的关系，它也可叫做前向链接。在使用中无论如何也不要将该属性与href定义的链接目标文档相混淆。尽管主流浏览器并未完全支持这个属性，但可以将它看作一种参数，并在脚本控制之下对其进行检查和修改。

示例
```
if (document.getElementById("link3").rel == "alternate stylesheet") {
    // 对替代的样式单进行处理
}
```
值 不区分大小写的单个字符串，但其选择范围仅限于HTML 4认可的标准链接类型。如：
642 alternate | appendix | bookmark | chapter | contents | copyright | glossary | help | index | next | prev | section | start | stylesheet | subsection
默认值 无。

rev

IE 4 NN n/a Moz all Saf all Op 7 DOM 1

可读/可写

此属性用于定义当前元素与链接目标之间的关系，通常也称之为“**反向链接**”。尽管主流浏览器并未完全支持这个属性，但可以将它看作一种参数，并在脚本控制之下对其进行检查和修改。

值 不区分大小写的单个字符串，但其选择范围仅限于HTML 4认可的标准链接类型。HTML 4认可的链接类型请参考上文中的rel属性。
默认值 无。

sheet

IE *n/a* NN *n/a* Moz *all* Saf *all* Op *n/a* DOM *2*

只读

当某个样式单被指定为link元素的目标时，此属性可以返回当前文档中对应styleSheet对象（W3C DOM称之为CSSStyleSheet）的引用。IE/Windows提供了一个类似的属性：styleSheet。

示例 `var extSS = document.getElementById("link3").sheet;`
值 styleSheet对象的引用。
默认值 无。

styleSheet

IE *5(Win)* NN *n/a* Moz *n/a* Saf *n/a* Op *n/a* DOM *n/a*

只读

这个属性虽然并不标准，但当某个样式单被指定为link元素的目标时，此属性可以方便地返回当前文档中对应styleSheet对象的引用。Mozilla和Safari也提供了一个类似的属性——sheet。

示例 `var extSS = document.getElementById("link3").styleSheet;`
值 styleSheet对象的引用。
默认值 无。

target

IE *4* NN *n/a* Moz *all* Saf *all* Op *7* DOM *1*

可读/可写

指明链接内容目标容器窗口或框架的名称。对于链接的样式单而言，此属性的默认值_self往往就是所要的设定值。

示例 `document.getElementById("link4").target = "frame2";`
值 窗口或框架的名称字符串，或下述字符串常量之一：_parent | _self | _top | _blank。
默认值 无。

type

IE *4* NN *n/a* Moz *all* Saf *all* Op *7* DOM *1*

可读/可写

针对从外部资源中加载的数据，这个属性可以指明其MIME类型声明。例如，对于一个外部样式单，其属性值就是text/css。这个信息通常设定在元素标签的type属性中。

示例

```
if (document.getElementById("myStyle").type == "text/css") {
    ...
}
```

值 MIME类型字符串。
默认值 无。

links

IE *3* NN *2* Moz *all* Saf *all* Op *7* DOM *1*

返回一个由a和area元素所组成的集合，这些元素均已指定了href属性从而使它们转化成链接。而此集合中的成员将按照它们在源代码中的次序进行排序。Navigator和Internet Explorer浏览器均允许使用数组符号来访问集合中的一个链接对象，例如document.links[0]、document.links["section3"]。如果须要将链接的名称作为索引值，那么一定要使用name属性的属性值，而不要使用id属性。而如果要通过锚链引用的id属性来访问其对象，那么此时可以使用document.all.*elementID*（仅适用于IE浏览器）或document.etElementById"*elementID*")。

对象模型引用方式 document.links
对象特定属性 length
对象特定方法 item()、amedItem()、ags()、rns()
对象特定事件 无。

644

length

IE 2 NN 3 Moz all Saf all Op 7 DOM 1

只读

返回集合中元素的数量。

示例
```
var howMany = document.links.length;
```
值 整数值。

item( )

IE 4 NN n/a Moz all Saf all Op 7 DOM 1

```
item(index) item(index[, subindex])
```

根据符合索引值（或IE/Windows中的任意索引及子索引值）的元素对象，返回一个单独的对象或一组对象集合。

返回值 一个对象或一个对象集合。如果并不存在与参数相匹配的对象，则返回null。

参数

index

当输入参数是一个从零开始的整数（根据Mozilla的需要）时，返回值是源代码内与某个具体项相对应的独立元素（内嵌在当前元素内）。如果参数为一个字符串（IE/Windows允许这种使用方式），则返回id或name属性与该字符串相符的元素组成的集合。

subindex

如果在IE/Windows中为第1个参数指定一个字符串值，那么可以将第2个参数指定为一个从零开始的索引值，这样就可以在集合中找到一个id或name属性与第1个参数的字符串相匹配的具体元素。

namedItem( )

IE 6 NN n/a Moz all Saf all Op 7 DOM 1

```
namedItem(IDOrName)
```

在W3C DOM浏览器中，根据符合参数字符串值的元素，此方法将返回单个对象或对象集合。

返回值 一个对象或对象集合。如果并不存在与参数相匹配的对象，则返回null。

参数

IDOrName

与目标元素的id或name属性包含相同值的字符串。

645

tags( )

IE 4 NN n/a Moz n/a Saf n/a Op 7 DOM n/a

```
tags(tagName)
```

从当前集合内的全部对象中，返回标签与*tagName*参数相匹配的对象集合。这个方法可将集合中的a和area元素区分开来。

返回值 对象集合。如果与参数相匹配的对象不存在，则返回一个长度为零的数组。

参数

tagName

包含元素标签的一个字符串，例如document.links.tags("a")。

urns( )

IE 5(Win) NN n/a Moz n/a Saf n/a Op n/a DOM n/a

```
urns(URN)
```

返回内嵌元素对象的集合，这些对象附加了一些行为，并且其URN符合输入的*URN*参数。

返回值 对象集合。如果与参数相匹配的对象不存在，则返回一个长度为零的数组。

参数

URN

一个包含本地或外部行为文件的统一资源命名的字符串。

LinkStyle

IE *4* NN *n/a* Moz *all* Saf *all* Op *7* DOM *2*

LinkStyle是W3C DOM的抽象对象，它可与link元素对象融合为一体。通过这种融合，Mozilla和Safari的link元素对象可以获得sheet属性，而通过sheet则可以提供一个链接到当前文档的styleSheet对象的引用。

对象模型引用方式 document.getElementById("*linkElementID*")
对象特定属性 sheet
对象特定方法 无。
对象特定事件 无。

sheet

IE *n/a* NN *n/a* Moz *all* Saf *all* Op *n/a* DOM *2*

只读

646

当某个样式单被指定为link元素的目标时，此属性可以返回当前文档中对应styleSheet对象（W3C DOM称之为CSSStyleSheet）的引用。IE/Windows为link元素提供了一个类似的属性——sytleSheet。

示例
```
var extSS = document.getElementById("link3").sheet;
```
值 styleSheet对象的引用。
默认值 无。

listing

IE *4* NN *n/a* Moz *n/a* Saf *n/a* Op *n/a* DOM *n/a*

listing对象对应于listing元素。

等价HTML元素 <listing>
对象模型引用方式 [window.]document.getElementById("*elementID*")
对象特定属性 无。
对象特定方法 无。
对象特定事件 无。

location

IE *3* NN *2* Moz *all* Saf *all* Op *7* DOM *n/a*

在每个窗口或框架中都存在一个location对象。对于窗口或框架已加载文档的URL，这个对象存储了关于这个URL的全部信息。如果为location对象的href指定一个新的URL，那么就会命令浏览器向该窗口或框架中载入一个新的页面。这是使用脚本加载新页面的主要方法，代码如下：

```
location.href = "newPage.html";
```

当然，在某个框架中也可以使用另一个框架的location对象引用来向该框架中加载新的文档，例如：

```
parent.otherFrameName.location.href = "newPage.html";
```

但如果这两个框架并不是来自于同一个域，或域相同但所在的服务器不同，那么出于安全考虑，会禁止通过脚本在一个框架中调用另一个框架的location对象。这样就可以防止恶意脚本在另一个框架中监视用户对外部站点的访问。在Navigator 4和Mozilla中，可以通过签名脚本解除这一安全约束，但须要用户首先给出明确的声明，才能访问脚本所在域之外的location对象信息。

647

作为一个与窗口相关的对象，location还不是W3C DOM Level 1或2正式规范的一部分。但脚本代码内已经广泛使用了这个对象及其属性，而且它们在将来很长一段时间内都会受到各种浏览器的支持。

对象模型引用方式 [windowRef.]location
对象特定属性 hash、host、hostname、href、pathname、port、protocol、search
对象特定方法 assign()、reload()、replace()
对象特定事件 无。

hash

IE 3 NN 2 Moz all Saf all Op 7 DOM n/a

可读/可写

指明URL地址中“#”后的部分内容，它指向文档中某个锚链位置。只有当用户已经明确导航到一个锚链并且尚未滚动到它时，这个属性才会包含具体的数据。须要注意的是，设置属性值时请不要包含“#”。

示例 location.hash = "section3";
值 字符串。
默认值 无。

host

IE 3 NN 2 Moz all Saf all Op 7 DOM n/a

可读/可写

提供当前文档所在服务器的主机名与端口组合。如果端口是URL的一部分，那么主机和端口之间使用冒号进行分隔，这与它们在URL中的形式一样。

示例

```
if (location.host == "www.megacorp.com:80") {
  ...
}
```

值 主机名字符串，可在其后加上冒号和端口号。
默认值 由服务器决定。

hostname

IE 3 NN 2 Moz all Saf all Op 7 DOM n/a

可读/可写

提供当前文档所在服务器的主机名，例如，包含服务器名称和域的一个地址。须要注意的是，hostname属性中并不包含端口号。

示例

```
if (location.hostname == "www.megacorp.com") {
   ...
}
```

值 主机名字符串（主机与域）。
默认值 由服务器决定。

href

IE 3 NN 2 Moz all Saf all Op 7 DOM n/a

可读/可写

提供窗口或框架中已加载文档的完整URL地址。为这个属性指定新的值可以向窗口或框架中载入载入新的文档，而IE还提供了一个等价的方法——window.navigate()。

示例 location.href = "http://www.megacorp.com";
值 由完整或相对URL路径组成的字符串。
默认值 无。

pathname

IE 3 NN 2 Moz all Saf all Op 7 DOM n/a

可读/可写

提供URL地址中包含的路径名信息。它由URL域名末字符之后的全部信息组成，其中还包括位于首位的斜线符号。

示例 `location.pathname = "/images/logoHiRes.gif";`
值 字符串。
默认值 无。

port

IE 3 NN 2 Moz all Saf all Op 7 DOM n/a

可读/可写

如果URL中存在端口号，那么这个属性就可以提供URL的端口信息。它包含URL域名末字符之后紧接着冒号的全部信息。注意，冒号并不是port属性值的一部分。

示例 `location.port = "80";`
值 字符串形式的数字值。
默认值 无。

protocol

IE 3 NN 2 Moz all Saf all Op 7 DOM n/a

可读/可写

提供URL地址中的协议信息。它包括URL内从起始部分到第1个冒号为止的所有信息。其中，典型值包括："http:"、"file:"、"ftp:"和"mailto:"。

示例

```
if (location.protocol == "file:") {
  // 将文档作为本地文件进行处理
}
```

值 字符串。
默认值 无。

search

IE 3 NN 2 Moz all Saf all Op 7 DOM n/a

可读/可写

提供URL地址中以“？”开头的URL编码信息部分。作为查询结果的文档可能也包含对应的查询部分，并将它作为window.location属性的一部分。在使用中也可以通过脚本来修改这个属性。这样一来，就可以将URL和搜索标准同时发送至服务器端。但是还必须预先知晓服务器所需的数据格式（通常是一些名称/值对），才能正确地完成这一功能。另外，通过将查询字符串附加到下一个页面的URL地址之后，可以在不同的页面间传递字符串数据。当附加的查询字符串并不会影响页面检索时，这些内容会进入新页面的location对象之中。然后，新页面中的脚本就可以读取并解析location.search属性，并将传入的值放置在变量中，以便其他处理过程进行调用。

示例 `location.search="?p=Tony+Blair&d=y&g=0&s=a&w=s&m=25";`
值 以“?”开头的字符串。
默认值 无。

assign( )

IE 3 NN 2 Moz all Saf all Op 7 DOM n/a

```
assign("URL")
```

曾经一度将不再向编程者开放这个方法，但到目前为止仍然可以使用这一方法。它的作用等效于向

location.href属性指定一个URL地址。因此，此处列出assign()方法仅仅只是为了保证本书的完整新，在实际中请勿使用这个方法。

返回值 无。

参数

650

URL

窗口或框架中将要加载的文档的完整或相对URL地址字符串。

reload( )

IE *4* NN *3* Moz *all* Saf *all* Op *7* DOM *n/a*

```
reload([unconditional])
```

硬重载与location对象有关的文档。这种重载会将页面中的表单元素设置为其默认值。如果要使用软重载，请使用history.go(0)。在默认情况下，reload()方法会执行一个条件获取动作，此时如果浏览器的缓存已打开，并且文档文件依然存在于文档之中，那么它会从缓存中重新获取该文件。如果要从服务器中重载文档，那么需要将方法的参数设置为true，以表明使用非条件获取。

返回值 无。

参数

unconditional

一个可选的布尔值。如果参数为true，那么浏览器会执行一个非条件重载，并从服务器上重载文档，否则将首先从缓存中加载数据。

replace( )

IE *4* NN *3* Moz *all* Saf *all* Op *7* DOM *n/a*

```
replace("URL")
```

将一个新文档加载至引用的窗口，并且在浏览器历史列表中使用新文档的URL替换当前文档的URL。因此通过这个方法可以从历史列表中删除某些并不希望出现的临时页面地址（即使按下“返回”按钮也不会再回到已被删除的页面），同时加入新载入的文档的地址。

返回值 无。

参数

URL

窗口或框架中将要加载的文档的完整或相对URL地址字符串。

locationbar

请参见directories。

map

IE *4* NN *n/a* Moz *all* Saf *all* Op *7* DOM *1*

map对象对应于map元素。

等价HTML元素 ap>

对象模型引用方式 [window.]document.getElementById("*elementID*")

651

对象特定属性 areas[]、name

对象特定方法 无。

对象特定事件 无。

areas[ ]

IE 4 NN n/a Moz all Saf all Op 7 DOM 1

只读

指明map元素中所有内嵌的area元素对象所组成的集合。

示例

```
for (var i = 0; i < document.getElementById("myMap").areas.length; i++) {
    oneMap = document.getElementById("myMap").areas[i];
    ...
}
```

值 area元素对像数组。

默认值 长度为0的数组。

name

IE 4 NN n/a Moz all Saf all Op 7 DOM 1

可读/可写

这是一个与客户端图像映射相关的标识符。该map元素所包含的area元素定义了图像的热点和对应的链接目标。此时，img元素的usemap属性将引用name属性为map元素指定的名称。这样就可以将一个图像与map定义绑定在一起。

示例 `document.getElementById("myMap").name = "altMap";`

值 区分大小写的标识符字符串，必须遵循标识符命名规则：不包含空白、不能以数字作为起始字符、除下划线外不允许使用其他标点符号。

默认值 无。

marquee

IE 4 NN n/a Moz 1.0.1 Saf n/a Op n/a DOM n/a

marquee对象对应于marquee元素。Mozilla以XUL扩展的形式实现这个对象。尽管已经将很多属性和方法暴露给脚本，但在实际使用中往往并不须要引用它们。如果对浏览器的内容实现感兴趣，那么可以从该模块的XML和JavaScript源代码中获得相关信息：*http://lxr.mozilla.org/seamonkey/source/layout/style/xbl-marquee/bl-marquee.xml*。

等价HTML元素 <marquee>

对象模型引用方式 [window.]document.getElementById("*elementID*")

对象特定属性 behavior、bgColor、dataFld、dataFormatAs、dataSrc、irection、height、ce、loop、scrollAmount、scrollDelay、trueSpeed、vspace、width

对象特定方法 start()、top()

对象特定事件

Event	IE	其他	DOM
bounce	•	–	–
finish	•	–	–
start	•	–	–

behavior

IE 4 NN n/a Moz 1.0.1 Saf n/a Op n/a DOM n/a

可读/可写

通过此属性可以设置marquee元素矩形框内HTML内容的运动方式。目前有3种运动方式可供选择。

示例 `document.getElementById("newsBanner").behavior = "slide";`

值

可选用以下几种方式之一，它们均不区分大小写。

`alternate`
相关内容在两侧边缘之间来回移动。
`scroll`
相关内容会在框内循环滚动，看起来它会从一端消失，然后又从另一端进入视野，`direction`属性可指定其滚动方向。
`slide`
相关内容首先会从一端进入视野，它走完全程后会静止在另一端，仍然由`direction`控制滚的方向。

默认值 `scroll`

653

bgColor

IE *4* NN *n/a* Moz *n/a* Saf *n/a* Op *n/a* DOM *n/a*

可读/可写

指明了元素的背景色。而样式单的`backgroundColor`属性并不能反映这个属性设定。即使使用明语颜色名称设置元素或对象的`bgcolor`属性，其返回值仍然是一个十六进制三元组。

示例 `document.getElementById("myBanner").bgColor = "yellow";`

值 一个十六进制的三元组或明语颜色名称。目前可用的明语颜色名称请参见附录A。

默认值 继承文档正文的背景色。

dataFld

IE *4* NN *n/a* Moz *n/a* Saf *n/a* Op *n/a* DOM *n/a*

可读/可写

此属性用于IE中的数据绑定，以便将一个远程数据源的列名同`marquee`元素的内容联系在一起。此时元素中还必须设定`datasrc`属性。如果`dataFld`和`dataSrc`属性均被设置为空字符串，那么就会打破元素与数据源之间的绑定关系。注意，只能在IE 5/Mac操作系统下实现文本文件数据源的绑定。

示例 `document.getElementById("myBanner").dataFld = "hotNews";`

值 数据源数据列内区分大小写的标识符字符串。

默认值 无。

dataFormatAs

IE *4* NN *n/a* Moz *n/a* Saf *n/a* Op *n/a* DOM *n/a*

可读/可写

本属性用于IE浏览器中的数据绑定。对于来自于数据源的原始资料，浏览器根据此属性值决定将它们视为纯文本亦或是HTML标签。

示例 `document.getElementById("myBanner").dataFormatAs = "text";`

值 IE浏览器可识别两种有效的设定值：`text | html`。

默认值 `text`

dataSrc

IE *4* NN *n/a* Moz *n/a* Saf *n/a* Op *n/a* DOM *n/a*

可读/可写

为实现IE数据绑定功能，此属性用于指定页面内`object`元素的ID值，以便在进行远程数据访问时加载数据源对象。如果`dataFld`和`dataSrc`属性均被设置为空字符串，那么就会打破元素与数据源之间的绑定关系。注意，只能在IE 5/Mac操作系统下实现文本文件数据源的绑定。

示例 `document.getElementById("myBanner").dataSrc = "DBSRC3";`

654

值 数据源内区分大小写的标识符字符串。

默认值 无。

direction

IE 4 NN n/a Moz 1.0.1 Saf n/a Op n/a DOM n/a

可读/可写

指明元素内容在元素空间内的滚动方向。

示例 `document.getElementById("myBanner").direction = "down";`

值 下列4种可用的滚动方向之一：`down | left | right | up`。

默认值 `left`

height, width

IE 4 NN n/a Moz 1.0.1 Saf n/a Op n/a DOM n/a

可读/可写

提供元素的高度和宽度，单位为像素。改变这些值后，页面的内容排列也会立刻随之发生改变。

示例 `document.getElementById("myBanner").height = 250;`

值 整数值。

默认值 无。

hspace, vspace

IE 4 NN n/a Moz n/a Saf n/a Op n/a DOM n/a

可读/可写

指出元素四周在水平和垂直方向上的边距值，单位为像素。`hspace`属性对元素左右两侧施加相等的影响，而`vspace`对元素上下两端的影响也是相同的。尽管这些边距与样式单设置的边距并不相同，但它们的视觉效果却是一致的。

示例

```
document.getElementById("myBanner").hspace = 5;
document.getElementById("myBanner").vspace = 8;
```

值 整数像素值。

默认值 `0`

loop

IE 4 NN n/a Moz n/a Saf n/a Op n/a DOM n/a

可读/可写

此属性可以设置`marquee`元素内容滚动显示的次数。当完成最后一次滚动后，相关内容会停留在一个固定的位置。由于连续的动画显示可能会干扰页面访问者，因此，相关内容滚动几次后就最好不要再让它继续滚动。

示例 `document.getElementById("myBanner").loop = 3;`

值 设置为正整数值时，完成该数值指定的滚动次数后就会停止下来。如果设置为`-1`，则不会停止滚动。

默认值 `-1`

scrollAmount

IE 4 NN n/a Moz 1.0.1 Saf n/a Op n/a DOM n/a

可读/可写

指明每次绘制滚动内容时的空间间隔。这个间隔越大，那么滚动过程中文本的移动速度越快。请同时参考`scrollDelay`属性。

示例 `document.getElementById("myBanner").scrollAmount = 4;`

值 正整数。

默认值 `6`

scrollDelay

IE 4 NN n/a Moz 1.0.1 Saf n/a Op n/a DOM n/a

可读/可写

指明每次绘制滚动内容时的时间间隔，单位为毫秒。这个时间延迟越大，那么滚动过程中文本的移动速度越慢。请同时参考scrollAmount属性。

示例 `document.getElementById("myBanner").scrollDelay = 100;`
值 正整数。
默认值 85（IE/Windows）；95（Macintosh）。

trueSpeed

IE 4 NN n/a Moz n/a Saf n/a Op n/a DOM n/a

可读/可写

当scrolldelay属性设定小于60毫秒时，此属性可以指明浏览器是否应该遵循这个时间间隔。默认值为false，这样可以防止错误设置导致滚动速度过快。

示例 `document.getElementById("myBanner").trueSpeed = "true";`
值 布尔值：true | false。
默认值 false

vspace

请参见hspace。

width

请参见height。

start( )

IE 4 NN n/a Moz 1.0.1 Saf 1.3/2 Op n/a DOM n/a

marquee元素内容滚动处于停止状态时，通过这个方法可以让它重新滚动起来。如果在一个已经停止的元素上调用这个方法，那么会同时触发onstart事件处理程序。

返回值 无。
参数 无。

stop( )

IE 4 NN n/a Moz 1.0.1 Saf 1.3/2 Op n/a DOM n/a

此方法可以让marquee元素停止滚动。而元素中的内容则会停留在方法调用时它所处的位置。如前文所述，可以调用start()方法以便重新开始滚动。

返回值 无。
参数 无。

MediaList

IE n/a NN n/a Moz all Saf 1.2 Op 9 DOM 2

MediaList是一个抽象对象，针对为sytleSheet对象指定的媒体，它代表W3C DOM中一个字符串名称集合。而sytleSheet对象的media属性所返回的值正是一个MediaList对象，须要注意的是，IE 6和7会错误地返回一个字符串值。对应的样式单媒体类型（如print、screen、aural等）可以通过link元素的media属性或style元素的@media规则来指定。但是除print、screen和默认的all之外的其他类型并未受到很好的支持，因此这个对象的详细信息并不是非常重要。

对象特定属性 length、mediaText
对象特定方法 appendMedium()、deleteMedium()、item()
对象特定事件 无。

length

IE *n/a* NN *n/a* Moz *all* Saf *1.2* Op *9* DOM *2*

只读

返回集合中数据项的数量。

示例
```
var howMany = document.styleSheets[0].media.length;
```
值 整数值。

mediaText

IE *n/a* NN *n/a* Moz *all* Saf *1.2* Op *9* DOM *2*

只读

返回由媒体名称所组成的字符串，不同的媒体名称之间使用逗号进行分隔。

示例
```
var allMedia = document.styleSheets[0].media.mediaText;
```
值 字符串。

appendMedium( ), deleteMedium( )

IE *n/a* NN *n/a* Moz *1.0.1* Saf *1.2* Op *9* DOM *2*

`appendMedium("mediumType") deleteMedium("mediumType")`

前者表示向列表中添加其他媒体类型，而后者可以从列表中删除某个类型。在早期的Mozilla浏览器（1.0之前）中，这个方法被错误的命名为append()和delete()。

返回值 无。

参数

mediumType

可识别的媒体类型字符串，如print、screen等。

item( )

IE *n/a* NN *n/a* Moz *all* Saf *1.2* Op *9* DOM *2*

`item(index)`

从集合中返回一个媒体名称字符串，该名称项的源代码次序可与参数传入的索引值相匹配。

返回值 字符串。

参数

index

一个从零开始的整数，依照源代码次序，对应于集合中的某个特定项。

menu

IE *4* NN *n/a* Moz *all* Saf *all* Op *7* DOM *1*

menu对象对应于menu元素。

等价HTML元素 <menu>
对象模型引用方式 [window.]document.getElementById("*elementID*")
对象特定属性 compact、ype
对象特定方法 无。
对象特定事件 无。

compact

IE 5(Mac)/6(Win) NN n/a Moz all Saf all Op 7 DOM 1

可读/可写

出于保证W3C DOM规范兼容性的需要，为menu对象提供了这个属性。然而主流浏览器并不会受这个属性的影响。

值 布尔值：true | false。

默认值 false

type

IE 6 NN n/a Moz n/a Saf n/a Op n/a DOM n/a

可读/可写

由于ol和ul元素对象之间的内部关系，在IE 6和IE 7中实现了这个属性。但menu元素对象会忽略这个属性。

menubar

请参见directories。

meta

IE 4 NN n/a Moz all Saf all Op 7 DOM 1

meta对象对应于meta元素。

等价HTML元素 <meta>

对象模型引用方式 [window.]document.getElementById("*elementID*")

对象特定属性 charset、content、httpEquiv、name、scheme、url

对象特定方法 无。

对象特定事件 无。

charset

IE 4 NN n/a Moz n/a Saf n/a Op n/a DOM n/a

可读/可写

根据href属性所指定的URL地址，指明其文件内容的字符编码方式。但这个属性并不会影响content属性所包含的名称/值对的charset属性。到目前为止，charset对文档几乎毫无影响。

示例

```
if (document.all.myMeta.charset == "csISO5427Cyrillic") {
  // 处理西里尔(cyrillic)字符集
}
```

值 是否区分大小写取决于字符集注册表，请访问以下链接获取更多详细信息：*http://www.iana.org/assignments/character-sets*。

默认值 由浏览器决定。

content

IE 4 NN n/a Moz all Saf all Op 7 DOM 1

可读/可写

此属性等价于名称/值对中的属性值。与之对应的元素content属性通常与name或http-equiv属性一起使用，而这两个属性则扮演了名称/值对中名称的角色。根据name或http-equiv属性值的不同，须要为content指定不同的属性值。有时，content还会包含多个值。在多个值的情况下，每个值之间使用分号进行分隔。某些属性值将组成有一定含义的名称/值对，例如用于刷新的meta元素的content属性值。在content的属性值中，第1个值为数字，它指定了在加载另一个文档前须要延迟的秒数；而第2个属性值则为一个名称/值对，它指定了延迟后所加载的URL文档地址。

尽管通过下面的这段代码可以在一个已加载的文档中改变content属性值，但是如果浏览器只依靠文档加载时的输入值，那么这段代码可能无法达到预期的效果。

示例

```
document.getElementById("refreshMeta").content = "5,http://www.giantco.com/basicindex.html";
```

值 字符串。

默认值 无。

httpEquiv

IE *4* NN *n/a* Moz *all* Saf *all* Op *7* DOM *1*

可读/可写

此属性等价于名称/值对中的名称。与之对应的http-equiv属性通常与content属性一起使用，此时这个content则扮演了名称/值对中“值”的角色。网页制作者也可以使用name属性来替代http-equiv属性，但每次只能使用这两者中的一个。另外，在脚本中只能调整meta元素标签中所使用的属性。而且一定要为content属性指定一个有效的值，以使得httpEquiv或name具有应有的含义。

示例

```
document.getElementById("refreshMeta").httpEquiv = "expires";
```

值 字符串。

默认值 无。

name

IE *4* NN *n/a* Moz *all* Saf *all* Op *7* DOM *1*

可读/可写

此属性是用于名称/值对的一个标识符，它是meta元素的一部分。通过name的属性值可以明确地表明使用该meta元素的目的，如“author”或“keywords”。值得注意的是，在同一个meta元素中只能为name或httpEquiv之一设置属性值，而不能同时对它们赋值。

示例

```
document.getElementById("detailMeta").name = "keywords";
```

值 字符串。

默认值 无。

scheme

IE *4* NN *n/a* Moz *all* Saf *all* Op *7* DOM *1*

可读/可写

它对应于scheme元素属性，但目前主流的浏览器尚未为它提供实际的功能。如果要了解该属性的设计目标，请参见第1章内meta元素中的scheme属性。

值 字符串。

默认值 无。

url

IE *4* NN *n/a* Moz *n/a* Saf *n/a* Op *n/a* DOM *n/a*

可读/可写

尽管曾经在IE浏览器中实现了这个属性，但目前看来官方已不再对这个属性提供任何支持。

mimeType

IE *5(Mac)* NN *3* Moz *all* Saf *all* Op *8* DOM *n/a*

mimeType对象附属于navigator对象。它代表了一个MIME类型规范。在从服务器加载外部文档数据之前，其属性可以让脚本获知浏览器是否能够处理一种特定的MIME类型。在选择了Navigator的“关于插件”菜单选项时，内部文档中涉及的所有属性都会显示出来。IE 5/Macintosh实现了这种机制，但IE/Windows则通过object元素对象使用了另外一种完全不同的系统来决定是否支持外部媒体类型。

对象模型引用方式 navigator.mimeTypes[i]
对象特定属性 description、enabledPlugin、suffixes、type
对象特定方法 无。
661 **对象特定事件** 无。

description

IE *5(Mac)* NN *3* Moz *all* Saf *all* Op *8* DOM *n/a*

只读

返回插件的详细描述说明。这一信息往往由开发者内嵌在插件之中。须要注意的是，即使对于同一个插件，在不同的操作系统之下其描述也可能迥然不同。

示例 `var descr = navigator.mimeTypes["video/mpeg"].description;`
值 字符串。
默认值 无。

enabledPlugin

IE *5(Mac)* NN *3* Moz *all* Saf *all* Op *8* DOM *n/a*

只读

返回一个plugin对象引用，浏览器将它对应的插件作为默认设置，以播放当前MIME类型的输入数据。然后就可以深入这个plugin对象的属性以获取其名称。

示例 `var plugName = navigator.mimeTypes["video/mpeg"].enabledPlugin.name;`
值 plugin对象引用。
默认值 无。

suffixes

IE *5(Mac)* NN *3* Moz *all* Saf *all* Op *8* DOM *n/a*

只读

返回一个与mimeType对象相关的文件后缀名称字符串列表，以说明该插件所能够支持的MIME类型，在此列表中，不同的名称之间使用逗号进行分隔。例如，QuickTime插件能够支持的视频文件后缀为：avi、vfw。

如果根据后缀名对浏览器中注册的所有mimeType对象进行遍历，那么还可以通过enabledPlugin属性获知是否已经为对应的mimeType对象安装了合适的插件。

示例 `var suff = navigator.mimeTypes["audio/mpeg"].suffixes;`
值 字符串。
默认值 无。

type

662 IE *5(Mac)* NN *3* Moz *all* Saf *all* Op *8* DOM *n/a*

只读

返回一个与mimeType对象相关的MIME类型字符串。例如，可以遍历所有的mimeType对象，并从中找到能够匹配特定MIME类型的对象（application/x-midi），然后再进一步检查浏览器是否支持并启用了该mimeType对象。

示例 `var MType = navigator.mimeTypes[3].type;`
值 字符串。
默认值 无。

MouseEvent

IE *n/a* NN *n/a* Moz *all* Saf *all* Op *7* DOM *2*

W3C DOM的MouseEvent是一个抽象对象，它包含每个W3C DOM鼠标事件实例所共享的属性和方法。这个对象从W3C DOM的Event和UIEvent对象中继承了相关特性。而且它的属性（例如点击时的坐标）和方法已

经与event对象融合在一起。请参见本章前文中有关event对象的讨论，以了解支持这一对象的特定属性和方法，以及其他事件类型如何继承这些属性与方法。

MutationEvent

IE *n/a* NN *n/a* Moz *all* Saf *all* Op *8* DOM *2*

W3C DOM的MutationEvent是一个抽象对象，它与那些关注于文档树改变的W3C DOM事件实例共享着相同的属性和方法。这个对象从W3C DOM的Event对象中继承了相关的特性。而且它的属性（例如修改时所涉及的节点的引用）和方法已经与event对象融合在一起。请参见本章前文中有关event对象的讨论，以了解支持这一对象的特定属性和方法，以及其他事件类型如何继承这些属性与方法。

NamedNodeMap

请参见相关属性。

navigator

IE *3* NN *2* Moz *all* Saf *all* Op *7* DOM *n/a*

从很多方面来看，navigator对象都代表了浏览器。而浏览器实际上位于文档对象模型领域之外。但即使如此，由于navigator对象允许脚本查看浏览器及其版本，因此它在脚本编程中依然扮演着一个重要的角色。除了Navigator和IE同时实现的几个关键属性之外，每种浏览器还扩展了一些属性以便为浏览器提供便利。尽管在IE中clientInformation对象与这个对象完全相同，但为了保持跨浏览器的兼容性，请在所有的浏览器中均使用navigator对象引用。

对象模型引用方式 navigator、[window.]navigator

对象特定属性 appCodeName、appMinorVersion、appName、appVersion、browserLanguage、cookieEnabled、cpuClass、language、mimeTypes[]、onLine、oscpu、platform、plugins[]、product、productSub、securityPolicy、systemLanguage、userAgent、userLanguage、userProfile、vendor、vendorSub

对象特定方法 javaEnabled()、preference()、taintEnabled()

对象特定事件 无。

appCodeName

IE *3* NN *2* Moz *all* Saf *all* Op *7* DOM *n/a*

只读

显示浏览器代号。几乎所有支持脚本的浏览器均会返回“Mozilla”，这是一个早期Navigator浏览器版本的代号，它实际上是“Mosaic”和“Godzilla”这两种免费浏览器名称的组合。尽管“Mozilla”是Netscape的企业吉祥物，但所有使用了Mosaic技术的浏览器（包括IE）均会返回Mozilla。

示例
```
var codeName = navigator.appCodeName;
```
值 Mozilla

默认值 Mozilla

appMinorVersion

IE *4* NN *n/a* Moz *n/a* Saf *n/a* Op *7* DOM *n/a*

只读

对于IE浏览器的后续版本而言，这个属性返回很多令人困惑的值，其中大部分无法用于判定浏览器的具体版本。例如，在Windows下的IE 5.x中，appVersion值为4.0，而appMinorVersion则返回小数中的第1个数字。

而在IE 6/Windows中，appMinorVersion却会返回一个字符串形式的版本代号或补丁代号，如;Q313675;。因此，使用时请倍加小心。

示例 `var subVer = navigator.appMinorVersion;`

值 字符串。

664 **默认值** 由浏览器版本决定。

appName

IE *3* NN *2* Moz *all* Saf *all* Op *7* DOM *n/a*

只读

显示浏览器的型号。

示例 `var isIE = navigator.appName == "Microsoft Internet Explorer";`

值 字符串。在Netscape Navigator、Mozilla和Safari中：Netscape；在IE中为Microsoft Internet Explorer，而在Opera 9中则为Opera。某些其他浏览器会返回类似的值以便与主流浏览器兼容。

默认值 由浏览器决定。

appVersion

IE *3* NN *2* Moz *all* Saf *all* Op *7* DOM *n/a*

只读

显示浏览器引擎的版本号，以及操作系统平台的简明信息（由userAgent返回的一些子信息）。例如，可能返回如下信息。

Internet Explorer：

```
4.0 (compatible; MSIE 6.0; Windows 98; Q312461)
4.0 (compatible; MSIE 7.0; Windows NT 5.1)
```

Mozilla：

```
5.0 (Macintosh; en-US)
```

Safari：

```
5.0 (Macintosh; U; PPC Mac OS X; en-us) AppleWebKit/418.8 (KHTML, like Gecko)
Safari/419.3
```

Opera：

```
9.01 (Windows NT 5.1; U; en)
```

须要注意的是，返回值中开头的版本号并不一定表明浏览器的真实版本，它实际上表明了浏览器的基本引擎版本。因此从IE 4到IE 7，返回的引擎号都是4.0，而Mozilla和Safari则总会返回5.0（即使它们使用了完全不同的引擎）。当然，浏览器的版本信息还可以在appVersion、userAgent或其他navigator对象属性中获取。但是，无论在哪种浏览器版本之中，在判断浏览器是否支持DOM或JavaScript特性时，都不要使用appVersion值的第一个字作为判断准则。尽管在第4版IE浏览器之前，这个属性可以正确地反映版本号，但现在早已不再是这种情况。

665 而且即使对象属性值中内嵌了具体的IE版本号，如MSIE X.XX，有时也会出现错误。因此鉴于以上的各种不定因素，使用navigator.userAgent属性来检查浏览器版本往往更为可靠。

示例 `var isVer4Min = parseInt(navigator.appVersion) >= 4;`

值 字符串。

默认值 由浏览器决定。

browserLanguage

IE *4* NN *n/a* Moz *n/a* Saf *n/a* Op *7* DOM *n/a*

只读

提供浏览器的默认书面语言类型。其他类型的浏览器请使用navigator.language属性。

示例 `var browLangCode = navigator.browserLanguage;`

值 字符串形式的语言代码，如“en”，请注意区分大小写。

默认值 由浏览器决定默认值。

cookieEnabled

IE *4* NN *n/a* Moz *all* Saf *all* Op *7* DOM *n/a*

只读

返回浏览器是否允许读写cookie数据。

示例

```
if (cookieEnabled) {
  setCookieData(data);
}
```

值 布尔值：true | false。

默认值 由浏览器的参数设定决定。

cpuClass

IE *4* NN *n/a* Moz *n/a* Saf *n/a* Op *n/a* DOM *n/a*

只读

返回当前客户端电脑的CPU型号参考信息。通常Intel处理器会返回“x86”，而PowerPC则会返回“PPC”。须要注意的是，这个信息只能说明基本的硬件类别，而无法表明操作系统、CPU速度或型号等信息。

示例

```
if (navigator.cpuClass == "PPC") {
  // 用于 PowerPC 客户端的专用脚本指令
}
```

值 字符串。

默认值 由客户机硬件决定。

language

IE *n/a* NN *4* Moz *all* Saf *all* Op *7* DOM *n/a*

只读

指明是为哪种书面语言类型而创建了当前浏览器。ISO 639语言代号方案已指明了具体的语言类型，如“en-us”。而IE浏览器则使用navigator.browserLanguage属性来提供这一信息。

示例

```
var mainLang = navigator.language;
```

值 不区分大小写的语言代码字符串。

默认值 由浏览器决定默认值。

mimeTypes

IE *5(Mac)* NN *4* Moz *all* Saf *all* Op *7* DOM *n/a*

只读

返回mimeType对象组成的数组，浏览器内已安装的插件能够对它们提供支持。出于兼容脚本语法的目的，IE/Windows支持这个属性，但它始终只会返回一个长度为零的空数组。请参见mimeType对象。

示例

```
var videoPlugin = navigator.mimeTypes["video/mpeg"].enabledPlugin;
```

值 mimeType对象数组。

默认值 由浏览器决定默认值。

onLine

IE *4* NN *n/a* Moz *1.7* Saf *n/a* Op *n/a* DOM *n/a*

只读

指明浏览器是设置为在线浏览还是离线浏览状态。在在线模式下，页面会调用一些实时服务器动作，而在离线模式下，则会避免这些调用。请使用这个布尔属性来构建条件判断语句。

示例

```
if (navigator.onLine) {
   document.write("<applet ...>");
   ...
}
```

值 布尔值：true | false。

默认值 true

oscpu

IE *n/a* NN *n/a* Moz *all* Saf *n/a* Op *n/a* DOM *n/a*

只读

返回一个有关客户机操作系统或中央处理器信息的字符串。对于不同的系统，这个属性值变化很大。Windows客户端大致可以分为NT和非NT两大类。其中，非NT包括Windows 95、98和ME，它们分别对应的oscpu值为Win95、Win98和Win 9X 4.9。而NT大类则包括Windows NT 4（WinNT 4.0）和Windows XP（Windows NT x.x）等。Macintosh系统会报告其CPU类型，但不会出现具体的Mac OS X信息或仅以简单的PPC、PPC Macos x表示。Unix系统则会同时报告其操作系统和CPU类型。oscpu实质上也是userAgent值的一部分。但在IE浏览器中，这一信息的输出格式和与之对应的cpuClass或userAgent属性格式并不相同。

667

示例

```
if (navigator.oscpu.indexOf("Win") != -1) {
   document.write("You are running a Windows computer.");
}
```

值 字符串。

默认值 由客户端系统决定。

platform

IE *4* NN *4* Moz *all* Saf *all* Op *7* DOM *n/a*

只读

返回浏览器所在的操作系统或硬件系统的名称。Windows 95/NT的对应值为Win32，而运行PowerPC CPU的Macintosh系统则为MacPPC。从目前已经测试的主要系统来看，主流浏览器均可以认可其返回值。通过在一个条件表达式内使用这个属性来判定客户端的基本性能，可以帮助页面根据具体设备优化其输出。

示例

```
if (navigator.platform == "Win32") {
   document.write("<link rel='stylesheet' type='text/css' href='css/stylePC.css'>");
}
```

值 字符串。

默认值 由客户端系统决定。

plugins[ ]

IE *5(Mac)* NN *3* Moz *all* Saf *all* Op *7* DOM *n/a*

只读

返回浏览器可以识别的插件对象所组成的集合。这样一来就可以使用脚本来判断浏览器是否支持某种外部媒体类型。在Windows中，IE 4及后续版本的IE浏览器均已实现了这个属性，但它完全是一种摆设，任何情况都只会返回一个长度为零的空数组。请参见mimeType和plugin对象。

示例 `var plugInCount = navigator.plugins.length;`

值 plugin对象引用数组。

默认值 无。

product, productSub

IE *n/a* NN *n/a* Moz *all* Saf *all* Op *n/a* DOM *n/a*

只读

返回一个字符串以指明其浏览器的软件引擎。在Mozilla和Safari中，product属性返回Gecko，而productSub

属性则会返回一个字符串形式的开发版本号。

示例 `var prod = navigator.product;`

值 字符串。

默认值 由浏览器决定。

securityPolicy

IE *n/a* NN *4* Moz *all* Saf *n/a* Op *n/a* DOM *n/a*

只读

在Navigator 4中，它返回一个表示浏览器加密级别的字符串，以表示浏览器所遵守的输入或输出加密策略。随着美国加密技术出口法规的放松，Mozilla在其所有浏览器版本中仅引用了一种加密类型。但实际上Mozilla只会为此属性返回一个空字符串。

值 字符串。

默认值 无。

systemLanguage

IE *4* NN *n/a* Moz *n/a* Saf *n/a* Op *n/a* DOM *n/a*

只读

指明操作系统所使用的默认书面语言代码。如果在页面中存在多语言内容，那么就可以根据这个属性来插入特定语言的内容。

示例

```
if (navigator.systemLanguage == "nl") {
  // document.write()某些荷兰语内容
}
```

值 语言代码，不区分大小写。

默认值 通常与浏览器的默认语言相同，例如，“en”就是美国IE浏览器的默认语言。

userAgent

IE *3* NN *2* Moz *all* Saf *all* Op *7* DOM *n/a*

只读

提供浏览器软件的相关信息，如版本号、操作系统平台和品牌等。由于appVersion和appName等属性只能提供部分信息（这些信息往往还会出现错误），这实际上是关于浏览器的最完整的信息组合。根据不同的浏览器，userAgent属性的返回值与下面这些示例比较类似：

```
Mozilla/4.0 (compatible; MSIE 7.0; Windows NT 5.1)
Mozilla/5.0 (Macintosh; U; PPC Mac OS X Mach-O; en-US; rv:1.8.0.6) Gecko/20060728
Firefox/1.5.0.6
Mozilla/5.0 (Macintosh; U; PPC Mac OS X; en-us) AppleWebKit/418.8 (KHTML, like Gecko)
Safari/419.3
Opera/9.01 (Windows NT 5.1; U; en)
```

由于实际的返回值与浏览器、版本，以及客户端所使用的代理服务器都有关，因此在使用时不要完全依靠数据的长度或某些数据所处的位置来做出判断。此时应该使用indexOf()方法来判断特定的字符串是否存在。例如，如果只须获取IE的版本号，那么可以使用下面这个方法：

```
function readIEVersion() {
   var ua = navigator.userAgent;
   var IEOffset = ua.indexOf("MSIE ");
   return parseFloat(ua.substring(IEOffset + 5, ua.indexOf(";", IEOffset)));
}
```

示例

```
if (navigator.userAgent.indexOf("MSIE") != -1) {
   var isIE = true;
}
```

值 字符串。
默认值 由浏览器决定。

userLanguage

IE 4 NN n/a Moz n/a Saf n/a Op 7 DOM n/a

只读

提供浏览器的默认书面语言，它由操作系统的用户配置设定有关。在默认情况下，这个属性等于browserLanguage属性。

示例
```
var userLangCode = navigator.userLanguage;
```
值 不区分大小写的语言代码字符串。
默认值 由浏览器决定默认值。

userProfile

IE 4 NN n/a Moz n/a Saf n/a Op n/a DOM n/a

只读

userProfile属性实际上是一个对象，它使得脚本可以访问保存在访问者用户配置文件中的个人信息，目前只有Win32系统下的IE 4到IE 6能够提供这一信息。请参见userProfile对象。

示例
```
navigator.userProfile.addReadRequest("vcard.displayname");
navigator.userProfile.doReadRequest("3", "MegaCorp Customer Service");
var custName = navigator.userProfile.getAttribute("vcard.displayname");
navigator.userProfile.clearRequest();
if (custName) {
  ...
}
```
值 userProfile对象引用，而IE 7会返回null。
默认值 由浏览器决定默认值。

vendor, vendorSub

IE n/a NN n/a Moz all Saf all Op n/a DOM n/a

只读

返回一个字符串以指明使用这个浏览器引擎的公司。例如，Netscape 7返回一个简单的“Netscape”字符串，Safari则会返回“Apple Computer, Inc.”，而Mozilla所拥有的Firefox浏览器会返回一个空字符串。vendorSub属性则会返回字符串形式的版本细节。须要注意的是，Safari并不支持vendorSub属性。

示例
```
if (parseFloat(navigator.vendorSub, 10) >= 6.2) {
   // 能够达到 Netscape 的最低需求
}
```
值 字符串。
默认值 由浏览器决定。

javaEnabled( )

IE 4 NN 3 Moz all Saf all Op 7 DOM n/a

返回一个布尔值，以表示浏览器是否已经开启了Java支持。显然，在一个无法执行脚本的浏览器中，这个方法并不能提供任何帮助，但如果可以使用脚本，那么它的确可以说明用户是否已经在浏览器设置中关闭了Java功能。

返回值 布尔值：true | false。
参数 无。

preference( )

IE n/a NN 4 Moz all Saf n/a Op n/a DOM n/a

```
preference(name[, value])
```

通过Navigator 4和Mozilla中的签名脚本，可以访问用户参数设定中的很多变量。这些标签中甚至还包括大量

的详细配置项，例如用户是否允许下载图片，或者是否启用了样式单等。在企业部署环境之内，网络管理员可以在脚本中使用这些设定以安装并控制用户设定。如果要获取更多有关于参数设定的信息，请访问Netscape开发者站点：*http://developer.netscape. com/docs/manuals/communicator/preferences/*。

返回值 参数值。

参数

name

字符串形式的参数名称，如general.always_load_images。

value

可选参数，可以设置由第一个输入参数指明的浏览器参数。

taintEnabled( )

IE *4* NN *3* Moz *all* Saf *n/a* Op *7* DOM *n/a*

返回一个布尔值，以表示浏览器是否已经开启了对数据污点（data tainting）的支持。这个安全机制从未在Navigator中完全实现过，但在新版本的Navigator和Mozilla浏览器中依然保存着这个方法，以便保证浏览器的后向兼容性。而出于相同的兼容性考虑，IE和Opera也包含了这个方法，不过它们始终会返回false。

返回值 布尔值：true | false。

参数 无。

nobr

IE *4* NN *n/a* Moz *all* Saf *all* Op *7* DOM *1*

nobr对象对应于nobr元素。

等价HTML元素 <nobr>

对象模型引用方式 [window.]document.getElementById("*elementID*")

对象特定属性 无。

对象特定方法 无。

对象特定事件 无。

Node

IE *5* NN *n/a* Moz *all* Saf *all* Op *7* DOM *1*

Node是W3C DOM中的一种抽象对象，它代表了文档中的基本内容构建块。在W3C DOM模型中所能够找到的所有分块内容都是节点，它们包括：标签间未命名的连续字符串、标签元素、名称/值属性对，以及有一定特殊作用的元素，如注释、DOCTYPE声明，甚至文档本身。

Node节点拥有大量的属性和方法，它们中的大部分都能够将节点与其四周的节点关联起来。下文中列出的所有以大写字母书写的属性都是常量或位掩码。在文档中，脚本能够读取并控制的所有对象都是基础Node对象的后代，这意味着与DHTML脚本能够协同工作的大多数常用内容承载对象都共享了Node节点的属性和方法集。为实现HTML元素所需要的功能，它们才产生了附加的属性和方法。

W3C DOM的这种节点化模型确立了文档内容中不同分块间的继承关系，尽管这一模型与第一代DOM和Microsoft DOM的快捷HTML相比存在更为繁琐的地方，但它的确可以表达出概念框架和颗粒度设定。而目前发展的最终目标则是为XML和HTML文档提供一种统一应用模型。

对象模型引用方式 Node

对象特定属性 ATTRIBUTE_NODE、CDATA_SECTION_NODE、DOCUMENT_FRAGMENT_NODE、COMMENT_NODE、DOCUMENT_NODE、DOCUMENT_POSITION_CONTAINED_BY、DOCUMENT_POSITION_CONTAINS、DOCUMENT_POSITION_DISCONNECTED、DOCUMENT_POSITION_FOLLOWING、DOCUMENT_POSITION_IMPLEMENTATION

_SPECIFIC、DOCUMENT_POSITION_PRECEDING、DOCUMENT_TYPE_NODE、ELEMENT_NODE、ENTITY_NODE、ENTITY_REFERENCE_NODE、NOTATION_NODE、PROCESSING_INSTRUCTION_NODE、TEXT_NODE、attributes、baseURI、childNodes、firstChild、lastChild、localName、namespaceURI、nextSibling、nodeName、nodeType、nodeValue、ownerDocument、parentNode、prefix、previousSibling、textContent

对象特定方法 appendChild()、cloneNode()、compareDocumentPosition()、getFeature()、getUserData()、hasAttributes()、hasChildNodes()、insertBefore()、isDefaultNamespace()、isEqualNode()、isSameNode()、isSupported()、lookupNamespaceURI()、lookupPrefix()、normalize()、removeChild()、replaceChild()、setUserData()

NodeFilter

IE *n/a* NN *n/a* Moz *1.0.1* Saf *1.3/2* Op *8* DOM *2*

NodeFilter节点为NodeIterator和treeWalker节点提供了一种机制，使用这种机制可以决定是否将某些节点或节点类型纳入到一个特定的节点列表之中。一旦要求NodeIterator和treeWalker节点指向这个序列中的下一节点，那么它们就会调用accept()方法。而NodeFilter对象还是两类常量集合的持有者，在很多构建方法或用户自定义的过滤方法中会使用这些常量。本章的treeWalker对象说明中已介绍了一个示例，而document.createTreeWalker()方法也说明了如何使用这些常量。

673

对象模型引用方式 NodeFilter

对象特定属性 FILTER_ACCEPT、FILTER_REJECT、FILTER_SKIP、SHOW_ALL、SHOW_ATTRIBUTE、SHOW_CDATA_SECTION、SHOW_COMMENT、SHOW_DOCUMENT、SHOW_DOCUMENT_FRAGMENT、SHOW_DOCUMENT_TYPE、SHOW_ELEMENT、SHOW_ENTITY、SHOW_NOTATION、SHOW_PROCESSING_INSTRUCTION、SHOW_TEXT

对象特定方法 accept()

对象特定事件 无。

accept( )

IE *n/a* NN *n/a* Moz *1.0.1* Saf *1.3/2* Op *8* DOM *2*

accept(*nodeReference*)

返回一个整数，以指明节点是否将要被NodeIterator或TreeWalker对象列表所包含。一旦调用了对象的指针移动方法，那么这些对象就会自动调用这个方法。

返回值 整数值，每个值都拥有一个与NodeFilter对象相关的对应常量值：1（NodeFilter.FILTER_ACCEPT）；2（NodeFilter.FILTER_REJECT）；3（NodeFilter.FILTER_SKIP）。

参数

nodeReference

文档树节点的引用。当NodeIterator或treeWalker对象调用这个方法时会自动向它传递这个参数。

NodeIterator

IE *n/a* NN *n/a* Moz *n/a* Saf *1.3/2* Op *8* DOM *2*

NodeIterator对象是一个实时节点列表，该列表内的节点都符合document.createNodeIterator()方法所定义的规则。这个列表是按照源代码顺序排序的一个简单节点列表，但其内的节点间并没有任何的父子关系。createNodeIterator()方法声明了列表的起始节点，并且还指明了应该从列表中过滤掉哪些节点。有关节点过滤的信息，请参考treeWalker对象。

674

NodeIterator对象实际上在列表中维持了一种指针对象。在移动指针时，这个对象的方法使得脚本可以访

问列表中的前一个或后一个节点。如果在NodeIterator创建之后脚本又修改了文档树，那么对应的修改会自动影响NodeIterator中的节点顺序。

对象模型引用方式 NodeIteratorReference
对象特定属性 expandEntityReference、filter、root、whatToShow
对象特定方法 detach()、nextNode()、previousNode()
对象特定事件 无。

expandEntityReference, filter, root, whatToShow

IE *n/a* NN *n/a* Moz *n/a* Saf *1.3/2* Op *8* DOM *2*

只读

请参见treeWalker对象的同名属性。

detach( )

IE *n/a* NN *n/a* Moz *n/a* Saf *1.3/2* Op *8* DOM *2*

将当前NodeIterator节点与文档树分离。一旦调用了这个方法，那么将无法再访问列表中的指针。

返回值 无。
参数 无。

nextNode( ), previousNode( )

IE *n/a* NN *n/a* Moz *n/a* Saf *1.3/2* Op *8* DOM *2*

将内部的NodeIterator指针向前（nextNode()）或向后（backward()）移动一个位置，同时返回移动后指针所指向的节点引用。

返回值 文档树中一个节点的引用。
参数 无。

NodeList

IE *5* NN *n/a* Moz *all* Saf *all* Op *7* DOM *1*

NodeList是W3C DOM的一个抽象对象，它代表了一个由所有节点组成的节点集合。任何返回节点集合的W3C DOM属性或方法都将返回一个NodeList对象。例如，Node对象的childNodes属性及Element对象的getElementsByTagName()方法均会返回NodeList对象。JavaScript将NodeList集合视为一个数组，因此它拥有一个为大家所熟知的length属性。脚本就可以通过整数数组索引或item()方法来引用数组内的任意一个节点项。

但某些节点类型也拥有其自身的集合，例如，节点属性集合是NamedNodeMap，而HTML元素节点集合则是HTMLCollection。这些集合对象拥有额外的属性和方法，它们也仅对集合内的节点有效。例如，由于文本节点（一种最简单的Node对象类型）的所有属性都无法容纳标识符，因此在NodeList对象中也不存在一个能够通过节点ID来引用节点的方法。但HTMLCollection对象（包含更为复杂的HTMLElement节点）则包含一个名为namedItem()的方法，它可以让脚本使用ID或整数索引来引用集合中的一个对象。在实际的脚本编程中，通过比较集合对象的属性和方法可以很明显地区分不同集合对象，请参见本章中attributes和image对象的相关说明。从另一个角度而言，W3C DOM术语并不是脚本中的一个重要因素。

对象特定属性 length
对象特定方法 item()

noframes, noscript

IE *5* NN *n/a* Moz *all* Saf *all* Op *7* DOM *1*

noframes对象对应于noframes元素，而noscript对象则对应于noscript元素。

等价HTML元素 <noframes>、<noscript>
对象模型引用方式 [window.]document.getElementById("*elementID*")
对象特定属性 无。
对象特定方法 无。
对象特定事件 无。

676

Notation

IE *n/a* NN *n/a* Moz *1.0.1* Saf *n/a* Op *8* DOM *1*

Notation是W3C DOM的一种抽象对象，它代表了DOCTYPE声明中的一部分。而且DOCTYPE所引用的公共ID和系统ID也可以通过Notation对象的对象来读取。

object

IE *4* NN *n/a* Moz *all* Saf *all* Op *7* DOM *1*

object对象对应于object元素。这是将其他媒体文件或外部输入嵌入一个文档的高级方法，当然，通过插件或IE/Windows中的ActiveX控件也可以达到相同的目的。在不同的浏览器品牌和版本中，这一对象的实现深度和实现质量也有所不同。另外，它的实现方式与IE/Windows中的ActiveX控件加载最为一致。

等价HTML元素 <object>
对象模型引用方式 [window.]document.getElementById("*elementID*")
对象特定属性 align、alt、altHtml、archive、BaseHref、border、classid、code、codeBase、codeType、contentDocument、data、dataFld、dataSrc、declare、form、height、hspace、name、object、standby、type、useMap、vspace、width
对象特定方法 无。
对象特定事件

处理程序	IE	Mozilla	Safari	Opera	DOM
abort	–	–	•	–	•
error	•	–	•	–	•
load	–	–	•	–	•

align

IE *4* NN *n/a* Moz *all* Saf *all* Op *7* DOM *1*

可读/可写

677

它决定了在四周的文本内容中object元素如何进行对齐。大多数属性值设定了元素与四周文本之间的竖直关系。例如，如果要将元素的底部与四周文本的基线对齐，那么align属性值应该是baseline。元素也能够沿着左或右边缘漂浮，以便使得四周的文本能够完全包围住该元素。

示例 document.getElementById("myObject").align = "absmiddle";
值 以下任意一个字符串对齐常量：absbottom | absmiddle | baseline | bottom | right | left | none | texttop | top。
默认值 bottom

alt

IE *6* NN *n/a* Moz *n/a* Saf *n/a* Op *n/a* DOM *n/a*

可读/可写

它指定了当浏览器无法下载图形或正在等待图像下载时，显示在页面中object元素所在位置的文字内容。估计微软公司是为了使用object元素（而不是img元素）显示图像才实现了这个并不标准的属性。

示例 document.getElementById("logoDisplay").alt = "MegaCorp Logo";

值 字符串。
默认值 无。

altHtml

IE *4* NN *n/a* Moz *n/a* Saf *n/a* Op *n/a* DOM *n/a*

可读/可写

当对象或小程序加载失败时，会显示它提供的HTML内容。它可以是一个文本消息、静态图片或任意其他适合于该场景的HTML内容。对于属性名称大小写的问题，IE浏览器的不同版本之间存在着一定的冲突。例如，在Win32版本中须要使用“altHtml”，而在Mac版本中则须要使用“altHTML”。

示例
```
document.getElementById("myObject").altHtml = "<img src='objectAlt.gif'>";
```
值 放置在引号之中的任意字符串，可以包含HTML标签。
默认值 无。

archive

IE *6* NN *n/a* Moz *all* Saf *all* Op *7* DOM *1*

可读/可写

它对应于`object`对象的`archive`属性，但当前主流浏览器实际上尚未赋以该属性任何功能。

值 字符串。
默认值 无。

BaseHref

IE *4(Win)* NN *n/a* Moz *n/a* Saf *n/a* Op *n/a* DOM *n/a*

只读

返回包含当前`object`对象的文档的URL地址。通常这个值与当前窗口的`location.href`完全相同。与其他属性名称不同的是，它以大写字符开头，因此使用时要特别注意大小写。

示例
```
var where = document.getElementById("myObject").BaseHref;
```
值 URL字符串。
默认值 当前文档的URL。

border

IE *6* NN *n/a* Moz *all* Saf *all* Op *7* DOM *1*

可读/可写

控制元素边框的厚度，单位为像素。为了保证跨浏览器的兼容性，请使用字符串作为属性值。

示例
```
document.getElementById("myObject").border = "5";
```
值 字符串形式的数字。
默认值 0

classid

IE *4* NN *n/a* Moz *n/a* Saf *n/a* Op *n/a* DOM *n/a*

只读

此属性提供了对象实现的URL。在IE中，这个URL可以指向客户机的*CLSID*目录（“`clsid:`”），其中保存着所有已注册的ActiveX控件（如DirectX或Media Player）的ID。一个比较奇怪的现象是，IE 4/Macintosh将这个属性命名为`classID`，但在IE 4.5中却须要使用小写名称，即`classid`。尽管HTML 4包含与之对应的元素属性，但W3C DOM还是忽略了这个对象属性。

示例
```
if (document.getElementById("soundObject").classid ==
    "clsid:83A38BF0-B33A-A4FF-C619A82E891D"){
    // 处理对应的声音对象
}
```

值 字符串，其中包括“clsid:”协议以指明本地ActiveX控件。
默认值 无。

code

IE *4* NN *n/a* Moz *all* Saf *all* Op *7* DOM *1*

只读

在浏览器支持的情况下，如果使用object元素代替applet元素，那么这个属性可以提供object元素code属性所指定的Java小程序的类文件名称。

示例

```
if (document.getElementById("clock").code == "Y2Kcounter.class") {
    // 处理已找到的类文件
}
```

值 字符串形式的小程序类文件名，注意区分大小写。
默认值 无。

codeBase

IE *4* NN *n/a* Moz *all* Saf *all* Op *7* DOM *1*

只读

针对由code或classid属性指定的类文件，此属性指明了保存该类文件的文件夹路径。要注意一点，codebase属性并不包含类文件的名称，它仅仅说明其所在路径。

示例

```
if (document.getElementById("clock").codeBase == "classes") {
    // 处理已找到的类文件目录
}
```

值 区分大小写的路径名称，往往与保存当前HTML文档的文件夹相关。
默认值 无。

codeType

IE *4* NN *n/a* Moz *all* Saf *all* Op *7* DOM *1*

可读/可写

对于classid属性所指出的对象类型，它给出了相应的提示信息。此信息可以帮助浏览器为需要多媒体播放器或插件的资源提前做好准备。如果codetype属性为空，那么浏览器会继而查看type属性的设定，当然该属性往往与data属性指定的URL的链接内容有关。如果这两个属性均未指定属性值或其值为空，浏览器才会从已下载的资源中获取内容类型信息。

示例

```
document.getElementById("gameTime").codeType = "application/x-crossword";
```

值 MIME类型，且不区分大小写。MIME类型目录可从此处获得：*http://www.iana.org/assignments/media-types/*。
默认值 无。

contentDocument

IE *n/a* NN *n/a* Moz *all* Saf *all* Op *8* DOM *2*

只读

指向object元素所创建的一个文档节点。

值 文档节点引用或null。
默认值 null

data

IE *4* NN *n/a* Moz *n/a* Saf *n/a* Op *n/a* DOM *n/a*

只读

此属性提供了引用文件的URL，object元素将加载该文件的数据信息。如果指定了codebase属性，那么将

根据它的属性值计算文件的相对URL路径，否则文件路径只能是与所在文档的URL有关的相对路径。

示例 `var objDataURL = document.getElementById("soundEffect").data;`

值 字符串形式的完整或相对URL。

默认值 无。

dataFld

IE *4* NN *n/a* Moz *n/a* Saf *n/a* Op *n/a* DOM *n/a*

可读/可写

此属性用于IE浏览器下的数据绑定，它将远程数据源的列名称与对象属性决定的object元素属性联系起来。此时元素中还必须设定datasrc属性。如果dataFld和dataSrc属性均被设置为空字符串，那么就会打破元素与数据源之间的绑定关系。

示例 `document.getElementById("myObject").dataFld = "streamURL";`

值 数据源数据列内区分大小写的标识符。

默认值 无。

dataSrc

IE *4* NN *n/a* Moz *n/a* Saf *n/a* Op *n/a* DOM *n/a*

可读/可写

为实现IE数据绑定功能，此属性用于指定页面内object元素的ID值，以便在进行远程数据访问时加载数据源对象。如果dataFld和dataSrc属性均被设置为空字符串，那么就会打破元素与数据源之间的绑定关系。

示例 `document.getElementById("myObject").dataSrc = "DBSRC3";`

值 数据源内区分大小写的标识符。

默认值 无。

declare

IE *6* NN *n/a* Moz *all* Saf *all* Op *8* DOM *1*

可读/可写

它对应于元素的declare属性。在主流浏览器中，这个属性对元素中的文档内容没有任何影响。

值 布尔值：`true | false`。

默认值 `false`

form

IE *4* NN *n/a* Moz *all* Saf *all* Op *7* DOM *1*

只读

返回包含当前元素的form元素的一个引用。只有当对象作为一个表单控件时这个属性才适用于object对象。

值 对象引用或null。

默认值 无。

height, width

IE *4* NN *n/a* Moz *all* Saf *all* Op *7* DOM *1*

可读/可写

提供元素的高度和宽度，单位为像素。改变这些值后，页面的内容排列也会立刻随之发生变化。

示例 `document.getElementById("myObject").height = 250;`

值 整数值。

默认值 无。

hspace, vspace

IE *4* NN *n/a* Moz *all* Saf *all* Op *7* DOM *1*

可读/可写

指出object元素四周在水平和垂直方向上的边距值，单位为像素。hspace属性对元素左右两侧施加的影响

是相等的，而vspace对元素上下两端的影响也完全相同。尽管这些边距与样式单设置的边距并不相同，但它们的视觉效果却是一致的。

示例

```
document.getElementById("myObject").hspace = 5;
document.getElementById("myObject").vspace = 8;
```

值 字符串形式的整数像素值。

默认值 0

name

IE *4* NN *n/a* Moz *all* Saf *all* Op *7* DOM *1*

可读/可写

这是一个与object元素相关的标识符。如果这个object对象是表单中的一员，那么当表单被提交时name属性的值将作为名称/值对的一部分被提交至服务器。

示例

```
document.getElementById("myObject").name = "company";
```

值 区分大小写的标识符字符串，必须遵循标识符命名规则：不包含空白、不能以数字作为起始字符、除下划线外不允许使用其他标点符号。

默认值 无。

object

IE *4* NN *n/a* Moz *n/a* Saf *n/a* Op *n/a* DOM *n/a*

只读

这个属性提供了已加载的对象的引用，以便明确地访问object元素的DOM属性，从而避免object对象的属性名称与其加载对象的内部属性名发生混淆。例如，如果object元素中加载的代码包含一个名为hspace属性，那么脚本在调用 document.getElementById ("reader").object.hspace时将获取其内部属性，而不是HTML元素的hspace属性。这个object属性通知JavaScript解析器访问元素内加载的对象的内部属性，而不是HTML元素自身的属性。

示例

```
var objCode = document.getElementById("reader").object.code;
```

值 对象引用。

默认值 无。

standby

IE *5(Mac)/6(Win)* NN *n/a* Moz *all* Saf *all* Op *8* DOM *1*

只读

在加载过程中，这个属性会承担起alt属性的责任并显示相关文本信息。但目前它对object元素没有任何影响。

值 字符串。

默认值 无。

type

IE *4* NN *n/a* Moz *all* Saf *all* Op *7* DOM *1*

可读/可写

在对象加载外部数据前，为浏览器提供该数据的参考MIME类型。如果codeType属性为null，那么浏览器就会查看这个type属性。

示例

```
if (document.getElementById("myObject").type == "image/jpeg") {
   ...
}
```

值 MIME类型，且不区分大小写。MIME类型目录可从此处获得：*http://www.iana.org/assignments/media-types/*。

默认值 无。

useMap

IE 6 NN n/a Moz all Saf all Op 7 DOM 1

683

可读/可写

提供同一文档内某个map元素的URL地址，这个元素包含着客户端图像映射热点区域及链接。这个值由一个“#”及object元素的usemap属性中的映射名称组成。

示例 `document.getElementById("logoViewer").useMap = "#altMap";`
值 一个以“#”开头，后跟map元素名称的字符串。
默认值 无。

vspace

请参见hspace属性。

width

请参见height属性。

ol

IE 4 NN n/a Moz all Saf all Op 7 DOM 1

ol对象对应于ol元素。

等价HTML元素
对象模型引用方式 [window.]document.getElementById("elementID")
对象特定属性 compact、start、type
对象特定方法 无。
对象特定事件 无。

compact

IE 4 NN n/a Moz all Saf all Op 7 DOM 1

可读/可写

如果将compact属性设置为true，它会命令浏览器以一种更为紧密的形式显示列表中的各项内容。但这个属性对主流浏览器没有任何影响。

示例 `document.getElementById("myOL").compact = true;`
值 布尔值：true | false。
默认值 false

684

start

IE 4 NN n/a Moz all Saf all Op 7 DOM 1

可读/可写

通过此属性可以为ol元素中的条目序列指定一个起始数字。当条目序列必须被主体内容打断时，此功能会带来很多便利。尽管此属性值为一个数字，但对应的阿拉伯数字、罗马数字或字母均可用于显示该值。如果并未指定具体的属性值，那么不同的浏览器会指定各自的默认值，请参见下文。

示例 `document.getElementById("sublist2").start = 6;`
值 整数值。
默认值 1（IE、Opera），-1（Mozilla），0（Safari）。

type

IE 4 NN n/a Moz all Saf all Op 7 DOM 1

可读/可写

此属性指明了列表中每个条目的前导数字或字符的样式。

示例 `document.getElementById("instruxList").type = "a";`

值

目前可用的值包括：A | a | I | i | 1。浏览器会自动完成排序工作，如下表所示。

类型	示例	类型	示例
A	A, B, C, ...	i	i, ii, iii, ...
a	a, b, c, ...	1	1, 2, 3, ...
I	I, II, III, ...		

默认值 尽管默认的显示效果与属性值为1时相同，但实际上并未指定具体默认值。

optgroup

IE *5(Mac)/6(Win)* NN *n/a* Moz *all* Saf *all* Op *7* DOM *1*

optgroup对象对应于optgroup元素，它必须内至于一个select元素之中，并且环绕一些option元素。浏览器对这个元素的支持细节请参见第1章中的optgroup元素。另外，disabled属性对这个对象也有效，并且还会影响到内嵌的option元素的禁用状态。

等价HTML元素 `<optgroup>`

对象模型引用方式 `[window.]document.getElementById("elementID")`

对象特定属性 label

对象特定方法 无。

对象特定事件 无。

label

IE *5(Mac)/6(Win)* NN *n/a* Moz *all* Saf *all* Op *9* DOM *1*

可读/可写

此属性对应于optgroup元素的label属性。

值 字符串。

默认值 无。

option

IE *3* NN *2* Moz *all* Saf *all* Op *7* DOM *1*

option对象对应于option元素，它必须内嵌在一个select元素中。由于select对象会将其包含的option对象作为其选项数组的一员，因此在引用option对象时往往须要首先使用其父级select对象。在现代浏览器中，也可以通过ID直接引用一个option对象。另外，在IE 4及后续浏览器及所有的W3C DOM浏览器中，disabled属性也是有效的。

自从Netscape 3及IE 4开始，后来的浏览器均可以修改同一个select对象中的选项集合。例如，如果须要使用另外一个相同长度的列表替换已存在的选择，那么可以针对select对象的options数组中的各个对象，将它们的text、value和selected属性值进行简单的调整。但是如果新列表所包含的选项数量与原有数量不同，那么删除已存在的option对象并插入新的对象可能是一种更好的方法。通过option对象的构造方法，可以同时创建多个对象实例，然后就可以将这些对象插入options数组中的具体位置。可以通过如下语法构造option对象：

```
var newOpt = new Option("text", "value", isDefaultSelectedFlag, isSelectedFlag);
```

而下面的这个方法则说明了如何改写select对象的选项列表：

```
function setSelect(selectElemObj) {
    // 移除现有的选项
    selectElemObj.options.length = 0;
    // 逐条新建选项并进行赋值
    selectElemObj.options[0] = new Option("Hercule Poirot", "poirot", false, false);
```

```
    selectElemObj.options[1] = new Option("Miss Marple", "marple", false, false);
    ...
}
```

699

在实际的构造环境中，往往会以一个对象数组的形式向页面传递所需要的构造参数值，这样就可以通过一个for循环来填充新的选项。另外，也可以参考options.add()方法（仅限于IE、Mozilla和Opera）及select.add()方法（IE 5及后续版本和W3C DOM浏览器）以了解修改选项的其他方法。

等价HTML元素 <option>

对象模型引用方式

```
[window.]document.formName.selectName.options[i]
[window.]document.forms[i].elements[i].options[i]
[window.]document.getElementById("elementID")
```

对象特定属性 defaultSelected、form、index、label、selected、text、value

对象特定方法 无。

对象特定事件 无。

defaultSelected

IE 3 NN 2 Moz all Saf all Op 7 DOM 1

可读/可写

它指明了元素标签中是否设置了selected属性。将当前的selected与defaultSelected属性值比较后，就可以知道select控件的状态是否在文档加载之后发生了改变。改变这个属性并不会影响当前的selected状态。

示例

```
var listItem = document.forms[0].selector.options[2];
if (listItem.selected != listItem.defaultSelected) {
    // 对状态改变作出响应
}
```

值 布尔值：true | false。

默认值 由HTML标签属性决定。

form

IE 4 NN n/a Moz all Saf all Op 7 DOM 1

只读

返回包含select元素及其选项的form对象的引用。

示例

```
var theForm = document.getElementById("myOption3").form;
```

值 form对象引用。

默认值 无。

687

index

IE 3 NN n/a Moz all Saf all Op 7 DOM 1

只读

返回当前option对象在select元素选项集合中的整数索引值（从零开始计数）。而select对象的selectedindex属性将返回当前已选中选项的索引值。

示例

```
var whichItem = document.getElementById("myOption3").index;
```

值 整数值。

默认值 无。

label

IE 5(Mac)/6(Win) NN n/a Moz all Saf all Op 9 DOM 1

可读/可写

此属性对应于option元素的label属性。这个属性主要用于层叠菜单，但它只有在IE 5/Mac中才能返回与text

属性相同的值，而且在其他环境中往往无法正常运作。

值 字符串。

默认值 无。

selected

IE 3 NN 2 Moz all Saf all Op 7 DOM 1

可读/可写

这个属性可用于判定某个列表选择是否已经被用户选中，一旦被选中就意味着其属性值将同表单一起被提交至服务器端。通过脚本也可以修改这个值以便选中某项。如果要找到已选中的选项，那么使用select对象的selectedIndex属性可能更为有效，这比遍历所有的option对象以查找selected属性值为true的对象要好的多。但一旦允许在select元素中选择多个选项，那么在这种情况下只能遍历所有的选项才能找到所有已选中的条目。

示例
```
document.forms[0].selectList.options[3].selected = true;
```
值 布尔值：true | false。

默认值 false

text

IE 3 NN 2 Moz all Saf all Op 7 DOM 1

可读/可写

这个属性能够提供与option元素相关的文本内容，因此它为读取或修改option元素的文本节点内容给出了一种方法。这些文本位于元素的起始和结束标签之间，并且将出现在select元素的选项之中。通过option对象的value属性，可以存储、获取或修改与这个列表项相关的隐藏值。

示例
```
var list = document.forms[0].selectList;
var listItemText = list.options[list.selectedIndex].text;
```
值 字符串。

默认值 无。

value

IE 4 NN 4 Moz all Saf all Op 7 DOM 1

可读/可写

此属性给出了一个与option元素相关的值。如果option元素已经设置了value属性，那么这个对象属性将返回其属性值，否则返回一个空字符串。

示例
```
var itemValue = document.forms[0].selectList.options[2]value;
```
值 字符串。

默认值 无。

options

IE 3 NN 2 Moz all Saf all Op 7 DOM 1

由内嵌在select对象中的option元素组成的数组。W3C DOM Level 2将这个对象归入HTMLOptionsCollection集合对象之中。

对象模型引用方式
```
[window.]document.formName.selectName.options
[window.]document.forms[i].elements[i].options
[window.]document.getElementById("selectElementID").options
```
对象特定属性 length、selectedIndex

对象特定方法 add()、item()、namedItem()、remove()、tags()、urns()

length

IE 3 NN 2 Moz all Saf all Op 7 DOM 1

可读/可写

返回集合中元素的数量。如果将这个属性的值设置为零，那么就会清空select元素中的所有选项。

示例 `var howMany = document.forms[0].mySelect.options.length;`

值 整数值。

689

selectedIndex

IE n/a NN n/a Moz all Saf all Op 7 DOM n/a

只读

返回一个从零开始的整数，以表示目前选中了集合中的哪个选项。这个属性是select元素对象selectedIndex属性的一个不规范的替代属性。

示例 `var whichOne = document.forms[0].mySelect.options.selectedIndex;`

值 一个从零开始的整数。

add( )

IE 4 NN n/a Moz n/a Saf all Op 7 DOM n/a

`add(elementRef[, index])`

将一个由createElement()方法创建的现有元素添加至当前集合。此时只能向集合中添加option类型的元素。除非在第2个参数中指定了索引值，否则在默认情况下均会将新元素添加到集合的最后一项中。如果要将一个新元素附加至select对象，请参考下面的这段示例代码：

```
var newElem = document.createElement("option");
newElem.text = "Freddie";
newElem.value = "Freddie Mercury";
document.forms[1].rockers.options.add(newElem);
```

在这段代码中，首先创建了一个普通的option对象。然后使用所需要的值填充其属性，并且将这个新元素添加至select元素。

在option对象的有关讨论中已经示范了如何实现跨平台以及如何保证后向兼容性。此外，请参考select.add()方法，这一方法可以在所有W3C DOM浏览器中正常工作。

返回值 无。

参数

elementRef

一个完整的元素对象的引用，通常由createElement()方法生成。

index

一个可选的整数，表示新元素应该放置在集合中的哪个位置。

item( )

IE 4 NN n/a Moz all Saf all Op 7 DOM 1

`item(index[, subindex]), item(index)`

根据符合索引值的元素对象，返回一个单独的对象或一组对象集合。

返回值 一个对象或一个对象集合。如果并不存在与参数相匹配的对象，则返回null。

690

参数

index

当输入参数是一个从零开始的整数时，返回值是源代码内与某个具体项相对应的独立元素（内嵌在当前元素之内），如果参数为一个字符串(仅适用于IE)，则返回id属性与该字符串相符的元素集合。

subindex

在IE中，如果为第1个参数指定一个字符串值，那么可以将第2个参数指定为一个从零开始的索引值，这样就可以从集合中找到一个id属性与第1个参数的字符串相匹配的具体元素。

namedItem( )

IE 6 NN n/a Moz all Saf all Op 7 DOM 1

```
namedItem("ID")
```

返回一个option对象，它对应于一个与输入的参数字符串相匹配的option元素。

返回值 一个option对象。如果并不存在与参数相匹配的对象，则返回null。

参数

ID

与目标元素的id属性包含相同值的字符串。

remove( )

IE 4 NN n/a Moz n/a Saf all Op 7 DOM n/a

```
remove(index)
```

从当前集合中删除一个元素。只须简单地设置一个从零开始计数的索引值，就可以从select元素的选项集合中删除对应的option元素。下面这段代码演示了如何从select对象中删除第一个选项：

```
document.forms[1].rockers.options.remove(0);
```

而在其他浏览器中删除一个option元素的方法与这个语句截然不同。删除时只须将集合中的对应选项指定为null。例如，在Mozilla浏览器中可以使用如下代码删除一个选项：

```
document.forms[1].rockers.options[0] = null;
```

尽管在不同浏览器内从select对象中删除一个选项的方法各有异同，但对它们而言options数组的长度都是相同的。

返回值 无。

参数

index

一个从零开始的整数，它指明了应该删除集合中的哪一项。

tags( )

IE 4 NN n/a Moz n/a Saf all Op 7 DOM n/a

```
tags("tagName")
```

从当前集合的全部内嵌对象中，返回标签与tagName参数相匹配的对象集合。尽管在所有的IE集合中均已实现了这个方法（请参见all.tags()方法），但对同一类型的元素集合而言完全它是一种冗余。

urns( )

IE 5(Win) NN n/a Moz n/a Saf n/a Op n/a DOM n/a

```
urns(URN)
```

请参见all.urns()方法。

output

IE n/a NN n/a Moz n/a Saf n/a Op 9 DOM n/a

output对象对应于Web Forms 2.0的output元素。请参考第1章中的output元素以便了解实现细节和具体用法。由于output对象也是一种Web Forms 2.0表单控件，因此它也会与input对象共享其属性和方法。此外，由于既不会校验这个对象的数据，也不会将它们随表单一起提交，因此这类方法对这个对象而言并没有什么作用。请参考input对象以了解下文中列出的属性和方法的具体细节信息。

等价HTML元素 <output>

对象模型引用方式 [window.]document.getElementById("*elementID*")

对象特定属性 defaultValue、form、forms、name、value、validationMessage、validity、willValidate

对象特定方法 checkValidity()、setCustomValidity()

p

IE 4 NN n/a Moz all Saf all Op 7 DOM 1

p对象对应于p元素。

等价HTML元素 <p>
对象模型引用方式 [window.]document.getElementById("*elementID*")
对象特定属性 align
对象特定方法 无。
对象特定事件 无。

692

align

IE 4 NN n/a Moz all Saf all Op 7 DOM 1

可读/可写

此属性决定了段落文本在p元素容器盒中的对齐方式。

示例 document.getElementById("myP").align = "center";
值 以下3个对齐常量字符串之一：center | left | right。
默认值 left

page

IE 4 NN n/a Moz all Saf all Op 7 DOM 1

page对象是通过@page CSS规则创建的一种特殊的样式规则类型。在W3C DOM中，这个对象（即CSSPageRule对象）继承了CSSRule对象的属性。但在IE 6/Windows中，page对象并未遵从这个继承结构。这个对象实际上为在将来的浏览器中实现完整的页面盒打下了基础。

等价HTML元素

```
<style type="text/css">
@page {specifications}
</style.
```

对象模型引用方式 [window.]document.styleSheets[i].pages[j]
对象特定属性 pseudoClass、selectorText、style
对象特定方法 无。
对象特定事件 无。

pseudoClass

IE 5.5 NN n/a Moz n/a Saf n/a Op n/a DOM n/a

只读

返回与@page规则相关的伪类名称。

示例 var pClass = document.styleSheets[2].pages[0].pseudoClass;
值 伪类名称字符串：:first | :left | :right。使用时不要遗漏了前面的冒号。
默认值 无。

693

selectorText

IE 5.5 NN n/a Moz n/a Saf n/a Op n/a DOM 2

可读/可写

提供了@page规则的选择器。

示例 document.styleSheets[2].pages[0].selectorText = ":right";
值 字符串。
默认值 无。

style

IE *n/a* NN *n/a* Moz *n/a* Saf *n/a* Op *n/a* DOM *2*

只读

返回代表@page规则的样式属性的style对象（在W3C DOM中是CSSStyleDeclaration类型）。

值 style对象引用。

默认值 无。

pages

IE *5.5* NN *n/a* Moz *n/a* Saf *n/a* Op *n/a* DOM *n/a*

提供styleSheet对象中内嵌的page对象（@page规则）所组成的一个数组。

对象模型引用方式 [window.]document.styleSheets[i].pages

对象特定属性 length

对象特定方法 item()

对象特定事件 无。

length

IE *5.5* NN *n/a* Moz *n/a* Saf *n/a* Op *n/a* DOM *n/a*

只读

返回集合中对象的数量。

示例

```
var howMany = document.styleSheets[0].pages.length;
```

值 整数值。

item( )

IE *5.5* NN *n/a* Moz *n/a* Saf *n/a* Op *n/a* DOM n/a

```
item(index)
```

返回一个page对象，它对应于一个与索引值相匹配的page元素。

返回值 page对象的引用。如果并不存在与参数相匹配的对象，则返回null。

参数

index

一个从零开始的整数，依照源代码次序，它对应于集合中的某个特定项。

param

IE *5* NN *n/a* Moz *all* Saf *all* Op *7* DOM *1*

parma对象对应于param元素，通过它可以将变量值传递到ActiveX对象（仅适用于IE/Windows）、Java小程序和某些插件。编写这些嵌入式的程序时就准备让它们在初始化时读取其需要的名称/值对，如声音文件的URL地址，因此程序启动时这些参数就会传递到程序之中。IE/Windows通常会为某些ActiveX控件分配一组参数，但可能只有其中很少的一部分会明确定义在代码之中。尽管这些属性都是可读可写的，但在页面加载之后再更新属性值并不会将它们传递至外部程序。

等价HTML元素 <param>

对象模型引用方式 [window.]document.getElementById("*elementID*")

对象特定属性 name、type、value、valueType

对象特定方法 无。

对象特定事件 无。

name

IE *5* NN *n/a* Moz *all* Saf *all* Op *7* DOM *1*

可读/可写

这是外部程序参数的名称，param元素将会为它提供一个对应的值。

示例 `var pName = document.getElementById("audioParam2").name;`
值 字符串。
默认值 无。

type

IE 5 NN n/a Moz all Saf all Op 7 DOM 1

695

可读/可写

这个属性会为param元素提供一个MIME类型，同时将valuetype属性设置为“ref”。

示例

```
if (document.getElementById("myParam").valueType == "ref") {
   var pType = document.getElementById("myParam").type;
}
```

值 不区分大小写的MIME类型字符串。MIME类型目录可从此处获得：*http://www.iana.org/assignments/media-types/*。
默认值 无。

value

IE 5(Mac)/6(Win) NN n/a Moz all Saf all Op 7 DOM 1

可读/可写

它可以为外部程序的某个参数指定一个字符串值。

示例 `var pVal = document.getElementById("volumeParam").value;`
值 字符串。
默认值 无。

valueType

IE 5(Mac)/6(Win) NN n/a Moz all Saf all Op 7 DOM 1

可读/可写

表示元素设定的参数分类。

示例 `var pValType = document.getElementById("volumeParam").valueType;`
值 字符串常量：`data | object | ref`。
默认值 `data`

password

请参见input。

personalbar

请参见directories。

plaintext

IE 4 NN n/a Moz all Saf all Op 7 DOM 1

696

plaintext对象对应于plaintext元素。须要注意的是，Win32版本的IE 4会错误地处理innerHTML、innerText、outerHTML及outerText的属性值。这些属性值都会包含元素起始标签之后的所有文档内容。另外，目前已不赞成使用这个元素，而推荐使用pre元素。

等价HTML元素 `<plaintext>`
对象模型引用方式 `[window.]document.getElementById("elementID")`

对象特定属性 无。
对象特定方法 无。
对象特定事件 无。

plugin

IE *5(Mac)* NN *3* Moz *all* Saf *all* Op *7* DOM *n/a*

plugin对象代表一个在浏览器启动时就已注册的插件。在使用时，一般会通过navigator.plugins数组来访问一个插件。在加载外部内容之前，还会使用navigator.mimeTypes数组及其相关的属性来判断是否已经安装所需的插件。通过这些属性提供的脚本访问方法可以获取位于Navigator的“**帮助**”菜单和IE/Macintosh的“**文件助手**”选项中的一些信息。而IE/Windows则使用了另外一种方法来判断浏览器是否对外部媒体提供支持。

对象模型引用方式 navigator.plugins[i]
对象特定属性 description、filename、length、name
对象特定方法 无。

description

IE *5(Mac)* NN *3* Moz *all* Saf *all* Op *7* DOM *n/a*

只读

这个属性给出了由插件制造商提供的简要说明。

示例
```
var descr = navigator.plugins[2].description;
```
值 字符串。
默认值 无。

filename

IE *5(Mac)* NN *3* Moz *all* Saf *all* Op *7* DOM *n/a*

只读

返回二进制插件文件的文件名称。某些早期浏览器版本会返回整个路径名，但现代浏览器则只会返回其文件名。

示例
```
var file = navigator.plugins[2].filename;
```
值 字符串。
默认值 无。

length

IE *5(Mac)* NN *3* Moz *all* Saf *all* Op *7* DOM *n/a*

只读

返回该插件所能支持的MIME类型的总数。请注意不要将这个属性与整个navigator.plugins数组的length属性混为一谈，后者表示浏览器中已知的plugin对象数量。

示例
```
var howManyMIMEs = navigator.plugins[2].length;
```
值 整数值。
默认值 无。

name

IE *5(Mac)* NN *3* Moz *all* Saf *all* Op *7* DOM *n/a*

只读

返回插件生产商提供的插件名称。但同一类型的插件在不同操作系统下并不能保证它拥有相同的文件名。

示例
```
var pName = navigator.plugins[2].name;
```
值 整数值。
默认值 无。

plugins

IE 4 NN 3 Moz all Saf all Op 7 DOM n/a

所有的主流浏览器均拥有一个plugins数组，但在不同浏览器中plugins集合内的对象也存在着巨大的差异。在Netscape Navigator、Mozilla、Safar和Opera中，plugins数组是navigator对象的一个属性。而navigator.plugins数组中的每一项都对应于浏览器中安装的一个插件，即浏览器在最后一次启动时已注册的插件。请参见plugin对象。

但在Windows中，IE的plugins集合属于document对象，它实质上是embeds集合的一个镜像，而后者则是文档中所有embed元素组成的集合。embed元素也许已经启动了一个插件，但实际情况也可能并非如此。在Navigator和Mozilla中可以使用JavaScript脚本访问已安装的插件，但Windows和Macintosh系统下的IE浏览器则并未提供类似的方法。

698

对象模型引用方式

IE

```
document.plugins
```

NN、Mozilla、Safari、Opera

```
navigator.plugins
```

对象特定属性 length

对象特定方法 item()、namedItem()、refresh()

length

IE 4 NN 3 Moz all Saf all Op 7 DOM n/a

只读

返回集合中元素的数量。

示例

```
var IEhowMany = document.plugins.length;
var NNhowMany = navigator.plugins.length;
```

值 整数值。

item( )

IE 4(Win) NN n/a Moz all Saf all Op 7 DOM n/a

```
item(index[, subindex]) item(index)
```

根据符合索引值（或任意索引及子索引值）的集合项，返回一个单独的对象或一组对象集合。

返回值 一个对象或一个对象集合。如果并不存在与参数相匹配的对象，则返回null。

参数

index

当输入参数是一个从零开始的整数时，返回值是源代码内与某个具体项相对应的独立元素（内嵌在当前元素内）。如果参数为一个字符串（仅适用于IE），则返回name属性与该字符串相符的元素集合。

subindex

在IE中，如果为第1个参数指定一个字符串值，那么可以将第2个参数指定为一个从零开始的索引值，这样就可以从集合中找到一个name属性与第1个参数的字符串相匹配的具体元素。

699

namedItem( )

IE 6 NN n/a Moz all Saf n/a Op 7 DOM n/a

```
namedItem("name")
```

返回单个plugin对象（NN）或embed对象（IE），它们对应于一个符合参数字符串值的元素。

返回值 一个plugin对象（NN）或embed对象（IE）。如果并不存在与参数相匹配的对象，则返回null。

参数

name

与目标对象的name属性值完全相同的字符串。

refresh( )

IE *5(Mac)* NN *3* Moz *all* Saf *all* Op *7* DOM *n/a*

命令浏览器注册已安装在插件目录下的所有插件。这样在不须要用户重启浏览器的情况下，就能够让浏览器掌握最新安装的插件的相关信息。

返回值 无。

参数 无。

popup

IE *5.5* NN *n/a* Moz *n/a* Saf *n/a* Op *n/a* DOM *n/a*

popup对象是一个毫无特色的矩形空间，它既不拥有浏览器窗口的典型外壳装饰（边框、滚动条和标题栏等），也无法从它引用到主文档。在使用脚本创建popup对象时，首先要为它指定尺寸和具体的位置，还要为document.body.innerHTML属性指定HTML字符串，以达到填充窗口的目的。另外，还须要为document.body.style属性或内部的样式元素指定背景色和边框，以便在文档中突出弹出区域。

尽管popup对象的本质与document对象很相似，但实际上它与window对象没有任何关系，而且也不会加载外部文档。但它拥有一个十分有用的特性，即能够超越框架或窗口给出操作系统级别的HTML内容容器。因此可以在下拉菜单或注释中使用这个对象，以便让它们悬浮于框架边框上或扩展到浏览器窗口边框之外。

popup是一种临时显示的元素。一旦在弹出框外点击鼠标都会让这个弹出框消失。正因为如此，在进行滚动或点击菜单项时，可以将所有的鼠标事件都指定给弹出框文档中的元素，以防止弹出框消失。另外，在弹出框中也可以包含图片内容。

在创建一个popup对象时，请使用window.createPopup()方法。下面这段代码演示了如何创建、设置并显示弹出框：

```
var popup = window.createPopup();
var bod = popup.document.body;
bod.style.border = "3px solid #ff8800";
bod.style.padding = "2px";
bod.style.backgroundColor = "lightyellow";
bod.innerHTML =
  "<p style='font-family:Arial, sans-serif; font-size:10px'>Some popup text.</p>";
popup.show(100, 100, 100, 26, document.body);
```

对象模型引用方式 *popupObjectRef*

对象特定属性 document、isOpen

对象特定方法 hide()、show()

对象特定事件 无。

document

IE *5.5* NN *n/a* Moz *n/a* Saf *n/a* Op *n/a* DOM *n/a*

只读

返回popup对象内document对象的引用。document对象的大部分常规属性都适用于弹出框的document对象。这个对象是为弹出框指定HTML内容的主要方法。尽管document是一个只读属性，但document对象的属性却是可读可写的，因此可以为其属性指定具体的值。

示例 *popupRef*.document.body.innerHTML = " Howdy, pardner! ";

值 document对象的引用。

默认值 当前document对象。

isOpen

IE *5.5* NN *n/a* Moz *n/a* Saf *n/a* Op *n/a* DOM *n/a*

只读

返回一个布尔值，以表示popup对象是否可见。即使popup对象已被隐藏，脚本也可以访问其内容。

示例

```
if (popupRef.isOpen) {
   popupRef.hide();
}
```

值 布尔值：true | false。

默认值 false

hide( )

IE 5.5 NN n/a Moz n/a Saf n/a Op n/a DOM n/a

此方法可以隐藏popup对象。通常由用户在弹出内容中触发相关的脚本来调用这个方法。

返回值 无。

参数 无。

show( )

IE 5.5 NN n/a Moz n/a Saf n/a Op n/a DOM n/a

```
show(left, top, width, height[, positioningElemRef])
```

为popup对象指定了相关参数后，可以使用这个方法将它显示出来。在参数中可设定其尺寸和位置。此时可以根据主文档中某个元素的位置随意设定弹出框的位置。而参数中的弹出位置和定位元素通常都来自于event对象的属性，如event.clientX、event.clientY和event.srcElement。

返回值 无。

参数

left

相对于屏幕左边缘的水平坐标，如果出现可选参数，则相对于该HTML元素，单位为像素。

top

相对于屏幕左边缘的竖直坐标，如果出现可选参数，则相对于该HTML元素，单位为像素。

width

整个弹出空间的外部宽度，单位为像素。

height

整个弹出空间的外部高度，单位为像素。

positioningElemRef

可选的参数，它可以指向任意一个脚本可以访问的元素引用。从而为left和top这两个参数确定一个参考坐标系。

pre

IE 4 NN n/a Moz all Saf all Op 7 DOM 1

pre对象对应于pre元素，它可用于预格式化相关文本。

等价HTML元素 <pre>

对象模型引用方式 [window.]document.getElementById("*elementID*")

对象特定属性 width

对象特定方法 无。

对象特定事件 无。

width

IE 6 NN n/a Moz all Saf all Op 7 DOM 1

可读/可写

为元素中按照单间隔排版的内容指定每行的字符数。

示例

```
document.getElementById("codeExample2").width = 40;
```

值 整数值。

默认值 -1（Mozilla）；-1（Safari、Opera）；none（IE）。

ProcessingInstruction

IE *n/a* NN *n/a* Moz *n/a* Saf *n/a* Op *n/a* DOM *1*

ProcessingInstruction是W3C DOM中的一个抽象对象，它代表一个包含应用程序指令的元素，但这个元素的相关内容并未作为文档内容树的一部分。XML文档会以*<?ProcessTarget InstructionText?>*的形式将这些元素标记出来。在W3C DOM中，这个元素的两个重要组成分别由target和data字符串属性表示。

q

IE *4* NN *n/a* Moz *all* Saf *all* Op *7* DOM *1*

q对象对应于q元素，这个元素用于行内引用。尽管IE/Windows已经在所有元素都引入了cite属性，但只有在IE/Macintosh及W3C DOM浏览器中这个对象才能够使用cite属性的真正含义。

等价HTML元素 <q>

对象模型引用方式 [window.]document.getElementById("*elementID*")

对象特定属性 无。

对象特定方法 无。

对象特定事件 无。

radio

请参见input。

Range

IE *n/a* NN *n/a* Moz *all* Saf *all* Op *8* DOM *2*

W3C DOM的Range对象在概念上与IE的TextRange对象很类似，它代表了文档中一串已显示的文本字符。当文本域中的字符数为零时，它就退化成为了一个字符插入点。Range对象会自动为文本域的起点与终点记录节点及字符偏移，因此通过它就能够复制已存在的文本内容、删除文本域内的文本或向文本域中插入新的文本。尽管节点性对Range对象很重要，但这些依然是在开发者的掌控之下。

通过document.createTextRange()方法可以创建一个Range对象，使用window.getSelection().getRangeAt(0)也可以将用户选择转化为一个文本选择域对象。一旦创建了文本选择域，那么就可以使用该对象的方法来调整其起点和终点以便包含目标文本分段。然后只须选择合适的附加方法来处理所选择的文本域即可。另外，在线参考V介绍了Range对象的详细使用方法，以及它与IE的TextRange对象的语法区别。

对象模型引用方式 document.createRange()

对象特定属性 collapsed、commonAncestorContainer、endContainer、endOffset、startContainer、startOffset

对象特定方法 cloneContents()、cloneRange()、collapse()、compareBoundaryPoints()、compareNode()、comparePoint()、createContextualFragment()、deleteContents()、detach()、exTRactContents()、insertNode()、intersectsNode()、isPointInRange()、selectNode()、selectNodeContents()、setEnd()、setEndAfter()、setEndBefore()、setStart()、setStartAfter()、setStartBefore()、surroundContents()、toString()

对象特定事件 无。

collapsed

IE *n/a* NN *n/a* Moz *all* Saf *all* Op *8* DOM *2*

只读

如果文本域的起点与终点重合，那么就返回布尔值true。此时表示文本域中未包含任何字符。这种缩为一点的文本域可以放置在文档中的任意位置。

示例

```
if (rng.collapsed) {
   // 处理已缩为一点的文本域
}
```

值 布尔值：true | false。

默认值 无。

commonAncestorContainer

IE *n/a* NN *n/a* Moz *all* Saf *all* Op *8* DOM *2*

只读

返回一个文档树节点的引用，该节点是容纳当前文本域起点和终点的临近的最外层容器。如果文本域的起点位于同一个节点之内，例如一个文本节点，那么commonAncestorContainer属性就会返回该节点的父节点的引用。IE中TextRange对象的parentElement()方法与这个属性等价。

示例 `var containingElem = rng.commonAncestorContainer;`

值 一个节点对象的引用，通常情况下是一个元素节点对象。

默认值 无。

endContainer

IE *n/a* NN *n/a* Moz *all* Saf *all* Op *8* DOM *2*

只读

返回包含当前文本域终点的文档树节点的引用。

示例 `var containingElemRight = rng.endContainer;`

值 指向节点对象的一个引用。

默认值 无。

endOffset

IE *n/a* NN *n/a* Moz *all* Saf *all* Op *8* DOM *2*

只读

返回endContainer属性指明的节点中所包含的字符数或节点数。如果endContainer是一个文本节点，那么endOffset属性表示该文本节点内所有的字符数。而如果endContainer是一个元素节点，那么这个属性则表示容器节点起点与文本域终点之间的节点数量。

请参考下面这段文档片段示例，它使用“|”表示了文本域的起点和终点，而文本域内则是一段使用粗体的文字。

```
<p>One paragraph with |a <span>nested</span>| element inside.</p>
```

须要注意的是，在这个文档片段内，起点位于文本节点之内，而终点则在span元素的结束标签之外。此时，Range对象的各个属性所表达的含义如下表所示：

属性	值	描述
commonAncestorContainer	[object HTMLParagraphElement]	文本域的起点和终点都位于 p 元素之中
startContainer	[object Text]	起点位于一个文本节点之内
startOffset	19	文本域的起点位于其容器（即文本节点）的第 20 个字符处（从 0 开始计数则为 19）
endContainer	[object HTMLParagraphElement]	终点位于 span 元素之后，这使得临近的外层 p 元素成为了结束点的容器对象
endOffset	2	在 endContainer 容器（即 p 元素）中，第一个节点是文本节点，span 元素是第二个节点，而终点则位于第三个对象起始部分

示例 `var rngEndOff = rng.endOffset;`
值 整数值。
默认值 无。

startContainer

IE *n/a* NN *n/a* Moz *all* Saf *all* Op *8* DOM *2*

只读

返回包含当前文本域起点的文档树节点的引用。

示例 `var containingElemLeft = rng.startContainer;`
值 指向节点对象的一个引用。
默认值 无。

startOffset

IE *n/a* NN *n/a* Moz *all* Saf *all* Op *8* DOM *2*

只读

返回`startContainer`属性指明的节点中所包含的字符数或节点数。如果`startContainer`是一个文本节点，那么`startOffset`属性表示该文本节点内所有的字符数。而如果`startContainer`是一个元素节点，那么这个属性则表示容器节点起始点与文本域起点之间的节点数量。详细信息请参考`endOffset`属性。

示例 `var rngStartOff = rng.startOffset;`
值 整数值。
默认值 无。

cloneContents( )

IE *n/a* NN *n/a* Moz *all* Saf *all* Op *8* DOM *2*

返回一个包含当前文本域中相关内容的`DocumentFragment`节点。在此方法中，会通过“克隆”过程处理所有的“悬挂”节点。

返回值 指向一个文档片段类型节点的引用。
参数 无。

cloneRange( )

IE *n/a* NN *n/a* Moz *all* Saf *all* Op *8* DOM *2*

返回一个复制了当前文本域的`Range`对象，其中还包括相关容器的引用。在创建一个新的`Range`对象时，这个方法可以完整地复制一个`Range`对象的细节信息。它和IE `TextRange`对象的`duplicate()`方法比较类型。

返回值 `Range`对象的引用。
参数 无。

collapse( )

IE *n/a* NN *n/a* Moz *all* Saf *all* Op *8* DOM *2*

`collapse(`*`toStartFlag`*`)`

将当前文本域缩小到一个插入点，即将起点和终点放置在同一个节点中的相同位置。方法中所需的布尔值参数表示是将该插入点放置在文本域的起点（`true`）还是终点（`false`）。在实际使用中，在将终点放置到正文的末尾以便执行下一次`String.indexOf()`搜索之前，脚本通常会将文本域收缩到其终点。

返回值 无。
参数

`toStartFlag`

方法中所需的布尔值参数表示是将该收缩位置放置在文本域的起点（`true`）还是终点（`false`）。

compareBoundaryPoints( )

IE *n/a* NN *n/a* Moz *all* Saf *all* Op *8* DOM *2*

```
compareBoundaryPoints(compareType, sourceRangeRef)
```

返回一个整数代码，以指明当前文本域的边界点与另一个文本域的边界点之间的相对位置。在最简单的情况下，这两个文本域的端点都位于同一个父级容器之中。这时由第一个参数决定用哪一个端点进行比较。比较时请使用下表所示的常量。

比较类型	说明
rng.START_TO_START	将当前域的起点与指定域的起点进行比较
rng.START_TO_END	将当前域的起点与指定域的终点进行比较
rng.END_TO_END	将当前域的终点与指定域的终点进行比较
rng.END_TO_START	将当前域的终点与指定域的起点进行比较

在比较过程中，如果在文档内第1个边界点位于第2个边界点之前，那么返回-1，如果第1个边界点位于第2个边界点之后，那么返回1，如果它们位于同一点，则返回0。它和IE TextRange对象的compareEndPoints()方法比较类似。

如果这两个文本域的边界点拥有不同的父级容器节点，那么此时的情况将更为复杂。在这种情况下，它们相对于容器节点的偏移量会影响最终的比较结果。由于用于比较的两个端点相对位置比较复杂，从而导致比较的结果也多种多样，因此脚本需要对边界点进行相对复杂的分析以便确保比较结果真实地反映了实际的情况。从另一方面来看，简单地探讨边界点的一致性会使得比较更为简单。此时只需要将比较结果限制在判断是否为0（或另一个值），就可以得到比较准确的比较结论。

返回值 整数值-1、0或1。

参数

compareType

对应于比较类型的整数（0至3）。在1.4版之前，Mozilla浏览器所使用的整数值与W3C DOM标准并不相符，但使用明语常量时则对应着正确的比较类型。

sourceRangeRef

另一个已定义好的Range对象的引用，有可能是使用cloneRange()方法保存的一个对象。

compareNode( )

IE *n/a* NN *n/a* Moz *all* Saf *n/a* Op *n/a* DOM *n/a*

```
compareNode(nodeReference)
```

这个方法只能在Mozilla浏览器中使用，它返回一个整数代码，以表示当前文本域与某些其他节点的相对位置。每个Mozilla的Range对象都拥有4个明语常量，它们均可被用作compareNode()方法的返回值。须要注意的是，这个方法会从nodeReference节点的视角进行比较，而不是从当前文本域的角度获得比较结果。

返回值及对应常量如下表所示。

常量	值	说明
rng.NODE_BEFORE	0	整个节点都位于文本域之前
rng.NODE_AFTER	1	整个节点都位于文本域之后
rng.NODE_BEFORE_AND_AFTER	2	当前文本域位于节点之
rng.NODE_INSIDE	3	整个节点都位于文本域所包含的范围之内

使用方法如下：

```
if (rng.compareNode(document.getElementById("myElem")) == rng.NODE_INSIDE) {
    // 处理当前文本域中包含的 myElem 节点
}
```

返回值 整数值，0、1、2或3。

参数

nodeReference

文档树中任意一个节点的引用。

comparePoint( )

IE *n/a* NN *n/a* Moz *all* Saf *n/a* Op *n/a* DOM *n/a*

```
comparePoint(nodeReference, offset)
```

这个方法只能在Mozilla浏览器中使用，它返回一个整数代码，以表示当前文本域与某些其他节点的相对位置及偏移量。须要注意的是，这个方法会从nodeReference节点及该节点内的offset偏移点的视角进行比较，而不是从当前文本域的角度获得比较结果。

返回值如下：

值	说明	值	说明	值	说明
-1	指定点位于文本域之前	0	指定点位于文本域之内	1	指定点位于文本域之后

使用方法如下：

```
if (rng.comparePoint(document.getElementById("myElem"), 2) == 0) {
    // 对当前域中 myElem 节点内偏移量为 2 的文本进行处理
}
```

返回值 整数值-1、0或1。

参数

nodeReference

文档树中任意一个节点的引用。

offset

整数偏移量，表示元素内的某个内嵌节点或文本节点内的字符。

createContextualFragment( )

IE *n/a* NN *n/a* Moz *all* Saf *all* Op *8* DOM *2*

```
createContextualFragment(contentString)
```

由于W3C DOM为包含标签的内容字符串仅提供了很有限的支持，因此设计了createContextualFragment()方法以便替代innerHTML属性。这个方法可以接收任何字符串（包括标签内容）作为参数，并且会返回一个DocumentFragment类型的节点以便将该节点附加或插入到文档树。但这个方法除了与W3C DOM的节点化目标更为一致之外，在Mozilla浏览器中完全可以使用innerHTML属性代替这个方法。

返回值 指向文档树之外的一个文档片段类型节点的引用。然后就可以将这个节点添加至文档树之中。

参数

contentString

字符串形式的文档内容，其中还可以包含标签和属性。

deleteContents( )

IE *n/a* NN *n/a* Moz *all* Saf *all* Op *8* DOM *2*

从文档树中移除当前文本域中的内容。如果该文本域是一个元素节点（例如，使用selectNode()方法创建了两个边界点的文本域），那么请使用deleteContents()方法从文档树中移除该节点，并将文本域缩小至一个插入点。此时对应的Range对象会继续保存在内存中，但它不再包含任何内容。如果希望在删除前保存其内容，那么请首先使用其他的Range对象方法，例如，首先调用cloneRange()，成功后再调用cloneContents()方法。

返回值 无。

参数 无。

detach( )

IE *n/a* NN *n/a* Moz *all* Saf *all* Op *8* DOM *2*

销毁当前的Range对象，一旦调用了这个方法，那么无论是调用该对象的方法还是访问其属性都会抛出一个INVALID_STATE_ERR类型的RangeException异常。

返回值 无。

参数 无。

extractContents( )

IE *n/a* NN *n/a* Moz *all* Saf *all* Op *8* DOM *2*

将文本域的内容从文档树中移除后，返回一个包含当前文本内容的DocumentFragment节点。如果在调用这个方法时出现问题，那么不妨先通过selectNodeContents()方法设置文本域的边距点。

返回值 指向一个文档片段类型节点的引用。

参数 无。

insertNode( )

IE *n/a* NN *n/a* Moz *all* Saf *all* Op *8* DOM *2*

```
insertNode(nodeReference)
```

在当前文本域的起始部位插入一个节点。当文本域已经缩小为一个文本插入点时这个方法最为有用。既可以通过document.createElement()方法动态创建这个节点，也可以从文档树中直接获取，在后一种情况中，会首先将它从原有位置删除然后再插入到当前域中。如果在一个已存在的文本节点中插入一个文本节点，那么最终会得到两个相邻的文本节点。此时，在它们的父级对象上调用normalize()方法就可以合并这两个文本节点。

返回值 Nothing

参数

nodeReference

任意一个文本、元素或文档片段节点的引用，此方法会将该节点插入域中。

intersectsNode( )

IE *n/a* NN *n/a* Moz *all* Saf *all* Op *8* DOM *2*

```
intersectsNode(nodeReference)
```

如果当前域中的任意部分与参数传入的文本或元素节点发生了重叠，那么返回布尔值true。如果脚本代码探测到它们的确存在交集，那么可以就使用compareNode()方法来获得更加详细的信息。

返回值 布尔值：true | false。

参数

nodeReference

文档树中任意一个文本或元素节点的引用。

isPointInRange( )

IE *n/a* NN *n/a* Moz *all* Saf *all* Op *8* DOM *2*

```
isPointInRange(nodeReference, offset)
```

如果参数指定的位置位于当前文本域内，那么返回布尔值true。

返回值 布尔值：true | false。

参数

nodeReference

文档树中任意一个文本或元素节点的引用。

offset

整数偏移量，表示元素内的某个内嵌节点或文本节点内的字符。

selectNode( ), selectNodeContents( )

IE *n/a* NN *n/a* Moz *all* Saf *all* Op *8* DOM *2*

`selectNode(`*nodeReference*`)` `selectNodeContents(`*nodeReference*`)`

这两个方法可以让文本域的边界点包围一个节点或仅包围节点中的内容。无论采用哪个方法，都不会高亮显示文档中的主体文本。

但是不同的方法对文本域对象的startContainer和endContainer属性的影响有所不同。在下面这段代码中，可以看到将一个元素节点和文本节点作为参数传入这些方法时对文本域有什么影响。在初始状态下，HTML片段如下：

```
<p>One paragraph with a <span id="myspan">nested</span> element inside.</p>
```

通过rng.selectNode(document.getElementById("myspan"))方法选择span元素后，文本域如下：

```
<p>One paragraph with a |<span id="myspan">nested</span>| element inside.</p>
```

此时，Range对象的各个属性的值如下表所示：

属性	值	说明
startContainer	[object HTMLParagraphElement]	起点位于 span 元素之前
startOffset	1	起点位于 p 元素的第 2 个(从 0 开始计数时即为 1)节点之中
endContainer	[object HTMLParagraphElement]	终点紧跟在 span 元素之后
endOffset	2	终点位于 endContainer 容器（即 p 元素）的第 3 个对象之中

通过rng.selectNodeContents(document.getElementById("myspan"))方法选择span元素的内容后，文本域如下：

```
<p>One paragraph with a <span id="myspan">|nested|</span> element inside.</p>
```

此时，Range对象的各个属性的值如下表所示：

属性	值	说明
startContainer	[object HTMLSpanElement]	起点位于 span 元素之内
startOffset	0	起始点位于 span 元素的第 1 个（从 0 开始计数时即为 0）节点之中
endContainer	[object HTMLSpanElement]	终点紧跟在 span 元素的内容之后
endOffset	1	在 endContainer 容器（即 span 元素）中，终点位于第 2 个对象所在的位置

712

通过rng.selectNode(document.getElementById("myspan").firstChild)方法选择span元素内的文本节点后，文本域如下：

```
<p>One paragraph with a <span id="myspan">|nested|</span> element inside.</p>
```

尽管此时传入的参数节点有所不同，但这个方法产生的新文本域与前一个方法所产生的一样。事实上，根据节点树的组成方式，此时Range对象各个属性的值如下表所示：

属性	值	说明
startContainer	[object HTMLSpanElement]	起点位于 span 元素之内
startOffset	0	起点位于 span 元素的第 1 个（从 0 开始计数时即为 0）节点之中
endContainer	[object HTMLSpanElement]	终点紧跟在 span 元素的内容之后
endOffset	1	在 endContainer 容器（即 span 元素）中，终点位于第 2 个对象所在的位置

通过rng.selectNodeContents(document.getElementById("myspan"))方法选择span元素内文本节点的内容后，文本域如下：

```
<p>One paragraph with a <span id="myspan">||nested</span> element inside:</p>
```

换句话说，此时文本域会成为一个位于文本节点开头的插入点，而该文本节点会成为其容器，其属性如下表所示。

属性	值	说明
startContainer	[object Text]	起始点位于文本节点开头部分
startOffset	0	起始点位于文本节点的第一个（从0开始计数时即为0）位置
endContainer	[object Text]	终点已收缩到起点所在的位置
endOffset	0	终点已收缩到起点所在的位置

对这两个方法而言，元素节点是最实用的输入参数。

返回值 无。

参数

nodeReference

文档树中任意一个文本或元素节点的引用。

setEnd( ), setStart( )

IE *n/a* NN *n/a* Moz *all* Saf *all* Op *8* DOM *2*

```
setEnd(nodeReference, offset) setStart(nodeReference, offset)
```

在文档树中为已经存在的Range对象的单个边界点指定具体位置。它们和IE TextRange对象的setEndPoint()方法比较类型。具体的位置转换依赖于参数传入的节点引用，以及与节点起始点和类型有关的偏移量。但nodeReference是一个元素节点时，偏移量表示其中的子节点计数，而当*nodeReference*为文本节点时，则表示字符计数。另外，如果要在一个节点的边缘上设定边界点，那么使用其他的相关方法（setEndAfter()等4个方法，请参见下文）更为简便。

返回值 无。

参数

nodeReference

文档树中任意一个文本或元素节点的引用。

offset

整数偏移量，表示元素内的某个内嵌节点或文本节点内的字符。

setEndAfter( ), setEndBefore( ), setStartAfter( ), setStartBefore( )

IE *n/a* NN *n/a* Moz *all* Saf *all* Op *8* DOM *2*

```
setEndAfter(nodeReference) setEndBefore(nodeReference)
setStartAfter(nodeReference) setStartBefore(nodeReference)
```

相对于节点的一条边，在文档树中为已经存在的Range对象的单个边界点指定具体位置。这些方法都假设在不考虑其他偏移量的情况下将文本域的端点放置在一个已有的节点之前或之后。文本域的边界点并不需要完全对称，这样就可以根据某个节点设置起点再根据另一个不同的节点指定终点。

返回值 无。

参数

nodeReference

文档树中任意一个文本或元素节点的引用。

surroundContents( )

IE *n/a* NN *n/a* Moz *all* Saf *all* Op *8* DOM *2*

```
surroundContents(parentNodeReference)
```

将当前文本域放入一个新的容器，通常会使用document.createElement()方法来创建一个新的元素节点作为容器。在使用这个方法时，要保证当前文本域的端点拥有相同的父级容器。

返回值 无。

参数

parentNodeReference

一个节点引用，该节点将成为文本域的新父级容器。

toString( )

IE *n/a* NN *n/a* Moz *all* Saf *all* Op *8* DOM *2*

返回文本域所包含的主体内容字符串。在这个返回值中并不会包含任何的标签或属性。

返回值 字符串。

参数 无。

RangeException

IE *n/a* NN *n/a* Moz *1.0.1* Saf *all* Op *8* DOM *2*

某些作用于W3C DOM Range对象上的操作会触发错误，或者用JavaScript 1.5的术语来说，即：如果出现错误就会抛出异常。W3C DOM定义了一个对象，它传达了一个来自于与Range对象相关的异常列表的代码号。例如，如果要通过设置文本域的边界来包含一些与内容无关的节点，如Attr节点，那么将该节点作为参数传入selectNode()方法时会就会抛出一个代码号为2的异常。这个数字对应的异常表示：尝试在文本域上执行一个非法或不符合逻辑的操作。

对象模型引用方式 *errorObjectReference*

对象特定属性 code

对象特定方法 无。

对象特定事件 无。

code

IE *n/a* NN *n/a* Moz *1.0.1* Saf *all* Op *8* DOM *2*

只读

提供与一个已定义的Range对象错误类型相对应的整数，如下表所示：

代码	常量	最有可能的原因
1	BAD_BOUNDARYPOINTS_ERR	在一个包含不合适的端点的文本域上调用 surroundContents()方法
2	INVALID_NODE_TYPE_ERR	在不适用的节点类型上调用了某个方法

值 整数值。

默认值 由错误决定。

RepetitionElement

IE *n/a* NN *n/a* Moz *n/a* Saf *n/a* Op *9* DOM *n/a*

Web Forms 2.0环境中设定的任何HTML元素都是RepetitionElement的一个实例，根据属性设置的不同，它既可以是一个重复块也可能是一个模板。这个对象包含很多属性和方法，在支持它的浏览器中（如Opera 9），它们还会成为HTML元素的组成部分。请参见本章前文中的共享属性和方法以了解它们的详细信息。

对象特定属性 repeatMax、repeatMin、repeatStart、repetitionBlocks、repetitionIndex、repetitionTemplate、repetitionType

对象特定方法 addRepetitionBlock()、addRepetitionBlockByIndex()、moveRepetitionBlock()、removeRepetitionBlock()

对象特定事件 无。

reset

请参见input。

rows

IE 4 NN n/a Moz all Saf all Op 7 DOM 1

提供一个由tr元素对象组成的一个集合，这些tr元素都包含在一个单独的table、tfoot或thead元素对象之中。无论表格中的行如何分组，rows集合都会包含table元素中的所有表格行。此集合中的成员按照它们在源代码中的次序进行排序。在IE中，既可以使用数组符号也可以使用括号的形式来访问集合中的某个具体行，如 document.getElementById ("myTable").rows[0]或document.all.myTable.rows(0)。

对象模型引用方式 document.getElementById("*tableOrSectionID*").rows

对象特定属性 length

对象特定方法 item()、namedItem()、tags()、urns()

对象特定事件 无。

length

IE 4 NN n/a Moz all Saf all Op 7 DOM 1

只读

返回集合中元素的数量。

示例
```
var howMany = document.getElementById("myTable").rows.length;
```
值 整数值。

item( )

IE 4 NN n/a Moz all Saf all Op 7 DOM 1

```
item(index[, subindex]) item(index)
```

根据符合索引值（IE中也可以使用index或subindex值）的元素对象，返回一个单独的tr对象或一组tr对象集合。

返回值 一个tr对象或一个tr对象集合。如果并不存在与参数相匹配的对象，则返回null。

参数

index

当输入参数是一个从零开始的整数时，返回值是源代码内与某个具体项相对应的独立元素（内嵌在当前元素之内），如果参数为一个字符串(仅适用于IE)，则返回id属性与该字符串相符的元素集合。

subindex

在IE中，如果为第1个参数指定一个字符串值，那么可以将第2个参数指定为一个从零开始的索引值，这样就可以从集合中找到一个id属性与第1个参数的字符串相匹配的具体元素。

namedItem( )

IE 6 NN n/a Moz all Saf all Op 7 DOM 1

```
namedItem("ID")
```

返回单个tr对象或tr对象集合，它们对应于符合参数字符串值的元素。

返回值 一个tr对象或一个tr对象集合。如果并不存在与参数相匹配的对象，则返回null。

参数

ID

与目标元素的id属性包含相同值的字符串。

tags( )

IE 4 NN n/a Moz n/a Saf all Op 7 DOM 1

717

```
tags(tagName)
```

从当前集合内的全部对象中，返回标签与tagName参数相匹配的对象集合。此时，由于所有的元素均拥有相同的tr标签，因此这是一个完全多余的方法。

返回值 对象集合。如果与参数相匹配的对象不存在，则返回一个长度为零的数组。

参数

tagName

小写字母形式的元素标签名字符串，如document.getElementById("myTable") rows.tags("tr")。

urns( )

IE 5(Win) NN n/a Moz n/a Saf n/a Op n/a DOM n/a

```
urns(URN)
```

请参见all.urns()方法。

rb, ruby, rt

IE 5 NN n/a Moz n/a Saf n/a Op n/a DOM n/a

在这3个与ruby文本相关的元素中，IE DOM正式将其中的ruby和rt作为对象进行支持。但rb元素实际上也可以被视为一个元素对象。Mozilla和Opera则将这些有效的HTML元素对象视为未知类型。关于这些元素的具体使用方法，请参见第1章中的ruby元素。作为脚本对象，它们只拥有普通HTML元素对象所共有的属性和方法。

对象模型引用方式 document.getElementById("*elementID*")
对象特定属性 无。
对象特定方法 无。
对象特定事件 无。

rule

请参见CSSRule。

rules

718

请参看CSSRules。

runtimeStyle

IE 5(Win) NN n/a Moz n/a Saf n/a Op n/a DOM n/a

在IE 5+/Windows中，所有HTML元素对象都将runtimeStyle对象作为其属性之一，这个对象可以看作是一个超级style对象，设定它的属性后，会替代代码中已存在的样式定义。因此，它可以取代全局、导入的、链接的及内联的样式定义。另外，这个对象的属性和方法几乎与style对象完全相同。

对象模型引用方式 [window.]document.getElementById("*elementID*").runtimeStyle
对象特定属性 请参看style对象。
对象特定方法 请参看style对象。
对象特定事件 无。

S

请参见b。

samp

请参见abbr。

screen

IE *4* NN *n/a* Moz *all* Saf *all* Op *7* DOM *n/a*

screen对象对应于显示浏览器视图的视频屏幕。尽管很多视频控制面板设定都会影响其属性值，但不同的浏览器品牌仅共享很少的一部分属性。

对象模型引用方式

NN

```
screen
```

IE

```
[window.]screen
```

对象特定属性 availHeight、availLeft、availTop、availWidth、bufferDepth、colorDepth、deviceXDPI、deviceYDPI、fontSmoothingEnabled、height、logicalXDPI、logicalYDPI、pixelDepth、updateInterval、width

对象特定方法 无。

对象特定事件 无。

availHeight, availWidth

IE *4* NN *4* Moz *all* Saf *all* Op *7* DOM *n/a*

只读

提供用户视频监视器的内容区域的高度和宽度，单位为像素。这个值并不包括24像素高的任务条（Windows），或者20像素高的系统菜单栏（Macintosh）。而IE/Macintosh会将菜单栏的高度错误地计算为24像素。另外，在使用这些值来模拟窗口最大化时，还须要调整窗口的左上角坐标。

示例

```
var newWind = window.open("","","height=" + screen.availHeight +
",width=" + screen.availWidth);
```

值 在水平和竖直空间上可用的整数像素值。

默认值 由用户的监视器尺寸决定。

availLeft, availTop

IE *n/a* NN *4* Moz *all* Saf *all* Op *n/a* DOM *n/a*

只读

为屏幕左侧和顶部边缘提供具体的坐标值，单位为像素。尽管在标准的Windows任务栏设定中这两个值均为0，但是依然会将任务栏拖至屏幕左侧或顶部，并且增大对应值以调节任务栏的空间。Navigator 4/Macintosh在固定的菜单栏之下才开始屏幕尺寸计算，而在Mac OS X系统中的Mozilla和Safari浏览器内，availTop属性返回22，表示菜单栏的高度。

示例

```
window.moveTo(screen.availLeft, screen.availTop);
```

值 整数值。

默认值 0（Windows）；20（Macintosh）。

bufferDepth

IE *4* NN *n/a* Moz *n/a* Saf *n/a* Op *n/a* DOM *n/a*

可读/可写

指明画外位图缓存设定。如果让bufferDepth与colorDepth两者数值相等，那么在某些客户端上可能让动画显示得更为平滑流畅。将bufferDepth设置为-1时，会强制IE浏览器根据屏幕的像素深度进行缓存，同时还

会也会自动将colorDepth设置为该值，如果将bufferDepth设定为其他可用值（1、4、8、15、16、24或32）则会让浏览器根据该像素深度进行缓存，而且还会将colorDepth也设置为相同的值。在实际应用中，应该使用更高的像素位设置客户端的显示，以便能够在脚本中使用较高的设定值。

示例 `screen.bufferDepth = 4;`

值 可使用如下整数值之一：-1 | 0 | 4 | 8 | 15 | 16 | 24 | 32。

默认值 0

colorDepth

IE *4* NN *4* Moz *all* Saf *all* Op *7* DOM *n/a*

只读

返回在视频监控器显示色彩时或图像缓存中所使用的每象素位数。尽管这个属性是只读的，但在IE中修改bufferDepth值也会对它产生影响。根据这个属性可以决定当前屏幕的色深并依此选择颜色。

示例

```
if (screen.colorDepth > 8) {
   document.getElementById("pretty").color = "cornflowerblue";
} else {
   document.getElementById("pretty").color = "blue";
}
```

值 整数值。

默认值 当前视频控制面板的设定值。

deviceXDPI, deviceYDPI, logicalXDPI, logicalYDPI

IE *6* NN *n/a* Moz *n/a* Saf *n/a* Op *n/a* DOM *n/a*

只读

这4个属性值都与显示器在水平（x）和竖直方向（y）上的每英寸点数解析度有关。设备密度属性会返回操作系统探测到的当前显示屏幕的实际像素密度。而逻辑密度则是大多数用户和页面编写者工作时所使用的“标准”像素密度，即在水平和竖直方向上均为每英寸96点。这两类属性可以让脚本检测用户是否使用了高于正常的像素密度，这可能导致固定尺寸的显示项在屏幕上显得很小，如图像以及字体等。在这种情况下，脚本可以在设备密度和逻辑密度之间确定一个比例因子，并将它应用于重要元素或整个document.body的style.zoom属性。使用高密度显示系统的用户可能已经设置了IE的应用程序选项，从而让IE能够自动进行比例转换，此时就不须要再进行类似的转换工作。

示例

```
var normDPI = 96;
if ((screen.deviceXDPI == screen.logicalXDPI) && (screen.deviceXDPI > normDPI)) {
   document.body.style.zoom = normDPI / screen.logicalXDPI;
}
```

值 整数值。

默认值 96

fontSmoothingEnabled

IE *4(Win)* NN *n/a* Moz *n/a* Saf *n/a* Op *n/a* DOM *n/a*

只读

如果用户在Windows显示控制面板中为字体显示启用了边缘平滑修正功能，那么将返回布尔值true。这一设定可能会影响文档中与字体相关的样式单设定。

示例

```
var styleFile = "css/corpStyle.css";
if (screen.fontSmoothingEnabled) {
   styleFile = "css/corpStyleFancy.css";
}
document.write("<link type='text/css' rel='stylesheet' href='" +
   styleFile + "'>");
```

值 布尔值：true | false。
默认值 false

height, width

IE 4 NN 4 Moz all Saf all Op 7 DOM n/a

只读

分别返回客户端视频监控器在水平和竖直方向上的像素值。注意，此时返回的是原始尺寸。如果要减去系统栏所占据的屏幕空间尺寸，请参见availHeight和availWidth。

示例

```
if (screen.height > 480 && screen.width > 640) {
    ...
}
```

值 整数像素值。
默认值 由视频监控器决定。

logicalXDPI, logicalYDPI

请参见deviceXDPI。

pixelDepth

IE n/a NN 4 Moz all Saf all Op 7 DOM n/a

可读/可写

返回在视频监控器显示色彩时使用的像素位数。这个值与colorDepth比较类似，但它并不会受自定义颜色面板的潜在影响。

示例

```
if (screen.pixelDepth > 8) {
    document.getElementById("pretty").color = "cornflowerblue";
} else {
    document.getElementById("pretty").color = "blue";
}
```

值 整数值。
默认值 当前视频控制面板的设定值。

updateInterval

IE 4 NN n/a Moz n/a Saf n/a Op n/a DOM n/a

提供屏幕更新的时间间隔，单位为毫秒。如果其值为0，则会让浏览器自动选择一个效果最好的平均刷新频率。这个时间间隔越长，那么缓存的动画内容越多，这样在每次显示当前状态时就会忽略这些缓存的数据。

示例

```
screen.updateInterval = 0;
```

值 任何大于或等于0的整数。
默认值 0

width

请参见height。

script

IE 4 NN n/a Moz all Saf all Op 7 DOM 1

script对象对应于script元素。在IE 4/Windows中，无法访问或设定innerHTML及innerText属性，但可以操作等价的text属性。IE 5/Macintosh则为这个对象实现了readyState属性（IE/Windows的所有元素均拥有

这个属性）。另外，如果在IE中使用src属性的setAttribu()方法来加载外部*.js*文件往往会存在问题，因此直接为src属性指定一个新的URL更为可靠。

等价HTML元素 <script>

对象模型引用方式 [window.]document.getElementById("*elementID*")

723

对象特定属性 charset、defer、event、htmlFor、src、text、type

对象特定方法 无。

对象特定事件

事件	IE	其他	DOM	事件	IE	其他	DOM
error	•	–	–	load	•	–	–

charset

IE *6* NN *n/a* Moz *all* Saf *all* Op *7* DOM *1*

可读/可写

指明脚本内容中所使用的字符编码。

示例

```
if (document.getElementById("myScript").charset == "csISO5427Cyrillic") {
   // 处理西里尔(cyrillic)字符集
}
```

值 是否区分大小写取决于字符集注册表，请访问以下链接获取更多详细信息：*http://www.iana.org/assignments/character-sets*。

默认值 由浏览器决定。

defer

IE *4* NN *n/a* Moz *all* Saf *all* Op *7* DOM *1*

可读/可写

指定浏览器在加载页面时是否应该直接显示正常的HTML内容，而不必查找脚本以生成相关内容。在运行时，须要在script元素标签内设定这个属性值。如果在元素标签内添加了defer属性，从而将这个属性值设置为true，那么浏览器在搜索document.write()指令时就不必解析script元素的内容而直接进行显示。而一旦文档加载完成，即使改变这个属性也不会对脚本或浏览器带来任何影响。尽管目前所有的主流浏览器均已实现了这个属性，但只有IE浏览器提供了完整的功能实现。

示例

```
if (document.getElementById("myScript").defer = = "true") {
   ...
}
```

724

值 布尔值：true | false。

默认值 false

event

IE *4* NN *n/a* Moz *all* Saf *all* Op *n/a* DOM *1*

只读

在event和for属性的帮助下，IE浏览器的事件模型允许将对象事件与script元素进行绑定，请查阅在线参考VI。event属性返回了元素中event属性的具体设定值。尽管某些其他浏览器也实现了这个属性，但它们均未提供相关的实际功能。

示例

```
if (document.getElementById("gizmoScript").event == "onresize") {
   ...
}
```

值 区分大小写的事件名称字符串。

默认值 无。

htmlFor

IE *4* NN *n/a* Moz *all* Saf *all* Op *n/a* DOM *1*

只读

返回script元素中for属性的值，即一个元素ID。这个属性指向一个元素的ID，触发一个特定事件（由event设定）时，该元素绑定的脚本就会处理这个事件。除IE之外，其他浏览器均未实现相关功能。

示例

```
if (document.getElementById("helpScript").htmlFor == "helpButton") {
    ...
}
```

值 字符串。

默认值 无。

src

IE *4* NN *n/a* Moz *all* Saf *all* Op *7* DOM *1*

可读/可写

提供当前script元素所导入的*.js*脚本文件的URL地址。如果在IE中为script元素指定了一个新的*.js*文件，原有*.js*文件的脚本并不会直接消失。但名称重复的属性和方法会被新文件中的定义所代替。

示例

```
if (document.getElementsByTagName("script")[1].src == "scripts/textlib.js") {
    ...
}
```

值 字符串形式的完整或相对URL。

默认值 无。

text

IE *4* NN *n/a* Moz *all* Saf *all* Op *9* DOM *1*

可读/可写

指明元素内的文本内容。须要注意的是，在不同浏览器中为这个对象指定脚本指令会产生不同的效果。Windows系统内新版本的IE浏览器会将新值添加到已有的脚本之中，Mozilla浏览器会忽略这些值，IE 5/Macintosh将这个属性视为只读属性，而Safari则只返回一个空字符串。

示例

```
var scriptText = document.getElementById("script3").text;
```

值 字符串。

默认值 无。

type

IE *4* NN *n/a* Moz *all* Saf *all* Op *7* DOM *1*

只读

本属性为脚本指令的类型给出了提示。该类型会通知浏览器使用哪种脚本引擎来解释脚本指令，例如“text/javascript”。与type相比，language属性则是用来定义编写元素指令时所使用的脚本语言种类，将来它最终将会被type属性所取代。

示例

```
var scriptMIMEtype = document.getElementById("script3").type;
```

值 字符串。

默认值 无。

scripts

IE *4* NN *n/a* Moz *all* Saf *all* Op *7* DOM *n/a*

此属性可以提供由文档内已定义或已导入的所有脚本组成的集合，而这些脚本可以既可以定义在head中也可以位于body部分之内。此集合中的成员按照它们在源代码中的次序进行排序。

对象模型引用方式 document.scripts
对象特定属性 length
对象特定方法 item()、namedItem()、tags()、urns()
对象特定事件 无。

726

length

IE 4 NN n/a Moz all Saf all Op 7 DOM n/a

只读

返回集合中元素的数量。

示例
```
var howMany = document.scripts.length;
```
值 整数值。

item( )

IE 4 NN n/a Moz all Saf all Op 7 DOM n/a

```
item(index[, subindex])
```

根据符合索引值的元素对象，返回一个单独的对象或一组对象集合。

返回值 一个对象或一个对象集合。如果并不存在与参数相匹配的对象，则返回null。

参数

index

当输入参数是一个从零开始的整数时，返回值是源代码内与某个具体项相对应的独立元素（内嵌在当前元素内）。如果参数为一个字符串，则返回id属性与该字符串相符的元素集合。

subindex

如果在IE中为第1个参数指定一个字符串值，那么可以将第2个参数指定为一个从零开始的索引值，这样就可以从集合中找到一个id属性与第1个参数的字符串相匹配的具体元素。

namedItem( )

IE 4 NN n/a Moz all Saf all Op 7 DOM n/a

```
namedItem("ID")
```

返回单个script对象或script对象集合，它们对应于符合参数字符串值的元素。

返回值 一个script对象或一个script对象集合。如果并不存在与参数相匹配的对象，则返回null。

参数

ID

与目标元素的id属性包含相同值的字符串。

727

tags( )

IE 4 NN n/a Moz all Saf all Op 7 DOM n/a

```
tags(tagName)
```

从当前集合内的全部对象中，返回标签与*tagName*参数相匹配的对象集合。此时，由于所有的元素均拥有相同的script标签，因此这是一个完全多余的方法。

返回值 对象集合。如果与参数相匹配的对象不存在，则返回一个长度为零的数组。

参数

tagName

小写字符形式的元素标签名字符串，如document.scripts.tags("script")。

urns( )

IE 5(Win) NN n/a Moz all Saf all Op 7 DOM n/a

```
urns(URN)
```

请参见all.urns()方法。

scrollbars

请参见directories。

select

IE 3 NN 2 Moz all Saf all Op 7 DOM 1

select对象对应于select元素。这是一个能够包含若干option元素的表单控件。注意，在Mozilla之前的Netscape Navigator浏览器中，无法使用disabled属性。

等价HTML元素 <select>

对象模型引用方式

```
[window.]document.formName.selectName
[window.]document.forms[i].elements[i]
[window.]document.getElementById("elementID")
```

对象特定属性 autofocus、data、dataFld、dataSrc、form、forms、labels、length、multiple、name、options[]、selectedIndex、selectedOptions、size、type、validationMessage、validity、value、willValidate

对象特定方法 add()、checkValidity()、dispatchChange()、dispatchFormChange()、item()、namedItem()、remove()、setCustomValidity()

对象特定事件

事件	IE	NN	Opera 9	其他	DOM
afterupdate	•	–	–	–	–
beforeupdate	•	–	–	–	–
change	•	•	•	•	•
formchange	–	–	•	–	–
forminput	–	–	•	–	–
invalid	–	–	•	–	–

autofocus

IE n/a NN n/a Moz n/a Saf n/a Op 9 DOM n/a

可读/可写

页面加载完成后，这个Web Forms 2.0扩展将会把焦点聚集在元素上。每个页面上只能有一个表单控件元素拥有此属性。

示例 document.getElementById("myselect").autofocus = true;

值 布尔值：true | false。

默认值 false

data

IE n/a NN n/a Moz n/a Saf n/a Op n/a DOM n/a

可读/可写

这个Web Forms 2.0扩展允许表单从一个外部XML文件中获取表单控件的初始值。Web Forms 2.0规范为使用XML文件提供了详细的结构和命名空间说明。如须获取更多的信息请访问此链接：*http://www.whatwg.org*。

值 URL字符串。

默认值 空字符串。

dataFld

IE 4 NN n/a Moz n/a Saf n/a Op n/a DOM n/a

可读/可写

此属性用于IE中的数据绑定，以便将一个远程数据源的列名同select对象的selectedIndex属性联系在一

起。此时元素中还必须设定datasrc属性。如果dataFld和dataSrc属性均被设置为空字符串，那么就会打破元素与数据源之间的绑定关系。注意，只能在IE 5/Mac操作系统下实现文本文件数据源的绑定。

729 **示例** `document.forms[0].mySelect.dataFld = "choice";`
值 数据源数据列内区分大小写的标识符。
默认值 无。

dataSrc

IE *4* NN *n/a* Moz *n/a* Saf *n/a* Op *n/a* DOM *n/a*

可读/可写

为实现IE数据绑定功能，此属性用于指定页面内object元素的ID值，以便在进行远程数据访问时加载数据源对象。如果dataFld和dataSrc属性均被设置为空字符串，那么就会打破元素与数据源之间的绑定关系。注意，只能在IE 5/Mac操作系统下实现文本文件数据源的绑定。

示例 `document.forms[0].mySelect.dataSrc = "DBSRC3";`
值 数据源内区分大小写的标识符。
默认值 无。

form

IE *3* NN *2* Moz *all* Saf *all* Op *7* DOM *1*

只读

返回包含当前元素的form元素的引用。在处理一个来自于该元素的事件时，对应的事件处理方法可以通过事件对象的target或srcElement属性来自动访问对应的select元素。而且通过读取form属性，脚本可以轻易地访问同一个表单中的其他控件。

示例 `var theForm = evt.srcElement.form;`
值 form元素对象的引用。
默认值 无。

forms

IE *n/a* NN *n/a* Moz *n/a* Saf *n/a* Op *9* DOM *n/a*

只读

这个Web Forms 2.0扩展返回一个对象引用数组（NodeList），与当前select元素相关联的form对象组成该数组。

示例 `var formList = document.getElementById("mySelect").forms;`
值 数组。
默认值 包含一个封闭表单元素的引用。

labels

IE *n/a* NN *n/a* Moz *n/a* Saf *n/a* Op *9* DOM *n/a*

只读

这个Web Forms 2.0扩展返回一个对象引用数组（HTMLCollection），与当前select元素相关联的label元素组成该数组。

730 **示例** `var allLabels = document.getElementById("mySelect").labels;`
值 label元素对象引用组成的数组。
默认值 空数组。

length

IE *3* NN *2* Moz *all* Saf *all* Op *7* DOM *1*

可读/可写

表示select对象中内嵌的option对象的数量。这个属性的返回值与select对象的options.length属性值相

同，并且能够安全地使用在for循环中以遍历内嵌的option对象。尽管W3C DOM声明这个属性是只读的，但在主流浏览器中有时也可以对它进行写操作，因此实际上可以调整其属性值。即使要修改这个这个属性，唯一的可能性也只是将它设置为0，以便从select对象中清空所有的选项。另外，如果使用IE 5及后续版本或W3C DOM浏览器，那么在编程时最好使用select.remove()和select.add()方法来修改内嵌在select元素中的option元素。

示例 `document.forms[0].mySelect.length = 0;`
值 整数值。
默认值 无。

multiple

IE *4* NN *n/a* Moz *all* Saf *all* Op *7* DOM *1*

可读/可写

指明浏览器是否用列表框的形式显示select元素，并允许用户在选项列表中选择多个选项。在默认情况下，size属性值与内嵌的option元素个数相当，但也可以修改其属性值设定。将multiple属性设置为false，并将size设置为1，就可以将滚动列表转变为弹出菜单的形式。用户按住Shift键后，就可以选择第1个和最后1个条目之间的连续项。如果要选择非连续项，那么Windows用户必须在选中每项时按住Ctrl键，而Mac用户则须要按下Command键。当size设置为1，从而显示一个弹出菜单时，multiple属性将会失效。

示例

```
if (document.entryForm.list3.multiple) {
    ...
}
```

值 布尔值：true | false。
默认值 false

name

IE *3* NN *2* Moz *all* Saf *all* Op *7* DOM *1*

可读/可写

这是一个与表单控件相关的标识符。提交表单时，此属性的值将作为名称/值对的一部分而被提交至服务器。由于根据控件类型可以通过其他手段指定控件标注，因此从用户的角度来看，控件名称是被隐藏的。与此同时，脚本也可以使用控件名称来引用对象。虽然新标准倾向于使用id属性，但很多浏览器依然须要为表单控件设置name属性，以便允许表单控件的值被正常提交。

示例 `document.orderForm.payment.name = "credcard";`
值 区分大小写的标识符字符串，必须遵循标识符命名规则：不包含空白、不能以数字作为起始字符、除下划线外不允许使用其他标点符号。
默认值 无。

options[ ]

IE *3* NN *n/a2* Moz *all* Saf *all* Op *7* DOM *1*

只读

返回当前元素内所有option对象组成的数组，在W3C DOM Level 2中，这个数组实质上是一个HTMLOptionsCollection对象。数组中的条目会根据源代码顺序从0开始建立索引。关于如何在保证后向兼容的情况下向这个集合中添加或删除option元素，请参见options对象。此外，在允许多项选择时，须要在select元素中遍历这个集合。

示例

```
var selVals = new Array();
for (var i = 0; i < document.forms[0].mySelect.length; i++) {
    if (document.forms[0].mySelect.options[i].selected) {
```

```
        selVals[selVals.length] = document.forms[0].mySelect.options[i].value;
    }
}
```

值 option对象数组。

默认值 无。

selectedIndex

IE 3 NN 2 Moz all Saf all Op 7 DOM 1

可读/可写

这个属性可以表示用户选中的选项的索引值。如果允许select元素进行多项选择，那么selectedIndex属性会返回第一个选中项的索引，请参见selected属性。如下文中的范例代码所示，可以使用这个属性来访问选中项的值或文本。

在现在浏览器中，如果不存在选中项，那么selectedIndex属性会返回0。另外，除Safari浏览器之外，将这个值设定为-1均会取消所有的选中项，并且显示效果也会恢复到未选中状态。

示例

```
var list = document.forms[0].selectList;
var listText = list.options[list.selectedIndex].text;
```

值 整数值。

默认值 0

size

IE 4 NN n/a Moz all Saf all Op 7 DOM 1

可读/可写

此属性可以控制每次出现在滚动选择列表中的选项的数量，它对应于select元素的size属性。将multiple属性为ture后，它会使用一个比实际选项数更小的值来替代size的值。将multiple属性设置为false，并将size设置为1，就可以将滚动列表转变为弹出菜单的形式。

示例

```
document.forms[0].choices.size = 6;
```

值 整数值。

默认值 无。

type

IE 4 NN 3 Moz all Saf all Op 7 DOM 1

只读

返回表单控件元素的类型。根据select元素是否能够进行多项选择，对应的select对象可以拥有两种不同的值。这个属性返回值中的字符均为小写形式。有时可能还须要遍历所有的表单元素，以便找到特定的类型执行某些操作，例如，在将所有text类型的表单控件清空的同时不改变其他控件。

值得注意的是，当select元素的size值大于1时，即使未设置multiple属性，Navigator 4仍然会错误地将这个元素的类型报告为select-multiple。

示例

```
if (document.forms[0].elements[3].type == "select-multiple") {
    ...
}
```

值 下列任意一个字符串常量之一：button | checkbox | file | hidden | image | password | radio | reset | select-multiple | select-one | submit | text | textarea。

默认值 由multiple的值决定。

validationMessage

IE *n/a* NN *n/a* Moz *n/a* Saf *n/a* Op *9* DOM *n/a*

只读

如果表单控件在Web Forms 2.0规范下校验失败，那么这个Web Forms 2.0的扩展属性会返回一个由浏览器生成的消息。由于返回空字符串就表示校验正常，这使得这个属性对面向文本的表单控件而言更有意义。

值 空字符串。

默认值 空字符串。

validity

IE *n/a* NN *n/a* Moz *n/a* Saf *n/a* Op *9* DOM *n/a*

只读

这个Web Forms 2.0扩展返回一个ValidityState对象。请参见ValidityState对象。

值 ValidityState对象。

默认值 ValidityState对象。

value

IE *4* NN *n/a* Moz *all* Saf *all* Op *7* DOM *1*

可读/可写

指明与表单控件相关的当前值，该值将与元素的名称/值对一起被提交至服务器端。虽然所有的值都是字符串形式，但它们可以表示其他类型的数据，如布尔量和数字值。在早期的浏览器（Mozilla之前的Netscape和IE 4之前的各种版本）中，脚本必须首先将select元素的selectedIndex属性作为索引从options数组中获得已选中的选项，然后再从每个option对象的selected属性中找到值为true的选中项，这样才能获得选中项的值。

示例

```
if (document.forms[0].medium.value == "CD-ROM") {
    ...
}
```

值 字符串。

默认值 无。

add( )

IE *5* NN *n/a* Moz *all* Saf *all* Op *7* DOM *1*

add(*newOptionElement*[, *positionIndex*]), add(*newOptionElement*, *optionElementReference*)

向当前select元素中添加一个新的option元素。然而不幸的是，IE和W3C DOM无法在方法参数的使用上达成一致。虽然所有的浏览器都需要一个新创建的option元素的引用，但第2个参数值会随浏览器的不同而发生改变。在IE中，第2个参数是可选的，它可以提供一个现有option元素的数值索引，并会将新的option元素插入到该元素之前。如果不设置第2个参数，那么新的option元素会附加到现有的option元素之后。但在W3C DOM浏览器中则必须使用第2个参数。此时，这个参数既可以是一个已存在的option元素的引用（在该option前插入新选项），也可以为null（将新option附加至末尾）。

返回值 无。

参数

newOptionElement

由脚本创建的option元素的引用（通常会使用document.createElement()方法创建新的选项）。

positionIndex

可选的IE整数参数，它指定一个现有option元素，此方法会将新的option元素插入到该元素之前。如果忽略这个参数或将它设置为-1就会将新的option元素附加到选项列表末尾。

optionElementReference

一个现有option元素的引用，新option元素会插入到该元素对象之前。也可以将这个值设置为null，从而将新的option放置在列表末尾。

checkValidity( )

IE *n/a* NN *n/a* Moz *n/a* Saf *n/a* Op *9* DOM *n/a*

这个Web Forms 2.0方法会返回一个布尔值，以表明表单控件是否能够达到其校验标准，归根到底，就是判断validity.valid属性是否为true。

返回值 布尔值：true | false。

参数 无。

dispatchChange( ), dispatchFormChange( )

IE *n/a* NN *n/a* Moz *n/a* Saf *n/a* Op *9* DOM *n/a*

这两个Web Forms 2.0方法为当前元素触发了change和formchange事件。因此，让mouseup事件处理器调用这两个方法之一，就可以将一个mouseup事件转换成为一个change或formchange事件。

返回值 无。

参数 无。

item( )

IE *5* NN *n/a* Moz *all* Saf *n/a* Op *7* DOM *1*

```
item(index[, subindex]), item(index)
```

根据符合索引值的元素对象，返回一个单独的内嵌option对象或一组内嵌option对象集合。

返回值 一个对象或一个对象集合。如果并不存在与参数相匹配的对象，则返回null。

参数

index

当输入参数是一个从零开始的整数时，返回值是源代码内与某个具体项相对应的独立元素（内嵌在当前元素内）。如果参数为一个字符串，则返回id属性与该字符串相符的元素集合。

subindex

如果在IE中为第1个参数指定一个字符串值，那么可以将第2个参数指定为一个从零开始的索引值，这样就可以从集合中找到一个id属性与第1个参数的字符串相匹配的具体元素。

namedItem( )

IE *6* NN *n/a* Moz *all* Saf *n/a* Op *7* DOM *1*

```
namedItem("ID")
```

返回单个内嵌option对象或内嵌option对象集合，它们对应于符合参数字符串值的元素。

返回值 一个option对象或一个option对象集合。如果并不存在与参数相匹配的对象，则返回null。

参数

ID

与目标元素的id属性包含相同值的字符串。

remove( )

IE *5* NN *n/a* Moz *all* Saf *all* Op *7* DOM *1*

```
remove(positionIndex)
```

根据参数指定的索引位置，将对应的option元素从当前select元素中删除。除了将select对象的length属性设置为0之外，还可以通过一个简单的循环结构删除所有的已有选项，如下所示：

```
while (selectElemRef.length > 0) {
    selectElemRef.remove(0);
}
```

从这一点来看，可以通过add()方法及options对象所讨论的各种不同途径达到增加新选项的目的。

返回值 无。

参数

positionIndex

从零开始的整数，指明须要删除内嵌选项集合中的哪一项。

setCustomValidity( )

IE *n/a* NN *n/a* Moz *n/a* Saf *n/a* Op *9* DOM *n/a*


```
setCustomValidity([errorString])
```

这个Web Forms 2.0方法可以设置validity属性（它本身就是一个ValidityState对象）的customError布尔值。但它对button元素无效，调用后反而会抛出一个NOT_SUPPORTED_ERR错误。

返回值 无。

参数

errorString

如果此参数为null或空字符串，将重新设置validity对象的customError属性，即表示表单控件无效。在当前会话期内，由于浏览器会记录一条错误信息，因此验证失败后会显示这条信息。

selection

IE *4* NN *n/a* Moz *all* Saf *all* Op *7* DOM *1*

selection对象代表文档中已经被用户或脚本选中的字符。不同的浏览器中这个对象也拥有不同的实体，而且脚本可以从两种互不兼容的方法出发来访问选中的内容。

在IE/Windows中，可以通过document.selection属性创建一个selection对象。然后只用通过selection的createRange()方法生成一个TextRange对象就可以对selection中的相关内容进行实际的操控。与此同时，使用TextRange的属性和方法就可以与选中内容进行交互。另外，还可以使用TextRange对象的select()方法将它转换成页面上的一段可视文本。然而我们只能在Win32版本的浏览器中建立起TextRange与IE selection对象之间的这种紧密联系。在IE的selection对象中，它还可以包含input和textarea元素中的选中文本。

但在IE/Macintosh中，并不存在selection对象。不过在这个环境中，实现了Navigator 4的document.getSelection()方法，并返回选中文本的字符串内容。说到这里，Mozilla浏览器也能支持这个Navigator 4方法，但是并不建议大家使用它，一旦使用就会在JavaScript控制台中显示一条警告。

而在Mozilla中，则可以通过window.getSelection()方法来创建一个selection对象。Mozilla的selection对象模拟了W3C DOM的Range对象的很多方法和属性。事实上，脚本正是通过Range对象才能在页面上高亮显示不连续的文本内容，此时脚本首先会创建一个Range对象，然后使用selection对象的addRange()方法将该Range添加到高亮显示的文本之中。而且Mozilla只能在主体内容上进行文本选择操作，而无法同时对可编辑文本框内的文本进行选择。

Safari对selection对象的支持非常有限，它几乎只能返回已选中的文本。Opera 8及其后续浏览器则紧跟IE对象的步伐，可以从选中文本中创建一个IE的TextRange对象，而Opera 9还实现了Mozilla的selection对象。在不同浏览器中，用户通过点击一个按钮或链接就取消当前文本选择的操作也不罕见。因此在这些浏览器中，所有与文本选择相关的脚本动作应该由onselect或onmouseup事件来触发，或通过计时器调用的方法来触发，请参见第5章中的window.setTimeout()方法。

对象模型引用方式

IE（Win）、Opera

```
document.selection
```

Mozilla、Safari、Opera 9

```
window.getSelection()
```

对象特定属性 anchorNode、anchorOffset、focusNode、focusOffset、isCollapsed、rangeCount、type、typeDetail

对象特定方法 addRange()、clear()、collapse()、collapseToEnd()、collapseToStart()、containsNode()、createRange()、

createRangeCollection()、deleteFromDocument()、empty()、extend()、getrangeAt()、removeAllRanges()、removeRange()、selectAllChildren()、selectionLanguageChange()、toString()

对象特定事件 无。

anchorNode, focusNode

IE *n/a* NN *n/a* Moz *all* Saf *n/a* Op *9* DOM *n/a*

只读

分别返回用户进行选择时的起始节点（anchor）和结束节点（focus）的引用。通常这些节点的类型都是文本节点。如果使用addRange()方法设置或扩展了文本选择，那么这两个属性将指向最近添加的文本区域所在的节点边界。

示例

```
var anchor = selectionRef.anchorNode;
if (anchor.nodeType == 3 && anchor.parentNode.tagName == "td") {
    // 处理位于表格单元格中的起始点
}
```

值 文档树节点的引用，null表示无选中内容。

默认值 null

anchorOffset, focusOffset

IE *n/a* NN *n/a* Moz *all* Saf *n/a* Op *9* DOM *n/a*

只读

返回从选中区域的起始定位节点或聚焦节点开始计数的字符位置或节点位置，请参见anchorNode和focusNode属性。如果节点是一个文本节点，那么以字符为偏移单位，当节点是元素节点时，则会以节点作为偏移单位。这种情况和Range对象的偏移量属性比较类似。通常这些属性值都是对文本节点内的字符进行计数。如果使用addRange()方法设置或扩展了文本选择，那么这两个属性将指向最近添加的文本区域所在的节点偏移量。

示例 `var selStartOffset = selectionRef.anchorOffset;`

值 整数值。

默认值 0

isCollapsed

IE *n/a* NN *n/a* Moz *all* Saf *n/a* Op *9* DOM *n/a*

只读

如果选中区域的起始边界和结束边界相同，那么返回布尔值true。

示例

```
if (selectionRef.isCollapsed) {
    // 选中域目前是一个插入点
}
```

值 布尔值：true | false。

默认值 true

rangeCount

IE *n/a* NN *n/a* Moz *all* Saf *n/a* Op *9* DOM *n/a*

只读

返回选中区域内Range对象的数量，在Mozilla中这些对象可以是不连续的。用户手动选择时通常只会产生一个Range，但使用addRange()方法可以将多个不连续的域添加到选中内容中。如果要检查每个高亮显示的分段的属性，请使用getrangeAt()方法。

示例 `var howMany = selectionRef.rangeCount;`
值 整数值。
默认值 0

type

IE *4(Win)* NN *n/a* Moz *n/a* Saf *all* Op *n/a* DOM *n/a*

只读

指明当前selection对象是包含若干个选中的字符还是仅仅包含一个插入点。

示例

```
if (document.selection.type == "Text") {
    ...
}
```

值 在IE中可以使用下面3个字符串常量之一：None | Text | Control。只有在使用HTML动态编辑并且可以进行控件选择时才会出现“Control”。在Safari中则可以使用下面3个字符串常量之一：None | Caret | Range。
默认值 无。

typeDetail

IE *5.5* NN *n/a* Moz *n/a* Saf *n/a* Op *n/a* DOM *n/a*

只读

对于那些可能须要使用IE浏览器组件的应用而言，这个属性通常会充当占位符的角色。这种应用在需要时还可以提供额外的选择类型信息。

addRange( )

IE *n/a* NN *n/a* Moz *all* Saf *n/a* Op *9* DOM *n/a*

`addRange(RangeReference)`

这个方法可以将一个Range对象转换成为页面上的一个高亮选中区域。如果需要，还可以将任意多个离散的区域添加到选中域中。每次调用此方法，都会增加selection对象的rangeCount属性。此外，selection中的各个Range还允许发生重叠。

```
var selRef = window.getSelection();
var rng = document.createRange();
rng.selectNodeContents(document.getElementById("myP"));
selRef.addRange(rng);
```

返回值 无。
参数

RangeReference
由Range对象方法创建的一个拥有边界点的Range对象的引用。

clear( )

IE *4(Win)* NN *n/a* Moz *n/a* Saf *n/a* Op *n/a* DOM *n/a*

清空文档内当前选中域中的全部内容。例如在下面这段代码中，如果用户选中了p元素中的内容，那么两秒钟后元素标签内定义的事件处理程序就会删除p元素中所有已选中的文本内容。

```
<p onselectstart="setTimeout('document.selection.clear()',2000);">
```

返回值 无。
参数 无。

collapse( )

IE *n/a* NN *n/a* Moz *all* Saf *n/a* Op *9* DOM *n/a*

`collapse(nodeReference, offset)`

此方法将当前选中区域缩小到两个输入参数所指定的点上。此后，之前的全部高亮选中区域都会恢复到普通的显示状态。

返回值 无。

参数

nodeReference

文档树中的一个文本或元素节点引用，选中区域将会被移动到该节点中。

offset

用以指定nodeReference节点中某个字符或节点的整数值，选中域将移动到它指定的位置。具体的数值与nodeReference节点有关，对文本节点而言，计数单位是字符，而对元素节点而言，计数单位是节点。

collapseToEnd( ), collapseToStart( )

IE *n/a* NN *n/a* Moz *all* Saf *n/a* Op *9* DOM *n/a*

此两个方法可以分别将当前选中区域缩小至selection对象的起始点（collapseToStart()）或结束点（collapseToEnd()）。此后，之前的全部高亮选中区域都会恢复到普通的显示状态。如果选中内容由多个区域组成，那么会根据组合选中区域的最外层边缘来决定这些方法中使用的起始和结束边界。选中域缩小后，它就只包含一个区域，即一个Range对象。

返回值 无。

参数 无。

containsNode( )

IE *n/a* NN *n/a* Moz *all* Saf *n/a* Op *9* DOM *n/a*

containsNode(*nodeReference*, *entirelyFlag*)

如果当前selection对象包含参数传入的节点，那么将返回布尔值true。第2个参数应该可以用于调整这种包含定义的松紧度，但从这个方法的具体反应来看它实际上退化成了一种标志。实质上将null作为第2个参数就可以完全保证此方法的正确性，此时这个方法会定义一种完全包含该节点的包含关系。

返回值 布尔值：true | false。

参数

nodeReference

文档树中任意一个可寻址的文本或元素节点的引用。

entirelyFlag

布尔值或null。将这个参数设置为true时表示，选中域只包含节点的部分内容时此方法将会返回true。

createRange( )

IE *4(Win)* NN *n/a* Moz *n/a* Saf *n/a* Op *8* DOM *n/a*

从当前selection对象中创建一个TextRange对象。此时可执行如下指令：

```
var myRange = document.selection.createRange();
```

此后脚本即可调用已选中的文本内容。

返回值 TextRange对象。

参数 无。

createRangeCollection( )

IE *5.5* NN *n/a* Moz *n/a* Saf *n/a* Op *n/a* DOM *n/a*

创建一个TextRange集合对象。使用此方法时必须要考虑到IE还能够支持选中多个离散区域这一特殊情况。

返回值 TextRange集合对象。

参数 无。

deleteFromDocument( )

IE *n/a* NN *n/a* Moz *all* Saf *n/a* Op *9* DOM *n/a*

从文档树中删除当前的selection节点。调用此方法后，会根据与Range.deleteContents()相同的规则调整节点的层次结构。

返回值 无。
参数 无。

empty( )

IE *4(Win)* NN *n/a* Moz *n/a* Saf *n/a* Op *n/a* DOM *n/a*

取消当前的选中区域，并将selection对象的type属性设置为None。但这并不会影响选中区域中的具体内容。

返回值 无。
参数 无。

extend( )

IE *n/a* NN *n/a* Moz *all* Saf *n/a* Op *9* DOM *n/a*

```
extend(nodeReference, offset)
```

根据参数传入的文档树节点和偏移量，将选中区域的结束边界移动到指定位置。但此方法并不会改变起始点的位置。

返回值 无。
参数

nodeReference
文档树中的一个文本或元素节点引用，选中区域的结束点将移动到该节点。

offset
用以指定*nodeReference*节点中某个字符或节点的整数值，选中域将移动到它指定的位置。具体的数值与*nodeReference*节点有关，对文本节点而言，计数单位是字符，而对元素节点而言，计数单位是节点。

getRangeAt( )

IE *n/a* NN *n/a* Moz *all* Saf *n/a* Op *9* DOM *n/a*

```
getRangeAt(rangeIndex)
```

根据参数传入的从零开始的索引值，返回selection对象中索引值与它相匹配的选中域的引用。对于相邻的多个选中域而言，这个参数应该为0。但对于离散的多个选中域，那些以Range对象的形式单独插入选中域中的所有区域都会被getrangeAt()返回。在实际使用过程中，请首先访问selection.rangeCount属性来获得selection对象所包含的Range对象的总数。此外，调用这个方法并不会打乱选中域中各区域的排列顺序。

返回值 一个Range对象引用。
参数

rangeIndex
一个从零开始的整数索引值。

removeAllRanges( )

IE *n/a* NN *n/a* Moz *all* Saf *n/a* Op *9* DOM *n/a*

删除当前选中域中的全部Range对象，此时文档树中的节点并不会受到影响。调用这个方法后，选中域会缩小到一点，并且其rangeCount属性值会变为0。

返回值 无。
参数 无。

removeRange( )

IE *n/a* NN *n/a* Moz *all* Saf *n/a* Op *9* DOM *n/a*

```
removeRange(rangeReference)
```

删除当前选中域中的某个Range对象，此时文档树中的节点并不会受到影响。如果在选中域中包含多个区域，那么可以遍历所有的Range对象，并根据某种判断标准从中删除一个或多个对象，代码如下：

```
var oneRange;
var sel = window.getSelection();
```

```
for (var i = 0; i < sel.rangeCount; i++) {
    oneRange = sel.getRangeAt(i);
    if (oneRange.someProperty == someDiscerningValue) {
        sel.removeRange(oneRange);
    }
}
```

返回值 无。

参数

rangeReference

一个已添加到当前选中域中的Range节点的引用。

selectAllChildren( )

IE *n/a* NN *n/a* Moz *all* Saf *n/a* Op *9* DOM *n/a*

```
selectAllChildren(elementNodeReference)
```

此方法会让selection对象包含参数传入的元素对象及其所有子节点。它实际上为通过脚本选中一个元素节点提供了一种便捷途径。在一个元素节点上使用这个方法后，会使得该元素节点成为选中域的定位节点及聚焦节点。但是将一个文本节点引用作为参数传入此方法后，会使得选中域缩小到一个点，并位于文本节点的首字符之前。如果在一个已存在的选中域上调用这个方法，那么包含该元素的新区域将替代原有的全部区域。

返回值 无。

参数

elementNodeReference

文档树中一个已变为选中域的元素节点的引用。

selectionLanguageChange( )

IE *n/a* NN *n/a* Moz *all* Saf *n/a* Op *9* DOM *n/a*

```
selectionLanguageChange(RTLFlag)
```

这个方法可以控制光标的方向性。

返回值 无。

参数

RTLFlag

布尔值：true表示从由到左，而false则表示从左到右。

toString( )

IE *n/a* NN *n/a* Moz *all* Saf *all* Op *8* DOM *n/a*

返回一个仅包含选中域中主题内容的字符串。这个字符串会忽略所有的标签和属性。

返回值 字符串。

参数 无。

small

请参见b。

span

743

744

IE *4* NN *n/a* Moz *all* Saf *all* Op *7* DOM *1*

span对象对应于span元素。这个元素主要是作为行内内容元素的样式规则的一种专用容器。它实际上也是一种典型的行内元素对象。从脚本编程的角度来看，Navigator 4中拥有定位样式的span元素和layer对象很类似。

等价HTML元素

对象模型引用方式 [window.]document.getElementById("*elementID*")

对象特定属性 无。
对象特定方法 无。
对象特定事件 无。

statusbar

请参见directories。

strike

请参见b。

strong

请参见abbr。

style (element)

IE *4* NN *n/a* Moz *all* Saf *all* Op *7* DOM *1*

style元素对象对应于HTML元素style。目前已经将这个对象从style对象中剥离出来，实际上文档内的每个元素都已经将它作为自身的一个属性。通过文档中的<style>标签就能够创建style元素对象，它拥有一个唯一的ID值，并且这个style对象还包含对应元素中所设置的所有样式属性及它们的当前值。

等价HTML元素 <style>
对象模型引用方式 [window.]document.getElementById("*elementID*")
对象特定属性 disabled、media、sheet、styleSheet、type
对象特定方法 无。
对象特定事件

事件	IE	其他	DOM	事件	IE	其他	DOM
error	•	–	–	load	•	–	–

disabled

IE *4* NN *n/a* Moz *all* Saf *all* Op *7* DOM *1*

可读/可写

指定是否应该将样式单中的规则应用于已选中的元素。在页面开发过程中，可以创建一个按钮来控制样式单效果的开启与禁用，这样就可以在各种浏览器中简便地观察不同的页面效果。

示例 document.getElementById("mainStyle").disabled = true;
值 布尔值：true | false。
默认值 false

media

IE *4* NN *n/a* Moz *all* Saf *all* Op *7* DOM *1*

可读/可写

本属性用于指明style元素中各种样式规则所需要的输出设备。当浏览器能够对文档内容进行修正，以便适合不同的设备（如手持电脑、文本-语音转换器或模糊电视设备等）时，media属性才会大有作用。

示例 document.getElementById("myStyle").media = "print";
值 可以使用以下字符串常量，如果须要设置多个值，请使用逗号进行分隔：all | print | screen。
默认值 all

sheet

IE *n/a* NN *n/a* Moz *all* Saf *all* Op *9* DOM *n/a*

只读

返回一个代表style元素定义的样式单的styleSheet对象，W3C DOM中的对应类型为CSSStyleSheet。虽然这种做法并不标准，但它的确为引用styleSheet对象提供了另外一种途径。但还是请使用document.styleSheets来访问样式单。

示例 `var oneSheet = document.getElementById("myStyle").sheet;`

值 styleSheet对象（W3C DOM中的CSSStyleSheet对象）的引用。

默认值 无。

746

styleSheet

IE *6* NN *n/a* Moz *n/a* Saf *n/a* Op *n/a* DOM *n/a*

只读

返回一个代表style元素定义的样式单的styleSheet对象。尽管浏览器中已出现了这个属性，但它似乎并未得到微软的官方支持。因此还是请使用document.styleSheets来访问样式单。

示例 `var oneSheet = document.getElementById("myStyle").styleSheet;`

值 styleSheet对象的引用。

默认值 无。

type

IE *4* NN *n/a* Moz *all* Saf *all* Op *7* DOM *1*

可读/可写

它表示style元素的type属性所指定的样式单MIME类型。

示例

```
if (document.getElementById("myStyle").type == "text/css") {
    // 一般不可能是其他样式类型
}
```

值 MIME类型字符串。

默认值 text/css

style, CSSStyleDeclaration

IE *4* NN *n/a* Moz *all* Saf *all* Op *7* DOM *2*

从最一般的意义上来看，style对象是通过脚本读写指定元素的独立CSS属性的入口点。这个style对象暴露（或有可能暴露）了浏览器所支持的每一个样式单属性。

但在实际使用中，从HTML元素对象的style属性中获取的style对象受到很大的限制，它只能反映已通过style属性明确定义在元素标签中的CSS设定或使用脚本为元素对象的style属性所指定的样式规则。但是那些与浏览器和文档相关的样式单也会影响元素的显示效果，比如内置的样式单，<style>元素中定义的样式单规则，以及link元素或@import规则导入的外部样式规则等。当然，有时也可以读取影响某个元素的CSS属性集合，但必须通过浏览器所指定的语法才能获得这些有效的样式定义。IE及Opera均使用元素的currentStyle属性，而W3C DOM浏览器则使用对应的document.defaultView.getComputedStyle()方法。尽管这两个方法所使用的语法不同，但它们都能返回一个对象，以便脚本检查每个有效的CSS属性值。

747

IE有3个与样式相关的对象——style、currentStyle和runtimeStyle，它们都能返回一个style对象并通过它暴露对应的CSS样式属性，而W3C DOM中的情况则略微复杂一些。Mozilla、Safari和Opera均实现了W3C DOM的CSSStyleDeclaration对象，并通过它向脚本操作开放所有的CSS属性。和IE的处理方式类似，此时还是须要通过元素对象的style属性以达到访问样式的目的，这也使得它能够在跨浏览器的应用中得以使用。

但在1.4版之前的Mozilla浏览器及Safari浏览器中，使用document.defaultView.getComputedStyle()读取有效的样式单时，返回的对象并未直接将CSS属性暴露为可进行脚本操作的属性。此时必须使用CSSStyleDeclaration的方法通过属性名称来检查对应的属性值。尽管这种方式比较繁琐，但它的确能够与贯穿在W3C DOM中的其他属性读取语法保持高度的一致性。幸运的是，从Mozilla 1.4和Opera 8开始，getComputedStyle()返回的CSSStyleDeclaration对象已经将所有的CSS属性都暴露为可脚本操作的属性。尽管这些属性值都是只读状态，但在脚本编程时完全可以根据一个只读的值来做出“if then”判断。此外，一旦通过元素对象的style属性设置了一个CSS属性，那么就可以在跨浏览器的环境中读取这个属性值，从而避免浏览器兼容性所带来的问题。

本节中列出了style对象的可用属性，以及W3C DOM的正式方法，可以通过这些方法及时地访问对应的属性。W3C DOM在一个名为CSS2Properties的对象中列出了style对象的大部分属性。W3C DOM规范提出这个CSS2Properties对象实际上为浏览器提供了一定的便利，在当前的Mozilla和Opera浏览器内部，这个对象暴露了已通过计算的样式属性。

下文中列出的style对象的脚本属性都对应于CSS属性。因此如果要获得某个特定属性的更多信息，不妨参见第4章中的对应内容。

对象模型引用方式

All

```
[window.]document.getElementById("elementID").style
```

IE

```
[window.]document.styleSheets[i].rules[j].style
[window.]document.styleSheets[i].rules[j].currentStyle
[window.]document.styleSheets[i].rules[j].runtimeStyle
```

W3C DOM

```
[window.]document.styleSheets[i].cssRules[j].style
```

对象特定属性

accelerator、azimuth、background、backgroundAttachment、backgroundColor、backgroundImage、backgroundPosition、backgroundPositionX、backgroundPositionY、backgroundRepeat、behavior、blockDirection、border、borderBottom、borderBottomColor、borderBottomStyle、borderBottomWidth、borderCollapse、borderColor、borderLeft、borderLeftColor、borderLeftStyle、borderLeftWidth、borderRight、borderRightColor、borderRightStyle、borderRightWidth、borderSpacing、borderStyle、borderTop、borderTopColor、borderTopStyle、borderTopWidth、borderWidth、bottom、captionSide、clear、clip、clipBottom、clipLeft、clipRight、clipTop、color、content、counterIncrement、counterReset、cssFloat、cssText、cue、cueAfter、cueBefore、cursor、direction、display、elevation、emptyCells、filter、font、fontFamily、fontSize、fontSizeAdjust、fontStretch、fontStyle、fontVariant、fontWeight、height、imeMode、layoutFlow、layoutGrid、layoutGridChar、layoutGridLine、layoutGridMode、layoutGridType、left、length、letterSpacing、lineBreak、lineHeight、listStyle、listStyleImage、listStylePosition、listStyleType、margin、marginBottom、marginLeft、marginRight、marginTop、markerOffset、marks、maxHeight、maxWidth、minHeight、minWidth、MozBorderRadius、MozBorderRadiusBottomleft、MozBorderRadiusBottomright、MozBorderRadiusTopleft、MozBorderRadiusTopright、MozOpacity、

749

opacity、orphans、outline、outlineColor、outlineOffset、outlineStyle、outlineWidth、overflow、overflowX、overflowY、padding、paddingBottom、paddingLeft、paddingRight、paddingTop、page、pageBreakAfter、pageBreakBefore、pageBreakInside、parentRule、pause、pauseAfter、pauseBefore、pitch、pitchRange、pixelBottom、pixelHeight、pixelLeft、pixelRight、pixelTop、pixelWidth、playDuring、posBottom、posHeight、position、posLeft、posRight、posTop、posWidth、quotes、richness、right、rubyAlign、rubyOverhang、rubyPosition、scrollbar3dLightColor、scrollbarArrowColor、scrollbarBaseColor、scrollbarDarkShadowColor、scrollbarFaceColor、scrollbarHighlightColor、scrollbarShadowColor、scrollbarTrackColor、size、speak、speakHeader、speakNumeral、speakPunctuation、speechRate、stress、styleFloat、tableLayout、textAlign、textAlignLast、textAutospace、texTDecoration、texTDecorationBlink、texTDecorationLineThrough、texTDecorationNone、texTDecorationOverline、textdecorationUnderline、textIndent、textJustify、textKashidaSpace、textOverflow、textShadow、texttransform、textUnderlinePosition、top、unicodeBidi、verticalAlign、visibility、voiceFamily、volume、whiteSpace、widows、width、wordBreak、wordSpacing、wordWrap、writingMode、zIndex、zoom

对象特定方法 getPropertyCSSValue()、getPropertyPriority()、getPropertyValue()、item()、removeProperty()、setProperty()

对象特定事件 无。

accelerator

IE *5(Win)* NN *n/a* Moz *n/a* Saf *n/a* Op *n/a* DOM *n/a*

详见下文

在Windows 2000和更新的Windows系统中，IE 5及后续版本的用户可以对选项进行设置，以便在用户按下Alt键时高亮显示各种指令的快捷键。accelerator键属性控制着是否将该元素视为一个可高亮显示的快捷键字符串。注意，在IE中它只能作为currentStyle（只读）和runtimeStyle（可读/可写）对象的属性。

示例 `document.getElementById("controlH").style.accelerator = true;`

值 布尔值：true | false。

默认值 false

750

azimuth, cue, cueAfter, cueBefore, elevation, pause, pauseAfter, pauseBefore, pitch, pitchRange, playDuring, richness, speak, speakHeader, speakNumeral, speakPunctuation, speechRate, stress, voiceFamily, volume

IE *n/a* NN *n/a* Moz *all* Saf *n/a* Op (见下文) DOM *2*

可读/可写

面向那些能够使用语音技术来朗读文档内容的浏览器，这一大组属性都来自于它们的CSS属性。尽管视力正常的用户也能够享用这些属性带来的便利，但Mozilla浏览器在默认情况下仍未包含这些功能。关于这些属性的详细信息，请参见第4章中的相关内容。

此外，Opera 9为其中的几个属性实现了它自己的版本，如opPhonemes、opVoicePitch、opVoicePitchRange、opVoiceRate、opVoiceStress及opVoiceVolume。使用这些属性可能须要浏览器提供独立的语音插件。

值 这些属性的值均为字符串。
默认值 无。

background

IE 4 NN n/a Moz all Saf all Op 7 DOM 2

可读/可写

它可以指明元素的background样式单属性。由于这是一种简写的属性，因此这个脚本属性通常由backgroundAttachment、backgroundColor、backgroundImage、backgroundPosition和backgroundRepeat的值组成，而各个值之间则使用空格进行分隔。在一个background值中可以包含一个或多个值，并且各个值的先后顺序也不受约束。须要注意的是，它只能作为IE中style和runtimeStyle对象的一个属性。

示例
```
document.getElementById("myDiv").style.background = "url(logo.gif) repeat-y";
```
值 由空格进行分隔的字符串值，对应于一个或多个独立的背景样式属性。
默认值 无。

backgroundAttachment

IE 4 NN n/a Moz all Saf 1.2 Op 7 DOM 2

可读/可写

设置图像贴附到元素上的具体方式。图像既可以固定在元素的可视区域内，也可以在文档滚动时随着元素一同运动。在滚动过程中，固定的贴附图片看起来就像是电影致谢内容后固定的背景图。

示例
```
document.getElementById("myDiv").style.backgroundAttachment = "fixed";
```
值 可以使用以下两个值之一：fixed | scroll。
默认值 scroll

backgroundColor

IE 4 NN n/a Moz all Saf all Op 7 DOM 2

可读/可写

此属性提供了元素的背景色。如果已经设置了backgroundImage，那么对应图像会覆盖背景色。而具有一定透明度的图像则可以显现出背景色。

示例
```
document.getElementById("highlighted").style.backgroundColor = "yellow";
```
值 任何有效的颜色设置或transparent。
默认值 transparent

backgroundImage

IE 4 NN n/a Moz all Saf all Op 7 DOM 2

可读/可写

为元素指定其背景图片的URL地址。如果已经设置了backgroundColor，那么背景图会覆盖背景色。而具有一定透明度的图像则可以显现出背景色。

示例
```
document.getElementById("navbar").style.backgroundImage = "url(images/navVisited.jpg)";
```
值 任意相对或绝对图像文件URL地址，请使用CSS的URL格式，即url(filePath)。
默认值 无。

backgroundPosition

IE 4 NN n/a Moz all Saf all Op 7 DOM 2

可读/可写

此属性指明背景图片的左、上位置与元素内容区域（包括填充）之间的相对关系。指定具体的位置时，既可以使用长度值（数值/单位或百分比），也可以按照top、right、bottom、left和center这几个常量的组

合进行设定。但是对于那些在页面中的重复出现的背景图片而言，这个属性将失去作用。须要注意的是，它只能作为IE中style和runtimeStyle对象的一个属性。

示例 `document.getElementById("div3").style.backgroundPosition = "20% 50%";`

752

值

一个字符串值，其中既可以仅包含一个数值，也可以包含两个由空格分隔的一对值。应该使用长度值（如30px 5px）、百分比（如50% 50%），以及含有明确意义的位置常量组合来清晰地表示这个属性值。

常量值对	百分比	常量值对	百分比
top left	0% 0%	center center	50% 50%
left top	0% 0%	right	100% 50%
top	50% 0%	right center	100% 50%
top center	50% 0%	center right	100% 50%
center top	50% 0%	bottom left	0% 100%
right top	100% 0%	left bottom	0% 100%
top right	100% 0%	bottom	50% 100%
left	0% 50%	bottom center	50% 100%
left center	0% 50%	center bottom	50% 100%
center left	0% 50%	bottom right	100% 100%
center	50% 50%	right bottom	100% 100%

从理论上来说，这些百分比值会被等价的常量值所替代。例如，“0%”意味着图像紧靠在元素块的左边缘或上边缘；“50%”表示图像在水平和竖直方向上均居中显示；而“100%”则会将图片置于右侧或底端。

默认值 0% 0%

backgroundPositionX, backgroundPositionY

IE *4* NN *n/a* Moz *n/a* Saf *1.3/2* Op *n/a* DOM *n/a*

可读/可写

它们表示背景图片的左、上位置与元素内容区域（包括填充）之间的相对关系。使用这两个属性就可以仅沿着一条轴向方向调整背景图片的位置，并同时保持另一个方向上的位置不变。

示例
```
document.getElementById("div3").style.backgroundPositionX = "20px";
document.getElementById("table2").style.backgroundPositionY = "10px;"
```

值 可以分别使用表示百分比块级元素所在空间的宽度和高度的百分比值来指定图像的位置。但最安全问题的做法还是使用像素值作为这两个属性的属性值。另外还要注意，此时只能识别top和left这两个常量。

默认值 0

753

backgroundRepeat

IE *4* NN *n/a* Moz *all* Saf *all* Op *7* DOM *2*

可读/可写

指明背景图像是否应该沿着轴线方向重复绘制。使用这种不断重复的背景图像就可以在水平和竖直方向上创造出条纹图案。

示例 `document.getElementById("div3").style.backgroundRepeat = "repeat-y";`

值 如果此属性被设置为no-repeat，那么只会在元素的backgroundPosition属性所指定的位置上出现一张图片。通常情况下会同时沿着两个方向重复绘制图像，但是将此属性设置为repeat-y后，就只会沿着竖直方向重复绘制，而repeat-x的效果也与之类似。如果要恢复到默认状态，那么请将属性值设置为“repeat”。

默认值 repeat

behavior

IE *5(Win)* NN *n/a* Moz *n/a* Saf *n/a* Op *n/a* DOM *n/a*

可读/可写

此属性可以控制是否为元素指定一个IE/Windows的外部行为。

示例
```
document.getElementById("div3").style.behavior = "url(#default#userData)";
```

值 CSS格式的URL值，它可以表示一个指向外部.htc文档的实际URL地址，也可以对应于一个在页面中加载ActiveX控件的object对象的ID值，还可以是一个内置的默认行为（此时格式为url(#default#*behaviorName*)）。

默认值 无。

blockDirection

IE *5(Win)* NN *n/a* Moz *n/a* Saf *n/a* Op *n/a* DOM *n/a*

只读

返回当前元素中脚本代码的书写方向。须要注意的是，它只能作为IE中currentStyle对象的一个属性。

示例
```
if (document.getElementById("myDIV").style.blockDirection = "rtl") {
  // 处理从左向右书写的文本内容
}
```

值 字符串常量：ltr | rtl。

默认值 ltr

border

IE *4* NN *n/a* Moz *all* Saf *all* Op *7* DOM *2*

754

可读/可写

在一条指令内，可以通过这个属性快速地读取或设定borderColor、borderStyle及borderWidth属性，从而控制元素四周的边框。如果要通过改变这个属性来达到影响元素边框显示效果的目的，必须首先指定一个边框样式，请参见borderStyle。如果缺少了边框样式，那么就会将它解释为无样式，即无边框。如果不希望4条边框完全相同，可以使用其他属性分别为它们单独设定宽度、样式和颜色。此外，只有那些明确定义在元素标签属性中的组合设置会反映在这个属性中，而通过这个属性设置的样式组合则可能会超出原有标签属性的范围。须要注意的是，在IE中它只能作为style和runtimeStyle对象的一个属性。

示例
```
document.getElementById("announce").style.border = "inset red 4px";
```

值 使用空格分隔各项属性值的字符串。对于borderStyle和borderWidth组合值，请参见本章中的对应属性。有关borderColor的详细信息，请参考第4章内与CSS颜色有关的部分。

默认值 无。

borderBottom, borderLeft, borderRight, borderTop

IE *4* NN *n/a* Moz *all* Saf *all* Op *7* DOM *2*

可读/可写

在一条指令内，可以通过这些属性快速地读取或设定borderColor、borderStyle以及borderWidth属性，从而控制元素的某条边框。如果要通过改变这些属性来达到影响元素边框显示效果的目的，必须首先指定一个边框样式，请参见borderStyle。如果缺少了边框样式，那么就会将它解释为无样式，即对应边上无边框。如果要将4条边设置为同样的风格，请参见border属性。此外，只有那些明确定义在元素标签属性中的组合设置会反映在这个属性中，而通过这个属性设置的样式组合则可能会超出原有标签属性的范围。需要注意的是，它只能作为IE中style和runtimeStyle对象的一个属性。

示例
```
document.getElementById("announce").style.borderBottom = "inset red 4px";
document.getElementById("announce").style.borderLeft = "solid #20ff00 2px";
```

```
document.getElementById("announce").style.borderRight = "double 3px";
document.getElementById("announce").style.borderTop = "outset red 8px";
```

值 使用空格分隔各项属性值的字符串。对于borderSideStyle和borderSideWidth组合值，请分别参考本章中的对应属性。有关borderSideColor的详细信息，请参见第4章内与CSS颜色有关的部分。

默认值 无。

755 borderBottomColor, borderLeftColor, borderRightColor, borderTopColor

IE 4 NN n/a Moz all Saf all Op 7 DOM 2

可读/可写

为元素某条边框提供颜色设置。须要注意的是，如果滥用这些属性，会将那些原本并不协调的颜色混合在一起。如果要通过一条指令来设定一组边框的颜色，请参见borderColor属性。

示例

```
document.getElementById("announce").style.borderBottomColor = "red";
document.getElementById("announce").style.borderLeftColor = "#20ff00";
document.getElementById("announce").style.borderRightColor = "rgb(100, 75, 0)";
document.getElementById("announce").style.borderTopColor = "rgb(90%, 0%, 25%)";
```

值 有关CSS颜色的详细信息，请参见第4章内与CSS颜色有关的部分。

默认值 无。

borderBottomStyle, borderLeftStyle, borderRightStyle, borderTopStyle

IE 4 NN n/a Moz all Saf all Op 7 DOM 2

可读/可写

为元素某条边框提供线条设置。border属性或borderStyle属性为4条边指定的样式也可以被这4个与边相关的属性所代替。如果要通过一条指令来设定一组边框的样式，请参见borderStyle属性。

示例

```
document.getElementById("announce").style.borderBottomStyle = "groove";
document.getElementById("announce").style.borderLeftStyle = "double";
document.getElementById("announce").style.borderRightStyle = "solid";
document.getElementById("announce").style.borderTopStyle = "inset";
```

值 此时的样式值是一组并不区分大小写的常量，它们分别对应于一种特定的边框线条显示方式。这些CSS样式常量包括：dashed、dotted、double、groove、hidden、inset、none、outset、ridge及solid。但并不是所有的浏览器均能识别CSS推荐标准中定义的所有常量值。请参见第4章中的border-style属性，以便了解可用的边框样式的完整信息。

默认值 无。

756 borderBottomWidth, borderLeftWidth, borderRightWidth, borderTopWidth

IE 4 NN n/a Moz all Saf all Op 7 DOM 2

可读/可写

为元素某条边框提供宽度设置。如果要通过一条指令来设定一组边框的宽度，请参见borderWidth属性。

示例

```
document.getElementById("announce").style.borderBottomWidth= "thin";
document.getElementById("announce").style.borderLeftWidth = "thick";
document.getElementById("announce").style.borderRightWidth = "2px";
document.getElementById("announce").style.borderTopWidth = "0.5em";
```

值 目前有3个可用的常量值，thin | medium | thick。它们决定了浏览器使用多少像素来绘制边框。使用一个长度值可以更为精确地控制边框宽度，请参见第4章中与CSS长度值有关的讨论。

默认值 medium

borderCollapse

IE 4 NN n/a Moz all Saf 1.3/2 Op 7 DOM 2

可读/可写

此属性可以设定table元素应该遵循哪个表格边框模型。

示例 document.getElementById("myTable").style.borderCollapse = "separate";

值 可以使用如下两个字符串常量，它们均不区分大小写：collapse | separate。

默认值 separate

borderColor

IE 4 NN n/a Moz all Saf all Op 7 DOM 2

可读/可写

这个属性可以快捷地为多条边框设定相同或不同的颜色。在这个属性值中，可以设定1到4个颜色值，每个颜色值间需要使用空格进行分隔。而颜色值的数量则决定了由哪些边接受指定的颜色值。

示例

```
document.getElementById("announce").style.borderColor = "red";
document.getElementById("announce").style.borderColor = "red green";
document.getElementById("announce").style.borderColor = "black rgb(100, 75, 0)
#c0c0c0";
document.getElementById("announce").style.borderColor = "yellow green blue
red";
```

值 这个属性接受由1到4个颜色值组成的字符串，这决定了将要为哪些边框设定颜色。请参见第4章中列出的border-color属性，以便了解颜色值数量会对这个属性产生哪些影响。

默认值 对象的color属性。

borderSpacing

IE 5 NN n/a Moz all Saf all Op 7 DOM 2

可读/可写

如果表格处于独立边框模式，那么这个属性可以控制表格单元格之间的间隔距离，它与table对象的cellSpacing比较类似。须要注意的是，IE 5/Macintosh不会响应对这个属性做出的任何修改。此外，它只能作为IE中style对象的一个属性。

示例 document.getElementById("myTable").style.borderSpacing= "12px";

值 字符串形式的CSS长度值，请参见第4章中与CSS长度有关的讨论。

默认值 无。

borderStyle

IE 4 NN n/a Moz all Saf all Op 7 DOM 2

可读/可写

这个属性可以快捷地为多条边框设定相同或不同的样式。在这个属性值中，可以设定1到4个样式值，每个不同值之间需要使用空格进行分隔。而样式值的数量则决定了由哪些边接受指定的样式。

示例

```
document.getElementById("announce").style.borderStyle = "solid";
document.getElementById("announce").style.borderStyle = "solid double";
document.getElementById("announce").style.borderStyle = "double groove groove
double";
```

值 此时的样式值是一组并不区分大小写的常量，它们分别对应于一种特定的边框线条显示方式。这些CSS样式常量包括：dashed、dotted、double、groove、hidden、inset、none、outset、ridge及solid。但并不是所有的浏览器均能识别CSS推荐标准中定义的所有常量值。请参见第4章中的border-style属性，以便了解可用的边框样式的完整信息。

这个属性接受由1到4个样式值组成的字符串，这决定了将要为哪些边框设定样式。请参见第4章中列出的border-style属性，以便了解样式值数量会对这个属性产生哪些影响。

默认值 none

758 borderWidth

IE 4 NN n/a Moz all Saf all Op 7 DOM 2

可读/可写

这个属性可以快捷地为多条边框设定相同或不同的宽度。在这个属性值中，可以设定1到4个宽度值，不同宽度值之间须要使用空格进行分隔。而宽度值的数量则决定了由哪些边接受指定的宽度。

示例 document.getElementById("founderQuote").style.borderWidth = "3px 5px";

值

目前有3个可用的常量值，thin | medium | thick。它们决定了浏览器使用多少像素来绘制边框。使用一个长度值可以更为精确地控制边框宽度，请参见第4章中有关CSS长度值的讨论。

这个属性接受由1到4个样式值组成的字符串，这决定了将要为哪些边框设定宽度。请参见第4章中列出的border-width属性，以便了解宽度值数量会对这个属性产生哪些影响。

默认值 medium

bottom

IE 5 NN n/a Moz all Saf all Op 7 DOM 2

可读/可写

对绝对定位的元素而言，它定义了元素盒（包括元素内容及底部填充、边框及边距）的下边缘相对于临近的最外层块级容器的下边缘的偏移量。当已定位元素使用根定位环境时，IE/Windows及Netscape 6会做出一些令人意想不到的反映。这两个浏览器不会将文档的底部作为相对边缘，而是使用浏览器窗口空间的底部作为相对边缘。这意味着元素底部的精确位置会随着用户浏览器窗口的尺寸发生改变。而IE 5/Macintosh会使用文档的底部作为相对边缘。这个差异使得在另一个已定位元素的内嵌元素中使用bottom属性变得更为实际。针对一个相对定位元素，元素通常会出现在容器内容中的某个位置，那么就根据内嵌位置的下边缘来决定偏移量。

如果在IE中要对这个属性值进行精确计算，请访问pixelBottom或posBottom样式属性，它们均会返回真实的数值。

示例 document.getElementById("blockD2").style.bottom = "35px";

值 一个由包含长度单位的数字值、百分比或auto组成的字符串。

默认值 auto

759 captionSide

IE 5(Mac) NN n/a Moz all Saf all Op 7 DOM 2

可读/可写

控制caption元素在表格框中的相对位置。

示例 document.getElementById("myTable").style.captionSide = "bottom";

值 以下字符串常量之一，不区分大小写：bottom | left | right | top。某些浏览器可能只支持bottom和top这两个值。

默认值 top

clear

IE 4 NN n/a Moz all Saf all Op 7 DOM 2

可读/可写

此属性可以决定是否允许对应元素与周边的浮动元素一起显示在同一个水平区域。在该元素附近往往存在另一个元素，而且其float样式属性值为left或right。如果要让当前元素与浮动块位于不同的水平区域内，

请将它们的clear属性设置为同一侧（left或right）。如果无法确定是否会出现重叠的现象，那么可以将clear属性设置为both。将元素的clear属性值设置为none之外的任意一个值之后，那么它就会显示在浮动元素下一行中的起始位置。

示例 `document.getElementById("myDiv").style.clear = "both";`
值 以下字符串常量之一，不区分大小写：`both | left | none | right`。
默认值 `none`

clip

IE 4 NN n/a Moz all Saf all Op 7 DOM 2

可读/可写

为可定位元素定义一个裁剪区域。裁剪区是元素层中可以显示元素内容的一个区域。在IE 4/Macintosh中，裁剪区可能无法正常工作。须要注意的是，它只能作为IE中style和runtimeStyle对象的一个属性。

示例 `document.getElementById("art2").style.clip = "rect(5px 100px 40px 0)";`
值 区分大小写的字符串，此时既可以使用常量“auto”，也可以使用CSS的clip属性来设定特定的图形（目前只能使用rect）及剪切区四条边相对于原始元素左上角的空间位置。请按照顺时针方向为裁剪矩形框的每条边指定长度值，即上、右、下、左。而CSS的长度设定请参考第4章中的相关内容。如果将clip属性设置为auto，那么裁剪区域会成为一个包含元素内容的矩形块。在IE中，这个宽度会扩展为临近的最外层容器的宽度，例如body元素的宽度。
默认值 无。

clipBottom, clipLeft, clipRight, clipTop

IE 5(Win) NN n/a Moz n/a Saf n/a Op n/a DOM n/a

只读

这4个属性分别返回可定位元素中剪切区域的一条边。它只能作为IE中currentStyle对象的一个属性。

示例 `var cl = document.getElementById("art2").style.clipLeft;`
值 不区分大小写的长度字符串，或者auto。而CSS的长度设定请参考第4章中的相关内容。
默认值 无。

color

IE 4 NN n/a Moz all Saf all Op 7 DOM 2

可读/可写

可用于设定元素的前景色（文本颜色）的样式单属性。对于某些图形元素而言，如表单控件等，color元素还可能会影响其元素边框或其他元素部件。但这些非正式的颜色设定往往只存在于特定的浏览器，它们在不同的浏览器上效果可能并不相同。

示例 `document.getElementById("specialDiv").style.color = "green";`
值 不区分字符大小写的CSS颜色规范，请参见第4章中的相关讨论。
默认值 `black`

content

IE 5(Mac) NN n/a Moz all Saf 1.3/2 Op 7 DOM 2

可读/可写

定义将要显示在元素之前或之后的额外内容及元素，这个属性将会与:before和:after伪类配合使用。尽管可以在IE 5/Macintosh和Netscape 6中使用这个属性，但此时它的值总是一个空字符串，而且改变这个属性值也不会对相关内容产生任何影响。

值 请参见第4章中关于CSS的content属性的讨论。
默认值 无。

counterIncrement, counterReset

IE *5(Mac)* NN *n/a* Moz *1.8* Saf *n/a* Op *7* DOM *2*

可读/可写

这两个属性是为在将来实现CSS规范的自动计数机制而保留的关键字。

761

值 请参见第4章中关于CSS的counterIncrement和counterReset属性的讨论。

默认值 无。

cssFloat

IE *5(Mac)* NN *n/a* Moz *all* Saf *all* Op *7* DOM *2*

可读/可写

控制一个元素的float属性，从而允许临近的文本内容能够包围住块级元素，如图像等。但在IE 5/Macintosh中改变这个值并不会产生任何实际效果。之所以为这个属性名称加上“css”前缀是为了防止与JavaScript的保留关键字float产生冲突。

示例 `document.getElementById("myDiv").style.cssFloat = "right";`

值 可以使用下面三个常量之一：left | right | none。

默认值 none

cssText

IE *4* NN *n/a* Moz *all* Saf *1.3/2* Op *7* DOM *2*

只读

返回一个字符串，表示已应用于元素的全部CSS样式单规则。如果规则中还包含简写的样式属性设定，如border等，那么浏览器会根据它们对实际值的判断返回修改后的样式单规则。如果将一个元素的style属性设置为“style="border: groove red 3px"”，那么IE/Windows会将其cssText属性描述为：

```
BORDER-RIGHT: red 3px groove; BORDER-TOP: red 3px groove;
BORDER-LEFT: red 3px groove; BORDER-BOTTOM: red 3px groove
```

IE/Macintosh则会描述为：

```
{BORDER-TOP: 3px groove red; BORDER-RIGHT: 3px groove red;
BORDER-BOTTOM: 3px groove red; BORDER-LEFT: 3px groove red}
```

而Mozilla浏览器的描述方式为：

```
border: 3px groove red;
```

从上面例子可以看出，要注意每种浏览器对值序列的不同处理方式。即便如此，依然可以按照任意顺序为这个属性指定简写的值设定。须要注意的是，它只能作为IE中style和runtimeStyle对象的一个属性。

示例 `document.getElementById("block3").style.cssText = "margin: 2px; font-size: 14pt";`

值 使用分号分隔多个样式属性的字符串。

默认值 无。

cue, cueAfter, cueBefore

请参见azimuth。

762

cursor

IE *4* NN *n/a* Moz *all* Saf *1.3/2* Op *7* DOM *2*

可读/可写

它可以指定屏幕指针位于元素之上时光标的具体样式。光标的精确外观主要依赖于操作系统。因此在部署一个改进的光标时，要首先了解不同类型的光标在浏览器和操作系统中的标准使用方式。用户通常都希望光标

在不同应用程序中都能够表达相同的含义。第4章中的图4-3给出了一份由IE浏览器提供图集，它绘出了每种光标常量对应的光标图形。

如果并未在全局范围内对光标进行设定，那么当指针移动到当前元素上方时这个属性就会影响光标的样式。

示例

```
document.getElementById("hotStuff").style.cursor = "pointer";
```

值

任意一个光标常量字符串，不同浏览器品牌和版本所支持的光标常量如下表所示：

光标名称	IE/Windows	IE/Mac	Mozilla	Safari	Opera
alias	n/a	n/a	n/a	n/a	n/a
all-scroll	6	n/a	1.8	n/a	n/a
auto	4	4	all	all	7
cell	n/a	n/a	1.8	n/a	n/a
col-resize	6	n/a	1.8	n/a	n/a
context-menu	n/a	n/a	1.8	n/a	n/a
copy	n/a	n/a	1.8	n/a	n/a
count-down	n/a	n/a	n/a	n/a	n/a
count-up	n/a	n/a	n/a	n/a	n/a
count-up-down	n/a	n/a	n/a	n/a	n/a
crosshair	4	4	all	all	7
default	4	4	all	all	7
e-resize	4	4	all	all	7
grab	n/a	n/a	<1	n/a	n/a
grabbing	n/a	n/a	<1	n/a	n/a
hand	4	4	n/a	all	7
help	4	4	all	all	7
move	4	4	all	all	7
n-resize	4	4	all	all	7
ne-resize	4	4	all	all	7
nesw-resize	n/a	n/a	1.8	n/a	n/a
no-drop	6	n/a	1.8	n/a	n/a
none	n/a	n/a	n/a	n/a	n/a
not-allowed	n/a	n/a	n/a	n/a	n/a
nw-resize	4	4	all	all	7
nwse-resize	n/a	n/a	1.8	n/a	n/a
pointer	4	4	all	all	7
progress	6	n/a	<1	n/a	9
row-resize	6	n/a	1.8	n/a	n/a
s-resize	4	4	all	all	7
se-resize	4	4	all	all	7
spinning	n/a	n/a	all	n/a	n/a
sw-resize	4	4	all	all	7
text	4	4	n/a	n/a	n/a
url(uri)	6	n/a	n/a	n/a	n/a

续表

光标名称	IE/Windows	IE/Mac	Mozilla	Safari	Opera
vertical-text	6	n/a	1.8	n/a	n/a
w-resize	4	4	all	all	7
wait	4	4	all	all	7

IE 6的外部URL设定还需要一个扩展名为*.cur*或*.ani*的光标文件的地址。

默认值 auto

direction

IE *5* NN *n/a* Moz *all* Saf *all* Op *7* DOM *2*

可读/可写

返回当前元素中脚本代码的书写方向。这个属性主要用于使用了多个脚本书写方向的元素，例如，将法语文本混杂在阿拉伯字符之中。

示例 `document.getElementById("term3").style.direction = "ltr";`

值 字符串常量：`ltr | rtl`。

默认值 `ltr`

display

IE *4* NN *n/a* Moz *all* Saf *all* Op *7* DOM *2*

可读/可写

控制着用于显示元素的CSS显示框类型。针对主体内容的最常用设定是指定元素的显示方式，即将它作为一个块级元素还是作为一个行内元素。如果将这个属性设定为none，就会隐藏当前元素，而其周围的内容则会充满该元素的原有空间。须要注意的是，某些显示框类型只适用于表格和列表。

示例 `document.getElementById("instructionDiv").style.display = "none";`

值

任意一个显示类型字符串，不同浏览器品牌和版本所支持的类型常量如下表所示：

显示类型	IE/Win	IE/Mac	NN	Mozilla	Safari	Opera	CSS
block	5	4	4	all	all	7	2
compact	n/a	n/a	n/a	n/a	n/a	n/a	<2.1
inline	5	4	4	all	all	7	2
inline-block	5.5	n/a	n/a	n/a	n/a	n/a	<2.1
inline-table	n/a	5	n/a	n/a	all	7	2
list-item	5	5	n/a	all	all	7	2
marker	n/a	n/a	n/a	n/a	n/a	n/a	<2.1
none	4	4	4	all	all	7	2
run-in	n/a	5	n/a	n/a	n/a	7	2
table	n/a	5	n/a	all	all	7	2
table-caption	n/a	5	n/a	all	all	7	2
table-cell	n/a	5	n/a	all	all	7	2
table-column-group	n/a	5	n/a	n/a	n/a	n/a	2
table-footer-group	5.5	5	n/a	all	all	7	2
table-header-group	5	5	n/a	all	all	7	2
table-row	n/a	5	n/a	all	all	7	2
table-row-group	n/a	5	n/a	n/a	n/a	n/a	2

默认值 由元素的具体类型决定。

elevation

请参见azimuth。

emptyCells

IE *5(Mac)* NN *n/a* Moz *all* Saf *1.3/2* Op *8* DOM *2*

可读/可写

如果表格按照默认状态显示彼此独立的单元格框，并且还为内嵌的td元素设定了边框，那么emptyCells样式属性就能够控制是否在空单元格周围显示边框。

示例 `document.getElementById("myTable").style.emptyCells = "hide";`

值 可以使用下面两个常量之一：hide | show。

默认值 show

filter

IE *4(Win)* NN *n/a* Moz *n/a* Saf *n/a* Op *n/a* DOM *n/a*

可读/可写

设定视觉、显示或混合滤镜，以用于显示或改变元素内容。视觉滤镜可以让元素内容产生旋转、发光、投影等诸多特效。在改变元素的可见性时就可以使用显示滤镜。显示滤镜的值决定了元素在从隐藏到显示或从显示到隐藏时使用哪种视觉效果。这些效果包括划入-划出、百叶窗及挡光板等。而混合滤镜则设置了状态转换的速度。尽管filter属性的确出现在IE/Macintosh中，但它并不能在该环境中正常运行。

示例 `document.getElementById("fancy").style.filter= "dropshadow()";`

值 每个filter属性都可能拥有多个相关的滤镜类型，而不同类型间均使用空格进行分隔。每个滤镜类型后都跟着一对圆括号，这样就可以为当前元素传递滤镜行为参数。每个参数通常都由一个名称/值对组成，并使用等号完成赋值过程。须要注意的是，从IE 5.5/Windows开始，微软就制定了一个崭新的滤镜语法规则。但到目前为止，仍然可以同时使用新、旧两种语法。请参见第4章中列出的filter样式单属性，以便了解滤镜设定与参数使用情况。

默认值 无。

font

IE *4* NN *n/a* Moz *all* Saf *all* Op *7* DOM *2*

可读/可写

通过这个属性可以在一条语句内快速地设定一个或多个与字体相关的属性，如fontFamily、fontSize、lineHeight、fontStyle、fontVariant及fontWeight等。可以用多个由空格分隔开来的值组成属性值，只要它们是有效的值类型，那么就会将它们应用于指定的字体属性。也可以从操作系统的下列默认字体中选择一个以加快这一设定的过程，如caption、icon、menu、message-box、small-caption、status-bar。

示例 `document.getElementById("subhead").style.font = "bolder small-caps 16pt";`

值 请参见相关的属性说明，以便了解各种字体属性的语法及使用示例。

默认值 无。

fontFamily

IE *4* NN *n/a* Moz *all* Saf *all* Op *7* DOM *2*

可读/可写

此属性为显示对象内容提供了一个按优先级排序的字体簇列表。在这个属性值中可以包含一个或多个使用逗号分隔的字体簇名称。如果一个字体簇名称由多个单词组成，那么应该将它放置在一对单引号之中。尽管它只能作为IE中style和runtimeStyle对象的一个属性，但是仍然可以在currentStyle中使用单独的字体属性。

示例

```
document.getElementById("subhead").style.fontFamily = "'Century Schoolbook', Times, serif";
```

值 任意数量的字体簇名称，请使用逗号进行分隔。使用时必须将包含多个单词的字体簇名称放置在一对单引号之中。可识别的常用字体簇名称包括：serif | sans-serif | cursive | fantasy | monospace。

默认值 由浏览器决定默认值。

fontSize

IE *4* NN *n/a* Moz *all* Saf *all* Op *7* DOM *2*

可读/可写

指明元素内的字体尺寸。可以通过多种方式设置字体尺寸。xx-small、x-small、small、medium、large、x-large、xx-large等常量定义了一种“绝对”尺寸。事实上，由于这些尺寸的参考点会随着浏览器和操作系统而发生变化，因此这种“绝对”尺寸也只能在一个个独立的浏览器和操作系统中奏效。但在同一个环境中，设置为large的元素的确比medium类型的元素要大一些，从这个角度来看，它们的确还是为网页开发者提供了一定的便利。

此外还存在另一组称之为相对尺寸的常量——lager、smaller。由于font-size样式属性继承自其父元素，因此会在父元素中应用这些相对尺寸以决定当前元素的字体尺寸。但最终还是由浏览器来决定这些字体尺寸的大小，而且很大因素上还取决于其父元素的具体字体尺寸。如果父级元素使用了一个绝对尺寸，如large，那么当其子元素的相对尺寸为larger时，意味着它在浏览器中显示的字体尺寸是x-large。当父元素的字体尺寸使用长度或百分比时，子元素字体尺寸的增加量并不确定。

如果要使用一个长度值作为fontsize的属性值，那么当长度单位为像素（px）或（em）时可以在不同的操作系统下获得更好的一致性，但单位为pt（point）时则没有这种效果。而使用em作为单位时，会根据父级元素的字体尺寸计算子元素的字体尺寸。最后，如果将fontSize设置为一个百分比，那么也会根据父级元素的字体尺寸估算子元素的字体大小。

示例
```
document.getElementById("teeny").style.fontSize = "x-small";
```

值 从以下分类中选取的一个字符串值，使用时不必区分大小写。对于一个绝对尺寸，可以使用一下常量之一：xx-small | x-small | small | medium | large | x-large | xx-large。对于一个相对尺寸，则可以使用large或smaller中的一个。对于长度值，请参考第4章中有关CSS长度值的内容。而使用百分比时，应使用“百分值%”的形式。

默认值 级元素的字体尺寸。

fontSizeAdjust

IE *5(Mac)* NN *n/a* Moz *all* Saf *n/a* Op *n/a* DOM *2*

可读/可写

此属性可以提供一个字体外观值，它通常是字体簇属性列表中位于首位的一个字体簇。这个属性会迫使其他备选字体簇在计算其字体尺寸时尽量接近主字体簇的设定。尽管这个属性的确是IE 5/Mac和Mozilla中style对象的成员之一，但无论是改变style元素属性还是使用脚本修改这个fontSizeAdjust，都不会改变字体显示的效果。

示例
```
document.getElementById("myDIV").style.fontSizeAdjust = "0.56";
```

值 数值或none。

默认值 none

fontStretch

IE *5(Mac)* NN *n/a* Moz *all* Saf *n/a* Op *n/a* DOM *2*

可读/可写

根据当前字体簇中的可用字符间距宽度，为元素设定字符间距。尽管这个属性的确是IE 5/Mac和Mozilla中style对象的成员之一，但无论是改变style元素属性还是使用脚本修改这个属性，都不会改变字体显示的效果。

示例 document.getElementById("myDIV").style.fontStretch= "ultra-condensed";
值 可以使用下面几个常量之一：normal | wider | narrower | ultra-condensed | extra-condensed | condensed | semi-condensed | semi-expanded | expanded | exTRa-expanded | ultra-expanded或none。
默认值 none

fontStyle

IE 4 NN n/a Moz all Saf all Op 7 DOM 2

可读/可写

从正常、斜体和倾斜这三者中选择一个作为元素内文本的字体样式。如果fontFamily中包含标注为斜体或倾斜的字体类型，fontStyle属性会首先从浏览器的系统中调用这些特殊的字体。但如果系统中并不存在指定的字体，那么就会倾斜普通的字体使它趋近于斜体字。针对指定的字体设置，会根据客户端计算机和打印机的品质生成一个电子字体类型并输出至打印机。个人计算机软件通常会包含位于“Style”文件夹之下的其他字体。其他类型的字体样式请参见fontVariant和fontWeight属性。

示例 document.getElementById("emphasis").style.fontStyle = "italic";
值 以下几个字符串常量值之一：normal | italic | oblique。
默认值 normal

fontVariant

IE 4 NN n/a Moz all Saf 1.3/2 Op 7 DOM 2

可读/可写

指定是否用小型大写字母的形式显示所有的字符，此时会将代码中的小写字母全部显示为较小的大写字母。如果在字体簇中正好包含一个小型大写字母的字体变量，那么浏览器应该自动使用该字体。然而浏览器很可能直接计算一个较小的字体尺寸用以设定小型大写字母的尺寸。IE 4则会根据父级元素的字体样式使用大写字母显示整个源代码中的内容。而后来的IE版本以及Mozilla、Safari和Opera等浏览器会使用两种不同的字体尺寸来显示大写字母。

示例 document.getElementById("emphasis").style.fontVariant = "small-caps";
值 下列字符串常量之一：normal | small-caps。
默认值 normal

fontWeight

IE 4 NN n/a Moz all Saf all Op 7 DOM 2

可读/可写

此属性用于设定元素字体的权重（粗细）。CSS提供的权重比例机制比大多数浏览器显示在屏幕上的内容要精细得多，但一旦将这些内容输出到打印机，那么这些经过精细调整的字体就会生效。字体的权重值从100递增到900，而每次递增都应以100为单位。因此fontWeight为100时表示最细的字体样式，而900则表示最粗的字体。normal等价于fontWeight为400，这是所有字体的默认粗细，而标准设定则为700。其他设定，如bolder和lighter，则表示根据父级元素的字体粗细指定一个相对权重。

示例 document.getElementById("hotStuff").style.fontWeight = "bold";
值 可使用如下的常量值之一：bold | bolder | lighter | normal | 100 | 200 | 300 | 400 | 500 | 600 | 700 | 800 | 900。
默认值 normal

height, width

IE 4 NN n/a Moz all Saf all Op 7 DOM 2

可读/可写

指明元素的高度和宽度及所用的度量单位。由于其属性值是包含度量单位的字符串，因此无法使用这两个属

性直接进行计算。但可以通过parseFloat()将这两个值转换为数值，也可以在IE中使用pixelHeight、pixelWidth、posHeight和posWidth来得到对应的值。此外，如果并未设置元素的position样式属性，那么即使修改了这些值可能也不会有任何的可视效果。

在IE 6和IE 7的标准兼容模式（此时document.compatType == "CSS1Compat"）中，只会将这些尺寸应用于元素的内容部分，而不包括其边框、填充或边距。例如，如果要将一个具有填充和边框的已定位元素的大小调整到一个精确的矩形尺寸，那么必须首先从其高度和宽度值中减去填充和边框的厚度，然后才能将整个元素设置为指定的大小。

769

示例 document.getElementById("viewArea").style.height = "450px";
值 由数字和长度度量单位或百分比组成的字符串。
默认值 无。

imeMode

IE *5(Win)* NN *n/a* Moz *n/a* Saf *n/a* Op *n/a* DOM *n/a*

可读/可写

在IE/Windows中控制是否让输入法编辑器出现在浏览器中，这种输入法编辑器可以支持中文、日文和韩文等多种语言。

示例 document.getElementById("nameEntry").style.imeMode = "active";
值 可以使用下面几个常量之一：active | auto | disabled | inactive。
默认值 auto

layoutFlow

IE *5(Win)* NN *n/a* Moz *n/a* Saf *n/a* Op *n/a* DOM *n/a*

可读/可写

主要在那些使用竖直排列的词句设置的语言中调用这个属性，它可以控制文本内容的前进方向。但从IE 5.5/Windows开始，它就已经被writingMode属性所取代。

值 下面3个字符串常量之一：horizontal | vertical-ideographic。
默认值 horizontal

layoutGrid

IE *5(Win)* NN *n/a* Moz *n/a* Saf *n/a* Op *n/a* DOM *n/a*

可读/可写

通过这个属性可以在一条语句内快速地设定一个或多个布局网格属性，如layoutGridChar、layoutGridLine、layoutGridMode和layoutGridType。目前主要在亚洲语言内容中使用这些属性。

示例 document.getElementById("subhead").style.layoutGrid = "2em strict";
值 请参见相关的属性说明，以便了解各种布局网格的相关属性的语法及使用示例。
默认值 无。

layoutGridChar

IE *5(Win)* NN *n/a* Moz *n/a* Saf *n/a* Op *n/a* DOM *n/a*

可读/可写

770

为块级元素指定亚洲语言字符网格的尺寸大小。

示例 document.getElementById("subhead").style.layoutGrid Char= "auto";
值 由CSS长度值组成的字符串，或者auto | none。
默认值 none

layoutGridLine

IE 5(Win) NN n/a Moz n/a Saf n/a Op n/a DOM n/a

可读/可写

为块级元素指定亚洲语言字符网格的行高。

示例 `document.getElementById("subhead").style.layoutGrid Line= "120%";`

值 由CSS长度值组成的字符串，或者auto | none。

默认值 none

layoutGridMode

IE 5(Win) NN n/a Moz n/a Saf n/a Op n/a DOM n/a

可读/可写

指定亚洲语言字符网格的样式应该是一维还是两维。

示例 `document.getElementById("subhead").style.layoutGrid Mode= "both";`

值 字符串常量：both | char（行内元素）| line（块级元素）| none。

默认值 both

layoutGridType

IE 5(Win) NN n/a Moz n/a Saf n/a Op n/a DOM n/a

可读/可写

控制布局网格对字符宽度变化的响应形式。

示例 `document.getElementById("subhead").style.layoutGrid Type = "strict";`

值 字符串常量：fixed | loose | strict。

默认值 loose

left

IE 4 NN n/a Moz all Saf all Op 7 DOM 2

可读/可写

对可定位的元素而言，它定义了元素盒（包括元素内容及左侧填充、边框及边距）的左边缘相对于临近的最外层块级容器的左边缘的偏移量。针对一个相对定位元素，元素通常会出现在容器内容中的某个位置，那么就会根据内嵌位置的左边缘来决定偏移量。

如果要对这个属性值进行精确计算，可以使用parseFloat()转换其返回值，在IE中还可以访问pixelLeft或posLeft样式属性，它们均会返回真实的数值。

示例 `document.getElementById("blockD2").style.left = "45px";`

值 一个由包含长度单位的数字值、百分比或auto组成的字符串。

默认值 auto

length

IE n/a NN n/a Moz all Saf all Op 7 DOM 2

只读

这是W3C DOM的CSSStyleDeclaration对象的一个正式属性，它可以返回当前对象所包含的独立的CSS属性的总数。

值 整数值。

默认值 随浏览器实现而发生改变。

letterSpacing

IE 4 NN n/a Moz all Saf all Op 7 DOM 2

可读/可写

指明元素内的字符间距。浏览器通常会根据字体设置和操作系统的字体显示方式来决定字符间距。如果在此

属性中使用负值，将缩小字符间距，此时一定要在不同的操作系统中对选中的字体进行充分的测试，以保证调整间距后文本的可读性。

示例 `document.body.style.letterSpacing = "1.1em";`

值 包括度量单位在内的长度值字符串或`normal`。如果根据现有的字体尺寸（如`em`或`ex`等）来决定度量单位才能得到最佳的显示效果。如果将此值设置为`normal`，那么就由浏览器来决定如何设置字符间距。

默认值 `normal`

lineBreak

IE *5(Win)* NN *n/a* Moz *n/a* Saf *n/a* Op *n/a* DOM *n/a*

可读/可写

控制日文文本中的换行规则。

示例 `document.body.style.lineBreak = "strict";`

值 字符串常量：`normal | strict`。

默认值 `normal`

lineHeight

IE *4* NN *n/a* Moz *all* Saf *all* Op *7* DOM *2*

可读/可写

指明行内容器盒（即容纳一行实体内容的方框）的高度。如果要了解不同类型的属性值在浏览器中的处理方式与继承特点，请参见第4章中的`line-height`属性。

示例 `document.getElementById("tight").style.lineHeight = "1.1em";`

值 长度值字符串或`normal`。

默认值 `normal`

listStyle

IE *4* NN *n/a* Moz *all* Saf *all* Op *7* DOM *2*

可读/可写

通过这个属性可以在一条指令语句内快速地设定3个列表样式属性。如果未在`listStyle`中明确设定某个列表样式属性，那么就会使用其默认值来设定该属性。针对`ol`与`ul`元素中列表项前自动显示的标记符号，这几个列表样式属性定义了它们的显示特征。尽管它只能作为IE中`style`和`runtimeStyle`对象的一个属性，但是仍然可以在`currentStyle`中使用单独的字体属性。

示例 `document.getElementById("itemList").style.listStyle = "square outside none";`

值 对于`listStyleType`、`listStylePosition`和`listStyleImage`这三个属性的可用属性值，请分别参考对应的属性项说明。可以在这个listStyle属性设定中以任意顺序包含1到3个值来设定对应的列表样式。

默认值 无。

listStyleImage

IE *4* NN *n/a* Moz *all* Saf *all* Op *7* DOM *2*

可读/可写

提供一个图像的URL地址，并以该图像作为列表项之前的标记符号。由于这个属性具有继承性，因此为列表中的单独一项进行设置（包括设置为`none`）之后，它都会代替其父级元素对象的同名属性。

示例

```
document.getElementById("itemList").style.listStyleImage ="url(images/3DBullet.gif)";
```

值 请使用`none`来替换父级元素中指定的图像。否则请按照CSS的URL格式提供一个图像文件的完整或相对URL地址，但要保证浏览器可以读取该文件的MIME类型。

默认值 none

listStylePosition

IE 4 NN n/a Moz all Saf all Op 7 DOM 2

可读/可写

指定标记符号是位于列表项容器框之内还是之外。当listStylePosition设置为inside并且列表项中的内容为文本时，标记符号会成为文本块中的一部分。在这种情况下，列表项的对齐（缩进）方式与正常情况完全一样，只是不存在伸出在外的标记符号。

示例 document.getElementById("itemList").style.listStylePosition = "inside";

值 下面两个字符串常量之一：inside | outside。

默认值 outside

listStyleType

IE 4 NN n/a Moz all Saf all Op 7 DOM 2

可读/可写

指定每一个列表项前显示的标记符号的具体类型。只有当listStyleImage为none或并未指定属性值时才会采用这个属性。这个属性的可用常量值主要分为两大类。其中一类主要用于ul元素，它可以显示实心圆、空心圆或实心正方形。而另一类则用于ol元素，此时可以使用阿拉伯数字、罗马数字（大小写均可）或字母表中的字符（大小写均可）来标识列表项。当浏览器和操作系统能够支持其他语言时，在序列中甚至还可以使用这些语言的字符。

示例 document.getElementById("itemList").style.listStyleType = "circle";

值

可以使用下列与列表容器有关的字符串常量值之一。对ul元素，请使用circle | disc | square。而在ol元素中，则可以使用decimal | decimal-leading-zero | lower-roman | upper-roman | lower-greek | lower-alpha | lower-latin | upper-alpha | upper-latin | hebrew | armenian | georgian | cjk-ideographic | hiragana | katakana | hiragana-iroha | katakana-iroha。下表中列出了ol元素中的排序效果。

类型	示例	类型	示例
decimal	1, 2, 3, ...	lower-roman	i, ii, iii, ...
decimal-leading-zero	01, 02, 03, ...	upper-alpha	A, B, C, ...
lower-alpha	a, b, c, ...	upper-roman	I, II, III, ...
lower-greek	α, β, γ, ...		

默认值 ul中为disc，而ol中为decimal。

margin

IE 4 NN n/a Moz all Saf all Op 7 DOM 2

可读/可写

通过这个属性可以在一条指令语句中快速地为元素的4条边设定边距宽度。边距实际上就是元素边缘向外扩展的一段空间，从而在该元素和相邻或内嵌元素之间提供一块额外的空白区域。在这个属性值中，可以设定1到4个边距值，每个边距值间须要使用空格进行分隔。而样式值的数量则决定了由哪些边接受指定的边距样式。

示例 document.getElementById("logoWrapper").style.margin = "5px 8px";

值 这个属性接受由1到4个样式值组成的字符串，这也决定了将要为哪几个边距设定样式。请参见第4章中列出的margin属性，以便了解字符串中的值数量会对这个属性产生哪些影响。可以使用长度、临近最外层容器尺寸的百分比或常量auto来设定这个边距值。

默认值 0

marginBottom, marginLeft, marginRight, marginTop

IE *4* NN *n/a* Moz *all* Saf *all* Op *7* DOM *2*

可读/可写

这4个属性设定了元素某条边的边距宽度。边距实际上就是沿着元素的边缘向外扩展的一块空间，但并不会将它视为元素高度或宽度的一部分。

示例

```
document.getElementById("logoWrapper").style.marginTop = "5px";
document.getElementById("navPanel").style.marginLeft = "10%";
```

值 可以使用长度、临近的最外层容器尺寸的百分比或常量auto作为边距值的宽度。

默认值 0

775

markerOffset

IE *n/a* NN *n/a* Moz *all* Saf *n/a* Op *7* DOM *n/a*

可读/可写

控制列表项标记符号与容纳文本内容的容器盒之间的距离。尽管可以在Mozilla和Opera中使用这个属性，但它的值总是一个空字符串，而且改变这个属性值也不会对页面中的内容产生任何影响。此外，CSS 2.1已经删除了与之对应的CSS属性。

值 包括度量单位在内的长度值字符串或auto。

默认值 无。

marks

IE *5(Mac)* NN *n/a* Moz *all* Saf *n/a* Op *7* DOM *2*

可读/可写

为@page规则设置剪切标记。尽管可以在IE 5/Macintosh、Mozilla和Opera中使用这个属性，但它的值总是一个空字符串，而且改变这个属性值也不会对页面中的内容产生任何影响。

值 以下字符串常量之一，不区分大小写：crop | cross | none。

默认值 none

maxHeight, maxWidth, minHeight, minWidth

IE (见下文) NN *n/a* Moz *all* Saf *all* Op *7* DOM *2*

可读/可写

为一个元素的高度和宽度指定限制范围。定义了这几个属性后，元素既不能超出“max”属性指定的空间尺寸，也不能小于“min”属性所指定的数值。值得注意的是，IE 6仅支持minHeight属性，而且只能用于tr、th和td元素。然而IE 7在CSS兼容模式下则完全正确地实现了这几个属性。

值 字符串形式的CSS长度值，请参见第4章。

默认值 无。

MozBorderRadius

IE *n/a* NN *n/a* Moz *all* Saf *n/a* Op *n/a* DOM *n/a*

可读/可写

通过这个属性可以快速地为一个或多个边框设定转角半径。

值 可以使用1到4个字符串形式的长度值。关于具体的属性值设定细节，请参见第4章中的-moz-border-radius。

默认值 0

776

MozBorderRadiusBottomleft, MozBorderRadiusBottomright, MozBorderRadiusTopleft, MozBorderRadiusTopright

IE *n/a* NN *n/a* Moz *all* Saf *n/a* Op *n/a* DOM *n/a*

可读/可写

以上每个属性均控制着一个边框转角的半径值。

值 字符串形式的长度值。关于具体的属性值设定细节，请参见第4章中的-moz-border-radius。
默认值 0

MozOpacity

IE *n/a* NN *n/a* Moz *all* Saf *n/a* Op *n/a* DOM *n/a*

可读/可写

定义元素不透明的程度。此属性值越小，则元素的透明度越高。这个Mozilla属性等价于微软的不透明属性，但在新版本的浏览器中，它已经被opacity属性所取代。

示例 `document.getElementById("menuWrapper").style.MozOpacity = "40%";`
值 介于0~1之间的某个数值。
默认值 1（完全不透明）。

opacity

IE *n/a* NN *n/a* Moz *1.7.2* Saf *1.2* Op *9* DOM *n/a*

可读/可写

定义元素不透明的程度。此属性值越小，则元素的透明度越高。

示例 `document.getElementById("menuWrapper").style.MozOpacity = "40%";`
值 介于0~1之间的某个数值。
默认值 1（完全不透明）。

orphans, widows

IE *5(Mac)* NN *n/a* Moz *all* Saf *n/a* Op *7* DOM *2*

可读/可写

对于分布在多个页面容器盒中的块级元素内容而言，在两个属性分别指定了元素在页面底部（orphans）或下一个页面顶部（widows）的最少行数。尽管这两个属性的确是IE 5/Mac、Mozilla和Opera中style对象的成员之一，但无论是改变style元素属性还是使用脚本修改这个属性，都不会改变打印输出的效果。

示例 `document.getElementById("sec23").style.orphans = "3";`
值 字符串形式的整数值。
默认值 无。

outline

IE *5(Mac)* NN *n/a* Moz *1.8.1* Saf *1.2* Op *7* DOM *2*

可读/可写

使用这个属性可以在一条指令语句内快速地获取或设定元素的轮廓属性，如outlineColor、outlineStyle和outlineWidth。只有为这个属性指定一个轮廓样式（请参见outlineStyle）才能影响元素的外观。轮廓与边框比较类似，但它并不占据元素的内容空间也不会影响元素的尺寸大小。

示例 `document.getElementById("announce").style.outline = "solid blue 4px";`
值 使用空格分隔各项属性值的字符串。对于outlineStyle和outlineWidth组合值，请参见本章中的对应属性。有关outlineColor的详细信息，请参见第4章内与CSS颜色有关部分。
默认值 无。

outlineColor

IE *5(Mac)* NN *n/a* Moz *1.8.1* Saf *1.2* Op *7* DOM *2*

可读/可写

控制元素轮廓的颜色。

示例 `document.getElementById("announce").style.outlineColor = "rgb(100, 75, 0)";`
值 可以使用CSS颜色值或常量invert。有关CSS颜色的详细信息，请参见第4章内与CSS颜色有关部分。

默认值 invert

outlineOffset

IE *5(Mac)* NN *n/a* Moz *1.8.1* Saf *1.2* Op *n/a* DOM *n/a*

可读/可写

控制外部轮廓与元素边框间的距离，在4个方向上的间距一般都完全相等。

示例 `document.getElementById("announce").style.outlineOffset = "5px";`

值 一个CSS长度值。此时只用一个值就可以控制轮廓的所有边。

默认值 0

778

outlineStyle

IE *5(Mac)* NN *n/a* Moz *1.8.1* Saf *1.2* Op *7* DOM *2*

可读/可写

控制元素轮廓的线形。

示例 `document.getElementById("announce").style.outlineStyle = "solid";`

值 此时的样式值是一组并不区分大小写的常量，它们分别对应于一种特定的轮廓及边框线条显示方式。这些CSS样式常量包括：`dashed | dotted | double | groove | hidden | inset | none | outset | ridge`及`solid`。

默认值 none

outlineWidth

IE *5(Mac)* NN *n/a* Moz *1.8.1* Saf *1.2* Op *7* DOM *2*

可读/可写

控制轮廓线条的粗细。

示例 `document.getElementById("announce").style.outlineWidth = "2px";`

值 目前有3个可用的常量值，`thin | medium | thick`。它们决定了浏览器使用多少像素来绘制边框。使用一个长度值可以更为精确地控制边框宽度，请参见第4章中有关CSS长度值的讨论。

默认值 medium

overflow

IE *4* NN *n/a* Moz *all* Saf *all* Op *7* DOM *2*

可读/可写

如果使用样式单规则确定了元素的边框，那么这个属性可以指明元素应该如何处理那些溢出边框的内容。关于`overflow`样式单属性的详细讨论请参见第4章。

示例 `document.getElementById("myDiv").style.overflow = "scroll";`

值 任意一个字符串常量：`auto | hidden | scroll | visible`。

默认值 visible

overflowX, overflowY

IE *5(Win)* NN *n/a* Moz *1.8* Saf *1.2* Op *7* DOM *n/a*

可读/可写

如果使用样式单规则确定了元素的边框，那么这个属性可以指明元素应该如何处理那些溢出水平边框（`overflowX`）或竖直边框（`overflowY`）的内容。

示例 `document.getElementById("myDiv").style.overflow X= "scroll";`

779

值 任意一个字符串常量：`auto | hidden | scroll | visible`。

默认值 visible

padding

IE 4 NN n/a Moz all Saf all Op 7 DOM 2

可读/可写

通过这个属性可以在一条指令语句中快速地为元素的4条边设定填充的宽度。填充实际上是围绕在元素内容容器盒四周并一直延伸到边框的一块空间区域，但它并不包括为元素指定的任何边框内容。填充区域会自动获取所在元素的背景图片或背景颜色。在向一个元素增加填充时，只会增加元素的矩形框大小而不会影响元素内容块的尺寸。在这个属性值中，可以设定1到4个填充值，每个填充值间则须要使用空格进行分隔。而样式值的数量就决定了由哪些边接受指定的样式。

示例 `document.getElementById("logoWrapper").style.padding = "3px 5px";`

值 这个属性接受由1到4个样式值组成的字符串，这决定了将要为哪几条边进行填充操作。请参见第4章中列出的padding属性，以便了解字符串中的值数量会对这个属性产生哪些影响。可以使用长度、临近最外层容器尺寸的百分比或常量auto来设定填充的宽度。

默认值 0

paddingBottom, paddingLeft, paddingRight, paddingTop

IE 4 NN n/a Moz all Saf all Op 7 DOM 2

可读/可写

这4个属性设定了元素某条边的填充宽度。填充实际上是位于元素边框与内容容器盒之间的一部分空间区域。但是并不会将填充作为元素宽度或高度的一部分。

示例

```
document.getElementById("logoWrapper").style.paddingTop = "3px";
document.getElementById("navPanel").style.paddingLeft = "10%";
```

值 可以使用长度、临近最外层容器尺寸的百分比或常量auto作为填充的宽度。

默认值 0

page

IE 5(Mac) NN n/a Moz all Saf n/a Op 7 DOM 2

可读/可写

如果一个已存在的@page规则拥有一个标识符，如“@page figures {size: landscape}”，那么为了将这个规则应用于当前块级元素，就须要使用此属性指出该规则的名称。尽管这个属性的确是IE 5/Mac、Mozilla和Opera中style对象的成员之一，但无论是改变style元素属性还是使用脚本修改这个属性，都不会改变打印输出的效果。

值 一个字符串标识符。

默认值 无。

pageBreakAfter, pageBreakBefore

IE 4 NN n/a Moz 1.0.1 Saf 1.3/2 Op 7 DOM 2

可读/可写

将文档发送至打印机时，这个属性定义了文档内容如何处理元素四周的分页符。和字处理程序中的效果类似，分页符并不会出现在可视化浏览器中，而连续的大量内容则会使得屏幕中出现翻页和滚动效果。与分页问题有关的详细讨论请参见第4章中的page-break-after和page-break-before样式属性。

示例

```
document.getElementById("hardBR").style.pageBreakAfter = "always";
document.getElementById("navPanel").style.paddingLeft = "10%";
```

值 所有支持这两个属性的浏览器都能识别下面这4个常量字符串值：always、auto、left、right。另外，IE/Windows中还可以将空字符串作为属性值，此时它与W3C的CSS常量avoid有相同的作用。

默认值 auto

pageBreakInside

IE *5(Mac)* NN *n/a* Moz *all* Saf *n/a* Op *7* DOM *2*

可读/可写

此属性决定了是否能够将元素分割到多个打印页面之内。尽管这个属性的确是IE 5/Mac、Mozilla和Opera中style对象的成员之一，但无论是改变style元素属性还是使用脚本修改这个属性，都不会改变打印输出的效果。

值 下面两个字符串常量之一：auto | avoid。

默认值 auto

parentRule

IE *n/a* NN *n/a* Moz *all* Saf *1.2* Op *7* DOM *2*

只读

这是W3C DOM的CSSStyleDeclaration对象的一个正式属性，它会返回当前样式单的父对象——styleSheet对象的引用。只有在嵌套的情况下才会出现这个属性，例如内嵌在一个@media块中的样式规则。

值 styleSheet对象的引用。

默认值 无。

pause, pauseAfter, pauseBefore, pitch, pitchRange

请参见azimuth。

pixelBottom, pixelLeft, pixelRight, pixelTop

IE *4* NN *n/a* Moz *n/a* Saf *all* Op *7* DOM n/a

可读/可写

对可定位的元素而言，这些属性定义了元素盒（包括元素内容及左侧填充、边框，以及边距）的边缘相对于临近的最外层块级容器的边缘的偏移量，单位为像素。针对一个相对定位元素，元素通常会出现在容器内容中的某个位置，那么就根据内嵌位置的边缘来决定偏移量。由于bottom、left、right和top属性的值是包含单位名称的字符串，这使得它们并不便于计算，因此可以在计算中使用“pixel”系列属性来替代这些属性。须要注意的是，在IE中它只能作为style和runtimeStyle对象的一个属性。

示例
```
document.getElementById("myDIV").style.pixelLeft++;
```

值 整数值。

默认值 无。

pixelHeight, pixelWidth

IE *4* NN *n/a* Moz *n/a* Saf *all* Op *7* DOM *n/a*

可读/可写

这两个属性可以分别指定元素的高度和宽度，单位为像素。由于height和width属性的值是包含单位名称的字符串，这使得它们并不便于计算，因此可以在计算中使用“pixel”系列属性来替代这两个属性。另外，如果并未设置元素的position样式属性，那么即使修改了这些值可能也不会有任何的可视效果。须要注意的是，在IE中它只能作为style和runtimeStyle对象的一个属性。

示例
```
var midWidth = document.getElementById("myDIV").style.pixelWidth/2;
```

值 整数值。

默认值 无。

playDuring

请参见azimuth。

posBottom, posLeft, posRight, posTop

IE 4 NN n/a Moz n/a Saf n/a Op 7 DOM n/a

可读/可写

对可定位的元素而言，这些属性定义了元素盒（包括元素内容及左侧填充、边框，以及边距）的边缘相对于临近的最外层块级容器的边缘的偏移量，单位为像素。针对一个相对定位元素，元素通常会出现在容器内容中的某个位置，那么就根据内嵌位置的边缘来决定偏移量。更为重要的是，这些属性的值均为数字，而且它们的度量单位与CSS的bottom、left、right或top属性的单位相同。由于bottom、left、right和top属性的值是包含单位名称的字符串，这使得它们并不便于计算，因此可以在计算中使用“pos”系列属性来替代这些属性。与此同时，浏览器会在指定的度量单位下完成所有的计算。与pixelBottom、pixelLeft、pixelRight和pixelTop相比，它们的不同在于“pixel”系列属性的值均是单位为像素的整数值。须要注意的是，在IE中它只能作为style和runtimeStyle对象的一个属性。

示例

```
document.getElementById("myDIV").style.posLeft =
  document.getElementById("myDIV").style.posLeft + 1.5;
```

值 浮点数。

默认值 无。

posHeight, posWidth

IE 4 NN n/a Moz n/a Saf n/a Op 7 DOM n/a

可读/可写

这两个属性可以分别指定元素的高度和宽度，其单位由CSS中与定位相关的属性决定。由于height和width属性的值是包含单位名称的字符串，使得它们并不便于计算，因此可以在计算中使用“pos”系列属性来替代这两个属性。与此同时，浏览器会在指定的度量单位下完成所有的计算。与pixelHeight、pixelWidth相比，它们的不同在于“pixel”系列属性的值均是单位为像素的整数值。须要注意的是，它只能作为IE中style和runtimeStyle对象的一个属性。

示例 `document.getElementById("myDIV").style.posWidth = 10.5;`

值 浮点数。

默认值 无。

position

IE 4 NN n/a Moz all Saf all Op 7 DOM 2

只读

对于可定位元素，这个属性返回为该元素的position样式单属性的属性值。尽管这个属性是可读可写的，但并不能通过它一个已定位的元素转变为一个静态元素，反之亦然。

示例 `var posType = document.getElementById("myDIV").style.position;`

值 字符串常量：absolute | fixed | relative | static。须要注意的是，Windows下的IE浏览器一直到版本6都不支持属性值“fixed”。

默认值 无。

quotes

IE 5(Mac) NN n/a Moz all Saf n/a Op 7 DOM 2

可读/可写

此属性可以指定用于表示引用标记的多对字符。关于实际使用中的注意事项请参见第4章中的quotes属性。如果在为这个属性写入新值时无法成功，请不要感到诧异。

值 由2个或4个引用字符串组成的字符串值。第一对字符串提供了一级引用的字符，而第二对字符串则为嵌套引用提供了所需的字符。

783

默认值 无。

richness

请参见azimuth。

right

IE *5* NN *n/a* Moz *all* Saf *all* Op *7* DOM *2*

可读/可写

对绝对定位的元素而言，它定义了元素盒（包括元素内容及底部填充、边框以及边距）的右边缘相对于临近的最外层块级容器的右边缘的偏移量。

如果在IE中要对这个属性值进行精确计算，请获取pixelRight或posRight样式属性，它们均会返回真实的数值。

示例 `document.getElementById("blockD2").style.right = "25px";`

值 一个由包含长度单位的数字值、百分比或auto组成的字符串。

默认值 `auto`

rubyAlign

IE *5* NN *n/a* Moz *n/a* Saf *n/a* Op *n/a* DOM *n/a*

可读/可写

此属性可以控制ruby元素中相关内容的对齐方式。须要注意的是，修改这个属性只能影响Windows系统中的IE浏览器。与ruby相关的样式都定义在CSS3中。

示例 `document.getElementById("myRuby").style.rubyAlign = "center";`

值 以下字符串常量之一，不区分大小写：`auto | center | distribute-letter | distribute-space | left | line-edge | right`。

默认值 `auto`

rubyOverhang

IE *5* NN *n/a* Moz *n/a* Saf *n/a* Op *n/a* DOM *n/a*

可读/可写

此属性可以控制ruby元素中相关内容的文字突出方式。须要注意的是，修改这个属性只能影响Windows系统中的IE浏览器。与ruby相关的样式都定义在CSS3中。

示例 `document.getElementById("myRuby").style.rubyOverhang="whitespace";`

值 以下字符串常量之一，不区分大小写：`auto | none | whitespace`。

默认值 `auto`

784

rubyPosition

IE *5* NN *n/a* Moz *n/a* Saf *n/a* Op *n/a* DOM *n/a*

可读/可写

此属性控制着ruby文本（rt元素）与相关的ruby基础文本显示在同一行中还是位于这些文本之上。须要注意的是，修改这个属性只能影响Windows系统中的IE浏览器。与ruby相关的样式都定义在CSS3中。

示例 `document.getElementById("myRuby").style.rubyPosition = "inline";`

值 以下字符串常量之一，不区分大小写：`above | inline`。

默认值 `above`

scrollbar3dLightColor, scrollbarArrowColor, scrollbarBaseColor, scrollbarDarkShadowColor, scrollbarFaceColor, scrollbarHighlightColor, scrollbarShadowColor, scrollbarTrackColor

IE *5.5* NN *n/a* Moz *n/a* Saf *n/a* Op *n/a* DOM *n/a*

可读/可写

它们控制着不同滚动条用户界面元素中特定组件的颜色，可以分别在applet、body、div、embed、object或textarea元素中使用这些属性。关于每个属性能控制哪个组件请参见第4章的详细说明。

示例 `document.getElementById("comments").style.scrollbarArrowColor = "rgb(100, 75, 0");`

值 不区分字符大小写的CSS颜色规范，请参见第4章中的相关讨论。

默认值 无。

size

IE *n/a* NN *n/a* Moz *all* Saf *n/a* Op *7* DOM *2*

可读/可写

在一个由@page规则定义的页面环境中，这个属性控制着页面尺寸或定向。尽管可以在Mozilla和Opera中使用这个属性，但它的值总是一个空字符串，而且这个属性对页面环境也没有任何影响。

值 字符串形式的CSS长度值或一个不须要区分大写的字符串：auto、andscape、portrait。对于长度值，如果只设定了一个值，那么会将它同时应用于高度和宽度，如果使用空格分隔了两个值，那么会将它们分别应用于宽度和高度。

默认值 auto

speak, speakHeader, speakNumeral, speakPunctuation, speechRate, stress

请参见azimuth。

styleFloat

IE *4* NN *n/a* Moz *n/a* Saf *n/a* Op *n/a* DOM *n/a*

可读/可写

指定元素应该排列在容器盒的哪一条边上，然后其他内容就可以环绕在元素周围。将这个属性设置为none后，会根据其源代码顺序显示元素，而且在元素所在的水平区域内至少会出现有一行环绕文本内容。有关此属性的详细信息请参见第4章中的float样式属性。另外，IE 5/Macintosh还提供了一个与之完全相同的属性——cssFloat。

示例 `document.getElementById("myDIV").style.styleFloat = "right";`

值 3个字符串常量之一：none | left | right。

默认值 无。

tableLayout

IE *5* NN *n/a* Moz *1.0.1* Saf *all* Op *7* DOM *2*

可读/可写

这个属性可以作为加载表格时的一个开关项，浏览器会根据其属性值按两种不同方式来绘制表格，第1种方式是按照第一行设定的列宽度绘制整个表格，第2种则是在表格数据加载完成后再根据单元格中的实际内容计算最优的列宽度。一旦表格已完全显示，那么改变这个属性也不会有任何的作用。

示例 `document.getElementById("myTable").style.tableLayout = "fixed";`

值 下列字符串常量之一：auto | fixed。

默认值 auto

textAlign

IE 4 NN n/a Moz all Saf all Op 7 DOM 2

可读/可写

决定了元素盒中所有文本的水平对齐方式。

示例 `document.getElementById("myDIV").style.textAlign = "right";`
值 下列字符串常量之一：`center | justify | left | right`。
默认值 由浏览器的默认语言决定。

textAlignLast

IE 5.5 NN n/a Moz n/a Saf n/a Op n/a DOM n/a

可读/可写

决定了元素盒中最后一行文本的水平对齐方式。如果在IE 5.5中使用了其他的专有文本对齐样式属性，那么使用这个样式属性可能有助于获得预期的外观。

示例 `document.getElementById("myDIV").style.textAlignLast = "justify";`
值 下列字符串常量之一：`auto | center | justify | left | right`。
默认值 `auto`

786

textAutospace

IE 5(Win) NN n/a Moz n/a Saf n/a Op n/a DOM n/a

可读/可写

控制表意（象形）字符和非表意字符之间的间隔，常用于亚洲语言。

示例 `document.getElementById("myDIV").style.textAutospace = "ideograph-numeric";`
值 下列字符串常量之一：`ideograph-alpha | ideograph-numeric | ideograph-parenthesis | ideograph-space | none`。
默认值 `none`

textDecoration

IE 4 NN n/a Moz all Saf all Op 7 DOM 2

可读/可写

为元素文本内容指定其他的附加修饰样式，如下划线、删除线、上划线及闪烁（可用于Navigator及CSS）。浏览器在内部使用这个样式属性，默认情况下会将下划线用于a元素，并将删除线用于strike元素，因此默认值与具体的元素类型有关。在这个属性值中可以设定多个样式值，每个不同值之间则须要使用空格进行分隔。尽管浏览器也接受blink，但它们并不会因此而让文本闪烁起来（IE 7除外）。这种文本修饰拥有不同寻常的父-子关系。虽然并不存在属性值继承，但在大多数情况下，文本的修饰效果会传递给元素的内嵌项。因此除非设定了其他替代样式，否则使用下划线的p元素会为内嵌的b元素的内容也加上下划线。此外，IE浏览器为每种修饰类型都增加了一个布尔属性。

示例 `document.getElementById("emphasis").style.textDecoration = "underline";`
值 除none外，可以使用下面4个常量字符串：`blink | line-through | overline | underline`。同时使用多个值时，请在列表内用空格进行分隔。
默认值 由元素和内部样式单决定。

textDecorationBlink, textDecorationLineThrough, textDecorationNone, textDecorationOverline, textDecorationUnderline

IE 4 NN n/a Moz n/a Saf n/a Op n/a DOM n/a

787

可读/可写

这几个属性分别表示是否在元素上开启特定的文本修饰功能。它们的对应值都可以赋予CSS样式属性text-decoration。由于IE（IE 7除外）并不会实现文本闪烁，所以它忽略了textdecorationBlink属性。如果将

textdecorationNone设置为true，则会将这几个相关的修饰属性都设置为false。须要注意的是，在IE 4/Macintosh中设定这些属性并不会给元素的文本内容带来任何影响。此外，使用textDecoration来代替这些属性是一种良好的习惯。

示例 document.getElementById("emphasis").style.textDecorationLineThrough = "true";

值 布尔值：true | false。

默认值 false

textIndent

IE 4 NN n/a Moz all Saf all Op 7 DOM 2

可读/可写

此属性指定了文本块中第1行的缩进量。它也只能对第1行产生影响。使用负值则会突出第1行，此时要保证突出的文本不会超出浏览器窗口或框架的左边缘。

示例 document.getElementById("firstGraph").style.textIndent = "0.5em";

值 字符串形式的CSS长度值（正负均可），请参见第4章。

默认值 0px

textJustify

IE 5 NN n/a Moz n/a Saf n/a Op n/a DOM n/a

可读/可写

如果将一个块级元素的text-align CSS属性或textAlign样式属性设置为justify，那么这个属性可以指定字符分散对齐的细节。

示例 document.getElementById("inset").style.textJustify = "distribute-center-last";

值 下列字符串常量之一：auto | distribute | distribute-all-lines | distribute-center-last | inter-cluster | inter-ideograph | inter-word | kashdia | newspaper。关于这些值的具体含义，请参见第4章中的text-justify属性。

默认值 auto

textKashidaSpace

IE 5.5 NN n/a Moz n/a Saf n/a Op n/a DOM n/a

可读/可写

在字符对齐样式为justify的块级元素中，如果包含阿拉伯文字，那么这个属性可以控制卡须达（kashida，即阿拉伯文字中的“كشيدة”）在空白空间中的比例。

示例 document.getElementById("inset").style.textKashidaSpace = "15%";

值 字符串形式的百分比值。

默认值 0%

textOverflow

IE 6 NN n/a Moz n/a Saf 1.3/2 Op n/a DOM n/a

可读/可写

当文本内容溢出了固定的容器盒空间时，这个属性可以指定是否在行尾显示一个省略符号（……），以表示还有更多的文本信息。使用这个功能时还须要将元素的overflow样式属性设置为hidden。

示例 document.getElementById("textBox").style.textOverflow = "ellipsis";

值 请使用以下两个常量字符串之一：clip | ellipsis。

默认值 clip

textShadow

IE *5(Mac)* NN *n/a* Moz *all* Saf *1.2* Op *7* DOM *2*

可读/可写

此属性控制着元素文本的阴影效果。尽管这个属性的确是IE 5/Mac、Mozilla和Opera中style对象的成员之一，但无论是改变style元素属性还是使用脚本修改这个属性，都不会改变元素文本的显示效果。

值 由一个或多个阴影规范组成的字符串。每个阴影规范都包含多个由空格分隔的效果值，例如颜色、距离文本右边缘的偏移量、距离文本下方的偏移量，以及一个可选的扩散半径。多个阴影规范间须要使用逗号进行分隔，而none则表示关闭阴影效果。

默认值 none

textTransform

IE *4* NN *n/a* Moz *all* Saf *all* Op *7* DOM *2*

可读/可写

控制元素文本的大小写转换。如果这个属性的值不为none，那么样式单会重新整理源代码中的所有字母，此时可能会改变源文本字符的大小写形式。

示例
```
document.getElementById("heading").style.textTransform = "capitalize";
```


值 值为none时会按照源文本中的原有形式显示文本。其他可用的常量字符串包括：capitalize、lowercase、uppercase。capitalize会将每个单词的首字母转变为大写形式。而lowercase和uppercase则使用小写或大写形式显示元素中的文本字符。

默认值 none

textUnderlinePosition

IE *5.5* NN *n/a* Moz *n/a* Saf *n/a* Op *n/a* DOM *n/a*

可读/可写

如果一个元素的text-decoration样式属性为underline，那么这个属性可以控制下划线是出现在文本之上还是文本下方。

示例
```
document.getElementById("heading").style.textUnderlinePosition = "above";
```

值 IE 5.5可以识别两个常量值：above、below。而IE 6则增加了另外两个值，auto和auto-pos，不过它们的效果看起来似乎完全一样。不同版本的默认值也不相同，IE 5.5为below，而IE 6则为auto。另外，在竖排的日文文本中，auto属性值会让IE 6在字符的右侧显示竖线。

默认值 none（IE 5.5）；auto（IE 6）。

top

IE *4* NN *n/a* Moz *all* Saf *all* Op *7* DOM *2*

可读/可写

对可定位的元素而言，它定义了元素盒（包括元素内容及顶部填充、边框，以及边距）的上边缘相对于临近的最外层块级容器的上边缘的偏移量。针对一个相对定位元素，元素通常会出现在容器内容中的某个位置，那么就根据内嵌位置的上边缘来决定偏移量。

如果要对这个属性值进行精确计算，那么可以使用parseFloat()转换其返回值，在IE中还可以访问pixelTop或posTop样式属性，它们均会返回真实的数值。

示例
```
document.getElementById("blockD2").style.top = "40px";
```

值 一个由包含长度单位的数字值、百分比或auto组成的字符串。

默认值 auto

unicodeBidi

IE *5* NN *n/a* Moz *all* Saf *all* Op *8* DOM *2*

可读/可写

这个属性与direction样式属性一起控制着内嵌的双向文本，如法语与阿拉伯语混写的文本。

示例 `document.getElementById("blockD2").style.unicodeBidi = "bidi-override";`
值 字符串常量：`bidi-override | embed | normal`。
默认值 `normal`

verticalAlign

IE 4 NN n/a Moz all Saf 1.2 Op 7 DOM 2

可读/可写

此属性决定了元素的竖直对齐方式。根据所选择的值，这个属性在两个领域内工作。关于`vertical-align`样式单属性的详细讨论请参见第4章。

示例 `document.getElementById("myDIV").style.verticalAlign = "text-top";`
值 可以使用字符串形式的绝对值（包括单位）、一个百分比（相对于临近外层容器盒元素）或以下的某个常量值之一：`bottom | top | baseline | middle | sub | super | text-bottom | text-top`。
默认值 `baseline`

visibility

IE 4 NN n/a Moz all Saf all Op 7 DOM 2

可读/可写

指明已定位元素的可见性的具体状态。当元素的`visibility`属性设置为`hidden`时，周围的其他内容也不会填充到此元素留下的空白空间。

示例 `document.getElementById("myDIV").style.visibility = "hidden";`
值 下面4个字符串常量之一：`collapse | hidden | inherit | visible`。
默认值 `visible`

voiceFamily, volume

请参见azimuth。

whiteSpace

IE 5(Mac)/5.5(Win) NN n/a Moz all Saf all Op 7 DOM 2

可读/可写

此属性可以设定如何处理源代码中的空白内容，例如引导空格（leading space）和换行等。

示例 `document.getElementById("myDIV").style.whiteSpace = "pre";`
值 下面3个字符串常量之一：`normal | nowrap | pre`。`normal`允许浏览器在块级元素中换行并忽略引导空格。`nowrap`不允许进行换行，但依然会忽略空格。而`pre`则会保留引导空格、额外的空白并允许在源代码中换行。值得注意的是，除非通过`DOCTYPE`元素将文档设置为标准兼容模式，否则IE 6/Windows并不会响应`pre`值。
默认值 `normal`

widows

请参见orphans。

width

请参见height。

wordBreak

IE 5(Win) NN n/a Moz n/a Saf n/a Op n/a DOM n/a

可读/可写

为包含表意（象形）语言的文本内容指定断字（word-break）样式。

示例 `document.getElementById("myDIV").style.wordBreak = "keep-all";`
值 下面3个字符串常量之一：`break-all | keep-all | normal`。
默认值 `normal`

wordSpacing

IE *4* NN *n/a* Moz *all* Saf *all* Op *7* DOM *2*

可读/可写

控制文字间的间距宽度。如果在某个受此属性控制的元素中还内嵌着其他的元素，那么IE 5/Macintosh在处理内嵌元素的字间距时可能会造成字符重叠。

示例 `document.getElementById("myDIV").style.wordSpacing = "1.0em";`
值 字符串形式的CSS长度值或常量`normal`。
默认值 `normal`

wordWrap

IE *5.5* NN *n/a* Moz *n/a* Saf *1.3/2* Op *n/a* DOM *n/a*

可读/可写

此属性为块级元素指定自动换行的样式。如果仅有一个字超出了元素容器盒的宽度，那么常用的处理方式就是让这部分内容超出容器盒宽度，而不进行换行处理。但也可以迫使长单词在容器盒的边缘发生换行。

示例 `document.getElementById("myDIV").style.wordWrap = "break-word";`
值 下列字符串常量之一：`break-word | normal`。
默认值 `normal`

writingMode

IE *5.5* NN *n/a* Moz *n/a* Saf *n/a* Op *n/a* DOM *n/a*

可读/可写

主要在那些使用竖直排列的词句设置的语言中调用这个属性，它可以控制文本内容的前进方向，即从左到右还是从右到左。

示例 `document.getElementById("myDIV").style.writingMode = "lr-tb";`
值 下面两个字符串常量之一：`lr-tb | tb-rl`。`tb-rl`还可以将某些语言的文字旋转90度。
默认值 `lr-tb`

zIndex

IE *4* NN *n/a* Moz *all* Saf *all* Op *7* DOM *2*

可读/可写

对一个已定位的元素而言，这个属性决定了在同一个父级容器内，这个元素相对于其他元素的叠放次序。请参见在线参考IV，它讲述了在多个容器内元素摆放层次之间的相互关系。

示例 `document.getElementById("myDIV").style.zIndex = "3";`
值 整数值。Mozilla认为应该使用字符串形式的属性值，而IE则返回一个数字。
默认值 `0`

zoom

IE *5.5* NN *n/a* Moz *n/a* Saf *n/a* Op *n/a* DOM *n/a*

可读/可写

此属性可以控制显示时的放大倍数。在高分辨率的显示器中显示网页内容时这个属性尤为有用。请参见`screen.logicalXDPI`属性。

示例 `document.body.style.zoom = "200%";`
值 百分比值（通常是`100%`），浮点倍数（通常为`1.0`）或常量`normal`。

默认值 normal

getPropertyCSSValue( )

IE *n/a* NN *n/a* Moz *1.7* Saf *n/a* Op *n/a* DOM *2*

```
getPropertyCSSValue("CSSAttributeName")
```

返回一个代表CSS值的对象。在W3C DOM中，此方法返回的CSSValue对象的属性可以显示属性/值对的文本内容，以及对应于一长串原始值类型的整数值，这些类型包括百分比、像素长度或RGB颜色。

返回值 CSSValue对象的引用。

参数

CSSAttributeName

来自于一个内联样式声明的CSS属性名称（并不是DOM版本的属性名称）。

getPropertyPriority( )

IE *n/a* NN *n/a* Moz *all* Saf *all* Op *9* DOM *2*

```
getPropertyPriority("CSSAttributeName")
```

返回与内联CSS属性有关的优先级（如!important）字符串。

返回值 字符串。

参数

CSSAttributeName

来自于一个内联样式声明的CSS属性名称（并不是DOM版本的属性名称）。

getPropertyValue( )

IE *5(Mac)* NN *n/a* Moz *all* Saf *all* Op *7* DOM *2*

```
getPropertyValue("CSSAttributeName")
```

返回内联CSS属性/值对的字符串。

返回值 字符串。

参数

CSSAttributeName

来自于一个内联样式声明的CSS属性名称（并不是DOM版本的属性名称）。

item( )

IE *5(Mac)* NN *n/a* Moz *all* Saf *all* Op *n/a* DOM *1*

```
item(index)
```

根据参数传入的索引值从源代码中找到与之对应的内联CSS属性/值对，并返回其属性名称。但Safari会错误地返回CSS值，而不是其名称。

返回值 字符串。IE/Macintosh返回全部为大写字符的名称字符串，而Mozilla则会返回全小写字符的字符串。

参数

index

一个从零开始的整数，依照源代码次序，对应于某个特定的CSS属性/值对。

removeProperty( )

IE *n/a* NN *n/a* Moz *all* Saf *all* Op *7* DOM *2*

```
removeProperty("CSSAttributeName")
```

删除内联的CSS属性/值对，并返回其属性值字符串。

返回值 字符串。

参数

CSSAttributeName

来自于一个内联样式声明的CSS属性名称（并不是DOM版本的属性名称）。

setProperty( )

IE *5(Mac)* NN *n/a* Moz *all* Saf *all* Op *7* DOM *2*

```
setProperty("CSSAttributeName", "value", "priority")
```

设定一个内联的样式属性/值对。如果已经存在同名属性，那么会将新值赋予该属性，否则向元素中添加这个属性和值。

返回值 无。

参数

CSSAttributeName

来自于一个内联样式声明的CSS属性名称（并不是DOM版本的属性名称）。

value

字符串形式的值，要确保其格式适用于该属性。

priority

赋值权限字符串，或者空字符串。

styleSheet, CSSStyleSheet

IE *4* NN *n/a* Moz *all* Saf *all* Op *7* DOM *1*

styleSheet对象（在W3C DOM抽象模型中为CSSStyleSheet对象，它就继承自DOM的StyleSheet对象）代表了一个样式单，既可以使用style元素创建这个样式单，也可以通过link元素或style元素的@import指令来导入样式单。这个对象不同于style元素对象，后者完整地表征了HTML元素style及其属性。而document.styleSheets[]集合就包含着零个或多个styleSheet对象。在所有支持这一对象的浏览器中，都可以使用disabled属性，这样就能通过一个布尔赋值来控制整个样式单的启用与禁用。

对象模型引用方式 [window.]document.styleSheets[i]

对象特定属性 cssRules[]、cssText、href、imports[]、media、ownerNode、ownerRule、owningElement、pages[]、parentStyleSheet、readOnly、rules[]、title、type

对象特定方法 addImport()、addRule()、deleteRule()、insertRule()、removeRule()

对象特定事件 无。

cssRules[ ]

IE *5(Mac)* NN *n/a* Moz *all* Saf *all* Op *7* DOM *2*

只读

返回当前styleSheet对象中内嵌的cssRule对象所组成的集合。IE中与之等价的是rules属性。请参见本章前文中的cssRules对象，以便了解这个集合对象的属性和方法，另外在cssRule对象则讲解了集合中每个对象的详细信息。

示例

```
var allCSSRules = document.styleSheets[0].cssRules;
```

值 一个CSSRules集合对象的引用。

默认值 长度为0的数组。

cssText

IE *5* NN *n/a* Moz *n/a* Saf *n/a* Op *n/a* DOM *n/a*

可读/可写

此属性包含着样式单中定义的全部规则的完整文本内容。如果要使用一个新的规则集来完全代替旧的设定，那么这个属性将大有用处。如果只想作用于IE中某条规则的文本，那么请访问该rule对象的cssText属性，例如从styleSheet对象的rules[i].cssText属性中获取。而在W3C DOM浏览器中，则可以使用cssRules[i].cssText属性。

示例

```
var allCSSText = document.styleSheets[0].cssText;
```

值 字符串。
默认值 空字符串。

href

IE 4 NN n/a Moz all Saf all Op 7 DOM 2

可读/可写 (IE)

这个属性对应于link元素的href属性所指定的URL地址。IE/Windows中可以对这个属性进行读写，但它在W3C DOM浏览器中是一个只读的属性。

示例
```
document.styleSheets[1].href = "css/altStyles.css";
```
值 由完整或相对URL路径组成的字符串。
默认值 无。

imports[ ]

IE 4 NN n/a Moz n/a Saf n/a Op n/a DOM n/a

只读

此属性返回一个由styleSheet对象组成的集合（数组），这些对象已通过@import规则导入到一个styleSheet对象之中。详细内容请参见本章中已介绍过的imports集合对象。对于W3C DOM浏览器，必须遍历styleSheet对象中的所有cssRule对象才能找到哪些对象的type属性值等于3（即IMPORT_RULE）。

示例
```
var allImportRules = document.styleSheets[0].imports;
```
值 一个imports集合对象的引用。
默认值 长度为0的数组。

796

media

IE 4 NN n/a Moz all Saf all Op 7 DOM 2

可读/可写 (IE)

为样式单控制的内容指定预期的输出设备，对应于link和style元素的media属性。IE使用字符串作为属性值，而W3C DOM浏览器则须要使用一个只读的MediaList对象。

示例
```
var theMedia = document.styleSheets[2].media;
theMedia = (typeof theMedia.mediaText != "undefined") ? theMedia.mediaText :
theMedia;
if (theMedia.match(/print/) {
    // 处理打印输出
}
```
值 IE中可以使用all | print | screen之一作为属性值，而W3C DOM浏览器则只能使用MediaList对象。
默认值 all

797

ownerNode

IE n/a NN n/a Moz all Saf all Op 7 DOM 2

只读

此属性会返回包含当前styleSheet对象的节点树对象的引用。根据样式单在文档中的不同定义方式，这个节点可能是一个style元素也可能是一个link元素。IE（Windows和Mac）中的等价属性是owningElement。

示例
```
var mama = document.styleSheets[2].ownerNode;
```
值 对象引用。
默认值 无。

ownerRule

IE *n/a* NN *n/a* Moz *all* Saf *all* Op *7* DOM *2*

只读

当使用@import规则将一个styleSheet对象导入文档时，此属性可以返回该@import规则对象（即W3C DOM的CSSImportRule对象）的一个引用。cssRule对象提供了一些应用于CSSImportRule对象的属性和方法。对于其他的样式类型，此属性均返回null。

示例 `var hostRule = document.styleSheets[2].ownerRule;`

值 对象引用或null。

默认值 null

owningElement

IE *4* NN *n/a* Moz *n/a* Saf *n/a* Op *n/a* DOM *n/a*

只读

返回定义了当前styleSheet对象的style或link元素对象的引用。每个文档都维持着一个由style和link元素定义的样式单对象集合。与之类似的W3C DOM属性是ownerNode。

示例 `var firstStyleID = document.styleSheets[0].owningElement.id;`

值 元素对象的引用。

默认值 无。

pages[ ]

IE *5.5* NN *n/a* Moz *n/a* Saf *n/a* Op *n/a* DOM *n/a*

只读

它可返回一个由styleSheet对象中内嵌的page对象（@page规则）所组成的一个集合（数组）。对于W3C DOM浏览器，必须遍历styleSheet对象中的所有cssRule对象才能找到type属性值等于6（即PAGE_RULE）的对象。请参见page对象。

示例 `var allPageRules = document.styleSheets[0].pages;`

值 一个pages集合对象的引用。

默认值 长度为0的数组。

parentStyleSheet

IE *4* NN *n/a* Moz *all* Saf *all* Op *7* DOM *2*

只读

对于一个由@page规则生成的styleSheet对象而言，parentStyleSheet属性将返回导入当前样式单的styleSheet对象引用。对于未导入的样式单，此属性会返回null。

示例 `var myMaker = document.styleSheets[0].parentStyleSheet;`

值 styleSheet对象的引用。

默认值 null

readOnly

IE *4* NN *n/a* Moz *n/a* Saf *n/a* Op *n/a* DOM *n/a*

只读

此属性可以指明能否通过脚本修改样式单。由于无法修改那些由link元素或@import规则导入的样式单，因此在这种情况下返回值为true。

值 布尔值：true | false。

默认值 false

rules[ ]

IE 4 NN n/a Moz n/a Saf all Op n/a DOM n/a

只读

返回由当前styleSheet对象中内嵌的rule对象所组成的集合。这个属性与W3C DOM中的cssRules属性等价。Safari可以支持这两个属性。请参见本章前文中的cssRules对象，以便了解这个集合对象的属性和方法，另外在cssRule对象中则讲解了集合中每个对象的详细信息。

示例 `var allrules = document.styleSheets[0].rules;`

值 一个rules集合对象的引用。

默认值 长度为0的数组。

title

IE 4 NN n/a Moz all Saf all Op 7 DOM 2

可读/可写

此属性暴露了拥有当前styleSheet对象的style或link元素的title属性。由于这个属性无法影响用户接口元素，因此可以在脚本控制下通过它向styleSheet对象传递其他字符串信息。

示例

```
if (document.styleSheets[2].title == "corpStyleWindows") {
  // 处理指定样式
}
```

值 字符串。

默认值 空字符串。

type

IE 4 NN n/a Moz all Saf all Op 7 DOM 2

只读

返回style或link元素的type属性所指定的样式单MIME类型。

示例

```
if (document.styleSheets[0].type == "text/css") {
  ...
}
```

值 字符串，典型的CSS样式单类型为“text/css”。

默认值 无。

addImport( )

IE 4 NN n/a Moz n/a Saf n/a Op n/a DOM n/a

```
addImport(url, [index])
```

为styleSheet对象添加一个外部样式单规范。

返回值 如果忽略了第2个参数并让浏览器在集合末尾插入新样式，那么会返回该样式在styleSheets[]集合中所处位置的整数索引值。

参数

url

样式单文件（.css）的完整或相对URL地址。

index

一个可选的整数，表示新元素应该放置在集合中的哪个位置。

addRule( )

IE 4 NN n/a Moz n/a Saf n/a Op n/a DOM n/a

```
addRule("selector", "style"[, index])
```

为样式单增加一条新规则。使用这个方法就可以通过脚本向一个已有的styleSheet对象中加入一条新的规则，语法如下：

```
document.styleSheets[1].addRule("p b","color:red");
```

此时，可以复制一个已存在于styleSheet中的选择器，从而使用新的规则来替代原来的规则。如果要将一个规则转化为一个明语样式规则，并用它创建一个可定位元素，那么这种做法是唯一禁止的做法，反之亦然。此外，新加入的规则和其他样式单规则一样也由相同的层叠规则管理。因此styleSheet对象中的新规则并不会替代元素style属性中的样式集。

返回值 早期的IE浏览器不会返回任何值。在较新的版本中，IE/Windows返回-1，而IE/Macintosh则会返回null。在将来的某一天，返回值可能会变为新规则所在位置的整数索引值。

参数

selector

字符串形式的样式规则选择器。

style

一个或多个样式“attribute:value”对。请按照普通样式单定义中的做法，使用分号分隔多个样式。

index

一个可选的整数，表示新元素应该放置在集合中的哪个位置。

deleteRule( )

IE *5(Mac)* NN *n/a* Moz *all* Saf *all* Op *7* DOM *2*

```
deleteRule(index)
```

从一个styleSheet对象中删除一条规则。参数传入的整数索引值指向cssRules数组中某条将要被删除的样式规则。须要注意的是，IE 5/Macintosh同时实现了微软的removeRule()方法和W3C DOM的deleteRule()方法。

返回值 无。

参数

index

一个从零开始的整数，指明应该删除cssRules集合中的哪一条规则。

insertRule( )

IE *5(Mac)* NN *n/a* Moz *all* Saf *all* Op *7* DOM *2*

```
insertRule("ruleText", index)
```

为样式单增加一条新规则。使用这个方法就可以通过脚本向一个已有的W3C DOM的styleSheet对象中加入一条新的规则，语法如下：

```
document.styleSheets[1].insertRule("p b {color:red}", 0);
```

此时可以复制一个已存在于styleSheet中的选择器，从而让新规则替代原来的规则。如果要将一个规则转化为一个明语样式规则，并用它创建一个可定位元素，那么这种做法是唯一禁止的做法，反之亦然。此外，新加入的规则和其他样式单规则一样也由相同的层叠规则管理。因此styleSheet对象中的新规则并不会替代元素style属性中的样式集。须要注意的是，IE 5/Macintosh为达到同一结果，既实现了W3C DOM的insertRule()方法，又实现了微软的addRule()方法。

返回值 新规则所在位置的整数索引值。

参数

ruleText

字符串形式的样式规则选择器，其格式与样式元素中指定的格式完全相同，如下所示：*selector {attribute:value; attribute:value;...}*.

index

一个可选的整数，表示新规则应该放置在cssRules集合中的哪个位置。

removeRule( )

IE *4* NN *n/a* Moz *n/a* Saf *n/a* Op *n/a* DOM *n/a*

```
removeRule(index)
```

从一个styleSheet对象中删除一条规则。参数传入的整数索引值指向rules数组中某条将要被删除的样式规则。

返回值 无。

参数

index

一个从零开始的整数，指明应该删除rules集合中的哪一条规则。

styleSheets, StyleSheetList

IE *4* NN *n/a* Moz *all* Saf *all* Op *7* DOM *2*

由document对象所包含的styleSheet对象组成的集合。W3C DOM中，这个集合的抽象表示称为StyleSheetList对象。虽然只能通过其整数索引号直接访问集合中的成员，但也可以遍历整个集合并检查每个样式单对象的属性（例如，selectorText属性）以便对它们进行区分。

对象模型引用方式 document.styleSheets

对象特定属性 length

对象特定方法 item()

对象特定事件 无。

length

IE *4* NN *n/a* Moz *all* Saf *all* Op *7* DOM *2*

802

只读

返回集合中元素的数量。

示例

```
var howMany = document.styleSheets.length;
```

值 整数值。

item( )

IE *4* NN *n/a* Moz *all* Saf *all* Op *7* DOM *2*

item(*index*)

返回一个styleSheet对象，其对应对象的源代码次序可与参数传入的索引值相匹配。

返回值 请参见styleSheet对象。如果并不存在与参数相匹配的对象，则返回null。

参数

index

一个从零开始的整数，依照源代码次序，对应于集合中的某个特定项（内嵌在当前document对象之中）。

sub, sup

IE *4* NN *n/a* Moz *all* Saf *all* Op *7* DOM *1*

sub对象对应于sub元素，而sup对象则对应于sup元素。浏览器倾向于使用较小的字体尺寸来显示这两个对象中的内容。IE 5/Macintosh还为这两个元素提供了两个特定的只读属性——height和width，但其他浏览器中的对象模型均未实现这两个属性。

等价HTML元素 <sub>、<sup>

对象模型引用方式 [window.]document.getElementById("*elementID*")

对象特定属性 无。

对象特定方法 无。

对象特定事件 无。

submit

803

请参见input。

sup

请参见sub。

table

IE 4 NN n/a Moz all Saf all Op 7 DOM 1

table对象对应于table元素。其他与table对象有关的对象包括：caption、col、colgroup、tbody、td、tfoot、thead和tr。

等价HTML元素 <table>

对象模型引用方式 [window.]document.getElementById("*elementID*")

对象特定属性 align、background、bgColor、border、borderColor、borderColorDark、borderColorLight、caption、cellPadding、cells[]、cellSpacing、cols、dataPageSize、frame、height、rows[]、rules、summary、tBodies[]、tFoot、tHead、width

对象特定方法 createCaption()、createTFoot()、createTHead()、deleteCaption()、deleteRow()、deleteTFoot()、deleteTHead()、insertRow()、lastPage()、moveRow()、nextPage()、previousPage()、refresh()

对象特定事件 无。

align

IE 4 NN n/a Moz all Saf all Op 7 DOM 1

可读/可写

定义元素在其容器内的水平对齐方式。

示例 document.getElementById("myTable").align = "center";

值 以下3个对齐常量之一：center | left | right。

默认值 left

background

IE 4 NN n/a Moz n/a Saf n/a Op n/a DOM n/a

可读/可写

提供表格背景图片的URL地址。如果同时为元素设置了backgroundColor，那么当图片加载失败时会出现该背景色，否则图片会覆盖背景色。

示例 document.getElementById("myTable").background = "images/watermark.jpg";

值 背景图片文件的完整或相对URL。

默认值 无。

bgColor

IE 4 NN n/a Moz all Saf all Op 7 DOM 1

可读/可写

指定该元素的背景色。而样式单的backgroundColor属性并不能反映这个属性设定。即使使用明语颜色名称设置元素或对象的bgcolor属性，其返回值仍然是一个十六进制三元组。

示例 document.getElementById("myTable").bgColor = "yellow";

值 一个十六进制的三元组或明语颜色名称。目前可用的明语颜色名称请参见附录A。

默认值 由浏览器和操作系统共同决定。

border

IE 4 NN n/a Moz all Saf all Op 7 DOM 1

可读/可写

指定了表格四周边框的厚度，单位为像素。这是表格的默认3D边框，请不要将它与样式单创建的边框混为一谈。

示例 `document.getElementById("myTable").border = 4;`

值 整数值。设置为0时则完全不显示边框。

默认值 0

borderColor

IE *4* NN *n/a* Moz *n/a* Saf *n/a* Op *n/a* DOM *n/a*

可读/可写

此属性指定了表格边框的颜色。IE会将颜色应用于组成单元格边框的4条线段之上。因此临近的单元格边框颜色并不会互相冲突。

示例 `document.getElementById("myTable").borderColor = "salmon";`

值 一个十六进制的三元组或明语颜色名称。不设置属性值就默认为黑色（`#000000`，black）。目前可用的明语颜色名称请参见附录A。

默认值 由操作系统决定。

805

borderColorDark, borderColorLight

IE *4* NN *n/a* Moz *n/a* Saf *n/a* Op *n/a* DOM *n/a*

可读/可写

如果在页面背景或默认颜色四周仔细地勾勒出明暗线条，就可以在IE中创造出3D效果的表格边框。通过为`borderColorDark`（单元格的左边缘与上边缘）和`borderColorLight`（单元格的右边缘与下边缘）属性指定属性值，还可以独立地控制明暗两种线条的颜色。

此时应该为这对属性设置互补的颜色。而对于具体的颜色设置并没有强制性要求，并不一定要为`borderColorDark`属性设置暗色。这两个属性仅仅是为了使得边框线条轮廓分明，改变它们就可以控制一些边框线条的颜色。

示例

```
document.getElementById("myTable").borderColorDark = "blue";
document.getElementById("myTable").borderColorLight = "cornflowerblue";
```

值 一个十六进制的三元组或明语颜色名称。如果不设置属性值就默认为黑色（`#000000`，black）。目前可用的明语颜色名称请参见附录A。

默认值 由操作系统决定。

caption

IE *4(Win)/5(Mac)* NN *n/a* Moz *all* Saf *all* Op *7* DOM *1*

只读

返回内嵌在表格中的`caption`元素的引用。通过这个引用就可以访问`caption`对象的属性和方法。在W3C DOM浏览器中，可以创建一个新的`caption`元素，并将这个其引用赋予`table`对象的`caption`属性，这样一来，这些浏览器中这个属性就成为了可读可写的属性。但在所有的浏览器中，均可以修改`caption`属性返回的`caption`对象的诸多属性。

示例 `var capText = document.getElementById("myTable").caption.innerHTML;`

值 对象引用。

默认值 无。

cellPadding

IE *4* NN *n/a* Moz *all* Saf *all* Op *7* DOM *1*

806

可读/可写

此属性决定了表格单元格的边框与内容之间的空白空间大小。值得注意的是，此属性将作用于单元格内的空白空间。当不显示表格边框时，此属性的细微调整往往不易察觉，但依然可以通过它来调整单元格之间的间隔。

示例 `document.getElementById("myTable").cellPadding = "15";`
值 字符串形式的长度值或百分比值。
默认值 0

cells

IE *5(Win)* NN *n/a* Moz *n/a* Saf *all* Op *9* DOM *n/a*

只读

返回由表格内所有td元素所组成的集合。数组中的各项均按照对应td元素在源代码中的顺序进行排序。在tr元素中这个属性使用得更为广泛。

示例 `var totCells = document.getElementById("myTable").cells.length;`
值 一个cells集合对象的引用。
默认值 长度为0的数组。

cellSpacing

IE *4* NN *n/a* Moz *all* Saf *all* Op *7* DOM *1*

可读/可写

此属性决定了表格单元格的外部边框之间的空白空间大小。如果表格拥有边框，那么设置cellspacing属性就会定义显示在单元格之间的边框的厚度。即使边框不可见，单元格间距也有利于增强表格的可读性。

示例 `document.getElementById("myTable").cellSpacing = "5";`
值 字符串形式的长度值或百分比值。
默认值 0（无表格边框），2（使用表格边框）。

cols

IE *4* NN *n/a* Moz *n/a* Saf *n/a* Op *n/a* DOM *n/a*

可读/可写

此属性指定了表格的列数。其对应的元素属性cols能够辅助浏览器来为表格显示做出准备。如果不使用此属性，那么浏览器只能依靠解析tr和td元素来决定表格的分割形式。尽管这个属性是可读可写的，但通过修改这个属性并不能改变表格的列结构。请参见本章中已介绍过的col对象。

示例 `document.getElementById("myTable").cols = 5;`
值 整数值。
默认值 无。

dataPageSize

IE *4* NN *n/a* Moz *n/a* Saf *n/a* Op *n/a* DOM *n/a*

可读/可写

本属性用于IE浏览器中的数据绑定。它可以建议浏览器显示一定的表格行来适应此属性所提供的数据源记录数。如果要在记录组之间实现跳转，请参见lastPage()、nextPage()和previousPage()方法。

示例 `document.getElementById("inventoryTable").dataPageSize = 10;`
值 整数值。
默认值 无。

frame

IE *4* NN *n/a* Moz *all* Saf *all* Op *7* DOM *1*

可读/可写

此属性决定了须要显示表格哪一侧的外部边框。但它并不会影响单元格之间的内部边框。

示例 `document.getElementById("orderForm").frame = "hsides";`

值

下列字符串形式的框架常量之一，均不区分大小写：

above
仅显示表格顶部的边框

below
仅显示表格底部的边框

border
与box相同，显示四周的所有边框

box
与border相同，显示四周的所有边框

hsides
仅显示表格顶部与底部的边框

lhs
仅显示表格左侧的边框

rhs
仅显示表格由侧的边框

void
隐藏所有的边框（HTML 4中的默认值）

vsides
仅显示表格左侧和右侧的边框

默认值 void（当border为0时），border（当border为其他任意值时）。

height, width

IE *4* NN *n/a* Moz *all* Saf *all* Op *7* DOM *1*

可读/可写

指定了元素的高度和宽度，单位为像素。改变这些值后，页面的内容排列也会立刻随之发生变化。由于W3C DOM浏览器认为表格的高度取决于每行中最高单元格的高度之和，因此在这些浏览器中只能使用width属性。

示例 `document.getElementById("myTable").height = 250;`
值 整数值。
默认值 无。

rows

IE *4* NN *n/a* Moz *all* Saf *all* Op *7* DOM *1*

只读

返回由表格内所有tr元素所组成的集合。通过这个属性也可以获得表格分段（tbody、tfoot和thead元素对象）中的行分组。

示例 `var allTableRows = document.getElementById("myTable").rows;`
值 一个rows集合对象的引用。
默认值 长度为0的数组。

rules

IE *4* NN *n/a* Moz *all* Saf *all* Op *7* DOM *1*

可读/可写

如果存在单元格之间的内部边框，那么通过此属性可以指定它们的显示位置。使用此属性，除了可以通过绘制边框使得表格以阵列的形式显示各单元格之外，还可以通过边框仅仅分隔出一些行、列或任意其他类型的单元格分组（如thead、tbody、tfoot、colgroup或col）。为了显示单元格的边框，必须使用border属性，其属性值既

可以是一个布尔变量也可以是一个特定的边框尺寸。使用时请不要将这个属性与styleSheet对象的rules[]集合混为一谈。通过脚本改变这一属性并不是总能获得预期的效果，这一情况在早期的Netscape 6中尤其明显。

示例 `document.getElementById("myTable").rules = "groups";`

值

下列字符串形式的规则常量之一，均不区分大小写：

all
显示每个单元格四周的边框

cols
仅显示表格列之间的边框

groups
显示单元格分组之间的边框，这些分组可通过thead、tfoot、tbody、colgroup或col等元素定义

none
隐藏所有内部边框

rows
仅显示表格行之间的边框

默认值 none（当border为0时）；all（当border为其他任意值时）。

609

summary

IE *6* NN *n/a* Moz *all* Saf *all* Op *7* DOM *1*

只读

此属性对应于HTML 4的summary属性，但它并未在浏览器中提供任何特殊功能。既便如此，这个属性仍然有它的用武之地，例如，可以在源代码中为它指定一个属性值，脚本在读取这个属性后就可以获得这些数据信息。

示例 `var data = document.getElementById("myTable").summary;`

值 字符串。

默认值 空字符串。

tBodies[ ]

IE *4(Win)/5(Mac)* NN *n/a* Moz *all* Saf *all* Op *7* DOM *1*

只读

返回由当前表格中tBody元素对象所组成的一个集合。每个table元素中至少包含一个内嵌的tBody元素对象。

示例 `var bodSections = document.getElementById("myTable").tBodies;`

值 tBody对象集合的引用。

默认值 长度为1的数组。

tFoot

IE *4(Win)/5(Mac)* NN *n/a* Moz *all* Saf *all* Op *7* DOM *1*

只读

如果表格中定义了一个tfoot元素，那么将返回该元素对象的引用。如果表格中并不存在tfoot元素，那么返回值为null。通过这个属性的返回值就可以访问tFoot元素对象的属性和方法。

示例 `var tableFootTxt = document.getElementById("myTable").tFoot.firstChild.nodeValue;`

值 tfoot元素对象的引用。

默认值 null

610

tHead

IE *4(Win)/5(Mac)* NN *n/a* Moz *all* Saf *all* Op *7* DOM *1*

只读

如果表格中定义了一个thead元素，那么返回该元素对象的引用。如果表格中并不存在thead元素，那么返回值为null。通过这个属性的返回值就可以访问tHead元素对象的属性和方法。

示例 `var tableHeadTxt = document.getElementById("myTable").tHead.firstChild.nodevalue;`
值 thead元素对象的引用。
默认值 `null`

width

请参见height。

createCaption( ), deleteCaption( )

IE 4 NN n/a Moz all Saf all Op 7 DOM 1

`deleteCaption()`方法表示从当前`table`元素中删除一个内嵌的`caption`元素，而`createCaption()`则表示将一个`caption`元素内嵌在当前`table`元素之中。如果不存在表格标题，那么创建方法可以创造一个空的标题元素，然后须要使用脚本向它填充标题文本。如果存在一个标题，那么通常会忽略这个方法，并非返回当前`caption`元素的引用。

返回值 Reference to new caption element (for `createCaption()`); nothing for `deleteCaption()`.
参数 无。

createTFoot( ), createTHead( ), deleteTFoot( ), deleteTHead( )

IE 4 NN n/a Moz all Saf all Op 7 DOM 1

这些方法分别表示在当前`table`元素中添加或删除一个`thead`或`tfoot`元素。如果表格中不存在表头和表尾分块，那么创建方法会生成一个空元素，此后必须使用`thead.insertRow()`和`tfoot.insertRow()`方法来填充这些区域。如果存在一个对应的分块，那么通常会忽略这个方法，并返回已存在的`thead`或`tfoot`元素的引用。

返回值 `createTFoot()`和`createTHead()`会返回新创建的元素的引用，而`deleteTHead()`和`deleteTFoot()`则不会返回任何数据。
参数 无。

deleteRow( )

IE 4 NN n/a Moz all Saf all Op 7 DOM 1

`deleteRow(`*index*`)`

从当前`table`元素中删除一个内嵌的`tr`元素。而参数传入的整数值就指向`rows`集合中待删除的`tr`元素。如果要使用新内容重构一个表格，那么首先须要将该表格清空，可以使用如下的迭代语句通过调用`deleteRow()`方法来达到这一目的：

```
while (tableReference.rows.length > 0) {
    tableReference.deleteRow(0);
}
```

返回值 无。
参数

index

一个从零开始的整数，依照源代码次序，它对应于集合中的某个特定`tr`元素（内嵌在当前元素之中）。

insertRow( )

IE 4 NN n/a Moz all Saf all Op 7 DOM 1

`insertRow(`*index*`)`

向当前`table`元素中插入一个内嵌的`tr`元素。参数传入的整数值指向`rows`集合中某个从零开始的位置，以便插入新的行。但在IE中也可以简单地使用“`-1`”作为参数，这时会将新行直接附加到集合的末尾。但此时加入的行只是插入了一个空元素，因此还须要使用`insertCell()`方法向该元素内增加单元格。然而不幸的是，在IE/Macintosh中使用脚本为表格添加行和单元格并不优雅，这种方法往往会产生十分硕大的行和单元格尺寸。对于非内嵌表格，可以使用`table`对象的简便方法来规避文档树节点的创建和插入工作。

返回值 新插入的行对象的引用。
参数
index
从零开始的一个整数，它对应于rows集合中的一个行对象，而新的行将插入到该行之前。

lastPage( ), nextPage( ), previousPage( )

IE *4/5* NN *n/a* Moz *n/a* Saf *n/a* Op *n/a* DOM *n/a*

这3个方法可以让数据绑定功能分别从数据源中读取最后一组、下一组或前一组数据，以便填充dataPageSize属性指定的数据记录。须要注意的是，lastPage()方法只适用于IE 5及后续版本的浏览器。
返回值 无。
参数 无。

moveRow( )

IE *5(Win)* NN *n/a* Moz *n/a* Saf *n/a* Op *n/a* DOM *n/a*

812
```
moveRow(indexToMove, destinationIndex)
```

将表格中的某一行从原有位置移动到另一个位置。第1个参数指定了待移动的行在rows集合中的索引位置（从零开始计数）。而第2个参数则表示会将该行移动到哪一行之前。作为table对象的一个方法，moveRow()的索引参数可以涵盖表格的首行，由于此行中可能包含th元素因此通常不会移动这一行。为了避免错误地移动了首行，可以在tbody对象中调用这个方法。
返回值 已被移动的行节点的引用。
参数
indexToMove
一个从零开始的指向待移行的整数。
destinationIndex
一个从零开始的指向目标位置的整数。

refresh( )

IE *4* NN *n/a* Moz *n/a* Saf *n/a* Op *n/a* DOM *n/a*

这个方法可以命令数据绑定功能从数据源重载当前页面的数据。如果表格须要频繁地从数据库获取改变后的数据，那么就可以创建setTimeout()循环来调用document.getElementById("myTable").refresh()方法，从而根据用户需要定时从服务器更新所需的信息。
返回值 无。
参数 无。

tags

IE *n/a* NN *|4|* Moz *n/a* Saf *n/a* Op *n/a* DOM *n/a*

tags对象是一种只能在Navigator 4中使用的样式单JavaScript语法。作为document对象的一个属性，tags对象常用于为构建特殊的HTML元素引用，以便获取或设置它们的样式属性。因此，tags对象的直接属性全部都是HTML元素类型。例如：

```
[document.]tags.p
[document.]tags.h1
```

但也不须要将所有的HTML元素都作为这个对象的属性。这些属性对应的元素引用都可用于类型为“text/javascript”的style元素。而此处也正是JavaScript语法为样式单属性赋值的地方，如下所示：

```
tags.p.color = "green";
tags.h1.fontSize = "14pt";
```

813 下文中列出的属性并不是tags对象自身的属性，实质上它们是与元素、类或JavaScript语法赋值语句指定的ID有关的样式单。将它们罗列在此只是为了便于说明，以及保证历史完整性。此外，本章分别列出了那些与元

素定位有关的专用属性和其他常规样式属性。关于这些属性的取值的详细信息，请参阅后续章节。

样式对象特定属性 backgroundColor、backgroundImage、borderBottomWidth、borderColor、borderLeftWidth、borderRightWidth、borderStyle、borderTopWidth、borderWidths()、color、display、fontFamily、fontSize、fontStyle、fontWeight、listStyleType、marginBottom、marginLeft、marginRight、margins()、marginTop、paddingBottom、paddingLeft、paddingRight、paddings、paddingTop、textAlign、textDecoration、textTransform、verticalAlign、whiteSpace

定位对象特定属性 background、bgColor、clip、left、top、visibility、zIndex

tBodies

IE *4* NN *n/a* Moz *all* Saf *all* Op *7* DOM *1*

这是由一个table元素包含的所有tbody元素组成的集合。此集合中的成员按照它们在源代码中的次序进行排序。

对象模型引用方式 document.getElementById("tableID").tBodies

对象特定属性 length

对象特定方法 item()、namedItem()、tags()、urns()

对象特定事件 无。

length

IE *4* NN *n/a* Moz *all* Saf *all* Op *7* DOM *1*

只读

返回集合中元素的数量。

示例
```
var howMany = document.getElementById("myTable").tBodies.length;
```

值 整数值。

item( )

IE *4* NN *n/a* Moz *all* Saf *all* Op *7* DOM *1*

```
item(index[, subindex]) item(index)
```

根据符合索引值(IE中也可以使用index或subindex值)的元素对象，返回一个单独的tBody对象或一组tBody对象集合。

返回值 一个tBody对象或一个tBody对象集合。如果并不存在与参数相匹配的对象，则返回null。

参数

index

当输入参数是一个从零开始的整数时，返回值是源代码内与某个具体项相对应的独立元素（内嵌在当前元素之内），如果参数为一个字符串（仅适用于IE），则返回id属性与该字符串相符的元素集合。

subindex

如果在IE中为第1个参数指定一个字符串值，那么可以将第2个参数指定为一个从零开始的索引值，这样就可以从集合中找到一个id属性与第1个参数的字符串相匹配的具体元素。

namedItem( )

IE *6* NN *n/a* Moz *all* Saf *all* Op *7* DOM *1*

```
namedItem("ID")
```

返回单个tBody对象或tBody对象集合，它们对应于符合参数字符串值的元素。

返回值 一个tBody对象或一个tBody对象集合。如果并不存在与参数相匹配的对象，则返回null。

参数

ID

与目标元素的id属性值相同的字符串。

tags( )

IE 4 NN n/a Moz n/a Saf n/a Op n/a DOM n/a

```
tags("tagName")
```

从当前集合的全部内嵌对象中，返回标签与tagName参数相匹配的对象集合。

urns( )

IE 5(Win) NN n/a Moz n/a Saf n/a Op n/a DOM n/a

```
urns(URN)
```

请参见all.urns()方法。

815 tbody, tfoot, thead

IE 4 NN n/a Moz all Saf all Op 7 DOM 1

tbody、tfoot和thead对象分别对应于tbody、tfoot和thead元素。从脚本编程的角度来看，可以将它们视为表格中的一种行分组容器。它们共享着相同的属性和方法，因此在使用脚本操作这些元素时要保证这些元素的HTML功能性。每个表格只能拥有一个tfoot和一个thead元素，但可以有多个tbody元素。另外，IE 4及后续版本，以及W3C DOM浏览器在默认情况下会为每个表格都创建一个tbody对象。这个默认的tbody元素会包含表格中的所有行。

等价HTML元素 <tbody>、<tfoot>、<thead>

对象模型引用方式

```
[window.]document.getElementById("elementID")
[window.]document.getElementById("tableID").tBodies[i]
[window.]document.getElementById("tableID").tfoot
[window.]document.getElementById("tableID").thead
```

对象特定属性 align、bgColor、ch、chOff、rows、vAlign

对象特定方法 deleteRow()、insertRow()、moveRow()

对象特定事件 无。

align

IE 4 NN n/a Moz all Saf all Op 7 DOM 1

可读/可写

在tbody元素控制的所有单元格中，此属性可以定义其内容的水平对齐方式。

示例 `document.getElementById("myTbody").align = "center";`

值 以下3个水平对齐常量字符串之一：center | left | right。

默认值 left

816 bgColor

IE 4 NN n/a Moz n/a Saf n/a Op n/a DOM n/a

可读/可写

指明tbody、tfoot或thead元素内单元格的背景色。样式单的backgroundColor属性并不能反映这个属性设定。即使使用明语颜色名称设置了元素或对象的bgcolor属性，其返回值还是一个十六进制三元组。

示例 `document.getElementById("myTable").tHead.bgColor = "yellow";`

值 一个十六进制的三元组或明语颜色名称。目前可用的明语颜色名称请参见附录A。

默认值 由浏览器和操作系统共同决定。

ch

IE 5(Mac)/6(Win) NN n/a Moz all Saf all Op 7 DOM 1

可读/可写

ch对应于char属性，它将一个文本字符定义为对齐参照点，列或列分组内的文本均依照该字符进行对齐。注

意，只有将align属性设置为char，此属性才有价值。但在实际使用中，即使表格的块元素对象拥有这个属性浏览器也不会对它的属性值作出任何响应。

示例 `document.getElementById("myTBody").ch = ".";`
值 单个字符组成的字符串。
默认值 无。

chOff

IE *5(Mac)/6(Win)* NN *n/a* Moz *all* Saf *all* Op *7* DOM *1*

可读/可写

定义一个偏移点，char属性所指定的字符将出现在单元格中偏移点所在位置。但在实际使用中，即使表格的块元素对象拥有这个属性浏览器也不会对它的属性值作出任何响应。

示例 `document.getElementById("myTBody").chOff = "80%";`
值 字符串形式的像素值或单元格宽度的百分比。
默认值 无。

rows

IE *4* NN *n/a* Moz *all* Saf *all* Op *7* DOM *1*

只读

返回由表格分块内所有tr元素所组成的集合。在IE/Windows中也可以获得整个表格的行分组。

示例 `var allTableRows = document.getElementById("myTFoot").rows;`
值 一个rows集合对象的引用。
默认值 长度为0的数组。

vAlign

IE *4* NN *n/a* Moz *all* Saf *all* Op *7* DOM *1*

可读/可写

针对tbody、tfoot或thead元素内的单元格，此属性指明其中文本的竖直对齐方式。

示例 `document.getElementById("myTbody").vAlign = "baseline";`
值 不区分大小写的字符串常量：baseline、bottom、middle、top。
默认值 middle

deleteRow( )

IE *4* NN *n/a* Moz *all* Saf *all* Op *7* DOM *1*

`deleteRow(`*index*`)`

从当前tbody、tfoot或thead元素中删除一个内嵌的tr元素。参数传入的整数值指向表格分块的rows集合中从零开始的某个tr元素。如果要使用新内容重构一个表格分块，那么首先须要将该表格分块清空，此时可以使用如下的迭代语句通过调用deleteRow()方法来达到这一目的：

```
while (tBodyReference.rows.length > 0) {
  tBodyReference.deleteRow(0);
}
```

返回值 无。
参数

index

一个从零开始的整数，依照源代码次序，它对应于集合中的某个特定tr元素（内嵌在当前元素之中）。

insertRow( )

IE *4* NN *n/a* Moz *all* Saf *all* Op *7* DOM *1*

`insertRow(`*index*`)`

向当前tbody、tfoot或thead元素中插入一个内嵌的tr元素。参数传入的整数值指向rows集合中某个从零开始

的位置，以便插入新的行。在IE中也可以简单地使用“-1”作为参数，此时会将新行直接附加到集合的末尾。但此时加入的行只是插入了一个空元素，因此还须要使用insertCell()方法向该元素内增加单元格。然而不幸的是，在IE/Macintosh中使用脚本为表格添加行和单元格并不十分可行，这种方法往往会产生十分硕大的行和单元格尺寸。对于非内嵌表格，可以使用表格分块对象的简便方法来规避文档树节点的创建和插入工作。

返回值 新插入的行对象的引用。

参数

index

从零开始的一个整数，它对应于rows集合中的一个行对象，而新的行将插入到该行之前。

moveRow( )

818

IE *5(Win)* NN *n/a* Moz *n/a* Saf *n/a* Op *n/a* DOM *n/a*

```
moveRow(indexToMove, destinationIndex)
```

将tbody、tfoot或thead元素中的某一行从其原有位置移动到同一表格分块中的另一个行位置。第一个参数指定了待移动的行在rows集合中的索引位置（从零开始）。而第二个参数则表示会将该行移动到哪一行之前。

返回值 已被移动的行节点的引用。

参数

indexToMove

一个从零开始的指向待移行的整数。

destinationIndex

一个从零开始的指向目标位置的整数。

td, th

IE *4* NN *n/a* Moz *all* Saf *all* Op *7* DOM *1*

td和th对象分别对应于td和th元素。从HTML机构的角度来看，这两个元素在表格中拥有不同的设计初衷，但从脚本编程的观点出发，它们拥有完全相同的属性和方法。

尽管表格的单元格元素可能会从其容器继承很多可视属性，如td元素从tbody或tr元素继承了bgColor属性，但并不会将这些继承的属性值自动赋予td对象。因此，一个拥有黄色背景色的单元格并不意味着其bgColor属性也被设置为黄色。

等价HTML元素 <td>、<th>

对象模型引用方式

```
[window.]document.getElementById("elementID")
[window.]document.getElementById("tableRowID").cells[i]
```

对象特定属性 abbr、align、axis、background、bgColor、borderColor、borderColorDark、borderColorLight、cellIndex、ch、chOff、colSpan、headers、height、noWrap、rowSpan、scope、vAlign、width

对象特定方法 无。

对象特定事件 无。

abbr

819

IE *5(Mac)/6(Win)* NN *n/a* Moz *all* Saf *all* Op *7* DOM *1*

可读/可写

它对应于abbr属性。但目前所有的主流浏览器均未实现其功能。

值 字符串。

默认值 空字符串。

align

IE 4 NN n/a Moz all Saf all Op 7 DOM 1

可读/可写

定义了单元格中相关内容的水平对齐方式。

示例 `document.getElementById("myTD").align = "center";`
值 以下3个对齐常量之一：`center | left | right`。
默认值 `left`

axis

IE 5(Mac)/6(Win) NN n/a Moz all Saf all Op 7 DOM 1

可读/可写

它对应于`axis`属性。但目前所有的主流浏览器均未实现其功能。

值 字符串。
默认值 空字符串。

background

IE 4 NN n/a Moz n/a Saf n/a Op n/a DOM n/a

可读/可写

指定单元格背景图片的URL地址。如果同时为元素设置了`bgColor`，那么当图片加载失败时会出现该背景色，否则图片会覆盖背景色。

示例 `document.getElementById("myTD").background = "images/watermark.jpg";`
值 背景图片文件的完整或相对URL。
默认值 无。

bgColor

IE 4 NN n/a Moz all Saf all Op 7 DOM 1

可读/可写

提供表格单元格的背景色。样式单的`backgroundColor`属性并不能反映这个属性设定。即使使用明语颜色名称设置元素或对象的`bgcolor`属性，其返回值仍然是一个十六进制三元组。

示例 `document.getElementById("myTD").bgColor = "yellow";`
值 一个十六进制的三元组或明语颜色名称。目前可用的明语颜色名称请参见附录A。
默认值 由浏览器和操作系统共同决定。

borderColor

IE 4 NN n/a Moz n/a Saf n/a Op n/a DOM n/a

可读/可写

此属性指定了元素边框的颜色。IE会将该颜色直接应用于组成单元格边框的4条线段之上。因此临近的单元格边框颜色并不会互相冲突。

示例 `document.getElementById("myTD").borderColor = "salmon";`
值 一个十六进制的三元组或明语颜色名称。不设置属性值就默认为黑色（`#000000`，`black`）。目前可用的明语颜色名称请参见附录A。
默认值 由操作系统决定。

borderColorDark, borderColorLight

IE 4 NN n/a Moz n/a Saf n/a Op n/a DOM n/a

可读/可写

如果在页面背景或默认颜色四周仔细地勾勒出明暗线条，就可以在IE中创造出3D效果的表格边框。通过为

borderColorDark（单元格的左边缘与上边缘）和borderColorLight（单元格的右边缘与下边缘）属性指定属性值，还可以独立地控制明暗两种线条的颜色。

此时应该为这对属性设置互补的颜色。而对于具体的颜色设置并没有强制性要求，并不一定要为borderColorDark属性设置暗色。这两个属性仅仅是为了使得边框线条轮廓分明，改变它们就可以控制一些边框线条的颜色。

示例

```
document.getElementById("myTD").borderColorDark = "blue";
document.getElementById("myTD").borderColorLight = "cornflowerblue";
```

值 一个十六进制的三元组或明语颜色名称。不设置属性值就默认为黑色（#000000，black）。目前可用的明语颜色名称请参见附录A。

默认值 由操作系统决定。

821

cellIndex

IE *4* NN *n/a* Moz *all* Saf *all* Op *7* DOM *1*

只读

返回一个从零开始的整数，以表示当前单元格在同一行内所有的td元素中所处的位置。这一数值由tr元素内全部td元素的源代码次序决定。

示例 `var whichCell = document.getElementById("myTD").cellIndex;`

值 整数值。

默认值 无。

ch

IE *5(Mac)/6(Win)* NN *n/a* Moz *all* Saf *all* Op *7* DOM *1*

可读/可写

此属性用于指定一个字符，该字符会作为某一单元格中文本对齐的参考点。注意，只有将align属性设置为char，此属性才有价值。但在实际使用中，即使表格的块元素对象拥有这个属性，浏览器也不会对它的属性值作出任何响应。

示例 `document.getElementById("myTD").ch = ".";`

值 单个字符组成的字符串。

默认值 无。

chOff

IE *5(Mac)/6(Win)* NN *n/a* Moz *all* Saf *all* Op *7* DOM *1*

可读/可写

定义一个偏移点，char属性所指定的字符将出现在单元格中偏移点所在位置。但在实际使用中，即使表格的块元素对象拥有这个属性，浏览器也不会对它的属性值作出任何响应。

示例 `document.getElementById("myTD").chOff = "80%";`

值 字符串形式的像素值或单元格宽度的百分比。

默认值 无。

colSpan

IE *4* NN *n/a* Moz *all* Saf *all* Op *7* DOM *1*

可读/可写

此属性决定了表格单元格扩展时所横跨的列数。colSpan的数值每增加一列，则对应的表格行中的td元素就应该减少一个。当align设置为“enter”或“right”时，则对应的单元格将根据其横跨的列数来计算整个td元素的宽度以便进行对齐。除非还为此单元格指定了一个rowspan属性，否则下一行将回复到正常列数。

示例 `document.getElementById("myTD").colSpan = 2;`
值 任意正整数，但通常为2或更大的数值。
默认值 1

headers

IE *5(Mac)/6(Win)* NN *n/a* Moz *all* Saf *all* Op *7* DOM *1*

可读/可写

此属性表示当前单元格所在列的列头单元格元素的ID值。但主流浏览器均未实现该属性的任何功能。
值 字符串形式的ID值。
默认值 无。

height, width

IE *4* NN *n/a* Moz *all* Saf *all* Op *7* DOM *1*

可读/可写

指定元素的高度和宽度，单位为像素。改变这些值后，页面的内容排列也会立刻随之发生变化。
示例 `document.getElementById("myTD").height = "250";`
值 字符串形式的像素值或一个百分比。
默认值 无。

noWrap

IE *4* NN *n/a* Moz *all* Saf *all* Op *7* DOM *1*

可读/可写

指明浏览器是否应该按照须要加宽单元格的宽度，以便将一行不间断的文本显示在单元格内的一整行上。当单元格内内容很多时，如果滥用此属性，那么为了显示所有内容会在页面上出现一个非常不便于使用的横向滚动条。
示例 `document.getElementById("myTD").noWrap = "true";`
值 布尔值：`true | false`。
默认值 `false`

rowSpan

IE *4* NN *n/a* Moz *all* Saf *all* Op *7* DOM *1*

可读/可写

此属性指定了表格单元格扩展时所横跨的列数。`rowSpan`的数值每增加一行，则下一个表格行中的`td`元素就应该减少一个。当`vAlign`设置为“`middle`”时，则对应的单元格将根据其横跨的行数来计算整个`td`元素的高度以便进行对齐。
示例 `document.getElementById("myTD").rowSpan = 12;`
值 任意正整数，但通常为2或更大的数值。
默认值 1

scope

IE *5(Mac)/6(Win)* NN *n/a* Moz *all* Saf *all* Op *7* DOM *1*

可读/可写

此属性对应于表格单元格元素的作用域属性。但主流浏览器实际上均未实现该属性的任何功能。
值 以下常量字符串之一：`cols | colgroup | rows | rowgroup`。
默认值 无。

vAlign

IE 4 NN n/a Moz all Saf all Op 7 DOM 1

可读/可写

此属性指定元素内容容器盒中所有文本的竖直对齐方式。

示例 `document.getElementById("myTD").vAlign = "baseline";`

值 不区分大小写的字符串常量：`baseline | bottom | middle | top`。

默认值 `middle`

width

请参见height。

text

IE 4 NN n/a Moz all Saf all Op 7 DOM 1

请参见input。

Text, TextNode

IE 5 NN n/a Moz all Saf all Op 7 DOM 1

在本书的很多位置都将Text对象称之为“文本节点”。而微软将这个对象称之为TextNode对象。这个对象所代表的子节点对象包含了元素起始和结束标签之间的文本字符。由于Text对象与基础的Node对象之间存在着一条遗传链（Node-CharacterData-Text），因此这个对象也是抽象W3C DOM模型中的一员。其父级Node对象自动为Text对象赋予了一长串属性和方法，在本章开始部分的共享条目里已对它们进行了说明。沿着这条继承链，Text对象还获得了一些附加的属性和方法，通过这些正式W3C DOM模型的结构化指令就可以操纵节点的内容。由于DOM独立于脚本编程语言，使用JavaScript这种脚本语言后，就会发现通过它来操纵这些属性和方法更为简单而有效。相关信息请参考第5章。在浏览器的客户端JavaScript环境中可以毫无顾虑地使用这些技术。

此外，只有使用文档树中的节点引用或通过`document.createTextNode()`的返回值才能让脚本涉及Text节点或IE中的TextNode对象。

对象模型引用方式

`elementReference.childReference`
`textNodeReference.siblingReference`

对象特定属性 `data`、`length`

对象特定方法 `appendData()`、`deleteData()`、`insertData()`、`replaceData()`、`splitText()`、`substringData()`

对象特定事件 无。

data

IE 5 NN n/a Moz all Saf all Op 7 DOM 1

可读/可写

此属性包含文本节点中的所有字符内容。其属性值与nodeValue属性的值相同，除了它的名称比较便于理解代码，这两个属性之间完全没有任何差异。

示例 `document.getElementById("myP").firstSibling.data = "Some new text.";`

值 字符串。

默认值 空字符串。

length

IE 5 NN n/a Moz all Saf all Op 7 DOM 1

只读

此属性提供文本节点的字符的数量。

示例 `var howMany = document.getElementById("myP").firstSibling.length;`
值 整数值。
默认值 0

appendData( )

IE *5(Mac)/6(Win)* NN *n/a* Moz *all* Saf *all* Op *7* DOM *1*

```
appendData("newText")
```

此方法将参数传入的字符串中的字符附加到当前文本节点末尾。由于节点中的内容不会对字符进行处理，因此如果要向节点中加入一个句子，须要让脚本控制句子间的间隔。

返回值 无。

参数

newText
将要附加到节点中的文本字符串。如果是一个字符串引用，如文档中另一个文本节点的data属性，那么就会将其引用的内容复制到附加位置。

deleteData( )

IE *5(Mac)/6(Win)* NN *n/a* Moz *all* Saf *all* Op *7* DOM *1*

```
deleteData(startOffset,count)
```

在当前文本节点的内容中，此方法可以从*startOffset*参数指定的位置开始顺着当前语言的正常文本方向删除*count*个字符。如果从删除位置开始，当前节点中剩余文本数据长度比指定的长度短，那么会直接删除节点末尾的所有字符并且不抛出任何异常。早期的Mozilla版本的浏览器在计算字符数时还会将源代码中的空格考虑在内。

返回值 无。

参数

startOffset
正整数，指定了删除时的字符起始位置。

count
正整数，指定了要删除的字符个数。

insertData( )

IE *5(Mac)/6(Win)* NN *n/a* Moz *all* Saf *all* Op *7* DOM *1*

```
insertData(startOffset, "newText")
```

此方法会在文本节点内某个字符位置插入一段新文本。

返回值 无。

参数

startOffset
正整数，指定了新文本将要插入的字符位置。

newText
将要插入到节点中的文本字符串。如果是一个字符串引用，如文档中另一个文本节点的data属性，那么就会将其引用的内容复制到插入位置。

replaceData( )

IE *5(Mac)/6(Win)* NN *n/a* Moz *all* Saf *all* Op *7* DOM *1*

```
replaceData(startOffset, count, "newText")
```

使用新文本替换当前文本节点中的文本。根据参数指定的起始位置和字符数，此方法会首先删除原有内容。然后将第3个参数传入的字符串放入这些字符所腾出的空间。须要注意的是，IE 5/Macintosh中的一个错误会将新文本剪切到与已删除的文本相同的长度。

返回值 无。

参数

startOffset

正整数，指定了删除时的字符起始位置。

count

正整数，指定了要删除的字符个数。

newText

将要插入到节点中的文本字符串。如果是一个字符串引用，如文档中另一个文本节点的data属性，那么就会将其引用的内容复制到附加位置。

splitText( )

IE *5(Mac)/6(Win)* NN *n/a* Moz *all* Saf *all* Op *7* DOM *1*

```
splitText(offset)
```

此方法会将当前文本节点分割为两个兄弟文本节点。

返回值 第2个文本节点的引用。

参数

offset

正整数，指定了文本分割所在的位置。

substringData( )

IE *5(Mac)/6(Win)* NN *n/a* Moz *all* Saf *all* Op *7* DOM *1*

```
substringData(startOffset, count)
```

从文本节点内容中返回一段指定的内容。根据参数指定的起始位置和字符数，此方法会复制指定的文本内容。

返回值 字符串。

参数

startOffset

正整数，指定了复制动作的字符起始位置。

count

正整数，指定了要复制的字符个数。

textarea

IE *4* NN *n/a* Moz *all* Saf *all* Op *7* DOM *1*

textarea对应于textarea元素，它是一种表单控件。这个对象是获取用户的多行文本输入并提交至服务器的主要手段。须要注意的是，在IE 4/Macintosh中无法使用其innerHTML属性。由于IE 4和Netscape 6之前的一些早期浏览器并不支持HTML元素寻址，这使得下文中列出的属性和方法只有很少的一部分能够应用于这些浏览器。而IE 5及后续版本均支持共享的doScroll()方法。

等价HTML元素 <textarea>

对象模型引用方式

```
[window.]document.formName.elementName
[window.]document.forms[i].elements[j]
[window.]document.getElementById("elementID")
```

对象特定属性 accept、autofocus、cols、dataFld、dataSrc、defaultValue、form、forms[]、inputmode、labels[]、maxlength、name、pattern、readOnly、required、rows、selectionEnd、selectionStart、status、textLength、type、validationMessage、validity、value、willValidate、wrap

对象特定方法 checkValidity()、createTextRange()、dispatchChange()、dispatchFormChange()、handleEvent()、select()、setCustomValidity()

对象特定事件

事件	IE	Mozilla	Safari	Opera	W3C DOM
change	•	•	—	•	•
scroll	•	—	—	—	—
select	•	•	—	•	•

accept

IE *n/a* NN *n/a* Moz *n/a* Saf *n/a* Op *9* DOM *n/a*

可读/可写

这个Web Forms 2.0扩展属性能够设定哪种或哪几种MIME类型的数据可以输入到textarea元素之中。除了简单文本之外，如果浏览器还为其他类型的内容提供了输入编辑环境，那么textarea元素可以通过此属性为这些内容做好准备，并且在提交表单时对它们进行编码。

示例 `document.forms[0].comments.accept = "message/news";`

值 MIME类型字符串。

默认值 由浏览器和操作系统共同决定。

autofocus

IE *n/a* NN *n/a* Moz *n/a* Saf *n/a* Op *9* DOM *n/a*

可读/可写

当页面加载完成后，这个Web Forms 2.0扩展会将焦点聚集在元素上。每个页面上只能有一个表单控件元素拥有此属性。

值 布尔值：`true | false`。

默认值 `false`

cols

IE *4* NN *n/a* Moz *all* Saf *all* Op *7* DOM *1*

可读/可写

此属性规定了textarea元素内文本编辑域的宽度。其属性值表示在文本编辑域的宽度内可显示的单宽度字符的数量。如果字体尺寸受样式单的影响，那么实际宽度也会随之而变。

示例 `document.forms[0].comments.cols = 60;`

值 整数值。

默认值 `20`（IE/Windows -1）、`-1`（Mozilla）或`0`（Safari/Windows）。

dataFld

IE *4* NN *n/a* Moz *n/a* Saf *n/a* Op *n/a* DOM *n/a*

可读/可写

此属性用于IE中的数据绑定，以便将一个远程数据源的列名与textarea对象的value属性联系在一起。此时元素中还必须设定datasrc属性。如果dataFld和dataSrc属性均被设置为空字符串，那么就会打破元素与数据源之间的绑定关系。注意，只能在IE 5/Mac操作系统下实现文本文件数据源的绑定。

示例 `document.myForm.myTextArea.dataFld = "description";`

值 数据源数据列内区分大小写的标识符。

默认值 无。

dataSrc

IE *4* NN *n/a* Moz *n/a* Saf *n/a* Op *n/a* DOM *n/a*

可读/可写

为实现IE数据绑定功能，此属性用于指定页面内object元素的ID值，以便在进行远程数据访问时加载数据源

对象。通过datafld属性指定来自于数据源中的绑定内容。如果dataFld和dataSrc属性均被设置为空字符串，那么就会打破元素与数据源之间的绑定关系。注意，只能在IE 5/Mac操作系统下实现文本文件数据源的绑定。

示例 `document.myForm.myTextArea.dataSrc = "DBSRC3";`

值 数据源内区分大小写的标识符。

默认值 无。

829

defaultValue

IE 3 NN 2 Moz all Saf all Op 7 DOM 1

只读

根据页面源代码中元素的起始标签和结束标签之间的内容，此属性返回textarea元素的默认文本。

示例

```
var txtAObj = document.forms[0].myTextArea;
if (txtAObj.value != txtAObj.defaultValue ) {
    ...
}
```

值 任何字符串。

默认值 无。

form

IE 3 NN 2 Moz all Saf all Op 7 DOM 1

只读

返回包含当前元素的form元素的引用。在处理一个来自于该元素的事件时，对应的事件处理方法可以通过事件对象的target或srcElement属性来自动访问对应的select元素。而通过读取form属性，脚本可以轻易地访问同一个表单中的其他控件。

示例 `var theForm = evt.srcElement.form;`

值 form元素对象的引用。

默认值 无。

forms

IE n/a NN n/a Moz n/a Saf n/a Op 9 DOM n/a

只读

这个Web Forms 2.0扩展返回一个与当前textarea元素相关联的form对象引用组成的数组（NodeList）。

示例 `var formList = document.getElementById("myTextarea").forms;`

值 数组。

默认值 包含一个封闭表单元素的引用。

inputmode

IE n/a NN n/a Moz n/a Saf n/a Op 9 DOM n/a

可读/可写

这个Web Forms 2.0 扩展会引导浏览器为某种书面语言显示合适的文本输入用户界面。它完全继承自W3C XForms 1.0规范，请访问*http://www.w3.org/TR/xforms/sliceE.html*。请查阅W3C XForms 1.0文档，以获取更多详细信息。

示例 `document.orderForm.searchText.inputmode = "hiragana";`

830

值 每种书面语言均伴随着一个可选择的修订标记。这些修订标记通常相当于Unicode脚本，请参见*http://www.unicode.org/unicode/reports/tr24/*。

默认值 无。

labels

IE *n/a* NN *n/a* Moz *n/a* Saf *n/a* Op *9* DOM *n/a*

只读

这个Web Forms 2.0扩展返回一个与当前表单控件元素相关联的label元素引用组成的数组（HTMLCollection）。

示例 `var textboxLabels = document.getElementById("myTextarea").labels;`

值 label元素对象引用组成的数组。

默认值 空数组。

maxlength

IE *n/a* NN *n/a* Moz *n/a* Saf *n/a* Op *9* DOM *n/a*

可读/可写

这个Web Forms 2.0扩展定义了textarea元素中可输入的最大字符数。实际使用时，当用户输入的字符数量超过maxlength定义的值时，浏览器会发出蜂鸣声或给出其他提示。注意，maxlength和size属性之间并没有任何内在的联系。如果maxlength允许的最大字符数超过了元素宽度可显示的字符数，那么浏览器会在元素中使用水平滚动（尽管对很多用户而言这显得比较笨拙），以便输入或修改文本内容。

示例 `document.getElementById("query").maxlength=10;`

值 正整数。

默认值 无限制。

name

IE *3* NN *2* Moz *all* Saf *all* Op *7* DOM *1*

可读/可写

这是一个与表单控件相关的标识符。当表单被提交时，此属性的值将作为名称/值对的一部分而被提交至服务器。由于根据控件类型可以通过其他手段指定控件标注，因此从用户的角度来看，控件名称是被隐藏的。与此同时，脚本也可以使用控件名称来引用对象。虽然新标准倾向于使用id属性，但很多浏览器依然须要为表单控件设置name属性，以便允许表单控件的值被正常提交。

示例 `document.orderForm.myTextArea.name = "customerComment";`

值 区分大小写的标识符，必须遵循标识符命名规则：不包含空白、不能以数字作为起始字符、除下划线外不允许使用其他标点符号。

默认值 无。

pattern

IE *n/a* NN *n/a* Moz *n/a* Saf *n/a* Op *9* DOM *n/a*

可读/可写

如果必须验证用户输入，那么通过这个Web Forms 2.0扩展就可以指定一个正则表达式来进行判断。

示例 `document.getElementById("partNum").pattern = "[A-Z][0-9]{7}"`

值 正则表达式（与JavaScript中的正则表达式相同，不能放置在斜线符号中）。

默认值 无。

readOnly

IE *4* NN *n/a* Moz *all* Saf *all* Op *7* DOM *1*

可读/可写

指明用户能否在页面上编辑表单元素。如果一个表单控件的readOnly属性被设置为true，那么脚本仍然可以修改其内容，但用户也许不能再做出任何改动。

示例 `document.forms[0].myTextArea.readOnly = "true";`

值 布尔值：true | false。

默认值 false

required

IE *n/a* NN *n/a* Moz *n/a* Saf *n/a* Op *9* DOM *n/a*

可读/可写

这个Web Forms 2.0 扩展表示表单提交时该textarea元素的值是否必须存在。如果元素未接收到任何值，那么就会将ValidityState对象的missingValue属性值设置为true。

值 布尔值：true | false。

默认值 false

rows

IE *4* NN *n/a* Moz *all* Saf *all* Op *7* DOM *1*

可读/可写

根据此属性指定的文本行数量可以设定元素textarea的高度，以便不使用滚动条就能完全显示这些文本。此属性值表示在滚动条出现之前，编辑域内所能显示的单间隔字符行的数量。如果字体尺寸受样式单的影响，那么实际高度也会随之而变。

示例

```
document.forms[0].comments.rows = 6;
```

值 整数值。

默认值 2(IE/Windows -1)、-1 (Mozilla)或0 (Safari/Windows) 。

selectionEnd, selectionStart

IE *n/a* NN *n/a* Moz *all* Saf *1.3/2* Op *8* DOM *n/a*

可读/可写

这两个属性使得脚本能够在文本相关的输入元素内方便地获取和设置文本选择的端点。它们的值均是从零开始的整数，对应于文本域中已输入的某个文本字符的位置。当这两个属性拥有相同的值时，其效果看起来就像一个文本插入点。例如，如果要将光标放在文本输入框的末尾，那么可以将这个两个值都设置为元素内文本的长度值，请参见textLength属性。在IE浏览器中，首先须要在元素内创建一个IE文本域，然后调整这个域的端点进而选择这个文本域才能达到相同的效果，请参见TextRange对象。

示例

```
var elem = document.forms[0].myTextarea;
elem.selectionEnd = elem.textLength;
elem.selectionStart = elem.textLength;
```

值 正整数。

默认值 无。

status

IE *4* NN *n/a* Moz *n/a* Saf *n/a* Op *n/a* DOM *n/a*

可读/可写

尽管IE中实现了这个属性，但它对textarea对象毫无作用。

值 布尔值：true | false；或者null。

默认值 null

textLength

IE *n/a* NN *n/a* Moz *1.4* Saf *n/a* Op *8* DOM *n/a*

只读

返回textarea元素中字符的数量。

值 整数值。

默认值 0

type

IE *4* NN *3* Moz *all* Saf *all* Op *7* DOM *1*

只读

返回表单控件元素的类型。返回值中的字符均为小写形式。有时可能须要遍历所有的表单元素，以便找到特定的类型执行某些操作，例如，在将所有textarea类型的表单控件清空的同时不改变其他控件。

示例

```
if (document.forms[0].elements[3].type == "textarea") {
  ...
}
```

值 下列字符串常量之一：button | checkbox | file | hidden | image | password | radio | reset | select-multiple | select-one | submit | text | textarea。

默认值 textarea

validationMessage

IE *n/a* NN *n/a* Moz *n/a* Saf *n/a* Op *9* DOM *n/a*

只读如果表单控件在Web Forms 2.0规范下校验失败，那么这个Web Forms 2.0的扩展属性会返回一个由浏览器生成的消息。如果返回空字符串就表示校验正常。

值 字符串。

默认值 空字符串。

validity

IE *n/a* NN *n/a* Moz *n/a* Saf *n/a* Op *9* DOM *n/a*

只读

这个Web Forms 2.0扩展返回一个ValidityState对象。请参见ValidityState对象。

值 ValidityState对象。

默认值 ValidityState对象。

value

IE *3* NN *2* Moz *all* Saf *all* Op *7* DOM *1*

可读/可写

指明与表单控件相关的当前值，该值将与元素的名称/值对一起被提交至服务器端。这些值均为字符串形式。

示例

```
var comment = document.forms[0].myTextArea.value;
```

值 字符串。

默认值 无。

willValidate

IE *n/a* NN *n/a* Moz *n/a* Saf *n/a* Op *9* DOM *n/a*

只读

这个Web Forms 2.0扩展返回一个布尔值，以表明表单控件是否能够达到Web Forms 2.0机制下的校验标准。

值 布尔值：true | false。

默认值 false

wrap

IE *4* NN *n/a* Moz *n/a* Saf *n/a* Op *9* DOM *n/a*

可读/可写

wrap属性会通知浏览器是否对元素内的文本进行换行，以及当软换行转换为硬换行时，这些文本是否应该被提交至服务器端。属性值为hard时会关闭自动换行，并将文本中的软换行转换为CR-LF（回车换行）符。而值为soft时会启用自动换行，但在提交至服务器的文本中并不会包含CR-LF符号。off意味着关闭自动换行。

示例 `document.forms[0].comments.wrap = "soft";`
值 下面3个字符串常量之一：`hard | off | soft`。
默认值 `soft`

checkValidity( )

IE *n/a* NN *n/a* Moz *n/a* Saf *n/a* Op *9* DOM *n/a*

这个Web Forms 2.0方法会返回一个布尔值，以表明表单控件是否能够达到其校验标准，归根到底，就是判断`validity.valid`属性是否为`true`。
返回值 布尔值：`true | false`。
参数 无。

createTextRange( )

IE *4(Win)* NN *n/a* Moz *n/a* Saf *n/a* Op *n/a* DOM *n/a*

此方法可以根据`textarea`对象的内容创建一个`TextRange`对象。请参见`TextRange`对象以获取更多详细信息。
返回值 `TextRange`对象。

dispatchChange( ), dispatchFormChange( )

IE *n/a* NN *n/a* Moz *n/a* Saf *n/a* Op *9* DOM *n/a*

这两个Web Forms 2.0方法为当前元素触发了`change`和`formchange`事件。因此让`onclick`事件处理程序调用这两个方法之一，就可以将一个`click`事件转换成为一个`change`或`formchange`事件。
返回值 无。
参数 无。

handleEvent( )

IE *n/a* NN *|4|* Moz *n/a* Saf *n/a* Op *n/a* DOM *n/a*

`handleEvent(event)`

命令对象接受并处理一个事件，该事件的描述将作为参数传入此方法。此时`document`对象必须拥有一个事件处理程序来处理这种类型的事件。但这个方法仅适用于Navigator 4。
返回值 无。
参数
event
一个Navigator 4的`event`对象。

select( )

IE *3* NN *2* Moz *all* Saf *all* Op *7* DOM *1*

选择所有显示在表单元素中的文本。在IE浏览器中，如果要在`textarea`元素内指定一个插入点，请参见`TextRange`对象。
返回值 无。
参数 无。

setCustomValidity( )

IE *n/a* NN *n/a* Moz *n/a* Saf *n/a* Op *9* DOM *n/a*

`setCustomValidity([errorString])`

这个Web Forms 2.0方法可以设置`validity`属性（它本身就是一个`ValidityState`对象）的`customError`布尔值。如果这个方法不支持某个元素类型，那么它将抛出一个NOT_SUPPORTED_ERR错误。
返回值 无。
参数
errorString
如果此参数为`null`或空字符串，将重新设置`validity`对象的`customError`属性，即表示表单控件无效。在当前会话期内，由于浏览器会记录一条错误信息，因此验证失败后会显示这条信息。

TextNode

请参见Text。

TextRange

IE *4(Win)* NN *n/a* Moz *n/a* Saf *n/a* Op *n/a* DOM *n/a*

TextRange对象在概念上与W3C DOM的Range对象很类似，它代表了文档中一串已显示的文本字符。当文本域中的字符数为零时，它就退化成为了一个字符插入点。

通过body、button、text或textarea对象的createTextRange()方法就可以创建一个TextRange对象。而selection对象的createRange()方法也可以将用户选中的内容转换成一个选择域。使用时请注意这两个方法在名称上的细微差别。一旦创建了文本选择域，那么就可以使用该对象的方法来调整其起点和终点以便包含目标文本分段。将目标文本置于文本域后，就可以对文本域对象的text进行赋值操作，以便修改、删除或插入文本。也可以使用pasteHTML()来向文本域中插入HTML内容。此外，还可调用那些能够执行文本修改的直接命令来处理文本域。关于TextRange对象的详细信息及使用范例请参见在线参考V。

836

在本章开始部分所列出的共享属性和方法中，有如下几个可以在这个对象内使用：offsetLeft、offsetTop、getBoundingClientRect()、getClientRects()和scrollIntoView()。须要注意的是，只能在Windows版本的IE浏览器中使用TextRange对象及其有关的功能。

对象模型引用方式

objectRef.createTextRange()
selectionObjectRef.createRange()

对象特定属性 boundingHeight、boundingLeft、boundingTop、boundingWidth、htmlText、text

对象特定方法 collapse()、compareEndPoints()、duplicate()、execCommand()、expand()、findText()、getBookmark()、inRange()、isEqual()、move()、moveEnd()、moveStart()、moveToBookmark()、moveToElementText()、moveToPoint()、parentElement()、pasteHTML()、queryCommandEnabled()、queryCommandIndeterm()、queryCommandState()、queryCommandSupported()、queryCommandText()、queryCommandValue()、select()、setEndPoint()

boundingHeight, boundingWidth

IE *4(Win)* NN *n/a* Moz *n/a* Saf *n/a* Op *n/a* DOM *n/a*

只读

这两个属性分别返回TextRange对象所占据的虚拟空间的高度和宽度，单位为像素。尽管并不能在文档中观察到一个TextRange对象，但它所占据的区域等价于高亮显示的选中内容所占的空间大小。这两个值也只是控制着选中域的最大高度和宽度。通过getBoundingClientRect()方法的返回值也可以计算出这两个属性的属性值。

示例
```
var rangeWidth = document.forms[0].myTextArea.createTextRange().boundingWidth;
```
值 整数值。

默认值 无。

837

boundingLeft, boundingTop

IE *4(Win)* NN *n/a* Moz *n/a* Saf *n/a* Op *n/a* DOM *n/a*

只读

这两个属性分别表示TextRange对象的虚拟空间的左、上边缘与浏览器窗口或框架的左、上边缘之间的距离。尽管并不能在文档中观察到一个TextRange对象，但它所占据的区域等价于高亮显示的选中内容所占的空间大小。值得注意的是，测量这两个值的参照对象是固定的窗口或框架的边缘，而不是选择域所在的文档。因此用户一旦滚动当前文档，那么这些值就会发生变化。

示例 `var rangeOffH = document.forms[0].myTextArea.createTextRange().boundingLeft;`
值 整数值。
默认值 无。

htmlText

IE *4(Win)* NN *n/a* Moz *n/a* Saf *n/a* Op *n/a* DOM *n/a*

只读

当某个元素是TextRange对象的基础元素时，这个属性可以说明该元素在文档中的所有HTML内容。例如，如果通过document.body.createTextRange()方法为body元素创建了一个TextRange对象，那么htmlText属性就能包含该body元素标签之间的所有HTML内容。

示例 `var rangeHTML = document.body.createTextRange().htmlText;`
值 字符串。
默认值 无。

text

IE *4(Win)* NN *n/a* Moz *n/a* Saf *n/a* Op *n/a* DOM *n/a*

可读/可写

此属性指明文本域中包含的文本内容。如果使用body元素创建了一个TextRange对象，那么这个属性只能拥有元素中已显示的文本，而不能包含任何的HTML标签内容。

示例 `var rangeText = document.body.createTextRange().text;`
值 字符串。
默认值 无。

collapse( )

IE *4(Win)* NN *n/a* Moz *n/a* Saf *n/a* Op *n/a* DOM *n/a*

838

```
collapse([start])
```

将TextRange对象缩小到一个插入点，并决定该插入点的具体位置。

返回值 无。
参数

start

可选的布尔值，true表示插入点将位于原选择域的起点，而false则表示终点。默认值为true。

compareEndPoints( )

IE *4(Win)* NN *n/a* Moz *n/a* Saf *n/a* Op *n/a* DOM *n/a*

```
compareEndPoints("type", comparisonRange)
```

这个方法将当前域的端点与参数传入的域的端点进行比较。方法的第一个参数决定了这两个域分别使用哪个端点进行比较。如果域中的第一个端点位于另一个端点之前，那么返回值为-1。如果比较后发现两个点均来自于同一个位置，那么返回值为0。当第一个端点位于另一个端点之后时，则返回1。例如，如果在变量r1中保存第一个域，然后创建一个新域r2，那么就可以通过比较r2的终点和r1的起点来获得它们之间的相对关系，代码如下：

```
r1.compareEndPoints("EndToStart", r2)
```

如果r1的终点和r2的起点重合，那么返回值就为0。

返回值 -1、0或1。
参数

type

下列字符串常量之一：StartToEnd | StartToStart | EndToStart | EndToEnd。

comparisonRange

已被创建并保存在变量中的一个TextRange对象。

duplicate( )

IE 4(Win) NN n/a Moz n/a Saf n/a Op n/a DOM n/a

这个方法可以根据当前域的相关参数创建一个新的TextRange对象。新创建的对象是一个独立的对象，即新旧两个对象并不相等，但在初始情况下它们的值完全相等。

返回值 TextRange对象。

参数 无。

execCommand( )

IE 4(Win) NN n/a Moz n/a Saf n/a Op n/a DOM n/a

839

```
execCommand("commandName"[, UIFlag[, value]])
```

在当前TextRange对象上执行已命名的命令。当TextRange对象是一个插入点时，大多数命令的工作情况最好。请参考附录D以获取命令列表。

返回值 布尔值：如果命令执行成功，返回true，否则返回false。

参数

commandName

命令名称的字符串，不须要区分大小写。请参见附录D。

UIFlag

可选用的布尔值，包括：true，显示被命令触发的任何用户界面，false，则会防止显示这些界面。

value

命令所需的参数值。

expand( )

IE 4(Win) NN n/a Moz n/a Saf n/a Op n/a DOM n/a

```
expand("unit")
```

扩展当前文本域以便涵盖由参数传入的文本单元。例如，若某位用户从文档中选择了一些字符，那么就可以创建一个文本域并涵盖选中位置的整个句子，相关代码如下：

```
var rng = document.selection.createRange();
rng.expand("sentence");
```

如果文本域的起点在延伸时须要穿越多个文本单元，那么expand()方法会将该文本域向下一个最靠近的单元进行扩展。

返回值 布尔值：如果此方法执行成功，返回true，否则返回false。

参数

unit

目标单元的字符串说明，不须要区分大小写：character | word | sentence | textedit。textedit会让当前域扩展到整个文本区域。

findText( )

IE 4(Win) NN n/a Moz n/a Saf n/a Op n/a DOM n/a

```
findText("string"[, searchScope][, flags])
```

根据第1个参数传入的字符串，从当前TextRange对象中找到与之匹配的内容。在默认情况下，进行字符串匹配时并不区分字符的大小写状态。如果存在匹配的内容，那么TextRange对象会将文本域的起始点和结束点放置在已找到的文本内容两端。如果还须要在文档中继续进行搜索，必须首先将文本域的起点放置到已找到的字符串的末尾，即调用collapse()方法。

后面的两个可选参数可以用来在文本域内限制搜索范围，或者增加一些匹配要求，如某个查询词等。

840

返回值 布尔值：如果找到匹配项，返回true，否则返回false。

参数

string

查询字符串，默认情况下不用区分字符大小写。

searchScope

相对于文本域起点的搜索字符数。正数表示向前搜索，而负数则向后搜索，一直搜索到文档内文本域的起点位置。

flags

表示搜索细节的整数代码：0表示复合单词部分匹配，1表示向后匹配，2表示必须匹配整个词，而4则表示区分字符大小写。

getBookmark( ), moveToBookmark( )

IE *4(Win)* NN *n/a* Moz *n/a* Saf *n/a* Op *n/a* DOM *n/a*

```
getBookmark(), moveToBookmark(bookmarkString)
```

这两个方法通常彼此协同工作，它们可以提供一种临时保存文本域并进行恢复的途径。getBookmark()方法会返回一个包含二进制数据的字符串。一旦将返回值保存在变量中之后，就可以根据需要通过脚本修改文本域。在这些操作之后，这个已保存的文本域还可以通过moveToBookmark()方法重新得以恢复，相关代码如下：

```
var rangeMark = myRange.getBookmark();
...
myRange.moveToBookmark(rangeMark);
```

返回值 布尔值：如果操作成功，返回true，否则返回false。

参数

bookmarkString

由getBookmark()方法返回的一个包含二进制数据的字符串。

inRange( )

IE *4(Win)* NN *n/a* Moz *n/a* Saf *n/a* Op *n/a* DOM *n/a*

```
inRange(comparisonRange)
```

这个方法可以判定参数传入的文本域是位于当前文本域之内还是与它相等。

返回值 布尔值：如果用于比较的文本域在当前域之内或与之相等，则返回true，否则返回false。

参数

comparisonRange

已被创建并保存在变量中的一个TextRange对象。

isEqual( )

IE *4(Win)* NN *n/a* Moz *n/a* Saf *n/a* Op *n/a* DOM *n/a*

```
isEqual(comparisonRange)
```

这个方法可以判定参数传入的文本域是否与当前文本域相等。

返回值 布尔值：如果用于比较的文本域与当前域相等，则返回true，否则返回false。

参数

comparisonRange

已被创建并保存在变量中的一个TextRange对象。

move( )

IE *4(Win)* NN *n/a* Moz *n/a* Saf *n/a* Op *n/a* DOM *n/a*

```
move("unit"[, count])
```

这个方法会将当前文本域缩小为一个插入点，并将它从其终点位置向前或向后移动一个或多个单元距离。

返回值 移动时跨越的单元数。

参数

unit

目标单元的字符串说明，不须要区分大小写：character、word、sentence、textedit。textedit会将插入点指向整个原始域的起点或终点。

count

一个可选的整数，用以表示插入点在移动时须要跨越的单元数。正数表示向前移动，而负值则表示向后移动。默认值为1。

moveEnd( ), moveStart( )

IE 4(Win) NN n/a Moz n/a Saf n/a Op n/a DOM n/a

```
moveEnd("unit"[, count]) moveStart("unit"[, count])
```

这两个方法会将当前域的终点和起点分别移动一个或多个单元。第二个可选参数可以指定移动的方向和单元数。将这个参数设为负值时，会让文本域的起点向着原始域的起点移动。如果要将终点向后移动若干个“word”单元，那么要注意应该以空格字符结束这个字。因此，如果findText()方法在文本域中找寻一段不以空格结尾的字符串，那么第一个moveEnd("word")方法会将终点放置在空格之后，而不是紧跟在找到的内容之后。

返回值 移动时所跨越的单元数。

参数

unit

目标单元的字符串说明，不须要区分大小写：character | word | sentence | textedit。textedit会将插入点指向整个原始域的起点或终点。

count

一个可选的整数，用以表示插入点在移动时须要跨越的单元数。正数表示向前移动，而负值则表示向后移动。默认值为1。

moveToBookmark( )

请参见getBookmark()。

moveToElementText( )

IE 4(Win) NN n/a Moz n/a Saf n/a Op n/a DOM n/a

```
moveToElementText(elementObject)
```

移动当前TextRange对象的起点和终点，从而让该文本域涵盖指定的HTML元素对象。最终得到的文本域中会包含元素的HTML内容。

返回值 无。

参数

elementObject

一个对象的脚本引用。它既可以是一个直接引用，即document.getElementById("*elementID*")，也可以是一个包含相同值的变量。

moveToPoint( )

IE 4(Win) NN n/a Moz n/a Saf n/a Op n/a DOM n/a

```
moveToPoint(x, y)
```

将文本域缩小为一个插入点，并根据参数传入的水平和竖直坐标将该点放置到指定的位置。当用户在浏览器窗口中点击一个点时可以通过这个方法来定义一个插入点。然后就可以使用诸如expand()的方法来扩大文本域，并包含字符、字句或整个文本域。

返回值 无。

参数

x

插入点相对于窗口或框架左边缘的水平坐标位置，单位为像素。

y

插入点相对于窗口或框架上边缘的竖直坐标位置，单位为像素。

parentElement( )

IE *4(Win)* NN *n/a* Moz *n/a* Saf *n/a* Op *n/a* DOM *n/a*

返回完全包含当前TextRange对象的临近最外层元素对象的引用。

返回值 元素对象的引用。

参数 无。

843

pasteHTML( )

IE *4(Win)* NN *n/a* Moz *n/a* Saf *n/a* Op *n/a* DOM *n/a*

```
pasteHTML("HTMLText")
```

将当前文本域中的内容替换为参数字符串传入的HTML内容。通常会在一个文本长度为零的文本域（即插入点）上使用这个方法。如果标签是原始代码的一部分，那么这些标签也会被显示出来。

返回值 无。

参数

HTMLText

将要被插入到文档的文档源代码。

queryCommandEnabled( )

IE *4(Win)* NN *n/a* Moz *n/a* Saf *n/a* Op *n/a* DOM *n/a*

```
queryCommandEnabled("commandName")
```

根据文档或选中内容的当前状态指出能否调用参数所指定的命令。

返回值 布尔值：如果可以调用，则返回true，否则返回false。

参数

commandName

命令名称的字符串，不须要区分大小写。请参见附录D。

queryCommandIndeterm( )

IE *4(Win)* NN *n/a* Moz *n/a* Saf *n/a* Op *n/a* DOM *n/a*

```
queryCommandIndeterm("commandName")
```

指明命令是否处于一个不确定的状态。

返回值 布尔值：true | false。

参数

commandName

命令名称的字符串，不须要区分大小写。请参见附录D。

queryCommandState( )

IE *4(Win)* NN *n/a* Moz *n/a* Saf *n/a* Op *n/a* DOM *n/a*

```
queryCommandState("commandName")
```

决定已命名命令的当前状态。

返回值 如果命令已完成，返回true；如果未完成，返回false；如果无法准确判断其状态，则返回null。

参数

commandName

命令名称的字符串，不须要区分大小写。请参见附录D。

844

queryCommandSupported( )

IE *4(Win)* NN *n/a* Moz *n/a* Saf *n/a* Op *n/a* DOM *n/a*

```
queryCommandSupported("commandName")
```

决定文档对象是否支持对应的命令。

返回值 布尔值：true | false。

参数

commandName

命令名称的字符串，不须要区分大小写。请参见附录D。

queryCommandText( )

IE *4(Win)* NN *n/a* Moz *n/a* Saf *n/a* Op *n/a* DOM *n/a*

```
queryCommandText("commandName")
```

返回与命令相关的文本。

返回值 字符串。

参数

commandName

命令名称的字符串，不须要区分大小写。请参见附录D。

queryCommandValue( )

IE *4(Win)* NN *n/a* Moz *n/a* Saf *n/a* Op *n/a* DOM *n/a*

```
queryCommandValue("commandName")
```

返回与命令相关的值，例如，选中部分的字体名称。

返回值 由具体命令决定。

参数

commandName

命令名称的字符串，不须要区分大小写。请参见附录D。

select( )

IE *4(Win)* NN *n/a* Moz *n/a* Saf *n/a* Op *n/a* DOM *n/a*

选择当前TextRange对象中的所有文本。这个方法会带给用户一些视觉上的效果，以表明脚本已经掌握了一段特殊的文本内容。例如，如果使用脚本调用findText()方法来进行文本查询，完成查询后就可以在结果域上使用scrollIntoView()和select()方法，以突出显示匹配的文本内容。

返回值 无。

参数 无。

setEndPoint( )

IE *4(Win)* NN *n/a* Moz *n/a* Saf *n/a* Op *n/a* DOM *n/a*

```
setEndPoint("type", comparisonRange)
```

将当前TextRange对象的终点放置到参数传入的另一个文本域的终点位置。

返回值 无。

参数

type

下列字符串常量之一：StartToEnd | StartToStart | EndToStart | EndToEnd。

comparisonRange

已被创建并保存在变量中的一个TextRange对象。

TextRectangle

IE *5(Win)* NN *n/a* Moz *n/a* Saf *n/a* Op *n/a* DOM *n/a*

TextRectRange对象包含一个隐形容器盒的4条边的具体坐标位置。有两个被所有元素对象与TextRange对象所共享的方法都能生成文本矩形的相关信息，但这两种信息之间存在着一定的不同。getClientRects()方法会返回由逐行文本矩形组成的集合，而getBoundingClientRect()方法则会返回一个将所有逐行文本矩形整合在一个坐标区域的TextRectRange对象。

但无论是调用哪个方法，都会立即获得这些矩形的坐标值。调整窗口大小或改变目标对象的内容均会改变实际的文本域矩形空间，但之前获得的TextRange对象并不会与这些变化保持同步。因此，在其他脚本须要处理文本域坐标位置之前再获得TextRectRange对象的值比较好。

对象模型引用方式

elementOrTextRangeReference.getBoundingClientRect()
elementOrTextRangeReference.getClientRects()[i]

对象特定属性 bottom、left、top、right
对象特定方法 无。
对象特定事件 无。

bottom, left, right, top

IE *5(Win)* NN *n/a* Moz *n/a* Saf *n/a* Op *n/a* DOM *n/a*

只读

这4个属性分别返回文本域矩形空间的4条边在浏览器窗口中的整数坐标值，单位为像素。须要注意的是，并是不以页面为参照物来度量这些值。因此文本所在的页面发生滚动时它们的值也会随之发生改变。

示例
```
var rightMostEdge = document.getElementById("myP").getBoundingClientRect().right;
```
值 整数像素度量。
默认值 无。

tfoot

请参见tbody。

th

请参见td。

thead

请参见tbody。

title

IE *4* NN *n/a* Moz *all* Saf *all* Op *7* DOM *1*

title对象对应于title元素。如果通过元素对象的id属性来引用对象时发生了问题，那么可以尝试使用标签访问方法，如document.getElementsByTagName("title")[0]。

等价HTML元素 <title>
对象模型引用方式 [window.]document.getElementById("*elementID*")
对象特定属性 text
对象特定方法 无。
对象特定事件 无。

text

IE *4* NN *n/a* Moz *all* Saf *all* Op *7* DOM *1*

可读/可写

指定元素内的文本内容。这是位于title元素的起始标签和结束标签之间的文本内容，浏览器窗口的标题栏或标签栏也会显示这些内容。此外，从浏览器中观察文档源代码时，并不会发现这个值的任何改动迹象。而且属性值的改变也不会影响IE/Windows的标题栏。

示例 `document.getElementsByTagName("title")[0].text = "Welcome, Dave!";`
值 字符串。
默认值 无。

toolbar

请参见directories。

tr

IE *4* NN *n/a* Moz *all* Saf *all* Op *7* DOM *1*

tr对象对应于tr元素。

等价HTML元素 `<tr>`
对象模型引用方式
`[window.]document.getElementById("elementID")`
`[window.]document.getElementById("tableID").rows[i]`
对象特定属性 align、bgColor、borderColor、borderColorDark、borderColorLight、cells[]、ch、chOff、height、rowIndex、sectionRowIndex、vAlign
对象特定方法 deleteCell()、insertCell()
对象特定事件 无。

align

IE *4* NN *n/a* Moz *all* Saf *all* Op *7* DOM *1*

可读/可写

定义了行内所有单元格内容的水平对齐方式。

示例 `document.getElementById("myTR").align = "center";`
值 以下3个对齐常量之一：center | left | right。
默认值 left

bgColor

IE *4* NN *n/a* Moz *all* Saf *all* Op *7* DOM *1*

可读/可写

指定当前行内所有表格单元格的背景色。而样式单的backgroundColor属性并不能反映这个属性设定。即使使用明语颜色名称设置元素或对象的bgcolor属性，其返回值仍然是一个十六进制三元组。

示例 `document.getElementById("myTR").bgColor = "yellow";`
值 一个十六进制的三元组或明语颜色名称。目前可用的明语颜色名称请参见附录A。
默认值 由浏览器和操作系统共同决定。

borderColor

IE *4* NN *n/a* Moz *n/a* Saf *n/a* Op *n/a* DOM *n/a*

可读/可写

此属性指定了元素边框的颜色。IE会将颜色应用于组成单元格边框的4条线段之上。因此临近单元格的边框颜色并不会互相冲突。

示例 `document.getElementById("myTR").borderColor = "salmon";`
值 一个十六进制的三元组或明语颜色名称。不设置属性值就默认为黑色（#000000，black）。目前可用的明语颜色名称请参见附录A。
默认值 由操作系统决定。

borderColorDark, borderColorLight

IE 4 NN n/a Moz n/a Saf n/a Op n/a DOM n/a

可读/可写

如果在页面背景或默认颜色四周仔细地勾勒出明暗线条，就可以在IE中创造出3D效果的表格边框。通过为borderColorDark（单元格的左边缘与上边缘）和borderColorLight（单元格的右边缘与下边缘）属性指定属性值，还可以独立地控制明暗两种线条的颜色。

此时应该为这对属性设置互补的颜色。而对于具体的颜色设置并没有强制性要求，并不一定要为borderColorDark属性设置暗色。这两个属性仅仅是为了使得边框线条轮廓分明，改变它们就可以控制一些边框线条的颜色。

示例

```
document.getElementById("myTR").borderColorDark = "blue";
document.getElementById("myTR").borderColorLight = "cornflowerblue";
```

值 一个十六进制的三元组或明语颜色名称。不设置属性值就默认为黑色（#000000，black）。目前可用的明语颜色名称请参见附录A。

默认值 由操作系统决定。

cells

IE 4 NN n/a Moz all Saf all Op 7 DOM 1

只读

返回由表格行中内嵌的td或th元素组成的集合。集合中的各项均以它们在源代码中的顺序进行排序。

示例

```
var allRowCells = document.getElementById("myTR").cells;
```

值 一个cells集合对象的引用。

默认值 长度为0的数组。

ch

IE 5(Mac)/6(Win) NN n/a Moz all Saf all Op 7 DOM 1

可读/可写

此属性用于指定一个字符，该字符会作为行内单元格中文本对齐的参考点。注意，只有将align属性设置为char，此属性才有价值。但在实际使用中，即使表格的块元素对象拥有这个属性，浏览器也不会对它的属性值作出任何响应。

示例

```
document.getElementById("myTR").ch = ".";
```

值 单个字符组成的字符串。

默认值 无。

chOff

IE 5(Mac)/6(Win) NN n/a Moz all Saf all Op 7 DOM 1

可读/可写

定义一个偏移点，char属性所指定的字符将出现在单元格中偏移点所在位置。但在实际使用中，即使表格的块元素对象拥有这个属性浏览器也不会对它的属性值作出任何响应。

示例

```
document.getElementById("myTR").chOff = "80%";
```

值 字符串形式的像素值或单元格宽度的百分比。

默认值 无。

height

IE 5 NN n/a Moz n/a Saf n/a Op n/a DOM n/a

可读/可写

本属性指明了当前行的高度。但在实际应用中，请使用元素的style.height而不是这个height属性来动态改变行的高度。

值 字符串形式的像素值或行高在整个表格中所占的百分比。
默认值 无。

rowIndex

IE *4* NN *n/a* Moz *all* Saf *all* Op *7* DOM *1*

850

只读

返回一个从零开始计数的整数，以表示当前行在整个表格内所有的tr元素中所处的位置。这一数值由tr元素的源代码次序决定。

示例
```
var whichRow = document.getElementById("myTR").rowIndex;
```
值 整数值。
默认值 无。

sectionRowIndex

IE *4* NN *n/a* Moz *all* Saf *all* Op *7* DOM *1*

只读

返回一个从零开始计数的整数，以表示当前行在行分组的所有的tr元素中所处的位置。thead、tbody和tfoot元素都可以视为一个行分组。具体的位置数值由tr元素的源代码次序决定。

示例
```
var whichRow = document.getElementById("myTR").sectionRowIndex;
```
值 整数值。
默认值 无。

vAlign

IE *4* NN *n/a* Moz *all* Saf *all* Op *7* DOM *1*

可读/可写

此属性指明了当前行内单元格内容的水平对齐的风格。

示例
```
document.getElementById("myTR").vAlign = "baseline";
```
值 不区分大小写的字符串常量：baseline | bottom | middle | top。
默认值 middle

deleteCell( )

IE *4* NN *n/a* Moz *all* Saf *all* Op *7* DOM *1*

```
deleteCell(index)
```

从当前tr元素中删除一个内嵌的td或th元素。参数传入的整数值指向行内cells集合中从零开始的某个td或th元素。

返回值 无。
参数

index

一个从零开始的整数，依照源代码次序，它对应于集合中的某个特定td元素（内嵌在当前元素之中）。

insertCell( )

IE *4(Win)* NN *n/a* Moz *all* Saf *all* Op *7* DOM *1*

851

```
insertCell(index)
```

向当前tr元素中插入一个内嵌的td元素。参数传入的整数值指向cells集合中某个从零开始的位置，以便插入新的单元格。在IE中也可以简单地使用-1作为参数，此时会将新单元格直接附加到集合的末尾。但此时插入的单元格仍然是一个空元素，可以使用不同的文档树修改技术向它填充所需的内容。然而不幸的是，在IE/Macintosh中使用脚本为表格添加行和单元格并不十分可行，这种方法往往会产生十分硕大的行和单元格尺寸。对于非内嵌表格，可以使用表格分块对象的简便方法来规避文档树节点的创建和插入工作。

返回值 新插入的单元格对象的引用。

参数

index

从零开始计数的一个整数，它对应于cells集合中的一个单元格对象，而新的单元格将插入到该单元格之前。

TreeWalker

IE *n/a* NN *n/a* Moz *1.0.1* Saf *1.3/2* Op *8* DOM *2*

TreeWalker对象是一个实时节点列表，该列表内的节点都符合document.createTreeWalker()方法所定义的规则。在这个列表中，它假设列表项之间的层次关系与这些列表项所对应的节点之间的层次结构完全相同。createTreeWalker()方法声明了列表的起始节点，并且还指明了应该从列表中过滤掉哪些节点或节点类。

TreeWalker对象实际上在列表中维持了一种指针。在此方法指定的方向上移动指针时，这个对象的各个方法使得脚本可以访问列表中的前一个或后一个节点。如果在TreeWalker创建之后脚本又修改了文档树，那么对应的修改会自动影响TreeWalker中的节点顺序。

虽然TreeWalker已经完全应用于HTML文档，但它在XML数据文档中的价值可能更高。例如，W3C DOM并未提供一种访问拥有同名属性的所有元素的简便方法。但可以定义一个TreeWalker对象来指向拥有该属性的所有节点，从而快速地连续访问这些对象。这样一来就不再须要编写繁琐的脚本代码来遍历所有节点以找寻所需的元素。例如，下面这个过滤方法只允许拥有author属性的节点成为TreeWalker对象的成员之一，代码如下：

```
function authorAttrFilter(node) {
    if (node.hasAttribute("author")) {
        return NodeFilter.FILTER_ACCEPT;
    }
    return NodeFilter.FILTER_SKIP;
}
```


这个方法的引用就可以成为createTreeWalker()方法的一个参数，从而对TreeWalker列表中的元素节点加以限制，代码如下：

```
var authorsOnly = document.createTreeWalker(document,
NodeFilter.SHOW_ELEMENT, authorAttrFilter, false);
```

创建TreeWalker对象后，就可以通过调用其方法从列表中获得某个节点的引用。调用TreeWalker的某个方法时，TreeWalker对象会根据方法所指定的指针移动方向，将滤镜应用于与内部指针当前位置相关的候选节点。一旦后续的文档树节点符合该方法的参数与过滤标准，方法就会返回该节点。拥有了返回的节点引用后，可以独立于TreeWalker列表中的节点项访问与该节点相关的任意DOM节点属性或方法。

对象模型引用方式 *TreeWalkerReference*

对象特定属性 currentNode、expandEntityReference、filter、root、whatToShow

对象特定方法 firstChild()、lastChild()、nextNode()、nextSibling()、parentNode()、previousNode()、previousSibling()、

对象特定事件 无。

currentNode

IE *n/a* NN *n/a* Moz *1.0.1* Saf *1.3/2* Op *8* DOM *2*

可读/可写

根据TreeWalker指针所在的位置返回对应节点的引用。但更为重要的是，将一个文档树节点的引用赋予此属性后，就可以为TreeWalker对象的指针指定一个新的位置。如果节点列表滤出了指定节点，那么在执行下一个方法调用时可能会认为该节点仍然存在于节点列表中。

示例 `myTreeWalker.currentNode = document.getElementById("main");`

值 文档树节点的一个引用。

默认值 文档的第1个节点。

expandEntityReference, filter, root, whatToShow

IE *n/a* NN *n/a* Moz *1.0.1* Saf *1.3/2* Op *8* DOM *2*

只读

这4个属性分别对应于`document.createTreeWalker()`方法在创建`TreeWalker`对象时传入的4个参数。

firstChild( ), lastChild( ), nextSibling( ), parentNode( ), previousSibling( )

IE *n/a* NN *n/a* Moz *1.0.1* Saf *1.3/2* Op *8* DOM *2*

这些方法分别返回`TreeWalker`对象中不同层次的节点引用。在`TreeWalker`的节点列表中，节点间的父子关系等同于它们在文档树中的父子关系。一旦调用以上这些方法之一，那么`TreeWalker`的内部指针会移动到节点列表内相关节点的临近位置。如果列表中与指定的节点引用相匹配的节点并不存在，那么此方法会返回`null`。这意味着在读取一个节点的属性时须要校验该节点的存在性，代码如下：

```
if (myTreeWalker.nextSibling()) {
    var theTag = myTreeWalker.currentNode.tagName;
}
```

如果不进行校验而直接访问了一个空引用的属性，那么就会导致一个引用错误。

返回值 文档树节点的一个引用。

参数 无。

nextNode( ), previousNode( )

IE *n/a* NN *n/a* Moz *1.0.1* Saf *1.3/2* Op *8* DOM *2*

这两个方法可以将内部的`NodeIterator`指针分别向前（`nextNode()`）或向后（`previousNode()`）移动一个位置，同时返回移动后指针所指向的节点引用。因此在执行这两个方法时会将节点层次视为一种扁平结构，即均以`NodeIterator`对象的方式进行节点访问。

返回值 文档树中一个节点的引用。

参数 无。

tt

请参见请b。

u

请参见b。

UIEvent

IE *n/a* NN *n/a* Moz *all* Saf *all* Op *7* DOM *2*

W3C DOM的`UIEvent`是一个抽象对象，它包含每个W3C DOM聚焦事件实例所共享的属性和方法。这个对象从W3C DOM的`event`对象中继承了相关的特性。而且它的属性和方法已经与`event`对象融合在一起。请参见本章前文中有关`event`对象的讨论，以了解支持这一对象的特定属性和方法，以及其他事件类型如何继承这些属性与方法。

854

ul

IE 4 NN n/a Moz all Saf all Op 7 DOM 1

ul对象对应于ul元素。

等价HTML元素
对象模型引用方式 [window.]document.getElementById("*elementID*")
对象特定属性 compact、type
对象特定方法 无。
对象特定事件 无。

compact

IE 4 NN n/a Moz all Saf all Op 7 DOM 1

可读/可写

如果将compact属性设置为true，它会命令浏览器以一种更为紧密的形式显示列表中的各项内容。但这个属性对主流浏览器没有任何影响。

示例 document.getElementById("myUL").compact = true;
值 布尔值：true | false。
默认值 false

type

IE 4 NN n/a Moz all Saf all Op 7 DOM 1

可读/可写

这个属性可以指定列表中前导符号的显示方式。

示例 document.getElementById("myUL").type = "square";
值 以下字符串常量之一：circle | disc | square。可以通过样式单增加一些有效的选项。
默认值 disc

userProfile

IE 4-6(Win) NN n/a Moz n/a Saf n/a Op n/a DOM n/a

userProfile对象对应于浏览器的用户配置文件中为当前用户存储的大量信息。这个对象拥有4个主要方法，它们是：

- 对配置文件中单独域的请求进行排列，如请求名称、邮件地址、电话号码等
- 显示请求对话框，以便让用户看到返回的请求内容或对某项请求的拒绝原因
- 获取信息
- 清空请求列表

855

一旦在用户许可下获得了请求信息，这些数据就会进入表单元素并被提交至服务器端。这一方法目前仅适用于Windows系统下的IE 4到IE 6浏览器。尽管IE/Macintosh会接受方法调用而且不会导致任何错误，但它并未为这些方法赋予任何实际的功能。请访问以下链接来获取用户配置文件的详细信息：*http://msdn.microsoft.com/workshop/management/profile/profile_assistant.asp*。

示例

```
navigator.userProfile.addReadRequest("vcard.displayname");
navigator.userProfile.doReadRequest("3", "MegaCorp Customer Service");
var custName = navigator.userProfile.getAttribute("vcard.displayname");
navigator.userProfile.clearRequest();
if (custName) {
    ...
}
```

对象模型引用方式 navigator.userProfile
对象特定属性 无。
对象特定方法 addReadRequest()、clearRequest()、doReadRequest()、getAttribute()
对象特定事件 无。

addReadRequest( )

IE *4-6(Win)* NN *n/a* Moz *n/a* Saf *n/a* Op *n/a* DOM *n/a*

```
addReadRequest("attributeName")
```

向查询队列中加入一个面向用户配置文件属性的访问请求，以便单独执行这些请求。添加到该队列的所有查询项均会被提交给用户，以便用户决定哪些项可以被提交至服务器端。如果要访问多个属性，那么请多次调用addReadRequest()方法。

返回值 布尔值：true，表示执行成功，执行失败则返回false。

参数

attributeName

下列任意一个属性名称字符串，不须要区分大写：vCard.Business.City、vCard.Business.Country、vCard.Business.Fax、vCard.Business.Phone、vCard.Business.State、vCard.Business.StreetAddress、vCard.Business.URL、vCard.Business.Zipcode、vCard.Cellular、vCard.Company、vCard.Department、vCard.DisplayName、vCard.Email、vCard.FirstName、vCard.Gender、vCard.Home.City、vCard.Home. Country、vCard.Home.Fax、vCard.Home.Phone、vCard.Home.State、vCard.Home.StreetAddress、vCard. Home.Zipcode、vCard.Homepage、vCard.JobTitle、vCard.LastName、vCard.MiddleName、vCard.Notes、vCard.Office、vCard.Pager。

clearRequest( )

IE *4-6(Win)* NN *n/a* Moz *n/a* Saf *n/a* Op *n/a* DOM *n/a*

此方法将清空待查询队列。当脚本成功获取所需信息后可以调用这个方法。这样就可以为下次查询做好准备。

返回值 无。
参数 无。

doReadRequest( )

IE *4-6(Win)* NN *n/a* Moz *n/a* Saf *n/a* Op *n/a* DOM *n/a*

```
doReadRequest(usageCode[, "friendlyName"[, "domain"[, "path"[, "expiration"]]]])
```

根据队列中的待查询项，这个方法会命令浏览器检查用户是否已经允许查询对应的属性。如果用户并未赋予查询权，那么这个方法就会显示一个对话框（即“配置文件助理”窗口），以便用户关闭某些查询项从而阻止将它们暴露给服务器。方法中的各个参数提供了话框中所须要显示的各项内容，以及用户许可维护所需的信息，这与cookie的管理方式比较类似。无论在查询队列中存在多少内容，都只须调用一次doReadRequest()方法。

返回值 在Windows中，无论用户如何响应“配置文件助理”对话框，这个方法都不会返回任何信息。而在Macintosh中，这个方法并不会显示“配置文件助理”对话框，并且会返回false。

参数

usageCode

下表中列出的任意一个整数代码，它对应于由互联网隐私工作组（Internet Privacy Working Group）定义的一段说明信息。

代码	含义
0	用于系统管理
1	用于研究和产品部署
2	用于支持并完成当前事务
3	用于内容定制及站点设计
4	用于改善站点内容，其中包括广告信息

续表

代码	含义
5	用于向访问者提示站点的更新内容
6	用于联系访问者以便推销服务与产品
7	用于链接其他信息
8	站点出于其他目的使用此信息
9	为定制或改进网站内容并设计站点而向他人透露相关信息
10	透露给可能会因服务和产品营销而跟您联系的人
11	透露给可能会因服务和产品营销而跟您联系的人，但此时还有机会阻止网站这样做
12	出于其他目的而向他人透露信息

friendlyName
一个可选的字符串，在该字符串中包含一个可识别的名称（和URL）以便用户了解该请求的来源。它可能就是一个法人身份。

domain
一个可选字符串，它包含了发起此请求的服务器域名。如果设置了截止日期，那么这个信息会同请求属性保存在一起，以防止来自于该域名的请求反复打搅用户。

path
一个可选字符串，它包含了发起此请求的服务器路径。如果设置了截止日期，那么这个信息会同请求属性保存在一起，以防止来自于该路径的请求反复打搅用户。

expiration
一个可选的字符串，它包含了用户许可的超时时间。但IE 4无法识别这个参数。

getAttribute( )

IE *4-6(Win)* NN *n/a* Moz *n/a* Saf *n/a* Op *n/a* DOM *n/a*

```
getAttribute("attributeName")
```

如果用户已经给予了访问许可，那么此方法会返回用户配置文件中某个属性的值。如果用户拒绝赋予访问许可，那么将返回null。此外，每次要获取一个属性值时都须要调用一个getAttribute()方法。

返回值 字符串或null。

参数

attributeName
addReadRequest()方法声明中的vCard属性名称之一。

ValidityState

IE *n/a* NN *n/a* Moz *n/a* Saf *n/a* Op *9* DOM *n/a*

在Web Forms 2.0中，每个能够接收用户输入的表单控件都拥有一个validity属性，而这个属性正是ValidityState对象的一个实例。这个对象包含9个布尔属性，如果对应的校验测试失败，那么其中的8个属性可能会被设置为true。input、textarea和select元素都提供了充足的附加属性，它们可以让页面作者预先设置值类型、大小写限制、文本模式甚至判断表单提交前用户输入是否已经完成。例如，如果要求一个文本input元素接收特定数值范围内的一个数字，那么当用户输入一个字母时浏览器会将validity.typeMismatch设置为true，而如果用户输入的数字大于预设范围，就会将validity.rangeOverflow设定为true。

当ValidityState对象的实例开始设置valid属性时，所有的这些属性就会发挥作用，因此可以将valid属性看作是元素是否通过校验的看门人。如果某个特定的校验错误属性的值为true，那么valid属性就会被设置为false。换话说，校验测试必须一路绿灯，即每个校验属性均为false，才能将valid属性设置为true。

下面这段代码将校验测试与一个output元素联系在了一起，发生错误时，由一个与时间相关的input元素向用户显示一个明语提示信息：

```
<script type="text/javascript"> // <![CDATA[
function validateField(evt) {
```

```
        var form = evt.target.form;
        var errField = form.elements[evt.target.name + "Error"];
        if (evt.target.validity.typeMismatch) {
           errField.value = 'You must enter a time (hh:mm).';
        } else if (evt.target.validity.stepMismatch) {
           errField.value = 'Appointments must begin at 0, 15, 30, or 45 past the hour.';
        } else if (evt.target.validity.rangeUnderflow) {
           errField.value = 'The earliest appointment is 9:00 am.';
        } else if (evt.target.validity.rangeOverflow) {
           errField.value = 'The last appointment is 5:00 pm.';
        } else if (evt.target.validity.valueMissing) {
           errField.value = 'You must enter a time.';
        } else {
           errField.value = '';
        }
        evt.preventDefault();
    // ]]>
    }
    </script>
    ...
    <form action="..." method="get" >
    <p><label>Desired appointment time:
        <input type="time" name="apptTime" min="09:00" max="17:00" value="09:00" step="900"
           required="required" onforminput="validateField(event)" />
    </label>
    <output name="apptTimeError" />
    </p>
    <p>
    <input type="submit" />
    </p>
    </form>
```


对象模型引用方式 `[window.]document.getElementById("elementID")`

对象特定属性 customError、patternMismatch、rangeOverflow、rangeUnderflow、stepMismatch、tooLong、typeMismatch、valid、valueMissing

对象特定方法 无。

customError

IE *n/a* NN *n/a* Moz *n/a* Saf *n/a* Op *9* DOM *n/a*

只读

如果此属性的值为`true`，则表示脚本对控件的操作发生了错误。

值 布尔值：`true | false`。

默认值 `false`

patternMismatch

IE *n/a* NN *n/a* Moz *n/a* Saf *n/a* Op *9* DOM *n/a*

只读

如果此属性值为`true`，表示输入值与正则表达式的指定值不符。如果文本域为空，那么这个属性的值应该为`false`。

值 布尔值：`true | false`。

默认值 `false`

rangeOverflow, rangeUnderflow

IE *n/a* NN *n/a* Moz *n/a* Saf *n/a* Op *9* DOM *n/a*

只读

如果这两个属性的值为`true`，则它们分别表示输入值（如数字、时间、日期或文件等）超出了`max`或`min`属性的设定值。

值 布尔值：true | false。
默认值 false

860

stepMismatch

IE *n/a* NN *n/a* Moz *n/a* Saf *n/a* Op *9* DOM *n/a*

只读

如果此属性的值为true，表示输入值不符合控件的step属性所指定的值增量参数。
值 布尔值：true | false。
默认值 false

tooLong

IE *n/a* NN *n/a* Moz *n/a* Saf *n/a* Op *9* DOM *n/a*

只读

如果此属性的值为true，表示输入值超出了控件的maxlength属性所确定的字符长度。
值 布尔值：true | false。
默认值 false

typeMismatch

IE *n/a* NN *n/a* Moz *n/a* Saf *n/a* Op *9* DOM *n/a*

只读

如果此属性的值为true，表示输入值的类型并非是控件的type属性或文件输入元素的accept属性所期望的类型。
值 布尔值：true | false。
默认值 false

valid

IE *n/a* NN *n/a* Moz *n/a* Saf *n/a* Op *9* DOM *n/a*

只读

如果此属性的值为true，表示其他属性值均为false，即控件已通过了校验。
值 布尔值：true | false。
默认值 false

valueMissing

IE *n/a* NN *n/a* Moz *n/a* Saf *n/a* Op *9* DOM *n/a*

只读

如果此属性的值为true，并且元素的required属性已被设置，则表示输入值为空。
值 布尔值：true | false。
默认值 false

861

var

请参见abbr。

ViewCSS

IE *4* NN *n/a* Moz *all* Saf *1.3/2* Op *7* DOM *1*

W3C DOM的ViewCSS对象是一个抽象对象，它将其getComputedStyle()方法提供给了document.defaultView对象。而document.defaultView对象则为给定的元素节点提供了实际显示的样式单属性。
对象模型引用方式 [window.]document.defaultView
对象特定属性 无。

对象特定方法 getComputedStyle()
对象特定事件 无。

getComputedStyle( )

IE *n/a* NN *n/a* Moz *all* Saf *1.3/2* Op *7* DOM *2*

```
getComputedStyle(elementNodeReference, "pseudoElementName")
```

根据第1个参数传入的元素对象引用，返回影响它的网状级联样式设定的样式对象（W3C DOM的术语称之为CSSStyleDeclaration）。如果要获取某个特殊的样式属性值，请首先使用这个方法获得一个样式对象，然后再调用该对象的getPropertyValue()方法，代码如下：

```
var compStyle = getComputedStyle(document.getElementById("myP"), "");
var pBGColor = compStyle.getPropertyValue("background-color");
```

请参见style对象以获取更多详细信息。

返回值 style（CSSStyleDeclaration）对象引用。
参数

elementNodeReference
文档树中一个元素节点的引用。

pseudoElementName
一个伪元素的名称（如first-line）或一个空字符串。

wbr

IE *4* NN *n/a* Moz *all* Saf *all* Op *7* DOM *1*

wbr对象对应于wbr元素。

等价HTML元素 <wbr>
对象模型引用方式 [window.]document.getElementById("*elementID*")
对象特定属性 无。
对象特定方法 无。
对象特定事件 无。

window

IE *4* NN *n/a* Moz *all* Saf *all* Op *7* DOM *1*

window对象代表了显示文档内容的浏览器窗口或框架。当脚本必须与其他框架或子窗口中的文档对象通信时，这个window对象在脚本编程中扮演了一个非常重要的角色。由于跨窗口的引用存在时间很短，因此管理多个窗口往往是一个很棘手的工作。HTML和XHTML标准的严格解释并不赞成使用多窗口，而且很多心存困惑的用户也会赞同这一决定。

由于窗口毕竟位于文档标记的作用域之外，因此W3C DOM Level 2没有为window对象提供深入全面的说明，但它仍然通过一种称为“视图”的对象展现了未来的应用方向。Netscape 6的document.defaultView属性可以返回文档的窗口，而且它的window对象还具有ViewCSS对象的方法以便获得DOM的getComputedStyle()方法。

自从诞生之日起，就可以使用脚本操纵window对象，而且它还拥有数量可观的属性和方法。但这些功能与特定的浏览器有关，因此在采用一个对象功能前请仔细评判它在各种浏览器中的兼容性。

对象模型引用方式 window、self、top、parent
对象属性 clientInformation、clipboardData、closed、Components、content、controllers、crypto、defaultStatus、dialogArguments、dialogHeight、dialogLeft、dialogTop、dialogWidth、directories、document、event、external、frameElement、frames[]、fullScreen、history、innerHeight、innerWidth、length、location、locationbar、menubar、name、navigator、netscape、offscreenBuffering、opener、outerHeight、outerWidth、pageXOffset、

863

pageYOffset、parent、personalbar、pkcs11、prompter、returnValue、screen、screenLeft、screenTop、screenX、screenY、scrollbars、scrollMaxX、scrollMaxY、scrollX、scrollY、self、sidebar、status、statusbar、toolbar、top、window

对象方法 addEventListener()、alert()、attachEvent()、back()、blur()、captureEvents()、clearInterval()、clearTimeout()、close()、confirm()、createPopup()、detachEvent()、disableExternalCapture()、dispatchEvent()、dump()、enableExternalCapture()、execScript()、find()、focus()、forward()、GeckoActiveXObject()、getComputedStyle()、getSelection()、home()、moveBy()、moveTo()、navigate()、open()、openDialog()、print()、prompt()、releaseEvents()、removeEventListener()、resizeBy()、resizeTo()、routeEvent()、scroll()、scrollBy()、scrollByLines()、scrollByPages()、scrollTo()、setInterval()、setTimeout()、showHelp()、showModalDialog()、showModelessDialog()、sizeToContent()、stop()

对象特定事件

事件	IE/Windows	Mozilla	Safari	Opera	DOM
afterprint	•	—	—	—	—
beforeprint	•	—	—	—	—
beforeunload	•	—	•	—	—
blur	•	•	•	•	—
dragdrop	—	•	—	—	—
error	•	•	•	•	—
focus	•	•	•	•	—
help	•	—	—	—	—
load	•	•	•	•	—
move	•	—	—	—	—
resize	•	•	•	•	—
scroll	•	•	•	•	—
unload	•	•	•	•	—

864

clientInformation

IE *4* NN *n/a* Moz *n/a* Saf *1.2* Op *n/a* DOM *n/a*

只读

此属性可以返回一个navigator对象。navigator对象的名称来自于一个知名的浏览器品牌。它的clientInformation属性是获取重要环境变量的一个通识方法，而一直以来，也都是通过navigator对象的属性和方法一样来获得这些环境信息。本章前文中已对navigator对象进行了单独说明。

示例

```
if (parseInt(window.clientInformation.appVersion) >= 4) {
    // 针对 IE 4 及后续版本的处理代码
}
```

值 navigator对象。

默认值 navigator对象。

clipboardData

IE *5(Win)* NN *n/a* Moz *n/a* Saf *n/a* Op *n/a* DOM *n/a*

只读

返回一个clipboardData对象，本章前文中已对clipboardData对象进行了单独的讨论。这个对象（可作为window或frame对象的一个属性进行访问）是一个临时容器。Windows系统下IE 5及后续版本的IE浏览器都可以使用这个临时容器来转移文本数据，一些脚本控制的操作尤其需要这个功能，例如模拟剪切、复制和粘贴，以及控制拖拽等。

示例

```
var rng = document.selection.createRange();
clipboardData.setData("Text",rng.text);
```

值 clipboardData对象。

默认值 clipboardData对象。

closed

IE 4 NN 3 Moz all Saf all Op 7 DOM n/a

只读

这是一个布尔属性，它的值表明了该窗口是否已被关闭。其值为true就表明无法再继续引用这个窗口的对象或脚本组件。这个方法常用于检测用户是否已经关闭了一个由window.open()方法所创建的子窗口。

示例

```
if (!newWindow.closed) {
  newWindow.document.close();
}
```

值 布尔值：true | false。

默认值 无。

865

Components, content, controllers, prompter, sidebar

IE n/a NN n/a Moz all Saf n/a Op n/a DOM n/a

只读

这些属性均与Mozilla的xpconnect（xpconnect，即Cross Platform Connect）服务有关。这些服务允许已通过安全审查的脚本与XPCOM（并不是微软的COM）对象协同工作，以便让使用Mozilla引擎的浏览器（如Netscape 6）能够扩展应用程序的功能。而访问这些服务须要开启安全权限（一般是通过签名脚本），如下所示：

```
netscape.security.PrivilegeManager.enablePrivilege("UniversalXPConnect");
    // xpconnect 处理代码
netscape.security.PrivilegeManager.revertPrivilege("UniversalXPConnect");
```

关于这种机制的更多详细信息，请访问*http://www.mozilla.org/scriptable/*。

crypto, pkcs11

IE n/a NN n/a Moz all Saf n/a Op n/a DOM n/a

只读

这两个属性会返回与Mozilla的公共密匙加密技术本质有关的对象。关于这个主题的更多详细，请访问*http://www.mozilla.org/projects/security/*。

defaultStatus

IE 3 NN 2 Moz all Saf all Op 7 DOM n/a

可读/可写

这个属性可以指定一段默认讯息，以便在浏览器并未执行任何加载动作时在浏览器窗口状态栏中显示这些信息。如果要临时改变这些讯息，可以设定窗口的status属性。而大多数支持脚本的浏览器品牌及相关版本在管理defaultStatus属性时都存在着一定的困难。它们可能会产生各种古怪的行为。

示例 window.defaultStatus = "Make it a great day!";

值 任何字符串。

默认值 无。

dialogArguments

IE 4 NN n/a Moz n/a Saf n/a Op n/a DOM n/a

只读

866

这是通过window.showModalDialog()或window.showModelessDialog()（仅适用于Windows系统下的IE 5

及后续版本浏览器）来创建一个模式对话框窗口时传入方法的字符串参数或其他数据类型的额外参数。最好通过位于对话框窗口文档中的脚本来访问这个属性，以便获得方法参数向这个新窗口中传递的数据。如果使用某种分界符将多个参数整合在传入的数据中，那么可以通过脚本将它们解析出来。

示例

```
// 这段代码位于对话框窗口中
var allArgs = window.dialogArguments;
var firstArg = allArgs.substring(0, allArgs.indexOf(";"));
```

值 字符串、数字、数组或对象。

默认值 无。

dialogHeight, dialogWidth

IE *4(Win)* NN *n/a* Moz *n/a* Saf *n/a* Op *n/a* DOM *n/a*

可读/可写

这两个属性可以分别表示通过`showModalDialog()`或`window.showModelessDialog()`方法所创建的模式对话框窗口的高度和宽度值。尽管IE浏览器也允许模式对话框窗口中的脚本修改这些属性，但修改后的值通常并不会改变对话框窗口的尺寸大小。而对话框开启方法会将这两个属性的初始值作为其输入参数传递给对话框。

示例 `var outerWidth = window.dialogWidth;`

值 字符串，其中包括独立单位，如`520px`。

默认值 无。

dialogLeft, dialogTop

IE *4* NN *n/a* Moz *n/a* Saf *n/a* Op *n/a* DOM *n/a*

可读/可写

这两个属性分别指定了模式对话框窗口的左、上边缘相对于显示屏左上角的偏移距离。尽管IE浏览器也允许模式对话框窗口中的脚本修改这些属性，但修改后的值通常并不会改变对话框窗口的相对位置。而对话框开启方法会将这两个属性的初始值作为其输入参数传递给对话框。

示例 `var outerLeft = window.dialogLeft;`

值 字符串，其中包括独立单位，如`80px`。

默认值 无。

directories, locationbar, menubar, personalbar, scrollbars, statusbar, toolbar

IE *n/a* NN *4* Moz *all* Saf *n/a* Op *n/a* DOM *n/a*

只读

这些属性均可返回Navigator浏览器某个窗口功能部件的引用（`directories`对Mozilla而言是一个新属性）。请参见本章中`directories`对象的相关讨论，以便了解如何通过签名脚本控制这些窗口部件的可见性。

示例

```
netscape.security.PrivilegeManager.enablePrivilege("UniversalBrowserWrite")
window.statusbar.visible = "false";
netscape.security.PrivilegeManager.revertPrivilege("UniversalBrowserWrite")
```

值 各自对应的对象引用。

默认值 无。

document

IE *3* NN *2* Moz *all* Saf *all* Op *7* DOM *2*

只读

返回窗口中包含的`document`对象的引用。在那些能够为W3C DOM文档树提供支持的浏览器中，这个属性指向窗口文档树中更为特殊的根`HTMLDocument`节点。W3C DOM甚至将这个属性也作为其`View`对象的成员之

一，这与浏览器窗口有些类似。通过这个属性，可以让脚本引用文档方法，以及文档内从“document”一词开始的所有内容。

示例 `var oneElem = document.getElementById("myP");`
值 根document对象的引用。
默认值 根document对象的引用。

event

IE 4 NN n/a Moz n/a Saf n/a Op 7 DOM n/a

只读

IE浏览器的事件模型会为每个用户或系统事件均生成一个event对象。这个event对象是window对象的一个属性。而Opera则实现了一种混合了IE和W3C DOM的事件机制。因此window.event属性返回的event对象同时具有IE和W3C DOM这两者的特征，如sourceElement和target属性。关于IE事件对象的详细信息，请参见在线参考VI，以及本章中列出的event对象。

示例

```
if (event.altKey) {
    // Alt 键被按下进行处理
}
```

值 event对象引用。
默认值 无。

external

IE 4(Win) NN n/a Moz n/a Saf n/a Op n/a DOM n/a

只读

返回一个external对象的引用，在安全许可允许的情况下，通过该引用可以访问浏览器引擎低级别的功能请参见本章前文中有关external对象的讨论。

示例 `external.AddFavorite("http://www.dannyg.com", "Danny Home Page");`
值 external对象引用。
默认值 external对象引用。

frameElement

IE 5.5 NN n/a Moz 1.0.1 Saf 1.2 Op 7 DOM n/a

只读

如果当前窗口是框架集中的一员或是一个内联框架，那么frameElement属性会返回一个frame或iframe元素对象的引用。但安全性约束却会阻止脚本访问这个属性。

示例 `var frameID = window.frameElement.id;`
值 frame或iframe对象引用，或者null。
默认值 null。

frames[ ]

IE 3 NN 2 Moz all Saf all Op 7 DOM n/a

只读

返回由当前窗口中实现的框架或内联框架的window对象所组成的集合。对一个框架集的parent或top窗口而言，这个数组包含其中的第一代frame窗口引用。数组的索引值既可以使用从零开始计数的整数（按照源代码顺序），也可以使用已指定给frame元素name属性的标识符。

示例 `parent.frames[1].myFunc();`
值 frame或window对象引用数组。
默认值 长度为0的数组。

fullScreen

IE *n/a* NN *n/a* Moz *1.4* Saf *n/a* Op *n/a* DOM *n/a*

只读

这个属性的返回值为布尔值false，即使已通过“查看”菜单将浏览器设置为全屏模式也是如此。

869

值 布尔值：true | false。

默认值 false

history

IE *3* NN *2* Moz *all* Saf *all* Op *7* DOM *n/a*

只读

此属性可为当前窗口或框架包含history对象。详细信息请参见history对象的相关讨论。

示例

```
if (self.history.length > 4) {
    ...
}
```

值 history对象引用。

默认值 当前history对象。

innerHeight, innerWidth

IE *n/a* NN *4* Moz *all* Saf *all* Op *7* DOM *n/a*

可读/可写

这两个属性可以分别指定浏览器窗口或框架中内容区域的高度和宽度，单位为像素。这个区域是文档内容出现时所占的空间范围，但并不包含所有的窗口“装饰”。IE中的对应属性，请参见body元素对象。

示例

```
window.innerWidth = 600;
window.innerHeight = 400;
```

值 整数值。

默认值 无。

length

IE *4* NN *n/a* Moz *all* Saf *all* Op *7* DOM *n/a*

只读

此属性可以表明当前窗口中内嵌的框架的数量。它与window.frames.length的返回值完全相同。而当窗口中未定义任何框架时，这个值为零。

示例

```
if (window.length > 0) {
    ...
}
```

值 整数值。

默认值 0

location

870

IE *3* NN *2* Moz *all* Saf *all* Op *7* DOM *n/a*

可读/可写

根据窗口或框架中当前加载的文档，返回包含其URL详细信息的location对象。如果要导航到另一个页面，可以为location.href属性赋予一个新URL地址，在IE浏览器中则可使用navigate()方法达到这一目的。请参见location对象。

示例 top.location.href = "index.html";

值 字符串形式的完整或相对URL。

默认值 当前location对象。

locationbar

请参见directories。

menubar

请参见directories。

name

IE 3 NN 2 Moz all Saf all Op 7 DOM n/a

可读/可写

这是一个框架或子窗口的标识符，可以将它指定给target属性，脚本也可以使用它引用对应的框架或子窗口。对一个框架而言，通常直接在frame元素标签的name属性上指定属性值，但也可以通过脚本对它进行修改。而子窗口的名称则会作为一个参数传递给window.open()。主浏览器窗口在默认情况下并没有窗口名称，但如果子窗口须要将一个链接或表单指向主窗口，那么也可以通过脚本为它指定一个合适的名称。

示例

```
if (parent.frames[1].name == "main") {
    ...
}
```

值 区分大小写的标识符，必须遵循标识符命名规则：不包含空白、不能以数字作为起始字符、除下划线外不允许使用其他标点符号。

默认值 无。

navigator

IE 4 NN n/a Moz all Saf all Op 7 DOM n/a

只读

返回一个navigator对象的引用。由于window引用是一个可选的属性，因此不包含window引用的语法可以正常运行在所有可执行脚本的IE和Navigator浏览器中。请参见navigator对象。

示例 `var theBrowser = navigator.appName;`

值 navigator对象引用。

默认值 navigator对象。

netscape

IE n/a NN 3 Moz all Saf n/a Op n/a DOM n/a

只读

返回netscape对象的一个引用，通过签名脚本和用户的明确许可，这个对象可以对应用程序的内部运作提供有限的脚本访问功能。例如，JavaScript可以通过netscape对象来访问PrivilegeManager。

示例 `netscape.security.PrivilegeManager.enablePrivilege("UniversalBrowserWrite");`

值 netscape对象引用。

默认值 netscape对象。

offscreenBuffering

IE 4 NN n/a Moz n/a Saf 1.2 Op n/a DOM n/a

可读/可写

指定浏览器是否应该使用画外缓冲来提高路径动画的执行效果。尽管在IE 5/Macintosh中实现了这个属性，但目前还不清楚它具体提供了哪些功能。Windows中较新版本的IE浏览器将这个属性与DirectX ActiveX控件联

系在了一起。在加载文档时，会将这个属性设置为auto。此后，通过为这个属性指定布尔值，脚本就可以打开或关闭缓冲功能。

示例 `window.offscreenBuffering = "true";`
值 布尔值：true | false。
默认值 auto

opener

IE 3 NN 3 Moz all Saf all Op 7 DOM n/a

可读/可写

这是一个对象引用，该对象对应的窗口或框架使用window.open()方法生成了当前窗口。这个属性允许来源窗口中的子窗口将对象引用、变量和方法整合在一起。如果要访问源窗口中的文档对象，可以使用opener属性作为起始，然后通过正常的文档对象层次来达到目的，下文示例等式中的左侧代码就演示了这一操作过程。而源窗口与被开启窗口间的关系并不是严格的父子关系。在用脚本操作窗口和框架的引用时，这里的父子关系还包含着其他的内涵。另外，通过showModalDialog()和showModelessDialog()创建的IE对话框窗口并不支持这个属性。此时必须在创建时使用参数来传递主窗口和对话框窗口之间的引用，并在关闭对话框窗口时通过returnValue属性来获得这个值。

示例 `opener.document.forms[0].importedData.value = document.forms[0].entry.value;`
值 window对象引用。
默认值 无。

outerHeight, outerWidth

IE n/a NN 4 Moz all Saf all Op 7 DOM n/a

• 可读/可写

这两个属性可以分别指定浏览器窗口或框架的高度和宽度，此时度量区域还包括所有的工具栏、滚动条或其他可见的窗口“装饰”，单位为像素。IE并未提供与之等价的属性。

示例

```
window.outerWidth = 80;
window.outerHeight = 600;
```

值 整数值。
默认值 无。

pageXOffset, pageYOffset

IE n/a NN 4 Moz all Saf all Op 7 DOM n/a

只读

这个两个属性可以分别表明页面内容向上或向左滚动的具体数值。例如，如果一个文档滚动后导致顶端有100像素高的文档（或称之为页面）内容不可见，那么这个窗口的pageYOffset值就是100。如果文档并未被滚动，那么这两个属性值均为0。

示例 `var vertScroll = self.pageYOffset;`
值 整数值。
默认值 0

parent

IE 3 NN 2 Moz all Saf all Op 7 DOM n/a

只读

返回包含父级window对象的引用，在这个window对象定义的框架集中包含了当前框架。在一个子框架中，可以使用parent属性来访问父级文档的变量或方法，也可以通过它来引用另一个子框架中的变量、方法和对

象。例如，如果子框架中的脚本必须引用另一个子框架（名为“content”）内的一个文本输入控件元素的内容，那么这个引用过程可以表示如下：

```
parent.content.document.forms[0].entryField.value
```

对于嵌套层次较深的框架还可以反复调用父级对象，例如，parent.parent.frameName。

示例 `parent.frames[1].document.forms[0].companyName.value = "MegaCorp";`
值 window对象引用。
默认值 当前window对象的引用。

personalbar

请参见directories。

pkcs11

请参见crypto。

prompter

请参见Components。

returnValue

IE *4(Win)/5(Mac)* NN *n/a* Moz *n/a* Saf *n/a* Op *n/a* DOM *n/a*

可读/可写

在关闭一个IE对话框窗口时会将这个值返回给主窗口。对话框窗口中运行的脚本为这个属性赋予的属性值会成为主窗口中showModalDialog()的返回值。例如，模式对话框窗口中的文档可能会使用下面这段指令为returnValue属性设置一个来自于对话框信息的属性值：

```
window.returnValue = window.document.forms[0].userName.value;
```

而主文档中可以通过下面这句代码创建一个对话框：

```
var userName = showModalDialog("userNamePrompt.html");
```

在对话框已关闭并且脚本也执行完毕后，无论对话框中为returnValue指定了什么值，都会将它赋予这个userName变量。

值 任意可通过脚本处理的数据类型。
默认值 无。

screen

IE *4* NN *n/a* Moz *all* Saf *all* Op *7* DOM *n/a*

只读

返回一个screen对象的引用。由于window引用是可选的，因此不包含window引用的语法也可以正常运行在所有可执行脚本的浏览器中。

示例 `var howHigh = screen.availHeight;`
值 screen对象引用。
默认值 screen对象。

screenLeft, screenTop

IE *5(Win)* NN *n/a* Moz *n/a* Saf *1.2* Op *7* DOM *n/a*

只读

这两个属性分别返回浏览器内容区域的左、上角相对于屏幕左、上角的坐标，单位为像素。最大化的IE浏览

器窗口返回的screenLeft值为0，而screenTop值则会随着用户选择的不同工具栏组件而发生变化。请使用window.moveTo()来改变窗口位置。

示例 `var fromTheTop = window.screenTop;`
值 整数值。
默认值 依赖于用户的设定。

screenX, screenY

IE *n/a* NN *n/a* Moz *all* Saf *1.2* Op *7* DOM *n/a*

可读/可写

这两个属性分别返回整个浏览器窗口（包括“装饰”）的左、上角相对于屏幕左、上角的坐标，单位为像素。在Windows系统中，由于最大化的浏览器窗口中的“装饰”内容会超出屏幕显示的范围，因此screenX和screenY的值为-4。可以通过这两个属性或window.moveTo()方法来调整窗口的位置。

示例 `var fromTheTop = window.screenY;`
值 整数值。
默认值 依赖于用户的设定。

scrollbars

请参见directories。

scrollMaxX, scrollMaxY

IE *n/a* NN *n/a* Moz *1.4* Saf *n/a* Op *n/a* DOM *n/a*

只读

返回当前窗口内容在水平（scrollMaxX）和竖直（scrollMaxY）轴向上可以滚动的像素距离。如果在某个轴向上不存在滚动条，那么由于窗口根本无法在这个方向上滚动，因此对应值将为0。将窗口的innerWidth和innerHeight值与对应的最大滚动属性值相加后就可以得到一个可滚动文档的实际高度和宽度。

示例 `var docHeight = window.scrollMaxY + window.innerHeight;`
值 整数值。
默认值 0

scrollX, scrollY

IE *n/a* NN *n/a* Moz *all* Saf *all* Op *n/a* DOM *n/a*

只读

这两个属性分别返回当前窗口沿着横轴（scrollX）和纵轴（scrollY）的滚动距离，单位为像素。如果要在IE中决定其取值，必须要将IE 6中的兼容性模式设定考虑在内。请参见第1章中与DOCTYPE元素的有关内容。在quirks模式中，以及在IE/Macintosh中，请使用document.body.scrollLeft和document.body.scrollTop属性。在IE 6标准兼容模式（此时document.compatMode == "CSS1Compat"）中则请使用document.body.parentNode.scrollLeft和document.body.parentNode.scrollTop来获取html元素的滚动距离。

示例 `var scrolledDown = window.scrollY;`
值 整数值。
默认值 0

self

IE *3* NN *2* Moz *all* Saf *all* Op *7* DOM *n/a*

只读

这个属性是当前窗口或框架的一个引用。此属性与window属性相同，但在引用了很多框架和窗口的复杂脚本代码中，使用这个属性有利于提高代码的可读性。请千万不要使用window.self这种方式来引用当前窗口或框架，这会使得某些浏览器版本完全不知所措。

示例 `self.focus();`

值 window对象引用。

默认值 当前窗口。

sidebar

请参见Components。

status

IE 3 NN 2 Moz all Saf all Op 7 DOM n/a

可读/可写

此属性可用于指定浏览器窗口状态栏中的文本。须要提示某些临时消息时就可以设定状态栏中的信息，例如当鼠标移动到某个图像上时可以给出提示。但为了防止链接欺骗（即链接上的文本显示为一个可信的站点，但其href却指向另一个恶意站点），当鼠标位于链接上时，现代浏览器倾向于不允许修改状态栏的内容。

此消息中的单、双引号必须进行转义，即“\'”。很多用户并不会在意状态栏中的附加信息，因此要避免将重要信息放置在此处。此外，在加载过程中或其他状态下，浏览器也会主动使用状态栏，此时就会和脚本中定义的临时消息发生冲突。如果要设置默认的状态栏消息，请参见defaultStatus属性。

示例

```
<...onmouseover="window.status='Table of Contents';return true"
onmouseout = "window.status = '';return true">
```

值 字符串。

默认值 空字符串。

statusbar

请参见directories。

toolbar

请参见directories。

top

IE 3 NN 2 Moz all Saf all Op 7 DOM n/a

只读

此属性是一个对应于浏览器窗口对象的引用。通过这个属性，内嵌在框架中的脚本指令可以引用浏览器窗口的属性和方法，以及那些保存在顶级窗口文档中的变量和方法。在实际应用中，不要从window.top建立引用，请直接使用top。如果要使用一个能够占据整个浏览器窗口的新文档来替代原有框架集，可以直接为top.location.href属性指定一个新的URL。

示例 `top.location.href = "tableOfContents.html";`

值 window对象引用。

默认值 浏览器窗口。

window

IE 3 NN 2 Moz all Saf all Op 7 DOM n/a

只读

此属性是一个对应于浏览器窗口对象的引用。

示例

```
if (window == top) {
    // 载入一个框架集
    location.href = "mainFrameset.html";
}
```

值 window对象引用。

默认值 浏览器窗口。

addEventListener( )

IE *n/a* NN *n/a* Moz *all* Saf *all* Op *7* DOM *n/a*

877

```
addEventListener("eventType", listenerFunction, useCapture)
```

尽管window对象并不是W3C DOM的官方内容，但主流的W3C DOM浏览器均为window对象实现了这个W3C DOM事件模型方法。请在本章前文罗列的共享方法中查看addEventListener()方法的相关讨论。

alert( )

IE *3* NN *2* Moz *all* Saf *all* Op *7* DOM *n/a*

```
alert("message")
```

此方法可以将参数传入的消息显示在一个警告对话框中。当这个对话框出现时，脚本也会停止执行。在对话框中，有个单独的按钮用来关闭这个对话框。此外，脚本并不能修改对话框窗口的标题栏，而在更早的浏览器版本中，还无法修改“JavaScript警告”图标。

返回值 无。

参数

message
: 任何字符串。

attachEvent( )

IE *5(Win)* NN *n/a* Moz *n/a* Saf *n/a* Op *7* DOM *n/a*

```
attachEvent("eventName", functionReference)
```

这个IE的事件模型方法已经被所有的元素对象所共享，于此同时，它也是window对象的成员方法之一。请在本章前文罗列的共享方法中查看attachEvent()方法的相关讨论。

back( )

IE *n/a* NN *4* Moz *all* Saf *n/a* Op *7* DOM *n/a*

跳转到当前窗口或框架的浏览历史列表中的后一个文档。实际上使用跨浏览器的history.back()方法是一种更好的选择。

返回值 无。

参数 无。

blur( )

IE *4* NN *3* Moz *all* Saf *all* Op *7* DOM *n/a*

从窗口中移除焦点并触发一个blur事件（仅限于IE浏览器）。这样一来其他元素就不会再接收到聚焦，但一旦打开了另一个浏览器窗口，那么当前窗口就会移动到窗口显示栈的末尾（只有在Opera中才不会给显示带来任何变化）。

返回值 无。

参数 无。

captureEvents( )

IE *n/a* NN *4* Moz *all* Saf *1.2* Op *7* DOM *n/a*

878

```
captureEvents(eventTypeList)
```

命令浏览器拦截一个特定类型的事件，以防止它们到达其预定目标对象。而调用这个方法的对象必须拥有针

对该事件类型的事件处理程序，以便处理捕获的事件。尽管这个方法最初只是Navigator 4事件模型的一部分，但Mozilla、Safari和Opera 等浏览器陆续为它提供了相关支持，它实际上已经为document对象创建了一种与W3C DOM捕获模式事件监听器相同的等价物。如果必须支持Navigator 4，那么可以继续使用这个方法，否则请使用在线参考VI中介绍的W3C DOM事件监听器语法规则。而Mozilla官方文档已经废弃了这个方法。

返回值 无。

参数

eventTypeList

一个使用逗号分隔的事件类型列表，对应的事件类型来自于可用的Event对象常量，如Event.CLICK或Event.MOUSEMOVE，注意区分字符大小写。

clearInterval( )

IE *4* NN *4* Moz *all* Saf *all* Op *7* DOM *n/a*

```
clearInterval(intervalID)
```

此方法可以关闭参数*intervalID*所指定的时间间隔循环动作。而window.setInterval()方法介绍了如何开启这种循环。

返回值 无。

参数

intervalID

调用setInterval()方法时返回的整数值。

clearTimeout( )

IE *2* NN *2* Moz *all* Saf *all* Op *7* DOM *n/a*

```
clearTimeout(timeoutID)
```

此方法可以关闭由*timeoutID*参数所指定的超时延迟计数器。而window.setTimeout()方法介绍了如何开启这种循环。

返回值 无。

参数

timeoutID

调用setTimeout()方法时返回的整数值。

close( )

IE *3* NN *2* Moz *all* Saf *all* Op *7* DOM *n/a*

此方法可以关闭当前窗口。在接收到用户通过安全对话框给出的明确允许之前，子窗口中的脚本无法关闭窗口。Mozilla和Safari浏览器会阻止窗口关闭其自身，而所有的浏览器均允许主窗口关闭它通过window.open()方法所打开的子窗口。

返回值 无。

参数 无。

confirm( )

IE *3* NN *2* Moz *all* Saf *all* Op *7* DOM *n/a*

```
confirm("message")
```

显示一个包含两个按钮和一段消息的对话框。当这个对话框出现时，脚本也会停止执行。在对话框的两个按钮中，一个表示“取消”操作，而另一个则接收用户的“确定”操作。但并不能通过脚本修改按钮上的文字。对话框中的文本消息应该提出一个问题，以便用户做出逻辑判断。当用户点击“取消”按钮时返回值为false，而点击“确定”按钮时则会返回true。

正因为这个方法会返回一个布尔值，因此可以将它放置在一个条件判断表达式中，如：

```
if (confirm("Reset the entire form?")) {
    document.forms[0].reset();
}
```

返回值 布尔值：true | false。

参数

message

任意字符串，通常是一个疑问句。

createPopup( )

IE *5.5* NN *n/a* Moz *n/a* Saf *n/a* Op *n/a* DOM *n/a*

这个方法将打开一个空白的弹出矩形窗口，可以使用HTML内容对其进行填充，而且它所在的空间可以位于框架边距之外甚至窗口边框之外。脚本必须通过这个方法返回的popup对象来指定具体内容（而不是一个外部URL）。其他详细信息，以及使用范例请参见popup对象。

返回值 popup对象引用。

参数 无。

detachEvent( )

IE *5(Win)* NN *n/a* Moz *n/a* Saf *n/a* Op *7* DOM *n/a*

```
detachEvent("eventName", functionReference)
```

这个IE的事件模型方法已经被所有的元素对象所共享，于此同时，它也是window对象的成员方法之一。请在本章前文罗列的共享方法中查看detachEvent()方法的相关讨论。

disableExternalCapture( ), enableExternalCapture( )

IE *n/a* NN *|4|* Moz *n/a* Saf *n/a* Op *n/a* DOM *n/a*

通过签名脚本和用户的明确许可，脚本可以捕获来自于域中其他窗口或框架的事件，但对那些在文档中使用事件捕获语法来进行处理的事件却无能为力。这个方法仅适用于Navigator 4。

返回值 无。

参数 无。

dispatchEvent( )

IE *n/a* NN *n/a* Moz *all* Saf *n/a* Op *7* DOM *n/a*

```
dispatchEvent(eventObjectReference)
```

尽管window对象并不是W3C DOM的官方内容，但Mozilla和Opera仍然为window对象实现了这个W3C DOM事件模型方法。请在本章前文罗列的共享方法中查看dispatchEvent()方法的相关讨论。

dump( )

IE *n/a* NN *n/a* Moz *1.4* Saf *n/a* Op *n/a* DOM *n/a*

```
dump(message)
```

这个方法可将参数传入的字符串输入到控制台或其他与标准输出（STDOUT）配套的设备中。第一次执行这个方法时，脚本必须从浏览器的配置首选项中启用这个方法。然而在新版本的浏览器中，再也不能通过“about:config”URI来访问这个选项关键字。

返回值 无。

参数

message

将要发送至STDOUT的消息字符串。

execScript( )

IE *4* NN *n/a* Moz *n/a* Saf *n/a* Op *n/a* DOM *n/a*

```
execScript(expressionList [, language])
```

此方法可以执行一个或多个由嵌入在浏览器中的脚本编程语言所编写的脚本表达式。这个表达式必须放置在一个单独的字符串中，多个表达式之间可以使用分号进行分隔，例如：

```
window.execScript("var x = 3; alert(x * 3)")
```

JavaScript语言是默认的脚本语言。如果须要查看脚本的执行结果，那么就须要像上例中一样，在脚本表达式中将处理结果显示出来。而execScript()方法本身并不会返回任何值。

返回值 无。

参数

expressionList

由一个或多个脚本表达式组成的字符串，不同表达式之间使用分号进行分隔。

language

表示脚本编程语言的字符串，目前可用的语言为：JavaScript | JScript | VBS | VBScript。

find( )

IE *n/a* NN *4* Moz *1.0.1* Saf *n/a* Op *n/a* DOM *n/a*

```
find("searchString"[, matchCase[, searchUpward]])
```

此方法可以在文档主体文本中查找一个字符串，并且选中第一个匹配项。根据参数设定，还可以在搜索时匹配字符大小写或从某个指定位置向上搜索文档内容。在Navigator 4中，可以使用document.getSelection()方法获取选中的匹配字符串的一个副本。而IE 4的TextRange对象并未提供类似的动态内容处理功能。

返回值 布尔值：如果找到匹配项，返回true，否则返回false。

参数

searchString

须要在文档中查找的字符串。

matchCase

布尔值：true表示查找时匹配字符大小写，而false表示可以忽略字符大小写。

searchUpward

布尔值：true表示从当前选中位置开始向上搜索文档，而false（默认值）则表示从当前选中位置开始向后搜索。

focus( )

IE *4* NN *3* Moz *all* Saf *all* Op *7* DOM *n/a*

此方法可以将对应窗口移至所有普通浏览器窗口之前，并在IE中触发onFocus事件。如果调用这个方法时另一个窗口恰好拥有焦点，那么该窗口就会收到一个onBlur事件。

返回值 无。

参数 无。

forward( )

IE *n/a* NN *4* Moz *all* Saf *n/a* Op *7* DOM *n/a*

跳转到当前窗口或框架的浏览历史列表中的前一个文档。如果并不包含任何前向浏览历史，那么这个方法并不会执行任何动作。

返回值 无。

参数 无。

GeckoActiveXObject( )

IE *n/a* NN *n/a* Moz *1.4* Saf *n/a* Op *n/a* DOM *n/a*


```
new GeckoActiveXObject("programID")
```

为允许Mozilla浏览器打开Windows媒体播放器的ActiveX控件而专门设计了这个构造方法，它对其他ActiveX控件没有任何作用。关于这一方法的实现细节请访问如下链接：*http://developer.mozilla.org/en/docs/Windows_Media_in_Netscape*。

返回值 播放器控件对象的引用，以便脚本对该播放器进行控制。

参数

programID

目标Windows媒体播放器控件的标识符。

getComputedStyle( )

IE *n/a* NN *n/a* Moz *all* Saf *n/a* Op *8* DOM *2*

```
getComputedStyle(elementNodeReference, "pseudoElementName")
```

根据第一个参数传入的元素对象引用，返回影响它的网状级联样式设定的样式对象（W3C DOM的术语称之为CSSStyleDeclaration）。如果要获取某个特殊的样式属性值，请首先使用这个方法获得一个样式对象，然后再调用该对象的getPropertyValue()方法，代码如下：

```
var compStyle = getComputedStyle(document.getElementById("myP"), "");
var pBGColor = compStyle.getPropertyValue("background-color");
```

请参见style对象以获取更多详细信息。 尽管Mozilla和Opera允许通过window对象方法这个方法，但W3C DOM更倾向于通过document.defaultView.getComputedStyle()来调用它。

返回值 style（CSSStyleDeclaration）对象引用。

参数

elementNodeReference

文档树中一个已变为选中域的元素节点的引用。

pseudoElementName

一个伪元素的名称（如first-line）或一个空字符串。

getSelection( )

IE *n/a* NN *n/a* Moz *all* Saf *all* Op *9* DOM *n/a*

返回Mozilla和Opera中的selection对象，然后可以将这个对象转化成为W3C DOM的Range对象。getSelection()替代了老旧的document.getSelection()方法，而Mozilla也反对使用后者。Safari浏览器则只会返回已被选中的文本。对应的IE操作是它的document.selection属性。关于如何使用选中文本的详细信息请参见selection对象。

返回值 selection对象引用。

参数 无。

home( )

IE *n/a* NN *4* Moz *all* Saf *n/a* Op *7* DOM *n/a*

此方法可以让浏览器跳转到主页设定的URL地址上。这与用户点击“主页”按钮的作用完全相同。

返回值 无。

参数 无。

moveBy( )

IE *4* NN *4* Moz *all* Saf *all* Op *7* DOM n/a

```
moveBy(deltaX, deltaY)
```

通过指定x、y两个轴向上的偏移量（单位为像素），这个方法可以很方便地改变当前窗口的具体位置。如果只沿着其中一条轴线移动，那么请将另一个值设置为0。此时，如果*deltaX*为正值，会让窗口向右侧移动，而负值则是向左侧移动。如果*deltaY*为正值，会让窗口向下移动，而负值则是向上移动。

返回值 无。

参数

deltaX

窗口在水平方向上的位置改变值，单位为像素。

deltaY

窗口在竖直方向上的位置改变值，单位为像素。

moveTo( )

IE 4 NN 4 Moz all Saf all Op 7 DOM n/a

```
moveTo(x, y)
```

这个方法可以很方便地将当前窗口移动到一个特定的坐标点上。值得注意的是，moveTo()方法中将使用屏幕坐标系。

返回值 无。

参数

x

与屏幕上端相对的正或负像素值。

y

与屏幕左边缘相对的正或负像素值。

navigate( )

IE 4 NN n/a Moz n/a Saf n/a Op 8 DOM n/a

```
navigate(URL)
```

为这个方法传入的参数可以让窗口或框架中载入新的文档。这是IE浏览器的特定方法，它与为window.location.href属性赋值所达到的效果完全相同。

返回值 无。

参数

URL

字符串形式的完整或相对URL。

open( )

IE 3 NN 2 Moz all Saf all Op 7 DOM n/a

```
open("URL", "windowName"[, "windowFeatures"])
open("URL", "windowName"[, "windowFeatures"][, replaceFlag])
```

这个方法在不关闭源窗口的同时会打开一个新窗口。此时可以指定一个URL以便新窗口进行加载，也可以将第1个参数设为空字符串，并通过脚本使用document.write()方法向窗口中写入内容。windowName参数可以为新窗口指定一个名称，从而让link或form元素的target属性使用。但这个名称并不能和框架名称一样用于脚本引用。如果脚本要引用一个子窗口，那么必须引用由window.open()方法返回的window对象。因此如果要保证脚本能够与这样创建的窗口进行通信，最好使用一个全局变量保存此方法的返回值，从而让其他脚本指令能够使用这个window对象。

通过这种方法创建的子窗口可能还会存在一个潜在的问题，当用户点击主窗口或使用脚本为主窗口赋予聚焦时，子窗口可能被主窗口挡住。因此在打开一个子窗口后应该为它调用一个focus()方法，以保证它在关闭前始终位于主窗口的前端。在连续调用window.open()方法时，如果后续方法中的windowName参数与之前的完全相同，那么就会自动找到已打开的窗口，即便该窗口位于主窗口下面也是如此，但此时并不会将该窗口移至前端。

方法中可选的第3个参数可以控制子窗口的多个实体属性。这个windowFeatures参数包含由多个属性/值对组成的字符串，每个属性/值对间使用逗号进行分隔，例如：

```
newWindow = window.open("someDoc.html","subWind",
"status,menubar,height=400,width=300");
newWindow.focus();
```

在默认情况下会开启窗口的所有属性，而且在打开一个新的子窗口时，子窗口的大小和通过浏览器的“文件”菜单打开的窗口大小完全相同。但一旦使用脚本为其中的某个属性进行赋值操作，那么所有的设定都会被关闭。因此请使用*windowFeatures*参数来指定需要开启的窗口特征。

如果在创建子窗口后立即引用该窗口，可能会引发某种问题，这看起来应该是一个时序问题（Timing Issue），而且它对IE/Windows的影响比其他组合大得多。从这一点看来，脚本应该是希望在窗口完全建立之后再对其进行引用。如果要解决这个问题，可以将引用子窗口的相关代码放置在一个独立的函数中，然后再通过setTimeout()方法调用该函数，通常暂停时间也仅仅只需要50毫秒而已。

而通过脚本来管理多个窗口往往比较困难。跨域的安全约束往往会阻止脚本执行的一些善意操作。在这种情况下，大多数浏览器会阻塞弹出窗口的调用，从而导致无法通过加载和卸载事件来促使浏览器打开一个单独的窗口。在Opera中，新窗口只能存在于主应用窗口之中，这一处理方式与其他浏览器有很大的不同。而用户也并不总是希望仅通过窗口本身来达到隐藏和显示窗口的目的，他们总想有一些其他的途径来控制窗口。因此如果用户使用更新的浏览器，那么请考虑使用其他可定位元素来模拟窗口。

返回值 window对象引用。

参数

885

URL

字符串形式的完整或相对URL。如果此参数为空字符串，那么就不会向窗口中加载任何文档。

windowName

target属性将要使用的一个窗口标识符。它与窗口中加载的文档的title属性并不相同。

windowFeatures

这个字符串参数表示将要在新窗口中开启的各种特征，不同特征间使用逗号进行分隔。而且在填写时请不要在逗号后使用空格。可以使用的特征列表很长，但其中的很多特征只适用于Navigator 4及其后续版本，而且由于这些特征关系到用户信息的保密性和安全性，因此使用它们时还需要签名脚本。下表中已列出了各种可用的特征。如果要开启一个窗口特征，那么请在此参数的字符串中加入它的名称，此时不需要考虑字符大小写。此外，只有那些能够说明空间尺寸的属性才须要明确赋值。

属性	IE	Mozilla	Safari	Opera	说明
alwaysLowered	n/a	<1.7	n/a	n/a	总是位于其他浏览器窗口之后。需要签名脚本
alwaysRaised	n/a	<1.7	n/a	n/a	总是位于其他的所有浏览器窗口之前。需要签名脚本
channelMode	4	n/a	n/a	n/a	在带有频道信息的剧院模式中进行显示
chrome	n/a	1.7	n/a	n/a	显示文档内容时不使用任何的装饰、用户界面功能或键盘命令。需要签名脚本
close	n/a	all	n/a	n/a	对于对话框类型的窗口，将它设置为no就可以删除关闭按钮。需要签名脚本
copyhistory	3	all	n/a	n/a	从源窗口中复制浏览历史列表到新窗口
dependent	n/a	all	n/a	n/a	如果关闭打开子窗口的窗口，那么也会关闭子窗口
dialog	n/a	1.2	n/a	n/a	隐藏窗口最大化及最小化控件
directories	3	all	n/a	n/a	显示目录按钮
fullscreen	4	n/a	n/a	n/a	隐藏标题栏或菜单
height	3	all	all	7	窗口高度，单位为像素
hotkeys	n/a	all	n/a	n/a	禁用菜单的键盘快捷键（除“退出”和“安全信息”之外）
innerHeight	n/a	all	n/a	n/a	定义内容区域的高度。每次均须要使用签名脚本
innerWidth	n/a	all	n/a	n/a	定义内容区域的宽度。每次均须要使用签名脚本
left	4	all	all	n/a	窗口左边缘距离屏幕左边缘的偏移量
location	3	all	all	n/a	文本域的显示位置（或地址）
menubar	3	all	n/a	n/a	显示菜单栏（在Mac中菜单栏总是可见的）
minimizable	n/a	1.2	n/a	n/a	在对话框形式的窗口中包含最小化控件
modal	n/a	1.2	n/a	n/a	打开一个模式对话框。需要签名脚本
outerHeight	n/a	all	n/a	n/a	窗口的总体高度。每次均须要使用签名脚本

886

续表

属性	IE	Mozilla	Safari	Opera	说明
outerWidth	n/a	all	n/a	n/a	窗口的总体宽度。每次均须要使用签名脚本
personalbar	n/a	all	n/a	n/a	与 Mozilla 中 directories 属性等价的特定属性
resizable	3	all	all	n/a	允许调整窗口大小（在 Mac 中总会允许窗口的大小调整）
screenX	n/a	all	n/a	n/a	窗口左边缘距离屏幕左边缘的偏移量。将窗口移出屏幕时须要使用签名脚本
screenY	n/a	all	n/a	n/a	窗口上边缘距离屏幕上边缘的偏移量。将窗口移出屏幕时须要使用签名脚本
scrollbars	3	all	all	n/a	如果窗口中的文档过大则显示滚动条
status	3	all	n/a	n/a	显示状态栏
titlebar	n/a	all	n/a	n/a	显示标题栏。将此值设置为 no 时会隐藏标题栏。需要签名脚本
toolbar	3	all	n/a	n/a	显示工具栏，其中包括返回、前进和其他按钮
top	4	all	all	7	窗口上边缘距离屏幕上边缘的偏移量
width	3	all	all	n/a	窗口宽度，单位为像素
z-lock	n/a	all	n/a	n/a	将新窗口固定在其他浏览器窗口之下。需要签名脚本

replaceFlag

仅适用于IE浏览器的一个布尔值，它可以控制新窗口的URL地址对浏览器中全局浏览历史列表的影响作用。将它设置为`true`时会使用新窗口的URL代替当前页面，此时将无法再通过“返回”按钮来访问当前页面，而将它设置为`false`则会将新窗口的URL正常地添加到浏览历史列表。

openDialog( )

IE *n/a* NN *n/a* Moz *1.0.1* Saf *n/a* Op *n/a* DOM *n/a*

```
openDialog("URL", "windowName"[, "windowFeatures"][, arg1[, arg2[, ...]]])
```

出于安全因素，这个方法只能用于XUL扩展。它实际上是`window.open()`方法的一个变种，但它为处理窗口特征提供了一些稍有不同的方法，并且还允许向新窗口传递一个或多个参数，这与IE的`showModalDialog()`方法有些类似。调用这个方法后，可以在子窗口加载的文档中通过`window.arguments`属性来访问这些已传入的参数。

返回值 `window`对象引用。

参数

URL

字符串形式的完整或相对URL。如果此参数为空字符串，那么就不会向窗口中加载任何文档。

windowName

`target`属性将要使用的一个窗口标识符。它与窗口中加载的文档的`title`属性并不相同。

windowFeatures

这个字符串参数表示将要在新窗口中开启的各种特征，不同功能间使用逗号进行分隔。有关详细信息，请参见`window.open()`。它还提供了另一个附加的特征值——`all`，通过它可以显示（`all=yes`）或隐藏（`all=no`）除`chrome`、`dialog`和`modal`之外的所有窗口特征。而剩余的这几个特征值可以单独进行控制。

arg1, arg2

任意形式的JavaScript值。子窗口中的脚本会将它们作为`window.arguments`数组中的条目进行访问。

print( )

IE *5* NN *4* Moz *all* Saf *all* Op *7* DOM *n/a*

这个方法可以为窗口或框架开启打印进程。在将文档发送至打印机之前，用户还必须确认打印对话框。这个方法的功效与点击浏览器的“打印”按钮或从“文件”菜单中选中“打印”功能完全相同。

返回值 无。

参数 无。

prompt( )

IE 3 NN 2 Moz all Saf all Op 7 DOM n/a

```
prompt("message", "defaultReply")
```

显示一个包含一段消息、一个单行文本输入域和两个按钮的对话框。当这个对话框出现时，脚本也会停止执行。这个消息应该要求用户在输入域中填写一个特定答案。在对话框的两个按钮中，一个表示“取消”操作，而另一个则表示用户“确定”将文本输入文本域中。但并不能通过脚本修改按钮上的文字。点击“取消”按钮后，这个方法会返回null，而点击“确定”按钮则会返回文本域中用户输入的所有内容，此时可能仅仅只是一个空字符串。获得返回值后，完全由脚本来测试用户提供的输入内容。

返回值 点击“确定”按钮，返回文本输入域中的文本，而点击“取消”时则返回null。

参数

message

任何字符串。

defaultReply

给出答案提示的任意字符串。注意，必须要为这个参数设置一个值，哪怕是一个空字符串。

releaseEvents( )

IE n/a NN 4 Moz all Saf 1.2 Op 7 DOM n/a

```
releaseEvents(eventTypeList)
```

此方法与window.captureEvents()相对，它根据参数列表中指定的特定事件名称，在窗口级关闭对应的事件捕获。尽管这个方法最初只是Navigator 4事件模型的一部分，但Mozilla、Safari和Opera 等浏览器陆续为它提供了相关支持，它实际上已经为document对象创建了一种与W3C DOM捕获模式事件监听器相同的等价物。如果必须支持Navigator 4那么可以继续使用这个方法，否则请使用在线参考VI中介绍的W3C DOM事件监听器语法规则。而Mozilla官方文档已经废弃了这个方法。

返回值 无。

参数

eventTypeList

一个使用逗号分隔的事件类型列表，对应的事件类型来自于可用的Event对象常量，如Event.CLICK或Event.MOUSEMOVE，注意区分字符大小写。

removeEventListener( )

IE n/a NN n/a Moz all Saf all Op 7 DOM n/a

```
removeEventListener("eventType", listenerFunction, useCapture)
```

尽管window对象并不是W3C DOM的官方内容，但主流的W3C DOM浏览器均为window对象实现了这个W3C DOM事件模型方法。请在本章前文罗列的共享方法中查看与removeEventListener()方法有关的讨论。

resizeBy( )

IE 4 NN 4 Moz all Saf all Op all DOM n/a

```
resizeBy(deltaX, deltaY)
```

根据参数指定的像素值，这个方法可以很方便地改变窗口的宽度和高度。如果只改变其中一个轴向数值，那么请将另一个值设置为0。*deltaX*为正值时会加宽窗口宽度，而负值则会使窗口变窄。与之类似，*deltaY*为正值时会使窗口更高，而负值则会缩短窗口。调用此方法后，左、上边缘会固定不动，只会根据设定移动右、下边缘。

返回值 无。

参数

deltaX

窗口在水平方向上的位置改变值，单位为像素。

deltaY

窗口在竖直方向上的位置改变值，单位为像素。

resizeTo( )

IE 4 NN 4 Moz all Saf all Op all DOM n/a

```
resizeTo(x, y)
```

根据参数指定的像素值，这个方法可以很方便地改变窗口的宽度和高度。调用此方法后，窗口的左、上边缘会固定不动，只会根据设定移动右、下边缘。

返回值 无。

参数

x

窗口的宽度，单位为像素。

y

窗口的高度，单位为像素。

routeEvent( )

IE n/a NN 4 Moz n/a Saf n/a Op n/a DOM n/a

```
routeEvent(event)
```

通常在事件处理方法中使用这个方法，它会命令Navigator 4让事件传递至其预期目标对象。

返回值 无。

参数

event

一个Navigator 4的`event`对象。

scroll( )

IE 4 NN 3 Moz all Saf all Op 7 DOM n/a

```
scroll(x, y)
```

此方法可以根据输入的参数设定当前窗口或框架内文档的滚动位置。如果要让这个文档返回到未曾滚动前的初始位置，请将两个参数均设置为0。

返回值 无。

参数

x

文档在水平方向上的滚动距离。

y

文档在竖直方向上的滚动距离。

scrollBy( )

IE 4 NN 4 Moz all Saf all Op 7 DOM n/a

```
scrollBy(deltaX, deltaY)
```

通过指定x、y两个轴向上的偏移量（单位为像素），这个方法可以将文档滚动到对应的具体位置。如果只改变其中一个轴向数值，那么请将另一个值设置为0。*deltaX*为正值时会将文档向左侧滚动，此时用户会观察到文档中的内容是向右边移动，而*deltaX*为负值时则会将文档向右移动。*deltaY*为正值时会将文档向上滚动，此时用户会观察到文档中的内容是向下移动，而*deltaY*为负值时则会将文档向下方移动。当文档滚动到零点后，就不能再继续滚动。

返回值 无。

参数

deltaX

文档在水平方向上滚动的距离，单位为像素。

deltaY

文档在竖直方向上滚动的距离，单位为像素。

scrollByLines( ), scrollByPages( )

IE *n/a* NN *n/a* Moz *all* Saf *n/a* Op *n/a* DOM *n/a*

```
scrollByLines(intervalCount) scrollByPages(intervalCount)
```

这两个方法可以将窗口向上（参数为正）或向下（参数为负）滚动若干行或页。这两个方法的作用与用户点击垂直方向上的箭头及“PageUp”、“PageDn”按键完全相同。

返回值 无。

参数

intervalCount

指定文档在竖直方向上滚动距离的数值。这个数字在`scrollByLines()`中表示滚动的行数，而在`scrollByPages()`中则表示页数。

scrollTo( )

IE *4* NN *4* Moz *all* Saf *all* Op *7* DOM *n/a*

```
scrollTo(x, y)
```

这个方法可以将窗口中的文档滚动到指定的位置，单位为像素。

返回值 无。

参数

x

窗口将要滚动到的水平位置，单位为像素。

y

窗口将要滚动到的竖直位置，单位为像素。

setInterval( )

IE *4* NN *4* Moz *all* Saf *all* Op *7* DOM *n/a*

```
setInterval("scriptExpression", msecs[, language])
setInterval(functionReference, msecs[, arg1, ..., argN])
```

此方法会开启一个计时器，从而在每个msecs周期内都调用一次脚本表达式。而在每次调用脚本表达式之间都可以运行其他脚本指令。如果要以固定的速率沿着某条路径移动一个元素，从而形成一个动画，那么这个方法就可以发挥它的作用。此时这个反复调用的动画函数可能和下面这句代码很相似：

```
intervalID = setInterval("advanceAnimation()", 500);
```

虽然这个方法的含义比较简单，但它所需的参数却着实令人迷惑。在最为简单的情况下，大多数跨浏览器应用都会在第2个参数规定的时间间隔（毫秒）内调用一个字符串形式的脚本函数。每次`setInterval()`方法被调用时，都会执行参数中指定的脚本表达式。因此，如果在这个表单式中置入了一些变量，那么在执行`setInterval()`方法前就应该设定好它们的值，即使在若干毫秒之后才使用这些变量也应该提前做好准备工作。

此外，IE浏览器还允许使用第3个参数来指定表达式允许时所需要的脚本编程语言。但如果这种语言不是VBScript就可以忽略这个参数。第1个参数也可以使用一个函数的引用，并传递一列以逗号进行分隔的参数列表以便该函数调用。这些参数将位于mssecs时间参数之后，并且可以使用任意形式的数据类型。

在实际应用中，应该使用一个全局变量保存这个方法返回的ID值，以便能够将它作为参数传入`clearInterval()`方法以便停止对应的计时循环。除非明确清除了定时处理循环，否则它会一直执行下去直到页面被卸载。

返回值 代表这个重复处理过程的整数标识符。

参数

scriptExpression

字符串形式任意脚本表达式，它通常是一个函数。函数的名称及括号都应该放置在参数的引号内。

functionReference

非字符串形式的函数引用（不包含括号的函数名称）或匿名函数。

msecs

调用scriptExpression或functionReference的时间间隔，单位为毫秒。

args

一个可选的以逗号分隔的参数列表，它将被传递到functionReference参数指定的函数之中。仅适用于Navigator浏览器。

language

一个可选的参数，它可以指明expression参数所用的脚本编程语言，默认语言为JavaScript。仅适用于IE/Windows。

setTimeout( )

IE *3* NN *2* Moz *all* Saf *all* Op *7* DOM *n/a*


```
setTimeout("scriptExpression", msecs[, language])
setTimeout(functionReference, msecs[, arg1, ..., argN])
```

这个方法可以在延迟msecs参数指定的时间段后调用一次scriptExpression或functionReference。当浏览器等待调用脚本表单式时还可以运行其他脚本。此时设置计时器的脚本指令可能和下面这句代码很相似：

```
timeoutID = setTimeout("finishWindow()", 50);
```

虽然这个方法的含义比较简单，但它所需的参数却着实令人迷惑。在最为简单的情况下，大多数跨浏览器应用都会在第2个参数规定的时间间隔（毫秒）内调用一个字符串形式的脚本函数。每次setTimeout()方法被调用时，都会执行参数中指定的脚本表达式。因此，如果在这个表单式中置入了一些变量，那么在执行setTimeout()方法前就应该设定好这些变量的值，即使在若干毫秒之后才使用这些变量也应该提前做好准备工作。

此外，IE浏览器还允许使用第3个参数来指定表达式允许时所需要的脚本编程语言。但如果这种语言不是VBScript就可以忽略这个参数。第1个参数也可以使用一个函数的引用，并传递一列以逗号进行分隔的参数列表以便该函数调用。这些参数将位于mssecs时间参数之后，并且可以使用任意形式的数据类型。

在实际应用中，应该使用一个全局变量保存这个方法返回的ID值，以便能够将它作为参数传入clearTimeout()方法以便在到时之前停止计时并调用被延迟的动作。

在某些情况下，setTimeout()可以起到和setInterval()方法相同的执行效果。例如，如果将setTimeout()方法放置在某个函数末尾，从而让该函数每次都调用同一个setTimeout()方法，那么就可以创建一个具有一个固定延时的函数循环调用。在setInterval()方法之前，早期的浏览器正是采取这种方式执行重复任务，例如在表单域或状态条中动态显示不断更新的数字时钟。

返回值　表示一个标识符的整数值。

参数

scriptExpression

字符串形式任意脚本表达式，它通常是一个函数。函数的名称及括号都应该放置在参数的引号内。

functionReference

非字符串形式的函数引用（不包含括号的函数名称）或匿名函数。

msecs

浏览器在调用脚本表达式之前所要等待的时间段，单位为毫秒。

args

一个可选的以逗号分隔的参数列表，它将被传递到*functionReference*参数指定的函数之中。仅适用于Navigator浏览器。

language

一个可选的参数，它可以指明expression参数所用的脚本编程语言，默认语言为JavaScript。仅适用于IE/Windows。

showHelp( )

IE 4(Win) NN n/a Moz n/a Saf n/a Op n/a DOM n/a

893

```
showHelp("URL")
```

使用WinHelp窗口显示URL参数指定的.*hlp*文档。

返回值 无。

参数

URL

字符串形式的完整或相对URL，用以表示一个WinHelp格式的文件。

showModalDialog( )

IE 4 NN n/a Moz n/a Saf 2.01 Op n/a DOM n/a

```
showModalDialog("URL"[, arguments[, "features"]])
```

这个方法可以显示一个特殊的窗口，在用户关闭这个窗口之前，它始终会位于所有的浏览器窗口之上。这种类型的窗口和通过window.open()方法生成的窗口有一定的区别。一旦打开了这种模式对话框窗口，它与源窗口之间就不会再有任何脚本关系。显示窗口内容的所有必要信息必须位于窗口加载的HTML文档之中或应该作为参数传入窗口。此后，模式对话框会让一段脚本设置它的returnValue属性，这个属性将作为showModalDialog()方法的返回值返回到打开该模式对话框的脚本指令中。

通过创建一个合适的数据结构就可以向模式对话框传递一系列的参数。对于只需要一个参数的情况，使用一个字符串就足矣。而对于多个参数值，既可以使用分界符将多个值放置在一个字符串内，也可以创建一个数组并将它作为showModalDialog()方法的第2个参数传入方法。此后，在模式对话框加载的文档中的一段脚本就可以检查window.dialogArguments属性，并根据须要解析这些变量。相关示例请参见dialogArguments属性。

此方法的第3个参数为一个可选参数，它可以设定对话框窗口的实际特征。而CSS样式语法则详细说明了这些特征。此时，dialogWidth、dialogHeight、dialogLeft和dialogTop等尺寸特征均应以像素为单位。调用一个模式对话框的简单范例如下所示：

```
var answer = window.showModalDialog("subDoc.html",argsVariable,
"dialogWidth:300px; dialogHeight:200px; center:yes");
```

在IE/Macintosh中，模式对话框在两个轴线上的尺寸至少是201像素。

须要注意的是，如果在模式对话框中加载框架集就会出现问题。在框架集中的框架内，脚本似乎无法引用父窗口或顶层窗口，也就无法访问窗口的close()方法或获得另一个框架中的内容。

返回值 在模式对话框窗口加载的文档内，为window.returnValue属性指定的属性值。

参数

URL

字符串形式的完整或相对URL。

894

arguments

数字、字符串或数组，此方法会将它们传递给模式对话框加载的文档。

features

使用分号分隔的各种样式属性及取值，它们将设置模式对话框的特征。各种可用的属性如下表所示。

特征	值	说明
center	yes、no、1、0、on、off	将对话框居中显示
dialogHeight	长度/单位	对话框的外部高度
dialogLeft	整数值	向左偏离中心位置的像素值
dialogTop	整数值	向上偏离中心位置的像素值
dialogWidth	长度/单位	对话框的外部宽度
edge	raised、sunken	边框和内容区域的过渡样式

续表

特征	值	说明
help	yes、no、1、0、on、off	在标题栏上显示帮助图标
resizable	yes、no、1、0、on、off	可调整对话框大小
status	yes、no、1、0、on、off	显示状态栏

showModelessDialog( )

IE *5(Win)* NN *n/a* Moz *n/a* Saf *n/a* Op *n/a* DOM *n/a*

```
showModelessDialog("URL"[, arguments[, "features"]])
```

这个方法可以显示一个特殊的窗口，它始终位于所有的浏览器窗口之上，但允许用户访问其他已打开的窗口及它们的内容。只要浏览器能够支持这个方法，那么它的参数和特征就和showModalDialog()方法完全一致。因此请参考showModalDialog()方法。

sizeToContent( )

IE *n/a* NN *n/a* Moz *all* Saf *n/a* Op *n/a* DOM *n/a*

这个方法可以让浏览器决定最合适的窗口尺寸以便显示窗口中的内容。它比较适用于显示一段有限信息的子窗口。

返回值 无。
参数 无。

stop( )

IE *n/a* NN *n/a* Moz *all* Saf *n/a* Op *7* DOM *n/a*

这个方法可以然窗口停止下载外部数据。它的作用与用户点击浏览器的“停止”按钮完全相同。

返回值 无。
参数 无。

xml

IE *5(Win)* NN *n/a* Moz *n/a* Saf *n/a* Op *n/a* DOM *n/a*

xml对象对应于微软专有的xml元素，它可以在一个HTML文档内创建一个XML数据岛。但这个功能在很大程度上已经被XMLHttpRequest对象所取代。

等价HTML元素 <xml>
对象模型引用方式 [window.]document.getElementById("*elementID*")
对象特定属性 src、XMLDocument
对象特定方法 无。
对象特定事件 无。

src

IE *5(Win)* NN *n/a* Moz *n/a* Saf *n/a* Op *n/a* DOM *n/a*

可读/可写

这个属性包含着数据岛内加载的外部XML文档的URL地址。如果要在事后加载一个新的文档，那么请为这个属性指定一个新的URL值。

示例 document.getElementById("xmlData").src = "xml/latestResults.xml";
值 完整或相对的URL字符串。
默认值 无。

XMLDocument

IE *5(Win)* NN *n/a* Moz *n/a* Saf *n/a* Op *n/a* DOM *n/a*

只读

这是微软的XML文档对象的一个引用。这个对象在很多方面都和W3C DOM的核心文档对象很类似，不过微

软提供了另一种不同的语法来从对象中读取和写入数据。详细信息请参考此链接：*http://msdn.microsoft.com/xml/reference/xmldom/start.asp*。

示例 `var xmlDoc = document.getElementById("XMLData").XMLDocument;`
值 一个微软XML文档对象的引用。
默认值 无。

XMLHttpRequest

IE *5(Win)* NN *n/a* Moz *all* Saf *1.2* Op *8* DOM *n/a*

`XMLHttpRequest`对象允许在当前已载入的Web页面和一个服务器之间实现后台通信，同时还不会打乱当前页面的显示效果。在典型情况下，服务器会响应浏览器的数据请求或数据提交，并且返回一个XML文档，而`XMLHttpRequest`对象会将该XML文档作为它的一个属性进行存储。在成功接收到XML文档之后，浏览器脚本可以解析这个XML数据，通常可以提取一些数据来更新当前Web页面的部分内容。具体的应用细节请参见在线参考VII。

除IE浏览器外，所有的浏览器均将`XMLHttpRequest`对象作为其固有的一个脚本对象。IE 5和IE 6则通过一个ActiveX控件来实现这个对象，但IE 7则引入了一个`XMLHttpRequest`对象，因此其他浏览器中使用的实例创建语法（调用`XMLHttpRequest()`构造方法）现在也可以在IE中正常工作。

从1.8版开始，Mozilla浏览器就为这个对象增加了一些属性、方法和事件，未来在其他浏览器中也可能会找到它们的身影。

对象模型引用方式 `new XMLHttpRequest()`
对象属性 `channel`、`multipart`、`readyState`、`responseBody`、`sponseText`、`onseXML`、`status`、`atusText`
对象方法 `abort()`、`addEventListener()`、`dispatchEvent()`、`getAllResponseHeaders()`、`getResponseHeader()`、`open()`、`overrideMimeType()`、`removeEventListener()`、`send()`、`etRequestHeader()`

对象事件

事件	IE/Windows	Mozilla	Safari	Opera	DOM
`error`	—	•	—	—	—
`load`	—	•	—	—	—
`progress`	—	•	—	—	—
`readystatechange`	•	•	•	•	—

channel

IE *n/a* NN *n/a* Moz *1.8* Saf *n/a* Op *n/a* DOM *n/a*

只读

`channel`属性可以返回一个通信频道的引用，`XMLHttpRequest`对象会使用这个频道来与服务器进行通信。下面这个链接给出了此属性的使用范例：*http://developer.mozilla.org/en/docs/Changing_the_Priority_of_HTTP_equests*。

值 一个内部通信频道的引用。
默认值 当前`XMLHttpRequest`对象的频道。

multipart

IE *n/a* NN *n/a* Moz *1.8* Saf *n/a* Op *n/a* DOM *n/a*

只读

Mozilla 1.8及后续版本允许从一个请求得到多个XML文档。如果要让`XMLHttpRequest`对象准备好执行这种处理模式，那么请将`multipart`属性设置为`true`。而服务器的第一个响应包必须拥有一个`multipart/x-mixed-`

replace类型的MIME值。客户端每收到一部分XML数据，XMLHttpRequest对象就会触发一个load事件。而在multipart流中的并发文档都会成为一个响应数据。

值 布尔值：true | false。
默认值 false

readyState

IE *5(Win)* NN *n/a* Moz *all* Saf *1.2* Op *8* DOM *n/a*

只读

返回一个对应于对象当前状态的整数代码。

值

整数值，其代表的含义如下表所示：

值	含义
0	未被初始化，即尚未调用 open() 方法
1	加载中，即尚未调用 send() 方法
2	已加载，即已调用了 send() 方法并且可以读取头部属性和状态属性
3	交互中，即正在进行传输，并且已经完成部分数据
4	完成，即所有传输操作已完毕

默认值 0

responseBody

IE *6* NN *n/a* Moz *n/a* Saf *n/a* Op *n/a* DOM *n/a*

只读

按照微软的“无符号字节数组”的形式返回响应的主体内容。由于这个数据并不是一种JavaScript值类型，因此它对Web页面开发者而言可能作用甚微甚至毫无作用。

值 字节数组。
默认值 无。

responseText

IE *5(Win)* NN *n/a* Moz *all* Saf *1.2* Op *8* DOM *n/a*

只读

以字符串的形式返回已接收的数据。

值 字符串。
默认值 无。

responseXML

IE *5(Win)* NN *n/a* Moz *all* Saf *1.2* Op *8* DOM *n/a*

只读

以一个XML document对象的形式返回已接收的数据。此后可以使用DOM节点树的解析技术（包括Mozilla中的E4X功能，请参见第5章）根据需要来提取数据，以便在网页中进行显示或进行进一步的计算工作。

值 document对象。
默认值 无。

status

IE *5(Win)* NN *n/a* Moz *all* Saf *1.2* Op *8* DOM *n/a*

只读

返回一个整数，它对应于一个来自服务器的响应代码。典型的响应代码为200（表示“OK”）或404（表示

“Not Found”）。值为200意味着服务器已经将所需的数据内容发送出来，而这部分数据所在的URL地址已经作为一个参数传递给get()方法。通过这个属性可以测试相关事务是否已经执行成功。

值 整数值。
默认值 0

statusText

IE *5(Win)* NN *n/a* Moz *all* Saf *1.2* Op *8* DOM *n/a*

只读

这个属性可以指明服务器返回的状态代码中的文字部分，如“OK”及“Not Found”。如果在脚本代码中使用了status属性，并且接收到一个无法理解的数字，那么请读取statusText属性来获得一个更为明晰的问题说明。

值 字符串。
默认值 空字符串。

abort( )

IE *5(Win)* NN *n/a* Moz *all* Saf *1.2* Op *8* DOM *n/a*

在调用send()方法之后才能调用这个方法，它会取消当前请求。此时status属性也会被设置为0。

返回值 无。
参数 无。

addEventListener( ), dispatchEvent( ), removeEventListener( )

IE *n/a* NN *n/a* Moz *1.4* Saf *n/a* Op *n/a* DOM *n/a*

Mozilla浏览器提供了一种DOM标准方式来管理已绑定到XMLHttpRequest对象上的事件，能够实现这一方式的部分原因是由于在Mozilla中有更多可以绑定的事件，而其他浏览器中仅实现了readystatechange事件。如果对这个对象特别感兴趣，那么在get()方法后就只能调用addEventListener()，这是因为get()方法会移除所有即将执行的事件监听者。请参见本章前文中的共享项以了解这3个方法的详细说明。

getAllResponseHeaders( )

IE *5(Win)* NN *n/a* Moz *all* Saf *1.2* Op *8* DOM *n/a*

返回从服务器获得的完整报头数据集合。每个报头数据都由若干个形式为“名称:值”的名称/值对组成，每个数据对都以回车换行符作为分界。例如：

```
ETag: "633bfe1-439-3d7a7301"
Content-Length: 1081
Keep-Alive: timeout=15, max=98
Content-Type: application/xml
Last-Modified: Sat, 07 Sep 2002 21:43:29 GMT
```

注意，在每个名称/值对末尾都有一个回车换行（CR/LF）。

返回值 字符串。
参数 空字符串。

getResponseHeader( )

IE *5(Win)* NN *n/a* Moz *all* Saf *1.2* Op *8* DOM *n/a*

```
getResponseHeader("headerName")
```

根据参数传入的字符串，返回其名称与该值相匹配的特定报头名称/值对的值。

返回值 字符串。
参数

headerName

由特殊的报头名称/值对中的名称组成的字符串。此时请不要将名称/值对的冒号也包含在内。

open( )

IE *5(Win)* NN *n/a* Moz *all* Saf *1.2* Op *8* DOM *n/a*

```
open("method", "URI"[, asyncFlag[, "username"[, "password"]]])
```

在准备实际请求时设置请求中的关键参数。此后会通过send()方法发送请求。

返回值 无。

参数

method

字符串形式的一个HTTP方法。请求数据时请使用GET，而在提交一个冗长的数据或SOAP命令时则应使用POST。

URI

远程数据或服务的统一资源标识符字符串。

asyncFlag

布尔值，用于指明是使用同步请求（false）还是异步请求（true）。在出现服务器或网络错误时，同步请求会让脚本陷入死锁。因此请使用异步请求，并让readystatechange或load事件触发脚本以处理已接收到的数据内容。默认值为true。

username

如果须要进行身份验证，那么就使用这个包含用户名的字符串。当脚本中包含一些可以被他人看到的敏感信息时要谨慎使用这个参数。

password

如果须要进行身份验证，那么就使用这个包含密码字符串。当脚本中包含一些可以被他人看到的敏感信息时要谨慎使用这个参数。

overrideMimeType( )

IE *5(Win)* NN *n/a* Moz *all* Saf *1.2* Op *8* DOM *n/a*

```
overrideMimeType("MIMEType")
```

在get()和send()方法之间调用这个方法，以迫使对象在接收自服务器的数据上应用特定的MIME类型。当服务器的输出并未被配置为XML内容类型（如text/xml）时就须要使用这个方法。

返回值 无。

参数

MIMEType

字符串形式的目标内容类型。XMLHttpRequest对象为了将数据加载至responseXML属性，必须以有效的XML内容类型来接收这些数据，如text/xml，也可以使用这个方法来指定具体的类型来达到相同的目的。

send( )

IE *5(Win)* NN *n/a* Moz *all* Saf *1.2* Op *8* DOM *n/a*

```
send(data)
```

根据open()方法中确定的规范，将数据发送至服务器端。

返回值 无。

参数

data

在数据发送请求时，有两种格式能够被所有浏览器所支持，而IE浏览器还支持另外一种额外的格式。各种浏览器都能够接受字符串格式或DOM的document对象格式。在开始传输前就会在内部对后者进行串行化处理，即将一个document对象转换成一个字符串。IE还允许使用字节数组作为请求数据。须要注意的是，这个数据中并不包含任何报头信息，须要通过setRequestHeader()方法来设置这些内容。

setRequestHeader( )

IE *5(Win)* NN *n/a* Moz *all* Saf *1.2* Op *8* DOM *n/a*

```
setRequestHeader("headerLabel", "value")
```

在open()和send()方法之间调用这个方法，它可以让脚本在即将发送的请求数据中分配一个或多个报头名称/值对。

返回值 无。

参数

headerLabel

作为报头信息的名称/值对中的名称部分。

value

作为报头信息的名称/值对中的值部分。

XMLSerializer

IE *n/a* NN *n/a* Moz *1.0.1* Saf *1.2* Op *8* DOM *n/a*

XMLSerializer对象将一个XML文档或节点对象转换成一个能够包含所有标记的字符串。浏览器会在内部使用这个对象来执行某些任务，如将XML内容转换成一个字符串，然后再通过`XMLHttpRequest.send()`方法发送该字符串数据。但外部脚本也可以使用这个对象来完成其他任务。虽然Mozilla暴露了这个对象的`serializeToStream()`方法，但比较奇怪的是，它的参数之一却使用了一种客户端脚本无法使用的数据类型。

对象模型引用方式 `new XMLSerializer()`

对象属性 无。

对象方法 `serializeToString()`

对象特定事件 无。

serializeToString( )

IE *n/a* NN *n/a* Moz *1.0.1* Saf *1.2* Op *8* DOM *n/a*

`serializeToString(`*DOMNode*`)`

根据参数传入的DOM节点，返回构成该节点的标记信息。这个节点既可以是整个文档（nodeType为9），也可以是其他的节点类型，其中还还包括文档片段。但在Safari 2.0.2之前只能使用完整的文档节点。

返回值 字符串

参数

DOMNode

任意类型的XML节点对象，Safari 2.0.2版之前必须是文档节点。

xmp

请参见pre。

第3章 事件参考

Event Reference

本章列出了当前主流浏览器中已实现的每一种事件类型，以及W3C规范中DOM Level 2事件模型的一系列专有事件。根据W3C DOM的事件模型中的格式，这些事件均依其类型名称按字母顺序进行排列。当使用元素对象属性或IE的`attachEvent()`方法来绑定事件时须要在这些事件类型名称前添加“on”前缀。在接下来的内容里列出了每个事件的版本信息，这样读者就可以很方便地检查是否能够在必须支持的浏览器中使用某个事件。在这些版本信息中可以很清晰地看到，每个事件是在 Internet Explorer（简称IE）、早期的Mozilla网景浏览器（Netscape Navigator，简称NN）、Mozilla浏览器（简称Moz）、Safari（简称Saf）、Opera（简称Op），以及W3C DOM规范的哪个版本中被首次引入。

如果IE中的某个事件标明了“Win”或“Mac”，那么就意味着该事件只支持Windows或Macintosh操作系统。此外，IE 5.5及后续版本的浏览器只适用于Windows系统。须要注意的是，有大量的事件类型只支持IE/Windows组合，而且它们中的大多数只能在数据绑定应用中使用。因此，如果要在多个浏览器下部署Web应用，就须要仔细考虑下文中列出的浏览器兼容性信息，以便找到在不同浏览器品牌和版本中都能正常工作的事件。在线参考VI中列出了一些指导方针和示例，该文对在多个浏览器类型中混合使用各种互不兼容的事件机制提供了许多帮助。

在下文列表中，“冒泡”指明了事件是否遵循事件冒泡传播（必须在支持事件冒泡的浏览器中才能够实现，请参见在线参考VI），而“可取消”则表示是否能够通过脚本指令取消事件的默认动作从而避免某些正常操作，如点击一个a元素时导航到一个新的URL地址。此外，“典型目标”通常指出了事件类型可用于哪些元素类型。而每种事件类型的特定元素支持请查阅第9章。

按字母排序的事件参考

abort

IE *3* NN *4* Moz *all* Saf *all* Op *all* DOM *2*

冒泡：否；可取消：否

如果由于用户的干扰（如点击“停止”按钮或立即跳转到另一个页面）或其他问题（如网络传输超时）导致无法完全加载`img`元素的内容，那么就会触发这个事件。W3C DOM只在`object`元素中应用了这个事件。在W3C的规范中，使用`object`是在页面中嵌入图像的一种理想方式，但这种方式并未在浏览器中得到广泛的支持。

典型目标 `img`元素。

activate

IE *5.5* NN *n/a* Moz *n/a* Saf *n/a* Op *n/a* DOM *n/a*

冒泡：是；可取消：否

当一个对象变为活动对象时会触发这个事件。此时通常会为对象赋予焦点，但已显示的元素在未拥有焦点的情况下也可以是活动元素。在一个页面中，同一时刻只能存在一个活动元素。相关信息请参考第2章中的共享对象的`setActive()`方法。在一个元素获得焦点后，会在`focusin`之前触发`activate`事件，然后才是`focus`事件。

典型目标 所有的已显示元素，以及document和window对象。

afterprint, beforeprint

IE *5 (Win)* NN *n/a* Moz *n/a* Saf *n/a* Op *n/a* DOM *n/a*

冒泡：否； 可取消：否

当用户在打印对话框中点击“打印”按钮之后，在为打印机组合打印内容前会触发beforeprint事件，当所有数据被发送到打印机后，则会触发afterprint事件。可以使用这两个事件来触发修改样式单或其他页面显示内容的函数，从而使得打印出的页面与原页面有一定的不同，然后再在屏幕上恢复页面显示。因此这个技术可用于替代样式单媒介设定。

典型目标 body和frameset元素及window对象。

afterupdate

IE *4(Win)* NN *n/a* Moz *n/a* Saf *n/a* Op *n/a* DOM *n/a*

冒泡：是；可取消：否

当通过IE的数据绑定机制将数据发送至可写的数据源对象并成功更新数据库后会触发这个事件。

典型目标 所有能够接受数据输入并支持数据绑定的元素。

beforeactivate

IE *6* NN *n/a* Moz *n/a* Saf *n/a* Op *n/a* DOM *n/a*

冒泡：是；可取消：偶尔

在一个对象成为活动对象前会触发这个事件。此时通常会为对象赋予焦点，但已显示的元素在未拥有焦点的情况下也可以是活动元素。在一个页面中，在同一时刻只能存在一个活动元素。请参考第2章中的共享对象的setActive()方法。如果一个元素获得了焦点，那么会按照下述顺序触发相关事件：beforeactivate、activate、focusin最后是focus。

如果取消了beforeactivate事件，那么该元素就不会变为活动元素，也不会获得焦点，但当用户直接点击或进行tab操作时还是会为元素赋予焦点。如果阻止某个元素获得焦点，那么就会自动在另一个元素中进行聚焦，当用户跳格选中被阻止的元素时会自动选中跳格移动顺序中的下一个可聚焦元素，而当用户点击了已被阻止的元素时，则会选中文档树中临近的最外层可聚焦父级元素。此外，如果使用setActive()或focus()方法来激活元素或为元素赋予焦点，那么通过取消这个事件也无法达到阻止聚焦的效果。

典型目标 所有的已显示元素，以及document和window对象。

beforecopy

IE *5 (Win)* NN *n/a* Moz *n/a* Saf *1.3/2* Op *n/a* DOM *n/a*

冒泡：是；可取消：是

无论用户是通过“编辑”菜单、键盘快捷键还是上下文菜单来将选中的内容复制到系统剪切板，都会在启动“复制”命令前触发这个事件。从这点上来看，由这个事件处理程序调用的函数可以执行正常的系统复制动作之外的其他附加动作或替代处理过程。例如，选中文本所在元素的一些附加信息（如有效的样式信息）就可以保存在IE的clipboardData对象（请参见第2章）中，而在beforepaste的事件处理程序中就可以处理这些信息。须要注意的是，取消beforecopy事件也无法阻止用户复制选中的内容。

在IE中，会在用户显示上下文菜单时（在选中任意的菜单选项之前）触发“before”类的事件。而事件会按照beforecut、beforecopy，然后是beforepaste的顺序依次触发。此外，在使用上下文菜单时，这3个事件组成的事件序列会被触发两次。

须要注意的是，在Safari 1.3/2中必须将onbeforecopy传递给addEventListener()来绑定事件，而不是beforecopy。

典型目标 除表单控件外的所有已显示元素。

beforecut

IE *5 (Win)* NN *n/a* Moz *n/a* Saf *1.3/2* Op *n/a* DOM *n/a*

冒泡：是；可取消：是

无论用户是通过“编辑”菜单、键盘快捷键还是上下文菜单来将选中的内容剪切到系统剪切板，都会在启动“剪切”命令前触发这个事件。从这点上来看，由这个事件处理程序调用的函数可以执行正常的系统剪切动作之外的其他附加动作或替代处理过程。例如，被选中文本所在元素的一些附加信息（如有效的样式信息）就可以保存在IE的clipboardData对象（请参见第2章）中，而在beforepaste的事件处理程序中就可以处理这些信息。须要注意的是，取消beforecut事件也无法阻止用户剪切选中的内容。

906

关于浏览器的特定注意事项请参考beforecopy事件。

典型目标 所有已显示的元素。

beforedeactivate

IE *5.5* NN *n/a* Moz *n/a* Saf *n/a* Op *n/a* DOM *n/a*

冒泡：是；可取消：是

无论是用户点击了另一个元素、跳格选中了另一个元素还是通过脚本调用了setActive()或focus()方法，在对象将其活动性交于对应的元素前均会触发这个事件。如果一个元素拥有焦点并且是一个活动元素，那么会按照下述顺序触发相关事件然后才失去焦点：beforedeactivate、deactivate及blur。由于可以取消beforedeactivate（但无法取消deactivate），因此取消这个事件就可以阻止元素失去活动性或焦点。如果有充分的理由这样做，并且不会给访问者带来困惑，那么这的确是一种可行的方法。

典型目标 所有的已显示元素，以及document和window对象。

beforeeditfocus

IE *5 (Win)* NN *n/a* Moz *n/a* Saf *n/a* Op *n/a* DOM *n/a*

冒泡：是；可取消：是

当用户通过点击或跳格选中某个可编辑元素，从而使它获得正式焦点前会触发这个事件。此处的可编辑元素包括面向文本的表单控件及可编辑的元素（请参见第2章中与所有元素相关的IE 5.5的contentEditable属性）。由这个事件的处理程序调用的函数可以执行一些附加的脚本动作，如在用户开始编辑前设定元素中文本的颜色。

典型目标 文本表单控件、处于编辑模式的已显示的元素（IE 5.5或后续浏览器才能够支持），以及由DHTML ActiveX编辑控件控制的内容（请参见*http://msdn.microsoft.com/en-us/library/aa741317.aspx*）。

beforepaste

IE *5(Win)* NN *n/a* Moz *n/a* Saf *1.3/2* Op *n/a* DOM *n/a*

冒泡：是；可取消：是

无论用户是通过“编辑”菜单、键盘快捷键还是上下文菜单来将选中的内容粘贴到系统剪切板，都会在启动“粘贴”命令前触发这个事件。如果须要从clipboardData对象中粘贴自定义信息（保存在beforecopy、copy、beforecut或cut事件处理程序中），那么须要让beforepaste和paste事件处理函数一起协同工作。此时，在beforepaste事件处理函数中将event.returnValue设置为false后就可以激活“编辑”菜单或上下文菜单中的“粘贴”一栏。当用户点选了“粘贴”菜单项时，paste事件处理程序就会从clipboardData对象中获取信息，此时可能会改变已选中元素的HTML内容，例如：

907

```
function handleBeforePaste() {
    event.returnValue = false;
}
function handlePaste() {
    if (event.srcElement.className == "OK2Paste") {
        event.srcElement.innerText = clipboardData.getData("Text");
    }
}
```

在这个粘贴操作中，由于脚本完成了所有的数据转换工作，也不须要进入编辑模式，因此并未使用系统剪切板。

关于浏览器的特定注意事项请参考beforecopy事件。

典型目标 所有已显示的元素，以及document对象。

beforeprint

请参见afterprint。

beforeunload

IE *4 (Win) /5 (Mac)* NN *n/a* Moz *1.7* Saf *1.3/2* Op *n/a* DOM *n/a*

冒泡：否；可取消：是

无论当前页面将要跳转到一个新的页面、进行提交表单或关闭窗口，在卸载当前文档前都会触发这个事件。这个事件的触发时间早于unload事件，这样就使得脚本及用户可以取消卸载动作。某些情况下会自动进行处理，从而防止恶意代码在页面中捕获用户操作。

在beforeunload的事件处理程序中，可以为event.returnValue属性指定一个字符串，从而让IE、Mozilla 1.8及后续版本显示一个对话框。用户通过这个对话框就可以选择是让页面继续停留在其所在的位置，还是根据用户请求执行跳转动作或关闭页面窗口。此时为事件属性赋予的字符串会成为对话框中消息的一部分，而对话框中的其他内容则已固化在浏览器中，无法通过脚本进行修改或删除。而用户点击对话框中的不同按钮就可以选择最终的执行动作。

典型目标 body和frameset元素，以及window对象。

beforeupdate

IE *4 (Win)* NN *n/a* Moz *n/a* Saf *n/a* Op *n/a* DOM *n/a*

冒泡：是；可取消：是

在通过IE的数据绑定机制将数据发送至可写的数据源对象前会触发这个事件。此时可以执行数据校验并取消数据更新。

典型目标 所有能够接受数据输入并支持数据绑定的元素。

blur

IE *3* NN *2* Moz *all* Saf *all* Op *all* DOM *2*

冒泡：否；可取消：否

在当前元素失去焦点或调用了blur()方法后会触发这个事件。在其他元素中，会在focus事件之前触发blur事件。

值得注意的是，在文本输入域中进行表单校验时应避免使用blur事件，如果校验程序在发现错误时会显示一个警告对话框，那么更不应使用这个事件。如果此时使用了这个事件，那么blur和focus事件之间的交互作用会导致一个无限循环，不断显示然后隐藏警告对话框。因此请使用change来代替这个事件。

尽管脚本浏览器从很早开始就能在表单控件和window对象中使用blur事件，但如果在其他元素中设置了tabindex属性，那么现在的浏览器实质上也可以在这类已显示的元素中触发该事件。须要注意的是，IE/Windows并不会在window对象上触发blur事件。

典型目标 在所有的浏览器中可以应用于input（文本或密码类型）、textarea、select及window对象，而在IE 5及后续浏览器和W3C DOM的浏览器中，所有已设定了tabindex属性值的已显示元素均可使用这个事件。

bounce

IE *4* NN *n/a* Moz *n/a* Saf *n/a* Op *n/a* DOM *n/a*

冒泡：否；可取消：是

如果marquee元素的移动特性设置为alternate，那么每当其中的文本接触到元素边缘并改变运动方向时就会触发这个事件。

典型目标 marquee元素。

cellchange

IE *5 (Win)* NN *n/a* Moz *n/a* Saf *n/a* Op *n/a* DOM *n/a*

冒泡：是；可取消：否

如果元素是一个数据绑定的数据源对象，那么每当远程数据库中的数据值发生改变时就会触发这个事件。

典型目标 object和applet元素。

change

IE *3* NN *2* Moz *all* Saf *all* Op *all* DOM *2*

冒泡：否（IE）；是（其他）；可取消：是（IE）；否（其他）

当一个文本表单控件或select元素失去焦点，并且其文本内容或选中项与元素在获得焦点时相比发生改变时就会触发这个事件。因此，在文本类型的input和textarea元素中使用这个事件可以对用户的输入内容进行校验。但我们仍然建议在form元素的submit事件处理程序中进行表单范围的数据校验，以保证最终数据的正确性。另外，这个事件触发时会早于blur事件。

典型目标 文本类型的input、textarea元素，以及select元素。

click

IE *3* NN *2* Moz *all* Saf *all* Op *all* DOM *2*

冒泡：是；可取消：是

当用户执行了一次鼠标点击或其他等效操作时会触发这个事件。一般情况下，如果一个可聚焦的元素（如按钮和链接）获得了焦点，那么此时按下“Enter”键或空格键就会产生等效的点击事件。而在支持accesskey的现代浏览器中，按下组合键也等效于一次点击事件。

对于鼠标点击动作而言，只有在同一个元素上按下并释放鼠标按键才会触发click事件。在这种情况下，鼠标事件会按照mousedown、mouseup和click的顺序依次触发。

在鼠标事件产生的event对象中，会将一些细节信息赋予其各个属性，如点击发生的坐标，事件发生时是否按下了修改键等。此外，通过mouseupdown或mouseup事件来获取按钮的信息往往更为可靠。事件处理函数可以获取这些属性的值以便掌握所需的信息。

尽管脚本浏览器从很早开始就能在表单按钮控件和链接对象中使用click事件，但现在的浏览器实质上也可以在其他已显示的元素中触发该事件。须要注意的是，在1.4版的Mozilla浏览器和1.3版的Safari浏览器之前，可以在容器类元素的子文本节点中触发鼠标事件，这意味着event对象的target属性可以引用节点对象而不是整个容器元素。关于如何利用这些特性并实现跨浏览器的实施方案请参见在线参考VI。

典型目标 在所有的浏览器中均可以将button、radio、checkbox、reset和submit类型的input元素、a元素，以及area对象作为目标；IE 4及后续版本还支持document和window对象；在IE 4之后的IE浏览器，以及除Opera 9之外的W3C DOM浏览器中，则可以将任意一个已显示的元素作为目标，而1.4版之前的Mozilla浏览器还可以在文本节点中使用这个事件。

contextmenu

IE *5 (Win)* NN *n/a* Moz *all* Saf *all* Op *n/a* DOM *n/a*

冒泡：是；可取消：是

在用户点击鼠标右键或鼠标控制面板中的第2个鼠标按钮后会触发这个事件。此时会在光标所在的位置为对应的页面内容显示上下文菜单。在Mozilla 1.4和Safari 1.3之前，事件目标也可以是一个文本节点。对于所有其他支持这一事件的浏览器和版本而言，事件目标或IE的`event.sourceElement`值往往是容器元素。如果要阻止上下文菜单的显示，那么请在`contextmenu`事件处理程序中将`event.returnValue`设置为`false`。但是隐藏上下文菜单可能会导致用户难于查看页面代码或保存图片，因此这并不是防止用户获取页面内容的有效方法。此外，如果用户禁用脚本指令，那么任何脚本解决方案也会随之失效。

910

典型目标 所有已显示的元素，以及`document`对象。

controlselect

IE *5.5* NN *n/a* Moz *n/a* Saf *n/a* Op *n/a* DOM *n/a*

冒泡：是；可取消：是

当页面处于编辑模式下，如果用户选择了一个可编辑元素（而不是其内容）那么就会触发这个事件。关于此事件的使用范例请参考`move`事件。

典型目标 所有已显示的元素，以及`document`对象。

copy

IE *5 (Win)* NN *n/a* Moz *n/a* Saf *1.3/2* Op *n/a* DOM *n/a*

冒泡：是；可取消：是

无论用户是通过“编辑”菜单、键盘快捷键还是上下文菜单来将选中的内容复制到系统剪切板，都会在启动“复制”命令后触发这个事件。此时这个事件的处理函数就可以将选中的一些附加数据放入`clipboardData`对象中，从而为复制动作给予补充，如果需要就可以在`paste`事件的处理程序中读出并处理这些数据。

如果要为其他非可编辑元素提供“复制”菜单命令，那么针对`copy`事件处理程序中的同一个对象，须要在`beforecopy`事件处理程序中将对应的`event.returnValue`设置为`false`。从另一方面来看，如果要防止用户复制页面中的主体内容，那么须要为`copy`事件处理程序将`event.returnValue`设置为`false`。但如果要防止用户复制重要的页面内容，请不要将这种方法视为牢不可破的解决途径。

此外还须要注意的是，在Safari 1.3/2中必须将`oncopy`传递给`addEventListener()`来绑定事件，而不是`copy`。

典型目标 除表单控件外的所有已显示元素。

cut

IE *5 (Win)* NN *n/a* Moz *n/a* Saf *1.3/2* Op *n/a* DOM *n/a*

冒泡：是；可取消：是

无论用户是通过“编辑”菜单、键盘快捷键还是上下文菜单来将选中的内容剪切到系统剪切板，都会在启动“剪切”命令后触发这个事件。如果要剪切页面中的主体内容，那么其容器元素必须处于编辑模式，请参见第2章中的共享属性`contendEditable`。此时这个事件的处理函数就可以将选中的一些附加数据放入`clipboardData`对象中，从而补充剪切动作，如果须要就可以在`paste`事件的处理程序中读出并处理这些数据。

如果要为其他非可编辑元素提供“剪切”菜单命令，那么针对`cut`事件处理程序中的同一个对象，须要在`beforecut`事件处理程序中将对应的`event.returnValue`设置为`false`。从另一方面来看，如果要防止用户剪切页面中的主体内容，那么须要为`cut`事件处理程序将`event.returnValue`设置为`false`。

911

须要注意的是，在Safari 1.3/2中必须将`oncut`传递给`addEventListener()`来绑定事件，而不是`cut`。

典型目标 所有已显示的元素。

Dataavailable, datasetchanged, datasetcomplete

IE *4 (Win)* NN *n/a* Moz *n/a* Saf *n/a* Op *n/a* DOM *n/a*

冒泡：是； 可取消：否

在一个作为数据绑定的数据源对象的元素中，当远程数据源通知它已经有可供查询的新数据（dataavailable）、修改了某个数据集（datasetchanged）或某个数据集已从查询中获取了所有的数据（datasetcomplete）等情况时，就会触发这个事件。此外，只能在那些能够与对应的远程数据进行异步链接的数据源对象中使用这些事件。

典型目标 object和applet元素，以及IE 5及后续版本中的xml元素。

dblclick

IE *4* NN *4* Moz *all* Saf *n/a* Op *all* DOM *n/a*

冒泡：是；可取消：是

当用户连续两次点击鼠标或进行了等价的操作（请参见click）后会触发这个事件。对双击事件而言，在触发它之前需要一系列的特殊导引事件。这个特殊的事件序列为：mousedown、mouseup、click、mouseup，最终为dblclick。而两次点击之间所需的具体时间间隔则由客户端计算机中鼠标控件控制面板来决定。

由于会在dblclick事件前触发click事件，因此相关的click事件处理程序只应该执行一些对dblclick没有什么影响的操作，如高亮显示一个元素等，这一过程与双击操作系统桌面上的图标比较类似。如果须要进行一次双击动作，那么就只能依据这个事件来执行一些有效的操作。在Mozilla 1.4和Safari 1.3之前，事件目标也可以是一个文本节点。而对于所有其他支持这一事件的浏览器和版本而言，事件目标或IE的event.sourceElement值往往是容器元素。

典型目标 除Macintosh之外的Navigator 4浏览器在a元素中支持这个事件，IE 4及后续版本，以及W3C DOM浏览器（除Safari 2以来的所有Safari浏览器之外）可以在可显示的元素及document对象中支持这个事件。

deactivate

IE *5.5* NN *n/a* Moz *n/a* Saf *n/a* Op *n/a* DOM *n/a*

冒泡：是；可取消：否

无论是用户点击了另一个元素、跳格选中了另一个元素还是通过脚本调用了setActive()或focus()方法，在对象将其活动性交于对应的对象后就会触发这个事件。如果一个元素拥有焦点并且是一个活动元素，那么会按照下述顺序触发相关事件，然后失去焦点：beforedeactivate、deactivate及blur。如果要阻止一个元素失去焦点或活动性，可以取消对应的beforedeactivate事件。

典型目标 所有的已显示元素，以及document和window对象。

DOMActivate

IE *n/a* NN *n/a* Moz *n/a* Saf *1.3/2* Op *n/a* DOM *2*

冒泡：是；可取消：是

当用户开始与一个元素进行交互时就会触发这个事件，如点击一个按钮或在一个文本框内输入字符。事件对象的detail属性会传出一个整数值，它包含该事件的更多信息：1表示未修正的单次点击；2表示按下修正键或双击所带来的相关动作，DOM规范称之为“超活动性”。

典型目标 所有可以正常接收焦点或已为tabindex属性赋值的已显示元素均可作为事件目标。

DOMAttrModified

IE *n/a* NN *n/a* Moz *all* Saf *n/a* Op *7* DOM *2*

冒泡：是；可取消：否

一旦通过脚本添加、删除或改变了某个元素的属性，那么就会触发这个事件。此时，必须通过`setAttribute()`或`removeAttribute()`方法来添加或删除元素属性，但也可以通过`setAttribute()`方法或为对应的元素对象属性指定一个新值来修改已有属性的属性值。这个事件对象的很多属性都可以传达事件的详细信息。请参见第2章中`event`对象中的下列属性：`attrName`、`attrChange`、`prevValue`、`newValue`，以及`relatedNode`。

典型目标 所有元素。

DOMCharacterDataModified

IE *n/a* NN *n/a* Moz *n/a* Saf *n/a* Op *n/a* DOM *2*

冒泡：是；可取消：否

当脚本改变了节点的`CharacterData`类型值时会触发这个事件。请参见第2章中`event`对象中的`prevValue`和`newValue`属性。

典型目标 `CharacterData`节点。

DOMContentLoaded

IE *n/a* NN *n/a* Moz *1.0.1* Saf *n/a* Op *9* DOM *n/a*

冒泡：是；可取消：是

当浏览器解析了所有的HTML源代码并在DOM中表示出所有的元素后就会立即触发这个事件。事件触发会在加载外部内容（如图像、声音文件等）之前完成，因此在触发`load`事件还需要一些时间完成加载工作。使用这个事件来替代`load`就可以允许DHTML脚本在加载外部内容的同时就开始修改文档树。但如果用户通过浏览器的历史菜单（如“返回”或“前进”按钮）来进行页面跳转则不会触发这个事件。

典型目标 `body`、`document`或`window`这3个DOM对象。

DOMControlValueChanged

IE *n/a* NN *n/a* Moz *n/a* Saf *n/a* Op *9* DOM *2*

冒泡：是；可取消：否

当Web Forms 2.0的表单控件值发生改变时就会触发这个事件。如果将这个事件与`form`元素绑定在一起，就可以在控件值发生改变时执行一段特定的脚本指令。

典型目标 Web Forms 2.0表单控件元素，以及`form`元素。

DOMFocusIn, DOMFocusOut

IE *n/a* NN *n/a* Moz *n/a* Saf *1.3/2* Op *9* DOM *2*

冒泡：是；可取消：否

当前元素接收到焦点（`DOMFocusIn`）或失去焦点（`DOMFocusOut`）时会触发对应的事件。在实际情况中，会在`focus`和`blur`事件之前触发这两个事件。它们与IE的`focusin`和`focusout`比较类似。

典型目标 所有可以正常接收焦点或已为`tabindex`属性赋值的已显示元素均可作为事件目标。

DOMFrameContentLoaded

IE *n/a* NN *n/a* Moz *1.0.1* Saf *n/a* Op *9* DOM *n/a*

冒泡：是；可取消：是

当浏览器解析了`frame`元素中加载的所有HTML源代码并在DOM中表示出全部元素后就会立即触发这个事件。事件触发会在加载外部内容（如图像、声音文件等）之前完成，因此在触发`load`事件还需要一些时间完

成加载工作。使用这个事件来替代load就可以允许DHTML脚本在加载外部内容的同时就开始修改文档树。但如果用户通过浏览器的历史菜单（如“返回”或“前进”按钮）来进行页面跳转则不会触发这个事件。通常会将这个事件与框架集文档中的frame元素绑定在一起。

典型目标 frame元素。

DOMMenuItemActive

IE *n/a* NN *n/a* Moz *1.0.1* Saf *n/a* Op *n/a* DOM *n/a*

冒泡：是；可取消：否

通过键盘的上、下箭头键选中option元素中的选项后就会触发这个事件。例如，如果为select元素指定了一个快捷键并且调用了那个快捷键组合，那么select元素就会触发focus事件。此时一旦按下上、下箭头键来选中选项就会触发DOMMenuItemActive事件。

典型目标 option元素。

DOMNodeInserted

IE *n/a* NN *n/a* Moz *all* Saf *1.3/2* Op *9* DOM *2*

冒泡：是；可取消：否

将一个节点明确地插入到一个已有的节点容器中后，就会在该节点上触发这个事件。在插入节点的容器元素中，如果指定了一个事件监听程序，那么就会在插入节点时在该节点上触发DOMNodeInserted事件。如果该事件会进一步向上传递，那么事件监听程序依然可以通过读取relatedNode事件属性来获得新容器的相关信息。须要注意的是，如果插入的节点来自于同一个文档中的另一个位置，那么在将该节点从原容器中移除前就会在节点上触发DOMNodeRemoved事件，因此这个事件可以向上传递至节点的原有容器。

典型目标 所有节点。

DOMNodeInsertedIntoDocument

IE *n/a* NN *n/a* Moz *n/a* Saf *n/a* Op *n/a* DOM *2*

冒泡：否；可取消：否

如果源节点来自于另一个文档，那么将该节点插入到一个已有的节点容器中时就在节点上触发这个事件。如果为这个事件和DOMNodeInserted事件指定事件监听程序，那么会首先触发DOMNodeInsertedIntoDocument事件。须要注意的是，如果插入的节点来自于另一个文档中的某个位置，那么一旦将该节点从其原有文档中移除时，它在离开原容器前就会在节点上触发DOMNodeRemovedFromDocument和DOMNodeRemoved事件，因此这个事件可以向上传递至节点的原有容器。

典型目标 所有节点。

DOMNodeRemoved

IE *n/a* NN *n/a* Moz *all* Saf *1.3/2* Op *n/a* DOM *2*

冒泡：是；可取消：否

将一个节点明确地从一个节点容器中删除时会在该节点上触发这个事件。如果在待删除节点的容器元素上指定一个事件监听程序，那么在删除节点时就会在该节点上触发DOMNodeRemoved事件。如果该事件会进一步向上传递，那么事件监听程序依然可以通过读取relatedNode事件属性来获得原容器的相关信息。

典型目标 所有节点。

DOMNodeRemovedFromDocument

IE *n/a* NN *n/a* Moz *n/a* Saf *n/a* Op *n/a* DOM *2*

冒泡：否；可取消：否

当一个节点由于要插入到另一个文档，从而须要从当前文档中移除时，就会在该节点上触发这个事件，这意味着该节点会与它所在的文档完全脱离。如果为这个事件和DOMNodeRemoved事件指定事件监听程序，那么会最后触发DOMNodeRemovedFromDocument事件。

典型目标 所有节点。

DOMSubtreeModified

IE *n/a* NN *n/a* Moz *n/a* Saf *1.3/2* Op *n/a* DOM *2*

冒泡：否；可取消：否

当文档中某个节点的内嵌节点结构发生改变时会触发这个事件。在众多的节点改变通知事件中，这个事件可以视为一个通用事件，而且往往会在最后才触发这个事件。

典型目标 所有节点。

DOMTitleModified

IE *n/a* NN *n/a* Moz *1.0.1* Saf *n/a* Op *n/a* DOM *2*

冒泡：否；可取消：是

当脚本改变了document.title的属性值时就会触发这个事件。

典型目标 document节点。

drag, dragend, dragstart

IE *5 (Win)* NN *n/a* Moz *n/a* Saf *1.3/2* Op *n/a* DOM *n/a*

冒泡：是；可取消：是

当用户开始拖拽一个已创建的选中域时，浏览器会首先在选中域的父元素中触发一个dragstart事件，然后再触发一系列的drag事件，当用户释放鼠标按钮时还会触发一个dragend事件。在拖拽的过程中，会在同一个元素中依次触发以上3种事件类型。

用户在操作时只会看到某种形式的鼠标指针在不断移动，而不会看到实际的元素浮动在页面四周。一旦用户按下鼠标键开始拖拽动作，在释放鼠标前都会不断触发drag事件。在其他元素中触发的与拖拽相关的事件也与之类似，例如当拖拽的鼠标指针出现在某个元素之上时会触发dragenter事件，但在这些情况中依然会首先触发drag事件。

拖拽过程中涉及的元素会依次收到如下事件：dragstart、drag（可能发生多次），最后是dragend。在拖拽动作的路径中所涉及的元素则会接收到dragenter、dragover，以及dragleave事件，而拖拽结束时所在的元素将会收到drop事件，这一事件会在dragend事件之前得以触发。而这些拖拽事件的具体触发数量则由拖拽动作的速度，以及客户端系统的速度共同决定。在一个速度较慢的机器上指定一次快速拖拽可能会使得某些事件无法得以触发。

典型目标 所有已显示的元素，以及document对象。Safari会在文本节点中触发这些事件。

dragdrop

IE *n/a* NN *|4|* Moz *n/a* Saf *n/a* Op *n/a* DOM *n/a*

冒泡：否；可取消：否

如果用户将一个桌面文件拖拽至文档中就会触发这个事件，但这仅适用于Navigator 4。通过签名脚本，浏览器可以读取该文件的文件名称。

典型目标 window对象。

dragend

请参见drag。

dragenter, dragleave, dragover

IE *5 (Win)* NN *n/a* Moz *n/a* Saf *1.3/2* Op *n/a* DOM *n/a*

冒泡：是；可取消：是

拖拽动作的路径中所涉及的元素通常会依次接收到dragenter、dragover和dragleave这3个事件。而这些拖拽事件的具体触发数量则由拖拽动作的速度，以及客户端系统的速度共同决定。在一个速度较慢的机器上指定一次快速拖拽可能会使得某些事件得以无法触发。

当用户将拖拽项放置到文档中某个目标上时，如果须要脚本执行某些自定义动作，那么请阻止dragenter和dragover事件的默认动作，这样一来目标对象的drop事件就可以在不受系统干扰的情况下完成正常拖拽响应之外的其他工作。关于拖拽相关的事件交互示例请参考第2章中的dataTransfer对象。

典型目标 所有已显示的元素，以及document对象。

dragstart

请参见drag。

drop

IE *5 (Win)* NN *n/a* Moz *n/a* Saf *1.3/2* Op *n/a* DOM *n/a*

冒泡：是；可取消：是

当用户完成拖拽操作释放鼠标按钮后，会在最后的目标元素上触发这个事件，而且它会在拖拽内容所在元素的dragend事件之前得以触发。如果将dragenter和dragover事件的event.returnValue设置为false，那么在最后的目标对象上就可以阻止这两个事件的触发，此时对象的drop事件就能够在不受系统常规拖拽响应干扰的情况下完成其工作。与拖拽有关的事件交互请参考第2章中的dataTransfer对象，其中的effectAllowed属性还可以控制鼠标指针在拖拽的释放目标对象上的具体样式。

典型目标 所有已显示的元素，以及document对象。

error

IE *4* NN *3* Moz *all* Saf *all* Op *all* DOM *2*

冒泡：否；可取消：是

根据事件处理程序所对应的元素或对象，当某个错误发生后就会触发这个事件。当元素加载外部内容时，如果发生错误就会在元素上触发error事件，例如，如果在img元素上使用了错误的URL地址就会触发这个事件。一旦在window对象（包括在<body>标签内直接指定的对象）中指定了事件处理程序，那么所有的运行时脚本错误（而非编译期语法错误）也会触发error事件。在早期的浏览器中，有些程序员使用一种脚本技术将错误提示信息显示在警告对话框中，以便捕获所有的运行时错误，其代码风格如下：

```
function doNothing() {return true;}
window.onerror = doNothing;
```

但由于这种方法须要在部署后才能发现错误，因此它并不便于代码调试与排错。关于如何在事件处理程序中处理错误信息请参考第2章中的Error对象。此外，越来越多的浏览器都开始使用更为现代的异常处理机制，请参见第5章中的try/catch语法。

典型目标 所有能够加载外部内容的元素，以及window对象和Mozilla中的XMLHttpRequest。

errorupdate

IE *4 (Win)* NN *n/a* Moz *n/a* Saf *n/a* Op *n/a* DOM *n/a*

冒泡：是；可取消：否

通过IE的数据绑定机制将数据发送至可写的数据源对象时，如果发生了错误就会触发这个事件。

典型目标 所有能够接受数据输入并支持数据绑定的元素。

filterchange

918

IE *4 (Win)* NN *n/a* Moz *n/a* Saf *n/a* Op *n/a* DOM *n/a*

冒泡：否； 可取消：否

当改变元素的过滤器的状态并且当一个转换过滤器或混合过滤器完成其动作时就会触发这个事件。此时就可以使用这个事件来叠加一系列的转换动作。

典型目标 大多数已显示的元素。

finish

IE *4* NN *n/a* Moz *n/a* Saf *n/a* Op *n/a* DOM *n/a*

冒泡：否；可取消：是

当marquee元素中的滚动文本完成最后的动作并停止循环滚动后就会触发这个事件。

典型目标 marquee元素。

focus

IE *3* NN *2* Moz *all* Saf *all* Op *all* DOM *2*

冒泡：否；可取消：否

如果当前元素通过用户动作（如点击或跳格等）或调用focus()方法获得了焦点，就会触发这个事件。在当前元素触发focus事件之前，会触发前一个已聚焦元素的blur事件。

尽管脚本浏览器从很早开始就能在表单控件和window对象中使用focus事件，但如果在其他元素中设置了tabindex属性，那么现在的浏览器实质上也可以在这类已显示的元素中触发该事件。须要注意的是，IE/Windows并不会在window对象上触发focus事件。

典型目标 在所有的浏览器中可以应用于input（文本或密码类型）、textarea、select及window对象，而在IE 5及后续浏览器和W3C DOM的浏览器中，所有已设定了tabindex属性值的已显示元素均可使用这个事件。

focusin, focusout

IE *6* NN *n/a* Moz *n/a* Saf *n/a* Op *n/a* DOM *n/a*

冒泡：是；可取消：否

当前元素接收到焦点（focusin）或失去焦点（focusout）时会触发对应的事件。在实际情况中，会在focus和blur事件之前触发这两个事件。在预料到元素会得到焦点或失去焦点时，微软建议使用这些事件处理程序来实现样式改变，以避免打乱正常的焦点聚焦或移动动作。

典型目标 所有可以正常接收焦点或已为tabindex属性赋值的已显示元素均可作为事件目标。

formchange

919

IE *n/a* NN *n/a* Moz *n/a* Saf *n/a* Op *9* DOM *n/a*

冒泡：否；可取消：否

在表单控件的change事件事件之后就会触发这个Web Forms 2.0事件。但与change事件不同的是，会首先按照源代码顺序依次在表单中的所有其他控件（除output元素之外）中触发formchange事件，最后才是整个form元素。因此一个表单控件就可以“知晓”表单中的其他控件何时发生了改变。

典型目标 Web Forms 2.0表单控件元素。

forminput

IE *n/a* NN *n/a* Moz *n/a* Saf *n/a* Op *9* DOM *n/a*

冒泡：否；可取消：否

在表单控件的input事件事件之后就会触发这个Web Forms 2.0事件。但与input事件不同的是，会首先按照源代码顺序依次在表单中的所有其他控件（除output元素之外）中触发forminput事件，最后才是整个form元素。因此一个表单控件就可以知晓表单中的其他控件何时发生了改变。

典型目标 Web Forms 2.0表单控件元素。

help

IE *4* NN *n/a* Moz *n/a* Saf *n/a* Op *n/a* DOM *n/a*

冒泡：是；可取消：是

当用户按下“F1”功能键时触发这个事件。如果此时某个元素拥有焦点，那么该元素就会收到这个事件通知。当然，也可以阻止浏览器的默认动作从而不显示IE的“帮助”窗口，并根据需要生成一个自定义的DHTML帮助系统。

典型目标 所有的已显示元素及document、window对象，但IE/Macintosh仅限于在window对象中使用。

input

IE *n/a* NN *n/a* Moz *n/a* Saf *n/a* Op *9* DOM *n/a*

冒泡：是；可取消：否

当用户在表单控件上进行输入时会触发这个Web Forms 2.0事件。例如，用户每次在文本输入域内按下按键就会触发一个input事件。因此脚本可以更快地完成输入校验，而不必一直等到元素失去焦点并触发change事件之后才执行相关动作。

典型目标 Web Forms 2.0表单控件元素。

invalid

IE *n/a* NN *n/a* Moz *n/a* Saf *n/a* Op *9* DOM *n/a*

冒泡：是；可取消：是

在提交表单前，如果表单控件的值不符合其他元素属性所规定的判断准则，就会触发这个Web Forms 2.0事件。相关示例请参见第1章中的output元素。

典型目标 Web Forms 2.0表单控件元素。

keydown, keyup

IE *4* NN *4* Moz *all* Saf *all* Op *all* DOM *3*

冒泡：是；可取消：是

当用户按下（keydown）并释放（keyup）一个键盘按键时会触发对应的事件。几乎所有的键盘键都可以在可聚焦的元素或对象上触发这两个事件，其中还包括各个功能键及定位键。这些事件所对应的event对象实例中就包含着按键（而非字符）的相关信息。关于如何跨浏览器处理键盘事件的详细信息请参见在线参考VI。尽管在实际应用中无法完全禁止“Ctrl”字符序列，但如果阻止了文本表单控件的keypress事件的默认动作，那么对应的字符也不会进入文本域中。

典型目标 所有可聚焦的已显示元素，以及document和window对象。

keypress

IE *4* NN *4* Moz *all* Saf *all* Op *all* DOM *n/a*

冒泡：是；可取消：是

当用户按下一个键盘字符键或一直按住某个键直到计算机的自动重复机制生效时就会触发这个事件。在自动

重复过程中，会在显示各个字符前触发一个新的事件。此时事件的触发顺序是keydown、keypress及keyup。在文本域中，字符出现在文本框之前就会触发keypress事件，因此，事件中的target/srcElement属性会控件显示字符之前就拥有具体的属性值。关于如何跨浏览器处理键盘事件的详细信息请参见在线参考VI。

典型目标 所有可聚焦的已显示元素，以及document对象。

layoutcomplete

IE *5.5* NN *n/a* Moz *n/a* Saf *n/a* Op *n/a* DOM *n/a*

冒泡：是；可取消：是

当一个用于打印预览的LayoutRect对象完成其内容显示时会触发这个事件。有关这个XML增强功能在IE 5.5及后续版本中的详细信息请访问如下链接：*http://msdn.microsoft.com/en-us/library/bb250434(VS.85).aspx*。

典型目标 XML LayoutRect对象。

load

IE *3* NN *2* Moz *all* Saf *all* Op *all* DOM *2*

冒泡：否；可取消：否

如果当前元素或对象完成了外部内容的加载或初始化工作就会触发这个事件。对于window对象而言，这可能是最为重要的一个事件，它表示在触发这个事件前文档及其内的元素已经完成了所有的加载工作，其中还包括外部内容的加载。一旦发现这个事件已被触发，那么脚本代码就可以引用文档树中的任意对象，并且不会引起相关错误。其他相关信息请参考DOMContentLoaded事件。

在frameset元素中，只有当它为所有的框架都触发了load事件后才会触发这个事件，此时load事件并不会从框架传递到框架集。须要注意的是，如果框架集完成初始化加载后，用户或脚本又向其中的某个框架中载入了新的页面，那么会在框架中而不是框架集中触发load事件。

尽管脚本浏览器从很早开始就能在window对象中使用load事件，但现在的浏览器实质上可以在所有已显示的元素中触发该事件。

典型目标 window对象（所有的浏览器）；img元素（从第4版浏览器开始）；所有能够加载外部内容的已显示元素（IE 4及后续版本，以及当前主流浏览器）；XMLHttpRequest（Mozilla 1.8及Safari 1.2）。

losecapture

IE *5 (Win)* NN *n/a* Moz *n/a* Saf *n/a* Op *n/a* DOM *n/a*

冒泡：否；可取消：否

当元素的事件捕获模式获得释放时会触发这个事件。请参考第2章中的共享方法setCapture()。

典型目标 所有已显示的元素。

message

IE *n/a* NN *n/a* Moz *n/a* Saf *n/a* Op *8* DOM *n/a*

冒泡：是；可取消：是

当一个文档在另一个文档上调用postMessage()方法时会触发这个事件，例如，调用iframe中的文档的postMessage()方法。通常会在目标document对象上绑定一个用于接收并处理事件的事件处理程序，并且event对象的属性还可以传达诸如domain和data一类的详细信息，以便目标文档进行下一步工作，如检查消息是否由预期发送者发送出来。此时的数据可以只是一个字符串，而两个文档则可分别来自于两个不同的域，这样就须要回避普通JavaScript处理中的跨域安全性约束问题。

典型目标 document对象。

mousedown, mouseup

IE *4* NN *4* Moz *all* Saf *all* Op *all* DOM *2*

冒泡：是；可取消：是

当用户按下（mousedown）或释放（mouseup）鼠标按键时会触发对应的事件。有关鼠标点击动作的事件会依照以下顺序依次触发：mousedown、mouseup及click。

在鼠标事件产生的event对象中，会将一些细节信息赋予其各个属性，如点击发生的坐标、使用了哪个鼠标按键，以及事件发生时是否按下了修改键等。事件处理函数可以获取这些属性的值以便掌握所需的信息。

须要注意的是，在1.4版的Mozilla浏览器和1.3版的Safari浏览器之前，可以在容器类元素的子文本节点中触发鼠标事件，这意味着event对象的target属性可以引用节点对象而不是整个容器元素。关于如何利用这些特性并实现跨浏览器的实施方案请参见在线参考VI。

典型目标　所有的已显示元素，但在Navigator 4中这两个事件受限于按钮类型的input元素，以及a和area元素。

mouseenter, mouseleave

IE *5.5* NN *n/a* Moz *n/a* Saf *n/a* Op *n/a* DOM *n/a*

冒泡：否；可取消：否

当用户将鼠标指针移入（mouseenter）或移出（mouseleave）一个元素的空间范围（包括边框及填充，但不包括边距）时会触发对应的事件。而光标每次进入和离开时，这两个事件均只会被触发一次。此外，这两个mouseover和mouseout事件的变形事件并不会向上传播。

典型目标　所有已显示的元素。

mousemove

IE *4* NN *4* Moz *all* Saf *all* Op *all* DOM *2*

冒泡：是；可取消：否

无论鼠标按键是否已被按下，一旦当鼠标指针移动到元素空间的上方就会触发这个事件。须要注意的是，这个事件会被反复触发，而其触发的频率则由鼠标移动的速度和系统资源决定。

在一个从鼠标事件中产生的event对象中，会将一些细节信息赋予其各个属性，如指针所在的坐标、事件发生时是否按下了修改键等。事件处理函数可以获取这些属性的值，以便掌握所需的信息。

在Navigator4中，只有在明确定义的事件捕获模式中才可以在window、document和layer对象中使用这个事件类型。须要注意的是，在1.4版的Mozilla浏览器和1.3版的Safari浏览器之前，可以在容器类元素的子文本节点中触发鼠标事件，这意味着event对象的target属性可以引用节点对象而不是整个容器元素。关于如何利用这些特性并实现跨浏览器的实施方案请参见在线参考VI。

典型目标　所有的已显示元素，但Navigator 4有些例外，请见上文。

mouseout, mouseover

IE *3* NN *2* Moz *all* Saf *all* Op *all* DOM *2*

冒泡：是；可取消：是

当用户将鼠标指针移入（mouseover）或移出（mouseout）一个元素的空间范围（包括边框及填充，但不包括边距）时会触发对应的事件。每当鼠标进入和离开时这两个事件就会被触发一次，但在Navigator 4/Windows中，会反复触发这两个事件，这与mousemove有些类似。

由于只有当窗口中另一个元素触发其mouseover事件之后才会触发mouseout事件，因此如果目标元素位于窗口或框架边缘，那么就可能不会触发mouseout事件，而且此时若用户将指针移除当前框架，则第1个框架的

body元素也不会收到mouseover事件。如果使用mouseout来恢复图像替换，那么用户就会看到一个被固定的图片。因此，须要为那些可替换的图片预留足够的空间以便妥善处理这些变化。

在一个从鼠标事件中产生的event对象中，会将一些细节信息赋予其各个属性，如点击发生的坐标，使用了哪个鼠标按键，事件发生时是否按下了修改键，以及指针从何处进入或向哪里移出等。事件处理函数可以获取这些属性的值以便掌握所需的信息。

须要注意的是，在1.4版的Mozilla浏览器和1.3版的Safari浏览器之前，可以在容器类元素的子文本节点中触发鼠标事件，这意味着event对象的target属性可以引用节点对象而不是整个容器元素。关于如何利用这些特性并实现跨浏览器的实施方案请参见在线参考VI。

尽管脚本浏览器从很早开始就能在表单中使用这些事件，但实质上只有现在的浏览器才可以在已显示的元素中触发它们。在以前的浏览器中，这些事件仅仅局限于a和area元素。

典型目标　所有的已显示元素，但也存在例外，请见上文。

mousewheel

IE *6* NN *n/a* Moz *n/a* Saf *n/a* Op *n/a* DOM *n/a*

冒泡：否；可取消：否

如果鼠标配备了滚轮，那么当用户转动滚轮时就会触发这个事件。此时event对象的wheelDelta属性可以表示转动的方向，以及转动的角度。

典型目标　所有已显示的元素，以及document对象。

move

IE *n/a* NN *4* Moz *n/a* Saf *n/a* Op *n/a* DOM *n/a*

冒泡：否；可取消：否

在Navigator 4中，如果用户或脚本移动了浏览器窗口就会触发这个事件。

典型目标　window对象。

move, moveend, movestart

IE *5.5* NN *n/a* Moz *n/a* Saf *n/a* Op *n/a* DOM *n/a*

冒泡：是；可取消：否

在编辑模式下，一个用于拖拽的可定位的元素会按下列顺序依次收到这些事件：movestart（开始进行拖动）、move（在拖动过程中重复出现），以及moveend（释放鼠标按钮）。下面这个示例使用了几个事件来示范IE编辑模式中的脚本编程。须要注意的是，如果将<!DOCTYPE>元素设定为标准兼容模式DTD，那么这种元素拖拽机制就无法在IE 6中正常工作。

```
<html>
<head>
<title>IE 5.5 (Windows) Edit Mode</title>
<style type="text/css">
    body {font-family:Arial, sans-serif}
    #myDIV  {position:absolute; height:100px; width:300px;}
    .regular {border:5px black solid; padding:5px; background-color:lightgreen}
    .moving {border:5px maroon dashed; padding:5px; background-color:lightyellow}
    .chosen {border:5px maroon solid; padding:5px; background-color:lightyellow}
</style>
<script type="text/javascript">
// 内嵌的拖动支持
document.execCommand("2D-position",false,true);
// 在两种模式间保留相关内容
var oldHTML = "";
```

```
// 设定静态编辑环境
function editOn() {
    var elem = event.srcElement;
    elem.className = "chosen";
}

// 设定特定的编辑——移动环境
function moveOn() {
    var elem = event.srcElement;
    oldHTML = elem.innerHTML;
    elem.className = "moving";
}

// 在拖动过程中显示坐标
function trackMove() {
    var elem = event.srcElement;
    elem.innerHTML = "Element is now at: " + elem.offsetLeft + ", " + elem.offsetTop;
}

// 关闭特定的编辑——移动环境
function moveOff() {
    var elem = event.srcElement;
    elem.innerHTML = oldHTML;
    elem.className = "chosen";
}

// 恢复原有环境（此时 wrapper 获取了 onfocusout）
function editOff() {
    var elem = event.srcElement;
    if (elem.id == "wrapper") {
        elem.firstChild.className = "regular";
    }
}

// 初始化各个事件处理程序
function init() {
    document.body.oncontrolselect = editOn;
    document.body.onmovestart = moveOn;
    document.body.onmove = trackMove;
    document.body.onmoveend = moveOff;
    document.body.onfocusout = editOff;
}

window.onload = init;
</script>
<head>
<body>
<div id="wrapper" contenteditable="true">
    <div id="myDIV" class="regular">
        This is a positioned element with some text in it.
    </div>
</div>
</body>
</html>
```


典型目标　所有已显示的元素。

paste

IE *5 (Win)* NN *n/a* Moz *n/a* Saf *1.3/2* Op *n/a* DOM *n/a*

冒泡：是；可取消：是

无论用户是通过“编辑”菜单、键盘快捷键还是上下文菜单来将选中的内容粘贴到系统剪切板，都会在启动“粘贴”命令后触发这个事件。这个事件的处理函数就可以从clipboardData对象中获得一些附加信息，通常可以在复制或剪切相关的事件处理程序中置入这些数据。

如果要为其他非可编辑元素提供“粘贴”菜单命令，那么针对paste事件处理程序中的同一个对象，须要在beforepaste事件处理程序中将对应的event.returnValue设置为false。从另一方面来看，如果要防止用户粘贴系统剪贴板中的内容，那么须要为paste事件处理程序将event.returnValue设置为false。

典型目标 所有已显示的元素。

926

progress

IE *n/a* NN *n/a* Moz *1.8* Saf *n/a* Op *n/a* DOM *n/a*

冒泡：否；可取消：否

通过XMLHttpRequest对象从服务器中获取数据时会周期性地触发这个事件。

典型目标 XMLHttpRequest对象。

propertychange

IE *5 (Win)* NN *n/a* Moz *n/a* Saf *n/a* Op *n/a* DOM *n/a*

冒泡：否；可取消：否

当脚本改变了元素的某个属性时会触发这个事件。直接对属性进行赋值，以及方法调用（如setAttribute()）都可以改变属性值。而改变元素对象的属性值也会触发这个事件，例如对元素的style对象的某个属性进行调整就会触发此事件。此时对应的event对象的propertyName属性就包含着当前被修改的属性的具体名称。

典型目标 所有的已显示元素，以及document对象。

readystatechange

IE *4* NN *n/a* Moz *1.0.1* Saf *1.2* Op *8* DOM *n/a*

冒泡：否；可取消：否

如果为元素修改readyState属性就会触发这个事件。对于元素属性的具体影响请参见第2章中的共享属性readyState。但值得注意的是，大多数浏览器仅在XMLHttpRequest对象中支持这个事件，请参见在线参考VII。

典型目标 加载外部内容的元素，其中还包括XMLHttpRequest对象。

reset

IE *4* NN *3* Moz *all* Saf *all* Op *all* DOM *2*

冒泡：否；可取消：是

无论是通过重置类型的input元素还是reset()脚本方法，一旦form元素收到重置表单的请求时均会触发这个事件。如果取消这个事件，那么在reset事件处理程序中的脚本代码就可以终止正常的表单重置操作。

典型目标 form元素。

resize

IE *4* NN *4* Moz *all* Saf *all* Op *all* DOM *2*

冒泡：是；可取消：否

用户或脚本改变了元素或对象的大小后就会触发这个事件。须要注意的是，如果改变一个窗口的大小还会强制改变页面中其他元素的大小。在IE中，来自于这些元素的resize事件会向上传递至body和window对象，这样可能会导致其resize事件处理程序在瞬间收到大量的事件处理要求。但在Safari中则只会在window对象

927

中触发这个事件，而不会影响到DOM树中的任何节点。

典型目标 在IE中所有具有实际尺寸的元素，以及window和document对象均可；在Mozilla和Opera中，可以以window和document对象为目标；而在Safari中则只能以window为目标。

resizeend, resizestart

IE *5.5* NN *n/a* Moz *n/a* Saf *n/a* Op *n/a* DOM *n/a*

冒泡：是；可取消：否

在编辑模式中，当用户通过拖放尺寸调整控点来改变元素大小时会触发这个事件。相关示例请参见move。

典型目标 所有已显示的元素。

rowenter, rowexit, rowsdelete, rowsinserted

IE *4 (Win)* NN *n/a* Moz *n/a* Saf *n/a* Op *n/a* DOM *n/a*

冒泡：是；可取消：否

在一个作为数据绑定的数据源对象的元素（通常是object对象）中，如果数据源或数据库改变了远程数据库表格中的某一行的具体信息，那么就会触发这个事件。

典型目标 object和applet元素，以及IE 5及后续版本中的xml元素。

scroll

IE *4* NN *n/a* Moz *n/a* Saf *n/a* Op *n/a* DOM *2*

冒泡：否；可取消：否

无论是通过用户操作还是脚本控制，每当元素显示滚动条或浏览器窗口执行滚动动作时均会触发这个事件。如果用户拖动了滚动条滑块，那么会在拖动过程中反复触发这个事件，而具体的触发频率则取决于系统速度。

典型目标 在默认情况下，所有可以使用滚动条的已显示元素均可，如textarea及window对象等，而那些已将overflow样式单属性设置为scroll的元素也可作为目标。在Mozilla和Safari中，可以以html元素为目标，而在Opera中，则可以body元素和document节点为目标。

select

IE *3* NN *2* Moz *all* Saf *all* Op *all* DOM *2*

冒泡：否（IE及Safari）；是（其他）；可取消：是

当用户在文本框（input或textarea元素）中拖拽一个选中域时会触发这个事件。但这个功能在Navigator 4/Windows中无法正常工作。而在windows系统中，IE 4及后续版本的浏览器还可将它应用于主体文本选择之中。

典型目标 在所有浏览器中，可用于文本类型的input和textarea元素，而在Windows系统下的IE 4及后续版本中，还可用于body元素。

selectionchange

IE *5.5* NN *n/a* Moz *n/a* Saf *n/a* Op *n/a* DOM *n/a*

冒泡：否；可取消：否

在IE的编辑模式下，当选择类型发生改变时会触发这个事件。

典型目标 document对象。

selectstart

IE *4* NN *n/a* Moz *n/a* Saf *1.3/2* Op *n/a* DOM *n/a*

冒泡：是；可取消：是

当用户开始在body元素或表单文本控件上拖拽一个选中域时立即就会触发这个事件。如果此时的选中域横跨多个元素，那也只会触发一个事件，并且会以选中域的起始元素作为目标。如果通过onselectstart="return false"这句代码在<body>标签中取消了这个事件，那么就可以防止恶意用户或用户无意间选中文本然后再对其进行滚动。

典型目标 所有已显示的元素。

start

IE *4* NN *n/a* Moz *n/a* Saf *n/a* Op *n/a* DOM *n/a*

冒泡：否；可取消：否

页面加载完成后，当marquee元素中的循环滚动文本开始移动时就会触发这个事件。

典型目标 marquee元素。

stop

IE *5 (Win)* NN *n/a* Moz *n/a* Saf *n/a* Op *n/a* DOM *n/a*

冒泡：否；可取消：否

一旦用户点击浏览器的“停止”按钮就会触发这个事件，即使文档及其内容已完成加载也不例外。

典型目标 document对象。

submit

IE *3* NN *2* Moz *all* Saf *all* Op *all* DOM *2*

冒泡：否（IE）；是（其他）；可取消：是

如果通过提交类型的input元素向form元素发出一个提交表单的请求，那么就会触发这个事件，但submit()脚本方法并不会触发此事件。在将表单发送至服务器之前，这个事件是启动客户端表单校验操作的最后一次机会。如果取消这个事件，那么在submit事件处理程序中的脚本代码就可以终止正常的表单提交操作。

典型目标 form元素。

unload

IE *3* NN *2* Moz *all* Saf *all* Op *all* DOM *2*

冒泡：否；可取消：否

无论当前页面将要跳转到一个新的页面、进行提交表单或关闭窗口，在卸载当前文档前都会触发这个事件。在支持此事件的浏览器中，它会在beforeunload事件之后才得以触发。在多数浏览器中，并不会等到unload事件处理函数执行完毕之后才开始实际的文档卸载工作。如果该函数所要执行的操作过多，特别是那些依赖于当前页面中某些脚本变量及元素的操作，那么在函数运行过程中相关的引用可能会成为无效引用，从而引发一些错误，如“未定义”或“对象未找到”等。因此在处理unload时请尽量短小快捷。

典型目标 body和frameset元素，以及window对象。

样式单属性参考

Style Sheet Property Reference

本章列出了当前主流浏览器中已实现的各种网页样式单属性，以及W3C规范为层叠样式单所指定的一些属性。在接下来的内容里列出了每个事件的版本信息，这样读者就可以很方便地检查是否能够在必须支持的浏览器中使用某个事件。在这些版本信息中可以很清晰地看到每个属性是在 Internet Explorer（简称IE）、早期的Mozilla 网景浏览器（Netscape Navigator，简称NN）、Mozilla浏览器（简称Moz）、Safari（简称Saf）、Opera（简称Op），以及W3C CSS规范的哪个版本中被首次引入。随着CSS 2.1的发布，已经从规范中删除若干个属性，本章中将它们标识为CSS版本“<2.1”，在实际应用中也不应该再使用这几个属性。

本章会依照CSS属性名称的字母顺序组织相关内容，而在每个属性的介绍中，则可以快速了解其值类型、实际代码示例，以及如何在JavaScript语言中调用可进行脚本操作的相关属性。目前在浏览器中已应用的一些样式单属性均属于CSS 3，但本章只明确标明了其中接近最终版本的部分内容。此外，Mozilla、Safari和Opera浏览器中引入的一些额外内容也是CSS 3初稿中的一部分。为了在正式规范定稿之前使用这些功能，同时避免它们与最终的CSS3规范发生命名冲突，每种浏览器均为这些属性赋予了其特定的前缀名称，以便进行标示，如：`-moz-`（Mozilla）、`-khtml-`（Safari），以及`-o-`（Opera）。须要注意的是，由于前缀中存在连字符，这导致无法在CSS Level 2中使用这些前缀。但这种做法显然是有意而为之的。此外，Mozilla和Safari的渲染引擎还支持很多以“`-moz-`”或“`-khtml-`”前缀开头的额外专有属性，它们不能用于网页，而只能在内部（如Mozilla中的XUL用户界面元素）或某些应用（如Apple Dashboard小程序或Macintosh应用中采用的WebKit）中使用。由于这些属性已经超出了DHTML文档创作的范畴，因此本章并未涉及这方面的内容。

属性值类型 931

在DOM中，很多样式单属性都共享类似的数据要求。因此为了使参考列表简单明了，本节仅详细介绍了其中一部分常用属性值类型。当属性与这些属性值类型中的某个类型有关时，请参照本节中的具体说明。

长度

长度值定义了文档真实状态的线性度量，通常它表示某段距离在水平或竖直方向上的高度、宽度、厚度或尺寸大小等。长度单位既可以是相对的也可以是绝对的。相对单位取决于用于显示文档的变量，如视频显示的点距值或像素密度等。CSS中的相对单位包括像素（`px`）、`em`及`ex`。其中，`em`是给定显示设备中显示元素字体时的实际高度，而`ex`则表示同等条件下小写字母“x”的高度。但当这两个单位用于定义`font-size`属性时会出现例外，此时它们还与父级元素的字体尺寸有关。

因此，当子元素将继承父级元素的相对长度值时要特别注意。在这种情况下，CSS推荐标准表示并不会继承其调整值，而会根据父元素中指定的属性值来获得其属性的具体计算值。例如，如果为`body`元素设定的`font-size`为`20pt`，并且`text-indent`为`2em`（即`40pt`），那么此`body`元素内的`p`元素或其他元素所继承的`text-indent`值将为`40pt`，即与其他元素的当前`font-size`值无关。为了替代继承的计算值，须要为`p`元素或其他元素重新设定`text-indent`值，或者改变它们的外部容器元素的`text-indent`值。Mozilla、Safari、Opera及IE/Macintosh

均会依照CSS标准执行以上相关计算。但在IE/Windows中，即使IE 7的标准兼容模式也会忽略这个约定。相反地，IE系列浏览器会重新计算已继承的相对样式值。在刚刚讨论的那个例子中，p元素的font-size将被设置为10pt，text-indent也不会从body元素继承一个40pt的计算值。此时会根据font-size（10pt）来重新计算p元素中未声明的text-indent值，并得到一个值为20pt的text-indent值。在所有平台中如果要实现CSS的像素级一致性完美表示，那么以上阐述的这种差异性会给跨浏览器部署带来一定的麻烦。

在描述字符尺寸时通常还会使用像素值，但这也存在着一定的隐患。随着输出设备具体像素密度的不同，像素尺寸也会发生改变。输出设备的像素密度越高，那么每个像素的真实尺寸就越小。为了在300dpi到600 dpi的打印机上进行打印输出，浏览器须要执行内部缩放计算，从而在每个像素上指定多个点，这样才会使得发送至打印机的文本字符与屏幕上的字符看起来更为一致。但不要期望在所有的监视器或打印机上均能输出完美的字符大小。对于长度值的设定而言，或多或少总会存在着一定的细微调整。

932

那些绝对长度单位往往只会用于具有常量物理属性的输出媒体，例如PostScript打印机。尽管交替使用绝对单位和相对单位也未尝不可，但一定要保证最终能够给页面访问者提供良好的输出效果。CSS中的绝对长度单位包括英寸（in）、厘米（cm）、毫米（mm）、派卡（pi），以及磅（pt）等。

URI、URL及IRI

通用资源标志符（Universal Resource Identifier，简称URI（译注1））是一个用于网络内容地址的广义术语。而国际化资源标识符(Internationalized Resource Identifier，简称IRI)则是一个可以包含非ASCII字符的Unicode字符地址。通用资源定位器（Universal Resource Locator，简称URL（译注2））是URI中的一种。由于大多数浏览器只关注URL，所以对网站建设者而言，可以将它们视同一体。另外，URL既可以是完整路径（包括协议、主机、域等部分），也可以是相对于当前文档URL的相对路径，但此时也有例外，例如：如果从一个*.css*文件（可能来自于另一个不同的目录）中导入了一个样式规则，那么相对URL将会使用该文件路径作为基路径。CSS中使用如下的特定属性语法格式来指定一个URI属性值：

```
propertyName: url("actualURL")
```

尽管可以省略*actualURL*两端的引号，但我们仍然建议使用引号。

颜色

可以使用RGB来指定一个颜色值，也可以通过一个等价明语常量来进行描述，相关内容请参考附录A。如果使用RGB形式的样式单属性值，那么可以采用以下3种格式之一：十六进制三元组、十进制值或百分比值。十六进制三元组由3对十六进制数字组成，每个数字的取值范围为从00到FF，分别对于红、绿、蓝3个颜色成分。将3对数字组合在一起，并在前面再加一个“#”号就可以表示1种颜色。因此，在全红（#ff0000）中仅包含red值（ff），其他两个色调均为空（00），即#ff0000，而纯蓝则由#0000ff表示。当然，在这种颜色表示方法中也可以使用大写字母。在这种记法中，当颜色值中出现成对的字符时可以采用一种简化的描述方式来指定十六进制字符。例如，#f0a可以解释为#ff00aa。

另一种RGB值则须要借助于rgb前缀，然后在rgb之后的括号中使用逗号分隔红、蓝、绿3种颜色值。采用十进制颜色值时，每种颜色的变化范围介于0到255之间，此时0表示完全缺失对应的颜色。当然，也可以使用百分比来表示每种值。下面列出的范例使用4种不同方法描述了纯蓝的颜色定义：

933

```
color: green
color: #00ff00
color: rgb(0, 255, 0)
```

译注1：W3C中全称为Uniform Resource Identifier。

译注2：W3C中全称为Uniform Resource Locator。

```
color: rgb(0%, 100%, 0%)
```

在最后两个例子中，如果设定的颜色值超出了允许的最大值，那么浏览器会将其值回复到最大值。

这种数值表达方式会导致数量巨大的可能组合，使用它们可以表示1600余万种颜色。在Web出现初期，典型的PC显示设备受处理能力和内存的限制，只能显示256色，这意味着如果页面访问者使用这样的显示设备无法区分上千万种颜色之间细微差别。因此，网页开发者往往使用一种称为“**网络安全色**”的颜色设置，它仅包含216种区别明显的颜色。尽管当今的计算机已经有充足的处理能力和内存来轻松适应这上千万种颜色，但一些页面设计者依然坚持使用更为有限的颜色集合以保证后向兼容性。一个可以适用于所有浏览器和计算机的颜色参考可以在此链接中获取：*http://www.w3school.com.cn/html/html_colors.asp*。

CSS2规范引入了另一种颜色命名机制，此时可以根据操作系统控制面板中为用户界面元素指定的颜色来定义所需的颜色。在按钮的标记文本、滚动条，以及3D阴影中就可以使用这种颜色指定方式。对于那些色盲用户而言，则须要选择对比度明显的色彩集合，以便他们清晰地分辨出不同的屏幕元素。但由于在不同的浏览器品牌和操作系统中默认的颜色显示也有较大的差异，因此请谨慎使用这种颜色设置。为了将这些颜色与样式关联起来，可以使用下列关键字来代替颜色属性值：`activeborder`、`activecaption`、`appworkspace`、`background`、`buttonface`、`buttonhighlight`、`buttontext`、`captiontext`、`graytext`、`highlight`、`highlighttext`、`inactiveborder`、`inactivecaption`、`inactivecaptiontext`、`infobackground`、`infotext`、`menu`、`menutext`、`scrollbar`、`threeddarkshadow`、`threedface`、`threedhighlight`、`threedlightshadow`、`threedshadow`、`window`、`windowframe`、`windowtext`。

选择符

大多数样式单规则都和样式单选择符标识不同的HTML元素或元素分组有关，例如类、ID及情景选择符（contextual selector），请参见第3章中的相关内容。表4-1列出了到CSS3为止，W3C推荐标准定义的各个选择符，以及各种主流浏览器版本对它们的支持情况。在“格式”一栏中，`E`和`F`代表了两种不同元素的元素名称。根据附录F所述，“Mozilla”一栏中列出的“m18”表示一个早期的里程碑式的发行号。

表 4-1：主要的 CSS 选择符

名称	格式	IE	Mozilla	Safari	Opera	CSS
相邻选择符	*E + F*	7	m18	all	7	2
属性选择符（名称）	*E[attr]*	7	m18	all	7	2
属性选择符（名称及值）	*E[attr="val"]*	7	m18	all	7	2
属性选择符（名称及任意值）	*E[attr~="val"]*	7	m18	all	7	2
属性选择符（名称及使用连字符分隔的任意值）	*E[attr|="val"]*	7	m18	all	7	2
属性选择符（名称并以该值作为起始）	*E[attr^="val"]*	7	1.0.1	all	9	3
属性选择符（名称并以该值作为结束）	*E[attr$="val"]*	7	1.0.1	all	9	3
属性选择符（名称并包含该值）	*E[attr*="val"]*	7	1.0.1	all	9	3
子对象选择符	*E > F*	7	m18	all	7	2
类选择符	*E.classname*	4	m18	all	7	1
包含选择符	*E F*	4	m18	all	7	1
同族选择符	*E ~ F*	7	1.7.2	n/a	9	3
ID 选择符	*E#id*	4	m18	all	7	1
类型选择符	*E*	4	m18	all	7	1
通配选择符	*	4	m18	all	7	2

伪元素及伪类选择符

在某些情况下，可能须要为一个已经明确定义的元素组件（伪元素）或呈现出同一种状态的所有元素（伪类）指定一种样式。

伪元素

由于使用这种CSS声明后，会使得浏览器向一个已存在的元素中插入一个人工元素，“伪元素”因此而得名。例如，CSS可以为一个块级元素的第1个字母，以及第1行定义伪元素。此时真实元素的HTML源代码可能和下面这行代码比较类似：

```
<p>A mere paragraph.</p>
```

935

但浏览器在实现“:first-letter”及“:first-line”伪元素时，实际上会按照如下所示的元素结构来处理p元素：

```
<p><p:first-line><p:first-letter>A</p:first-letter> mere paragraph.</p:first-line></p>
```

“</p:first-line>”结束标签的实际位置则由p元素的具体显示内容来决定。如果将这段文字变窄，并且第1行在“mere”之后进行换行，那么“:first-line”伪元素的隐式结束标签就会位于“mere”的空格之后。这样一来，就可以为块级元素中的特定部分指定各种不同的样式属性，例如首字符下沉效果：

```
p:first-letter {font-size: 36pt; font-weight: 600; font-family: Rune, serif; float: left}
```

或者在第1行中使用大写字母：

```
p:first-line{text-transform: uppercase}
```

使用这种方式的好处在于，无论如何定义伪元素结构或进行样式指定，文档树都不会受到影响。在上面这个简单的p元素示例中，元素中仅包含一个文本子节点。

为了区分来自于CSS标记中不同伪类的伪元素，CSS3引入了一种使用双冒号的新标记方式，例如：

```
p::first-line{text-transform: uppercase}
```

当然，实现了这种新标记方式的浏览器（除IE之外的所有主流浏览器）也同时支持旧的标记方式，以保证后向兼容性。

到CSS2为止，已定义了4种伪元素，如表4-2所示。须要注意的是，“:first-letter”伪元素目前仅承认如下类型的样式属性：background、border、clear、color、float、font、letter-spacing、line-height、margin、padding、text-decoration、text-shadow、text-TRansform、vertical-align（此时float应为none），以及word-spacing。而“:first-line”伪元素则承认如下类型的样式属性：background、clear、color、font、letter-spacing、line-height、text-decoration、text-shadow、text-transform、vertical-align，以及word-spacing。CSS3还引入了一种可选的伪元素——“::selection”，可以在用户选中域中使用它的样式属性，例如，可以在友好打印模式下通过它来设计选中的文本。

表 4-2：CSS2 中的伪元素

名称	IE/Windows	IE/Mac	Moz/Saf/Op	CSS	描述
:after	n/a	n/a	all	2	位于元素之后（请参见 content 属性）
:before	n/a	n/a	all	2	位于元素之前（请参见 content 属性）
:first-letter	5.5	5	all	1	块级元素的第一个字母
:first-line	5.5	5	all	1	块级元素的第一个行

伪类

936

对a元素而言，存在以下几种易于分辨的元素状态：未访问过的链接、正被点击的链接，以及过去曾经被访问过的链接。这些状态就被称之为“**伪类**”，它们的工作方式类似于类选择符定义，但在其元素标签中并不

须要进行标示。伪类通常可以视为对另一个选择符的某种修改。在下面这个例子中，请注意“:hover”伪类如何根据一种规则来影响所有的a元素，并且在拥有特定ID的a元素上应用一个额外的颜色属性。

```
a {text-decoration: none}
a:hover {text-decoration: underline}
#specialA:hover {color: red}
```

当然，也并不总是基于元素状态进行伪类的分类。文档中的上下文环境、页面位置（左或右），甚至于语言都可以用来赋予伪类。例如，当与“:first-child”伪类相关的元素是文档树中第1个子元素时，“:first-child”可将该元素转变为一个特殊的类。因此，在类名为“section”的任意容器中，如果p元素是其第1个子元素，那么就会应用下面定义的这种文本尺寸样式。

```
.section > p:first-child {font-size: 110%}
```

在此处使用“>”子对象选择符就是为了将“p:first-child”伪类的应用限制在容器的第1个子对象之中。此时如果去掉了这个子对象选择符，当p元素是其他容器的第1个子对象时也会应用这个样式规则。

表4-3列出了CSS2支持的所有伪类。它们在主流浏览器中的实现比较分散。

表 4-3：CSS2 中的伪类

名称	IE/Windows	IE/Mac	Moz/Saf/Op	CSS	描述
:active	4	4	all	1	用户正在点击一个 a 元素
:first	n/a	n/a	n/a	2	文档的第 1 页（根据“@page”声明）
:first-child	n/a	5	all	2	作为另一元素的第 1 个子对象的任意元素
:focus	n/a	5	all	2	拥有焦点的任意元素
:hover	4	4	all	2	光标正位于该元素之上（仅适用于 IE 4-6 及 IE 7“quirks”模式中的 a 元素）
:lang (代码)	n/a	5	n/a	2	拥有相同语言代码的元素
:left	n/a	n/a	n/a	2	左向页面（根据“@page”声明）
:link	4	4	all	1	未曾访问过的 a 元素
:right	n/a	n/a	n/a	2	右向页面（根据“@page”声明）
:visited	4	4	all	1	在浏览器的历史记录中曾经被访问过的 a 元素

而CSS3则引入了大量的新伪类，并且已经在最新的主流浏览器中实现了其中的一部分。通过这些新的选择符，就能够为符合特定上下文判定条件的元素指定样式属性，例如，表格中的每一列。而且此时也不必再通过HTML标记来声明大量的类属性。某些选择符还允许样式单补充或替代浏览器的默认显示状态，如元素禁用状态或一种选中的按钮。在CSS3中，到最新的W3C工作草案为止的各种选择符均已罗列在表4-4中。

表 4-4：CSS3 中的伪类

名称	IE	Mozilla	Safari	Opera	CSS	描述
:checked	n/a	n/a	n/a	9	3	已选中的单选或多选按钮
:disabled	n/a	n/a	n/a	9	3	被禁用的可聚焦元素
:empty	n/a	1.0.1	all	n/a	3	不包含子节点的元素
:enabled	n/a	n/a	n/a	9	3	已启用的可聚焦元素
:first-of-type	n/a	n/a	n/a	n/a	3	符合标签名称并且在其父元素中为第 1 个子对象的任意元素
:invalid	n/a	n/a	n/a	9	n/a	值无效的 Web Forms 2.0 控件元素
:last-child	n/a	1.0.1	n/a	n/a	3	作为另一元素的最后一个子对象的任意元素
:last-of-type	n/a	n/a	n/a	n/a	3	符合标签名称并且在其父元素中为最后一个子对象的任意元素

续表

名称	IE	Mozilla	Safari	Opera	CSS	描述
:not(选择符)	n/a	n/a	n/a	n/a	3	不符合选择符的元素
:nth-child(an+b)	n/a	n/a	n/a	n/a	3	在a分组内位于第b位的每个子元素
:nth-last-child(an+b)	n/a	n/a	n/a	n/a	3	在a分组内按倒序排列位于第b位的每个子元素
938 :nth-last-of-type (an+b)	n/a	n/a	n/a	n/a	3	标签名相同，且在a分组内按倒序排列位于第b位的每个子元素
:nth-of-type(an+b)	n/a	n/a	n/a	n/a	3	标签名相同，在a分组内位于第b位的每个子元素
:only-child	n/a	n/a	n/a	n/a	3	没有任何同级兄弟元素的某个元素
:only-of-type	n/a	n/a	n/a	n/a	3	标签名相同，且没有任何同级兄弟元素的某个元素
:root	n/a	n/a	n/a	n/a	3	HTML元素
:target	n/a	n/a	n/a	n/a	3	某个锚链的目标元素

@规则

CSS2为声明与指令（或称之为命令）定义了一种可扩展的结构，作为样式单定义的一部分。由于这些规则以“@”符号开头，后跟声明标识符，因此将它们统称为“@规则”。每个规则还包含一个或多个描述符，以便定义规矩的特性参数。

尽管@规则往往是样式单中的第1个声明，但在实际应用中，每个样式单使用一个单独的@规则可能更好，@media尤其如此。例如，下面这段代码为文档显示提供了两种不同的样式特性，它们分别适用于屏幕显示（相对字符尺寸）和纸面打印（绝对字符尺寸）。

```
<style type="text/css">
@media screen {
    body {font-size: 14px}
}
</style>
<style type="text/css">
@media print {
    body {font-size: 12pt}
}
</style>
```

在IE浏览器中，可以使用“@font-face”规则来下载字体定义文件，而且它还可以将每个字体定义与字体簇名称关联起来，以便在随后的样式指令中使用。下文这个示例下载了一个IE浏览器可以采用的字体文件，并且将其定义指派给一个名为“Stylish”的字体簇：

```
<style type="text/css">
@font-face {
    font-family: Stylish;
    font-weight: normal;
939
    font-style: normal;
    src: url(fonts/stylish.eot);
}
</style>
```

在同一个样式单中，IE浏览器允许定义多个“@font-face”规则。关于如何创建字体定义文件，以便在Windows及Macintosh系统中的IE浏览器下使用，请访问此链接：*http://msdn.microsoft.com/zh-cn/library/ms53030 3*

(en-us,VS.85).aspx。

CSS3命名空间通过@namepace规则进行定义，目前部分主流浏览器可支持这个规则。给出定义后，任意选择符如果须要引用命名空间中定义的元素名称必须首先引用该空间的名称。语法格式为：首先是一个管道字符（“|”），后接元素名称。例如，下面这个规则将会在products命名空间中定义的td元素上使用蓝色：

```
@namespace products url(http://www.example.com/DTD/productDB);products|td {color: blue}
```

表4-5列出了CSS和主流浏览器支持的各种@规则。

表 4-5：CSS 的@规则

名称	IE/Windows	Mozilla	Safari	Opera	CSS	描述
@namespace	n/a	1.0.1	1.3/2	9	2	用于外部样式单文件的字符集
@charset	5	m18	n/a	7	2	用于外部样式单文件的字符集
@font-face	4	n/a	n/a	n/a	n/a	在内嵌字体与客户端系统字体（或下载的字体）之间进行字体匹配的字体说明
@import	4	m18	all	7	1	导入一个外部样式单。对于层叠样式的影响请参见第 3 章
@media	5	m18	all	7	2	为一个或多个样式单规则定义输出媒体类型。为同一个选择符指定的规则往往位于不同的@media 规则之中，例如“@media print”或“@media screen”，在特定的媒体中显示文档时，会使用与该媒体对应的规则
@page	n/a	n/a	n/a	7	2	定义页面框的尺寸、边距、打印方向、裁剪标记，以及与页面相关的其他属性，以便对文档打印进行管理

常规约定

贯穿本章内容的CSS语法说明均遵循以下方针：

- 使用“等宽”字体显示的词为关键字或常量值，使用时请保持原样。
- 使用“等宽斜体”字体显示的词为占位符。
- “[]”中的内容为可选内容。
- 使用“|”分隔的两个或多个值表示在该处可以使用的可选值。
- 在某个值后可能使用大括号显示一列数字，如{1,2}。这两个数字表示可以使用的最小值和最大值。
- 使用“||”分隔的多个值指出了必须出现的一个或多个值，但并不表示出现的具体顺序。

“应用于”表示该样式属性能够影响哪些HTML元素。其中，某些样式属性只能应用于块级元素、行内元素或置换元素。块级元素总是从一个新的行开始，并且在元素末尾进行强制换行，例如h1、p元素等。而行内元素则可放置在一个文本行之内，并且不会干扰内容的排列，如em元素等。置换元素既可以是一个块级元素也可以是一个行内元素，但在动态改变其内容时并不需要重新排列其四周的内容。由于可以替换img元素矩形空间中的图像源文件，因此可以将img元素归为此类。

本章中列出“初始值”的目的与其他参考章节中的“默认值”相同。而采用这一术语主要是为了与CSS规范中的相关术语保持一致。

很多属性说明中都包含一个称为“对象模型引用方式”的内容，它展示了如何在脚本中像引用浏览器文档对象模型的属性（明确地说，就是style对象的属性）一样引用该属性。关于样式属性与等价的脚本代码在兼容性方面的优劣请查阅第2章，这些脚本代码往往与本章中列出的样式单属性实现差异较大。

941 按字母属性排列的属性参考

azimuth

IE *n/a* NN *n/a* Moz *n/a* Saf *n/a* Op *n/a* CSS *2*

通过继承得到：是

azimuth用于设置声音源（如一个文本-语音转换浏览器）的水平角度从而让收听者位于圆形声音空间的中心，这类似于拥有环绕立体声设备的剧院。同时请参见elevation属性。

CSS语法 azimuth: *angle* | *angleConstant* || *direction*

值

除inherit之外至少包含两个值。其中一个是顺时针方向的角度，第2个则表示向左或向右的20度增量运动。angle值的取值范围为-360到+360之间，单位为“deg”，如“90deg”。当值为“0deg”时，表示它正好位于收听者之前。如果要将角度设置到收听者左侧，那么该值可以是“-90deg”或“270deg”。也可以从一组角度说明常量中选择一个作为*angleConstant*的值，而这些常量值则对应于圆周上的一些固定点。如果添加了“behind”修改量，那么该值就会从收听者之前转换到收听者之后。

常量值	等价值	常量值	等价值
center	0deg	center behind	180deg
center-right	20deg	center-right behind	160deg
right	40deg	right behind	140deg
far-right	60deg	far-right behind	120deg
right-side	90deg	right-side behind	90deg
left-side	270deg	left-side behind	270deg
far-left	300deg	far-left behind	240deg
left	320deg	left behind	220deg
center-left	340deg	center-left behind	200deg

对于*direction*而言，则可以从leftwards和rightwards常量中选择一个作为其值。这些设定可以在指定的方向上将声音偏移20度。

初始值 center

示例

```
h1 {azimuth: 45deg}
p.aside {azimuth: center-right behind}
```

应用于 所有元素。

942 background

IE *4* NN *n/a* Moz *all* Saf *all* Op *all* CSS *1*

通过继承得到：否

使用这个属性可以在一个属性指令内快速地设定5个单独而又相关的背景样式属性。此时可以按照任意顺序排列属性值，彼此之间使用空格进行分隔。须要注意的是，尽管Navigator 4并未为这个属性提供官方支持，但某些值组合可以正常工作。

CSS语法 background: *background-attachment* || *background-color* || *background-image* || *background-position* || *background-repeat*

值 由5个背景样式属性值以任意顺序组成的值。如果某些属性并未指定具体的值，那么将会采用其初始值。关于预期值的详细信息请分别参考各个样式属性说明。

初始值 无。

示例 body {background: url(watermark.jpg) repeat fixed}

应用于 所有元素。
对象模型引用方式 [window.]document.getElementById("*elementID*").style.background

background-attachment

IE 4 NN n/a Moz all Saf 1.2 Op all CSS 1

通过继承得到：否

通过背景图片属性将一个图像应用于元素背景时，background-attachment可以设定该图像是否应该随文档一同滚动。图像既可以固定在元素的可视区域内，也可以在文档内容滚动时随着元素一同运动。在滚动过程中，固定的贴附图片看起来就像是电影致谢内容后固定的背景图。

CSS语法 background-attachment: fixed | scroll
值 fixed会让图像在元素视区内保持不动，而scroll则会使图像与文档内容一同滚动。
初始值 scroll
示例 body {background-attachment: fixed}
应用于 所有元素。
对象模型引用方式
[window.]document.getElementById("*elementID*").style.backgroundAttachment

background-color

IE 4 NN 4 Moz all Saf all Op all CSS 1

943

通过继承得到：否

此属性设定了元素的背景色。尽管内嵌元素的background-color属性看起来是继承于其父元素，但实质上该属性的初始值为transparent，这使得其临近的最外层元素可以透过当前元素的空白部分显示出来。

CSS语法 background-color: *color* | transparent
值 任何有效的颜色设置或transparent。
初始值 transparent
示例 .highlighter {background-color: yellow}
应用于 所有元素。
对象模型引用方式
[window.]document.getElementById("*elementID*").style.backgroundColor

background-image

IE 4 NN 4 Moz all Saf all Op all CSS 1

通过继承得到：否

此属性可用于设定元素的背景图。如果同时为元素设置了background-color，那么当图片加载失败时会出现该背景色，否则图片会覆盖背景色。而具有一定透明度的图像则可以显现出元素的背景色。另请参见background-attachment属性。

CSS语法 background-image: *uri* | none
值 如果要指定一个URL，请使用url()封装该属性值。如果不想在元素背景中载入任何图片，那么可以忽略这个属性或将它设置为“none”。
初始值 none
示例 h1 {background-image: url(watermark.jpg)}
应用于 所有元素。
对象模型引用方式
[window.]document.getElementById("*elementID*").style.backgroundImage

background-position

IE 4 NN n/a Moz all Saf all Op all CSS 1

944

通过继承得到：否

此属性可以为background-image属性所确定的背景图片指定左、上边缘的具体位置。

CSS语法

```
background-position: [percentage | length] {1,2} |
  [top | center | bottom] || [left | center | right]
```

值 可以使用一个或两个百分比值来指定图像起始的位置，它们分别表示块级元素所在空间的宽度和高度的百分数。如果只使用一个百分比值，那么它会作用于水平尺度，而竖直尺度则会自动设置为50%。除百分比外，还可以使用长度值，此时可以采用最适用于输出媒介的长度单位。当然，也可以将百分比与长度混合起来一同使用。此外，还有两类常量值集合可用来生成组合值，以便取代所需的数值。此时可以从top left、top right或bottom center中选择一个作为属性值。须要注意的是，无论采取哪种方式来指定这两个值，都必须使用空格进行分隔。

初始值 0% 0%

示例

```
div.marked {background-image: url(watermark.jpg); background-position: center top}
```

应用于 块级元素及置换元素。

对象模型引用方式

```
[window.]document.getElementById("elementID").style.backgroundPosition
```

background-position-x, background-position-y

IE 4 NN n/a Moz n/a Saf 1.3/2 Op n/a CSS n/a

通过继承得到：否

此属性可以为background-image属性所确定的背景图片指定左（x）、上边缘（y）的具体位置。

CSS语法

```
background-position-x: [percentage | length] | [left | center | right ]
background-position-y: [percentage | length] | [top | center | bottom]
```

值 可以使用百分比值来指定图像起始的位置，它们分别表示块级元素所在空间的宽度或高度的百分数。除百分比外，还可以使用长度值，此时可以采用最适用于输出媒介的长度单位。此外，可以采用与轴线有关的特定常量值来取代所需的数值。

初始值 0%

示例

```
div.marked {background-image: url(watermark.jpg); background-position-x: center}
```

应用于 块级元素及置换元素。

对象模型引用方式

```
[window.]document.getElementById("elementID").style.backgroundPositionX
[window.]document.getElementById("elementID").style.backgroundPositionY
```

background-repeat

IE 4 NN n/a Moz all Saf all Op all CSS 1

通过继承得到：否

指明是否应该沿着轴线方向重复绘制背景图像。使用这种不断重复的背景图像就可以在水平和竖直方向上创造出条纹图案。

CSS语法

```
background-repeat: no-repeat | repeat | repeat-x | repeat-y
```

值 如果此属性被设置为“no-repeat”，那么只会在元素内background-position属性所指定的位置出现一张图像，默认位置为左上角。通常情况下会同时沿着两个方向重复绘制图像，但是将此属性设置为repeat-y后，就只会沿着竖直方向重复绘制，而repeat-x的效果也与之类似。

初始值 repeat

示例

```
body {background-image: url(icon.gif); background-repeat: repeat-y}
```

应用于 所有元素。

对象模型引用方式

```
[window.]document.getElementById("elementID").style.backgroundRepeat
```

behavior

IE *5(Win)* NN *n/a* Moz *n/a* Saf *n/a* Op *n/a* CSS *n/a*

通过继承得到：否

此属性可以将一个外部行为定义与元素联系起来。

CSS语法 `behavior: uri[, uri[, ...]]`

值

CSS格式的URL值，它可以表示一个指向外部*.htc*文档的实际URL地址，也可以对应于一个在页面中加载ActiveX控件的`object`对象的ID值，还可以是一个内置的默认行为（此时格式为`url(#default#behaviorName)`）。默认的行为名称包括：`anchorClick`、`anim`、`clientCaps`、`download`、`homePage`、`httpFolder`、`mediaBar`、`saveFavorite`、`saveHistory`、`saveSnapshot`、`userData`。

有关这些默认行为的具体信息，以及使用它们所需的安全条件请访问如下链接：*http://msdn.microsoft.com/en-us/library/ms531425(VS.85).aspx*。

初始值 无。

示例 `input.numOnly {behavior: url(numInput.htc)}`

应用于 所有元素。

对象模型引用方式

```
[window.]document.getElementById("elementID").style.behavior
[window.]document.getElementById("elementID").behaviorUrns[i]
```


border

IE *4* NN *n/a* Moz *all* Saf *all* Op *all* CSS *1*

通过继承得到：否

在一条指令内，可以通过这个属性快速地设定元素四周4条边框的宽度、样式，以及颜色。如果未在其中明确设定某个样式属性，那么就会使用其默认值来设定该属性。如果不希望4条边框完全相同，可以使用其他属性分别为它们单独设定宽度、样式和颜色。

由于浏览器对边框默认的属性定义存在着一定的差异性，因此每个样式单的`border`规则都应该引入宽度和样式设定。如果这两个属性设置失败，可能会导致在某种浏览器中无法看到边框。

CSS语法 `border: border-width || border-style || color | transparent`

值 对于*border-width*和*border-style*属性值，请分别参考本章中的对应属性。有关*color*的详细信息，请参考本章前文中与颜色有关的部分。

初始值 无。

示例 `p {border: 3px groove darkred}`

应用于 所有元素，但在Windows系统中的IE 4和5浏览器中，只能应用于块级元素及置换元素。

对象模型引用方式 `[window.]document.getElementById("elementID").style.border`

border-bottom, border-left, border-right, border-top

IE *4* NN *n/a* Moz *all* Saf *all* Op *all* CSS *1*

通过继承得到：否

在一条指令内，可以通过这个属性快速地设定元素某条边框的宽度、样式，以及颜色。如果未在其中明确设定某个样式属性，那么就会使用其默认值来设定该属性。

CSS语法

```
border-bottom: border-bottom-width || border-bottom-style || color | TRansparent
border-left: border-left-width || border-left-style || color | transparent
```

```
border-right: border-right-width || border-right-style || color | TRansparent
border-top: border-top-width || border-top-style || color | transparent
```

值 对于宽度和样式属性值，请分别参考本章中的border-bottom-width和border-bottom-style属性。有关*color*的详细信息，请参考本章前文中与颜色有关的部分。

初始值 无。

示例

```
p {border-bottom: 3px solid lightgreen}
p {border-left: 6px solid lightgreen}
p {border-right: 3px solid lightgreen}
p {border-top: 6px solid lightgreen}
```

应用于 所有元素，但在Windows系统中的IE 4和5浏览器中，只能应用于块级元素及置换元素。

对象模型引用方式

```
[window.]document.getElementById("elementID").style.borderBottom
[window.]document.getElementById("elementID").style.borderLeft
[window.]document.getElementById("elementID").style.borderRight
[window.]document.getElementById("elementID").style.borderTop
```

border-bottom-color, border-left-color, border-right-color, border-top-color

IE *4* NN *n/a* Moz *all* Saf *all* Op *all* CSS *2*

通过继承得到：否

这几个属性可以分别设定元素对应边框的颜色。须要注意的是，如果滥用这些属性，会将那些原本并不协调的颜色混合在一起。如果要通过一条指令来设定多条边框的颜色，请参见border-color属性。

CSS语法

```
border-bottom-color: color | transparent
border-left-color: color | transparent
border-right-color: color | transparent
border-top-color: color | transparent
```

值 有关*color*的详细信息，请参考本章前文中与颜色有关的部分。另请参见transparent。

初始值 无。

示例

```
p {border-bottom-color: gray}
div {border-left-color: #33c088}
p.special {border-right-color: rgb(150, 75, 0)}
h3 {border-top-color: rgb(100%, 50%, 21%)}
```

应用于 所有元素，但在Windows系统中的IE 4和5浏览器中，只能应用于块级元素及置换元素。

对象模型引用方式

```
[window.]document.getElementById("elementID").style.borderBottomColor
[window.]document.getElementById("elementID").style.borderLeftColor
[window.]document.getElementById("elementID").style.borderRightColor
[window.]document.getElementById("elementID").style.borderTopColor
```

border-bottom-style, border-left-style, border-right-style, border-top-style

IE *4* NN *n/a* Moz *all* Saf *all* Op *all* CSS *2*

通过继承得到：否

这几个属性可以分别设定元素对应边框的线条样式。通过这些特定的边缘属性，可以改变已由border、border-style属性所指定的样式类型，但在样式单规则中，这些属性应该位于其他设定的源代码之后。如果要通过一条指令来设定多条边框的样式，请参见border-style属性。

CSS语法

```
border-bottom-style: style
border-left-style: style
border-right-style: style
border-top-style: style
```

值

此时的样式值是一组常量，它们分别对应于一种特定的边框线条显示方式。但并不是所有的浏览器版本均能识别CSS推荐标准中定义的所有常量值。各种浏览器所支持的样式类型如下表所示。

值	IE/Windows	NN	其他	CSS	值	IE/Windows	NN	其他	CSS
dashed	5.5	6	all	1	inset	4	4	all	1
dotted	5.5	6	all	1	none	4	4	all	1
double	4	4	all	1	outset	4	4	all	1
groove	4	4	all	1	ridge	4	4	all	1
hidden	n/a	6	all	2	solid	4	4	all	1

在不同浏览器中，它们对样式值的解释也不尽相同。图4-1显示了IE、Firefox、Safari及Opera 9绘制的各种样式图像。从图中可以看出，不同浏览器的显示效果完全不一样。

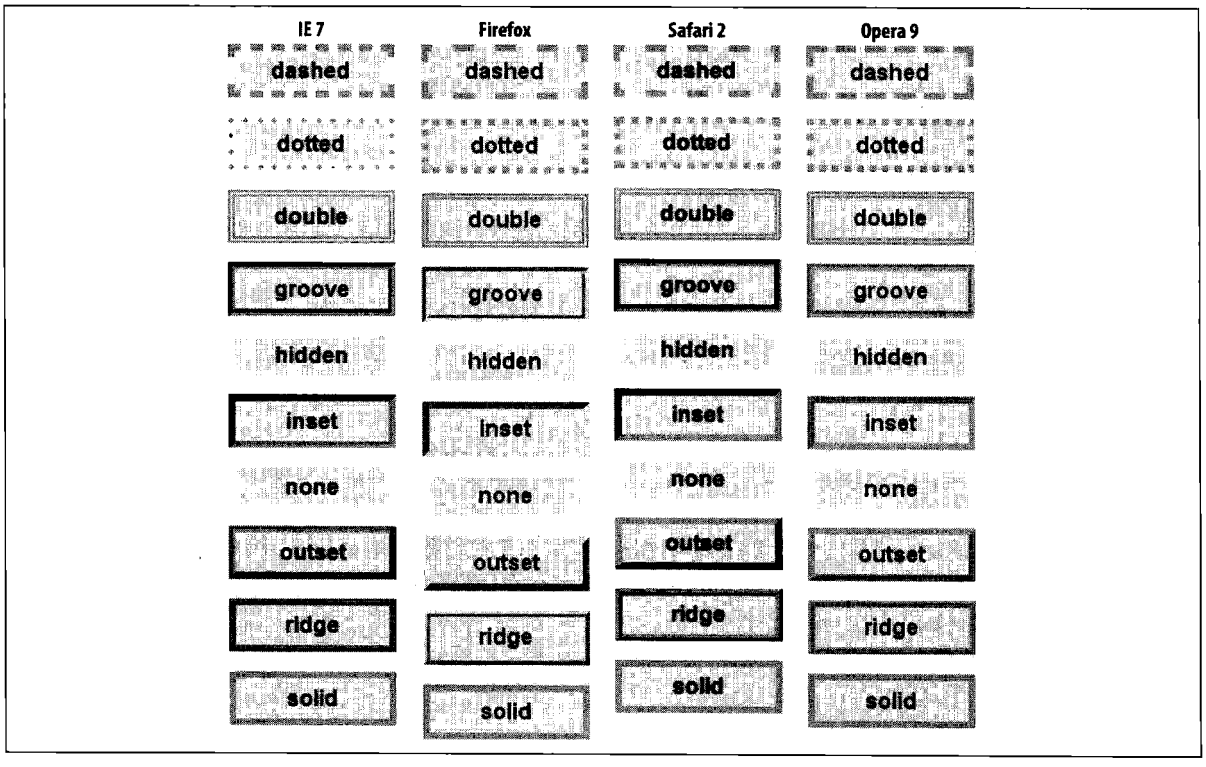


图 4-1：边框样式图库

初始值

none

示例

```
p {border-style: solid; border-bottom-style: none}
div {border-left-style: ridge}
```

应用于

所有元素。

对象模型引用方式

```
[window.]document.getElementById("elementID").style.borderBottomStyle
[window.]document.getElementById("elementID").style.borderLeftStyle
[window.]document.getElementById("elementID").style.borderRightStyle
[window.]document.getElementById("elementID").style.borderTopStyle
```

border-bottom-width, border-left-width, border-right-width, border-top-width

IE *4* NN *4* Moz *all* Saf *all* Op *all* CSS *1*

通过继承得到：否

这几个属性可以分别设定元素对应边框的宽度。如果要通过一条指令来设定多条边框的宽度，请参见border-width属性。

950

CSS语法

```
border-bottom-width: thin | medium | thick | length
border-left-width: thin | medium | thick | length
border-right-width: thin | medium | thick | length
border-top-width: thin | medium | thick | length
```

值 目前有3个可用的常量值，thin、medium和thick。它们决定了浏览器使用多少像素来绘制边框。使用一个长度值可以更为精确地控制边框宽度，请参见本章中有关CSS长度值的讨论。

初始值 medium

示例

```
h2 {border-bottom-width: 2px}
div {border-left-width: thin}
p.special {border-right-width: 0.5em}
```

951

应用于 所有元素，但在Windows系统中的IE 4和5浏览器中，只能应用于块级元素及置换元素。

对象模型引用方式

```
[window.]document.getElementById("elementID").style.borderBottomWidth
[window.]document.getElementById("elementID").style.borderLeftWidth
[window.]document.getElementById("elementID").style.borderRightWidth
[window.]document.getElementById("elementID").style.borderTopWidth
```

border-collapse

IE *5 (Win)* NN *n/a* Moz *all* Saf *all* Op *all* CSS *2*

通过继承得到：是

通过这个属性，可以设定临近的表格元素（如单元格、行分组或列分组）是单独显示还是忽略之间的填充与边距合并在一起。如果将表格设置为分离的边框模式，那么在支持该模式的目标浏览器中，就应该设置其border-spacing及empty-cells样式属性。

CSS语法 border-collapse: collapse | separate

值 常量：collapse | separate。

初始值 separate

应用于 table元素。

border-color

IE *4* NN *4* Moz *all* Saf *all* Op *all* CSS *1*

通过继承得到：否

这个属性可以快捷地为多条边框设定相同或不同的颜色。在Navigator 4中只允许使用一个单独的值，并且会

将该值应用于元素的4条边。而在其他支持这一属性的浏览器中，可以为它设定1~4个颜色值，每个颜色值间需要使用空格进行分隔。而颜色值的数量则决定了由哪些边接受指定的颜色值。

CSS语法 `border-color: color {1,4}`

值

在现代浏览器中，这个属性接受由1~4个颜色值，具体的数量主要取决于将要为哪些边框设定特定的颜色。下表解释了值的数量与位置所代表的实际含义。

值数量	效果
1	将 4 条边框均设置为该值
2	顶部与底部的边框设置为第 1 个值，而左侧和右侧边框设定为第 2 个值
3	顶部边框设置为第 1 个值，左侧和右侧边框设定为第 2 个值，而底部边框设置为第 3 个值
4	依次分别设定顶部、右侧、底部和左侧边框的值

初始值 与元素的color样式属性相同，此外，如果并未明确指定该样式，那么会从父元素中继承其样式值。

示例

```
h2 {border-color: red blue red}
div {border-color: red rgb(0,0,255) red}
```

应用于 所有元素，但在Windows系统中的IE 4和5浏览器中，只能应用于块级元素及置换元素。

对象模型引用方式

```
[window.]document.getElementById("elementID").style.borderColor
```

border-spacing

IE *n/a* NN *n/a* Moz *all* Saf *all* Op *all* CSS *2*

通过继承得到：否

此属性可以定义一个表格内所有单元格边框之间的空间大小。在设定这个属性时，须要将border-collapse设定为separate，通常这是该属性的默认值。如果仅引入了一个长度值，那么会同时在水平和竖直这两个单元格间隔上应用这个值，而当使用了两个值时，会将第1个值应用于水平间距，而将第2个值应用于竖直间距。表格单元格中的各种空间定义请参见图4-2。

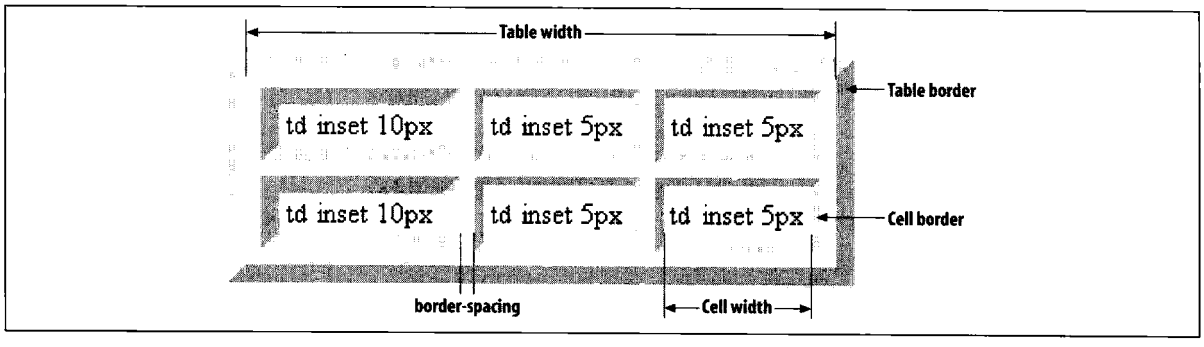


图 4-2：表格元素几何定义

CSS语法 `border-spacing: length[length]`

值 请参见本章中有关CSS长度值的讨论。如果希望在某个轴向上单元格间距为零，那么请将其值设定为0。

初始值 0

应用于 table元素。

border-style

IE 4 NN 4 Moz all Saf all Op all CSS 1

通过继承得到：否

这个属性可以快捷地为多条边框设定相同或不同的样式。而在主流浏览器中，可以为它设定1~4个边框样式值，每个值之间须要使用空格进行分隔。而样式值的数量则决定了由哪些边接受指定的样式。

CSS语法 `border-style: borderStyle {1,4}`

值

此时的样式值是一组常量，它们分别对应于一种特定的边框线条显示方式。具体的可用值请参见`border-bottom-style`。

在不同浏览器中，它们对样式值的解释也完全相同。图4-1显示了IE 7、Firefox、Safari及Opera 9绘制的各种样式图像。从图中可以看出，不同浏览器的显示效果完全不一样。

这个属性可接受1~4个由空格分隔的*borderStyle*样式值，这决定了将要为哪些边框设定特定的样式类型。下表解释了值的数量与位置所代表的实际含义。

值数量	效果
1	将4条边框均设置为该值
2	顶部与底部的边框设置为第1个值，而左侧和右侧边框设定为第2个值。
3	顶部边框设置为第1个值，左侧和右侧边框设定为第2个值，而底部边框设置为第3个值。
4	分别为顶部、右侧、底部和左侧边框设定对应值

初始值 none

示例

```
h1 {border-style: ridge; border-width: 3px}
div {border-style: solid double; border-width: 4px}
```

应用于 所有元素，但在Windows系统中的IE 4和5浏览器中，只能应用于块级元素及置换元素。

对象模型引用方式

```
[window.]document.getElementById("elementID").style.borderStyle
```

954

border-width

IE 4 NN 4 Moz all Saf all Op all CSS 1

通过继承得到：否

这个属性可以快捷地为多条边框设定相同或不同的宽度值。在这个属性值中，可以设定1~4个宽度值，不同宽度值之间须要使用空格进行分隔。而宽度值的数量则决定了由哪些边接受指定的宽度。

CSS语法 `border-width: thin | medium | thick | length {1,4}`

值

目前有3个可用的常量值，thin、medium和thick。它们决定了浏览器使用多少像素来绘制边框。使用一个长度值可以更为精确地控制边框宽度，请参见本章中有关CSS长度值的讨论。

这个属性可接受1~4个由空格分隔的*borderWidth*样式值，这也决定了将要为哪些边框设定特定的样式类型。下表解释了值的数量与位置所代表的实际含义。

值数量	效果
1	将4条边框均设置为该值
2	顶部与底部的边框设置为第1个值，而左侧和右侧边框设定为第2个值。
3	顶部边框设置为第1个值，左侧和右侧边框设定为第2个值，而底部边框设置为第3个值。
4	分别为顶部、右侧、底部和左侧边框设定对应值

初始值 medium

示例

```
h1 {border-style: ridge; border-width: 3px 5px 3px}
div {border-style: solid double; border-width: 4px}
```

应用于 所有元素，但在Windows系统中的IE 4和5浏览器中，只能应用于块级元素及置换元素。

对象模型引用方式

```
[window.]document.getElementById("elementID").style.borderWidth
```

bottom

IE *5* NN *n/a* Moz *all* Saf *all* Op *all* CSS *2*

通过继承得到：否

CSS规范调用这个属性来定义一个已定位元素的边距边缘底部相对于临近最外层块级容器的底边的实际位置。如果已定位元素使用根文档作为定位环境，那么容器块的尺寸就由未滚动文档的浏览器窗口决定。换句话说，在一个位于顶层的已定位元素中，浏览器不会将文档的底部作为相对边缘，而使用浏览器窗口空间的底部作为相对边缘。这意味着元素底部的精确位置会随着用户浏览器窗口的尺寸发生改变。针对一个相对定位元素，元素通常会出现在容器内容中的某个位置，那么就根据内嵌位置的下边缘来决定偏移量。

955

CSS语法 `bottom: length | percentage | auto`

值 请参见本章中有关CSS长度值的讨论。在某些定位环境中还可以使用负值，但此时请在所有的浏览器中进行测试，以保证其正确性。也可以使用百分比值作为其属性值，此时应该根据临近的最外层容器的高度来计算该百分数。如果将此属性设置为auto，那么就会让浏览器自主决定元素盒的底部偏移量在其容器盒内的偏移值。

初始值 auto

应用于 所有已定位元素。

对象模型引用方式

```
[window.]document.getElementById("elementID").style.bottom
```

caption-side

IE *5 (Mac)* NN *n/a* Moz *all* Saf *all* Op *all* CSS *2*

通过继承得到：是

此属性可以决定caption元素是位于table元素中表格内容的上方还是下方。可以用它来代替caption元素中的align属性设定，目前已经不再提倡使用align属性。

CSS语法 `caption-side: top | bottom | left | right`

值 以下4个常量字符串之一：top | bottom | left | right。但CSS2.1已经删除了left和right这两个值。

初始值 top

应用于 caption元素。

对象模型引用方式

```
[window.]document.getElementById("elementID").style.captionSide
```

clear

IE *4* NN *4* Moz *all* Saf *all* Op *all* CSS *1*

通过继承得到：否

此属性可以决定一个块级元素是否允许它自己与临近的浮动元素一起显示在同一个水平区域，例如一个图像元素。在该元素附近往往存在另一个元素，而且其float样式属性值为left或right。如果要让当前元素与浮动块位于不同的水平区域内，请将它们的clear属性设置为同一侧（left或right）。如果无法确定是否会出现重叠的现象，那么可以将clear属性设置为both。如果将元素的clear属性值设置为none之外的任意值，那么它就会显示在浮动元素下一行中的起始位置。

956

CSS语法 `clear: both | left | none | right`

值 以下常量值之一：both | left | none | right。

初始值 none

示例

```
<img src="logo.gif" height="40" width="60" style="float: right">
<h1 style="clear: right">Giantco Corporation</h1>
```

应用于 块级元素。

对象模型引用方式 [window.]document.getElementById("*elementID*").style.clear

Clip

IE *4* NN *4* Moz *all* Saf *all* Op *all* CSS *2*

通过继承得到：否

为可定位元素定义一个裁剪区域。裁剪区是元素层中可以显示元素内容的一个区域。如果在裁剪一个元素时遇到了问题，那么请将包含相关内容的元素放置在一个块级元素之中，并使用目标区域设置其clip属性。

CSS语法 clip: rect(*lengthTop lengthRight lengthBottom lengthLeft*) | auto

值

到CSS2.1为止，clip属性唯一认可的外形为rect。将来可能陆续承认其他的外形。

请按照顺时针方向为裁剪矩形框的每条边指定长度值，即上、右、下、左。请参见本章中有关CSS长度值的讨论。如果将clip属性设置为auto，那么裁剪区域会成为一个包含元素内容的矩形块。

初始值 auto

示例

```
<span style="position: absolute; clip: rect(10px 110px 80px 10px)">
<img src="desk1.gif" height="90" width="120">
</span>
```

应用于 块级元素、置换元素及已定位的元素。

对象模型引用方式 [window.]document.getElementById("*elementID*").style.clip

color

IE *4* NN *4* Moz *all* Saf *all* Op *all* CSS *1*

通过继承得到：是

定义元素中文本的前景色。对于某些图形元素而言，如表单控件等，color属性还可能会影响其元素边框、校验标记或其他元素部件。但这些非正式的颜色设定往往只存在于特定的浏览器，它们在不同的浏览器上效果可能并不相同。

CSS语法 color: *color*

值 请参见本章中与颜色属性值有关的讨论。

初始值 black

示例 th {color: darkred}

应用于 所有元素。

对象模型引用方式 [window.]document.getElementById("*elementID*").style.color

content

IE *n/a* NN *n/a* Moz *all* Saf *all* Op *all* CSS *2*

通过继承得到：否

定义将要在当前元素之前和/或之后显示的实际内容或内容来源。在CSS术语中，这种内容称之为“**生成的内容**”。只有当某个真实元素与“:before”或“:after”伪元素关联起来时才能设定这个属性。例如，此时可以形成如下的样式单规则：

```
blockquote:after {content:"(Reprinted by permission.)"}
```

从上例可以看出，尽管其定义的内容并未成为文档树的一部分，但在blockquote元素之后仍然附加了一个限定语。此时文本内容中的HTML标签不会得到解析，但如果情况允许，也可以使用一个外部文档作为其内容属性。

CSS语法 content: *string* | *uri* | *counter* | attr(*attrName*) | open-quote | close-quote | no-open-quote | no-close-quote | none | normal

值

使用以“no”开头的引用类型就可以指定不显示引号的引用方式。如果要使用多个以空格分隔的字符串，那么应该将它们放置在“content:”属性名之后。

尽管并不是所有的浏览器均能够支持“*counter*”，但它对于那些能够受益于客户端节号生成的文档而言比较有用。针对文章分段、插图等排序元素，CSS的计数器为样式单提供了一种计数控制机制。在这种机制中，假设计数并不是实际内容的一部分，但它应该由文档中已显示的元素上下文环境来决定其具体取值。因此，即使在编辑阶段从文档中删除了一段已计数的文本内容，在显示页面时也会自动调整文档中的段落计数。

958

在实际应用中，只须要为计数器指定一个标示符就可以让它开始执行基本工作，因此可以在同一个文档中使用多个计数器，如可以同时使用一个段落计数器及一个段落单元计数器。其他CSS属性（counter-increment及counter-reset）则须要一些指向已标示的计数器的数值，以便对计数序列进行控制。下面这段样式单规则在每个h1元素之前均插入了一个分段标记及序号，在文档显示过程中，每次将该样式应用于一个h1元素后就会对该计数器加一：

```
h1:before {counter-increment: secNum; content: "Section " counter(secNum) ". "}
```

当主流浏览器中实现计数器时，它们就可以在高度结构化的长文档中发挥有效作用。

初始值 ""（即空字符串）。

示例 p.note:before {content: "==>"}

应用于 所有元素加上一个“:before”和/或“:after”伪元素。

counter-increment, counter-reset

IE *n/a* NN *n/a* Moz *1.8* Saf *n/a* Op *7* CSS *2*

通过继承得到：否

在生成的内容（请参见content属性）中，这两个属性控制着CSS计数器的计数序列。counter-increment属性设定了计数器在文档显示过程中每次改变的趋势与具体数值。而counter-reset属性则可以将计数器设置到某个特定值，其默认值为0。

CSS语法

counter-increment: *counterID* [*posOrNegInteger*] | none
counter-reset: *counterID* [*posOrNegInteger*] | none

值 *counterID*是指定给content: counter(*counterID*)样式属性的一个标识符。*counterID*之后的可选整数值须要使用空格进行分隔。此外，将ID与其对应整数值对组成字符串后，就可以将同一个样式属性中的多个计数器ID组合在一起，该字符串中各个值对之间请使用空格进行分隔。

初始值 none

示例 h1 {counter-reset: subSection}

应用于 所有元素。

cue

IE *n/a* NN *n/a* Moz *n/a* Saf *n/a* Op *n/a* CSS *2*

959

通过继承得到：否

这是一个听觉样式单属性，它可用于快速设定cue-before及cue-after这两个属性。在阅读文档内容时，提示音（通常也称之为听觉按钮）常用于给出分界提示。cue属性通常由声音文件的URI构成。

CSS语法 `cue: cue-before || cue-after`

值 如果存在两个值，那么第1个会用于cue-before属性，而后一个则用于cue-after属性。如果仅存在一个值，那么会为cue-before和cue-after这两个属性值指定同一个属性值。

初始值 none

应用于 所有元素。

cue-after, cue-before

IE *n/a* NN *n/a* Moz *n/a* Saf *n/a* Op *n/a* CSS *2*

通过继承得到：否

这是一个听觉方面的样式单属性，在阅读文档内容时，提示音（通常也称之为听觉按钮）常用于给出分界提示。cue-before及cue-after属性都是声音文件的URI地址，当文本-语音转换程序或其他听觉媒体进行朗读时，会在内容之前及之后播放为它们指定的声音文件。

CSS语法

```
cue-after: uri | none
cue-before: uri | none
```

值 任意一个声音文件的相对或绝对URL（CSS格式）地址，但要保证浏览器能够支持这种MIME类型的声音文件。如果收听者能够听出起始和截止位置，那么也可以在同一个样式选择符中为这两个属性指定相同的值。

初始值 none

示例 `li {cue-before: url(ding.wav); cue-after: url(dong.wav)}`

应用于 所有元素。

cursor

IE *4* NN *n/a* Moz *all* Saf *all* Op *all* CSS *2*

通过继承得到：是

它可以指定屏幕指针位于元素之上时光标的具体样式。光标的精确外观主要依赖于操作系统。因此在部署一个改进的光标时，要首先了解不同类型的光标在浏览器和操作系统中的标准使用方式。用户通常都希望光标在不同应用程序中都能够表达相同的含义。图4-3给出了一份由IE浏览器提供的图集，它绘出了每种光标常量对应的光标图形。

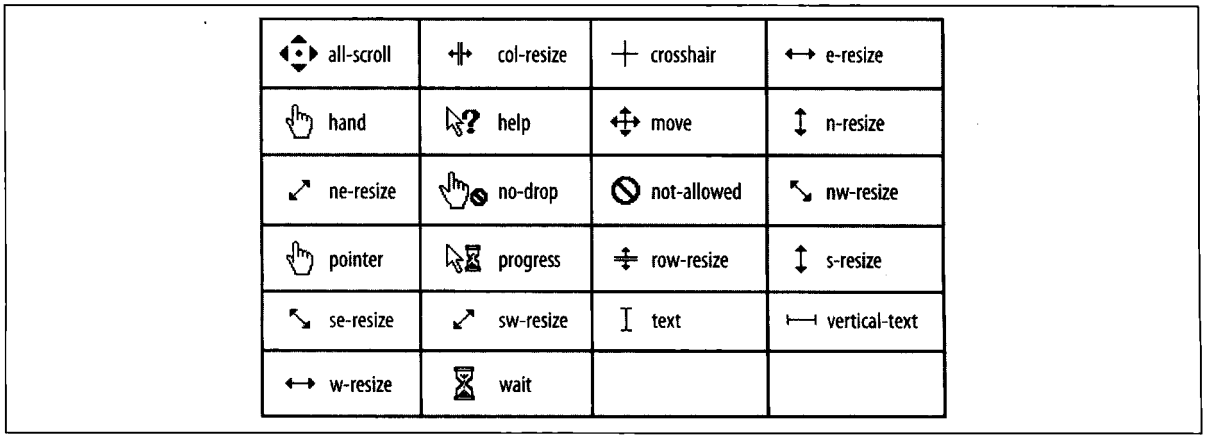


图 4-3：IE 光标图集

CSS语法 `cursor: cursorType || uri`

值

每种光标类型都对应于一种已实现的光标名称。下班中列出了不同浏览器，以及CSS标准所支持的各种光标类型，其中3*表示CSS3的建议值。

光标名称	IE/Windows	IE/Mac	Mozilla	Safari	Opera	CSS
alias	n/a	n/a	n/a	n/a	n/a	3*
all-scroll	6	n/a	1.8	n/a	n/a	3*
auto	4	4	all	all	7	2
cell	n/a	n/a	1.8	n/a	n/a	3*
col-resize	6	n/a	1.8	n/a	n/a	3*
context-menu	n/a	n/a	1.8	n/a	n/a	3*
copy	n/a	n/a	1.8	n/a	n/a	3*
count-down	n/a	n/a	n/a	n/a	n/a	n/a
count-up	n/a	n/a	n/a	n/a	n/a	n/a
count-up-down	n/a	n/a	n/a	n/a	n/a	n/a
crosshair	4	4	all	all	7	2
default	4	4	all	all	7	2
e-resize	4	4	all	all	7	2
grab	n/a	n/a	<1	n/a	n/a	n/a
grabbing	n/a	n/a	<1	n/a	n/a	n/a
hand	4	4	n/a	all	7	n/a
help	4	4	all	all	7	2
move	4	4	all	all	7	2
n-resize	4	4	all	all	7	2
ne-resize	4	4	all	all	7	2
nesw-resize	n/a	n/a	1.8	n/a	n/a	3*
no-drop	6	n/a	1.8	n/a	n/a	3*
none	n/a	n/a	n/a	n/a	n/a	3*
not-allowed	n/a	n/a	n/a	n/a	n/a	3*
nw-resize	4	4	all	all	7	2
nwse-resize	n/a	n/a	1.8	n/a	n/a	3*
pointer	4	4	all	all	7	2
progress	6	n/a	<1	n/a	9	2.1
row-resize	6	n/a	1.8	n/a	n/a	3*
s-resize	4	4	all	all	7	2
se-resize	4	4	all	all	7	2
spinning	n/a	n/a	all	n/a	n/a	n/a
sw-resize	4	4	all	all	7	2
text	4	4	n/a	n/a	n/a	2
url(uri)	6	n/a	n/a	n/a	n/a	2
vertical-text	6	n/a	1.8	n/a	n/a	3*
w-resize	4	4	all	all	7	2
wait	4	4	all	all	7	2

须要注意的是，IE 6及后续版本还实现了可下载的光标。此时IE的外部URL设定还需要一个扩展名为*.cur*或*.ani*的光标文件地址，在该文件中通过图形程序创建了一个Windows光标图像。

初始值 auto
示例 a.helpLink {cursor: help}
应用于 所有元素。

direction

IE *5* NN *n/a* Moz *all* Saf *all* Op *all* CSS *2*

通过继承得到：是

设置内容中内嵌部分（如文本等）的流动方向，以及表格单元格沿列方向的填充顺序。与大多数元素的dir属性相比，direction样式属性可以替代浏览器为其他语言或特定内容所指定的默认显示方向。

962

CSS语法 direction: ltr | rtl
值 可以从两个方向常量中选择一个。ltr表示从左到右，而rtl则表示从右到左。
初始值 ltr
应用于 所有元素。

display

IE *4* NN *4* Moz *all* Saf *all* Op *all* CSS *1*

通过继承得到：否

这是一个多用途的属性，它可以决定浏览器如何处理环绕在每个元素及文本节点四周的隐形框。例如，块级元素的显示特征就与一个行内元素的显示效果大相径庭。由于显示框所占据的空间会受到边框、显示规则等各种外部因素的影响，因此CSS规范为它提供了很多种类型。由于浏览器的内置样式单并不会为不同变量指出的具体不同之处，因此在实际使用中很难发觉不同显示类型之间的差异，例如，将table的display样式属性设置为block或table时，显示方式就完全相同。与此同时，display样式还可以替代元素的默认显示效果，例如将一个块状表显示为一个内嵌表。

此外，display设定还可以应用于任意其他类型的元素，如divs及spans，这样就可以根据其值名称赋予这些元素具体的显示效果。例如，如果要按照表格的形式显示一个层级元素集合（如一个XML查询结果），那么只须为集合中的元素（分别对应于分组、行和单元格）指定表格相关的display属性值就可达到预期的效果。

此外，在DHTML中，display样式属性常用于通过脚本完成元素显示与完全隐藏的转化工作。当属性值为“none”时，就会隐藏对应的元素，而元素四周的内容就会立即填满该元素原来所占据的空间位置。这与visibility属性完全不同，visibility在隐藏元素的同时还会保留其空间位置。如果要恢复元素的默认显示状态，那么在浏览器能够给予支持的情况下，可以为该元素重新设定一个常用的显示类型（block和inline）或与元素有关的特定显示样式（如为li元素设定list-item样式）。

CSS语法 display: *displayType*

963

值

虽然CSS规范认可很多种显示类型，但浏览器能够支持的类型极为有限。下表中列出了不同浏览器对显示类型的支持情况。如果浏览器能够支持list-item，以及与表格相关的所有值，那么就意味着可以在所有容器中正常使用这些属性值。

显示类型	IE/Windows	IE/Mac	NN	Mozilla	Safari	Opera	CSS
block	5	4	4	all	all	7	2
compact	n/a	n/a	n/a	n/a	n/a	n/a	<2.1
inline	5	4	4	all	all	7	2
inline-block	5.5	n/a	n/a	n/a	n/a	n/a	<2.1
inline-table	n/a	5	n/a	n/a	all	7	2
list-item	5	5	n/a	all	all	7	2

续表

显示类型	IE/Windows	IE/Mac	NN	Mozilla	Safari	Opera	CSS
marker	n/a	n/a	n/a	n/a	n/a	n/a	<2.1
none	4	4	4	all	all	7	2
run-in	n/a	5	n/a	n/a	n/a	7	2
table	n/a	5	n/a	all	all	7	2
table-caption	n/a	5	n/a	all	all	7	2
table-cell	n/a	5	n/a	all	all	7	2
table-column-group	n/a	5	n/a	n/a	n/a	n/a	2
table-footer-group	5.5	5	n/a	all	all	7	2
table-header-group	5	5	n/a	all	all	7	2
table-row	n/a	5	n/a	all	all	7	2
table-row-group	n/a	5	n/a	n/a	n/a	n/a	2

初始值 由元素的具体类型决定。
示例 `.hidden {display: none}`
应用于 所有元素，但某些显示类型仅适用于特定元素。
对象模型引用方式 `[window.]document.getElementById("elementID").style.display`

elevation

IE *n/a* NN *n/a* Moz *n/a* Saf *n/a* Op *n/a* CSS *2*

通过继承得到：否

elevation用于设置声音源（如一个文本-语音转换浏览器）的竖直角度，从而让收听者位于三维声音空间的中心，这类似于拥有环绕立体声设备的剧院。另请参见azimuth属性。

CSS语法 `elevation: angle | angleConstant`

值

可以使用一个特定的角度（单位为度）或5个常量之一作为属性值。角度值的取值范围为-90到+90之间，单位为“deg”，如“90deg”。“0deg”表示声源与收听者的耳朵位于同一个竖直层面。如果要让声源角度位于该层面之上，那么必须使用一个正值，如“45deg”，反之则须要使用负的角度值，如“-45deg”。也可以从一组角度说明常量中选择一个作为*angleConstant*的值，而这些常量值对应于位于收听层面之上及之下的一些固定点。

值	等价数值	值	等价数值
above	90deg（位于收听者正上方）	level	0deg（与收听者的耳朵齐平）
below	-90deg（位于收听者正下方）	lower	从当前位置向下移动 10 度
higher	从当前位置向上移动 10 度		

与azimuth属性组合后，就可以将声源放置在“球形环绕声音舞台”中的任意一点上。

初始值 level
示例

```
h1 {elevation: -45deg}
p.heavenly {elevation: above}
```

应用于 所有元素。

empty-cells

IE *n/a* NN *n/a* Moz *all* Saf *1.3/2* Op *8* CSS *2*

通过继承得到：否

在表格中此属性控制着是否显示空td元素的边框和背景。隐藏空单元格时，并不会改变四周单元的实际位置。此时空单元格会完全透明，表格背景就会透过这些空单元格显示出来。

CSS语法 empty-cells: show | hide
值 两个常量值之一：show | hide。
初始值 show
示例 td {border: salmon inset 3px; empty-cells: hide}
应用于 td元素。

filter（旧样式）

IE 4 (Win) NN n/a Moz n/a Saf n/a Op n/a CSS n/a

通过继承得到：否

设定视觉、显示或混合过滤器，以用于显示或改变元素内容。视觉过滤器可以让元素内容产生旋转、发光、投影等诸多特效。在改变元素的可见性或显示性时就会使用显示滤镜，显示过滤器的值决定了元素在从隐藏到显示或从显示到隐藏时使用哪种视觉效果。这些效果包括划入-划出、百叶窗，以及挡光板等。而混合过滤器则设置了能见度状态转换的速度。

CSS语法

filter: *filterType1*(*paramName1*=*value1*, *paramName2*=*value2*,...)
filterType2(*paramName1*=*value1*,...) ...

值 每个filter属性都可能拥有多个相关的过滤器类型，而不同类型间均使用空格进行分隔。每个过滤器类型后都跟着一对圆括号，这样就可以为当前元素传递过滤器行为参数。每个参数通常都由一个名称/值对组成，并使用等号完成赋值过程。*filterType*的取值与参数请参见下文中“注意事项”。

初始值 无

示例

```
.fastStuff {filter: blur(add=true, direction=225)}
```

应用于 body、button、img、input、marquee、table、td、textarea、tfoot、th、thead、tr元素，以及绝对定位的div和span元素。

对象模型引用方式

[window.]document.getElementById("*elementID*").filters["*filterName*"]

注意事项 第1代过滤器（至少会一直使用到IE 7）大致可以划分为3种类型：视觉、显示及混合。每种类型都拥有一组参数名称。不妨在一个过滤器属性赋值中混合若干种过滤器，并且对不同的组合进行一些有趣实验。但在指定过滤器时，要仔细遵守对应元素各种的限制因素。

视觉过滤器及其参数如下所示。

alpha()

控制透明程度。opacity及finishopacity的值可以介于透明（0）与不透明（100）之间。style参数用于设定不透明的梯度形状，其中包括：均匀（0）、线性（1）、放射性（2）或矩形（3）。startX和startY设定了不透明梯度的起始位置的水平坐标和竖直坐标，而finishX及finishY则设定了结束位置的坐标。

blur()

设定元素的出现方式。add参数指明了将原始图像添加到模糊图像（1）还是直接忽略（0）。direction则设定了模糊图像相对于原始图像所处位置的角度，可用的角度常量为：above（0）、above-right（45）、right（90）、below-right（135）、below（180）、below-left（225）、left（270）、above-left（315）。而strength则指出了拉伸模糊图片时具体像素值。

chroma()

设置色彩透明度。color参数指定了将要产生透明效果的十六进制三元组颜色值。

dropShadow()

为视深创建一个偏移阴影。color参数指定了将要产生投影效果的十六进制三元组颜色值。offx和offy分别指定了投影和元素在两条轴向上的像素距离，正值表示右/下，负值则表示左/上。positive参数用于指定是否只允许正像素值产生投影（1）或也允许为透明像素建立阴影效果（0）。

fliph()
为元素生成一个水平镜像。

flipV()
为元素生成一个竖直镜像。

glow()
在外边缘添加光晕效果。color参数指定了光晕效果的十六进制三元组颜色值，而strength则设定了光晕强度（1-255）。

gray()
去掉元素的色彩但保留其亮度，即使用黑白两色来显示元素。

invert()
反转元素的色调、饱和度，以及亮度（HSV）等级。

light()
在元素上产生一个模拟光源，目前有很多滤镜方法可用于设定特定的光源类型、位置、强度和颜色等。

mask()
在元素上生成一个透明的遮罩。color参数指定了透明区域所采用的十六进制三元组颜色值。

shadow()
显示元素的立体投影。color参数指定了阴影区域所采用的十六进制三元组颜色值，direction则设定了阴影相对于原始图像所处位置的角度，可用的角度常量为：above（0）、above-right（45）、right（90）、below-right（135）、below（180）、below-left（225）、left（270）、above-left（315）。

wave()
沿x轴对元素图像进行正弦波失真并进行显示。add参数指明了是将原始图像添加到波浪图像（1）还是直接忽略（0）。freq设定了视觉失真中使用的波纹频率，light设定了光强（0至100），phase设定正弦波的偏移百分值（0到100分别对应于0到360度），而strength则设定了光效强度（0至255）。

xRay()
仅显示元素边缘轮廓。

混合与显示转换滤镜及其相关参数如下所示：

blendTrans()
为元素显示产生淡入淡出效果。duration参数设置了变换效果持续时间所对应的浮点值（*seconds.milliseconds*）。

revealTrans()
在元素不同显示状态之间设定一个切换效果。duration参数设置了切换效果持续时间所对应的浮点值（*seconds.milliseconds*）。transition是一个很关键的整数值，它对应于以下切换类型之一。

值	切换类型	值	切换类型
0	从大至小的矩形	12	随机溶解
1	从小至大的矩形	13	从上下向中间生成
2	从大至小的圆形	14	从中间向上下展开
3	从小至大的圆形	15	从两边向中间生成
4	向上推开	16	从中间向两边展开
5	向下推开	17	从右上向左下展开
6	向右推开	18	从右下向左上展开
7	向左推开	19	从左上向右下展开
8	垂直形百叶窗	20	从左下向右上展开
9	水平形百叶窗	21	随机水平细纹
10	水平棋盘	22	随机垂直细纹
11	垂直棋盘	23	随机选取一种特效

这两种转换滤镜均拥有3个方法：apply()、play()及stop()。在改变元素的可见性属性或其他视觉属性时，可使用apply()来暂停元素的动画显示，而在滤镜上调用play()方法则可以让用户看到其切换动画，示例代码如下所示：

```
document.getElementById("myImg").filters["revealTrans"].apply();
document.getElementById("myImg").src = "newPix.jpg";
document.getElementById("myImg").filters["revealTrans"].play();
```

该元素的样式单规则可进行如下设置：

```
img {filter: revealTrans(transition=2, duration=3)}
```

执行脚本指令时，从一个图片转换到另一个图片过程中会采用“从大至小的圆形”的切换效果。

Filter（新样式）

IE *5.5* NN *n/a* Moz *n/a* Saf *n/a* Op *n/a* CSS *n/a*

通过继承得到：否

在DXImageTransform ActiveX控件的帮助下，它可以设定用于显示或改变元素内容的静态或切换滤镜，但目前只能在Windows系统下IE 5.5之后的浏览器中使用这一属性。这种新滤镜机制的设计意图与旧样式相同，但调用ActiveX控件的语法是全新的，其中包含很多新的滤镜名称。

CSS语法

```
filter: progid:DXImageTransform.Microsoft.filterType1(paramName1=value1,
paramName2=value2,...)
progid:DXImageTransform.Microsoft.filterType2(paramName1=value1,...) ...
```

值 在使用滤镜类型之前必须首先引用ActiveX控件（progid:DXImageTransform.Microsoft.），并且在一个filter类型属性上使用多个滤镜类型时须要使用空格进行分隔。每个滤镜类型后都跟着一对圆括号，这样就可以为当前元素传递滤镜的行为参数。每个参数通常都由一个名称/值对组成，并使用等号完成赋值过程。控制着元素切换的滤镜类型也拥有一些特定的方法，当改变元素的某些可视属性时，可以通过脚本来调用这些方法以便暂停显示，然后再播放新的切换动作。*filterType*的取值与参数请参见下文中的“注意事项”。

初始值 无。

示例

```
.fastStuff {filter: progid:DXImageTransform.Microsoft.MotionBlur(add=1, direction=225)}
```

应用于 所有元素。

对象模型引用方式

```
[window.]document.getElementById("elementID").filters["DXImageTransform.Microsoft.filterName"]
```

注意事项

由于这个ActiveX控件只能在IE/Windows组合中正常工作，因此对它进行详细说明将会超出本书的核心领域。本章中仅简单介绍了这种仅适用于IE 5.5+的新滤镜机制，下表中列出了DXImageTransform ActiveX控件中可用的各种静态和切换滤镜，以及其工作方式的简单说明。与样式单规则有关的特定属性细节，以及每个滤镜中可使用的属性和方法请访问如下链接：*http://msdn.microsoft.com/en-us/library/ms532847(VS.85).aspx*。

滤镜名称	类型	描述
Alpha()	静态	控制透明或不透明程度。
Barn()	切换	采用模拟开关门效果进行切换，它包含速度、动作，以及方位等多个属性
BasicImage()	静态	为所有的元素类型设定多种滤镜类型，如镜像、不透明、灰化等，在脚本控制下还可以对元素进行旋转并改变其颜色遮罩
Blinds()	切换	采用百叶窗开关效果进行切换，它还包含方向及窗条厚度等多个属性
Blur()	静态	控制元素的模糊程度

续表

滤镜名称	类型	描述
Checkerboard()	切换	采用类似国际象棋棋盘的网格推拉效果进行切换，它包含方向、速度，以及方块大小等多个属性
Chroma()	静态	控制特定颜色的透明度
Compositor()	静态	合成色彩滤镜效果
DropShadow()	静态	生成阴影效果，它包含颜色及阴影深度等多个属性
Emboss()	静态	控制浮雕纹理效果
Engrave()	静态	控制镂雕纹理效果
Fade()	切换	在不同视图之间实现渐隐切换效果，它包含速度，以及重叠度等多个属性
Glow()	静态	控制外边缘的光晕效果
Gradiant()	静态	在元素背景上应用色彩梯度效果
GradientWipe()	切换	在擦除线上使用梯度混合的擦除渐变效果，它包含速度、梯度厚度，以及方向等多个属性
ICMFilter()	静态	在元素上应用一个外部的图像颜色管理配置文件（Image Color Management profile）
Inset()	切换	沿着水平轴与竖直轴，控制对角扩张的切换效果
Iris()	切换	控制一种缩放切换效果，它包含速度、方向（缩或放），以及剪切轮廓（如圆形、十字形、钻石形及方形和星形）等多个属性
Light()	静态	通过脚本的特殊控制，为元素提供直接的光照效果
MaskFilter()	静态	为某种颜色提供一个透明遮罩
Matrix()	静态	控制对象内容的旋转、反转及元素尺寸调整
MotionBlur()	静态	通过人工模糊化来模拟运动效果
Pixelate()	切换	让目标内容不断地进行扩展/收缩和模糊/聚焦变化
RadialWipe()	切换	通过不同的样式选择，如时钟状、楔状或放射状，动态切换视图效果
RandomBars()	切换	通过扩展/收缩线条动态切换视图效果，它包含方向和速度等属性
RandomDissolve()	切换	使用随机像素改变效果来切换视图显示
Shadow()	静态	为元素内容建立阴影效果
Slide()	切换	使用滑条抽离效果来切换视图显示
Spiral()	切换	使用螺旋变形效果来切换视图显示
Stretch()	切换	使用不同的拉伸变形效果来切换视图显示
Strips()	切换	使用条纹效果来切换视图显示
Wave()	静态	在元素上附加正弦波失真效果
Wheel()	切换	使用以元素中心为核心的轮辐发散效果来切换视图显示
ZigZag()	切换	通过横块擦除效果来切换视图显示

值得注意的是，如果成功部署这些滤镜，尤其将它们应用于复杂的内容时，那么就必须通过广泛的实验与测试，以保证使用中的滤镜不会使浏览器崩溃。

float

IE *4* NN *4* Moz *all* Saf *all* Op *all* CSS *1*

通过继承得到：否

指定元素应该排列在容器盒的哪一条边上，然后其他内容就可以环绕在元素周围。将这个属性设置为none后，会根据其源代码顺序显示元素，而且在元素所在的水平区域内至少会出现有一行环绕文本内容。

由于JavaScript已经将float作为一个其语言中的一个保留关键字，因此在使用JavaScript的对象模型中就无法

再将它作为style对象的一个属性名称。为了规避这一问题，IE采用“styleFloat”作为属性名称，而W3C DOM则使用“cssFloat”。

CSS语法 `float: alignmentSide | none`

值 *alignmentSide*可以是left或right常量之一。

初始值 none

示例 `img.navButton {float: right}`

应用于 除已定位元素或生成的内容之外的所有元素。

对象模型引用方式

```
[window.]document.getElementById("elementID").style.styleFloat
[window.]document.getElementById("elementID").style.cssFloat
```

font

IE 4 NN n/a Moz all Saf all Op all CSS 1

通过继承得到：是

通过这个属性可以在一条语句内快速地设定多个与字体相关的属性。尽管在属性的必选和可选值要求方面，某些浏览器比其他浏览器要宽松得多，但至少应该为这个属性依次指定*font-size*属性及字体（通过*font-family*或*CSS2FontConstant*）属性。而此属性值中其他以空格分隔的值类型的排列顺序并没有严格要求。在CSS2中，某些附加的简写常量还应用了一些已命名的系统字体，对于与字体相关的属性而言，这些字体均对应于固定值。

CSS语法 `font: font-style || font-variant || font-weight || font-size[/line-height] || font-family | CSS2FontConstant`

值

请参见相关的属性说明，以便了解各种字体属性与行属性的语法及使用示例。在*line-height*值前的正斜线（“/”）表示允许使用第2个长度值（其值仍然必须位于取值范围之内）。须要注意的是，*line-height*长度值必须位于font-size之后，而不能单独使用。

CSS2字体常量包括：caption、icon、menu、message-box、small-caption、status-bar。这些常量与客户端使用的浏览器与操作系统有关。因此在不同操作系统上，它们的显示外观可能也会存在不同，但某些特殊的字体类型仍会符合用户的预期效果。换句话说，只有当字体样式能够真实反映系统或浏览器的效果时才应该使用这些样式。

初始值 无。

示例

```
body {font: 12px serif}
h2 {font: bolder small-caps 16px "Lucida Console", Arial, sans-serif}
.iconCaption {font: 10px/1.1em caption}
```

应用于 所有元素。

对象模型引用方式 `[window.]document.getElementById("elementID").style.font`

font-family

IE 4 NN 4 Moz all Saf all Op all CSS 1

通过继承得到：是

此属性为显示对象内容设置了一个按优先次序排序的字体系列。在这个属性值中可以包含一个或多个使用逗号分隔的字体簇名称。如果一个字体簇名称由多个单词组成，那么应该将它放置在一对单引号之中。

在一个字体簇中可能包含多个字体定义。例如，Helvetica字体簇可能包含一个粗体字体和一个斜体字体，这

两种字体完全不同，而不是粗体和斜体的混合版本。通过名称指定字体簇时，浏览器会在客户端系统中进行查询，以确定是否已经存在同名的可用字体。如果不存在，那么浏览器会接着查看列表中的下一个字体簇。因此，在字体簇名称序列中，最好首先放入不常见的字体名，然后再放入常用字体。最终获得的字体簇名称应该是一个通配字体（如serif、sans-serif、cursive、fantasy或monospace），并且要使之最接近类似的目标字体。但值得注意的是，在某个操作系统上已广泛安装的一些字体并不见得流行于另一个操作系统。

遵循CSS2规范的浏览器应该能智能识别Unicode字符编码，并且能够找到为特定语言提供服务的已命名字体簇。当然，如果已经获得了font-family的值，那么最理想的情况是让浏览器将来自于同一个元素中不同语言与书写系统的字体完全混合在一起，这样就不会出现无字体可用的窘境。

CSS语法 font-family: *fontFamilyName [, fontFamilyName [, ...]]*

值 任意数量的字体簇名称，请使用逗号进行分隔。使用时必须将包含多个单词的字体簇名称放置在一对单引号之中。可识别的常用字体簇名称包括：serif | sans-serif | cursive | fantasy | monospace。

初始值 由浏览器决定默认值。

示例 body {font-family: "Century Schoolbook", Times, serif}

应用于 所有元素。

对象模型引用方式 [window.]document.getElementById("*elementID*").style.fontFamily

注意事项

当浏览器未包含页面所需的特殊字体时，IE浏览器提供了一种字体定义文件下载功能。网页开发者必须通过浏览器指定的字体转换工具来开发这类字体定义文件。通过@font-face样式单规则就可以下载字体定义文件，并且将该字体描述与任意的字体簇名称联系起来，例如：

```
@font-face {font-family: Neato; src: url(http://www.giantco.com/fonts/neato.eot}
```

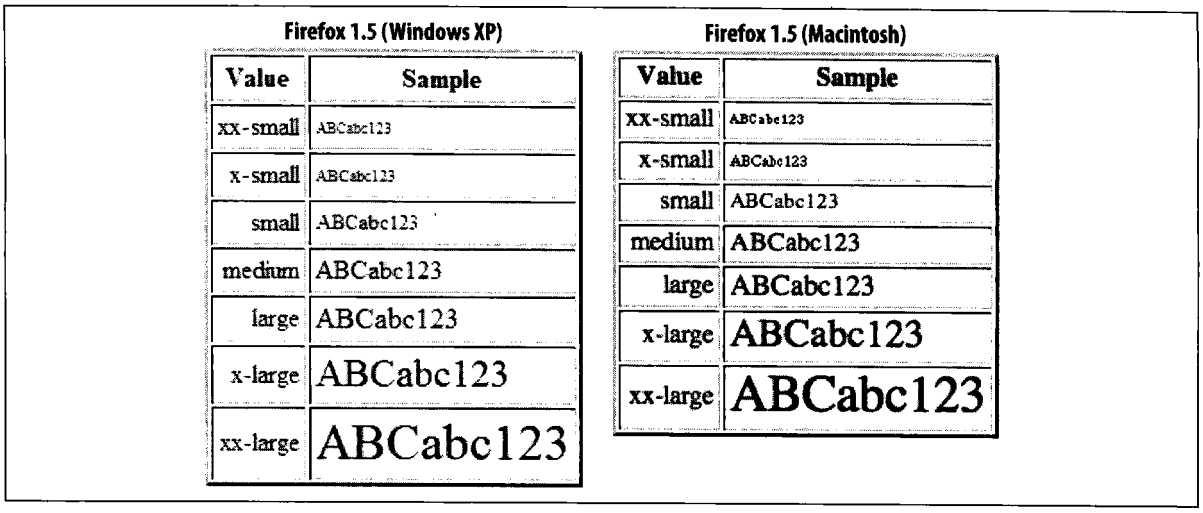
关于如何部署这种类型的样式规则的详细信息，请参见本章前文中的“@规则”一节。进行这些操作后，就可以在常规的font-family样式属性中指定所需的字体。如果字体正在下载过程中，那么浏览器会用另一种字体来显示页面，直到完成字体下载。从这一点来说，页面会随着字体下载的完成而重新排列。

font-size

IE *4* NN *4* Moz *all* Saf *all* Op *all* CSS *1*

通过继承得到：是

定义元素内的字体尺寸。可以通过多种方式设置字体尺寸。xx-small、x-small、small、medium、large、x-large和xx-large等常量定义了一种绝对尺寸。事实上，由于这些尺寸的参考点会随着浏览器和操作系统而发生变化，因此这种“绝对”也能在一个个独立的浏览器和操作系统中奏效。图4-4对同一个视频监控器上的不同字体尺寸进行了比较。但在同一个环境中，设置为large的元素确实比medium类型的元素要大一些，从这个角度来看，它们的确还是为网页开发者提供了一定的便利。



Firefox 1.5 (Windows XP)

Value	Sample
xx-small	ABCabc123
x-small	ABCabc123
small	ABCabc123
medium	ABCabc123
large	ABCabc123
x-large	ABCabc123
xx-large	ABCabc123

Firefox 1.5 (Macintosh)

Value	Sample
xx-small	ABCabc123
x-small	ABCabc123
small	ABCabc123
medium	ABCabc123
large	ABCabc123
x-large	ABCabc123
xx-large	ABCabc123

图 4-4：在 Windows 和 Mac 操作系统平台下，Firefox 1.5 中字体尺寸常量的显示效果

此外还存在另一组称之为相对尺寸的常量——lager、smaller。由于font-size样式属性继承自其父元素，因此会在父元素中应用这些相对尺寸以决定当前元素的字体尺寸。但最终还是由浏览器来决定这些字体尺寸的大小，而且很大因素上还取决于其父元素的具体字体尺寸。如果父级元素使用了一个绝对尺寸，如large，那么当其子元素的相对尺寸为larger时，意味着它在浏览器中显示的字体尺寸是x-large。当父元素的字体尺寸使用长度或百分比时，子元素字体尺寸的增加量并不确定。

如果要在font-size属性中使用一个长度值，那么请选择一个能够让字体在输出媒体上处于最佳显示状态的长度单位，例如，在屏幕显示时请使用像素（px），而在打印输出时则请使用磅（pt）或ems（em）。而使用em作为单位时，会根据父级元素的字体尺寸计算子元素的字体尺寸。最后，如果将font-size设置为一个百分比值，那么也会根据父级元素的字体尺寸估算子元素的字体大小。

973

须要注意的是，某些浏览器会忽略开发者对字体大小的精确控制，转而使用它们自身（或用户）的设定来确立一个“媒体”尺寸。正是出于这种原因，很多设计者倾向于使用尺寸相关的常量来制定其font-size规范机制。但这样做就意味着放弃了在不同浏览器和操作系统之间对绘制效果的控制权，而且在显示时使用这种方式还须要建立非常严格的控制手段，而这往往会导致全面失败。

CSS语法 font-size: *absoluteSize* | *relativeSize* | *length* | *percentage*

值 对于一个绝对尺寸，可以使用以下常量之一：xx-small | x-small | small | medium | large | x-large | xx-large。对于一个相对尺寸，则可以使用以下常量之一：larger | smaller。关于长度值，请参见本章前文中有关CSS长度值的讨论。而使用百分比时，应使用“百分值%”的形式。

初始值 在body元素中为medium，而在所有其他元素中则为其父级元素的font-size值。

示例

```
body {font-size: 14pt}
p.teeny {font-size: x-small}
em {font-size: larger}
span.larger {font-size: 150%}
```

974

应用于 所有元素。

对象模型引用方式 [window.]document.getElementById("*elementID*").style.fontSize

font-size-adjust

IE *n/a* NN *n/a* Moz *n/a* Saf *n/a* Op *n/a* CSS *<2.1*

通过继承得到：是

在替换字体时，允许元素保留“首选”字体的x-height（以ex为单位）。z因子是em类型的字体与x-height字体的转换比例。在同一页面中，由于不同字体即使使用了相同的字体尺寸也会存在一定的大小差异，因此可以使用z因子来计算转换比例并将它应用于其他字体。即使最终的字体尺寸仍然会与“首选”字体设定存在一定的出入，但感觉上它们的字体大小会非常接近。这也有利于字体的横向排列，即使用不同的字体簇也会在同一位置产生换行。

CSS语法 `font-size-adjust: 0.47`

值 用于描述首选字体（可从字体生产商处获得）的外观值，或者none。

初始值 none

应用于 所有元素。

对象模型引用方式

`[window.]document.getElementById("elementID").style.fontSizeAdjust`

font-stretch

IE *n/a* NN *n/a* Moz *n/a* Saf *n/a* Op *n/a* CSS *<2.1*

通过继承得到：是

根据指定的字体簇为已显示的字体设定字符间距。

CSS语法 `font-stretch: stretchType | normal`

值 对于一个绝对尺寸，可以使用以下常量之一：ultra-condensed | extra-condensed | condensed | semi-condensed | semi-expanded | extra-expanded | ultra-expanded。对于一个相对尺寸，可以使用以下常量之一：narrower | wider。

初始值 normal

应用于 所有元素。

对象模型引用方式 `[window.]document.getElementById("elementID").style.fontStretch`

font-style

IE *4* NN *4* Moz *all* Saf *all* Op *all* CSS *1*

通过继承得到：是

从正常、斜体和倾斜这三者中选择一个作为元素内文本的字体样式。如果font-family中包含标注为斜体或倾斜的字体类型，那么font-style属性会从浏览器的系统中调用这些特殊的字体。但如果系统中并不存在指定的字体，那么就会倾斜普通的字体使它趋近于斜体字。针对指定的字体设置，会根据客户端计算机和打印机的品质生成一个电子字体类型并输出至打印机。个人电脑软件通常会引入其他类型的字体，对于其他类型的字体“样式”请参见font-variant及font-weight。

CSS语法 `font-style: fontStyle`

值 以下常量之一：normal | italic | oblique。对斜体与倾斜这两种设置，浏览器倾向于采用相同的处理方式。

初始值 normal

示例

```
h2 em {font-style: italic}
```

应用于 所有元素。

对象模型引用方式 `[window.]document.getElementById("elementID").style.fontStyle`

font-variant

IE 4 NN n/a Moz all Saf 1.3/2 Op all CSS 1

通过继承得到：是

指定是否用小型大写字母的形式显示所有的字符，此时会将代码中的小写字母全部显示为较小的大写字母。如果在字体簇中正好包含一个小型大写字母的字体变量，那么浏览器应该自动使用该字体。然而浏览器很可能直接计算出一个较小的字体尺寸，并用它设定小型大写字母的尺寸。但在Windows系统中，IE 6之前的IE浏览器会根据父级元素的字体样式使用大写字母来显示整个源代码中的内容。

CSS语法 `font-variant: fontVariant`

值 可使用如下的常量值之一：`normal | small-caps`。

初始值 `normal`

示例 `em {font-variant: small-caps}`

应用于 所有元素。

对象模型引用方式

976 `[window.]document.getElementById("elementID").style.fontVariant`

font-weight

IE 4 NN n/a Moz all Saf all Op all CSS 1

通过继承得到：是

此属性用于设定元素字体的权重（粗细）。CSS提供的权重比例机制比大多数浏览器显示在屏幕上的内容要精细得多，但一旦将这些内容输出到打印机，这些经过精细调整的字体就会生效。字体的权重值从100递增到900，而每次递增都应以100为单位。因此，在显示字符时最细的情况是`font-weight`值等于`100`，而它等于900时则为最粗的情况。`normal`等价于`font-weight`为`400`，这是所有字体的默认粗细，标准设定为`700`。其他设定，如`bolder`和`lighter`，则表示根据父级元素的字体粗细指定一个相对权重。

关于权重值与字体簇名称的对应关系，以及某些字体定义格式的内部特性，都由CSS2规范给出了实现准则。例如，OpenType字体定义格式就为9种字体权值提供了接口。在这种情况下，`font-weight`的数字属性值就可以直接与该字体的权值定义对应起来。如果字体簇包含中某个字体名称包含单词“Medium”，以及一个标记的“Book”、“Regular”、“Roman”或“Normal”，那么“Medium”字体的权重值就等于500，而其他的则为400。而字体名包含单词“Bold”的字体的权值则为700。对于那些并未包含全部9种权值的字体簇而言，浏览器应该能够进行一定地自动调整，但此时可能会造成某些权值生成的字体大小与其他权值生成的大小完全相同。

CSS语法 `font-weight: fontWeight`

值 可使用如下的常量值之一：`bold | bolder | lighter | normal | 100 | 200 | 300 | 400 | 500 | 600 | 700 | 800 | 900`。

初始值 `normal`

示例 `p em {font-weight: bolder}`

应用于 所有元素。

对象模型引用方式 `[window.]document.getElementById("elementID").style.fontWeight`

height

977

IE 4 NN 4 Moz all Saf all Op all CSS 1

通过继承得到：否

设置块级元素、置换元素，以及已定位元素中内容的高度，此时不包括边框、填充和边距。

但在IE 6的标准兼容模式下计算元素的高度时，IE/Windows会将顶部与底部的边距、填充，以及边框考虑在内，请参见第1章中的DOCTYPE元素。如果浏览器遵循CSS标准，那么高度只会应用于元素的内容部分，而不应包括其边框、填充或边距。

CSS语法 `height: length | percentage | auto`

值 请参见本章中有关CSS长度值的讨论。如果将此值设定为auto，那么就会让浏览器根据显示内容所需的空间大小来决定元素盒的高度。

初始值 auto

示例

```
div#announce {height: 240}
textarea {height: 90%}
```

应用于 Navigator 4：所有的绝对定位元素；IE 4：applet、div、embed、fieldset、hr、iframe、img、input、marquee、object、span、table及textarea元素；IE 5、Mozilla、Safari和Opera：除无法替代的行内元素、表格列元素及表格分组元素之外的所有元素。

对象模型引用方式 `[window.]document.getElementById("elementID").style.height`

ime-mode

IE *5 (Win)* NN *n/a* Moz *n/a* Saf *n/a* Op *n/a* CSS *n/a*

通过继承得到：否

在IE/Windows中控制是否让输入法编辑器出现在浏览器中，这种输入法编辑器可以支持中文、日文和韩文等多种语言。

CSS语法 `ime-mode: active | auto | disabled | inactive`

值 以下常量值之一：active | auto | disabled | inactive。

初始值 auto

示例 `input {ime-mode: active}`

应用于 input和textarea元素。

对象模型引用方式 `[window.]document.getElementById("elementID").style.imeMode`

!important

IE *4* NN *n/a* Moz *all* Saf *all* Op *all* CSS *1*

通过继承得到：否

在层叠次序方面增加一个属性设定的权重（重要性）。这个关键字实质上是一个声明，而不是一个属性，但它却可以影响所有的属性设定。在进行指令编写时，须要在属性值与important关键字之间添加一个感叹号（“!”）。在感叹号周围增加若干个空格也是允许的。其他信息请参见第3章。

CSS语法 `!important`

值 此声明不须要赋值。

示例 `p {font-size: 14pt !important}`

应用于 所有元素。

对象模型引用方式

```
[window.]document.getElementById("elementID").style.getPropertyPriority("styleProperty")
```

layer-background-color, layer-background-image

IE *n/a* NN *|4|* Moz *n/a* Saf *n/a* Op *n/a* CSS *n/a*

通过继承得到：否

这个属性仅适用于Navigator 4，它允许将一个已定位元素的背景色或背景图延伸进填充区域，直到边框为止。其取值与CSS的background-color和background-image相同。请参见background-color、background-image及padding。

layout-flow

IE *5.5* NN *n/a* Moz *n/a* Saf *n/a* Op *n/a* CSS *n/a*

通过继承得到：是

主要在那些使用竖直排列的词句设置的语言中调用这个属性，它可以控制文本内容的前进方向，即从左到右还是从右到左。Microsoft建议使用writing-mode来取代这个属性。

CSS语法 layout-flow: horizontal | vertical-ideographic

值 两个常量值之一：horizontal | vertical-ideographic。

初始值 horizontal

示例 body {layout-flow: vertical-ideographic}

应用于 所有元素。

对象模型引用方式 [window.]document.getElementById("*elementID*").style.layoutFlow

layout-grid

IE *5 (Win)* NN *n/a* Moz *n/a* Saf *n/a* Op *n/a* CSS *n/a*

通过继承得到：是

通过这个属性可以在一条语句内快速地设定一个或多个布局网格属性，如layoutGridChar、layoutGridLine、layoutGridMode和layoutGridType。目前主要在亚洲语言内容中使用这些属性。

CSS语法 layout-grid: *layout-grid-mode* | *layout-grid-type* | *layout-grid-line* | *layout-grid-char*

值 请参见相关的属性说明，以便了解各种字体属性与行属性的语法及使用示例。

初始值 both loose none none

示例 body {layout-grid: both fixed 14px 14px}

应用于 所有元素。

对象模型引用方式 [window.]document.getElementById("*elementID*").style.layoutGrid

layout-grid-char

IE *5 (Win)* NN *n/a* Moz *n/a* Saf *n/a* Op *n/a* CSS *n/a*

通过继承得到：是

为块级元素指定亚洲语言字符网格的尺寸大小。

CSS语法 layout-grid-char: *length* | auto | none

值 以绝对长度单位度量的长度值，或者百分比值。也可以使用auto或none这两个常量之一。

初始值 none

示例 body {layout-grid-mode: both; layout-grid-char: 14px}

应用于 所有元素。

对象模型引用方式

[window.]document.getElementById("*elementID*").style.layoutGridChar

layout-grid-line

IE *5 (Win)* NN *n/a* Moz *n/a* Saf *n/a* Op *n/a* CSS *n/a*

通过继承得到：是

为块级元素指定亚洲语言字符网格的行高。

CSS语法 layout-grid-line: *length* | auto | none

值 以绝对长度单位度量的长度值，或者百分比值。也可以使用auto或none这两个常量之一。

初始值 none

示例 body {layout-grid-mode: both; layout-grid-line: 14px}

应用于 所有元素。

对象模型引用方式

[window.]document.getElementById("*elementID*").style.layoutGridLine

layout-grid-mode

IE *5 (Win)* NN *n/a* Moz *n/a* Saf *n/a* Op *n/a* CSS *n/a*

通过继承得到：是

指定亚洲语言字符网格的样式应该是一维还是两维。

CSS语法 layout-grid-mode: *gridMode*

值 以下常量之一：both | char（行内元素） | line（块级元素） | none。

初始值 both

示例 body {layout-grid-mode: both}

应用于 所有元素。

对象模型引用方式

[window.]document.getElementById("*elementID*").style.layoutGridMode

layout-grid-type

IE *5 (Win)* NN *n/a* Moz *n/a* Saf *n/a* Op *n/a* CSS *n/a*

通过继承得到：是

控制布局网格对字符宽度变化的响应形式。

CSS语法 layout-grid-type: *gridType*

值 以下常量之一：fixed | loose | strict。

初始值 fixed

示例 div.kor {layout-grid-type: strict}

应用于 块级元素。

对象模型引用方式

[window.]document.getElementById("*elementID*").style.layoutGridType

left

IE *4* NN *4* Moz *all* Saf *all* Op *all* CSS *2*

通过继承得到：否

对可定位的元素而言，它定义了元素的左边缘相对于临近最外层块级容器左边缘的偏移量。针对一个相对定

位元素，元素通常会出现在容器内容中的某个位置，会根据其内嵌位置的左边缘来决定偏移量。

CSS语法 `left: length | percentage | auto`

值 请参见本章中有关CSS长度值的讨论。在某些定位环境中还可以使用负值，但此时请在所有的浏览器中均进行充分测试，以保证其正确性。也可以使用百分比值作为其属性值，此时应该根据临近的最外层容器的宽度来计算该百分数。如果将它设置为auto，那么就会让浏览器根据正常的元素排列决定其容器盒，进而计算元素盒在容器盒中的左偏移量。

初始值 auto

示例

```
h1 {position: relative; left: 2em}
#logo {position: absolute; left: 80px; top: 30px}
```

应用于 已定位元素。

对象模型引用方式 `[window.]document.getElementById("elementID").style.left`

letter-spacing

IE *4* NN *n/a* Moz *all* Saf *all* Op *all* CSS *1*

通过继承得到：是

指明元素内的字符间距。浏览器通常会根据字体设置和操作系统的字体显示方式来决定字符间距。如果要替代这个设定，那么请为letter-spacing属性设置一个长度值。如果在此属性中使用负值，将缩小字符间距，此时一定要在不同的操作系统中对选中的字体进行充分测试，以保证调整间距后文本的可读性。

CSS语法 `letter-spacing: length | normal`

值 请参见本章中有关CSS长度值的讨论。根据现有的字体尺寸（如em或ex等）来决定度量单位通常可以得到最佳的显示效果。如果将此值设置为normal，则表示由浏览器来决定如何设置字符间距。

初始值 normal

示例

```
.tight {letter-spacing: -0.03em}
blockquote {letter-spacing: 1.1em}
```

应用于 所有元素。

对象模型引用方式

`[window.]document.getElementById("elementID").style.letterSpacing`

line-break

IE *5(Win)* NN *n/a* Moz *n/a* Saf *n/a* Op *n/a* CSS *n/a*

通过继承得到：是

控制日文文本中的换行规则。

CSS语法 `line-break: normal | strict`

值 以下常量之一：normal | strict。

初始值 normal

示例 `p {letter-break: strict}`

应用于 块级元素。

对象模型引用方式 `[window.]document.getElementById("elementID").style.lineBreak`

line-height

IE 4 NN 4 Moz all Saf all Op all CSS 1

通过继承得到：是

设置行内容器盒（即容纳一行实体内容的方框）的高度。在正常状态下，由行内文本中最高字体的line-height值或最高的对象来决定该行的高度。

CSS语法 line-height: normal | *number* | *length* | *percentage*

值 值为normal时，会让浏览器计算整个元素的行间距，因此会产生一个计算值并被其内嵌元素所继承。大于0的*number*值会让当前元素中的font-size设置效果倍增，因此，如果内嵌元素从其父级元素中继承了line-height乘数，那么该数也会作用于当前元素的font-size设定。注意，此时继承的是乘数而不是父级元素中的计算结果。*length*会为行容器盒指定一个实际的高度值，而*percentage*值则会将一个乘数应用于当前元素的字体尺寸。在这种情况下，内嵌元素也可以继承其计算值。

初始值 normal

示例

```
p {line-height: normal}      /* Browser default; actual value is  inheritable */
p {line-height: 1.1}         /* Number value; the number value is inheritable */
p {line-height: 1.1em}       /* Length value; the actual value is inheritable */
p {line-height: 110%}        /* Percentage value; percentage times font size */
                             /* is inheritable /*
```

应用于 所有元素。

对象模型引用方式 [window.]document.getElementById("*elementID*").style.lineHeight

list-style

IE 4 NN n/a Moz all Saf all Op all CSS 1

通过继承得到：是

通过这个属性可以在一条指令语句内快速设定3个列表样式属性。如果未在其中明确设定某个样式属性，那么就会使用其默认值来设定该属性。针对ol与ul元素中列表项前自动显示的标记符号，这几个列表样式属性定义了它们的显示特征。

CSS语法 list-style: *list-style-type* || *list-style-position* || *list-style-image*

值 关于这些属性所能够使用的属性值，请分别参见list-style-type、list-style-position和list-style-image属性。可以在这个list-style属性设定中以任意顺序包含1~3个值来设定对应的列表样式。

初始值 无

示例 ul {list-style: square outside none}

应用于 dd、dt、li、ol和ul元素，以及任意其他已指定了display: list-item样式属性的元素。

对象模型引用方式 [window.]document.getElementById("*elementID*").style.listStyle

list-style-image

IE 4 NN n/a Moz all Saf all Op all CSS 1

通过继承得到：是

提供一个图像的URL地址，并以该图像作为列表项之前的标记符号。由于这个属性具有继承性，因此为列表中的单独一项进行设置之后，它都会代替其父级元素对象的同名属性。

CSS语法 list-style-image: none | *uri*

值 对uri而言，请按照CSS格式提供一个图像文件的完整或相对URL地址，但要保证浏览器可以读取该文件的MIME类型。

初始值 none

示例

```
ul {list-style-image: url(images/folder.gif)}
li.file {list-style-image: url(images/doc.gif)}
```

应用于 dd、dt、li、ol和ul元素，以及任意其他已指定了display: list-item样式属性的元素。

对象模型引用方式

```
[window.]document.getElementById("elementID").style.listStyleImage
```

list-style-position

IE 4 NN n/a Moz all Saf all Op all CSS 1

通过继承得到：是

指定标记符号是位于列表项容器框之内还是之外。当list-style-position设置为inside并且列表项中的内容为文本时，标记符号会成为文本块中的一部分。在这种情况下，列表项的对齐（缩进）方式与正常情况完全一样，只是不存在伸出在外的标记符号。表4-5演示了这两种设置的具体效果。

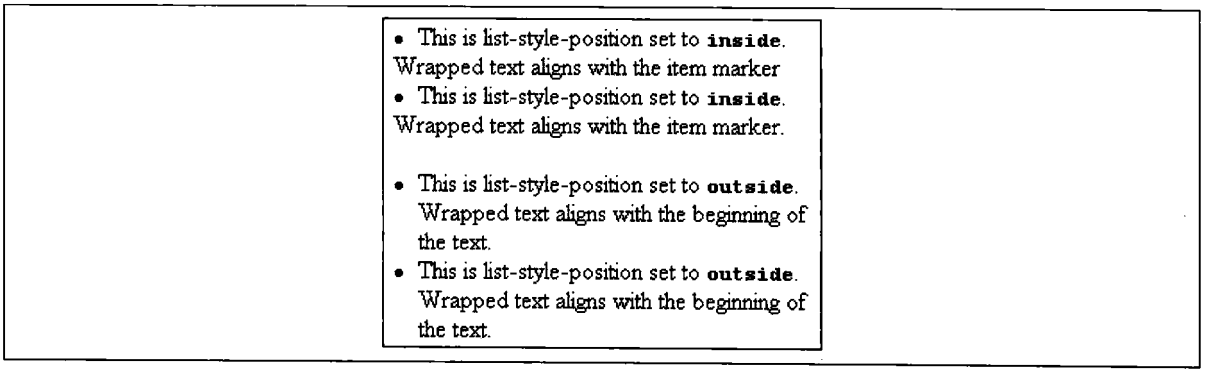


图 4-5：不同 list-style-position 设置的显示结果

CSS语法 list-style-position: inside | outside

值 可使用如下的常量值之一：inside | outside。

初始值 outside

示例 ul {list-style-position: inside}

应用于 dd、dt、li、ol和ul元素，以及任意其他已指定了display: list-item样式属性的元素。

对象模型引用方式

```
[window.]document.getElementById("elementID").style.listStylePosition
```

985

list-style-type

IE 4 NN 4 Moz all Saf all Op all CSS 1

通过继承得到：是

指定每一个列表项前显示的标记符号的具体类型。只有当list-style-image为none或未指定属性值时，才能应用这个属性。这个属性的可用常量值主要分为两大类。其中一类主要用于ul元素，它可以显示实心圆、空心圆或实心正方形。而另一类则用于ol元素，此时可以使用阿拉伯数字、罗马数字（大小写均可）或字母表中的字符（大小写均可）来标识列表项。当浏览器和操作系统还支持其他语言时，在序列中甚至还可以使用这些语言的字符。

CSS语法 `list-style-type: listStyleType`

值

可以使用下列与列表容器有关的常量值之一。ul元素：circle、disc、square。而在ol元素中则为：decimal | decimal-leading-zero | lower-roman | upper-roman | lower-greek | lower-alpha | lower-latin | upper-alpha | upper-latin | hebrew | armenian | georgian | cjk-ideographic | hiragana | katakana | hiragana-iroha | katakana-iroha。下表中列出了ol元素中的排序效果。

类型	示例	类型	示例
decimal	1, 2, 3, ...	lower-roman	i, ii, iii, ...
decimal-leading-zero	01, 02, 03, ...	upper-alpha	A, B, C, ...
lower-alpha	a, b, c, ...	upper-roman	I, II, III, ...
lower-greek	α, β, γ, ...		

986

初始值 disc（第1层的ul元素），decimal（ol元素）。

示例

```
ul {list-style-type: circle}
li {list-style-type: upper-roman}
```

应用于 dd、dt、li、ol和ul元素，以及任意其他已指定了display: list-item样式属性的元素。

对象模型引用方式

```
[window.]document.getElementById("elementID").style.listStyleType
```

margin

IE 4 NN 4 Moz all Saf all Op all CSS 1

通过继承得到：否

通过这个属性可以在一条指令语句中快速地为元素的4条边设定边距宽度。边距实际上就是元素边缘向外扩展的一段空间，从而在该元素和相邻或内嵌元素之间提供一块额外的空白区域。在这个属性值中，可以设定1~4个边距值，每个边距值间须要使用空格进行分隔。而样式值的数量就决定了由哪些边接受指定的样式。

CSS语法 `margin: marginThickness | auto {1,4}`

值

这个属性可接收由1~4个样式值，这决定了将要为哪些边距设定样式。*marginThickness*的值可以是长度、临近最外层容器尺寸的百分比或常量auto。下表解释了值的数量与位置所代表的实际含义。

值数量	效果
1	将4边的边距均设置为该值
2	顶部与底部的边距设置为第1个值，而左侧和右侧边距设定为第2个值。
3	顶部边距设置为第1个值，左侧和右侧边距设定为第2个值，而底部边距设置为第3个值
4	分别为顶部、右侧、底部和左侧边距设定对应值

初始值 0

示例 `p.highlight {margin: 10px 20px}`

应用于 所有元素。

987

对象模型引用方式 `[window.]document.getElementById("elementID").style.margin`

margin-bottom, margin-left, margin-right, margin-top

IE 4 NN 4 Moz all Saf all Op all CSS 1

通过继承得到：否

这4个属性设定了元素某条边的边距宽度。边距实际上就是沿着元素的边缘向外扩展的一块空间，但并不会将它视为元素高度或宽度的一部分。

CSS语法

```
margin-bottom: marginThickness | auto
margin-left: marginThickness | auto
margin-right: marginThickness | auto
margin-top: marginThickness | auto
```

值 marginThickness的值可以是长度、临近最外层容器尺寸的百分比或常量auto。

初始值 0

示例

```
blockquote {margin-left: 20; margin-top: 10}
#narrowCol {margin-left: 30%; margin-right: 30%}
```

应用于 所有元素。

对象模型引用方式

```
[window.]document.getElementById("elementID").style.marginBottom
[window.]document.getElementById("elementID").style.marginLeft
[window.]document.getElementById("elementID").style.marginRight
[window.]document.getElementById("elementID").style.marginTop
```

marker-offset

IE n/a NN n/a Moz n/a Saf n/a Op n/a CSS <2.1

通过继承得到：否

控制列表项标记符号与容纳文本内容的容器盒之间的距离。此时须要将列表项元素设置为显示marker样式。

CSS语法 marker-offset: *length* | auto

值 长度值（请参见本章前文中长度值的相关讨论），或者auto常量。

初始值 auto

示例 li:before {display: marker; marker-offset: 4em}

应用于 设置为marker显示模式的列表元素，通常与“:before”或“:after”伪类一同使用。

marks

IE n/a NN n/a Moz n/a Saf n/a Op n/a CSS 2

Inherited: n/a

这是一个页面语境属性，它指定了是否应该在页面内容区域之外显示裁剪标记或定位标记。使用时，只能在@page规则中设置这个属性。关于如何部署这种类型的样式规则的详细信息，请参见本章前文中的“规则”一节。

CSS语法 marks: *markType* | none

值 可用的*markType*值包括crop和cross这两个常量。crop标记指出了页面中的哪些位置应该被裁剪，而cross标记则主要用于对齐和定位。

初始值 none

示例 @page {marks: crop}

应用于 页面环境上下文。

对象模型引用方式 [window.]document.getElementById("*elementID*").style.marks

max-height, min-height

IE（见下文） NN *n/a* Moz *all* Saf *all* Op *all* CSS *2*

通过继承得到：否

这两个属性分别为元素盒确定了其最大高度和最小高度。使用这两个属性后，可以限制元素的高度而不必再考虑元素排列时的自然高度。

如果为某个元素设置了max-height属性，而该元素的内容却超出了最大值所能容纳的范围，那么就应该同时将overflow样式属性设置为hidden，这样裁剪下超出的内容。如果不这样做，溢出的内容就会渗入到随后的元素内容之中。设置了属性值后，无论对元素盒进行收缩还是拉伸，都必须根据元素内容的多少或最小高度来决定容器盒的尺寸大小。

IE 6仅支持min-height属性，而且只有当表格的table-layout样式属性设定为fixed时，才能在其中的td、th及tr元素中使用这个属性。这就和CSS2规范发生了冲突，该规范明确表示这些属性不应该影响与表格相关的任何元素。然而IE 7在CSS兼容模式下则完全正确地实现了这两个属性。此外，IE 5/Macintosh对这两个属性均不支持。

CSS语法

```
max-height: length | percentage | none
min-height: length | percentage | none
```

值 请参见本章中有关CSS长度值的讨论。也可以根据元素容器计算出百分比值。而当此值为none时，就会解除所有的限制，此时会让元素内容自然地排列。

初始值 max-height和min-height均为none。

应用于 见上文。

对象模型引用方式

```
[window.]document.getElementById("elementID").style.minHeight
[window.]document.getElementById("elementID").style.maxHeight
```

max-width, min-width

IE *7* NN *n/a* Moz *all* Saf *all* Op *all* CSS *2*

通过继承得到：否

这两个属性分别为元素盒确定了其最大宽度和最小宽度。使用这两个属性后，可以限制元素的宽度而不必再考虑元素在其父容器中排列时的自然宽度。IE 7必须允许CSS兼容模式才能使用这两个属性。

CSS语法

```
max-width: length | percentage | none
min-width: length | percentage | none
```

值 请参见本章中有关CSS长度值的讨论。也可以根据元素容器计算出百分比值。而当此值为none时，就会解除所有的限制，此时会让元素内容自然地排列。

初始值 max-width和min-width均为none。

应用于 所有元素。

对象模型引用方式

```
[window.]document.getElementById("elementID").style.minWidth
[window.]document.getElementById("elementID").style.maxWidth
```

-moz-border-radius-bottomleft,
-moz-border-radius-bottomright,
-moz-border-radius-topleft,
-moz-border-radius-topright

-moz-border-radius

IE *n/a* NN *n/a* Moz *all* Saf *n/a* Op *n/a* CSS *n/a*

通过继承得到：否

通过这个属性可以快速地为一个或多个边框设定转角半径，而颜色值的数量则决定了哪些边接受指定的颜色值。须要注意的是，这个值的约定与CSS3初稿中border-radius属性值的设置并不相同。

CSS语法 `-moz-border-radius: radius {1,4}`

值

可以使用长度值来定义边框转角半径，以表示转角所在的虚拟圆周的半径长度。此值越大，则转角越圆。在使用屏幕进行显示时，使用像素作为长度单位为宜。当然也可以使用一个百分数来表示圆弧，此时0%表示没有圆弧，而50%表示最大圆弧。边框圆弧并不会遮挡元素内容。

这个属性可接收由1~4个*radius*值，这决定了将要为哪些边距设定转角效果。下表解释了值的数量与位置所代表的实际含义。

值数量	效果
1	将4个转角均设置为同一个值
2	左上与右下的转角设置为第1个值，而右上和左下的转角则设定为第2个值
3	左上的转角设置为第1个值，右上和左下的转角设定为第2个值，而右下的转角则设置为第3个值
4	依次为左上、右上、右下和左下这4个转角设置的对应值

初始值 0

示例

```
div.hotbox {-moz-border-radius: 20px}
div.circle {-moz-border-radius: 50%}
```

应用于 所有元素。

-moz-border-radius-bottomleft, -moz-border-radius-bottomright, -moz-border-radius-topleft, -moz-border-radius-topright

IE *n/a* NN *n/a* Moz *all* Saf *n/a* Op *n/a* CSS *n/a*

通过继承得到：否

以上每个属性分别控制着一个边框转角的半径长度。须要注意的是，这个值的约定与CSS3初稿中转角特有的border-radius属性的值设置并不相同。

CSS语法

```
-moz-border-radius-bottomleft: radius
-moz-border-radius-bottomright: radius
-moz-border-radius-topleft: radius
-moz-border-radius-topright: radius
```

值 请参见-moz-border-radius。

初始值 0

示例 `div.bizarro {-moz-border-radius-topright:10%; -moz-border-radius-bottomright:10% }`

应用于 所有元素。

-moz-opacity

IE *n/a* NN *n/a* Moz *all* Saf *n/a* Op *n/a* CSS *n/a*

通过继承得到：否

定义元素不透明的程度。此属性值越小，则元素的透明度越高。这是Mozilla浏览器中的专有属性，但自从Mozilla 1.7.2开始，它已被CSS3的opacity属性所取代。

CSS语法 `-moz-opacity: alphaValue`

值 根据浮点数值的取值大小决定元素的不透明程度，该值的取值范围为从0.0到1.0。当值为1.0时则完全不透明。早期的Mozilla浏览器还支持百分比值，但目前已放弃了这种取值方式。

初始值 1

示例 `div#watermark {-moz-opacity: 0.4}`

应用于 所有元素。

对象模型引用方式 `[window.]document.getElementById("elementID").style.MozOpacity`

opacity

IE *n/a* NN *n/a* Moz *1.7.2* Saf *1.2* Op *9* CSS *3*

通过继承得到：否

定义元素不透明的程度。此属性值越小，则元素的透明度越高。尽管从技术上说这个属性并不存在继承性，但一个容器内所有子元素的透明度均受其父容器相关设定的控制。

CSS语法 `opacity: alphaValue`

值 根据浮点数值的取值大小决定元素的不透明程度，该值的取值范围为从0.0到1.0。当值为1.0时则完全不透明。

初始值 1

示例 `div#watermark {opacity: 0.35}`

应用于 所有元素。

对象模型引用方式 `[window.]document.getElementById("elementID").style.opacity`

orphans

IE *5 (Mac)* NN *n/a* Moz *n/a* Saf *n/a* Op *7* CSS *2*

通过继承得到：是

当页面中出现分页符时，这个属性设置了留在页面底部的段落的最小行数。出现分页符后在页面顶部显示的行数设置请参见widows属性。

CSS语法 `orphans: lineCount`

值 表示行数的整数值。

初始值 2

应用于 块级元素。

对象模型引用方式 `[window.]document.getElementById("elementID").style.orphans`

outline

IE *5 (Mac)* NN *n/a* Moz *1.8.1* Saf *1.2* Op *7* CSS *2*

过继承得到：否

在一条指令内，可以通过这个属性快速地设定元素外观轮廓上4条边的宽度、样式及颜色。如果未在其中明确设定某个样式属性，那么就会使用其默认值来设定该属性。

元素的外观轮廓与边框之间的差别主要表现在两个方面。首先，在CSS盒模型中，外观轮廓并不占据任何空间。它简单悬浮于元素上方，与边框所在的矩形重叠。其二，CSS的外观轮廓并不仅限于矩形，例如，可以沿着一个未对齐段落的外形来绘制外观轮廓。在IE 5/Macintosh中，仅能绘制矩形外观。在早期的Mozilla浏览器版本中，有一组以“-moz-”前缀开头的CSS2外观属性，如-moz-outline-color等。此外还有一些针对外观半径转角的专有属性，它们类似于-moz-border-radius属性。

CSS语法 `outline: border-color || border-style || outline-width`

993

值 请分别参见下文中的对应属性。

初始值 无。

示例 `blockquote {outline: darkred ridge 5px}`

应用于 所有元素。

对象模型引用方式 `[window.]document.getElementById("elementID").style.outline`

outline-color

IE *5 (Mac)* NN *n/a* Moz *1.8.1* Saf *1.2* Op *7* CSS *2*

过继承得到：否

控制元素四周外观轮廓的颜色。

CSS语法 `outline-color: color`

值 一个CSS颜色值。此时只用一个值就可以控制轮廓的所有边。CSS规范还提倡一个称为invert的常量，它会对背景色进行一种算法转换，但IE 5/Mac并不支持这个值。

初始值 在IE 5/Macintosh中为black。而CSS2规范建议将invert作为默认值。

示例

```
h2 {outline-color: salmon}
div {outline-color: rgb(0,0,255)}
```

应用于 所有元素。

对象模型引用方式 `[window.]document.getElementById("elementID").style.outlineColor`

outline-offset

IE *n/a* NN *n/a* Moz *1.8.1* Saf *1.2* Op *n/a* CSS *3*

过继承得到：否

控制外部轮廓与元素边框间的距离，在4个方向上的间距一般都完全相等。

CSS语法 `outline-offset: length`

值 一个CSS长度值。此时只用一个值就可以控制轮廓的所有边。

初始值 0

示例 `h2 {outline-offset: 3px}`

994

应用于 所有元素。

对象模型引用方式 `[window.]document.getElementById("elementID").style.outlineOffset`

outline-style

IE *5 (Mac)* NN *n/a* Moz *1.8* Saf *1.2* Op *7* CSS *2*

通过继承得到：否

控制元素四周外观轮廓的样式。这些边缘设计与边框样式设计基本相同。

CSS语法 `outline-style: borderStyle`

值 此时的样式值是一组常量，它们分别对应于一种特定的边框线条显示方式。常量列表与示例请参见border-style。此时只用一个值就可以控制轮廓的所有边。

初始值 none

示例

```
h2 {outline-style: solid}
div {outline-style: groove}
```

应用于 所有元素。

对象模型引用方式 [window.]document.getElementById("*elementID*").style.outlineStyle

outline-width

IE *5 (Mac)* NN *n/a* Moz *1.8.1* Saf *1.2* Op *7* CSS *2*

通过继承得到：否

控制元素四周外观轮廓的厚度。如果要防止四周内容显示在元素外观轮廓之下，那么就须要考虑在元素四周增加一定的边距。

CSS语法 outline-width: thin | medium | thick | *length*

值 目前有3个可用的常量值，thin、medium和thick。它们决定了浏览器使用多少像素来绘制外观轮廓。使用一个长度值可以更为精确地控制边框宽度，请参见本章中有关CSS长度值的讨论。此时只用一个值就可以控制轮廓的所有边。

初始值 medium

示例

```
h1 {outline-style: ridge; outline-width: 5px}
div {outline-style: solid; outline-width: 2px}
```

应用于 所有元素。

对象模型引用方式

```
[window.]document.getElementById("elementID").style.outlineWidth
```


overflow

IE *4* NN *n/a* Moz *all* Saf *all* Op *all* CSS *2*

通过继承得到：否

当元素内容所需的显示空间超出了特定容器的高度或宽度时，此属性定义了元素如何处理这些超出的内容。除了某些具有固定宽度的特殊内容类型（如pre元素）之外，在默认情况下，元素会遵循width属性设定而在元素的高度范围内处理溢出的问题。如果要追求跨浏览器的兼容性，那么为body元素指定overflow属性来控制滚动条的显示往往存在一定的风险。此时要在IE/Windows的“quirks”及标准兼容模式下彻底测试相关代码。

如果设置为visible，那么就会扩展容器块，以便完全显示元素内容的整个高度和宽度。此时如果已经为元素设定了填充、边距和边框，那么会在扩展的内容块周围保留这些内容。如果已经指定了元素的高度和宽度，以及背景图像或颜色，那么当元素内容超出了指定尺寸时具体的显示结果会随浏览器的不同而发生变化。当IE/Windows处于“quirks”模式下时，它会扩展背景的高度以便适应元素内容，此时会将后续内容下移，以便容纳溢出的内容。但过去的浏览器及处于标准兼容模式的IE 7浏览器，会强制保留背景矩形框的指定尺寸，此时，多出的内容会溢出到框外，而且会盖住该元素之后的其他内容。由于这是overflow样式属性的默认值，因此如果要限制元素的具体尺寸，最好指定其他的溢出值或修剪已定位元素的矩形块大小。

设置为hidden时会强制矩形框遵守其高度和宽度设定，这可能会导致因矩形块过小而截去部分元素内容。此时依然会保留边距和边框，但在截去元素的边会失去原有的填充空间。此外，这个值也不会出现滚动条。

设置为scroll后，无论是否需要，通常都会在当前块矩形框中产生水平方向，以及竖直方向的滚动条。但只有在的确须要进行滚动时才会激活对应方向上的滚动条。

设置为auto后，只有块中内容须要滚动时才会生成滚动条。实际上，当元素内容为文本时，浏览器倾向于仅使用竖直方向上的滚动条，以便让元素内容遵循其容器的特定宽度。

CSS语法 overflow: *overFlowType*

966 **值** 以下常量之一：auto | hidden | scroll | visible。

初始值 visible

示例

```
div.aside {position: absolute; top: 200px; left: 10px; height: 100px;
width: 150px; overflow: scroll}
```

应用于 块级元素、置换元素及已定位的元素。

对象模型引用方式 [window.]document.getElementById("*elementID*").style.overflow

overflow-x, overflow-y

IE *5* (Win) N *n/a* Moz *1.8* Saf *1.2* Op *all* CSS *n/a*

过继承得到：否

当元素内容所需的显示空间超出了容器的宽度（x）或高度（y）时，这两个属性定义了元素如何处理这些超出的内容。这两个属性的运行方式与常规的overflow属性完全一致，只不过它们分别只能操纵一个轴向而已。如果只期望在元素中出现一个水平或竖直滚动条，那么这两个属性就可以提供很多帮助。请参见overflow属性的相关讨论。

CSS语法

```
overflow-x: overFlowType
overflow-y: overFlowType
```

值 以下常量之一：auto | hidden | scroll | visible。

初始值 visible

示例

```
body {overflow-x: hidden; overflow-y: scroll}
```

应用于 块级元素、置换元素及已定位的元素。

对象模型引用方式

```
[window.]document.getElementById("elementID").style.overflowX
[window.]document.getElementById("elementID").style.overflowY
```

padding

IE *4* NN *4* Moz *all* Saf *all* Op *all* CSS *1*

通过继承得到：否

通过这个属性可以在一条指令语句中快速地为元素的4条边设定填充的宽度。填充实际上是围绕在元素内容容器盒四周并一直延伸到边框的一块空间区域，但它并不包括为元素指定的任何边框内容。填充区域会自动获取所在元素的背景图片或背景颜色。在向一个元素增加填充时，只会增加元素的矩形框大小而不会影响元素内容块的尺寸。在这个属性值中，可以设定1~4个填充值，每个填充值间则须要使用空格进行分隔，而样式值的数量就决定了由哪些边接受指定的样式。

997 **CSS语法** padding: *paddingThickness* {1,4}

值

这个属性可接收由1~4个样式值，这决定了将要为哪些边设定填充样式。*paddingThickness*的值既可以是长度值，也可以是临近最外层元素尺寸的百分比。下表解释了值的数量与位置所代表的实际含义。

值数量	效果
1	将4边填充均设置为该值
2	顶部与底部的填充设置为第1个值，而左侧和右侧填充设定为第2个值。
3	顶部填充设置为第1个值，左侧和右侧填充设定为第2个值，而底部填充设置为第3个值
4	依次分别设定顶部、右侧、底部和左侧填充的值

初始值 0，而IE/Windows为td和th元素特别指定的默认值为1。

示例
```
p.highlight {padding: 10px 20px}
```

应用于 IE 5/Macintosh、IE 5.5/Windows及W3C浏览器：所有元素；Windows系统中IE 5之前的浏览器：body、caption、div、iframe、marquee、table、td、textarea、tr及colgroup元素。但CSS2.1删除了对tr、thead、tbody、tfoot、col及colgroup这几个元素的支持。

对象模型引用方式
```
[window.]document.getElementById("elementID").style.padding
```

padding-bottom, padding-left, padding-right, padding-top

IE *4* NN *4* Moz *all* Saf *all* Op *all* CSS *1*

通过继承得到：否

这4个属性设定了元素某条边的填充宽度。填充实际上是围绕在元素内容容器盒四周并一直延伸到边框的一块空间区域，但它并不包括为元素指定的任何边框内容。填充区域会自动获取所在元素的背景图片或背景颜色。在向一个元素增加填充时，只会增加元素的矩形框大小而不会影响元素内容块的尺寸。

CSS语法
```
padding-bottom: paddingThickness
padding-left: paddingThickness
padding-right: paddingThickness
padding-top: paddingThickness
```

值 *paddingThickness*的值既可以是长度值，也可以是临近最外层容器元素尺寸的百分比。

初始值 0，而IE/Windows为td和th元素特别指定的默认值为1。

示例
```
blockquote {padding-left: 20; padding-top: 10}
#narrowCol {padding-left: 30%; padding-right: 30%}
```

应用于 IE 5/Macintosh、IE 5.5/Windows及W3C浏览器：所有元素；Windows系统中IE 5之前的浏览器：body、caption、div、iframe、marquee、table、td、textarea、tr及colgroup元素。但CSS2.1删除了对tr、thead、tbody、tfoot、col及colgroup这几个元素的支持。

对象模型引用方式
```
[window.]document.getElementById("elementID").style.paddingBottom
[window.]document.getElementById("elementID").style.paddingLeft
[window.]document.getElementById("elementID").style.paddingRight
[window.]document.getElementById("elementID").style.paddingTop
```

page

IE *5 (Mac)* NN *n/a* Moz *n/a* Saf *n/a* Op *n/a* CSS *2*

通过继承得到：否

通过一个指定给@page规则的标识符，此属性可以将一个块级元素与一个@page规则联系起来。

CSS语法
```
page: pageRuleIdentifier | auto
```

值 *pageRuleIdentifier*是赋予同一个文档中@page规则的一个名称。

初始值 auto

示例 table#results {page: printTable}

应用于 块级元素。

对象模型引用方式 [window.]document.getElementById("*elementID*").style.page

page-break-after, page-break-before

IE 4 NN n/a Moz 1.0.1 Saf all Op all CSS 2

通过继承得到：否

将文档发送至打印机时，这个属性定义了文档如何处理元素四周的分页符。和字处理程序中的效果类似，分页符也不会出现在可视化浏览器中，而连续的大量内容则会使屏幕中出现翻页和滚动效果。

此时主要依靠CSS2的“页面盒”概念来正确处理页面打印问题，这里的“页面盒”就是最终到达打印页面的矩形区域，而分页样式属性则辅助浏览器来控制每个页面盒的准确内容。如果没有任何外来辅助或设置为auto模式，那么浏览器在进行打印分页时会在每一页中尽可能多地填充打印内容，这与过去的处理方式相同。

如果要让分页符位于一个元素之上，那么请在该元素上使用“page-break-before: always”来设置样式。与之类似，如果要让分页符位于元素之后，则请使用“page-break-after: always”。例如，如果要让一种特殊类型的br元素在它们之后进行换行，那么就应该按照如下方式设置样式规则选择符：

```
<style type="text/css">
br.pageEnd {display: block; page-break-after: always}
</style>
```

此后，文档中无论何时需要一个页面分页，都可以引入如下的标签内容：

```
<br class="pageEnd">
```

属性设定为left和right时就假设浏览器可以从右向页面中发现左向页面，以进行双面打印（由CSS2指定）。由于可能会为每一侧的装订线设置不同的边距值，因此在指出页面分页如何开始一个新的分段时，须要提供足够的分页以便在目标页面上容纳新内容。例如，如果希望每个h1元素都起始于右向页，那么可以对其分页样式进行如下设置：

```
h1 {page-break-before: right}
```

这个属性会强制浏览器在h1元素前放置1~2个分页符，以保证该元素起始于一个右向页面。当浏览器为左值或右值生成第2个分页符时，这意味着浏览器会为第2个分页符生成一个空白的页面盒。

这两个属性的具体实现目前还是受到了些许限制。大多数现代浏览器支持always和auto，而Opera 9目前则支持right及left。

CSS语法

```
page-break-after: breakType
page-break-before: breakType
```

值 以下4个常量：always | auto | left | right（但可以将left和right与always视为一致）。CSS2增加了avoid，该值会要求浏览器尽量避免在对应元素上进行分页。

初始值 auto

示例 div.titlePage {page-break-before: always; page-break-after: always}

应用于 块级元素。

对象模型引用方式

```
[window.]document.getElementById("elementID").style.pageBreakAfter
[window.]document.getElementById("elementID").style.pageBreakBefore
```

page-break-inside

IE *n/a* NN *n/a* Moz *n/a* Saf *n/a* Op *7* CSS *2*

通过继承得到：是

此属性可以定义是否允许在元素内部出现页面分页。如果希望在同一个页面上完整打印包含多个块元素的容器，那么这个属性就非常有用。

CSS语法 page-break-inside: *breakType*

值 两个常量值之一：avoid | auto。

初始值 auto

示例 div.together {page-break-inside: avoid}

应用于 块级元素。

pause

IE *n/a* NN *n/a* Moz *n/a* Saf *n/a* Op *n/a* CSS *2*

通过继承得到：否

对于听觉样式单而言，通过这个属性可以在一个指令内快捷地设置pause-after和pause-before属性。可以为这个属性提供一个或两个值。

CSS语法 pause: time | percentage {1,2}

值

这个属性可以接收1个或2个字，这取决于要为pause-after和pause-before属性设定什么值。使用单个值的pause属性会将该值同时应用于pause-after和pause-before属性。而当提供2个值时，会将第1个值赋予pause-before，第2个则会指定给pause-after。

time的值是一个浮点数，其后还要跟计时单位标识符“ms”（毫秒）或“s”（秒）。因此，这些设定提供了暂停的绝对持续时间。*percentage*的值与speech-rate属性设定的每分钟单词数的倒数有关。由于speech-rate控制着每个单词所耗费的平均时间，因此pause设定为100%意味着暂停时间与朗读一个字所需的时间相等，而50%则表示一半的时间。

初始值 由浏览器决定。

应用于 所有元素。

pause-after, pause-before

IE *n/a* NN *n/a* Moz *n/a* Saf *n/a* Op *n/a* CSS *2*

通过继承得到：否

对于听觉类样式单而言，这两个属性可以分别设置在朗读当前元素之前或之后的暂停时间间隔。可以为同一个元素同时赋予这两个属性，那么在朗读该元素前后都会暂停一小段时间。

CSS语法

pause-after: *time* | *percentage*
pause-before: *time* | *percentage*

值 time的值是一个浮点数，其后还要跟计时单位标识符ms（毫秒）或s（秒）。这些设定提供了暂停的绝对持续时间。percentage的值与speech-rate属性设定的每分钟单词数的倒数有关。由于speech-rate控制着每个单词所耗费的平均时间，因此pause设定为100%意味着暂停时间与朗读一个字所需的时间相等，而50%则表示一半的时间。

初始值 由浏览器决定。

应用于 所有元素。

pitch

IE *n/a* NN *n/a* Moz *n/a* Saf *n/a* Op *n/a* CSS *2*

通过继承得到：否

对于听觉类样式单而言，这个属性设定了用于文本朗读发音的平均音调频率。

CSS语法 `pitch: frequency | frequencyConstant`

值 *frequency*可以是任意的正浮点数，其单位为Hz（赫兹）或kHz（千赫兹），如500Hz或5.5kHz。此外，也可以使用以下常量值：x-low、low、medium、high、x-high。但到目前为止，CSS2工作草案尚未为这些常量值指定特定的频率值。

初始值 medium

应用于 所有元素。

pitch-range

IE *n/a* NN *n/a* Moz *n/a* Saf *n/a* Op *n/a* CSS *2*

通过继承得到：否

1002 对于听觉类样式单而言，这个属性设定了用于文本朗读发音的音调频率变化范围。

CSS语法 `pitch-range: number`

值 任何大于或等于0的数值。值为0表示单音，50表示可以提供一个常用的音频范围，而大于50时，声音听起来就比较悦耳了。

初始值 50

应用于 所有元素。

play-during

IE *n/a* NN *n/a* Moz *n/a* Saf *n/a* Op *n/a* CSS *2*

通过继承得到：否

对于听觉类样式单而言，这个属性设定了发声机制中背景声音的混响效果。

CSS语法 `play-during: uri [mix | repeat] | auto | none`

值 uri可用于指明背景声音文件的链接地址。在实际使用中，可以为父元素的play-during属性指定背景声音，并与当前元素的背景音混合起来。如果背景音的长度比朗读元素内所有文本所需的时间短，那么就可以使用repeat常量来通知浏览器不断重复播放背景音，直到正文朗读完毕为止。auto表示浏览器会不断重复播放其父级元素的背景音。而none则意味着不会从当前元素或其父元素上听到任何背景音乐。

初始值 auto

应用于 所有元素。

position

IE *4* NN *4* Moz *all* Saf *all* Op *all* CSS *2*

通过继承得到：否

此属性可以设置元素是否是一个可定位元素，以及它的具体定位类型。目前有两种主要的可定位元素类型，即relative和absolute，而fixed这种类型只能在部分浏览器中得以使用。更多细节及使用范例请参见第5章。

CSS语法 `position: positionConstant`

值

1003 浏览器和CSS标准所能识别的各个常量值如下表所示：

值	IE/Windows	IE/Mac	NN	其他	值	IE/Windows	IE/Mac	NN	其他
absolute	4	4	4	all	relative	4	4	4	all
fixed	7	5	6	all	static	4	4	6	all

static实质上是一个非定位元素，它始终会处于正常页面流所指定的位置。而使用“fixed”的元素的具体位置则与窗口（视区）有关，即使页面已经滚动到其他位置，它也会保留在原地不动。

初始值 static

应用于 所有元素。

对象模型引用方式 [window.]document.getElementById("*elementID*").style.position

注意事项 根据CSS的position属性的不同设置，Navigator 4会按照处理layer元素方式的来处理绝对定位元素，而相对对位元素的处理方式则与ilayer元素相同。但浏览器在显示实际的元素与模拟实现时，最终结果始终还是存在着一定的差异，因此在Navigator 4中直接使用实际的layer及ilayer元素往往更为可靠。

quotes

IE *5 (Mac)* NN *n/a* Moz *all* Saf *n/a* Op *all* CSS *2*

通过继承得到：是

这个属性可以控制文本中生成的引号所用的具体字符类型。此时假设引号并不是具体内容的一部分，而是通过语境线索由浏览器生成出来的，例如使用q元素来生成一对引号。此外，这个属性必须与content属性共同使用，而“:before”及“:after”伪类则决定了开、闭引号的出现位置，代码如下：

```
q {quotes: "«" "»" "'" "'"}
q:before {content: open-quote}
q:after {content: close-quote}
```

CSS语法 quotes: *openString closeString* [*nestedOpenString nestedCloseString*] | none

值 一对或两对引号符号。第2对引号会作为内嵌引号进行使用。此时不允许使用其他实体字符。

初始值 由浏览器及系统语言决定。

应用于 所有元素。

对象模型引用方式 [window.]document.getElementById("*elementID*").style.quotes

注意事项 某些浏览器对这个属性的支持并不是特别好。例如，IE 5/Macintosh不会真正地响应quotes属性，而只会为content属性替换标准的两层引号。Mozilla则只能采用第1层的引号。此外，在文本编辑器中，ASCII字符集之外的其他字符符号可能无法与源代码中的字符融合在一起。鉴于以上原因，在部署这个属性前请进行测试。

richness

IE *n/a* NN *n/a* Moz *n/a* Saf *n/a* Op *n/a* CSS *2*

通过继承得到：是

对于听觉类样式单而言，这个属性设定了用于文本朗读发音的音色。

CSS语法 richness: *number*

值 使用一个正浮点数来表示发音的尖锐程度。50是正常值。此值越小，则声音越柔软、圆润，而随着值的增加，声音也会变得更为高亢、尖锐。

初始值 50

应用于 所有元素。

right

IE *5* NN *n/a* Moz *all* Saf *all* Op *all* CSS *2*

通过继承得到：否

对可定位的元素而言，它定义了元素的右边缘相对于临近最外层块级容器的右边缘的偏移量。针对一个相对

定位元素，元素通常会出现在容器内容中的某个位置，那么就根据内嵌位置的右边缘来决定偏移量。

CSS语法 right: *length* | *percentage* | auto

值 请参见本章中有关CSS长度值的讨论。在某些定位环境中还可以使用负值，但此时请在所有的浏览器中进行测试，以保证其正确性。也可以使用百分比值作为其属性值，此时应该根据临近的最外层容器的宽度来计算该百分数。须要注意的是，最终的设置结果可能正好跟预期的相反，例如，0%将意味着右边缘紧贴着定位内容的右边缘，而100%却会将元素向左整个推出视图。如果将此属性设置为auto，那么就会让浏览器自主决定元素盒的右侧偏移量在其容器盒内的偏移值。

初始值 auto

应用于 已定位元素。

对象模型引用方式 [window.]document.getElementById("*elementID*").style.right

1005

ruby-align

IE *5* NN *n/a* Moz *n/a* Saf *n/a* Op *n/a* CSS *3*

通过继承得到：否

此属性可以控制ruby元素中相关内容的对齐方式。

CSS语法 ruby-align: *alignType* | auto

值 以下常量之一：auto | center | distribute-letter | distribute-space | left | line-edge | right。更多相关信息，请访问*http://www.w3.org/tr/css3-ruby*。

初始值 auto

应用于 在IE中，只能在ruby元素中使用这个样式，但CSS 3规范初稿则建议在所有包含ruby文本的元素中使用此样式。

对象模型引用方式 [window.]document.getElementById("*elementID*").style.rubyAlign

ruby-overhang

IE *5* NN *n/a* Moz *n/a* Saf *n/a* Op *n/a* CSS *3*

通过继承得到：是

此属性可以控制ruby元素中相关内容的文字突出方式。

CSS语法 ruby-overhang: *alignType* | auto

值 以下常量之一：auto | none | whitespace。更多相关信息，请访问*http://www.w3.org/tr/css3-ruby*。

初始值 auto

应用于 ruby元素，或者已将display属性设置为ruby-text的其他任意元素。

对象模型引用方式 [window.]document.getElementById("*elementID*").style.rubyOverhang

1006

ruby-position

IE *5* NN *n/a* Moz *n/a* Saf *n/a* Op *n/a* CSS *3*

通过继承得到：是

此属性控制着内嵌rt元素中的文本是应该与相关的基础文本显示在同一行中还是位于这些文本之上。

CSS语法 ruby-position: *positionType*

值 IE可以识别above和inline这两个常量，而CSS3规范初稿则倾向于使用以下常量：after | before | inline | right。更多相关信息，请访问*http://www.w3.org/tr/css3-ruby*。

初始值 above（IE）；before（CSS3）。

应用于 ruby元素，或者已将display属性设置为ruby-text的其他任意元素。

对象模型引用方式 [window.]document.getElementById("*elementID*").style.rubyPosition

scrollbar-3dlight-color, scrollbar-arrow-color, scrollbar-base-color, scrollbar-darkshadow-color, scrollbar-face-color, scrollbar-highlight-color, scrollbar-shadow-color, scrollbar-track-color

IE *5.5* NN *n/a* Moz *n/a* Saf *n/a* Op *n/a* CSS *n/a*

通过继承得到：否

它们分别控制着不同滚动条用户界面元素中特定组件的颜色。下表列出了每个属性所控制的滚动条部位。

属性	描述
scrollbar-3dlight-color	滚动滑块的顶部和左侧边缘，以及箭头按钮框
scrollbar-arrow-color	箭头按钮框中的箭头
scrollbar-base-color	滚动条的整体色调
scrollbar-darkshadow-color	滚动滑块的底部和右侧边缘，以及箭头按钮框
scrollbar-face-color	滚动条表面以及滑道
scrollbar-highlight-color	用于创建3D效果的白色像素，以及滑道的替换像素
scrollbar-shadow-color	滚动条3D界面中暗边的颜色
scrollbar-track-color	整个滑道的颜色

如果有兴趣，可根据上表所述，将滚动条的多个部件及颜色组合起来进行实验。

CSS语法

```
scrollbar-3dlight-color: color
scrollbar-arrow-color: color
scrollbar-base-color: color
scrollbar-darkshadow-color: color
scrollbar-face-color: color
scrollbar-highlight-color: color
scrollbar-shadow-color: color
scrollbar-track-color: color
```

值 CSS颜色值。

初始值 由用户的显示控制面板设定决定。

示例 `textarea {scrollbar-face-color: lightyellow}`

应用于 applet、bdo、body、custom、div、embed、object，以及textarea元素。

对象模型引用方式

```
[window.]document.getElementById("elementID").style.scrollbar3dLightColor
[window.]document.getElementById("elementID").style.scrollbarArrowColor
[window.]document.getElementById("elementID").style.scrollbarBaseColor
[window.]document.getElementById("elementID").style.scrollbarDarkShadowColor
[window.]document.getElementById("elementID").style.scrollbarFaceColor
[window.]document.getElementById("elementID").style.scrollbarHighlightColor
[window.]document.getElementById("elementID").style.scrollbarShadowColor
[window.]document.getElementById("elementID").style.scrollbarTrackColor
```

size

IE *n/a* NN *n/a* Moz *n/a* Saf *n/a* Op *all* CSS *2*

通过继承得到：n/a

设置一个页面盒的尺寸及打印方向。这个属性主要用于对打印页面进行排版，但它对屏幕显示的内容或裁剪内容可能并无任何影响。使用时，只能在@page规则中设置这个属性。

CSS语法 size: [*length* {1,2}] auto | portrait | landscape

值 如果指定了一个或两个length值，那么无论纸张的具体大小，该页面盒将拥有绝对尺寸。但如果未设置length，那么页面盒的尺寸就与所选择的纸张大小有关。如果只设定了一个长度值，就会将它同时应用于页面盒的高度和宽度，如果使用了两个值，那么前者用于控制盒宽，而后者则控制盒高。须要牢记于心的是，打印机常常会在已显示的页面盒四周加上少量的边距空间。将此值设置为auto后，通过在@page声明中加入margin属性，还可以在页面盒周围增加更多的空间。

1008

初始值 auto

示例 @page{size: landscape}

应用于 页面环境上下文。

speak

IE *n/a* NN *n/a* Moz *n/a* Saf *n/a* Op *n/a* CSS *2*

通过继承得到：是

对于听觉类样式单而言，这个属性指定了浏览器的文本-语音转换设备是否应该朗读元素内容，以及在朗读时是应该读出整个单词还是逐个读出字符。

CSS语法 speak: *speechType*

值 有3种可用的常量值：none | normal | spell-out。none意味着关闭声音。normal则表示打开声音并按文本中的单词进行朗读。而spell-out则表示读出元素内容的每个字符，这在abbr和acronym元素中比较有用。浏览器在朗读时不会出现延迟或暂停，请参见volume: silent属性值。

初始值 normal

应用于 所有元素。

speak-header

IE *n/a* NN *n/a* Moz *n/a* Saf *n/a* Op *n/a* CSS *2*

通过继承得到：是

对于具有文本-语音转换功能的浏览器而言，可以读出每个单元格中的具体内容，而这个属性则指定了浏览器是仅朗读一次单元格头部内容，还是在每次朗读单元格内容前均读出其头部内容。

CSS语法 speak-header: *headerFrequency*

值 有两个可用的常量值：once | always。

初始值 once

应用于 th元素。

speak-numeral

IE *n/a* NN *n/a* Moz *n/a* Saf *n/a* Op *n/a* CSS *2*

1009

通过继承得到：是

对于听觉样式单而言，这个属性设置了在朗读数值时是读出单个的数字（如，一四二），还是将它们作为整体（如，一百四十二）。朗读数字时所用的语言由元素的lang属性值决定。

CSS语法 speak-numeral: *numeralType*

值 有两个可用的常量值：digits | continuous。

初始值 continuous

应用于 所有元素。

speak-punctuation

IE *n/a* NN *n/a* Moz *n/a* Saf *n/a* Op *n/a* CSS *2*

通过继承得到：是

对于听觉样式单而言，这个属性决定了遇到标点符号时是将其读出（如，“逗号”），还是根据不同的标点产生一个自然的停顿。

CSS语法 speak-punctuation: *punctuationType*

值 有两个可用的常量值：code、none。code表示遇到标点时应该读出该标点的名称。

初始值 none

应用于 所有元素。

speech-rate

IE *n/a* NN *n/a* Moz *n/a* Saf *n/a* Op *n/a* CSS *2*

通过继承得到：是

对于听觉类样式单而言，这个属性设定了文本朗读时每分钟读出的单词数。

CSS语法 speech-rate: *wordsPerSecond* | *speedConstant*

值

*wordsPerSecond*可以是任意一个正浮点数，并且不需要单位。也可以使用以下常量作为属性值：

值	含义	值	含义
x-slow	80 字/分钟	x-fast	500 字/分钟
slow	120 字/分钟	slower	每分钟减少 40 字
medium	180-200 字/分钟	faster	每分钟增加 40 字
fast	300 字/分钟		

初始值 medium

应用于 所有元素。

stress

IE *n/a* NN *n/a* Moz *n/a* Saf *n/a* Op *n/a* CSS *2*

通过继承得到：是

对于听觉类样式单而言，这个属性设定了发音中的重音（声调变化）。

CSS语法 stress: *stressLevel*

值 *stressLevel*可以是任意一个正浮点数，并且不需要单位。50是正常值。

初始值 50

应用于 所有元素。

table-layout

IE *5* (Win) NN *n/a* Moz *1.0.1* Saf *all* Op *7* CSS *2*

通过继承得到：否

此属性决定了浏览器是通过计算得到整个表格数据的高度和宽度再开始绘制表格，还是依照table元素的尺寸属性，并使用第1行元素的宽度来绘制表格内容。当此属性设置为auto时，浏览器必须加载所有的表格单元及内容后才能绘制出第1行数据，这会给表格绘制带来一定的滞后。而设置为fixed时，则会立即开始表格绘制，这对绘制大型表格大有裨益。如果后续行中的内容比固定尺寸宽，那么除非将overflow样式属性设置为visible，否则就会对超出的内容进行裁剪，但前一种做法可能会导致后续的单元格内容出现混乱。

CSS语法 table-layout: *layoutType*

值 有两个可用的常量值：auto | fixed。

初始值 auto

应用于 table元素。

text-align

IE *4* NN *4* Moz *all* Saf *all* Op *all* CSS *1*

1011

通过继承得到：是

决定元素中所有文本的水平对齐方式。由于这个属性从父级元素中继承而来，因此可以在一个容器中进行设置，从而使它作用于所有的内嵌元素，例如div元素中的p元素。目前各浏览器均可支持center、left及right这3个值。此外，虽然justify并不是CSS的规定值，但它仍然可以在IE 5和后续版本的IE浏览器及其他主流浏览器中正常工作。

CSS语法 text-align: *alignment*

值 以下4个常量字符串之一：center | justify | left | right。

初始值 由浏览器决定。

示例

```
p.rightHand {text-align: right}
blockquote {text-align: justify}
```

应用于 块级元素，但在IE 5+/Windows、Mozilla、Safari和Opera中，右对齐也可以在文本类型的input及textarea元素中使用。

对象模型引用方式 [window.]document.getElementById("*elementID*").style.textAlign

text-align-last

IE *5.5* NN *n/a* Moz *n/a* Saf *n/a* Op *n/a* CSS *n/a*

通过继承得到：是

决定了元素盒中最后一行文本的水平对齐方式。

CSS语法 text-align-last: *alignment*

值 以下常量之一：auto | center | justify | left | right。auto会获得元素继承的text-align属性值。

初始值 auto

示例 blockquote {text-align-last: center}

应用于 块级元素。

对象模型引用方式 [window.]document.getElementById("*elementID*").style.textAlignLast

text-autospace

IE *5(Win)* NN *n/a* Moz *n/a* Saf *n/a* Op *n/a* CSS *n/a*

1012

通过继承得到：否

控制表意（象形）字符和非表意字符之间的间隔，常用于亚洲语言。

CSS语法 text-autospace: *spacingType*

值 以下常量之一：ideograph-alpha | ideograph-numeric | ideograph-parenthesis | ideograph-space | none。

初始值 none

示例 div {text-autospace: ideograph-numeric}

应用于 所有元素。

对象模型引用方式 [window.]document.getElementById("*elementID*").style.textAutospace

text-decoration

IE 4 NN 4 Moz all Saf all Op all CSS 1

通过继承得到：否

为元素文本内容指定其他的附加修饰样式，如下划线、删除线、上划线，以及闪烁（可用于Navigator及CSS）。在这个属性值中可以设定多个样式值，每个不同值之间则须要使用空格进行分隔。比较幸运的是，主流浏览器会忽略blink设定。而Navigator 4则无法识别overline。

这种文本修饰拥有不同寻常的父-子关系。尽管并不存在属性值继承，但在大多数情况下，文本的修饰效果仍会传递给元素的内嵌项。因此，除非设定了其他替代样式，否则使用下划线的p元素会为内嵌的span元素的内容也加上下划线。

CSS语法 text-decoration: *decorationStyle* | none

值 除none外，可以使用如下4个常量之一：blink | line-through | overline | underline，但Mozilla浏览器会忽略blink。

初始值 none

示例 div.highlight {text-decoration: underline}

应用于 所有元素。

对象模型引用方式

[window.]document.getElementById("*elementID*").style.textDecoration
[window.]document.getElementById("*elementID*").style.textDecorationBlink
[window.]document.getElementById("*elementID*").style.textDecorationLineThrough
[window.]document.getElementById("*elementID*").style.textDecorationNone
[window.]document.getElementById("*elementID*").style.textDecorationOverLine
[window.]document.getElementById("*elementID*").style.textDecorationUnderline

text-indent

IE 4 NN 4 Moz all Saf all Op all CSS 1

通过继承得到：是

此属性指定了文本块中第1行的缩进量。它也只能对第1行产生影响。使用负值则会突出第1行，此时要保证突出的文本不会超出浏览器窗口或框架的左边缘。

CSS语法 text-indent: *length* | *percentage*

值 请参见本章中有关CSS长度值的讨论。在某些定位环境中还可以使用负值，但此时请在所有的浏览器中进行测试，以保证其正确性。也可以使用百分比值作为其属性值，此时应该根据临近的最外层容器的宽度来计算该百分数。

初始值 0

示例

```
body {text-indent: 2em}
p.firstGraphs {text-indent: 0}
```

应用于 块级元素。

对象模型引用方式 [window.]document.getElementById("*elementID*").style.textIndent

text-justify

IE *5* NN *n/a* Moz *n/a* Saf *n/a* Op *n/a* CSS *n/a*

通过继承得到：是

如果将一个块级元素的CSS属性text-align设置为justify，那么这个属性可以指定字符分散对齐的细节。这个属性主要是为亚洲语言或其他非拉丁系语言而设计的。

CSS语法 `text-justify: justificationType`

值

下表中的任意一个常量。

值	含义
auto	使浏览器自动选择最优类型
distribute	与 newspaper 类似，但它已为亚洲语言进行了一定的优化
distribute-all-lines	对齐各行，其中包括最后一行，这可能会导致出现非常宽的单词拼写
distribute-center-last	对齐各行，并将最后一行居中
inter-cluster	对齐缺少词间距的所有行
inter-ideograph	对齐包含象形文字的所有行
inter-word	通过增加词间的空白距离来对齐各行，通常用于拉丁系语言中
kashida	通过拉长字符笔画对齐阿拉伯文本，只能用于 IE 5.5 及后续版本
newspaper	通过增加词间与字符间的空白距离来对齐各行

1014

初始值 0

示例 `div#col1 {text-align: justify; text-justify: newspaper}`

应用于 块级元素。

对象模型引用方式 `[window.]document.getElementById("elementID").style.textJustify`

text-kashida-space

IE *5.5* NN *n/a* Moz *n/a* Saf *n/a* Op *n/a* CSS *n/a*

通过继承得到：是

在字符对齐样式为justify的块级元素中，如果包含阿拉伯文字，那么这个属性可以控制卡须达（kashida）在空白空间中的比例。

CSS语法 `text-kashida-space: length | percentage`

值 请参见本章中有关CSS长度值的讨论。也可以使用百分比值作为其属性值，此时应该根据临近的最外层容器的宽度来计算该百分数。

初始值 0%

示例 `div#col1 {text-align: justify; text-justify: newspaper; text-kashida-space: 5%}`

应用于 块级元素。

对象模型引用方式

`[window.]document.getElementById("elementID").style.textKashidaSpace`

text-overflow

IE *6* NN *n/a* Moz *n/a* Saf *1.3/2* Op (见下文) CSS *n/a*

通过继承得到：否

当文本内容溢出了固定的容器盒时，这个属性可以指定是否在行尾显示一个省略符号（“...”），以表示还有更多的文本信息。使用这个功能时还须要将元素的overflow样式属性设置为hidden。此外，Opera 9还实现了一个与之对等的专有版本，即“-o-text-overflow”。

1015

CSS语法 text-overflow: *overflowType*

值 两个常量值之一：clip | ellipsis。

初始值 clip

示例 td {overflow: hidden; white-space: nowrap; text-overflow: ellipsis}

应用于 块级元素。

对象模型引用方式 [window.]document.getElementById("*elementID*").style.textOverflow

text-shadow

IE *n/a* NN *n/a* Moz *n/a* Saf *1.2* Op *n/a* CSS *<2.1*

通过继承得到：否

为当前元素中的文本设置阴影效果。一个文本元素可以拥有多个阴影，而且每个阴影还可以拥有其独有的颜色、竖直偏移、水平偏移，以及模糊半径，但每个阴影只会存在于其自身的细分层中，而且将第1个阴影规范放置在整个阴影堆栈的底部。针对每个阴影的赋值之间使用空格进行分隔，而在多个阴影的值分组间则采用逗号进行分隔。

CSS语法

text-shadow: [*color*] *horizLength vertLength blurRadiusLength*,
[[*color*] *horizLength vertLength blurRadiusLength*] | none

值 如果忽略了*color*属性值，那么阴影就会直接采用元素的*color*属性值，该值既可能来自于自身也可能继承自其父元素。而*color*属性的放置位置则没有什么要求，放在阴影长度值的前后均可。请参见本章中有关CSS颜色值的讨论。*horizLength*与*vertLength*为长度值（请参见本章开头部分），而且它们的正负号还指明了阴影从元素文本开始的偏移方向。对*horizLength*而言，正值会将阴影放置在元素左侧，负值则对应于右侧。而对*vertLength*而言，正值会将阴影放置在文本下方，负值则与之相反。模糊半径也是一个长度值，它指定了从文本字符边缘开始出现的阴影范围。

初始值 none

应用于 所有元素。

text-transform

IE *4* NN *4* Moz *all* Saf *all* Op *all* CSS *1*

通过继承得到：是

控制元素文本的大小写转换。如果这个属性的值不为none，那么样式单会重新整理源代码中的所有字母，此时可能会改变源文本字符的大小写形式。

CSS语法 text-transform: *caseType* | none

值 值为none时会按照源文本中的原有形式显示文本。其他可用的常量值为capitalize、lowercase、uppercase。capitalize会将每个单词的首字母转变为大写形式，而lowercase和uppercase则使用小写或大写形式显示元素中的文本字符。

初始值 none

示例 h2 {text-transform: capitalize}

应用于 所有元素。

对象模型引用方式 [window.]document.getElementById("*elementID*").style.textTransform

text-underline-position

IE *5.5* NN *n/a* Moz *n/a* Saf *n/a* Op *n/a* CSS *n/a*

通过继承得到：否

如果一个元素的text-decoration样式属性为underline，那么这个属性可以控制下划线是出现在文本上方还是文本下方。此属性主要用于显示在竖直列中的亚洲语言。

CSS语法 text-underline-position: *positionType* | none

值 IE 5.5可以识别两个常量值：above | below。而IE 6则增加了另外两个值，auto和auto-pos，不过它们的效果看起来似乎完全一样。不同版本的默认值也不相同，IE 5.5为below，而IE 6则为auto。另外，在竖排的日文文本中，auto属性值会让IE 6在字符的右侧显示竖线。

初始值 none

示例 h2 {text-underline-position: above}

应用于 所有元素。

对象模型引用方式

[window.]document.getElementById("*elementID*").style.textUnderlinePosition

top

IE *4* NN *4* Moz *all* Saf *all* Op *all* CSS *2*

通过继承得到：否

对已定位元素而言，它定义了元素的顶部边距边缘相对于临近最外层块级容器上边缘的偏移量。

CSS语法 top: *length* | *percentage* | auto

值 请参见本章中有关CSS长度值的讨论。在某些定位环境中还可以使用负值，但此时请在所有的浏览器中进行测试，以保证其正确性。也可以使用百分比值作为其属性值，此时应该根据临近的最外层容器的高度来计算具体的百分比。如果将此属性设置为auto，那么就会让浏览器自主决定元素盒的顶部偏移量在其容器盒内的偏移值。

初始值 auto

示例

```
h1 {position: relative; top:2em}
#logo {position: absolute; left:80px; top:30px}
```

应用于 已定位元素。

对象模型引用方式 [window.]document.getElementById("*elementID*").style.top

unicode-bid

IE *5* NN *n/a* Moz *all* Saf *all* Op *8* CSS *2*

通过继承得到：否

这个属性与direction样式属性一起控制着内嵌的双向文本，如德语与阿拉伯语混写的文本。

CSS语法 unicode-bidi: *embeddingType*

值 可使用如下的常量值之一：bidi-override | embed | normal。

初始值 normal

示例 div.multiLingual {unicode-bidi: embed}

应用于 所有元素。

对象模型引用方式 [window.]document.getElementById("*elementID*").style.unicodeBidi

vertical-align

IE 4 NN n/a Moz all Saf 1.2 Op all CSS 1 1018

通过继承得到：否

针对这个属性目前存在两类值，而且它们对于行内元素的影响效果也各不相同。其着眼点主要在于，每个行内元素均拥有一个行框以容纳其内容。top和bottom这两个值会影响线框内文本显示情况，它们可以让文本分别填满线框的顶部或底部。

此外，可将此属性用于由内嵌的span元素控制的文本，而图像和表格也可以使用这个样式属性。整个元素框相对于父级元素内容在竖直方向上的定位则会受到所有其他vertical-align设置的影响。其默认值baseline意味着行框已被定位，这样一来，行框中的文本（或元素的最底端）与父级元素中的文本所在的基线就会完全齐平。正因为如此，em元素就可以成为其自身的行框元素，而且与包含它的p元素排列在相同的基线上。属性的剩余常量值决定了如何根据其父级行对元素的行框进行定位。正的百分比值或长度值会将元素放置在基线以上的指定位置，而负值则会让元素位于基线之下。此时是根据行的高度来计算百分比值的。

CSS语法 `vertical-align: vertAlignType | length | percentage`

值

对元素自身内部的文本进行对齐的可用常量值有两个，即bottom和top。

而相对于四周文本行进行元素行框对齐的6个常量为：baseline | middle | sub | super | text-bottom | text-top。baseline会让元素与其父级元素的基线保持齐平。Middle会使元素的竖直中点位于基线上方，高度为其父级元素字体的x高度的一半。sub和super分别会为上标和下标改变元素的位置，但它们并不会创建真正的上标或下标内容，因此也不会调整字体尺寸。text-bottom会让元素底部对齐父级元素文本底部，而text-top则会让它们的顶部对齐。

初始值 baseline

示例 `span.sup {vertical-align: super; font-size: smaller}`

应用于 仅行内元素。

对象模型引用方式 `[window.]document.getElementById("elementID").style.verticalAlign`

visibility

IE 4 NN 4 Moz all Saf all Op all CSS 2 1019

通过继承得到：是

此属性可以控制是否在页面上绘制对应元素。使用visibility属性来隐藏一个元素后，会在文档中保留该元素原来所占据的空间位置。如果要让周围的内容进入隐藏元素的原有空间，那么请参见display属性。将此属性应用于表格行相关的元素时，CSS规范建议collapse值应该缩紧表格，但主流浏览器均未这样执行，例如，Mozilla 1.8、Safari 2和Opera 9只会简单地隐藏对应行，并留下空白的行空间。

如果将它设置为inherit，那么应该可以继承visibility属性。这意味一旦隐藏父元素，也会将其子元素一并隐藏起来。但如果将子元素的visibility属性设置为visible，那么在上述情况下仍然可以保持子元素的可见性。

CSS语法 `visibility: visibilityType`

值 以下常量值之一：collapse | hidden | inherit | visible。请注意，IE/Windows无法识别collapse，而Navigator 4只允许控制已定位元素的可见性。

初始值 visible

示例 `#congrats {visibility: hidden}`

应用于 所有元素。

对象模型引用方式 `[window.]document.getElementById("elementID").style.visibility`

voice-family

IE *n/a* NN *n/a* Moz *n/a* Saf *n/a* Op *n/a* CSS *2*

通过继承得到：否

对于听觉样式单而言，听觉浏览器应该尽量使用这个属性设置的声音簇名称来朗读页面内容。允许使用多个由逗号分隔的值来为属性赋值。这个属性的相关功能与视觉浏览器中的font-family比较类似。

CSS语法 `voice-family: voiceFamilyName [, voiceFamilyName [, ...]]`

值 *voiceFamilyName*就是一个针对某个通用声音名称或听觉浏览器所提供的声音类型的标识符。与font-family设置相同，应该根据需要指定多个声音类型，而且类型列表应从特有的声音开始，并以最通用的类型结尾。

初始值 由浏览器决定。

应用于 所有元素。

volume

IE *n/a* NN *n/a* Moz *n/a* Saf *n/a* Op *n/a* CSS *2*

通过继承得到：否

对于听觉类样式单而言，这个属性设定了发音元素的动态范围（柔和度及音高）。由于正常发音应包含变调，因此不能在所有时间都只使用一个绝对的音量。通常会将此属性设置为中等大小的音量。

CSS语法 `volume: number | percentage | volumeConstant`

值 *number*可以是任意一个数字。number为0时表示在设备噪声与周围噪声环境中可以听到的最小声音等级，而100则应该表示在同等环境下能够保持声音悦耳的最高等级。*percentage*值的计算与父级元素的volume属性设定有关。其音量设定的等效常量及其对应的数值如下：silent (no sound) | x-soft (0) | soft (25) | medium (50) | loud (75) | x-loud (100)。

初始值 medium

应用于 所有元素。

white-space

IE *5(Mac)/5.5 (Win)* NN *4* Moz *all* Saf *all* Op *all* CSS *1*

通过继承得到：是

此属性可以设定浏览器如何显示元素源代码中的空白内容，如额外的字符空格及回车等。在正常情况下，HTML会忽略额外的空白，并且让周围的内容占据它们原来所占据的空间。例如，在单词之间只保留一个空格，而且在段落中必须使用br元素进行强制换行。如果将此属性设置为pre，那么就会像在pre元素中一样处理空白。即便如此，尽管浏览器会使用等宽字体显示pre元素，但将普通元素设置为“white-space: pre”时，仍然会保留其原有的字体特征。

CSS语法 `white-space: whiteSpaceType`

值 以下5个常量值之一：normal | nowrap | pre | pre-line | pre-wrap（最后两个扩展自pre）。值为normal时会按照常规的HTML规则处理空白。nowrap会命令浏览器（除Navigator 4之外）忽略代码文本中的换行，并且只有当页面中某处出现明确的HTML换行时才会在该处进行换行，例如添加一个br元素。pre则让浏览器显示页面作者在源代码中输入的所有空白，而且不会改变元素的字体设定。

初始值 normal

示例

```
div.example {white-space: pre}
```

应用于 所有元素。

widows

IE *5 (Mac)* NN *6* Moz *n/a* Saf *n/a* Op *7* CSS *2*

通过继承得到：是

当页面中出现分页符时，这个属性设置了出现在页面顶部的段落的最小行数。出现分页符前在页面底部显示的行数设置请参见orphans属性。

CSS语法 `widows: lineCount`

值 表示行数的整数值。

初始值 2

应用于 块级元素。

width

IE *4* NN *4* Moz *all* Saf *all* Op *all* CSS *1*

通过继承得到：否

设置块级元素、置换元素，以及已定位元素中内容的宽度，此时不包括边框、填充和边距。

但在IE 6的标准兼容模式下计算元素的宽度时，IE/Windows会将左右两侧的边距、填充，以及边框考虑在内，请参见第1章中的DOCTYPE元素。如果浏览器遵循CSS标准，那么宽度只会应用于元素的内容部分，而不包括其边框、填充或边距。

CSS语法 `width: length | percentage | auto`

值 请参见本章中有关CSS长度值的讨论。如果将此值设定为auto，那么就会让浏览器在当前窗口宽度限制下，根据显示内容所需的空间大小来决定元素盒的宽度。

初始值 auto

示例

```
div#announce {position: relative; left: 30; width: 240}
textarea {width: 80%}
```

应用于 Navigator 4：所有的绝对定位元素；IE 4：applet、div、embed、fieldset、hr、iframe、img、input、marquee、object、span、table及textarea元素；IE 5、Mozilla、Safari和Opera：除无法替代的行内元素、表格列元素及表格分组元素之外的所有元素。

对象模型引用方式 `[window.]document.getElementById("elementID").style.width`

word-break

IE *5 (Win)* NN *n/a* Moz *all* Saf *all* Op *all* CSS *n/a*

通过继承得到：否

为包含表意（象形）语言的文本内容指定断字（word-break）样式。

CSS语法 `word-break: breakType`

值 可使用如下的常量值之一：break-all | keep-all | normal。

初始值 normal

示例

```
div {word-break: keep-all}
```

应用于 块级元素及与表格相关的元素。

对象模型引用方式 `[window.]document.getElementById("elementID").style.wordBreak`

word-spacing

IE *4 (Mac)/6 (Win)* NN *n/a* Moz *all* Saf *all* Op *all* CSS *1*

通过继承得到：是

当文本并未受到外部字间距的约束（如设置为justify的align属性）时，可以通过此属性设定字间距。如果在某个受此属性控制的元素中还内嵌着其他的元素，那么IE 5/Macintosh在处理内嵌元素的字间距时可能会造成字符重叠。

CSS语法 word-spacing: *length* | normal

值 normal会让浏览器根据其显示计算来处理字间距。请参见本章中有关CSS长度值的讨论。

初始值 normal

应用于 所有元素。

对象模型引用方式 [window.]document.getElementById("*elementID*").style.*wordSpacing*

word-wrap

1023

IE *5.5* NN *n/a* Moz *n/a* Saf *1.3/2* Op *n/a* CSS *n/a*

通过继承得到：是

此属性为块级元素指定自动换行的样式。如果仅有一个字超出了元素容器盒的宽度，那么常用的处理方式就是让这部分内容超出容器盒宽度，而不进行换行处理。当其值为break-word时，也可以迫使长单词在容器盒的边缘发生换行。

CSS语法 word-wrap: *wrapStyle*

值 以下常量值之一：break-word | normal。

初始值 normal

应用于 块级元素、行内元素及已定位的元素。

对象模型引用方式 [window.]document.getElementById("*elementID*").style.wordWrap

writing-mode

IE *5.5* NN *n/a* Moz *all* Saf *all* Op *all* CSS *n/a*

通过继承得到：是

主要在那些设置使用竖直排列形式的语言中调用这个属性，它可以控制文本内容的前进方向，即从左到右还是从右到左。

CSS语法 writing-mode: *direction*

值 以下常量值之一：lr-tb | tb-rl。tb-rl还可以将某些语言的文字旋转90度。

初始值 lr-tb

应用于 所有元素。

对象模型引用方式 [window.]document.getElementById("*elementID*").style.writingMode

z-index

IE *4* NN *4* Moz *all* Saf *all* Op *all* CSS *2*

通过继承得到：否

对一个已定位的元素而言，这个属性设定了在同一个父级容器内，这个元素相对于其他元素的叠放次序。请参见在线参考IV，它讲述了在多个容器内元素摆放层次之间的相互关系。

1024

CSS语法 z-index: *integer* | auto

值 任意整数值。而auto则等价于0。当同一个父级容器中的所有元素有相同的z-index值时，那么叠放次序将由元素的源代码顺序决定。

初始值 auto

示例 div#instrux {position: absolute; left: 50; top: 70; z-index: 2}

应用于 已定位元素。

对象模型引用方式 [window.]document.getElementById("*elementID*").style.zIndex

注意事项 在很多浏览器中，无论如何设定z-index属性值，总会在一个已定位元素之前显示表单控件，如按钮、文本框及特别的select元素等。这种显示机制就意味着一个已定位元素可能会发现位于正常内容排列中的一个表单控件突然被放置到它之前，例如，一个下拉菜单。对于这种情况，除非将表单控件或其表单容器的visibility属性设定为hidden，否则当已定位元素可见时没有任何解决办法。值得庆幸的是，IE 7已修正了这个问题。

zoom

IE *5.5* NN *n/a* Moz *n/a* Saf *n/a* Op *n/a* CSS *n/a*

通过继承得到：否

此属性可以控制显示元素内容时的放大倍数。在高分辨率的显示器中显示网页内容时，这个属性尤为有用。请参见第2章中的screen.logicalXDPI属性。

CSS语法 zoom: *scale* | *percentage* | normal

值 可以使用浮点数、比例系数（通常为1.0）或百分比（通常为100%）来表示放大倍数。

初始值 normal

示例 body {zoom: 200%}

应用于 所有元素。

对象模型引用方式 [window.]document.getElementById("*elementID*").style.zoom

JavaScript核心语言参考

JavaScript Core Language Reference

在本书的前几章中，我们已经介绍了动态HTML中与外观相关的各部分内容，无论是元素、对象还是样式，这些信息都可以呈现在页面之上。而本章中所涉及的脚本编程则可以看作是以上这些要素的融合剂，通过JavaScript可以访问并控制这些要素，而这也正是动态HTML中“动态”一词的精髓所在。本章所述的核心脚本编程语言均能够在各种跨浏览器的应用之中使用。这意味着我们将省略某些相关内容，例如VBScript、ActiveX控件及通过LiveConnect访问的Java类等，并且只会以能够广泛用于所有脚本浏览器的核心语言为主体内容。

正如在线参考I所述，Netscape发明了JavaScript语言。对应的Microsoft版本则称为JScript。但对所有源于JavaScript的语言而言，还有一种为大家所认可的通用命名标准，即与浏览器无关的版本——ECMAScript。浏览器生产商和ECMA（European Computer Manufacturers Association）标准小组经过大量协商，最终在脚本编程语言的核心元素实现方面达成了一致，从而才产生了ECMAScript。在编写核心脚本语言代码时，最为重要的一点就是清晰地获知各种版本的浏览器所能够支持的语言版本。在本章列出的各个说明项中，可以快速地找到每个核心语言对象、属性、方法、函数、运算符，以及控制指令被首度引入时所对应的浏览器版本。

随着`script`元素的`type`属性使用率的提升，较之以往，JavaScript的版本问题已经不再特别重要，在使用时必须了解特定浏览器能否支持对象、属性或方法的日子已经一去不返。与使用JavaScript版本号来转换脚本相比，现在所使用的对象检测技术更具优势。因此，本章所列出的各项内容中并未涉及JavaScript的版本号。

关于静态对象

与Java这种面向对象的重量级语言不同，基于对象的JavaScript语言几乎不涉及传统的面向对象的各种内容。因此，脚本程序员也不会以静态对象及对象实例化的观点来思考问题。但在某些情况下，这种观点还有其用武之地。

某些核心语言对象表现得就像一个真正的静态对象。`Math`对象就是一个很好的范例，它不仅包含一系列的属性与方法，而且在使用它们进行数学计算时并不需要建立一个对象实例。

与Math相反，`Date`对象虽然也是一个静态对象，但每次创建一个新的日期时均会生成该对象的一个实例，代码如下：

```
var now = new Date();
```

在这个例子中，变量`now`就是`Date`对象反映某个时间点的一个实例。通过这个实例提供的一系列方法就可以获取日期及时间信息，并且重新设置这些信息。虽然这些方法实际上就存在于静态对象之中，但由于对象实例中包含着能够影响这些方法的值，因此必须通过对象实例才能访问它们。此外，尽管对象是通过继承才获得的这些方法，但JavaScript很少用到这个术语。当然，在某些情况下，脚本也需要直接使用`Date`静态对象以获得其他的帮助，调用`UTC()`方法就是一例。

对大多数对象而言，要么属于静态对象，如`Math`等，要么必须创建对象实例之后才能使用，如`Object`、`Array`及`Number`等。只有很少的一部分的对象可以同时工作在上述两个模式之下，根据使用时具体是需要对象实例的

数据还是静态属性或方法就可以选择调用的模式。上文中介绍过的Date对象就是一个很好的例子。此外，RegExp对象也可以完成这两种功能，在执行一个相关的方法时它会创建一个正则表达式对象。而在某些情况下，通过访问静态对象（如String、Array等），使用prototype属性为其对象原型赋予新的属性和方法就可以达到修改其基本行为特征的目的。这些修改后的对象所产生的实例就会继承对象原型新增的属性或方法。

Mozilla 的读、写方法

由于预料到将来的ECMA会采用一些新的语言特性，因此基于Mozilla的浏览器定义了一种新的机制，以便定义读取（获得）及写入（设置）对象属性的函数，并将它们附加到对象之上。为防止与最终的标准语法产生冲突，Mozilla版本采用了一种特殊的“双-双”下划线语法，以便任意对象的prototype属性使用。此时会在方法名称的两端均添加一对下划线，如下所示：

```
objectName.prototype._ _defineGetter_ _("propertyName",functionReference);
objectName.prototype._ _defineSetter_ _("propertyName",functionReference);
```

1027

与简单指定自定义原型属性相比，这个机制的不同之处在于，在获得或设置一个属性值时需要更多的脚本指令，而它们则由原型方法中引用的函数决定。

为了证明这一功能，下面的示例代码使用了一个DOM对象为所有的HTML元素对象提供一个innerText属性。虽然W3C DOM并未提供这个属性，但在页面脚本中加入示例中的方法代码后，就可以在页面内所有元素中添加innerText属性。尽管此处定义的方法为匿名方法，但任何方法引用都可以满足它的要求。在跨浏览器应用中，必须保护这些脚本指令，以便只有当HTMLElement可以引用，并且已实现了_ _defineGetter_ _()或_ _defineSetter_ _()方法后再运行这些代码。

```
HTMLElement.prototype._ _defineGetter_ _("innerText", function () {
  var rng = document.createRange();
  rng.selectNode(this);
  return rng.toString();
});
HTMLElement.prototype._ _defineSetter_ _("innerText", function(newTxt)) {
  var rng = document.createRange();
  rng.selectNodeContents(this);
  rng.deleteContents();
  this.appendChild(document.createTextNode(newTxt));
  return newTxt;
});
```

一旦脚本指令尝试读取元素的innerText属性，就会执行与_ _defineGetter_ _()方法相关的函数，并返回目标值。相反地，当为元素的innerText属性赋值时，_ _defineSetter_ _()方法中的函数也会运行。

支持 XML 的 ECMAScript

有朝一日，当ECMAScript标准（ECMA-262）非常稳定时，它就会继续向前发展。独立的ECMA-257文档通过加入一些原生的对象对JavaScript语言进行了扩展，以便创建并操纵XML数据。这种支持XML标准的新ECMAScript就简称为E4X。由于在JavaScript与W3C DOM中，元素、属性、子节点及很多相关概念都是E4X中不可分割的组成部分，因此它成为了JavaScript和W3C DOM的联系纽带。但E4X也是对JavaScript的一种自然扩展，它使得XML成为了一种拥有方法与构造函数的数据类型。

AJAX应用程序是E4X扩展发挥作用的重要场所，在AJAX中，新的对象促进了XML文档元素数据的获取，从而避免了繁琐的DOM节点树解析操作。同样地，像SOAP应用一样，当AJAX应用程序的服务器端需要从客户端获取XML表单中的数据时，E4X对象通过XMLHttpRequest对象实例提交数据，以简化传输时XML内容的创建。

1028

对于DOM中的元素或元素分组而言，E4X标准使用术语“property”来进行指代。这样做的部分原因在于，E4X语法通过ECMAScript属性语法来引用元素。而本章在讨论这些主题时，亦会沿用其常用称呼——“元素”，以便与DOM脚本程序员对这些数据的考虑方式保持一致。

E4X标准的很多开发工作都是在Mozilla基金会的倡导下开展起来的。因此，基于Mozilla的浏览器从1.8版起就开始支持E4X功能。而在那些尚未支持E4X的浏览器中，有些E4X代码会导致脚本错误。尽管将`script`元素的`type`属性设置为“`type="text/javascript; e4x=1"`”就可以让大多数的浏览器回避E4X代码，但并不是所有的浏览器均能够识别这种非常规的“`text/javascript`”类型扩展，因此在页面载入过程中，这一问题会导致载入过程在E4X符号上出现阻塞。

ECMAScript 的保留关键字

在标识符命名时，请勿用下面这些区分大小的单词：`abstract`、`boolean`、`break`、`byte`、`case`、`catch`、`char`、`class`、`const`、`continue`、`debugger`、`default`、`delete`、`do`、`double`、`else`、`enum`、`export`、`extends`、`final`、`finally`、`float`、`for`、`function`、`goto`、`if`、`implements`、`import`、`in`、`instanceof`、`int`、`interface`、`long`、`native`、`new`、`package`、`private`、`protected`、`public`、`return`、`short`、`static`、`super`、`switch`、`synchronized`、`this`、`throw`、`throws`、`transient`、`try`、`typeof`、`var`、`void`、`volatile`、`while`、`with`。

在以上这些单词中，很多关键字已经存在于现有的JavaScript版本之中，而剩余部分则可能用于未来的JavaScript版本。

此外，在E4X脚本代码的标识符名称中，请勿用下面这些区分大小的单词：`each`、`namespace`、`XML`。

核心对象

ActiveXObject

IE *4(Win)* NN *n/a* Moz *n/a* Saf *n/a* Op *n/a* ECMA *n/a*

在页面与ActiveX控件（IE术语称之为“自动化”对象）之间，IE/Windows提供了一个直接的接口，并在Windows系统中对其进行了注册。通过创建一个`ActiveXObject`实例，就可以为脚本提供一个ActiveX控件的引用，从而访问控件的属性或调用其方法。自动化对象中某些尚未暴露的方法与属性须要从MSDN（Microsoft Developer Network）主页中获取，其链接如下：*http://msdn.microsoft.com*。此外，OLE/COM对象查看器（OLE/COM Object Viewer）也可以用于观察这些持久化对象。不妨从以下链接中获取相关信息：*http://msdn.microsoft.com/zh-cn/library/d0kh9f4c(VS.80).aspx*。`GetObject()`全局函数还提供了一种通过本地路径名获取一个自动化对象引用的方法。

ActiveXObject创建方式

```
var myObj = new ActiveXObject(appName.className[, remoteServerName])
```

属性 无。

方法 无。

arguments

IE *4* NN *3* Moz *all* Saf *all* Op *7* ECMA *1*

每个函数在调用时均拥有一个`arguments`对象，可以将它作为该函数的一个属性进行访问。浏览器会自动创建这个对象，而且无法在拥有该函数的环境之外进行创建。例如，请考虑下面这个典型的函数定义：

```
function myFunc() {
// 函数脚本指令
}
```

函数中的脚本指令可以通过以下引用来访问`arguments`对象：`arguments`。

这个对象总是包含一个callee属性，它是一个对应于所有相同函数的引用，本章后续内容中会解释这个属性。此外，通过数组符号还可以使用arguments对象来访问函数中的每个参数值。在上例中，myFunc()函数中的代码可以通过如下引用来访问传入的参数：arguments[0]。

关于它的实际应用说明，请参见本章后文中Function对象的arguments属性。

属性 callee、length

方法 无。

1030

callee

IE 5(Mac)/5.5(Win) NN n/a Moz all Saf all Op 7 ECMA 1

只读

提供一个引用，以指向创建arguments对象的函数。因此在匿名函数中，脚本指令可以通过这个属性来递归调用同一个（未命名）函数，示例代码如下：

示例

```
myObj.doThis = function(input) {
     // 根据输入参数值进行操作的函数指令
     if (!someCondition) {
          // 递归调用这个匿名函数
          arguments.callee(input);
     }
}
```

值 函数对象引用。

length

IE 4 NN 3 Moz all Saf all Op 7 ECMA 1

只读

返回当前调用中传递给函数的参数数量。函数中已定义的参数变量的数目不会影响返回值。

示例

```
function myFunc() {
   for (var i = 0; i < arguments.length; i++) {
      ...
   }
}
```

值 整数值。

Array

IE 4 NN 3 Moz all Saf all Op 7 ECMA 1

数组就是由一个或多个数据块组成的有序集合。JavaScript数组项可以由任意的数据类型组成，而且可以在同一个数组中使用不同的数据类型。对于数组中的每个数据项均拥有一个索引值。默认情况下，这些索引值是从零开始的整数，即，第1个数据的索引值为0。索引值也可以是字符串，但字符串索引值看起来更像一个数组对象的属性，而且无法影响其数字索引，正因为如此，无法通过数组的length属性来迭代字符串型的索引，而只能使用for-in循环完成迭代过程。在同一个数组对象中，彼此独立的字符型索引和整数索引可以共存。

1031

但无论是通过名称还是索引值来访问数组中的某一项都需要使用方括号，例如：cars[0]、cars["Ford"]。

此外，还可以通过由数组组成的数组来模拟多维数组。此时可以通过以下语法来引用二维数组中的某一项：myArray[x][y]。

JavaScript数组中的数据项数量，即数组长度，可能会随着时间而发生改变。因此，不需要使用特定的大小来初始化一个空数组，即使这样做了也不会带来什么益处。如果要向一个非固定长度的数组中插入新的数据，那么可以进行如下操作：

```
cars[cars.length] = "Bentley";
```

从IE 4和Navigator 4开始，可以使用一种简便技术来创建数组，即，使用方括号来包含一些使用文本符号表示的值。

数组创建方式

```
var myArray = new Array();
var myArray = new Array(sizeInteger);
var myArray = new Array(element0, element1, ..., elementN);
var myArray = [element0, element1, ..., elementN];
```

属性 constructor、length、prototype

Methods concat()、every()、filter()、forEach()、indexOf()、join()、lastIndexOf()、map()、pop()、push()、reverse()、shift()、slice()、some()、sort()、splice()、toLocaleString()、toString()、unshift()

constructor

IE 4 NN 4 Moz all Saf all Op 7 ECMA 1

可读/可写

它指向Array对象实例创建函数的一个引用，即浏览器中的Array()构造函数。

示例

```
if (myVar.constructor == Array) {
   // process native string
}
```

值 函数对象引用。

length

IE 4 NN 3 Moz all Saf all Op 7 ECMA 1

可读/可写

提供数组中数字索引项的计数值。如果创建该数组的构造函数指定了一个初始长度，那么即使数据尚未完全占据整个数组，length属性也会返回该初始长度。

示例

```
for (var i = 0; i < myArray.length; i++) {
   ...
}
```

值 整数值。

prototype

IE 4 NN 3 Moz all Saf all Op 7 ECMA 1

可读/可写

这是静态Array对象的一个属性。可以使用这个属性为当前文档中将要创建的数组实例指定新的属性和方法。例如，下面这个方法创建了一个按照降序排列的元素数组，不同元素间使用回车符号进行分隔：

```
function formatAsList() {
   var output = "";
   for (var i = this.length - 1; i >= 0; i--) {
      output += this[i] + "\n";
   }
   alert(output);
}
```

如果想通过一个数组来调用这个方法，那么就可以将其引用赋予prototype属性，并根据需要对它进行命名：

```
Array.prototype.showReverseList = formatAsList;
```

如果脚本依次创建一个数组：

```
var stooges = new Array("Moe", "Larry", "Curly", "Shemp");
```

那么新数组就会包含一个可用的showReverseList()方法。其调用方法如下：

```
stooges.showReverseList();
```

通过相同的方法还可以添加其他属性。这样一来，就可以在不打乱数组数据排序的前提下为数组添加其他信息，例如，其创建事件等。当新文档载入窗口或框架时，就会再次刷新静态Array对象。

示例　　`Array.prototype.created = "";`

值　　任意数据，其中包括方法引用。

concat( )

IE *4* NN *4* Moz *all* Saf *all* Op *7* ECMA *1*

```
concat(item1[, item2[, ...itemN]])
```

根据方法参数，返回一个由当前数组对象和另外一个或多个数组对象（或其他值）组成的新数组。

```
var combinedArray = myArray1.concat(myArray2, someValue);
```

在这一处理过程中，并不会改变原有数组。

返回值　　一个Array对象。

参数

item1...itemN

任意的JavaScript值，其中包括其他数组。

every( )

IE *n/a* NN *n/a* Moz *1.8* Saf *n/a* Op *n/a* ECMA *n/a*

```
every(functionRef[, thisObject])
```

在数组的每个元素上执行一个函数（回调）操作。如果每个函数调用均返回true，那么every()函数才会返回true，一旦某个函数调用返回了false，那么every()函数就会停止其他的外部函数调用，并返回false。

调用时，会通过3个参数将数组元素的信息传递给回调函数：元素值、元素的索引值及数组引用，示例代码如下：

```
function allNumbers(val, i, array) {
    return (typeof val == "number");
}
var myArray = [10, 100, 1000];
var result = myArray.every(allNumbers);
// 对此数组而言，返回结果为 true
```

这时允许回调函数修改原始数组，这样就有可能影响接下来的回调操作。在调用every()时，只有数组长度是固定不变的，而且在回调的过程中会向回调函数传入元素的当前值。如果在回调过程中某个元素已被删除，那么传入回调函数中的元素值为“undefined”。

如果在作用域的限制下，回调函数还需要使用其他对象，那么可以使用第2个参数传递该对象的引用，相关示例请参见forEach()。

返回值　　布尔值。

参数

functionRef

可以接收数组值、数组索引及数组引用这3个参数的函数的引用，而且该函数应该返回一个布尔值。

thisObject

一个可选的对象引用，回调函数中的指令需要该对象的作用域。

filter( )

IE *n/a* NN *n/a* Moz *1.8* Saf *n/a* Op *n/a* ECMA *n/a*

```
filter(functionRef[, thisObject])
```

在数组的每个元素上执行一个函数（回调）操作，并返回一个由所有已通过回调函数测试的值组成的新数组，即，回调函数在检测该数组中的值时均会返回true。除非回调函数对原数组进行修改，否则原数组不会发生改变。

调用时，会通过3个参数将数组元素的信息传递给回调函数：元素值、元素的索引值以及数组引用，示例代码请参见every()。该方法应该返回一个布尔值，以表明该值是否应该成为filter()返回的新数组中的一员。需要注意的是，返回的数组可能会比原数组小。

由于允许回调函数修改原始数组，这样就有可能影响接下来的回调操作。在调用filter()时，只有数组长度是固定不变的，而且在回调的过程中会向回调函数传入元素的当前值。如果在回调过程中某个元素已被删除，那么传入回调函数中的元素值为“undefined”。

如果在作用域的限制下，回调函数还需要使用其他对象，那么可以使用第2个参数传递该对象的引用。

返回值 数组。

参数

functionRef

可以接收数组值、数组索引及数组引用这3个参数的函数引用。而且该函数应该返回一个布尔值。

thisObject

一个可选的对象引用，回调函数中的指令需要该对象的作用域。

forEach( )

IE *n/a* NN *n/a* Moz *1.8* Saf *n/a* Op *n/a* ECMA *n/a*

```
forEach(functionRef[, thisObject])
```

在数组的每个元素上执行一个函数或回调操作，与上文中两个函数不同的是，它不会返回任何值。

调用时，会通过3个参数将数组元素的信息传递给回调函数：每次迭代时所需的一个元素值、该元素索引值及数组引用。

这时允许回调函数修改原始数组，这样就有可能影响接下来的回调操作。在调用forEach()时，只有数组长度是固定不变的，而且在回调的过程中会向回调函数传入元素的当前值。如果在回调过程中某个元素已被删除，那么传入回调函数中的元素值为“undefined”。

如果在作用域的限制下，回调函数还需要使用其他对象，那么可以使用第2个参数传递该对象的引用。在下面这段代码中，myObj对象会作为第2个参数传递给doNumberTest()方法，以增加该对象中对象属性的值。

```
var myObj = {
    successCounter: 0,
    doNumberTest: function(val) {
        if (typeof val == "number") {
            this.successCounter++;
        }
    },
    report: function () {
        alert("Successful " + this.successCounter + " times!");
    }
};
var myArray = [10, "100", 1000];
myArray.forEach(myObj.doNumberTest, myObj);
```

此后，可以调用myObj.report()来读取myObj.successCounter的累加值。在这个例子中，由于数组中包含两个数字，因此每次调用forEach()方法会使得该值增加两次。

返回值 无。

参数

functionRef

可以接收数组值、数组索引及数组引用这3个参数的函数引用。该函数不应该返回任何值。

thisObject

一个可选的对象引用，回调函数中的指令需要该对象的作用域。

indexOf( )

IE *n/a* NN *n/a* Moz *1.8* Saf *n/a* Op *9* ECMA *n/a*

```
indexOf(searchElement[, fromIndex])
```

返回一个从零开始的整数值，以指明待查询元素的第1个实例在数组中的具体位置。如果未在数组中找到对应元素，那么返回值为-1。查询时，总是从数组的起始位置开始，一直到其末尾才结束，而且会执行JavaScript的严格相等运算符（===），这意味着只有当值与数据类型均相同时才能匹配成功。实际上，通过这个方法可以避免编写循环代码来遍历所有的数组值，而快捷地找到数组中某个已知量的索引值，或者检查某个值是否位于数组之中。

返回值 整数值。如果待查询元素位于数组中，那么返回0或者其他正整数，否则返回-1。

参数

searchElement

任意的JavaScript值，其中包括其他数组或DOM对象。

fromIndex

一个从零开始的整数值，以指明查询的起始元素。负值表示从元素末尾开始的某个值，但查询过程始终遵循从头至尾这一固定方向。

join( )

IE *4* NN *3* Moz *all* Saf *all* Op *7* ECMA *1*

```
join(["delimiterString"])
```

返回由数组所包含的所有项组成的一个字符串。通过方法的参数，可以设置数组项之间的分隔符。需要注意的是，只有能够通过整数索引值来访问的数组项才会进入此字符串。而那些通过字符串进行索引的值会视为数组对象的属性，它们与通过整数来进行索引的值彼此独立。因此join()方法只对整数索引项有效。

返回值 字符串。

参数

delimiterString

任何字符串。非文本或数字字符必须使用URL编码进行编码，如“%0D”等价于回车符号。默认的分隔符为逗号，即“,”。

lastIndexOf( )

IE *n/a* NN *n/a* Moz *1.8* Saf *n/a* Op *9* ECMA *n/a*

```
lastIndexOf(searchElement[, fromIndex])
```

当从数组末尾开始进行搜索时，返回一个从零开始的整数值，以指明待查询元素的第1个实例在数组中的具体位置。如果未在数组中找到对应元素，那么返回值为-1。查询时，总是从数组的末尾开始，一直到其起始位置结束，而且会执行JavaScript的严格相等运算符（===），这意味着只有当值与数据类型均相同时才能匹配成功。这个方法与indexOf()很类似，只是它们的搜索方向完全相反。

返回值 整数值。如果待查询元素位于数组中，那么返回0或者其他正整数，否则返回-1。

参数

searchElement

任意的JavaScript值，其中包括其他数组或DOM对象。

fromIndex

一个从零开始的整数值，以指明查询的起始元素。负值表示从元素末尾开始的某个值，但查询过程始终遵循从尾至头这一固定方向。

map( )

IE *n/a* NN *n/a* Moz *1.8* Saf *n/a* Op *n/a* ECMA *n/a*

```
map(functionRef[, thisObject])
```

在数组的每个元素上执行一个函数（回调）操作，并返回一个由回调函数返回值组成的新数组。除非回调函数对原数组进行修改，否则原数组不会发生改变。

调用时，会通过3个参数将数组元素的信息传递给回调函数：元素值、元素的索引值及数组引用。这个回调函数应该返回一个值，然后它将成为map()返回数组中的一个元素，示例代码如下：

```
function makeStringy(val, i, array) {
   return val.toString();
}
var myArray = [10, 100, 1000];
var newArray = myArray.map(makeStringy);
```

这时允许回调函数修改原始数组，这样就有可能影响接下来的回调操作。在调用map()时，只有数组长度是固定不变的，而且在回调的过程中会向回调函数传入元素的当前值。如果在回调过程中某个元素已被删除，那么传入回调函数中的元素值为“undefined”。

如果在作用域的限制下，回调函数还需要使用其他对象，那么可以使用第2个参数传递该对象的引用。

返回值 数组。

参数

functionRef

可以接收数组值、数组索引及数组引用这3个参数的函数引用，而且该函数应该返回一个值。

thisObject

一个可选的对象引用，回调函数中的指令需要该对象的作用域。

pop( )

IE *5.5(Win)* NN *4* Moz *all* Saf *all* Op *7* ECMA *2*

返回数组中最后一项的值，并将它从数组中删除。同时会将数组的长度减一。

返回值 任何JavaScript值。

参数 无。

push( )

IE *5.5(Win)* NN *4* Moz *all* Saf *all* Op *7* ECMA *2*

```
push(item1[, item2[, ...itemN]])
```

向数组末尾附加一项或多项数据。同时会使数组的长度相应增加。

返回值 push操作完成后数组的长度值。

参数

item1...*itemN*

使用逗号分隔的一个或多个JavaScript值，其中包括对象引用。

reverse( )

IE *4* NN *3* Moz *all* Saf *all* Op *7* ECMA *1*

反转数组中所有数据的排列顺序，并返回这个按倒序排序的数组的副本。reverse()方法不仅会重新排列数组中的值，而且它还会返回反转数组的副本。

返回值 一个Array对象。

参数 无。

shift( )

IE *5.5(Win)* NN *4* Moz *all* Saf *all* Op *7* ECMA *2*

返回数组中第1项的值，并将它从数组中删除。同时会将数组的长度减一。

返回值 任何JavaScript值。

参数 无。

slice( )

IE 4 NN 2 Moz all Saf all Op 7 ECMA 2

```
slice(startIndex[, endIndex])
```

返回由主数组中连续项组成的数组。方法参数决定了选中范围的起点和终点位置。

返回值 一个Array对象。

参数

startIndex

一个从零开始的整数，表示当前数组内数据子集中第1项所在的位置。

endIndex

一个从零开始的整数，表示当前数组内数据子集中最后项所在的位置。如果忽略这一参数，那么会从*startIndex*指定的位置一直持续到数组末尾为止。

some( )

IE n/a NN n/a Moz 1.8 Saf n/a Op n/a ECMA n/a

1039

```
some(functionRef[, thisObject])
```

在数组中的每一项上执行一个函数（回调）操作，如果所有元素的回调操作均返回true，那么此方法也返回true，如果所有元素的回调操作均返回false，那么就会返回false。

调用时，会通过3个参数将数组元素的信息传递给回调函数：元素值、元素的索引值及数组引用，而且该函数应该返回一个布尔值。

这时允许回调函数修改原始数组，这样就有可能影响接下来的回调操作。在调用some()时，只有数组长度是固定不变的，而且在回调的过程中会向回调函数传入元素的当前值。如果在回调过程中某个元素已被删除，那么传入回调函数中的元素值为“undefined”。

如果在作用域的限制下，回调函数还需要使用其他对象，那么可以使用第2个参数传递该对象的引用。

返回值 布尔值。

参数

functionRef

可以接收数组值、数组索引及数组引用这3个参数的函数引用，而且该函数应该返回一个布尔值。

thisObject

一个可选的对象引用，回调函数中的指令需要该对象的作用域。

sort( )

IE 4 NN 3 Moz all Saf all Op 7 ECMA 1

```
sort([compareFunction])
```

根据每个数组项值对应字符串的ASCII值进行排序，或根据指定的比较函数进行排序。在比较过程中sort()方法会反复调用比较函数，并将来自于数组中的两个值传入该方法中。此时的比较函数应该返回一个整数值，下表解释了它在sort()函数中的具体含义：

值	含义	值	含义	值	含义
<0	第二个值应该位于第一个值之后	0	两个值的排列顺序不用发生改变	>0	第一个值应该位于第二个值之后

下面这个比较函数会依照数字顺序排列数组中的所有值：

```
function doCompare(a, b) {
    return a - b;
}
```

1040

如果要通过这个函数来排列一个数组，那么请使用如下指令：

```
myArray.sort(doCompare);
```

当sort()方法完成其工作时，它肯定已经将数组所有值依次发送至doCompare()函数，每次会比较两个值，并根据冒泡排序方式来重新排列数组。

如果数组的元素中包含对象，那么可以通过那些对象的属性值来进行排序。例如，下面这个排序函数会根据对象的age属性进行排序：

```
function compareByAge(a, b) {
    return a.age - b.age;
}
```

sort()方法不仅会重新排列数组中的值，而且它还会返回已排序数组的副本。

返回值 一个已根据排序准则排序的Array对象。

参数

compareFunction

可以接受两个参数并返回一个整数结果的函数的引用。

splice( )

IE *5.5* NN *4* Moz *all* Saf *all* Op *7* ECMA *2*

```
splice(startIndex, deleteCount[, item1[, item2[, ...itemN]]])
```

从数组内删除零个或多个连续项，并向它们原来所在的位置插入新值。数组的长度值会根据实际情况自动进行调整。

返回值 由被删除项组成的Array对象。

参数

startIndex

一个从零开始的整数，表示当前数组内数据子集中第1项所在的位置。

deleteCount

一个整数值，表示从*startIndex*开始需要从数组中删除多少项。

item1...*itemN*

以逗号分隔的JavaScript值列表，会将它们插入到被删除项所在的位置。这些项的具体数目并不一定要与*deleteCount*值相对应。

toLocaleString( )

IE *5.5* NN *n/a* Moz *all* Saf *all* Op *7* ECMA *2*

返回一个以逗号进行分隔的字符串值，从理论上说，其格式与所用语言及浏览器的默认语言有关。具体的实现细节随浏览器和数据类型的不同而发生变化。IE 5.5及后续版本会将所有类型的数字转换成包含两位小数的十进制数字符串，但当数组项为引用时则会触发一个错误。Mozilla会保留数字中的整数部分，并以“[object *objectType*]”的形式显示对象引用。ECMA标准将具体的解释权留给了浏览器生产商。

返回值 使用逗号分隔的字符串。

参数 无。

toString( )

IE *4* NN *3* Moz *all* Saf *all* Op *7* ECMA *1*

返回一个使用逗号分隔各值的字符串，它等价于调用将逗号作为参数的Array.join()方法。所有的值均会被转换成等价的字符串，其中包括对象引用（IE/Windows使用[object]，而IE 5/Macintosh、Mozilla和Safari则使用[object *objectType*]）。

返回值 使用逗号分隔的字符串。

参数 无。

unshift( )

IE 5.5(Win) NN 4 Moz all Saf all Op 7 ECMA 2

```
unshift(item1[, item2[, ...itemN]])
```

在数组头部插入一项或多项数据。数组长度会根据插入项的数目发生改变，而此方法则会返回数组的新长度。

返回值 整数值。

参数

item1...itemN

可以使用一个或多个JavaScript值，各个值之间使用逗号进行分隔。

Boolean

IE 4 NN 3 Moz all Saf all Op 7 ECMA 1

Boolean对象可以描述任何赋值为true或false的数据。由于将true或false值指定给变量时，浏览器会自动Boolean对象，因此一般来说并不需要担心类型转换问题。但那些放置在引号中的属性值只会被视为字符串。

Boolean对象创建方式

```
var myValue = new Boolean()
ar myValue = new Boolean(BooleanValue);
var myValue = BooleanValue;
```

属性 constructor、prototype

方法 toString()、valueOf()

constructor

IE 4 NN 4 Moz all Saf all Op 7 ECMA 1

可读/可写

它指向Boolean对象实例创建函数的一个引用，即浏览器中的Boolean()构造函数。

示例

```
if (myVar.constructor == Boolean) {
   // 处理本地字符串
}
```

值 函数对象引用。

prototype

IE 4 NN 3 Moz all Saf all Op 7 ECMA 1

可读/可写

这是静态Boolean对象的一个属性。可以使用这个属性为当前文档中将要创建的Boolean实例指定新的属性和方法。相关示例请参考Array.prototype属性说明。通常并不需要为Boolean对象创建新的原型属性或方法。

示例

```
Boolean.prototype.author = "DG";
```

值 任意数据，包括方法的引用。

toString( )

IE 4 NN 4 Moz all Saf all Op 7 ECMA 1

以字符串数据类型的形式返回对象值。在警告对话框或文档内需要显示布尔值时，由于浏览器会自动将布尔值转换为字符串，因此在实际使用中通常并不需要这个方法。

返回值 "true" | "false"

参数 无。

valueOf( )

IE 4 NN 4 Moz all Saf all Op 7 ECMA 1

以布尔数据类型的形式返回对象值。通过简单的赋值来创建Boolean对象时并不需要使用这个方法。

返回值 布尔值：true | false。

参数 无。

Date

IE 3 NN 2 Moz all Saf all Op 7 ECMA 1

Date对象是一个静态对象，可以通过几种不同的构造函数来生成其实例对象。每个Date对象实例都是一个日期与时间的快照，这些实例均以相对于1970年1月1日0时的毫秒数来度量时间。负的毫秒数表示该日期之前的时间，而正数则表示该日期之后的时间。

使用日期的一种典型方式就是为当前时间，或者指定的时间及日期生成一个新的Date对象实例，而这些实例均以客户端的本地时间为基准。然后，就可以使用各种日期方法来获取或设置该时间的构成要素，如分钟、小时、日期或月份等。浏览器内部会以毫秒数的形式存储一个协调世界时日期（Coordinated Universal Time，UTC，它与格林尼治标准时间相同）。当命令浏览器提供该时间的某个构成要素时，它会自动根据客户端计算机控制面板中的时钟与时区设定，自动将该时间转换成本地时区。如果控制面板中的设置出现错误，那么时间与日期的计算也会出现偏差。

Creating a Date Object

```
var now = new Date();
var myDate = new Date("month dd, yyyy hh:mm:ss");
var myDate = new Date("month dd, yyyy");
var myDate = new Date(yy, mm, dd, hh, mm, ss);
var myDate = new Date(yy, mm, dd);
var myDate = new Date(milliseconds);
```

属性 constructor、prototype

方法 getDate()、getDay()、getFullYear()、getHours()、getMilliseconds()、getMinutes()、getMonth()、getSeconds()、getTime()、getTimezoneOffset()、getUTCDate()、getUTCDay()、getUTCFullYear()、getUTCHours()、getUTCMilliseconds()、getUTCMinutes()、getUTCMonth()、getUTCSeconds()、getVarDate()、getYear()、parse()、setDate()、setFullYear()、setHours()、setMilliseconds()、setMinutes()、setMonth()、setSeconds()、setTime()、setUTCDate()、setUTCFullYear()、setUTCHours()、setUTCMilliseconds()、setUTCMinutes()、setUTCMonth()、setUTCSeconds()、setYear()、toDateString()、toGMTString()、toLocaleDateString()、toLocaleString()、toLocaleTimeString()、toString()、toTimeString()、toUTCString()、UTC()、valueOf()

constructor

IE 4 NN 4 Moz all Saf all Op 7 ECMA 1

可读/可写

它指向Date对象实例创建函数的一个引用，即浏览器中的Date()构造函数。

示例

```
if (myVar.constructor == Date) {
// 处理本地字符串
}
```

值 函数对象引用。

prototype

IE 4 NN 3 Moz all Saf all Op 7 ECMA 1

可读/可写

这是静态Date对象的一个属性。可以使用这个属性为当前文档中将要创建的Date实例指定新的属性和方法。

相关示例请参考Array.prototype属性说明。

示例 `Date.prototype.author = "DG";`

值 任意数据，包括方法的引用。

getDate( )

IE 3 NN 2 Moz all Saf all Op 7 ECMA 1

返回某个Date对象实例所指定月份中的日历日期。

返回值 1到31的整数。

参数 无。

getDay( )

IE 3 NN 2 Moz all Saf all Op 7 ECMA 1

返回一个整数，用以表示Date对象实例指定日期在星期中具体对应于哪一天。

返回值 从0到6的整数。周日为0，周一为1，而周六为6。

参数 无。

getFullYear( )

IE 4 NN 4 Moz all Saf all Op 7 ECMA 1

返回某个Date对象实例所指定的日期的年份值。

返回值 整数值。Navigator 4的返回值不会小于0。IE及其他主流浏览器则可以返回负的年份值。

参数 无。

getHours( )

IE 3 NN 2 Moz all Saf all Op 7 ECMA 1

返回一个从零开始的整数，用以表示Date对象实例指定的时间具体对应于一天中的哪个小时。此时使用24小时计时系统。

返回值 介于0到23的整数。

参数 无。

getMilliseconds( )

IE 4 NN 4 Moz all Saf all Op 7 ECMA 1

返回一个从零开始的整数，用以表示Date对象实例所指定的时间具体位于哪一毫秒。

返回值 介于0到999的整数。

参数 无。

getMinutes( )

IE 3 NN 2 Moz all Saf all Op 7 ECMA 1

返回一个从零开始的整数，用以表示Date对象实例指定的时间具体位于哪一分钟。

返回值 介于0到59的整数。

参数 无。

getMonth( )

IE 3 NN 2 Moz all Saf all Op 7 ECMA 1

返回一个从零开始的整数，用以表示Date对象实例指定的时间所对应的月份值。由于这个方法的值从零开始计数，因此脚本程序员在第1次使用时难免会感到困惑。

返回值 介于0到11的整数。1月为0，二月为1，而十二月为11。

参数 无。

getSeconds( )

IE 3 NN 2 Moz all Saf all Op 7 ECMA 1

返回一个从零开始的整数，用以表示Date对象实例所指定的时间具体位于哪一秒。

返回值 介于0到59的整数。

参数 无。

getTime( )

IE 3 NN 2 Moz all Saf all Op 7 ECMA 1

返回一个从零开始的整数，用以表示从1970年1月1日开始，Date对象实例所指定的时间所经历的毫秒数。

返回值 整数值。

参数 无。

getTimezoneOffset( )

IE 3 NN 2 Moz all Saf all Op 7 ECMA 1

返回一个从零开始的整数值，以便为Date对象实例指出客户端计算机时钟与GMT时间之间的时差，单位为分钟。GMT时区以西的时区为正值，而以东的时区则为负值。在早期的浏览器中，尤其在Macintosh版本中，这个方法总是漏洞百出。

返回值 介于-720与720之间的整数。

参数 无。

getUTCDate( )

IE 4 NN 4 Moz all Saf all Op 7 ECMA 1

根据浏览器中存储的UTC时间，返回Date对象实例所指定的时间在当月中的日历日期。

返回值 1到31的整数。

参数 无。

getUTCDay( )

IE 4 NN 4 Moz all Saf all Op 7 ECMA 1

根据浏览器中存储的UTC时间，返回一个整数，用以表示Date对象实例指定的日期在星期中具体对应于哪一天。

返回值 从0到6的整数。周日为0，周一为1，而周六为6。

参数 无。

getUTCFullYear( )

IE 4 NN 4 Moz all Saf all Op 7 ECMA 1

根据浏览器中存储的UTC时间，返回Date对象实例所指定的时间的具体年份数字。

返回值 整数值。Navigator 4的返回值不会小于0。IE及其他主流浏览器可以返回负的年份值。

参数 无。

getUTCHours( )

IE 4 NN 4 Moz all Saf all Op 7 ECMA 1

根据浏览器中存储的UTC时间，返回一个从零开始的整数，用以表示Date对象实例指定的时间对应于一天中的哪个小时。此时使用24小时计时系统。

返回值 介于0到23的整数。

参数 无。

1047

getUTCMilliseconds( )

IE 4 NN 4 Moz all Saf all Op 7 ECMA 1

根据浏览器中存储的UTC时间，返回一个从零开始的整数，用以表示Date对象实例指定的时间具体位于哪一毫秒。

返回值 介于0到999的整数。

参数 无。

getUTCMinutes( )

IE 4 NN 4 Moz all Saf all Op 7 ECMA 1

根据浏览器中存储的UTC时间，返回一个从零开始的整数，用以表示Date对象实例指定的时间具体位于哪一分钟。

返回值 介于0到59的整数。

参数 无。

getUTCMonth( )

IE 4 NN 4 Moz all Saf all Op 7 ECMA 1

根据浏览器中存储的UTC时间，返回一个从零开始的整数，用以表示Date对象实例指定的时间具体位于哪一月。由于这个方法的值从零开始计数，因此脚本程序员在第1次使用时难免会感到困惑。

返回值 介于0到11的整数。一月为0，二月为1，而十二月为11。

参数 无。

getUTCSeconds( )

IE 4 NN 4 Moz all Saf all Op 7 ECMA 1

根据浏览器中存储的UTC时间，返回一个从零开始的整数，用以表示Date对象实例指定的时间具体位于哪一秒。

返回值 介于0到59的整数。

参数 无。

getVarDate( )

IE 4 NN n/a Moz n/a Saf n/a Op n/a ECMA n/a

按照VT_DATE格式返回日期值，以用于面向Windows的应用程序，如ActiveX控件及VBScript。此值无法用于JavaScript的日期计算。

返回值 VT_DATE格式的日期值（无法用于JavaScript的日期计算）。

参数 无。

1048

getYear( )

IE 3 NN 2 Moz all Saf all Op 7 ECMA n/a

返回一个数字值，以表示一个Date对象实例的年份值，但它的执行结果并不规范。从理论上说，这个方法应该返回一个从1900开始计算的年份值。因此，位于1900到1999之间的年份值会返回一个一位或两位的数字值。但是，一旦日期到达2000年，那么这个模式就会出现错误。因此，与getFullYear()方法类似，某些浏览器也让getYear()方法返回四位数字的年份值。基于这个原因，如果浏览器能够兼容getFullYear()方法，那么就最好直接使用该方法。需要注意的是，ECMA已经不再支持getYear()，而getFullYear()则依然可以使用。

返回值 针对1900到1999的年份值，返回一个介于0至99的整数，而针对2000年以后的年份，某些浏览器会返回一个四位整数，而其他浏览器则会继续从1900开始计算，并返回一个大于100的整数。

参数 无。

parse( )

IE 3 NN 2 Moz all Saf all Op 7 ECMA 1

```
parse("dateString")
```

静态Date对象方法，返回参数字符串指定的日期所对应的等价毫秒数。

返回值 使用毫秒数表示的日期。

参数

dateString

源自Date对象的任意有效时间日期字符串。相关示例请参见toString()、toGMTString()及toLocaleString()方法。

setDate( )

IE 3 NN 2 Moz all Saf all Op 7 ECMA 1

```
setDate(dateInt)
```

为Date对象实例设置其在月份中的日期。如果指定的日期超出了对象当前月的末尾日期，那么该对象会在后续月份中重新设置日期。例如，如果将Date对象实例设置到2007年12月25日，那么使用如下代码就可以找到该日期十天之后的日历日期：

```
myDate.setDate(myDate.getDate() + 10);
```

通过日期计算，myDate等价于为2008年1月4日。

返回值 使用毫秒数表示的新日期。

参数

dateInt

日期整数。

setFullYear( )

IE 4 NN 4 Moz all Saf all Op 7 ECMA 1

```
setFullYear(yearInt)
```

为Date对象实例指定年份。

返回值 使用毫秒数表示的新日期。

参数

yearInt

整数值。Navigator 4不允许使用小于0的参数值。在IE及其他主流浏览器中则可以使用负的年份值。

setHours( )

IE 3 NN 2 Moz all Saf all Op 7 ECMA 1

```
setHours(hourInt)
```

为Date对象实例设置其在一天中的小时值。此时使用24小时计时系统。如果指定的时间超出了对象当前天的末尾，那么该对象会在后续日期中重新设置时间。

返回值 使用毫秒数表示的新日期。

参数

hourInt

一个从零开始的整数。

setMilliseconds( )

IE 4 NN 4 Moz all Saf all Op 7 ECMA 1

```
setMilliseconds(msInt)
```

为Date对象实例设置其毫秒数。

返回值 使用毫秒数表示的新日期。

参数

msInt

一个从零开始的毫秒整数。

setMinutes( )

IE 3 NN 2 Moz all Saf all Op 7 ECMA 1

```
setMinutes(minuteInt)
```

为Date对象实例的小时与日期设置分钟值。

返回值 使用毫秒数表示的新日期。

参数

minuteInt

一个从零开始的整数。

setMonth( )

IE 3 NN 2 Moz all Saf all Op 7 ECMA 1

```
setMonth(monthInt)
```

为Date对象实例的日期设置月份值。由于这个方法的值从零开始计数，因此脚本程序员在第一次使用时难免会感到困惑。

返回值 使用毫秒数表示的新日期。

参数

monthInt

一个从零开始的整数。一月为0，二月为1，而十二月为11。如果设置更大的值，会使得该对象进入下一年。

setSeconds( )

IE 3 NN 2 Moz all Saf all Op 7 ECMA 1

```
setSeconds(secInt)
```

为Date对象实例设置秒数。

返回值 使用毫秒数表示的新日期。

参数

secInt

一个从零开始的整数。

setTime( )

IE 3 NN 2 Moz all Saf all Op 7 ECMA 1

```
setTime(msInt)
```

使用从1970年1月1日开始计时的毫秒数来设定当前的Date对象。

返回值 使用毫秒数表示的新日期。

参数

msInt

毫秒整数。

setUTCDate( )

IE 4 NN 4 Moz all Saf all Op 7 ECMA 1

```
setUTCDate(dateInt)
```

根据浏览器中存储的UTC时间，设置Date对象实例所指定的时间在当月中的日期。如果指定的日期超出了对象当前月的末尾日期，那么该对象会在后续月份中重新设置日期。对于这类“setUTC”方法，Safari浏览器会面临夏时制问题，并且直到2.02版这一问题才得到修正。

返回值 使用毫秒数表示的新UTC日期。

参数

dateInt

整数值。

setUTCFullYear( )

IE 4 NN 4 Moz all Saf all Op 7 ECMA 1

1051

```
setUTCFullYear(yearInt)
```

根据浏览器中存储的UTC时间，为Date对象实例设置具体的年份数字。

返回值 使用毫秒数表示的新UTC日期。

参数

yearInt

整数值。Navigator 4不允许使用小于0的参数值。在IE及其NN 6中则可以使用负的年份值。

setUTCHours( )

IE 4 NN 4 Moz all Saf all Op 7 ECMA 1

```
setUTCHours(hourInt)
```

根据浏览器中存储的UTC时间，设置Date对象实例在当天的小时数。此时使用24小时计时系统。

返回值 使用毫秒数表示的新UTC日期。

参数

hourInt

一个从零开始的整数。

setUTCMilliseconds( )

IE 4 NN 4 Moz all Saf all Op 7 ECMA 1

```
setUTCMilliseconds(msInt)
```

根据浏览器中存储的UTC时间，为Date对象实例设置具体的毫秒数。

返回值 使用毫秒数表示的新UTC日期。

参数

msInt

一个从零开始的整数。

setUTCMinutes( )

IE 4 NN 4 Moz all Saf all Op 7 ECMA 1

```
setUTCMinutes(minuteInt)
```

根据浏览器中存储的UTC时间，为Date对象实例的小时和日期设置具体的分钟数。

返回值 使用毫秒数表示的新UTC日期。

参数

minuteInt

一个从零开始的整数。

1052

setUTCMonth( )

IE 4 NN 4 Moz all Saf all Op 7 ECMA 1

```
setUTCMonth(monthInt)
```

根据浏览器中存储的UTC时间，设置Date对象实例的月份值。由于这个方法的值从零开始计数，因此脚本程序员在第1次使用时难免会感到困惑。

返回值 使用毫秒数表示的新UTC日期。

参数

monthInt

一个从零开始的整数。一月为0，二月为1，而十二月为11。如果设置更大的值，会使得该对象进入下一年。

setUTCSeconds( )

IE 4 NN 4 Moz all Saf all Op 7 ECMA 1

```
setUTCSeconds(secInt)
```

根据浏览器中存储的UTC时间，为Date对象实例设置具体的秒数。

返回值 使用毫秒数表示的新UTC日期。

参数

secInt

一个从零开始的整数。

setYear( )

IE 3 NN 2 Moz all Saf all Op 7 ECMA n/a

```
setYear(yearInt)
```

设定Date对象实例的年份值。请使用setFullYear()来代替这个方法。需要注意的是，ECMA已经不再支持setYear()，但依然可以继续使用setFullYear()。

返回值 使用毫秒数表示的新日期。

参数

yearInt

表示年份的四位整数值，有时也使用两位数字来表示。

toDateString( )

IE 5.5 NN n/a Moz all Saf all Op 7 ECMA 3

返回一个包含Date对象实例中日期部分的字符串。具体的日期格式由浏览器及语言控制，但U.S.英语版本的现代浏览器总是以“*Ddd Mmm dd yyyy*”的形式返回日期字符串。

返回值 字符串。

参数 无。

1053

toGMTString( )

IE 3 NN 2 Moz all Saf all Op 7 ECMA 1

以标准格式返回Date对象实例GMT值的字符串版本。此方法并不会改变原有的Date对象。在新浏览器中，建议使用toUTCString()来代替toGMTString()。

返回值 如下格式的字符串：日期缩写, *dd mmm yyyy hh:mm:ss* GMT。例如：Mon 05 Aug 2002 02:33:22 GMT

参数 无。

toLocaleDateString( )

IE 5.5 NN n/a Moz all Saf all Op 7 ECMA 3

返回一个包含Date对象实例中日期部分的字符串。由浏览器及语言控制具体的日期格式。

返回值

存在多种适用的日期格式。例如，在U.S.英语版本的浏览器中就包括以下格式。

平台	字符串值	平台	字符串值
Internet Explorer 7	Sunday, April 01, 2007	Safari	April 1, 2007
Mozilla/Win	Sunday, April 01, 2007	Opera	4/1/2007
Mozilla/Mac	04/01/2007		

参数 无。

toLocaleString( )

IE *3* NN *2* Moz *all* Saf *all* Op *7* ECMA *1*

从Date对象实例中返回包含日期和时间的本地时区字符串。返回的数据格式会根据所在的国家或操作系统规范进行本地化操作。

返回值

存在多种适用的日期格式。例如，在U.S.英语版本的浏览器中就包括以下格式。

平台	字符串值
Internet Explorer 7	Sunday, April 01, 2007 10:30:00 AM
Mozilla/Win	Sunday, April 01, 2007 10:30:00 AM
Mozilla/Mac	Sun Apr 1 10:30:00 2007
Safari	April 1, 2007 10:30:00 AM PDT
Opera	4/1/2007 10:30:00 AM

参数 无。

toLocaleTimeString( )

IE *5.5* NN *n/a* Moz *all* Saf *all* Op *7* ECMA *3*

返回一个包含Date对象实例中时间部分的字符串。由浏览器及语言控制具体的日期格式。

返回值

存在多种适用的日期格式。例如，在U.S.英语版本的浏览器中就包括以下格式。

平台	字符串值。	平台	字符串值。
Internet Explorer 7	10:30:00 AM	Safari	10:30:00 AM PDT
Mozilla/Win	10:30:00 AM	Opera	10:30:00 AM
Mozilla/Mac	10:30:00		

参数 无。

toString( )

IE *4* NN *2* Moz *all* Saf *all* Op *7* ECMA *1*

当浏览器需要在对话框或屏幕上显示时间时，通常浏览器会自动调用这个方法来获得Date对象实例中日期数据字符串。

返回值

存在多种适用的日期格式。在U.S.英语版本的浏览器中就包括以下格式。

平台	字符串值。
Internet Explorer 7	Sun Apr 1 10:30:00 PDT 2007
Mozilla/Win	Sun Apr 01 2007 10:30:00 GMT-0700 (太平洋夏令时，PDT)
Mozilla/Mac	Sun Apr 01 2007 10:30:00 GMT-0700 (PDT)
Safari	Sun Apr 01 2007 10:30:00 GMT-0700
Opera	Sun, 01 Apr 2007 10:30:00 GMT-0700

参数 无。

toTimeString( )

IE 5.5 NN n/a Moz all Saf all Op 7 ECMA 3

1055

返回一个包含Date对象实例中时间部分的字符串。由浏览器及语言控制具体的日期格式。

返回值

平台	字符串值。
Internet Explorer 7	10:30:00 PDT
Mozilla/Win	10:30:00 GMT-0700 (太平洋夏令时, PDT)
Mozilla/Mac	10:30:00 GMT-0700 (PDT)
Safari	10:30:00 GMT-0700
Opera	10:30:00 GMT-0700

参数 无。

toUTCString( )

IE 4 NN 4 Moz all Saf all Op 7 ECMA 1

以标准格式返回Date对象实例UTC值的字符串。此方法并不会改变原有的Date对象。在新浏览器中，建议使用toUTCString()来代替toGMTString()。

返回值 如下格式的字符串：日期缩写, *dd mmm yyyy hh:mm:ss* GMT。例如：Mon 05 Aug 2002 02:33:22 GMT。

参数 无。

UTC( )

IE 3 NN 2 Moz all Saf all Op 7 ECMA 1

```
Date.UTC(yyyy, mm, dd[, hh[, mm[, ss[, msecs]]]])
```

这是Date对象的一个静态方法，它可返回保存在浏览器内部的Date对象的日期数字值。与Date对象构造函数所需的参数不同，UTC()方法的参数值必须是UTC时间才能保证返回值的正确性。这个方法并不会产生一个日期对象，这一功能应由Date对象构造函数完成。

返回值 整数，表示由参数传入的日期的UTC毫秒值。

参数

yyyy
四位数字的年份值。

mm
两位数的月份值（0~11）。

dd
两位数的日期值（0~31）。

hh
可选参数，使用24小时制的两位数的小时值（0~23）。

1056

mm
可选参数，两位数的分钟值（0~59）。

ss
可选参数，两位数的秒值（0~59）。

msec
可选参数，最后一个完整秒之后的毫秒数（0~999）。

valueOf( )

IE 4 NN 4 Moz all Saf all Op 7 ECMA 1

返回对象的值。

返回值 整数毫秒值。
参数 无。

Enumerator

IE 4(Win) NN n/a Moz n/a Saf n/a Op n/a ECMA n/a

如果一个ActiveX控件的属性或方法返回一个值集合，那么使用JavaScript对集合的常规处理方式（即视为数组）将无法处理这类数据。通过控制列表项的指针，Enumerator对象为JavaScript提供了一种有效途径来引用这类集合中各个数据项。更多详细信息请参考如下链接：*http://msdn.microsoft.com/en-us/library/6ch9zb09(vs.71).aspx*。

Enumerator创建方式 `var myEnumObj = new Enumerator(externalCollection);`
属性 无。
方法 `atEnd()`、`item()`、`moveFirst()`、`moveNext()`

atEnd( )

IE 4(Win) NN n/a Moz n/a Saf n/a Op n/a ECMA n/a

如果Enumerator指向集合中的最后一个数据项，那么就返回布尔值true。

返回值 布尔值：`true | false`。
参数 无。

item( )

IE 4(Win) NN n/a Moz n/a Saf n/a Op n/a ECMA n/a

返回集合中指针当前所指位置的对应值。

返回值 来自于集合的数字、字符串或其他值。
参数 无。

moveFirst( ), moveNext( )

IE 4(Win) NN n/a Moz n/a Saf n/a Op n/a ECMA n/a

在集合内调整指针所处的位置，`moveFirst()`会将指针指向集合中的第1项，而`moveNext()`则表示将指针向后移动一位。

返回值 无。
参数 无。

Error

IE 5(Win) NN n/a Moz all Saf all Op 7 ECMA 3

在脚本处理过程中，一旦发生了错误，那么已实现try-catch异常处理机制的浏览器就会自动创建一个Error对象。当然，也可以主动创建一个Error对象实例，并将它抛出。try-catch结构中的catch部分会以参数的形式收到Error对象实例，然后其中的脚本代码就可以通过Error的对象属性获取该错误的详细信息。

Error对象创建方式 `var myError = new Error("errorMessage");`
属性 `constructor`、`description`、`fileName`、`lineNumber`、`message`、`name`、`number`、`prototype`
方法 `toString()`

constructor

IE 5(Win) NN n/a Moz all Saf all Op 7 ECMA 3

可读/可写

它指向Error对象实例创建函数的一个引用，即浏览器中的`Error()`构造函数。

示例

```
if (myVar.constructor == Error) {
    // 处理本地字符串
}
```

值 函数对象引用。

description

IE *5(Win)* NN *n/a* Moz *n/a* Saf *n/a* Op *n/a* ECMA *n/a*

可读/可写

为错误提供明语描述，通常它与IE脚本错误提示对话框中出现的信息完全相同。但请尽可能使用更新的message属性。

示例

```
if (myError.description.indexOf("Object expected") != -1) {
// 处理 "Object expected" 错误
}
```

值 字符串。

fileName

IE *n/a* NN *n/a* Moz *all* Saf *n/a* Op *n/a* ECMA *n/a*

可读/可写

指明脚本错误发生时所在页面的URL地址。每当错误发生时，这个信息都会出现在Mozilla浏览器的JavaScript/错误控制台窗口之中。

示例

```
var sourceFile = myError.fileName;
```

值 URL字符串。

lineNumber

IE *n/a* NN *n/a* Moz *all* Saf *n/a* Op *n/a* ECMA *n/a*

可读/可写

指明当前脚本错误发生在源代码中的哪一行。每当错误发生时，这个信息都会出现在Mozilla浏览器的JavaScript/错误控制台窗口之中。

示例

```
var errorLine = myError.lineNumber;
```

值 字符串格式的数字值。

message

IE *5(Win)* NN *n/a* Moz *all* Saf *all* Op *7* ECMA *3*

可读/可写

为错误提供明语描述。目前并没有为这种描述信息规定标准的格式及内容。

示例

```
if (myError.message.indexOf("defined") != -1) {
    // 处理因未进行定义而引发的相关错误
}
```

值 字符串。

name

IE *5.5* NN *n/a* Moz *all* Saf *all* Op *7* ECMA *3*

可读/可写

指明当前错误类型的字符串。这个属性的默认值为Error。但浏览器也可能反馈其他类型，如EvalError、RangeError、ReferenceError、SyntaxError、TypeError和URIError等，如果浏览器可提供相关支持，

那么还可以报告一个特定的W3C DOM错误类型。

示例

```
if (myError.name == "SyntaxError") {
    // 处理语法错误
}
```

值 字符串。

number

IE *5(Win)* NN *n/a* Moz *n/a* Saf *n/a* Op *n/a* ECMA *n/a*

可读/可写

提供一个对应于IE错误的数字。如果要获得一个有意义的数字，那么必须在该数字上使用一个二进制位运算。此时请使用如下代码：

```
var errNum = ErrObj.number & 0xFFFF;
```

然后，再将结果值与Microsoft的数值列表进行比较，请参见如下链接：*http://www.027-8.com/news/javascript/jsmscRunTimeErrors.htm*。

示例 `var errNo = myError.number;`

值 数字值。

prototype

IE *5(Win)* NN *n/a* Moz *all* Saf *all* Op *7* ECMA *3*

可读/可写

这是静态`Error`对象的一个属性。可以使用这个属性为当前文档中将要创建的`Error`实例指定新的属性和方法。相关示例请参考`Array.prototype`属性说明。

示例 `Error.prototype.custom = true;`

值 任意数据，包括方法的引用。

stack

IE *n/a* NN *n/a* Moz *1.0.1* Saf *n/a* Op *n/a* ECMA *n/a*

只读

提供一个函数列表，有时还会提供一个事件，以指明错误发生的位置。如果错误可以追溯到一个函数，那么这个堆栈信息还会提供其所在的源代码行数。因此，这个属性是一个非常有用的调试工具。

值 多行字符串。

toString( )

IE *5(Win)* NN *n/a* Moz *all* Saf *all* Op *7* ECMA *3*

返回一个表示该对象的字符串，但在不同的浏览器中返回值各不相同。IE浏览器会返回`[object Error]`，而Mozilla浏览器则会返回由`name`及`message`属性组成的连续信息。

返回值 字符串。

参数 无。

Function

IE *3* NN *2* Moz *all* Saf *all* Op *7* ECMA *1*

函数是由一条或多条脚本指令组成的指令分组，在页面加载过程中及页面加载完成之后的任意时间都可以对其进行调用。调用函数时只需要在脚本指令中包含函数名及其参数列表，或者将它作为值赋给HTML标签中的事件处理属性。

1060 从第一款脚本浏览器诞生以来，都是在script元素中使用一个名称定义来创建一个函数，例如：

```
function funcName() {...}
```

现在的新浏览器还允许使用另外两种函数构造方式，请参见下文中的“函数创建方式”。通过这两种方式创建的函数称之为匿名函数，无法通过名称对它们进行引用。当属性赋值需要引用一个函数时，这种匿名函数比较有用，此时不会因函数的名称定义而使变量命名空间发生混乱。

函数还可以接收0个或多个参数。在定义函数时，可在函数名称后的括号对中定义多个由逗号分隔的参数变量，例如：

```
function doSomething(param1, param2, ... paramN) {...}
```

参数值可以是任意一种JavaScript数据类型，其中包括对象引用及数组。如果调用函数时提供的参数数目与函数定义中的参数数目并不相等，通常也不会带来什么问题。函数对象会将收到的所有参数都放入一个数组，即arguments，函数内的脚本指令代码可以通过它提取参数数据。

当函数的最后一行指令代码执行完毕时，它会将执行结果返回给调用指令。此时需要通过return指令来将结果值返回给调用指令。此外，无论return指令位于函数指令代码中的哪个位置，当执行到该指令时就会终止函数指令执行过程，并将控制权返回给调用代码，通常还会传回一个返回值。如果函数中条件结构的某个分支返回了一个值，那么其他分支，包括主分支都必须返回一个值，即使返回值为null也必须进行设定。尽管IE对于这一问题显得比较宽容，但为每个分支都添加一个返回值的确是一个良好的编程习惯。

在一个文档中，函数还可以访问那些定义在函数之外的全局变量。但通过var关键字定义在函数内的所有变量，则只能由函数内的指令进行访问。

如果要引用文档中定义的函数对象，那么请使用不包含参数的函数名称来完成这一个工作。例如，可以通过如下语法将一个函数指定给一个事件处理属性：

```
objReference.eventHandlerProperty = functionName;
```

从第4版浏览器开始，可以在一个函数内内嵌另一个函数，例如：

```
function myFuncA() {
    statements
        function myFuncB() {
    statements
    }
}
```

内嵌函数，即上文中的myFuncB，只能被临近的外层函数中的脚本指令调用。

此外，所有的函数都属于其所在的窗口。因此，如果脚本必须调用同辈框架中的一个函数，那么在引用时需要引入框架及函数名称，如下所示：

```
parent.otherFrame.someFunction()
```

1061 其他信息请参见本章下文中的return及yield。

Function对象创建方式

```
function myFunction([param1[, param2[,...paramN]]]) {
    statement(s)
}
var functionRef = function ([param1[, param2[,...paramN]]]) {
    statement(s)
};
var myFunction = new Function([param1[,...paramN], "statement1[; ...
statementN;"])objectRef.methodName = function([param1[, param2 [,...paramN]]]) {
    statement(s)
};
```

属性 arguments、arity、caller、constructor、length、prototype
方法 apply()、toString()、call()、valueOf()

arguments

IE 4 NN 3 Moz all Saf all Op 7 ECMA 1

只读

返回一个arguments对象，它拥有那些作为参数传递给函数的数据值。由于每个输入参数都会根据参数传入的顺序而拥有一个数字索引值，因此函数内的脚本指令可以通过数组语法来返回这些值。arguments数组的具体内容与函数定义中的参数变量彼此独立。因此，如果函数定义了两个参数变量，而调用指令传入了10个参数，那么arguments数组仍然会根据它们传入的顺序依次截获这10个值。函数内的脚本指令就可以检查arguments数组的长度并根据需要提取所需的参数值。这样一来，就可以允许函数处理数目不定的参数。

对大多数浏览器而言，可以简单地通过属性名称对此对象进行引用，如arguments[2]。不过某些旧浏览器还需要函数对象的名称。但所有的浏览器都可以识别后一种引用方式。

示例

```
function myFunc()
    for (var i = 0; i < myFunc.arguments.length; i++) {
        ...
    }
}
```

值 一个argument对象。

arity

IE n/a NN 4 Moz all Saf n/a Op n/a ECMA n/a

只读

返回一个整数值，以表示该函数定义的参数数量。可以在函数之外的指令代码中校验这个属性，以便为参数传递做准备。它的返回值与length属性相等。

示例 `var paramCount = myFunction.arity;`
值 整数值。

caller

IE 4 NN 3 Moz all Saf n/a Op n/a ECMA n/a

只读

返回一个函数对象的引用，该函数中的指令代码调用了当前函数。只有运行在函数中的脚本指令才能读取这个属性，以便引用其调用函数。1.0版之前的Mozilla浏览器会忽略这个属性，但后续版本已经恢复正常。

示例

```
function myFunc()
    if (myFunc.caller == someFuncZ) {
        // 当 someFuncZ 调用当此函数时进行处理
    }
}
```

值 Function对象。

constructor

IE 4 NN 4 Moz all Saf all Op 7 ECMA 1

可读/可写

它指向Function对象实例创建函数的一个引用，即浏览器中的Function()构造函数。

示例

```
if (myVar.constructor == Function) {
// 处理本地函数
}
```

值 函数对象引用。

length

IE 4 NN 4 Moz all Saf all Op 7 ECMA 1

只读

返回一个整数值，以表示该函数定义的参数数量。可以在函数之外的指令代码中校验这个属性，以便为参数传递做准备。

1063 **示例** `var paramCount = myFunction.length;`

值 整数值。

prototype

IE 4 NN 3 Moz all Saf all Op 7 ECMA 1

可读/可写

这是静态Function对象的一个属性。可以使用这个属性为当前文档中将要创建的函数实例指定新的属性和方法。相关示例请参考Array.prototype属性说明。

示例 `Function.prototype.author = "DG";`

值 任意数据，包括方法的引用。

apply( )

IE 5.5 NN n/a Moz all Saf all Op 7 ECMA 3

`apply([thisObjectRef[, argumentsArray]])`

调用当前函数，通常需要指定一个对象作为该函数中的this引用对象。函数所需的参数会作为apply()方法的第2个参数以数组的形式进行传递。此方法可以同匿名函数或已命名函数协同工作。尽管使用这个方法的机会很少，但如果需要获取一个函数引用，并且在一个对象环境中调用该函数，那么这个方法的确可以提供一定的灵活性。

首先，考虑下面这个脚本函数，此时会将它作为一个自定义对象的方法。

```
// 函数定义
function myFunc(parm1, parm2, parm3) {
    // 脚本指令
}
// 自定义对象的构造函数
function customObj(arg1, arg2) {
    this.property1 = arg1;
    this.property2 = arg2;
    this.method1 = myFunc;
}
var myObjA = new CustomObj(val1, val2);
var myObjB = new CustomObj(val3, val4);
```

执行myFunc()函数的最常用方式是将它作为对象的一个方法进行调用，例如：

```
myObjA.method1(parmValue);
```

但是也可以通过该函数的一个引用来调用这个函数，并且可以让该函数认为它的调用者也是一个对象，代码如下：

```
myFunc.apply(myObjB, [parmVal1, parmVal2, parmVal3]);
```

如果函数（如本例中的myFunc）中的某条指令包含this关键字，那么它就会成为apply()方法第1个参数传入的对象的一个引用，如本例中的myObjB。

返回值 无。

1064 **参数**

thisObjectRef

一个对象引用，以作为函数的调用环境。

argumentsArray

一个数组，其中的数据会作为参数值传递给函数。此时会按照数据项在数组中的排列顺序依次传递给函数。

call( )

IE *5.5* NN *n/a* Moz *all* Saf *all* Op *7* ECMA *3*

```
call([thisObjectRef[, arg1[, arg2,[...argN]]]])
```

调用当前函数，通常需要指定一个对象作为该函数中的this引用对象。可以将函数所需的参数使用逗号进行分隔并传递给call()方法的附加参数。除参数的组成形式不一样外，call()及apply()方法会完成相同的任务。请参见apply()方法以获取更多详细信息。

返回值 无。

参数

thisObjectRef

一个对象引用，以作为函数的调用环境。

arg1,...argN

一个以逗号分隔的参数列表，它将被传递到函数之中。

toString( )

IE *4* NN *4* Moz *all* Saf *all* Op *7* ECMA *1*

以字符串数据类型的形式返回对象值，即函数源代码。如果要在警告对话框或文档内显示对象内容，由于浏览器会自动将对象值转换为字符串，因此在实际使用中通常并不需要这个方法。

返回值 字符串。

参数 无。

valueOf( )

IE *4* NN *4* Moz *all* Saf *all* Op *7* ECMA *1*

返回对象的值。在显示该值时，浏览器会将它转换为字符串，但真实的值应该是Function对象的一个实例。

返回值 函数对象引用。

参数 无。

Global

IE *3* NN *2* Moz *all* Saf *all* Op *7* ECMA *1*

在所有能够运行JavaScript的浏览器内，每个窗口或框架中都存在着Global对象。虽然并不需要明确引用这个对象，但的确需要引用它的属性和方法来完成一些任务，如，通过其parseInt()或parseFloat()方法将字符串转换为数字。它的属性可视为常量，因此可以将它们作为一种评判准则。对于全局对象而言，它已经将其成员完全暴露给页面中的所有脚本指令。

属性 Infinity、NaN、undefined

方法 atob()、btoa()、decodeURI()、decodeURIComponent()、encodeURI()、encodeURIComponent()、escape()、eval()、GetObject()、isFinite()、isNaN()、isXMLName()、parseInt()、parseFloat()、ScriptEngine()、ScriptEngineBuildVersion()、ScriptEngineMajorVersion()、ScriptEngineMinorVersion()、unescape()、unwatch()、watch()

Infinity

IE *4* NN *4* Moz *all* Saf *all* Op *7* ECMA *1*

只读

提供一个正无穷大的数字，如果加上“-”运算符，则表示负无穷大。此时是从实际的角度而不是理论角度来探讨这个无穷大的。在JavaScript的世界中，所有小于Number.MIN_VALUE或大于Number.MAX_VALUE的值都是一个无穷大值。这种定义方式的确非常通俗。

示例 `var authorEgo = Infinity;`
值 `Infinity`

NaN

IE 4 NN 3 Moz all Saf all Op 7 ECMA 1

只读

非数字均可以使用这个值来表示。如果由于操作数的缺陷导致一个数字操作产生了一个非数字的结果，那么JavaScript就会返回这个值。如果要检查某个是否为一个数字，那么请使用`isNaN()`全局变量而不是直接与这个值进行比较。这个全局属性就是`Number.NaN`的估计值。

值 `NaN`

undefined

IE 5.5 NN n/a Moz all Saf all Op 7 ECMA 2

只读

尽管很早以前`undefined`这个数据类型就出现在ECMAScript和浏览器中，但直到不久之前它才成为`Global`对象的正式属性之一。你依然可以在旧浏览器中正常地使用这个数据类型，而不会引发兼容性冲突。

值 `undefined`

atob( ), btoa( )

IE n/a NN 4 Moz all Saf n/a Op n/a ECMA n/a

`atob("base64`*`EncodedData`*`"), btoa("`*`stringToBeEncoded`*`")`

这两个方法可以将任意字符串（包括含有二进制数据及Unicode值的字符串）转换为U.S.-ASCII字符集中的65字符子集。通过这种base64编码方案，就可以使用最基本的传输机制来传送任意数据。互联网工程任务组（Internet Engineering Task Force）制定的RFC1521阐明了这种编解码转换的基本原理与内部机制，相关信息请访问如下链接：*http://www.ietf.org/rfc/rfc2045.txt*。

请使用`btoa()`方法将字符串数据编码为base64模式的数据。编码后的结果数据将包含如下ASCII字符：a~z、A~Z、0~9，以及3个符号：/、+和=。编码后，可使用`atob()`方法将base64模式的编码数据解码为原始数据。

返回值 一个字符串。

参数

`base64`*`EncodedData`*

一个包含base64数据的字符串，可以在客户端上完成编码工作，也可以在服务器端完成编码后再将数据发送至客户端。

`stringToBeEncoded`

将被编码为base64模式的字符串，以便内部或外部使用。例如，可以将一个编码后的数据指定给`input`元素的`value`属性，并将它提交至能够接收base64数据的服务器处理进程。

decodeURI( )

IE 5.5 NN n/a Moz all Saf all Op 7 ECMA 3

`decodeURI("`*`encodedURI`*`")`

将参数传入的大多数URI编码值还原为初始值并返回。这个方法只能在那些使用`encodeURI()`方法进行编码后的字符上正常工作。

返回值 一个字符串。

参数

`encodedURI`

包含完整或相对URL地址编码的字符串。

decodeURIComponent( )

IE 5.5 NN n/a Moz all Saf all Op 7 ECMA 3


```
decodeURIComponent("encodedURIComponent")
```

将参数传入的所有URI编码值还原为初始值并返回。通常会在除协议之外的URI地址信息上使用这个方法。这个方法已经取代了被ECMA废止的unescape()函数。

返回值 一个字符串。

参数

encodedURIComponent

包含完整或相对URL地址编码或其中部分内容的字符串。

encodeURI( )

IE 5.5 NN n/a Moz all Saf all Op 7 ECMA 3

```
encodeURI("URIString")
```

对参数值中可进行URI编码的信息进行处理并返回编码后的转义字符串。在转换的过程中，此方法不会处理下列字符：

```
; / ? : @ & = + $ , #
```

这些字符是URI字符串中的有效符号，它们会保持原样，而且也不应该对它们进行转换，否则会导致URI失效。这个方法已经取代了被ECMA废止的escape()函数。

返回值 一个字符串。

参数

URIString

未进行编码的完整或相对URL地址字符串。

encodeURIComponent( )

IE 5.5 NN n/a Moz all Saf all Op 7 ECMA 3

```
encodeURIComponent("URIComponentString")
```

除拉丁字符集A~Z、a~z、数字0~9，以及某些符号（- _ . ! ~ * () '及空格）之外，此方法会对参数字符串中的所有其他字符进行编码，并返回其转义字符串。通常会在除协议之外的URI地址信息上使用这个方法。

返回值 一个字符串。

参数

URIComponentString

包含未进行编码的完整或相对URL地址或其中部分内容的字符串。

escape( )

IE 3 NN 2 Moz all Saf all Op 7 ECMA 1


```
escape("string"[, 1])
```

将函数参数传入的字符串进行编码并返回URL编码后的字符串。除* _ + - . /（IE中还包括@）之外，URL编码会将大多数非文字数字字符转换为十六进制值。例如空格会转换为%20。由于已使用加号来分隔搜索字符串中的各个组成部分，因此URL编码字符串并不会编码这个符号。如果在编码的过程中也要对加号进行转换，那么通过Navigator 4为此方法提供的第2个参数（数字1）就可以开启这种转换功能。需要注意的是，目前反对使用这个方法，而建议使用encodeURI()以及encodeURIComponent()方法，而且ECMA 3规范也已经删除了这个方法。

返回值 一个字符串。

参数

string

任何字符串。

eval( )

IE 3 NN 2 Moz all Saf all Op 7 ECMA 1

```
eval("string")
```

根据函数参数传入的参数字符串，返回它所描述的对象的一个引用。假设某个表单拥有一系列文本域，如entry1、entry2、entry3等，如果使用eval()函数将对象名称的字符串表达式转换成对象引用，那么就可以依据对象名称通过for循环遍历所有的表单项，代码如下：

```
for (var i = 1; i <=5; i++) {
 oneField = eval("document.forms[0].entry" + i);
 oneValue = oneField.value;
 ...
}
```

然而需要注意的是，eval()方法可以说是整个JavaScript语言中效率最低、性能最差的方法之一。仅拥有字符串ID或名称时，有很多更为有效的其他途径来引用文档树中的节点，例如document.getElementById()方法，而对于旧浏览器而言，还可以使用document.forms、document.images及document.formRef.elements数组的已命名索引等。

返回值 对象引用。

参数

string

表示一个对象引用的任意字符串。

GetObject( )

IE 5(Win) NN n/a Moz n/a Saf n/a Op n/a ECMA n/a

```
GetObject("localPathName"[, appName.objectType])
```

根据参数指明的路径名称，返回客户端机器上提供的一个ActiveX对象的引用。这实际上也为创建ActiveXObject实例提供了一种替代方案。除了指明控件的路径名之外，还可以指定一个数据文件以便和控件应用一同打开。在使用时，请在参数后附加一个感叹号（!）和文件名称，使它们作为*localPathName*参数的一部分。关于调用ActiveX对象（自动化对象）的更多信息，请访问如下链接：*http://msdn2.microsoft.com/en-us/library/7tf9xwsc.aspx*。

1069

返回值 对象引用。

参数

localPathName

包含自动化对象完整路径名（包括磁盘分卷）的字符串。

appName.objectType

引用一个特别的应用，以及自动化对象所支持的对象类型的通用语法，该自动化对象的路径由第1个参数指定。

isFinite( )

IE 4 NN 4 Moz all Saf all Op 7 ECMA 1

```
isFinite(expression)
```

如果作为参数传入的数值大于等于Number.MIN_VALUE并且小于等于Number.MAX_VALUE，那么就返回布尔值true。而参数输入的字符串值会使该函数返回false。

返回值 布尔值：true | false。

参数

expression

任何JavaScript表达式。

isNaN( )

IE 3 NN 2 Moz all Saf all Op 7 ECMA 1

```
isNaN(expression)
```

如果作为参数传入的表达式并非是一个数字值，那么就返回布尔值true。任何求值后为NaN的表达式，如parseInt("abc")，均会使此方法返回true。

返回值 布尔值：true | false。

参数

expression

任何JavaScript表达式。

isXMLName( )

IE n/a NN n/a Moz 1.8.1 Saf n/a Op n/a ECMA E4X

```
isXMLName(string)
```

如果作为参数传入的字符串是代表XML元素或属性的一个有效本地名称，那么此方法会返回布尔值true。

返回值 布尔值：true | false。

参数

string

任何JavaScript字符串。

parseInt( )

IE 3 NN 2 Moz all Saf all Op 7 ECMA 1

```
parseInt("string "[, radix])
```

根据参数传入的字符串中的数字，返回一个整数值，其数据类型既可以使用八进制也可以为十进制。字符串值必须以一个数字开头，否则将返回NaN。如果字符串以数字开头，但后续内容为文字或空格，那么就只会将位于头部的数字转换为整数。因此，可以使用如下的表达式从属性返回的长字符串中提取位于前部的整个版本号：

```
parseInt(navigator.appVersion)
```

而可选的参数“radix”可以指定向函数传递的基数。需要注意的是，如果字符串中的数字以0开头，那么通常会按照八进制数进行处理，这往往会导致一个错误的结果。因此如果只考虑十进制数的话，最好在所有的parseInt()函数中都应将第2个参数值指定为10。

返回值 整数值。

参数

string

由一个或多个数字、+或-开头的任意字符串。

radix

一个整数形式的基数，如2、8、10及16。

parseFloat( )

IE 3 NN 2 Moz all Saf all Op 7 ECMA 1

```
parseFloat(string)
```

根据参数传入的字符串中的数字，返回一个数字值，返回值既可以是整数也可以是一个浮点数。字符串值必须以一个数字开头，否则将返回NaN。如果字符串以数字开头，但后续内容为文字，那么就只会将位于头部的数字转换为整数。因此，可以使用如下的表达式从属性返回的长字符串中提取位于前部的完整版本号，如4.03：

```
parseFloat(navigator.appVersion)
```

如果转换值在小数点右侧不包含任何非零值，那么就返回一个整数。因此，只有当数字满足一定要求时才会返回浮点数。

返回值 数字值。

参数

1071

string

由一个或多个数字、+或-开头的任意字符串。

ScriptEngine( ), ScriptEngineBuildVersion( ), ScriptEngineMajorVersion( ), ScriptEngineMinorVersion( )

IE *4* NN *n/a* Moz *n/a* Saf *n/a* Op *n/a* ECMA *n/a*

这些仅适用于IE的函数会透露一些有关脚本引擎的信息，如调用函数的引擎名称（JScript、VBScript或VBA），以及已安装的引擎的版本号。对于Jscript而言，其版本号与浏览器中安装的*Jscript.dll*文件的版本有关。在版本号数字中，小数点左侧的数字表示主版本号，右侧的数字则表示副版本号。更为细致的版本信息被称为内部版本号，微软使用它在开发与发布的过程中追踪各个发布阶段。

返回值 `ScriptEngine()`可返回以下脚本引擎名称之一：`Jscript | VBA | VBScript`。其他函数则返回整数值。

参数 无。

unescape( )

IE *3* NN *2* Moz *all* Saf *all* Op *all* ECMA *1*

`unescape(string)`

将函数参数传入的URL编码字符串进行解码并返回解码后的字符串。除`* _ + - . /`（IE中还包括`@`）之外，URL编码会将非文字数字字符转换为十六进制值。例如空格会转换为`%20`。需要注意的是，目前反对使用这个方法，而建议使用`decodeURI()`以及`decodeURIComponent()`方法。而且ECMA 3规范也已经删除了这个方法。

返回值 字符串。

参数

string

任何URL编码字符串。

unwatch( ), watch( )

IE *n/a* NN *4* Moz *all* Saf *n/a* Op *n/a* ECMA *n/a*

`unwatch(property), watch(property, funcHandler)`

这两个Navigator和Mozilla的专用函数主要由JavaScript调试程序在内部进行调用。当脚本指令针对某个对象调用`watch()`函数时，参数值包括需要对其值进行监控的参数，以及通过赋值指令改变该属性值时所要调用函数的引用。如果要关闭监控操作，那么请针对之前监控的特定属性调用`unwatch()`函数。

返回值 无。

参数

1072

property

将要进行监控的对象属性名称。

funcHandler

当监控属性的值发生改变时，需要调用的函数的名称。

Iterator

IE *n/a* NN *n/a* Moz *1.8.1* Saf *n/a* Op *n/a* ECMA *n/a*

Iterator可以视为对象的一个只读属性阅读器，在它的帮助下，可以按照对象属性最初定义的顺序依次单步遍历对象的所有属性。JavaScript的for-in控制结构实质上使用了与之相同的技术，但通过创建一个自定义的迭代对象；还可以使用其next()方法来读取对象中的下一个属性。在两次调用next()之间，迭代对象实例会记住序列中的下一位是哪一项。当next()方法发现无更多属性可读时，它就会抛出一个StopIteration异常。

通过调用Iterator()构造函数，可以创建一个指定的迭代对象，此时需要将待迭代对象的引用作为参数传入此函数。第2个参数是一个可选的布尔值，当将它设置为true时，会取消next()方法返回的属性值。

Iterator对象创建方式 `var myIterator = new Iterator(object[, propertyNameOnlyFlag]);`

属性 constructor、prototype

方法 next()

constructor, prototype

IE *n/a* NN *n/a* Moz *1.8.1* Saf *n/a* Op *n/a* ECMA *n/a*

可读/可写

请参见Array对象描述中的对应属性。

next( )

IE *n/a* NN *n/a* Moz *1.8.1* Saf *n/a* Op *n/a* ECMA *n/a*

```
next()
```

返回迭代序列中的下一个属性。这个方法会返回一个属性/值对数组或一个属性名称字符串。读取该属性后，直到再次调用此方法之前，迭代器都会保持“指针”位置不变。如果读取最后一个属性之后继续调用此方法，就会抛出StopIteration异常。在实际使用中，既可以单独调用此方法，也可以在一个循环控制结构中进行调用，如while结构。

返回值 如果创建迭代器时同时使用了属性名称及属性值，那么就会返回一个由两个元素组成的数组，否则只会返回属性名称字符串。当它返回数组时，第1个元素为属性名称，第2个元素则为属性值，该值可以是任意一种数据类型。

参数 无。

Math

IE *3* NN *2* Moz *all* Saf *all* Op *7* ECMA *1*

只能以静态对象的形式使用Math对象，此时会将它作为数学常量及数学运算库。因此也不存在构造函数。Math对象的属性均为常量值，对象方法在对某个值进行数学运算后会返回一个数值，在调用方法过程中，并不会改变这个数值的原始值。

可通过如下语法来调用Math对象的属性或方法：

```
Math.propertyName
Math.method(param1[, param2])
```

在脚本代码中，一定要保证Math对象名称的“M”为大写字母。此外，所有调用Math对象的表达式都会计算出或返回一个值。

属性 E、LN10、LN2、LOG10E、LOG2E、PI、SQRT1_2、SQRT2

方法 abs()、acos()、asin()、atan()、atan2()、ceil()、cos()、exp()、floor()、log()、max()、min()、pow()、random()、round()、sin()、sqrt()、tan()

E

IE 3 NN 2 Moz all Saf all Op 7 ECMA 1

只读

返回欧拉常量。

示例 `var num = Math.E;`
值 2.718281828459045

LN2

IE 3 NN 2 Moz all Saf all Op 7 ECMA 1

只读

返回底为2的自然对数。

示例 `var num = Math.LN2;`
值 0.6931471805599453

LN10

IE 3 NN 2 Moz all Saf all Op 7 ECMA 1

只读

返回底为10的自然对数。

示例 `var num = Math.LN10;`
值 2.302585092994046

LOG2E

IE 3 NN 2 Moz all Saf all Op 7 ECMA 1

只读

返回以2为底的欧拉对数常量。

示例 `var num = Math.LOG2E;`
值 1.4426950408889634

LOG10E

IE 3 NN 2 Moz all Saf all Op 7 ECMA 1

只读

返回以10为底的欧拉对数常量。

示例 `var num = Math.LOG10E;`
值 0.4342944819032518

PI

IE 3 NN 2 Moz all Saf all Op 7 ECMA 1

只读

返回圆周率。

示例 `var num = Math.PI;`
值 3.141592653589793

SQRT1_2

IE 3 NN 2 Moz all Saf all Op 7 ECMA 1

只读

返回0.5的平方根。

示例 `var num = Math.SQRT1_2;`
值 0.7071067811865476

SQRT2

IE 3 NN 2 Moz all Saf all Op 7 ECMA 1

只读

返回2的平方根。

示例 `var num = Math.SQRT2;`
值 1.4142135623730951

abs( )

IE 3 NN 2 Moz all Saf all Op 7 ECMA 1

`abs(number)`

返回参数传入值的绝对值。

返回值 大于或等于0的数值。
参数
number
任意数值。

acos( )

IE 3 NN 2 Moz all Saf all Op 7 ECMA 1

`acos(number)`

返回参数传入值的反余弦值，单位为弧度。

返回值 数字值。
参数
number
从-1到1的任意数值。

asin( )

IE 3 NN 2 Moz all Saf all Op 7 ECMA 1

`asin(number)`

返回参数传入值的反正弦值，单位为弧度。

返回值 数字值。
参数
number
从-1到1的任意数值。

atan( )

IE 3 NN 2 Moz all Saf all Op 7 ECMA 1

`atan(number)`

返回参数传入值的反正切值，单位为弧度。

返回值 数字值。
参数
number
介于正负无穷之间的任意数值。

atan2( )

IE 4 NN 2 Moz all Saf all Op 7 ECMA 1

`atan2(x, y)`

返回一条直线与笛卡尔点x、y之间形成的弧度值。

返回值 介于-PI与PI之间的弧度值。

参数

x

任意数值。

y

任意数值。

ceil( )

IE 3 NN 2 Moz all Saf all Op 7 ECMA 1

```
ceil(number)
```

返回大于或等于函数参数，并且与之最接近的整数值。

返回值 整数值。

参数

number

任意数值。

cos( )

IE 3 NN 2 Moz all Saf all Op 7 ECMA 1

```
cos(number)
```

返回参数传入值的余弦值。

返回值 数字值。

参数

number

任意数值。

exp( )

IE 3 NN 2 Moz all Saf all Op 7 ECMA 1

```
exp(number)
```

根据参数传入的值number，返回欧拉常量number次幂的值。

返回值 数字值。

参数

number

任意数值。

floor( )

IE 3 NN 2 Moz all Saf all Op 7 ECMA 1

```
floor(number)
```

返回小于或等于函数参数，并且与之最接近的整数值。

返回值 整数值。

参数

number

任意数值。

log( )

IE 3 NN 2 Moz all Saf all Op 7 ECMA 1

```
log(number)
```

返回参数传入值的自然对数（即，底为欧拉常量）。

返回值 数字值。

参数

number

任意数值。

max( )

IE 3 NN 2 Moz all Saf all Op 7 ECMA 1

```
max(number1, number2)
```

返回两个参数中的较大值。

返回值 数字值。

参数

number1

任意数值。

number2

任意数值。

min( )

IE 3 NN 2 Moz all Saf all Op 7 ECMA 1

```
min(number1, number2)
```

返回两个参数中的较小值。

返回值 数字值。

参数

number1

任意数值。

number2

任意数值。

pow( )

IE 3 NN 2 Moz all Saf all Op 7 ECMA 1

```
pow(number1, number2)
```

以第1个参数值为底，第2个参数为幂，返回计算结果。

返回值 数字值。

参数

number1

任意数值。

number2

任意数值。

random( )

IE 3 NN 2 Moz all Saf all Op 7 ECMA 1

返回一个介于0到1之间的伪随机数。如果要获得介于0和另一个最大值之间的伪随机整数，可以使用下列公式：

```
Math.floor(Math.random() * (n+1))
```

此时*n*是可接受的数值范围中的最大整数值。如果要获得介于一个非零值和另一个最大值之间的伪随机整数，可以使用下列公式：

```
Math.floor(Math.random() * n - m + 1) + m
```

此时*m*是可接受范围内的最小整数，而*n*等于该范围中的最大整数值。需要注意的是，在Windows和Macintosh系统下的Navigator 2中，`Math.random()`方法无法正常工作。

返回值 0到1之间的数字，但不包括1。

参数 无。

round( )

IE 3 NN 2 Moz all Saf all Op 7 ECMA 1

```
round(number)
```

根据四舍五入规则返回一个整数值。如果传入的参数值大于或等于x.5，那么返回的值为x+1，否则返回值为x。

返回值 整数值。

参数

number

任意数值。

sin( )

IE 3 NN 2 Moz all Saf all Op 7 ECMA 1

```
sin(number)
```

返回参数传入值的正弦值，单位为弧度。

返回值 数字值。

参数

number

任意数值。

sqrt( )

IE 3 NN 2 Moz all Saf all Op 7 ECMA 1

```
sqrt(number)
```

返回参数传入值的平方根。

返回值 数字值。

参数

number

任意数值。

tan( )

IE 3 NN 2 Moz all Saf all Op 7 ECMA 1

```
tan(number)
```

返回参数传入值的正切值，单位为弧度。

返回值 Number

参数

number

介于正负无穷之间的任意数值。

Namespace

IE n/a NN n/a Moz 1.8.1 Saf n/a Op n/a ECMA E4X

Namespace的对象实例是一种将URI与元素本地名称前缀联系起来的对象。在这个URI所指向的位置中，包含着为该命名空间定义的一系列元素。命名空间对象实例拥有两个属性——prefix和uri，它们的值均为字符

串。在某些XML及XMLList对象的方法中，会将这个对象作为它们的参数值。

通过调用Namespace()构造函数可以创建一个命名空间对象，此时可以选用0至2个函数参数。在最常用的构造函数调用中，会使用两个参数字符串，第1个是前缀，而第2个为URI，如下所示：

```
var myNamespace = new Namespace("ns", "http://www.example.com/schema");
```

Namespace对象创建方式 `var myNamespace = new Namespace([prefix],[uri]);`
属性 constructor、prefix、prototype、uri
方法 toString()

constructor, prototype

IE *n/a* NN *n/a* Moz *1.8.1* Saf *n/a* Op *n/a* ECMA *n/a*

可读/可写

请参见Array对象描述中的对应属性。

prefix

IE *n/a* NN *n/a* Moz *1.8.1* Saf *n/a* Op *n/a* ECMA *n/a*

只读

包含命名空间前缀的字符串。如果字符串为空，就表示该命名空间是XML数据的默认命名空间。

示例 `var myPrefix = myNamespace.prefix;`
值 字符串。

uri

IE *n/a* NN *n/a* Moz *1.8.1* Saf *n/a* Op *n/a* ECMA *n/a*

只读

包含命名空间URI的字符串。

示例 `var myNSUri = myNamespace.uri;`
值 字符串。

toString( )

IE *n/a* NN *n/a* Moz *1.8.1* Saf *n/a* Op *n/a* ECMA *n/a*

以字符串数据类型的形式返回对象值。对于一个Namespace对象的实例而言，它会返回uri的属性值。

返回值 字符串。
参数 无。

Number

IE *4* NN *3* Moz *all* Saf *all* Op *7* ECMA *1*

Number对象可以表示任意的数字值，它既可以代表一个整数，也可以表示一个浮点数。由于在使用一个数值或将它赋值给一个变量时，浏览器会自动将它转换成一个Number对象实例，因此基本上并不需要过多地担心Number对象。从另一方面来说，访问此对象静态属性的可能性还更大一些，不过也只会在数学中才会用到这些属性。

Number对象创建方式

```
var myValue = number;
var myValue = new Number(number);
```

属性 constructor、MAX_VALUE、MIN_VALUE、NaN、NEGATIVE_INFINITY、POSITIVE_INFINITY、prototype、
方法 toExponential()、toFixed()、toLocaleString()、toPrecision()、toString()、valueOf()

constructor

IE 4 NN 4 Moz all Saf all Op 7 ECMA 1

可读/可写

它指向Number对象实例创建函数的一个引用，即浏览器中的Number()构造函数。

示例

```
if (myVar.constructor == Number) {
// 处理本地函数
}
```

值 函数对象引用。

MAX_VALUE

IE 4 NN 3 Moz all Saf all Op 7 ECMA 1

只读

等于JavaScript可以处理的最大值。

示例 `var tiptop = Number.MAX_VALUE;`

值 1.7976931348623157e+308

MIN_VALUE

IE 4 NN 3 Moz all Saf all Op 7 ECMA 1

只读

等于JavaScript可以处理的最小值。

示例 `var itsybitsy = Number.MIN_VALUE;`

值 5e-324

NaN

IE 4 NN 3 Moz all Saf all Op 7 ECMA 1

只读

非数字均可以使用这个值来表示。如果由于操作数的缺陷导致一个数字操作产生了一个非数字的结果，那么JavaScript就会返回这个值。如果要检查某个值是否为一个数字，那么请使用isNaN()全局变量而不是直接与这个值进行比较。

值 NaN

NEGATIVE_INFINITY, POSITIVE_INFINITY

IE 4 NN 3 Moz all Saf all Op 7 ECMA 1

只读

这两个值分别表示超出Number.MIN_VALUE及Number.MAX_VALUE限制范围的值。

示例 `Number.NEGATIVE_INFINITY`

值 `-Infinity; Infinity`

prototype

IE 4 NN 3 Moz all Saf all Op 7 ECMA 1

可读/可写

静态Number对象的一个属性。可以使用这个属性为当前文档中将要创建的Number实例指定新的属性和方法。相关示例请参考Array.prototype属性说明。通常并不需要为Number对象创建新的原型属性或方法。

示例 `Number.prototype.author = "DG";`

值 任意数据，包括方法的引用。

toExponential( )

IE 5.5 NN n/a Moz all Saf all Op 7 ECMA 3

```
toExponential(fractionDigits)
```

使用JavaScript指数表示法显示数字对象的值，并返回结果字符串。这个单独的参数指明了字符串中小数点右侧数字的具体位数。例如，如果一个变量包含数字9876.54，那么在调用toExponential(10)之后，得到的结果为“9.8765400000E+3”，此时使用0补齐最右侧的小数位，使得小数位总长度达到10。如果指定的小数位比原始数字的小数位少，那么就会对返回值进行四舍五入操作。

返回值 字符串。

参数

fractionDigits

一个整数，指明返回的字符串中小数点右侧数字的具体位数。

toFixed( )

IE 5.5 NN n/a Moz all Saf all Op 7 ECMA 3

```
toFixed(fractionDigits)
```

返回一个使用固定小数位数来显示数字对象值的字符串，在金融计算结果中这一方法比较有用。如果指定的小数位比原始数字的小数位少，那么就会对返回值进行四舍五入操作。但只会在最后一个待显示数字的下一位数字上进行四舍五入操作。

返回值 字符串。

参数

fractionDigits

一个整数，指明返回的字符串中小数点右侧数字的具体位数。

toLocaleString( )

IE 5(Mac)/5.5(Win) NN n/a Moz all Saf all Op 7 ECMA 3

返回数字对象值对应的字符串。ECMA标准并未对返回值的具体格式提出要求，而且随着币制的不同，其格式也会发生改变。在U.S.英语系统中，Windows平台下的IE 5.5及后续版本在四舍五入后会返回包含两位小数的字符串，并使用逗号标明千位、百万位等关键位。IE 5/Macintosh则不会在数字中进行标示。而Mozilla、Safari和Opera将不会执行任何特殊的格式化操作。

返回值 字符串。

参数 无。

toPrecision( )

IE 5.5 NN n/a Moz all Saf all Op 7 ECMA 3

```
toPrecision(precisionDigits)
```

返回一个使用固定小数位数来显示数字对象值的字符串。如果参数中指定的小数位比原始数字的小数位少，那么就会使用指数表示法来显示返回值。此时会对截取的值进行四舍五入，但只会在最后一个待显示数字的下一位数字上进行四舍五入操作。

返回值 字符串。

参数

precisionDigits

一个整数，指明返回的字符串中数字的具体位数。

toString( )

IE 4 NN 4 Moz all Saf all Op 7 ECMA 1

以字符串数据类型的形式返回对象值。如果要在警告对话框或文档内显示Number对象的值，由于浏览器会自

动将对象值转换为字符串，因此在实际使用中通常并不需要这个方法。

返回值 字符串。

参数 无。

valueOf( )

IE 4 NN 4 Moz all Saf all Op 7 ECMA 1

返回对象的值。

返回值 一个数字值。

参数 无。

Object

IE 4 NN 3 Moz all Saf all Op 7 ECMA 1

除了作为所有原生JavaScript对象的基础之外，Object对象还是JavaScript对象的抽象模型，其中包括自定义的脚本对象。可以使用Object对象在脚本中生成拥有自定义属性和方法的对象，从而使它具有自身的行为特征。首先通过构造函数创建一个空对象，然后为该对象的新属性指定属性值，这是本对象最具代表性的使用方法。

在Navigator 4、IE 5及后续版本，以及所有现代脚本浏览器中，还可以通过一种特殊的文字型语法指定属性及其属性值，在这一过程中还会创建一个Object对象实例，例如：

```
var myObject = {propName1:propValue1[, propName2:propValue2[,
...propNameN:propValueN]]}
```

在脚本中，针对结构化的自定义数据，可以将这些对象作为数据结构来使用，这与创建一个包含已命名索引值的数组非常相似。

Object对象创建方式

```
var myObject = new Object()
var myObject = {propName1:propVal1[, propName2:propVal2[,...N]]};
var myObject = new constructorFuncName([propVal1[, propVal2[,...N]]]);
```

属性 constructor、prototype

方法 hasOwnProperty()、isPrototypeOf()、propertyIsEnumerable()、toLocaleString()、toString()、valueOf()

constructor

IE 4 NN 4 Moz all Saf all Op 7 ECMA 1

可读/可写

它指向Object对象实例创建函数的一个引用，即浏览器中的Object()构造函数。

示例

```
if (myVar.constructor == Object) {
// 处理本地字符串
}
```

值 函数对象引用。

prototype

IE 4 NN 4 Moz all Saf all Op 7 ECMA 1

可读/可写

这是静态Object对象的一个属性。可以使用这个属性为当前文档中将要创建的Object对象实例指定新的属性和方法。相关示例请参考Array.prototype属性说明。

示例 `Object.prototype.author = "DG";`

值 任意数据，包括方法的引用。

hasOwnProperty( )

IE 5.5 NN n/a Moz all Saf 2.02 Op 7 ECMA 3

```
hasOwnProperty("propertyName")
```

在当前对象的实例创建之后，如果其构造函数（或文字赋值）中包含的属性能够与参数传入的属性名称相匹配，那么就返回布尔值true。但通过prototype为对象指定的属性并不会被认为是该对象所拥有的属性。

返回值 布尔值：true | false。

参数

propertyName

包含对象属性名称的字符串。

isPrototypeOf( )

IE 5.5 NN n/a Moz all Saf all Op 7 ECMA 3

```
isPrototypeOf(objectReference)
```

如果当前对象与参数传入对象的原型继承链中的某一点相同，那么就会返回true。需要注意的是，不同的浏览器实现可能会产生不同的结果。

返回值 布尔值：true | false。

参数

objectReference

一个对象引用，该对象可能与当前对象共享原型继承链。

propertyIsEnumerable( )

IE 5.5 NN n/a Moz all Saf n/a Op 7 ECMA 3

```
propertyIsEnumerable("propertyName")
```

如果参数传入的名称所对应的属性能够在对象中通过for-in属性进行检查，那么就会返回布尔值true。

返回值 布尔值：true | false。

参数

propertyName

包含对象属性名称的字符串。

toLocaleString( )

IE 5.5 NN n/a Moz all Saf n/a Op 7 ECMA 3

在决定如何对对象实例的字符串表达式进行本地化方面，浏览器具有充分的自由度。到目前为止，这个方法与toString()方法完全相同，并返回[object Object]。

返回值 字符串。

参数 无。

toString( )

IE 4 NN 4 Moz all Saf all Op 7 ECMA 1

以字符串形式返回对象值。在最新的浏览器中，这个值为[object Object]。

返回值 字符串。

参数 无。

valueOf( )

IE 4 NN 4 Moz all Saf all Op 7 ECMA 1

返回对象的值。

返回值 一个对象引用。

参数 无。

QName

IE n/a NN n/a Moz 1.8.1 Saf n/a Op n/a ECMA E4X

QName对象实例就是一个用于表示XML限定名称的对象。该XML名称会用于元素或元素属性。QName对象实例拥有两个属性——localName和uri，它们的值均为字符串。XML对象的name()方法会返回这种对象类型的返回值。

1087

通过调用QName( )构造函数可以创建一个QName对象，此时可以选用0到2个函数参数。在最常用的构造函数调用中，会使用两个参数字符串，第1个为Namespace对象引用（QName对象从该对象处继承其uri属性值），第2个则是用于本地名称的字符串，如下所示：

```
var myNamespace = new Namespace("ns", "http://www.example.com/schema");
var myQname = new QName(myNamespace, "widgetNumber");
```

QName对象创建方式

```
var myQname = new QName([namespaceObject],["localName"]);
```

属性 Constructor、localName、prototype、uri

方法 toString()

constructor, prototype

IE n/a NN n/a Moz 1.8.1 Saf n/a Op n/a ECMA n/a

可读/可写

请参见Array对象描述中的对应属性。

localName

IE n/a NN n/a Moz 1.8.1 Saf n/a Op n/a ECMA n/a

只读

限定名称中的本地名称字符串。

示例

```
var myName = myQname.localName;
```

值 字符串。

uri

IE n/a NN n/a Moz 1.8.1 Saf n/a Op n/a ECMA n/a

只读

包含命名空间URI的字符串。

示例

```
var myNSUri = myQname.uri;
```

值 字符串。

1088

toString( )

IE n/a NN n/a Moz 1.8.1 Saf n/a Op n/a ECMA n/a

以字符串数据类型的形式返回对象值。对于一个QName对象实例而言，这个方法返回的uri值之后还有一对冒

号（在命名空间与本地名称之间的JavaScript/E4X定界符）及本地名称。

返回值 “*uri::localName*”形式的字符串。

参数 无。

RegExp

IE *4* NN *4* Moz *all* Saf *all* Op *all* ECMA *3*

RegExp对象是一个静态对象，它既可以生成正则表达式的实例，也可以监控当前窗口或框架内的所有正则表达式。RegExp对象实例在下文中的正则表达式对象说明中进行介绍。

正则表达式可以协助脚本指令找到符合某些字符模式或特征的文本信息。例如，正则表达式可以快速判断出文本域中的某项数据是否是一个五位数的数字。制定匹配规则时需要掌握一种独立的标记语法，由于这一内容已超出了本书的范围，因此请参考其他相关书籍。不过在下文中还是对正则表达式语法进行了小结。

在某些浏览器中，RegExp对象的属性保存着正则表达式最后一次运算后的结果信息。因此，可以预见的是，正则表达式执行完毕后就会修改这些属性。这些运算包括正则表达式对象实例的各种方法，如exec()、test()等，还包括能够将正则表达式作为参数的String对象方法，如match()、replace()以及split()等。这些属性中的一部分还会传递给正则表达式对象，以便为正则表达式的下次运算做准备。

在支持正则表达式的浏览器中，所有的属性都拥有一个详细的名称和一个以$开头的简称。

属性 index input、lastIndex、lastMatch、lastParen、leftContext、multiline、prototype、rightContext、$1、$2、$3、$4、$5、$6、$7、$8、$9

index

IE *4* NN *n/a* Moz *n/a* Saf *n/a* Op *n/a* ECMA *n/a*

只读

这是一个从零开始的整数，它对应于一个字符所在位置的索引值，以表示最近一次搜索的起始位置。而lastIndex属性则提供了其终点位置。

示例

```
var srchStart = RegExp.index;
```

值 整数值。

input

IE *4* NN *4* Moz *all* Saf *n/a* Op *n/a* ECMA *n/a*

可读/可写

这是一个用来与规则表达式进行比较的主体字符串。如果这个主体字符串是作为方法的一个参数传递给正则表达式的，那么其值应该为null。其简写形式为$_，即小写的美元符号。但JavaScript 1.5已经放弃了这个属性。

示例

```
RegExp.input = "Four score and seven years ago...";
```

值 字符串。

lastIndex

IE *4* NN *n/a* Moz *n/a* Saf *n/a* Op *n/a* ECMA *n/a*

可读/可写

这是一个从零开始的索引值，将在该数值指定的字符位置开始执行下一次模式匹配搜索。在一个新搜索中，其值为0。如果要在不同的位置开始搜索，或须要略过某些字符，那么也可以手动设置这个值。目前大多数浏览器的正则表达式对象都支持这一属性。

示例 `myRE.lastIndex = 30;`

值 整数值。

lastMatch

IE *5(Mac)/5.5(Win)* NN *4* Moz *all* Saf *n/a* Op *n/a* ECMA *n/a*

只读

返回符合正则表达式的字符串，作为最近一次的运算结果。它可以简写为`$&`。JavaScript 1.5已经放弃了这个属性。

示例 `var matched = RegExp.lastMatch;`

值 字符串。

lastParen

IE *5(Mac)/5.5(Win)* NN *4* Moz *all* Saf *n/a* Op *n/a* ECMA *n/a*

只读

根据正则表达式中使用括号括起的最后一部分内容，返回与之匹配的字符串作为最近一次的执行结果。它可以简写为`$+`。JavaScript 1.5已经放弃了这个属性。

示例 `var myValue = RegExp.lastParen;`

值 字符串。

leftContext, rightContext

IE *5(Mac)/5.5(Win)* NN *4* Moz *all* Saf *n/a* Op *n/a* ECMA *n/a*

只读

`leftContext`属性返回最近获得的匹配结果之前的字符串内容，但其中并不包含结果字符串。`rightContext`属性则将返回位于匹配结果之后，直到主体字符串结束的字符串。它们可以分别简写为`$\``、`$'`。由于在同一个主体字符串上进行的并发搜索会朝着字符串末尾不断前进，因此`leftContext`的起始位置也会随之发生改变。JavaScript 1.5已经放弃了这两个属性。

示例

```
var wholeContext = RegExp.leftContext + RegExp.lastMatch + RegExp.rightContext;
```

值 字符串。

multiline

IE *n/a* NN *4* Moz *all* Saf *n/a* Op *n/a* ECMA *n/a*

可读/可写

尽管已在Navigator和Mozilla浏览器中实现了这个属性，但也只应该通过`RegExp`对象实例来读取此属性。请参见下文中的正则表达式对象。

prototype

IE *4* NN *4* Moz *all* Saf *all* Op *7* ECMA *3*

可读/可写

请参见Array对象中的同名属性。

$1, ..., $9

IE *4* NN *4* Moz *all* Saf *n/a* Op *n/a* ECMA *n/a*

只读

它们分别表示正则表达式的返回结果中位于括号内的子组件内容。这些结果分别保存在各个独立的属性中，并使用`$1~$9`这9个简写符号进行了标注。它们的具体顺序由子组件左括号的具体位置决定，位于最左侧的子

组件结果将保存在$1中。在使用正则表达式的String对象方法中，可以直接将这些属性作为方法的参数，请参见String.replace()方法。JavaScript 1.5已经放弃了这些属性。

示例 RegExp.$2

值 字符串。

regular expression

IE 4 NN 4 Moz all Saf all Op 7 ECMA 3

一个正则表达式对象就是RegExp对象的一个实例。每个正则表达式都包含一个规则，使用它就可以在一个字符串中找出所有的匹配内容。正则表达式的规则既可以是简单的字符串，也可以是功能更为强大的表达式，不过后者还需要借助一种自成一体的符号语言。正则表达式在JavaScript 1.2中的实现方式与Perl中的实现方式很类似。

在创建一个正则表达式对象时，首先要将判断规则放置在两个正斜线之间，然后再将整个表达式赋予某个变量。例如，下面这句指令创建了一个正则表达式，此时的判断规则仅仅是一个简单的单词：

```
var re = /greet/;
```

此后，所有需要使用这个规则来搜索字符串的方法都可以将re变量作为它的一个参数，当然，也可以将表达式直接作为方法参数，这样就不需要再将表达式赋值给某个变量。

正则表达式的标记符号还包括一些元字符（metacharacter），可以使用这些符号承担某些复杂的构想，如，单词两侧的边界、任意的数字及一个或多个字符等。例如，如果要使用上文中列出的规则来进行匹配搜索，当规则是一个单词而不是某个单词中的一部分（如greetings）时，就可以使用下面这个加入了元字符的正则表达式，它表明该规则应包括单词两侧的边界：

```
var re = /\bgreet\b/;
```

下表总结了JavaScript 1.2中使用的正则表达式标记符号。

字符	匹配	示例
\b	单词边距	/\bto/ 匹配 "tomorrow" /to\b/ 匹配 "Soweto" /\bto\b/ 匹配 "to"
\B	非单词边界	/\Bto/ 匹配 "stool"和"Soweto" /to\B/ 匹配 "stool"和"tomorrow" / \Bto\B/ 匹配 "stool"
\d	0~9的数字字符	/\d\d/ 匹配 "42"
\D	非数字	/\D\D/ 匹配 "to"
\s	单个空白单元	/under\sdog/ 匹配 "under dog"
\S	单个非空白单元	/under\Sdog/ 匹配 "under-dog"
\w	字母、数字或下划线	/1\w/ 匹配 "1A"
\W	非字母、数字或下划线	/1\W/ 匹配 "1%"
.	除换行符之外的任何字符	/../ 匹配 "Z3"
[...]	方括号内的任意一个字符	/J[aeiou]y/ 匹配 "Joy"
[^...]	不在字符集中的任意字符	/J[^eiou]y/ 匹配 "Jay"
*	重复前面的子表达式零次或多次	/\d*/ 匹配 ""、"5"或 "444"
?	重复前面的子表达式零次或一次	/\d?/ 匹配 ""或"5"
+	重复前面的子表达式一次或多次	/\d+/ 匹配 "5"或"444"
{n}	重复n次	/\d{2}/ 匹配 "55"

续表

字符	匹配	示例
{n,}	重复n次或更多次	/\d{2,}/ 匹配 "555"
{n,m}	重复至少n此，最多m次	/\d{2,4}/ 匹配 "5555"
^	一个字符串或文本行的起始位置	/^Sally/ 匹配 "Sally says..."
$	一个字符串或文本行的结束位置	/Sally.$/ 匹配 "Hi, Sally."

创建一个正则表达式时，可以让表达式在全局范围内均正常工作，并且忽略匹配内容的大小写形式。开启这些功能开关的修改符分别为字母g和i。对这两个修改符而言，既可以单独使用其中之一，也可以以gi的形式同时使用。此外，修改符m的含义请参见multiline属性。

一旦使用正则表达式标记符号确立了一个判断模式，那么正则表达式对象的相关方法，以及那些以这个正则表达式作为参数的String对象的方法就会执行相关搜索动作。

正则表达式对象的创建方式

```
var regExpressionObj = /pattern/ [g | i | m];
var regExpressionObj = new RegExp(["pattern", ["g" | "i" | "m"]]);
```

属性 constructor、global、ignoreCase、lastIndex、multiline、source

方法 compile()、exec()、test()

constructor

IE *4* NN *4* Moz *all* Saf *all* Op *7* ECMA *3*

可读/可写

请参见Array对象中的同名属性。

global, ignoreCase

IE *5(Mac)/5.5(Win)* NN *4* Moz *all* Saf *all* Op *7* ECMA *3*

只读

如果创建一个正则表达式对象实例时分别设置了g或i修改符，那么就返回布尔值true。如果正则表达式对象同时设置了这两个修改符（gi），那么就必须对每个属性进行单独测试。

示例

```
if (myRE.global && myRE.ignoreCase) {
    ...
}
```

值 布尔值：true | false。

lastIndex

IE *4* NN *4* Moz *all* Saf *all* Op *7* ECMA *3*

可读/可写

这是一个从零开始的索引值，它对应于字符串中下一次模式匹配搜索的起始字符位置。在一个新搜索中，其值为0。如果要在不同的位置开始搜索，或者需要略过某些字符，也可以手动设置这个值。

示例 myRE.lastIndex = 30;

值 整数值。

multiline

IE *5.5* NN *n/a* Moz *all* Saf *all* Op *7* ECMA *3*

可读/可写

在对象构造函数中是否出现“m”标志决定着RegExp对象multiline属性的值。换句话说，如果在脚本中明

确设定了“m”标志，那么RegExp对象multiline属性的值就将为true，例如：

```
var re = /you/gm;
```

执行完上述这句代码指令后，re.multiline的值就为true。

尽管这个属性的名称暗含着“多行”之意，但“m”标志对于将文本匹配搜索扩展到多行并没有什么影响。其实无论何时，所有的搜索都会在多行文本上进行。实质上只有当正则表达式包含^或$符号时，“m”标志才能发挥作用。以^符号为例，这个符号表示判断规则必须从字符的起始位置开始。请考虑下面这个多行字符串：

```
Are\n
you\n
happy?
```

由于在字符串中存在“you”，因此正则表达式/you/能够找到匹配的内容。但由于这个字符串并不是以“you”开头的，因此正则表达式/^you/无法找到匹配的字符串。但使用“m”标志后，会将多行字符串中的每个物理行均视为字符串的开始，因此/^you/m就可以找到一个匹配字符串。

但并不是所有的浏览器均能够识别这个标志，因此使用“m”标志的时候需要小心谨慎。在旧浏览器中使用“m”标志会导致脚本错误。

示例

```
if (re.multiline) {
    ...
}
```

值　布尔值。

source

IE 4　NN 4　Moz all　Saf all　Op 7　ECMA 3

只读

返回用于创建正则表达式的字符串。但这个值中并不包含表达式两端的斜线分界符。

示例

```
var myREasString = myRE.source;
```

值　字符串。

compile( )

IE 3　NN 2　Moz all　Saf all　Op 7　ECMA n/a

```
compile("pattern"[, "g" | "i" | "m"])
```

将一个正则表达式规则编译为一个真实的正则表达式对象。当脚本执行过程中某个正则表达式的规则可能会发生改变时，就会使用这个方法来编译正则表达式对象。

返回值　一个正则表达式对象实例的引用。

参数

pattern

引用字符串形式的任意正则表达式规则。此外，必须以独立参数的形式单独提供g、i、m等修改符。

exec( )

IE 4　NN 4　Moz all　Saf all　Op 7　ECMA 3

```
exec(string)
```

针对当前正则表达式规则，在参数传入的字符串上执行一次搜索操作。通常会采用下述流程执行这一过程：

```
var myRE = /somePattern/;
var resultArray = myRE.exec("someString");
```

静态RegExp对象和正则表达式对象实例的属性都会根据搜索结果进行更新。此外，exec()方法会返回一个数据数组，它与RegExp对象的属性非常相似。这个返回的数组包括以下属性：

index
: 从零开始的索引值，表示字符串中能够匹配判断规则的起始字符所在位置。

input
: 被搜索的原始字符串。

[0]
: 能够匹配判断规则的字符串。

[1]...[n]
: 其他规则部分的查询结果字符串。

可以将exec()方法的执行结果放置在一个变量之中，而RegExp的属性值在执行下一次运算时就会发生改变。如果将正则表达式设置为全局搜索，那么对myRE.exec("*someString*")进行连续调用时，会从上一次找到的匹配位置开始继续进行搜索。

如果调用exec()后未找到匹配信息，那么此方法就会返回null。

返回值 如果执行成功，返回匹配信息的数组；如果未找到匹配信息，则返回null。

参数

string
: 待搜索的字符串。

test( )

IE *4* NN *4* Moz *all* Saf *all* Op *7* ECMA *3*

```
test(string)
```

如果在参数传入的字符串中能够找到与正则表达式相匹配的内容，那么就返回布尔值true，否则返回false。对于搜索结果，此方法并未提供其他信息。与其他方法相比，这个方法能够最快速地判断出某个字符串中是否包含与正则表达式相匹配的信息。

返回值 布尔值：true | false。

参数

string
: 待搜索的字符串。

String

IE *3* NN *2* Moz *all* Saf *all* Op *7* ECMA *1*

String对象可以表示任何包含零个或多个字符的序列，可以将它们视为纯粹的文本。目前已有的大量方法主要可以分为两种。针对各种HTML字符排版，第1种方法会将字符串放置在一对HTML标签中。这些方法主要用于辅助某些使用document.write()来动态创建页面内容的脚本指令，但这一功能已经逐渐被样式单所取代。第2种重要的方法类别中由更为传统的字符串解析与处理方法组成，它们可用于字符与子字符串的搜索与复制、大小写转换，还可以将字符串列表转换成JavaScript数组。

总体来说，完全可以通过简单的赋值语句，从一个字符串值明确地创建出一个字符串对象：

```
var myString = "howdy";
```

但在某些情况下，可能需要使用静态String对象的构造函数来创建一个字符串对象。例如，将字符串值传递给Java小程序时，通常需要采用以下方式生成的字符串：

```
var myString = new String("howdy");
```

除构造函数、prototype属性及fromCharCode()方法之外，String对象实例可以使用其他的所有属性和方法，但静态String对象不行。

String对象创建方式

```
var myValue = "someString";
var myValue = new String("someString");
```

属性 constructor、length、prototype

方法 anchor()、big()、blink()、bold()、charAt()、charCodeAt()、concat() fixed()、fontcolor()、fontsize()、fromCharCode()、indexOf()、italics()、lastIndexOf()、link()、localeCompare()、match()、replace()、search()、slice()、small()、split()、strike()、sub()、substr()、substring()、sup()、toLocaleLowerCase()、toLocaleUpperCase()、toLowerCase()、toString()、toUpperCase()、valueOf()

constructor

IE 4 NN 4 Moz all Saf all Op 7 ECMA 1

可读/可写

它指向String对象实例创建函数的一个引用，即浏览器中的String()构造函数。

示例

```
if (myVar.constructor == String) {
    // 处理本地字符串
}
```

值 函数对象引用。

length

IE 3 NN 2 Moz all Saf all Op 7 ECMA 1

只读

提供字符串中字符的数量。如果为字符串指定了新值或将它与其他字符串合并，那么字符串就会动态改变其长度值。

示例

```
for (var i = 0; i < myString.length; i++) {
    ...
}
```

值 整数值。

prototype

IE 4 NN 3 Moz all Saf all Op 7 ECMA 1

可读/可写

这是静态String对象的一个属性。可以使用这个属性为当前文档中将要创建的String实例指定新的属性和方法。相关示例请参考Array.prototype属性说明。

示例 `String.prototype.author = "DG";`

值 任意数据，包括方法的引用。

anchor( )

IE *3* NN *2* Moz *all* Saf *all* Op *7* ECMA *n/a*

```
anchor("anchorName")
```

返回内嵌在锚链标签（<a>）中的字符串的一个副本。参数传入的值会指定给标签的name属性。

返回值 a元素中的字符串。

参数

anchorName

用于name属性值的字符串。

big( )

IE *3* NN *2* Moz *all* Saf *all* Op *7* ECMA *n/a*

返回内嵌在<big>标签中的字符串的一个副本。

返回值 big元素中的字符串。

参数 无。

blink( )

IE *3* NN *2* Moz *all* Saf *all* Op *7* ECMA *n/a*

返回内嵌在<blink>标签中的字符串的一个副本。

返回值 blink元素中的字符串。

参数 无。

bold( )

IE *3* NN *2* Moz *all* Saf *all* Op *7* ECMA *n/a*

返回内嵌在<b>标签中的字符串的一个副本。

返回值 b元素中的字符串。

参数 无。

charAt( )

IE *3* NN *2* Moz *all* Saf *all* Op *7* ECMA *1*

```
charAt(positionIndex)
```

根据参数传入的索引值，返回由对应位置上的字符形成的单字符字符串。如果只需要字符串中已知位置上的一个字符，那么就可以使用此方法替代substring()。

返回值

一个单字符字符串。在较新的浏览器版本中，如果参数值指定的字符位置超出了字符串的长度，那么就会返回一个空字符串。

参数

positionIndex

一个从零开始的整数。

charCodeAt( )

IE 4 NN 4 Moz all Saf all Op 7 ECMA 1

1098

```
charCodeAt(positionIndex)
```

根据参数传入的索引值，返回对应位置上字符的十进制Unicode编码数值。对于常用的字母数字字符而言，其Unicode值与ASCII值相对应。

返回值

正整数。如果参数值指定的字符位置超出了字符串的长度，那么就返回NaN。

参数

positionIndex

一个从零开始的整数。

concat( )

IE 4 NN 4 Moz all Saf all Op 7 ECMA n/a

```
concat(string2)
```

将参数传入的字符串附加到当前字符串对象，并返回结果字符串。这个方法与使用“+”或“+=”运算符来连接字符串的执行结果完全相同。这两种字符串连接方式都不会在两个字符串之间插入空格。

返回值 字符串。

参数

string2

任何字符串。

fixed( )

IE 3 NN 2 Moz all Saf all Op 7 ECMA n/a

返回内嵌在<tt>标签中的字符串的一个副本。

返回值 tt元素中的字符串。

参数 无。

fontcolor( )

IE 3 NN 2 Moz all Saf all Op 7 ECMA n/a

```
fontColor(color)
```

返回内嵌在字体标签（<font>）中的字符串的一个副本。参数传入的值会指定给标签的color属性。

返回值 font元素中的字符串。

参数

color

用于color属性值的字符串。

fontsize( )

IE 3 NN 2 Moz all Saf all Op 7 ECMA n/a

1099

```
fontSize(size)
```

返回内嵌在字体标签（<font>）中的字符串的一个副本。参数传入的值会指定给标签的size属性。

返回值 font元素中的字符串。

参数

size

用于size属性值的一个整数。

fromCharCode( )

IE 4 NN 4 Moz all Saf all Op 7 ECMA 1

```
String.fromCharCode(num1, [, num2,[...numN]])
```

这是一个静态方法，它会将参数传入的所有Unicode值进行转换，然后返回由这些字符组成的字符串。例如，下面这个表达式将返回“xyz”。

```
String.fromCharCode(120, 121, 122)
```

返回值 一个字符串。

参数

num1...numN

由一个或多个整数组成的列表，各整数间使用逗号进行分隔，同时也不要使用引号。

indexOf( )

IE 3 NN 2 Moz all Saf all Op 7 ECMA 1

```
indexOf(searchString[, startPositionIndex])
```

返回一个从零开始的整数值，以指明*searchString*参数字符串在当前字符串中的起始位置。通常会从第1个字符（索引值为0）开始进行搜索，但通过第2个参数可以在字符串中指定一个比较靠后的搜索起始位置。如果不存在相匹配的字符串，则返回-1。通过这个后向兼容方法可以快速查看一个字符串是否包含另一个字符串，即如果返回值为-1，就表示字符串中并未包含*searchString*。如果返回值是另外一个数字，无论具体的值为多少，都表示*searchString*位于当前字符中。在支持正则表达式的浏览器中，String对象的search()方法具有与之相同的功能。

返回值 整数值。

参数

searchString

需要在当前字符串对象中进行查找的字符串。

startPositionIndex

一个从零开始的整数，它指出了在当前字符串对象内对searchString进行搜索的起始位置。

italics( )

IE 3 NN 2 Moz all Saf all Op 7 ECMA n/a

返回内嵌在<i>标签中的字符串的一个副本。

返回值 i元素中的字符串。

参数 无。

lastIndexOf( )

IE 3 NN 2 Moz all Saf all Op 7 ECMA 1

```
lastIndexOf(searchString[, startPositionIndex])
```

返回一个从零开始的整数值，以指明*searchString*参数字符串在当前字符串对象中的起始位置。这个方法的工作方式与indexOf()方法比较类似，但它将从字符串末尾或某个索引位置开始进行查找。但即使从字符串的末尾开始进行搜索，*startPositionIndex*参数依然是根据字符串的起始位置计算索引值，此方法的返回值亦同此理。如果不存在相匹配的字符串，则返回-1。

返回值 整数值。

参数

searchString

需要在当前字符串对象中进行查找的字符串。

startPositionIndex

一个从零开始的整数，它指出了在当前字符串对象内对searchString进行搜索的起始位置。此外，即使从字符串的末尾开始进行搜索，依然是根据字符串起始位置来计算此参数值。

link( )

IE 3 NN 2 Moz all Saf all Op 7 ECMA n/a

```
link(URL)
```

返回内嵌在锚链标签（<a>）中的字符串的一个副本。参数传入的值会指定给标签的href属性。

返回值 a元素中的字符串。

参数

URL

用于href属性值的字符串。

localeCompare( )

IE 5.5 NN n/a Moz all Saf all Op 7 ECMA 3

```
localeCompare(string2)
```

根据浏览器及操作系统决定的字符串本地顺序，返回一个数字以指明当前字符串排列于参数字符串之前、之后还是与之相等。如果当前字符串位于参数字符串之前，将返回一个负数；如果它们相同，则返回值为0；如果当前字符串位于参数字符串之后，那么返回值就是一个正数。

此外，由于每个浏览器都能够决定适当的本地顺序，因此如果字符串中的字符超出了拉丁字符集的范围，那么应该谨慎使用这个方法。而不同的浏览器可能也会计算出不同的返回值。

返回值 整数值。

参数

string2

任何字符串。

match( )

IE 4 NN 4 Moz all Saf all Op 7 ECMA 3

```
match(regexpression)
```

如果在创建正则表达式时使用了“g”标志，那么就会根据参数传入的正则表达式，从当前字符串内找到能够与之相匹配的内容，并返回一个字符串数组。例如，如果将一个表示五位数的正则表达式传入此方法，那么它就会返回一个由主字符串中所有五位数组成的数组。而RegExp静态对象的属性也会受到这个方法的影响。

如果定义正则表达式时并未声明“g”标记符号，那么在返回的数组对象中，第1项（索引值为0）是匹配字符串，该对象的两个属性分别是index（指明匹配字符串在主字符串中的起始位置）和input（字符串的副本）。

返回值 一个字符串数组。

参数

regexpression

一个正则表达式对象。创建正则表达式对象的语法请参见本章前文。

replace( )

IE 4 NN 4 Moz all Saf all Op 7 ECMA 3

```
replace(regexpression, replaceString)
```

使用*replaceString*字符串替代所有与*regexpression*参数字符串相匹配的内容后，返回新字符串。在这一过程中，并不会改变原字符串，因此需要使用一个变量来获取返回值以便保存所做的改动。

返回值 一个字符串。

参数

regexpression

一个正则表达式对象。如果期望replace()方法能够对整个字符串执行操作，那么请在正则表达式中设置全局开关（g）。创建正则表达式对象的语法请参见本章前文。

replaceString

用于替代匹配内容的字符串。

search( )

IE 4 NN 4 Moz all Saf all Op 7 ECMA 3

```
search(regexpression)
```

返回一个从零开始的索引值，以指明当前字符串中能够匹配*regexpression*参数规则的第1个字符。这个方法与indexOf()比较类似，只是其搜索判断准则是一个正则表达式而不是一个单纯的字符串。

返回值 整数值。

参数

regexpression

一个正则表达式对象。创建正则表达式对象的语法请参见本章前文。

slice( )

IE 4 NN 4 Moz all Saf all Op 7 ECMA 3

```
slice(startPositionIndex, endPositionIndex])
```

返回当前字符串中的一个子串。根据第1个参数指定的索引值，此方法会从主字符串中对应的字符开始复制子串。第2个参数是一个可选参数，它指明了子串结束位置所在的索引值。这个值也可以是一个负数，这表明此时应从主字符串末尾向前计算索引量。如果调用此方法时未提供第2个参数，那么子串会一直延伸到主字符串末尾。

返回值 字符串。

参数

startPositionIndex

一个从零开始的整数，它指出了当前字符串对象中内容复制的起始位置。

endPositionIndex

一个从零开始的整数，它指出了当前字符串对象中内容复制的结束位置。负值表示从字符串末尾向前计数。

small( )

IE 3 NN 2 Moz all Saf all Op 7 ECMA n/a

返回内嵌在<small>标签中的字符串的一个副本。

返回值 small元素中的字符串。

参数 无。

split( )

IE4 *3* NN *3* Moz *all* Saf *all* Op *7* ECMA *1*

```
split(delimiter [, limitInteger])
```

返回一个新数组对象，该数组内元素由当前字符串的分段组成。根据第1个参数指定的分界字符串，当前字符串会在分界符所在位置被分割成为多个数组数据项。此时分界符并不会成为数组中的一部分。实际使用过程中，没必要为承载split()方法的执行结果而提前声明一个数组。例如，如果一个字符串包含一列由逗号分隔的名称，那么就可以使用如下指令将它转换为一个数组：

```
var listArray = stringList.split(",");
```

当然，可以使用一个正则表达式作为方法参数来切分字符串。

返回值 数组。

参数

delimiter

字符串或正则表达式，用以定义应该在哪些位置将主字符串切分成结果数组的元素。

limitInteger

一个可选的整数，用以限制结果数组中数据项的个数。

strike( )

IE *3* NN *2* Moz *all* Saf *all* Op *7* ECMA *n/a*

返回内嵌在<strike>标签中的字符串的一个副本。

返回值 strike元素中的字符串。

参数 无。

sub( )

IE *3* NN *2* Moz *all* Saf *all* Op *7* ECMA *n/a*

返回内嵌在<sub>标签中的字符串的一个副本。

返回值 sub元素中的字符串。

参数 无。

substr( )

IE *4* NN *4* Moz *all* Saf *all* Op *7* ECMA *n/a*

```
substr(startPositionIndex [, length])
```

返回一个从当前字符串中抽取的字符串。执行此方法时，会从第1个参数指定的字符索引位置开始抽取当前字符串中的内容。如果此时未提供其他参数，那么抽取的子串会一直延伸到主字符串末尾。第2个参数则指明了将要从主字符串中提取的字符数目。与这个方法不同的是，substring()方法中的第2个参数用以指明子串在主串中的结束位置。

返回值 一个字符串。

参数

startPositionIndex

一个从零开始的整数，它指出了当前字符串对象内复制的起始位置。

length

一个可选的整数参数，它指明了此次所要提取的字符的数量。

substring( )

IE 3 NN 2 Moz all Saf all Op 7 ECMA 1

```
substring(startPositionIndex, endPositionIndex)
```

返回一个从当前字符串中抽取的字符串。执行此方法时，会从第1个参数指定的字符索引位置开始抽取当前字符串中的内容，而结束位置则由第2参数指定的索引值决定。例如，`"Frobnitz".substring(0,4)`会返回索引值介于0到3之间的字符串，即`Frob`。而在与之相对应的`substr()`方法中，它的两个参数分别指定了文本提取的起始位置和所要提取的字符数，即字符串长度。

返回值 一个字符串。

参数

startPositionIndex

一个从零开始的整数，它指出了当前字符串对象内复制的起始位置。

endPositionIndex

一个从零开始的整数，它指出了当前字符串对象内复制的结束位置。换句话说，提取的字符串从*startPositionIndex*开始，直到*endPositionIndex*指定的位置结束，但不包括*endPositionIndex*所指定的那个字符。

sup( )

IE 3 NN 2 Moz all Saf all Op 7 ECMA n/a

返回内嵌在`<sup>`标签中的字符串的一个副本。

返回值 sup元素中的字符串。

参数 无。

toLocaleLowerCase( ), toLocaleUpperCase( )

IE 5.5 NN n/a Moz all Saf all Op 7 ECMA 3

这两个方法分别以全小写字母或全大写字母的形式返回当前字符串的副本。它们的工作方式与对应的常规方法（`toLowerCase()`、`toUpperCase()`）基本一致，但对于某些拥有字符映射的非拉丁字符集而言，可能还需要进行特定的内部处理。

返回值 字符串。

参数 无。

toLowerCase( ), toUpperCase( )

IE 3 NN 2 Moz all Saf all Op 7 ECMA 1

以全小写字母或全大写字母的形式返回当前字符串的副本。如果要使用调整后的字符串替换当前字符串，那么可以将此方法的返回值指定给原字符串变量：

```
myString = myString.toUpperCase();
```

通常，使用这两个方法之一就可以在不区分大小写的情况下对两个字符串进行比较。在将某个字符串与用户的输入内容进行比较时，由于用户输入的大小写情况不定，那么这种做法就十分方便，例如：

```
if (document.forms[0].entry.value.toLowerCase() == compareValue) {
    ...
}
```

返回值 字符串。

参数 无。

toString( ), valueOf( )

IE *4* NN *4* Moz *all* Saf *all* Op *7* ECMA *1*

返回当前对象的字符串值。

返回值 字符串。

参数 无。

VBArray

IE *4(Win)* NN *n/a* Moz *n/a* Saf *n/a* Op *n/a* ECMA *n/a*

VBArray对象可以让JavaScript与Visual Basic安全数组进行通讯。这种类型的数组均处于只读状态，它可以是多维数组，有时ActiveX控件也会将它作为返回值。JavaScript能够通过这个对象的一些方法来访问VBArray数据。更多详细信息请访问此链接：*http://msdn2.microsoft.com/en-us/library/3s0fw3t2.aspx*。

VBArray对象创建方式 `var myVBA = new VBArray(externalArray);`

属性 无。

方法 `dimensions()`、`getItem()`、`lbound()`、`toArray()`、`ubound()`

dimensions( )

IE *4(Win)* NN *n/a* Moz *n/a* Saf *n/a* Op *n/a* ECMA *n/a*

返回一个表示VBArray数组维数的整数值。

返回值 整数值。

参数 无。

getItem( )

IE *4(Win)* NN *n/a* Moz *n/a* Saf *n/a* Op *n/a* ECMA *n/a*

`getItem(dim1[, dim2[,...dimN]])`

返回VBArray中某个数据项的值。其参数指明了数组中目标数据所在的具体位置。

返回值 来自于VBArray的数字、字符串或其他值。

参数

dimN

表示数组内目标数据所在位置的整数值。对于多维VBArray数组，应使用由逗号分隔的若干数字来指定具体的位置。

lbound( ), ubound( )

IE *4(Win)* NN *n/a* Moz *n/a* Saf *n/a* Op *n/a* ECMA *n/a*

`lbound(dim) ubound(dim)`

针对一个特定维数的VBArray数组，这两个方法分别用于指定最低和最高索引值。

返回值 整数值。

参数

dim

表示数组内目标数据所在位置的整数值。

toArray( )

IE *4(Win)* NN *n/a* Moz *n/a* Saf *n/a* Op *n/a* ECMA *n/a*

返回一个由当前VBArray数组转换而成的JavaScript数组。

返回值 数组。

参数 无。

XML

IE *n/a* NN *n/a* Moz *1.8.1* Saf *n/a* Op *n/a* ECMA *E4X*

E4X标准中的XML对象实例就是一个XML数据容器。从一个空文本节点到一个复杂的XML元素，都可以归为XML数据。一个SOAP查询记录，或若干数据查询库结果组成的容器对象也可能属于XML数据。

目前有两种主要途径用以创建XML对象。第1种是通过传统的对象构造函数调用，例如：

```
var myXml = new XML();
```

如果未向构造函数传入任何参数，那么构造函数将返回一个空文本节点。当然，也可以传入一个XML文档要素的字符串作为参数。这类XML文档既可以是一个文本节点值，也可以是内嵌的XML元素，或者其他类型。

```
var myXml = new XML("<season><bowl><number>I</number><year>1967</year><winner>
Packers</winner><loser>Chiefs</loser></bowl></season>");
```

一旦获得了一个XML对象实例，就可以通过常用的JavaScript属性标记来获取元素的值。假定某个实例是数据的根节点，那么就可以使用点号（“.”）语法来遍历其层次结构，例如：

```
var bowlYear = myXml.bowl.year;
```

在上面这句代码中，即使从myXml对象中“提取”的的确是一个XML对象实例，并且拥有全部的实例方法，但从狭义范围来看，它也仅仅只是一个year元素而已。

当一个脚本引用指向的元素重复出现于XML数据中的同一层时，那么表达式就会产生一个XMLList对象，以包含该层中所有同名元素的XML对象实例。例如，如果超级杯（Super Bowl）数据在若干年内拥有多个bowl元素（请参见在线参考IV中列出的XML文件，此处的多个示例中均用了这个文件），那么下面这个表达式就会产生一个包含所有bowl元素的XML列表：

```
var allBowls = myXml.bowl;
```

因此，如果通过如下代码继续向下搜索，那么该表达式就会产生另一个XMLList对象，这个对象中包含着分布于bowl元素中的所有year元素：

```
var allYears = myXml.bowl.year;
```

当一个引用需要跨越多个包含层次时，E4X还为这种情况提供了一种快捷符号——后代存取器（..）。根据表达式中该符号两侧的设置，会对左侧指定的XML对象进行搜索，以便找到所有名称能够与右侧信息相匹配的元素实例，此时元素的内嵌深度并不会对结果产生影响。例如，如果要获取所有year元素的引用，可使用如下的后代存取器语法：

```
var allYears = myXml..year;
```

获得多个同名元素后，如果能够获知某个元素的子元素及其值，那么也可以从这些元素中找出对应的元素或类似元素。例如，假设一个XML对象包含多个bowl元素。如果要从中提取winner元素值为“Packers”的所有bowl元素，那么就应该使用如下表达式：

```
var thePack = myXml.bowl.(winner=="Packers");
```

这样就可以获得一个XML对象集合（XMLList），其中包含零个或多个符合指定条件的bowl元素。

使用一个标准的JavaScript赋值运算符（=）可以将一个新值插入到元素的文本节点之中，例如：

```
myXml.bowl.number = "II";
myXml.bowl.year = 1968;
```

与之类似，还可以通过赋值运算符添加元素、属性及XML数据值。例如，使用如下指令代码可以将一个winscore元素及其值添加到上述XML对象的bowl元素中：

```
myXml.bowl.winscore = 35;
```

得到的结果XML如下所示（已经过了美化排版）：

```
<season>
  <bowl>
    <number>I</number>
    <year>1967</year>
    <winner>Packers</winner>
    <loser>Chiefs</loser>
    <winscore>35</winscore>
  </bowl>
  ...
</season>
```


如果要引用一个元素的某个属性，请使用@前缀。例如，使用如下指令代码可以将一个元素加入number元素：

```
myXml.bowl.number.@type = "Roman Numeral";
```

那么得到的结果元素为：

```
<number type="Roman Numeral">I</number>
```

正如在创建数组与对象时可以赋予其文字内容一样，在创建XML对象时也可以设置其所属的文字信息。此时标签的左尖括号表示一个XML对象文本起点，而对应的结束标签则表示其终点。需要注意的是，不要在具体的文本值周围使用引号，但在右表达式中会内嵌引号以包围属性值。例如：

```
var myXml = <season>
  <bowl>
    <number type="Roman Numeral">I</number>
    <year>1967</year>
    <winner>Packers</winner>
    <loser>Chiefs</loser>
  </bowl>
</season>;
```

由于“text/javascript; e4x=1”类型的标记符号与常规的“text/javascript”类型略有不同，如果一个浏览器无法对它进行识别，就会引发后向兼容性问题。这样一来，浏览器就会将左尖括号错误地解释为一个运算符。

如果一个变量包含一个累加的字符串或一个计算得到的结果值，那么将这类JavaScript计算值置入一个XML对象时，就应该使用一对大括号（{}）来包围表达式，如下所示：

```
// 包含一些属性/值的全局对象示例
var teams = {_1967: {winner:"Packers", loser:"Chiefs"}, _1968: {...}, ...};
// 像元素值一样使用对象属性
var myXml = <season>
  <bowl>
    <number>I</number>
    <year>1967</year>
    <winner>{teams._1967.winner}</winner>
    <loser>{teams._1967.loser}</loser>
  </bowl>
```

```
</season>;
```

使用JavaScript语法还可以创建一个空元素，然后再使用子节点对它进行填充，代码如下：

```
var bowlXML = <bowl/>;
bowlXML.number = "I";
bowlXML.number.@type = "Roman Numeral";
bowlXML.year = 1967;
...
```

1109 当JavaScript的对象/属性赋值技术能够正常工作时，那么就存在很多XML对象方法可以用来控制XML对象内容，其中包括在指定位置插入或移除元素等。

如果要将一个XML对象实例转换成为一个字符串，以便上传至服务器，那么此时可以使用toString()方法。

下文所述的所有属性及defaultSettings()、setSettings()和settings()方法均为静态XML对象的成员，而所有其他方法均需要与对象实例一同使用。

XML对象创建方式

```
var myXml = new XML([XMLString]);
var myXml = XMLMarkup;
```

属性

```
constructor                 ignoreComments              ignoreProcessingInstructions
ignoreWhitespace            prettyIndent                prettyPrinting
prototype
```

方法

```
addNamespace()              appendChild()               attribute()
attributes()                child()                     childIndex()
children()                  comments()                  contains()
copy()                      defaultSettings()           descendants()
elements()                  hasComplexContent()         hasOwnProperty()
hasSimpleContent()          inScopeNamespaces()         insertChildAfter()
insertChildBefore()         length()                    localName()
name()                      namespace()                 namespaceDeclarations()
nodeKind()                  normalize()                 parent()
prependChild()              processingInstructions()    propertyIsEnumerable()
removeNamespace()           replace()                   setChildren()
setLocalName()              setName()                   setNamespace()
setSettings()               settings()                  text()
toString()                  toXMLString()               valueOf()
```

constructor, prototype

IE *n/a* NN *n/a* Moz *1.8.1* Saf *n/a* Op *n/a* ECMA *n/a*

可读/可写

请参见Array对象描述中的对应属性。

ignoreComments

IE *n/a* NN *n/a* Moz *1.8.1* Saf *n/a* Op *n/a* ECMA *E4X*

可读/可写

1110 命令构造函数不要创建注释节点，即使源代码中包含注释也不会进行创建。由于其默认值为true，因此如果希望在XML节点中引入注释节点，那么就应该关闭这个功能。

示例 `XML.ignoreComments = false;`

值 布尔值。

ignoreProcessingInstructions

IE *n/a* NN *n/a* Moz *1.8.1* Saf *n/a* Op *n/a* ECMA *E4X*

可读/可写

命令构造函数不要创建处理指令节点，即使源代码中包含这些内容也不会进行创建。由于其默认值为true，因此如果希望在XML节点中引入处理指令节点，那么就应该关闭这个功能。

示例 `XML.ignoreProcessingInstructions = false;`

值 布尔值。

ignoreWhitespace

IE *n/a* NN *n/a* Moz *1.8.1* Saf *n/a* Op *n/a* ECMA *E4X*

可读/可写

命令构造函数丢弃无关紧要的空白字符（空格、回车、换行及跳格等）。由于其默认值为true，因此如果希望在XML节点中引入这些空白内容，那么就应该关闭这个功能。

示例 `XML.ignoreWhitespace = false;`

值 布尔值。

prettyIndent

IE *n/a* NN *n/a* Moz *1.8.1* Saf *n/a* Op *n/a* ECMA *E4X*

可读/可写

将一个XML对象转换为一个字符串时，如果已经开启了XML.prettyPrinting属性，那么这个属性就可以控制浏览器用于元素缩进的空格数。其默认值为2。

示例 `XML.prettyIndent = 3;`

值 任何大于或等于0的整数。

prettyPrinting

IE *n/a* NN *n/a* Moz *1.8.1* Saf *n/a* Op *n/a* ECMA *E4X*

可读/可写

将一个XML对象通过toString()或toXMLString()方法转换为字符串时，这个属性可以控制是在输出内容中加入空白，还是仅输出一个连续字符串。具体的缩进量则由XML.prettyIndent属性控制。默认的输出样式是使用美化效果，即此属性值为true，并且每个层次均缩进两个空格。

示例 `XML.prettyPrinting = false;`

值 布尔值。

addNamespace( )

IE *n/a* NN *n/a* Moz *1.8.1* Saf *n/a* Op *n/a* ECMA *E4X*

`addNamespace(`*`namespaceObject`*`)`

向一个命名空间声明插入XML对象的根结点。相关信息请参见Namespace对象。

应用于 XML

返回值 已被修改的XML对象实例的引用。

示例

```
myXml.addNamespace(new Namespace("ex", "http://www.example.com/ns"));
```

参数

namespaceObject

一个命名空间对象实例的引用。

appendChild( )

IE *n/a* NN *n/a* Moz *1.8.1* Saf *n/a* Op *n/a* ECMA *E4X*

```
appendChild(childXMLObject)
```

在调用此方法的根节点中，在其他所有子节点之后插入一个XML对象（通常是一个元素或文本点）。

应用于 XML

返回值 已被修改的XML对象实例的引用。

示例

```
myXml.bowl.(year == 1995)[0].appendChild(<stadium>Joe Robbie Stadium</stadium>);
```

参数

childXMLObject

一个XML对象实例的引用。

attribute( )

IE *n/a* NN *n/a* Moz *1.8.1* Saf *n/a* Op *n/a* ECMA *E4X*

```
attribute("attributeName")
```

根据参数传入的字符串，找到属性能够与之相匹配的所有XML对象，并返回由它们组成的XMLList对象。

应用于 XML、XMLList

返回值 包含零个或多个XML对象的XMLList对象。

示例

```
var typeList = myXml.bowl.number.attribute("type");
```

参数

attributeName

需要在当前XML对象内查询的属性名称字符串。

attributes( )

IE *n/a* NN *n/a* Moz *1.8.1* Saf *n/a* Op *n/a* ECMA *E4X*

返回由当前XML对象的所有属性值组成的XMLList对象。

应用于 XML、XMLList

返回值 包含零个或多个XML对象的XMLList对象。

示例

```
var valueList = myXml.bowl.number.attributes();
```

参数 无。

child( )

IE *n/a* NN *n/a* Moz *1.8.1* Saf *n/a* Op *n/a* ECMA *E4X*

```
child("propertyName")
```

根据参数传入的字符串，返回由当前XML对象中所有名称与之相匹配的子元素组成的XMLList对象。如果要将XML记录中的数据值分配到一个HTML表格列中，那么通过这个方法就可以为该表格列找到所有待分配的值。

应用于 XML、XMLList

返回值 包含零个或多个XML对象的XMLList对象。

示例 `var yearsList = myXml.bowl.child("year");`

参数

propertyName

需要在当前XML对象内进行查询的子元素名称字符串。

childIndex( )

IE *n/a* NN *n/a* Moz *1.8.1* Saf *n/a* Op *n/a* ECMA *E4X*

返回一个从零开始计数的整数值，以指明当前XML对象在其父元素中的位置。

应用于 XML

返回值 任何大于或等于0的整数。

示例 `var where = myXml.employee.(id == "2f4");`

参数 无。

children( )

IE *n/a* NN *n/a* Moz *1.8.1* Saf *n/a* Op *n/a* ECMA *E4X*

返回由当前XML对象中所有子元素的引用组成的一个XMLList对象。这个XMLList对象中的数据项与当前对象中的直接子元素完全一致。但由于列表中的每一项都是一个XML对象引用，因此也可以进一步获取这些对象的内嵌子元素。

应用于 XML、XMLList

返回值 XMLList对象。

示例 `var kids = myXml.children();`

参数 无。

comments( )

IE *n/a* NN *n/a* Moz *1.8.1* Saf *n/a* Op *n/a* ECMA *E4X*

从当前XML对象的直接子元素中找出所有注释元素，并返回由这些元素的引用组成的一个XMLList对象。需要注意的是，如果使用脚本来创建XML对象实例，那么必须将`XML.ignoreComments`属性设置为`false`才能使注释成为XML对象的组成部分。

应用于 XML、XMLList

返回值 XMLList对象。

示例 `var topLevelComments = myXml.comments();`

参数 无。

contains( )

IE *n/a* NN *n/a* Moz *1.8.1* Saf *n/a* Op *n/a* ECMA *E4X*

`contains("value")`

如果当前XML对象（或XMLList中的对象）的值能够与参数字符串相匹配，那么就返回`true`。

应用于 XML、XMLList

返回值 布尔值。

示例 `if (myXml.bowl.winner.contains("Packers")) {...};`

参数

value

与当前对象的值进行比较的字符串。

copy( )

IE *n/a* NN *n/a* Moz *1.8.1* Saf *n/a* Op *n/a* ECMA *E4X*

返回当前XML对象或XMLList对象的一个副本。而对象中的所有内嵌元素也将包含在副本中。

应用于 XML、XMLList

返回值 根据当前的对象类型，返回XML或XMLList对象。

示例

```
var oneCopy = myXml.copy();
```

参数 无。

defaultSettings( )

IE *n/a* NN *n/a* Moz *1.8.1* Saf *n/a* Op *n/a* ECMA *E4X*

```
XML.defaultSettings()
```

返回一个JavaScript对象，它包含所有为静态XML对象提供的全局属性，如ignoreComments、ignoreProcessingInstructions、ignoreWhitespace、prettyIndent以及prettyPrinting等，并且将这些属性设置为默认值。如果将这个方法的返回结果作为setSettings()方法的一个参数，那么就可以通过一条指令将对象的所有属性恢复为默认值。

返回值 JavaScript对象。

示例

```
XML.setSettings(XML.defaultSettings());
```

参数 无。

descendants( )

IE *n/a* NN *n/a* Moz *1.8.1* Saf *n/a* Op *n/a* ECMA *E4X*

```
descendants(["elementName"])
```

返回一个XMLList对象，如果调用此方法时未设置参数，那么此对象中将包含当前对象的所有后代元素，否则仅包含那些名称与参数相匹配的部分元素。此外，descendants()方法还涉及内嵌的后代元素。在这个列表中，每个后代元素均会成为一个一级数据项，这意味对于一个复杂的XML数据集，返回的列表将非常大。

应用于 XML、XMLList

返回值 XMLList。

示例

```
var allDescendants = myXml.descendants();
```

参数

elementName

用于同当前元素中所有内嵌元素的名称进行比较的字符串。

elements( )

IE *n/a* NN *n/a* Moz *1.8.1* Saf *n/a* Op *n/a* ECMA *E4X*

```
elements("name")
```

返回一个XMLList对象，此对象中将包含当前对象的所有子元素（如果调用此方法时未设置参数），或者那些名称与参数相匹配的部分元素。此时返回列表中只会包含元素对象，而注释及处理指令都将会被忽略。

应用于 XML、XMLList

返回值 包含零个或多个XML对象的XMLList对象。

示例

```
var yearsList = myXml.bowl.elements("year");
```

参数

name

需要在当前XML对象内进行查询的子元素名称字符串。

hasComplexContent( )

IE *n/a* NN *n/a* Moz *1.8.1* Saf *n/a* Op *n/a* ECMA *E4X*

如果当前XML对象包含的元素还拥有子元素，那么就称它包含复杂内容，此时将返回true。

应用于 XML、XMLList

返回值 布尔值。

示例

```
if (myXml.employee[3].hasComplexContent()) {
    // 脚本指令
};
```

参数 无。

hasOwnProperty( )

IE *n/a* NN *n/a* Moz *1.8.1* Saf *n/a* Op *n/a* ECMA *E4X*

```
hasOwnProperty("propertyName")
```

如果在当前XML对象包含的元素中，能够找到名称与参数传入的字符串相匹配的元素，那么就返回true。在静态XML对象中也可以使用这个方法，此时可以用它来检查某些属性和方法是否存在，例如XML.hasOwnProperty("addNamespace")。

应用于 XML、XMLList

返回值 布尔值。

示例

```
if (myXml.hasOwnProperty("age")) {
    // 脚本指令
};
```

参数

propertyName

要在当前XML对象中进行查找的元素名称字符串，或静态XML对象的属性、方法名称。

hasSimpleContent( )

IE *n/a* NN *n/a* Moz *1.8.1* Saf *n/a* Op *n/a* ECMA *E4X*

当前XML对象所包含的元素中，如果某个元素并未包含子元素，或者该元素就是一个文本节点或元素节点，那么就返回true。值得注意的是，如果一个元素仅拥有一个文本节点，并且不包含其他子元素，那么就可以称它包含简单内容。

应用于 XML、XMLList

返回值 布尔值。

示例

```
if (myXml.employee[3].age.hasSimpleContent()) {
    // 脚本指令
};
```

参数 无。

inScopeNamespaces( )

IE *n/a* NN *n/a* Moz *1.8.1* Saf *n/a* Op *n/a* ECMA *E4X*

根据当前XML对象的父级元素返回一个由Namespace对象组成的数组。如果未明确声明任何命名空间，那么返回值将是一个空白数组。

应用于 XML

返回值 数组。

示例

```
var nsList = myXml.inScopeNamespaces();
```

参数 无。

insertChildAfter( ), insertChildBefore( )

IE *n/a* NN *n/a* Moz *1.8.1* Saf *n/a* Op *n/a* ECMA *E4X*

```
insertChildAfter(referenceChild, newChild)
insertChildBefore(referenceChild, newChild)
```

这两个方法可以将一个新子元素（第2个参数）插入到当前XML对象的指定元素（第1个参数）之前或之后，并返回更新后的XML对象的引用。如果第1个参数传入的值为null，那么insertChildAfter()会将新的子元素作为第1个子元素，而insertChildBefore()则会将它作为最后一个子元素。此方法会因此而修改当前对象，并且返回修改后的对象的引用。

应用于 XML

返回值 XML对象。

示例

```
myXml.bowl[0].insertChildBefore(myXml.bowl[0].winner[0], <stadium>Lambeau Field</
stadium>);
```

参数

referenceChild

当前对象中某个子元素的引用，它会作为插入的参考点。如果将它设置为null，那么就表示插入后在新的子元素之前或之后不存在任何其他子元素。

1117

newChild

一个将要被插入的XML对象。

length( )

IE *n/a* NN *n/a* Moz *1.8.1* Saf *n/a* Op *n/a* ECMA *E4X*

返回一个XML或XMLList对象实例所包含的子对象的具体数目。XML对象的返回值总是1，而XMLList对象则返回实际的子对象数目。

1116

应用于 XML、XMLList

返回值 整数值。

示例

```
var howMany = myXml.bowl.length();
```

参数 无。

localName( )

IE *n/a* NN *n/a* Moz *1.8.1* Saf *n/a* Op *n/a* ECMA *E4X*

返回XML对象限定名称中的本地名称部分。如果一个元素的名称中并不包含命名空间前缀，那么localName()和name()这两个方法的返回值就将完全相同。

应用于 XML

返回值 字符串。

示例

```
var myXml = <results xmlns:libBook="http://catalog.example.edu/schema">
    <libBook:title libBook:rarebooks="true">De Principia</libBook:title></results>;
var elemLocName = myX.children()[0].localName();
    // 结果为“title”。
```

参数 无。

name( )

IE *n/a* NN *n/a* Moz *1.8.1* Saf *n/a* Op *n/a* ECMA *E4X*

返回XML对象的整个限定名称。如果一个元素的名称中并不包含命名空间前缀，那么localName()和name()这两个方法的返回值就将完全相同。

应用于 XML

返回值 字符串。

示例

```
var myXml = <results xmlns:libBook="http://catalog.example.edu/schema">
    <libBook:title libBook:rarebooks="true">De Principia</libBook:title></results>;
var elemLocName = myX.children()[0].localName();
    // 结果为“title”。
```

参数 无。

namespace( )

IE *n/a* NN *n/a* Moz *1.8.1* Saf *n/a* Op *n/a* ECMA *E4X*

```
namespace(["prefix"])
```

返回一个Namespace对象，该对象的属性承担着为当前XML对象的命名空间指定的前缀与URI信息。调用此方法时也可以输入一个可选的前缀作为参数，这样一来，就只会返回一个与这个前缀相匹配的结果值。如果参数传入的前缀并不匹配，那么此方法将返回undefined。

应用于 XML

返回值 Namespace对象。

示例

```
var myXml = <results xmlns:libBook="http://catalog.example.edu/schema">
    <libBook:title libBook:rarebooks="true">De Principia</libBook:title></results>;
var elemLocName = myX.children()[0].namespace("libBook");
    // 结果为一个 Namespace 对象。
```

参数

prefix

一个限定名前缀字符串（不包括冒号）。

namespaceDeclarations( )

IE *n/a* NN *n/a* Moz *n/a* Saf *n/a* Op *n/a* ECMA *E4X*

返回一个Namespace对象数组。该数组中的每一项都是一个与当前XML对象相关的命名空间，其中还包括在其父级元素内定义的命名空间。

应用于 XML

返回值 数组。

参数 无。

nodeKind( )

IE *n/a* NN *n/a* Moz *1.8.1* Saf *n/a* Op *n/a* ECMA *E4X*

该属性会返回一个字符串，以指明当前XML对象所代表的DOM节点的具体类型。

应用于 XML

返回值

返回值可以是如下几个字符串之一：element、attribute、comment、processing-instruction及text。

参数 无。

normalize( )

IE *n/a* NN *n/a* Moz *1.8.1* Saf *n/a* Op *n/a* ECMA *E4X*

与同名DOM方法类似，此方法会将当前元素或XMLList中的相邻文本节点组合成一个独立的文本节点。值得注意的是，插入到一个元素中的各个文本节点都会将它们作为独立的节点，如果使用美化后的格式进行打印输出，那么元素中的每个文本节点都会占据一整行。

应用于 XML、XMLList

返回值

已对内容进行格式化的一个XML或XMLList值，具体的返回值类型由调用此方法的对象类型决定。

参数 无。

parent( )

IE *n/a* NN *n/a* Moz *1.8.1* Saf *n/a* Op *n/a* ECMA *E4X*

返回当前XML或XMLList对象的父级元素的一个引用。

应用于 XML、XMLList

返回值

一个XML或XMLList对象引用，具体的返回值类型由调用此方法的对象决定。如果当前对象没有父级元素，那么此方法会返回一个空对象。此外，如果当前对象是一个元素列表，即拥有多个父级元素，那么此方法将返回undefined。

示例

```
var parentElem = myXml.bowl.parent();
```

参数 无。

prependChild( )

IE *n/a* NN *n/a* Moz *1.8.1* Saf *n/a* Op *n/a* ECMA *E4X*

```
prependChild(newChild)
```

将一个XML对象作为第1个子元素插入到当前XML对象，然后再返回当前对象的一个引用。此时原始对象也

会因此而发生改变。

应用于 XML

返回值 XML对象。

示例

```
var txt = new XML("Super Bowl ");
myXml.bowl[20].number[0].prependChild(txt);
```

参数

newChild

一个XML对象实例。

processingInstructions( )

IE *n/a* NN *n/a* Moz *1.8.1* Saf *n/a* Op *n/a* ECMA *E4X*

```
processingInstructions(["name"])
```

如果调用此方法时未传入参数，那么它将返回由当前XML对象中所有处理指令节点组成的一个XMLList对象。当然，也可以将一个名称作为参数传入此方法，这样一来，就会返回由名称与之匹配的所有处理指令节点组成的一个XMLList对象。

应用于 XML、XMLList

返回值 XMLList对象。

参数

name

一个处理指令节点的名称字符串。

propertyIsEnumerable( )

IE *n/a* NN *n/a* Moz *1.8.1* Saf *n/a* Op *n/a* ECMA *E4X*

使用这个方法后，就可以通过脚本来控制是否将XML对象中的一个元素暴露给检查程序，如一个for-in结构。然而在第一版E4X中，并未赋予这个方法任何有效的操作。

应用于 XML、XMLList

返回值 布尔值。

removeNamespace( )

IE *n/a* NN *n/a* Moz *1.8.1* Saf *n/a* Op *n/a* ECMA *E4X*

```
removeNamespace(namespaceObject)
```

从当前XML对象中删除一个命名空间声明。然而值得注意的是，那些使用已删除的命名空间前缀的子元素及属性名称依然会让其前缀与命名空间保持关联。此方法执行结束后会返回修改后的XML对象的一个引用。

应用于 XML

返回值 XML对象。

示例

```
var myX = <results xmlns:libBook="http://catalog.example.edu/schema">
    <libBook:title libBook:rarebooks="true">De Principia</libBook:title></results>;
var ns = new Namespace("libBook","http://catalog.example.edu/schema");
myX.removeNamespace(ns);
```

```
// 返回的结果值 (toXMLString()) 为:
//   <results>
//      <libBook:title libBook:rarebooks="true">De Principia</libBook:title>
//   </results>
```

参数

namespaceObject

一个Namespace对象实例。

replace( )

IE *n/a* NN *n/a* Moz *1.8.1* Saf *n/a* Op *n/a* ECMA *E4X*

```
replace(elementNameStringOrNumber, newValue)
```

使用一个XML对象，包括XML或XMLList实例，来替代当前对象中的一个或多个子元素，第1个参数必须使用一个字符串或一个从零开始的整数来引用一个子元素或一组名称相同的元素，以说明被替代元素所在的位置。如果被替代元素本身就是其他元素的父级元素，那么在方法执行过程中所有的后代元素也会被同时删除。如果第1个参数指定的元素名称对应于多个同名元素，那么会在使用第2个参数对应的实例同时对它们进行替换。如果某个非空元素被一个空元素所取代，那么在处理过程中会丢失该元素的原有信息。此方法执行结束后会返回修改后的XML对象的一个引用。

应用于 XML

返回值 XML对象。

示例

```
// 使用一个空 track 元素替换所有的 track 元素
myXml.mix.replace("track", <track/>);

// 使用 anno 元素替代所有的 year 元素，但保持其值不变
for each (var elem in myXml.bowl) {
    elem.replace("year", <anno>{elem.year.toString()}</anno>);
}
```

参数

elementNameStringOrNumber

既可以是一个子元素名称的字符串，也可以是一个用于指定当前XML对象中某个子元素的索引值。

newValue

一个XML或XMLList实例，其中包括文本节点。

setChildren( )

IE *n/a* NN *n/a* Moz *1.8.1* Saf *n/a* Op *n/a* ECMA *E4X*

```
setChildren(newValue)
```

使用一个XML对象，包括XML或XMLList实例，来替代当前对象中的所有子元素，此方法执行结束后会返回修改后的XML对象的一个引用。

应用于 XML

返回值 XML对象。

示例

```
// 大规模裁员
myXml.employees.setChildren(<vacancy/>);
```

参数

newValue

一个XML或XMLList实例，其中包括文本节点。

setLocalName( )

IE *n/a* NN *n/a* Moz *1.8.1* Saf *n/a* Op *n/a* ECMA *E4X*

```
setLocalName("newName")
```

使用参数传入的新值来替代当前XML对象中的本地名称部分。而原有的前缀并不会发生改变。

应用于 XML

返回值 无。

示例

```
var myX = <results xmlns:libBook="http://catalog.example.edu/schema">
    <libBook:title libBook:rarebooks="true">De Principia</libBook:title></results>;
var ns = new Namespace("libBook","http://catalog.example.edu/schema");
myX.ns::title.setLocalName("bookTitle");
// 元素会变为:
//    <libBook:bookTitle libBook:rarebooks="true">De Principia</libBook:bookTitle>
```

参数

newName

元素的新本地名称。

setName( )

IE *n/a* NN *n/a* Moz *1.8.1* Saf *n/a* Op *n/a* ECMA *E4X*

```
setName("newName")
```

使用参数传入的新值来替代当前XML对象的名称。如果元素原有的名称中包含一个命名空间前缀，那么该前缀也会被删除。

应用于 XML

返回值 无。

示例

```
var myX = <results xmlns:libBook="http://catalog.example.edu/schema">
    <libBook:title libBook:rarebooks="true">De Principia</libBook:title></results>;
var ns = new Namespace("libBook","http://catalog.example.edu/schema");
myX.ns::title.setName("bookTitle");
// 元素会变为:
//     <bookTitle libBook:rarebooks="true">De Principia</bookTitle>
```

参数

newName

元素的新名称。

setNamespace( )

IE *n/a* NN *n/a* Moz *1.8.1* Saf *n/a* Op *n/a* ECMA *E4X*

```
setNamespace(namespaceObject)
```

使用参数传入的新值来替代当前XML对象的命名空间。如果要为一个尚未声明命名空间的元素或属性指定命名空间，那么请使用addNamespace()方法。

应用于 XML

返回值 无。

1123

1122

示例

```
var myX = <results xmlns:libBook="http://catalog.example.edu/schema">
    <libBook:title libBook:rarebooks="true">De Principia</libBook:title></results>;
var ns = new Namespace("libBook","http://catalog.example.edu/schema");
var updatedNS = new Namespace("dg", "http://www.dannyg.com/ns");
myX.ns::title.setNamespace(updatedNS);
// 元素会改变为:
//    <results>
//      <dg:title xmlns:dg="http://www.dannyg.com/ns" libBook:rarebooks=
"true">De Principia</libBook:title>
//    </results>
```

参数

namespaceObject

一个Namespace对象实例。

setSettings( )

IE *n/a* NN *n/a* Moz *1.8.1* Saf *n/a* Op *n/a* ECMA *E4X*

```
XML.setSettings([settingsObject])
```

使用这个方法就可以通过一条指令来控制静态XML对象的所有全局属性。如果不设置参数，或者将参数设置为null或undefined，那么就会让所有的属性设定都恢复到默认值。否则请传入一个对象，并且保证它所拥有的5个属性与静态XML对象的5个全局属性完全相同，即，ignoreComments、ignoreProcessingInstructions、ignoreWhitespace、prettyIndent和prettyPrinting。

返回值 无。

参数

settingsObject

此时可以使用一个JavaScript对象，它的5个属性及属性值如下所示：

```
{ignoreComments: Boolean, ignoreProcessingInstructions: Boolean,
ignoreWhitespace: Boolean, prettyPrinting: Boolean, prettyIndent: Integer}
```

settings( )

IE *n/a* NN *n/a* Moz *1.8.1* Saf *n/a* Op *n/a* ECMA *E4X*

```
XML.settings()
```

返回一个JavaScript对象，它包含所有为静态XML对象提供的全局属性和属性值，如ignoreComments、ignoreProcessingInstructions、ignoreWhitespace、prettyIndent以及prettyPrinting等。在临时改变一个或多个属性前，可以使用这个方法来保存目前的设定。然后，使用保存下的值作为setSettings()方法的一个参数，这样就可以通过一条指令将对象的所有属性恢复为初始值。

返回值 JavaScript对象。

参数 无。

text( )

IE *n/a* NN *n/a* Moz *1.8.1* Saf *n/a* Op *n/a* ECMA *E4X*

返回由当前XML对象内所有子文本节点组成的XMLList对象。如果当前XML对象表示一个独立元素，那么返回的XMLList对象也只拥有一个数据项，即表示文本节点的一个XML对象。通过这一方法提供的XMLList对象，可以获得所有同类子节点的文本节点。例如，在线参考IV提供的超级杯XML数据中存在很多bowl元素，每个元素均包含一个拥有文本节点的year元素。如果要获取一个由数据中所有year元素的文本节点组成的XMLList对象，就可以使用如下的表达式：

```
myXml.bowl.year.text()
```

通常还可以依靠隐式类型转换获得XMLList对象中每个元素的字符串值，以便将来在XML数据之外使用脚本来操作这些元素值。如果上述方法遇到了问题，那么请在各个元素项上使用toString()方法来获得其字符串值。

应用于 XML、XMLList

返回值 Xmllist object.

参数 无。

toString( )

IE *n/a* NN *n/a* Moz *1.8.1* Saf *n/a* Op *n/a* ECMA *E4X*

对于一个使用XML对象来表示的元素而言，这个方法可以返回其文本节点或属性节点的字符串值。如果在一个XML对象实例或一个XMLList对象中仅包含一项数据，那么返回的字符串通常是可以直接使用的数据。但是，如果当前元素是一个由多个元素组成的XMLList对象，那么所有的值就会被压缩到一处，而且在各个元素值之间也不存在任何分界符。

当一个表达式需要一个字符串值，而脚本代码提供的是一个XML对象时，由于JavaScript会自动执行隐式类型转换，因此在大多数情况下都不需要调用toString()方法。

应用于 XML、XMLList

返回值 字符串。

参数 无。

toXMLString( )

IE *n/a* NN *n/a* Moz *1.8.1* Saf *n/a* Op *n/a* ECMA *E4X*

返回当前XML对象的字符串值，其中可以包括数据源代码中的所有标签、属性及命名空间声明。这个方法可以作用于简单的XML对象，也可以在XMLList对象上使用。此外，在这个方法生成的输出内容中，会自动采用美化输出方式。

应用于 XML、XMLList

返回值 字符串。

参数 无。

valueOf( )

IE *n/a* NN *n/a* Moz *1.8.1* Saf *n/a* Op *n/a* ECMA *E4X*

返回当前XML对象的一个引用。

应用于 XML、XMLList

返回值 XML对象。

参数 无。

XMLList

IE *n/a* NN *n/a* Moz *1.8.1* Saf *n/a* Op *n/a* ECMA *E4X*

E4X标准中的XMLList对象的一个实例实际上就是零个或多个XML对象的容器。例如，假设XML数据包含一个外部元素和一个重复出现的内嵌元素，如使用一个根元素<order>来表示客户订购信息，其中的内嵌<item>元素则表示定单中的每个产品。根结点（order）的引用只会返回一个元素。而对于重复的数据项元

素（order.item）则会返回一个XMLList对象，而且该对象中的数据个数与XML数据中的item元素个数完全相等。此外，列表中的每个元素可能还拥有附加的子节点。

通过JavaScript的数组符号，可以引用XMLList对象中的任意一个数据项，如下所示：

```
order.item[3]
```

但XMLList和JavaScript数组之间也只有这一个相似之处。E4X提供了一种特有的控制指令——for-each-in，使用它可以简便地遍历XMLList对象中的所有对象。

XML和XMLList对象之间的差异有时很模糊，E4X显然是有意而为之。实际上XMLList对象实例的方法就是XML对象实例的方法的一个子集。请参见XML对象说明中列出的所有方法，并且找到“应用于”标题，以便确认那些能够在XMLList对象实例中使用的所有方法。

XMLlist对象创建方式

```
var myXmlList = new XMLList([XMLList]);
```

属性 无。

方法 attribute()、attributes()、child()、children()、comments()、contains()、copy()、descendants()、elements()、hasComplexContent()、hasOwnProperty()、hasSimpleContent()、length()、normalize()、parent()、processingInstructions()、propertyIsEnumerable()、text()、toString()、toXMLString()、valueOf()

运算符（Operators）

+

IE 3 NN 2 Moz all Saf all Op 7 ECMA 1

加法运算符可以与数字及字符串协同工作，但它的执行结果会随着运算对象的数据类型而发生改变。当两端的运算对象都是数字时，结果为两个数字之和；当两端均为字符串时，结果为这两个字符串合并后的字符串；当一侧为数字而另一侧为字符串时，会首先将数字数据类型转换为一个字符串，然后再将两个字符串进行合并。如果要将一个字符串运算对象转换为一个数字，请使用本章前文中提到的parseInt()或parseFloat()方法。

在一个E4X环境中，加法运算符还可以将XML片段进行整合，并生成XML或XMLList对象。

示例

```
var mySum = number1 + number2;
var newString = "string1" + "string2";
var newXML = <element1>textA</element1> + <element2>textB</element2>; // 下式仅适用于E4X
```

+=

IE 3 NN 2 Moz all Saf all Op 7 ECMA 1

这是一个增值（add-by-value）运算符。这类运算符将一个常规的赋值运算符（=）与另外一个其他类型的运算符联合在一起，以便通过在左、右操作数上进行指定的操作来完成赋值。例如，如果将一个字符串存储在一个名为a的变量中，那么就可以通过“+=”运算符向它附加一个字符串：

```
a += " and some more.";
```

如果不采用这种方法，那么常规表达式的结构如下：

```
a = a + " and some more";
```

下表列出了所有与之类似的赋值运算符。

运算符	示例	等价表达式
+=	a += b	a = a + b
-=	a -= b	a = a - b
*=	a *= b	a = a * b
/=	a /= b	a = a / b
%=	a %= b	a = a % b
<<=	a <<= b	a = a << b
>>=	a >>= b	a = a >> b
>>>=	a >>>= b	a = a >>> b
&=	a &= b	a = a & b
|=	a |= b	a = a | b
^=	a ^= b	a = a ^ b

在E4X环境中，这种增值运算符还可以用于向一个已存在的XML对象附加XML元素。

示例

```
output += "<H1>Section 2</H1>";
total *= .95;
```

&&

IE 3 NN 2 Moz all Saf all Op 7 ECMA 1

逻辑与运算符用于比较两个布尔表达式以获得其真实性。如果两个表达式的值均为true，那么&&运算符的执行结果也为true，如果两个表达式中的一个或两个值为false，那么其运算结果就为false。

此时使用的布尔表达式既可以由一个比较表达式（使用其他比较运算符）组成，也可以是其他各种数据值。下表列出了常用数据类型、值及等价的布尔值。

数据类型	等价的布尔值
除 0 之外的数字	true
0	false
任意非空字符串	true
空字符串	false
任意对象	true
null	false
undefined	false

使用上表列出的信息，就可以在&&运算符的帮助下创建复合条件。例如，如果要判断某人是否在表单域中输入了大于100的数字，那么该判断条件应与下文类似：

```
var userEntry = document.forms[0].entry.value ;
if (userEntry && parseInt(userEntry) >= 100) {
    ...
}
```


如果用户未输入任何值，那么userEntry就将是一个空字符串。在复合条件中，当第1个操作数为false时，根据&&运算符规则，整个表达式会返回false。由于将从左到右计算复合条件表达式，因此第1个操作数的false值会让该条件立即返回false，这意味着此时并不会对第2个操作数进行判断。

示例

```
if (a <= b && b >= c) {
    ...
}
```

=

IE 3 NN 2 Moz all Saf all Op 7 ECMA 1

赋值运算符会将右侧操作数的计算值赋予运算符左侧的变量。完成该操作后，变量中的数据类型仍然与其初始值的数据类型相同。使用赋值运算符时还可以采用一种链式结构，此时会从整个指令的右侧开始进行赋值，一直到左侧结束。因此，执行下列表达式后，3个变量的值均等于25。

```
a = b = c = 25;
```

对于其他值类型而言，运算符右侧还可以是JavaScript符号，如数组（方括号，[]）、对象（大括号，{}），以及E4X中的XML数据（尖括号，<>）。

示例

```
var myName = "Theodore Roosevelt";
var now = new Date();
```

&

IE 3 NN 2 Moz all Saf all Op 7 ECMA 1

按位与运算符会在两个操作数的二进制值之上执行二进制数学运算。此时会在这两个数的每个对应位上执行一个布尔与操作。如果同一列中的两个操作数均为1，那么位于该列的结果值就为1。其他数值组合则将产生0。这个运算符最终得到的是一个与二进制结果等价的十进制值。例如，3和6的二进制分别为0011和0110。对这两个值进行与操作后，二进制的结果为0010，而与之等价的十进制值则为2。

示例

```
var n = 3 & 6;
```

<<

IE 3 NN 2 Moz all Saf all Op 7 ECMA 1

按位左移运算符会将第1个操作数按照第2个操作数指定的列数向左进行移动。例如，如果将二进制数0011（十进制为3）向左移动2位，那么其二进制结果为1100，即十进制数12。

示例

```
var shifted = 3 << 2;
```

~

IE 3 NN 2 Moz all Saf all Op 7 ECMA 1

这是按位非运算符。这个一元运算符会将一个二进制数的每一位进行反转。例如，假设有一个二进制数0110（即十进制数6）。在执行完按位非操作后，其二进制结果为1001，此时已经将所有的0转换为1。而对应的十进制值是一个负数（-5）。

示例

```
var n = ~6;
```

|

IE 3 NN 2 Moz all Saf all Op 7 ECMA 1

按位或运算符会在两个操作数的二进制值之上执行二进制数学运算。此时会在这两个数的每个对应位上执行一个布尔或操作。如果同一列中的两个操作数均为0，那么位于该列的结果值就为0。其他数值组合则将产生1。这个运算符最终得到的是一个与二进制结果等价的十进制值。例如，3和6的二进制分别为0011和0110。对这两个值进行或操作后，二进制的结果为0111，而与之等价的十进制值则为7。

示例

```
var n = 3 | 6;
```

>>

IE 3 NN 2 Moz all Saf all Op 7 ECMA 1

按位右移运算符会将第1个操作数按照第2个操作数指定的列数向右进行移动。例如，如果将二进制数0110（十进制为6）向右移动2位，那么其二进制结果为0001，即十进制数1。此时所有溢出其右边界的数位均会被丢弃。

示例

```
var shifted = 6 >> 2;
```

^

IE 3 NN 2 Moz all Saf all Op 7 ECMA 1

按位异或运算（XOR）符会在两个操作数的二进制值之上执行二进制数学运算。此时会在这两个数的每个对应位上执行一个布尔异或操作。如果同一列中的两个操作数中只有一个1，那么位于该列的结果值就为1。其他数值组合则将产生0。这个运算符最终得到的是一个与二进制结果等价的十进制值。例如，3和6的二进制分别为0011和0110。对这两个值进行异或操作后，二进制的结果为0101，与之等价的十进制值则为5。

示例

```
var n = 3 ^ 6;
```

>>>

IE 3 NN 2 Moz all Saf all Op 7 ECMA 1

这是按位零填充右移运算符。这个运算符会将第1个操作数按照第2个操作数指定的列数向右进行移动。使用按位右移运算符（>>）时，从左侧填充进的新数位为1，而在按位零填充右移中，在左侧填充的数字是0。此时所有溢出其右边界的数位均会被丢弃。Microsoft也将这个运算符称为无符号右移运算符。

示例

```
var shifted = 6 >>> 2;
```

,

IE 3 NN 2 Moz all Saf all Op 7 ECMA 1

逗号运算符可以分隔位于同一行脚本代码中的各个表达式。它可以被用于很多场合。例如，如果要声明多个变量，那么就可以使用如下语法：

```
var varName1, varName2, ... varNameN;
```

由于多个脚本指令也可以放置在同一行中，因此，下面这行脚本代码表示依次调用两个警告对话框（当用户关闭第1个对话框后才会出现第2个）：

```
alert("Howdy"), alert("Doody");
```

另外一个应用场景是for循环，如果需要在循环中同时调用两个或多个变量，那么就可以借助于逗号运算符：

```
for (var i = 0, var j = 2; i < 20; i++, j++) {
    ...
}
```

示例

```
var isCSS, isIEMac;
```

?:

IE 3 NN 2 Moz all Saf all Op 7 ECMA 1

条件运算符为if-else控制结构提供了一种简洁的语法格式。在部署这个运算符时，需要包括3个组成部分：一个条件及两个指令。如果条件判断为true，那么就会执行第1条指令，否则执行第2条指令。其语法如下所示：

```
condition ? statement1 : statement2
```

也可以嵌套使用这些运算符，从而在一条指令中添加多个判断路径。在下面的语法中，如果*conditionA*为false，那么才会继续判断*conditionB*，并且根据*conditionB*的结果返回*statement2*或*statement3*的值作为整个表达式的值：

```
conditionA ? statement1 : (conditionB ? statement2 : statement3)
```

这个操作也仅仅只是看上去比较简洁而已。它与一个if-else结构调用的内部处理流程完全相同。

示例

```
var newColor = (temp > 100) ? "red" : "blue";
```

--

IE 3 NN 2 Moz all Saf all Op 7 ECMA 1

自减运算符会将一个变量表达式的当前值减1。将这个运算符放置在变量之前或之后表达的含义有所不同。当运算符位于变量之前时，那么变量会首先减1，然后才参与当前指令的计算。例如，在如下指令序列中：

```
var a, b;
a = 5;
b = --a;
```


在将a赋值给b时，首先会将a减1。因此，当这些指令运行完毕时，a和b的值均为4。与之相反的是，在如下指令序列中：

```
var a, b;
a = 5;
b = a--;
```

首先会将a赋值给b，然后再将a减小。当这些指令执行完毕时，a等于4而b的值则为5.

这一特性影响着for循环计数变量的定义与使用方式。通常，在循环运行指令后，通过一个循环计数器从最大值开始向后递减计数。因此，大多数循环计数器都会将此运算符放置在计数变量之后：

```
for (var i = 10; i >=0; i--) {...}
```

示例

```
--n
n--
```

/

IE 3 NN 2 Moz all Saf all Op 7 ECMA 1

除法运算符会使用右侧值除运算符左侧的值。此时左右两个操作数都必须是数字，而使用这个运算符的表达式将计算出一个数字。

示例

```
var myQuotient = number1 / number2;
```

==

IE 3 NN 2 Moz all Saf all Op 7 ECMA 1

等于运算符会对两个操作数进行比较并返回一个布尔结果。当script元素所指定的JavaScript版本不同时，这个运算符的具体行为也存在着一定的差异。如果将language属性设置为JavaScript或JavaScript 1.1，那么某些操作数会自动进行转换，如下表所示：

左操作数	右操作数	描述
对象引用	对象引用	比较两个对象引用的等价性
任意数据类型	布尔变量	将布尔操作数转换成一个数字（1对应于true，0对应与false），并且与另一个操作数进行比较

续表

左操作数	右操作数	描述
对象引用	字符串	通过 toString() 将对象转换为字符串，再对这两个字符串进行比较
字符串	数字	将字符串转换为一个数字，并比较数值大小

当明确将script元素设置为language="JavaScript1.2"或更高版本时，浏览器更倾向于根据字面意思来进行同等性判断，这意味着不会进行自动数据转换。因此，由于自动数据转换的存在，在大多数情况下下面这个表达式的值为true，但在JavaScript 1.2或更高版本的脚本中，其值为false。

```
123 == "123"
```

由于DOM及XHTML标准并未提供一个具体的位置来指定脚本语言版本，因此应该尽量规避这些特殊情况。如果脚本需要测试操作数的绝对同等性，那么请使用新的“===”（同一性运算符）。而对于一些典型值的同等性判断，标准的等于运算符就已经完全胜任了。

无论具体的JavaScript版本是什么，一旦要对对象的值进行比较，都应该根据toString()或valueOf()等方法给出的值来进行比较。

示例

```
if (n == m) {
    ...
}
```

>

IE 3 NN 2 Moz all Saf all Op 7 ECMA 1

大于运算符会对运算符两边操作数的值进行比较。如果左操作数的数值比右操作数大，那么所在的表达式的值就为true。在比较字符串时，会将它们转换成为Unicode值以便进行比较。

示例

```
if (a > b) {
    ...
}
```

>=

IE 3 NN 2 Moz all Saf all Op 7 ECMA 1

大于等于运算符会对运算符两边操作数的值进行比较。如果左操作数的数值比右操作数大或二者相等，那么所在表达式的值就为true。在比较字符串时，会将它们转换成为Unicode值以便进行比较。

示例

```
if (a >= b) {
    ...
}
```

===

IE 4 NN 4 Moz all Saf all Op 7 ECMA 3

严格相等（或称为全相等）运算符会对两个操作数进行比较并返回一个布尔结果。只有当两个操作数的值与数据类型均相同时此运算符才会返回true，在比较的过程中并不会进行自动数据类型转换。更为自由的同等性比较请参见等于运算符（==）。

示例

```
if (n === m) {
    ...
}
```

++

IE 3 NN 2 Moz all Saf all Op 7 ECMA 1

自增运算符会将一个变量表达式的当前值加1。将这个运算符放置在变量之前或之后表达的含义有所不同。当运算符位于变量之前时，那么变量会首先加1，然后才参与当前指令的计算。例如，在如下指令序列中：

```
var a, b;
a = 5;
b = ++a;
```

在将a赋值给b时，首先会将a加1。因此，当这些指令运行完毕时，a和b的值均为6。与之相反的是，在如下指令序列中：

```
var a, b;
a = 5;
b = a++;
```

首先会将a赋值给b，然后再将a增大。当这些指令执行完毕时，a等于6而b的值则为5。

这一特性影响着for循环计数变量的定义与使用方式。通常，在循环运行指令后，通过一个循环计数器从最小值开始向前递增计数。因此，大多数循环计数器都会将此运算符放置在计数变量之后：

```
for (var i = 10; i >=0; i++) {...}
```

示例

```
++n
n++
```

!=

IE 3 NN 2 Moz all Saf all Op 7 ECMA 1

不等于运算符会对两个操作数进行比较并返回一个布尔结果。随着script元素所指定的不同JavaScript版本，这个运算符的具体行为也存在着一定的差异。如果将language属性设置为JavaScript或JavaScript 1.1，那么某些操作数也会像在等于运算符中一样自动进行转换。当script元素设置为language="JavaScript1.2"或更高版本时，这一情况就会出现一些变化。浏览器更倾向于根据字面意思来进行不等性判断，这意味着不会进行自动数据转换。因此，由于自动数据转换的存在，在大多数情况下下面这个表达式的值为true，但在属于JavaScript 1.2或更高版本的脚本中，其值为false。

```
123 != "123"
```

由于DOM及XHTML标准并未提供一个具体的位置来指定脚本语言版本，因此应该尽量规避这些特殊情况。如果脚本需要测试操作数的绝对不等性，那么请使用新的“!==”（运算符）。而对于一些典型值的不等性判断，标准的不等于运算符就已经完全适用了。

1134

无论具体的JavaScript版本，一旦要对对象的值进行比较，都应该根据toString()或valueOf()等方法给出的值来进行比较。

示例

```
if (n != m) {
    ...
}
```

<

1133

IE 3 NN 2 Moz all Saf all Op 7 ECMA 1

小于运算符会对运算符两边操作数的值进行比较。如果左操作数的数值比右操作数小，那么所在的表达式的值就为true。在比较字符串时，会将它们转换成为Unicode值以便进行比较。

示例

```
if (a < b) {
    ...
}
```

<=

IE 3 NN 2 Moz all Saf all Op 7 ECMA 1

小于等于运算符会对运算符两边操作数的值进行比较。如果左操作数的数值比右操作数小或二者相等，那么所在的表达式的值就为true。在比较字符串时，会将它们转换成为Unicode值以便进行比较。

示例

```
if (a <= b) {
    ...
}
```

%

IE 3 NN 2 Moz all Saf all Op 7 ECMA 1

取模运算符会使用右侧值除运算符左侧的值。如果除后还存在一个余数，那么表达式的计算值就等于该余数对应的一个整数。如果不存在余数，则返回0。此时左右两个操作数都必须是数字，使用这个运算符的表达式将计算出一个数字。即使对余数不感兴趣，通过这个运算符也能快速地判断出两个值是否可以完全除尽。

示例

```
if ((dayCount % 7) > 0) {
    ...
}
```

*

IE 3 NN 2 Moz all Saf all Op 7 ECMA 1

乘法运算符会使用右侧值乘以运算符左侧的值。此时左右两个操作数都必须是数字，使用这个运算符的表达式将计算出一个数字。

示例

```
var myProduct = number1 * number2;
```

-

IE 3 NN 2 Moz all Saf all Op 7 ECMA 1

这是一个求负运算符。这个一元运算符会求取一个操作数的负值。例如，在如下指令中：

```
a = 5;
b = -a;
```

b的值会变为-5。如果在一个负值上使用一个求负运算符，那么就会将它转换成一个正值。

示例

```
var myOpposite = -me;
```

!==

IE 4 NN 4 Moz all Saf all Op 7 ECMA 3

严格不相等（或称为非全等）运算符会对两个操作数进行比较并返回一个布尔结果。只有当两个操作数的值与数据类型完全相等时，这个运算符才会返回false。对于不是特别严格的比较，请参见不等运算符（!=）。

示例

```
if (n !== m) {
    ...
}
```

!

IE 3 NN 2 Moz all Saf all Op 7 ECMA 1

这是逻辑非运算符。这个一元运算符会对一个布尔型的操作数执行取非运算。应该在明确的布尔值上使用逻辑非运算符，例如某个比较结果或一个布尔属性设定等。

示例

```
if (a == !b) {
    ...
}
```

||

IE 3 NN 2 Moz all Saf all Op 7 ECMA 1

逻辑或运算符用于比较两个布尔表达式以获得其真实性。如果两个表达式其中之一的值为true，那么||运算符的执行结果为true，如果两个表达式中的值均为false，那么其运算结果会是false。此时使用的布尔表达式既可以由一个比较表达式（使用其他比较运算符）组成，也可以是其他各种数据值。关于常用数据类型、值及等价的布尔值，请参见本章前文所述的逻辑与运算符。

在||运算符的帮助下，可以创建一个复合条件。例如，如果要判断两个条件之一或两者是否为真，那么可以创建如下的判断条件：

```
var userEntry1 = document.forms[0].entry1.value;
var userEntry2 = document.forms[0].entry2.value;
if (userEntry1 || userEntry2) {
    ...
}
```

在这个复合条件中，||运算符在得出结果前需要判断两个操作数之一或两者是否为true。如果用户在第1个域中输入了文本，那么此复合条件将立即变为true。如果仅在第2个文本域中进行了输入，那么就需要对第2个操作数进行判断。由于其判断结果为true，即非空的字符串，那么此复合条件的判断结果也为true。只有当两个操作数均为false时，复合判断条件的判断结果才会为false。

示例

```
if (a <= b || b >= c) {
    ...
}
```

-

IE 3 NN 2 Moz all Saf all Op 7 ECMA 1

减法运算符会使用左侧值减去运算符右侧的值。此时左右两个操作数都必须是数字，使用这个运算符的表达式将计算出一个数字。

示例

```
var myDifference = number1 - number2;
```

delete

IE 4 NN 4 Moz all Saf all Op 7 ECMA 1

delete运算符会将一个属性从一个对象中删除，或者将一个数据项从一个脚本生成的数组中删除。需要注意的是，删除一个数组项并不会改变数组的长度或现有各项的数字索引值，此时只是将已删除项的值简单设置为undefined而已。因此delete运算符并不是一个内存管理工具。

在E4X环境中，delete运算符还可以作用于一个XML对象内的元素及属性。它可以删除元素或属性的引用。如果在一个XMLList对象中删除一个数据项，那么位于该项数据后的所有项的索引值将会被减1，而不会留下空白项，这与之前删除数组数据项的情况有所不同。

示例

```
delete myString.author;
```

In

IE *5.5* NN *n/a* Moz *all* Saf *all* Op *7* ECMA *n/a*

in运算符可以让脚本快速地获知一个对象是否已经实现了一个特定的属性或方法。左操作数是一个包含属性或方法（不包括括号）名称的字符串，而右操作数则是一个对象引用。如果在查找的过程中需要借助于DOM引用必须的点号，那么请将它们放置在表达式中的节点引用端。换句话说，请不要使用"style.filter" in document.body这种写法，而应使用"filter" in document.body.style。目前浏览器几乎均未实现这个未来的ECMA运算符。如果不是这样，它将是对象探测领域的一个有用工具。

示例

```
if ("createDocument" in document.implementation) {
    //请放心使用 document.implementation.createDocument()
}
```

Instanceof

IE *5(Win)* NN *n/a* Moz *all* Saf *all* Op *7* ECMA *n/a*

instanceof运算符可以让脚本决定一个对象（左操作数）是否是一个已知对象的实例（或继承自已知对象）。从某些方面来看，这个运算符与typeof运算符比较类似，不过它并不会返回一个宽泛的对象类型，相反地，使用instanceof运算符的表达式会为这种更为特殊的对象类型测试返回一个布尔值。事实上，可以在对象上查找自定义对象，而在Mozilla和Opera中，还可以查找W3C DOM的树对象原型。鉴于typeof运算符作用于一个数组时会返回一个object，因此可以由此发现某个对象是否是一个数组实例：

```
myVar instanceof Array
```

然而值得注意的是，如果上述表达式的结果为true，那么在下式中：

```
myVar instanceof Object
```

数组就是一个根Object对象的后代，因此它也是该根对象的一个实例。

在Mozilla和Opera中，操作数之一或两个操作数均可以是DOM元素对象的引用，因此下述表达式是合法的，并且可以在这两个浏览器上正常工作：

```
document.getElementById("widget") instanceof HTMLDivElement
```

示例

```
if (theVal instanceof Array) {
    //请放心地将 theVal 作为数组来使用
}
```

New

IE *3* NN *2* Moz *all* Saf *all* Op *7* ECMA *1*

new运算符可以创建以下ECMA标准静态对象的实例：

```
Array
Boolean
Date
Function
Number
Object
RegExp
String
```

这个运算符还可以创建以下的E4X静态对象：

```
Namespace
QName
XML
XMLList
```

1138

包含这个运算符的表达式会生成一个对象实例。换句话说，调用这个运算符就会让JavaScript查找一个同名的构造函数。因此new运算符还可以作用于那些通过自定义构造函数形成的自定义对象。在IE/Windows系统中，它还可以创建一些专有对象实例，如ActiveX以及VBArray对象。

目前的语法规则允许使用以下3种方式来使用这个运算符，即对静态对象进行命名、使用空括号的静态对象，以及将参数放置在括号内的静态对象：

```
var myArray = new Array;
var myArray = new Array();
var myArray = new Array("Larry", "Moe", "Curly");
```

但在以上3种方式中，只能保证最后两个示例能在所有的脚本浏览器中正常工作。除Date对象之外，如果在创建对象的过程中忽略了所需的参数，那么新生成的实例将仅拥有已指定给静态对象prototype属性的相关属性。

示例

```
var now = new Date();
```

this

IE 3 NN 2 Moz all Saf all Op 7 ECMA 1

指向当前对象。例如，在一个表单控件对象的事件处理程序中，可以将当前对象作为一个参数传入处理函数：

```
<input type="text" name="ZIP" onchange="validate(this);">
```

在一个自定义的对象构造函数中，这个关键字对应于对象本身，可以通过它来为其属性指定属性值，有时甚至还可以用来创建属性：

```
function CD(label, num, artist) {
    this.label = label;
    this.num = num;
    this.artist = artist;
}
```

在一个函数内部，this关键字则对应于该函数对象。然而，如果该函数被划归为自定义对象构造函数的一个方法，那么this就指向执行这个方法的对象实例。

示例

```
<input type="text" name="phone" onchange="validate(this.value);">
```

typeof

IE 3 NN 3 Moz all Saf all Op 7 ECMA 1

typeof运算符返回数据类型的6种字符串描述之一。这些返回值类型包括：

```
boolean
function
number
object
string
undefined
```

1139

object类型还包括数组，但这个运算符并未提供与对象类型或数组相关的详细信息。

在E4X环境中，typeof运算符则会返回XML及XMLList对象实例的XML值。

示例

```
if (typeof someVar == "string") {
    ...
}
```

void

IE 4 NN 3 Moz all Saf all Op 7 ECMA 1

这个一元运算符会对表达式进行计算，但无论表达式的计算结果如何，都会统一返回一个undefined值。这个运算符通常与调用各种方法的JavaScript:伪URL一同使用。如果此时被调用的方法返回一个值，那么调用它的表达式仍然会忽略其返回值。

示例

```
<a href="javascript: void getSound();" >...</a>
```

控制指令（Control Statements）

break

IE 3 NN 2 Moz all Saf all Op 7 ECMA 1

停止执行当前循环，并将控制权返回给当前循环结束位置后的下一个脚本指令。需要注意的是，如果并未声明一个标记参数，那么break指令的作用域就位于其所在的循环内部。如果要跳出一个内嵌循环，那么可以在每一个内嵌层上指定一个标记，并将目标标记作为break指令的参数。请参见label指令。

语法

```
break [label]
```

示例 请参见label指令。

catch

IE 3 NN 2 Moz all Saf all Op 7 ECMA 1

请参见try指令。

continue

IE 3 NN 2 Moz all Saf all Op 7 ECMA 1

停止执行当前循环，并返回循环顶部，从下一个参数值开始进入循环。如果使用了一个内嵌的循环结构，那么可以在每一个内嵌层上指定一个标记，并将目标标记作为continue指令的参数。请参见label指令。

语法

```
continue [label]
```

示例

```
outerLoop:
for (var i = 0; i <= maxValue1; i++) {
    for (var j = 0; j <= maxValue2; j++) {
        if (j*i == magic2) {
            continue outerLoop;
        }
    }
}
```

do/while

IE 4 NN 4 Moz all Saf all Op 7 ECMA 3

当判断条件为真时，在循环中执行脚本指令。由于此时是在循环尾部进行条件判断，因此循环中的脚本指令至少会被执行一次。调用这个循环结构时，循环中的指令代码应该能够改变条件判断表达式的值，这一点非常重要，否则会进入无限循环状态。

语法

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

示例

```
var i = 1;
do {
    window.status = "Loop number " + i++;
} while (i <= 10)
window.status = "";
```

for

IE 3 NN 2 Moz all Saf all Op 7 ECMA 1

这个循环结构可以允许将一些指令反复执行多次，通常可以控制具体的执行次数。

语法

```
for ([initExpression]; [condition]; [updateExpression]) {
    statements
}
```

示例

```
var userEntry = document.forms[0].entry.value;
var oneChar;
for (var i = 0; i < userEntry.length; i++) {
    oneChar = userEntry.charAt(i);
    if (oneChar < "0" || oneChar > "9") {
        alert("The entry must be numerals only.");
    }
}
```

for-in

IE 3 NN 2 Moz all Saf all Op 7 ECMA 1

这个结构是for循环的一种变异体，它可以提取一个对象的属性名称及对应值。浏览器设定的各个可枚举属性（以及Mozilla、Safari和Opera中的方法）都会出现在这个结构的输出中。

在E4X环境内，对象引用就是一个XML对象实例，并且会自动将它视为一个XMLList对象。因此，在迭代的过程中，循环变量和列表中的一个索引的作用很类似。例如，如果myXml是一个拥有12个数据记录的XMLList对象，在它的每个记录中均包含一个名为productID的元素，那么就可以使用如下代码遍历这些内嵌元素：

```
for (var i in myXml..productID) {
    alert("Product ID=" + myXml..productID[i]);
}
```

语法

```
for ([var] varName in objectRef) {
    statements
}
```

示例

```
function showProps() {
    objName = "image";
    obj = document.images[0];
    var msg = "";
    for (var i in obj) {
```

```
        msg += objName + "." + i + "=" + obj[i] + "\n";
    }
    alert(msg);
}
```

for-each-in

IE *n/a* NN *n/a* Moz *1.8.1* Saf *n/a* Op *n/a* ECMA *E4X*

这是一个为E4X实现而定制的变种for-in循环。在E4X环境中，for-in结构会为循环变量指定一个索引值。而在for-each-in结构中，它会将一个对象引用赋予该变量，以便遍历XMLList中的所有对象。

语法

```
for each ([var] varName in objectRef) {
    statements
}
```

示例

```
for (var i in myXml..productID) {
    alert("Product ID=" + i);
}
```

if

IE *3* NN *2* Moz *all* Saf *all* Op *7* ECMA *1*

这是一个简单的条件判断指令，它可以提供一个可选择的执行路径。

语法

```
if (condition) {
    statement(s) if true
}
```

示例

```
if (myDateObj.getMonth() == 1) {
   calcMonthLength();
}
```

if-else

IE *3* NN *2* Moz *all* Saf *all* Op *7* ECMA *1*

这是一个条件判断指令，它可以根据判断结构提供两条不同的执行路径。当然，在if-else指令指定的路径内，还可以内嵌其他的if或if-else指令。

语法

```
if (condition) {
    statement(s) if true
} else {
    statement(s) if false
}
```

示例

```
var theMonth = myDateObj.getMonth();
if (theMonth == 1) {
    monLength = calcLeapMonthLength();
} else {
    monLength = calcMonthLength(theMonth);
}
```

label

IE 4 NN 4 Moz all Saf all Op 7 ECMA 3

可以为任意的指令执行分块（包括控制结构）指定标记标识符，使用标记后，就可以让内嵌在控制结构中的break及continue指令跳出内嵌层，并且还可以让跳转目的地位于这两个指令的正常作用域之外的其他层次位置。

语法

```
labelName:
```

示例

```
outerLoop:
for (var i = 0; i <= maxValue1; i++) {
    for (var j = 0; j <= maxValue2; j++) {
        if (i == magic1 && j == magic2) {
            break outerLoop;
        }
    }
}
```


return

IE 3 NN 2 Moz all Saf all Op 7 ECMA 1

停止执行当前函数。return指令可以放置在函数中的任意位置，包括控制结构之内。使用时可以根据需要指定一个值，并将它返回到函数的调用指令。返回值类型可以是任意一种JavaScript数据类型。如果返回一个值的return指令位于循环或其他控制结构内，那么执行树中的每一条分支中都应该设定一个return指令，如果函数会执行到主执行域附近或函数末尾，那么还应设定默认的return指令。

语法

```
return [value]
```

示例

```
function validateNumber(form) {
    var oneChar;
    for (var i = 0; i < userEntry.length; i++) {
        oneChar = form.entry.value.charAt(i);
        if (oneChar < "0" || oneChar > "9") {
            return false;
        }
    }
    return true;
}
```

switch-case

IE 4 NN 4 Moz all Saf all Op 7 ECMA 3

为一个表达式的多个条件状态简便地提供各种执行路径。当可选的break指令出现在每个case块的末尾时，它会打断switch指令的执行进程，并防止不慎执行了default块。

语法

```
switch (expression) {
    case label1:
        statements
        [break;]
    case label2:
        statements
        [break;]
    ...
    [default:
       statements]
}
```

示例

```
var productList = document.forms[0].prodList;
var chosenItem = productList.options[productList.selectedIndex].value;
switch(chosenItem) {
    case "Small Widget":
        document.forms[0].price.value = "44.95";
        break;
    case "Medium Widget":
        document.forms[0].price.value = "54.95";
        break;
    case "Large Widget":
        document.forms[0].price.value = "64.95";
        break;
    default:
        document.forms[0].price.value = "Nothing Selected";
}
```

1144

throw

IE 5 NN n/a Moz all Saf all Op 7 ECMA 3

触发一个异常状态，并随之传递一个值。尽管也可以传出一个简单的字符串，但从理论上说，应该传递出一个JavaScript的Error对象实例，并使用充足有效的信息填充该实例对象，以便catch指令能够根据错误信息做出智能判断。使用throw指令时，必须将它放置在try-catch结构的try部分中。

语法

```
throw value;
```

示例

```
function processNumber(inputField) {
    try {
        var inpVal = parseInt(inputField.value, 10);
        if (isNaN(inpVal)) {
            var msg = "Please enter a number only.";
            var err = new Error(msg);
            if (!err.message) {
                err.message = msg;
            }
            throw err;
        }
        // 处理数字
    }
    catch (e) {
        alert(e.message);
        inputField.focus();
        inputField.select();
    }
}
```

try-catch

IE 5 NN n/a Moz all Saf all Op 7 ECMA 3

这个结构提供了一种整洁的方法来捕获错误和异常，并优雅地进行处理。在使用时，这个异常处理结构中的两个组成部分均不能被忽略。如果在try部分中发生了错误，那么脚本执行会立即跳转到catch块中，其中的脚本代码就可以显示警告对话框，修改数据，或执行其他动作来防止JavaScript进一步引发错误。

自然发生的异常（例如，并非由throw抛出的异常）会将一个Error对象实例作为参数传递给catch块。catch块中的指令可以检查Error对象的各个属性，以便决定如何处理已截获的异常。因此，一个catch块能够处理不同的异常与错误。

1145

但是，只能在支持它的浏览器中使用try-catch结构。为了防止旧浏览器看到这个结构，应该将所有的相关代码放置在至少需要JavaScript 1.5才能执行的<script>标签中，并设定language = "JavaScript1.5"或后续属性。

语法

```
try { statement(s) that could cause error
}
catch (errorInfo) { process error(s) gracefully
}
```

示例

```
function insertOneNode(baseNode, newNode, position) {
    try {
        baseNode.insertBefore(newNode, baseNode.childNodes[position]);
    }
    catch (e) {
        // 处理W3C DOM异常类型
        switch (e.name) {
            case "HIERARCHY_REQUEST_ERR" :
                // 处理错误的树层次引用
                break;
            case "NOT_FOUND_ERR" :
                // 处理错误的节点引用
                break;
            default :
                // 处理所有其他异常
        }
    }
    return true;
}
```

while

IE 3 NN 2 Moz all Saf all Op 7 ECMA 1

只要判断条件为真，就在循环中执行脚本指令。由于此时是在循环开始处进行条件判断，可以设想，如果条件为假，那么就不会执行循环中的脚本指令。调用这个循环结构时，循环中的指令代码应该能够改变条件判断表达式的值，这一点非常重要。否则会进入无限循环状态。

语法

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

示例

```
var i = 0;
while (!document.forms[0].radioGroup[i].checked) {
    i++;
}
alert("You selected item number " + (i+1) + ".");
```

with

IE 3 NN 2 Moz all Saf all Op 7 ECMA 1

with指令可以在内嵌的所有指令代码的作用域中添加一个对象。当某些代码都依赖于一个特定的对象引用时，这一指令可以适当减小代码量。但需要注意的是，with结构的效率通常都非常低。如果将对象引用指定给一个本地变量，并在函数中使用该变量，那么执行效率会得到大幅提升。

语法

```
with (objectRef) {
    statements
}
```

示例

```
with (document.forms[0]) {
    name1 = firstName.value;
    name2 = lastName.value;
    mail = eMail.value;
}
```


yield

IE *n/a* NN *n/a* Moz *all* Saf *1.8.1* Op *n/a* ECMA *n/a*

将yield关键字插入到函数的循环块后，就可以将一个函数转换成所谓的“生成器”，这类函数应该被指定给一个变量或作为一个属性值，这样一来，该引用每次都可以循环一次循环块。

对函数进行首次调用时，会正常执行循环块之前的所有指令。但这一执行过程会在yield指令之前临时暂停下来。yield指令也会像return一样返回一个值，但生成器函数会一直等到循环块运行一次后才会返回该值。

通过在函数中调用next()方法，可以获得目前已生成的值。如果在yield指令之后还存在其他函数指令代码，那么此时才会执行这些指令，但在进入下一次循环时，又会在yield指令前暂停。

语法

```
function myFunction()
    statements
    while (true)
        yield value;
        statements
    }
}
```


示例

```
function getWords(txt) {
    var words = txt.split(" ");
    for (var i = 0; i < words.length; i++) {
        yield words[i].length;
    }
}

function getAverageWordLength(txt) {
    var lengthCounter = getWords(txt);
    var count = 0;
    var total = 0;
    try {
        while (total += lengthCounter.next()) {
            count++;
        }
    }
    catch(e) {}
    alert("Average word length is: " + (total/count).toFixed(2) + " characters.");
}
```

其他指令（Miscellaneous Statements）

//, /*...*/

IE 3 NN 2 Moz all Saf all Op 7 ECMA 1

通过这两个注释指令可以在脚本中输入一些非执行文本。在任意一行指令中，所有位于//符号后的内容都会被语言解释器所忽略。而该行之后的脚本代码，只要未以//符号开头，浏览器就会对其进行解析。

对于多行注释块，可以使用/*符号作为块起始，然后是由多行源代码组成的注释块内容，最后以*/符号结束该块。解释器会继续解析该符号之后的非注释代码。

示例

```
// 单行注释

/*
多行
注释
*/
```

@cc_on, @if, @end, @set

IE 4(Win) NN n/a Moz n/a Saf n/a Op n/a ECMA n/a

IE/Windows引入了一种被称为条件编译（*conditional compilation*）的脚本功能。依靠这种模式，一旦使用@cc_on指令开启了条件编译模式，就允许JScript指令在一种可实验的条件环境下运行。如果将条件编译指令放置在JavaScript注释中，那么条件指令就只能在Windows系统中的IE 4及后续浏览器中执行，而不会与其他浏览器产生冲突。

其名称中的“条件”一词来自于那些可以反映环境属性的大量全局属性（全部以@符号开头），例如脚本引擎版本、操作系统及CPU类型等。在很多浏览器的navigator对象属性中，都可以获得这些条件信息，因此也不是只有它设定的环境才拥有这些信息。

如果要使用条件编译，那么请在脚本中引入如下指令：

```
/*@cc_on @*/
```

但要注意的是，一旦开启了这个模式，那么在当前页面内就无法再将它关闭。

当浏览器运行在Jscript 5.6或更高版本时（IE 6以上），下面这段代码展示了@if及相关指令如何在窗口状态条中显示某些环境信息：

```
/*@cc_on @*/
/*@if (@_jscript_version >= 5.6 && @_x86)
    status = "Now running JScript version " + @_jscript_version +
    " with Intel inside.";
    @else @*/
    status = "Have a nice day.";
/*@end @*/
```

使用@set指令，可以在条件编译块中将一个数字或布尔值（不包括字符串）赋予一个变量（包含@前缀）。

```
@set @isOK = @_win32
```

一旦对变量进行了初始化，就可以在整个页面内的脚本指令中使用该变量。需要注意的是，在条件编译中编写@指令的Visual Basic感知语法时，不允许在指令末尾使用分号。

此外，由于这些条件编译指令只会在受控的环境版本下得以编译，因此它对IE部署环境比较有用，通过这一

模式，可以为旧IE浏览器屏蔽新的语言功能，如try-catch结构等，以防止产生编译错误。但对一个部署于多个浏览器的应用而言，如果采用这种模式，最多也只能在IE环境下进行调试，仍然可能不适合于所有的应用部署环境。

示例 请参见上文。

function

IE 3 NN 2 Moz all Saf all Op 7 ECMA 1

function关键字表示开始定义一个命名函数。关于匿名函数的相关信息，请参见Function对象。

示例

```
function myFunc(arg1, arg2) {
    //函数脚本指令
}
```

let

IE n/a NN n/a Moz 1.8.1 Saf n/a Op n/a ECMA n/a

如果在定义变量时，要求它的作用域不同于脚本中的其他全局作用域或局部作用域，那么就可以使用let关键字。例如，首先将表达式定义在括号内，然后在一个指令内使用这些变量，就可以将变量作用域限制在一个表达式内，请参考下例中alert()函数内的指令代码：

```
// 局部变量
var x = 1;
var y = 10;
alert(let(x = 10, y = y * 5) x*y);
alert(x*y);
```

在第1个警告对话框中，计算值为500；而第2个对话框则会显示10。

如果在一个块中使用let指令，那么通过let定义的变量、常量或函数的作用域就会被限制在该块中，也只能在块内使用这些已定义的值。另外，还可以使用let指令来定义for循环的计数变量，这样就不会与同一函数中的其他循环产生冲突。

示例

```
for (let i = 0; i < myArray.length; i++) {
    // 此处的指令会将 i 作为索引变量来使用
}
```

var

IE 3 NN 2 Moz all Saf all Op 7 ECMA 1

用于定义一个新变量的关键字。尽管全局变量也可以不使用这个关键字，但在定义新变量时最好每次都使用这个关键字。如果在一个函数内使用var关键字定义了一个变量，那么该函数内的指令就可以调用这个变量。

也可以简单地声明一个或多个变量名称，在这种情况下其初始值为null。当然，也可以使用一个值来初始化一个新变量。

示例

```
var a, b, c;
var myName = "Susan";
```

转义字符串字符（Special（Escaped）String Characters）

1150

\char

IE 3 NN 2 Moz all Saf all Op 7 ECMA 1

JavaScript提供了一种机制以便在字符串中引入常见的空白字符，以及那些可能与字符串表达式产生冲突的字符。这个关键字以一个反斜线字符（\）开头，后跟一个表示特定含义的单字符。下表列出了各个转义字符及对应的含义。

转义字符	描述
\b	退格
\t	水平制表符
\n	换行
\v	竖直制表符
\f	换页
\r	回车
\"	双引号“
\'	单引号‘
\\	反斜线

这些字符能够在警告、确认及提示对话框的文本中发挥作用。例如，如果要在警告对话框的多个段落间显示空白行，那么就可以插入换行符：

```
alert("First paragraph.\n\nSecond paragraph.")
```

值得注意的是，应该将这些字符应用于字符串，并且不要让它们影响HTML中回车的排版样式。

术语表
Glossary

绝对定位（absolute positioning）

在临近的外层容器的坐标系统内为元素设置精确位置就称为绝对定位。对于一个绝对定位的元素而言，它不会再位于其四周的HTML源代码形成的内容流中，而只会存在于自身的透明层中。

抽象对象（abstract object）

相对于那些能够与脚本代码建立联系的真实对象而言，抽象对象是对其他对象所拥有的特征的一种描述。目前的文档对象模型设计者通常会将多个抽象对象的特征混合为一个单独的脚本对象。例如，在Mozilla浏览器载入的文档中，p元素对象就继承了W3C DOM规范内某些抽象对象的属性、方法以及事件处理程序，这些对象包括：Node、Element、HTMLElement、HTMLParagraphElement、ElementCSSInlineStyle及EventTarget对象。

可访问性（accessibility）

为让残障用户尽可能地充分利用页面内容而需要考虑的设计问题。例如，对那些有视力障碍的用户而言，听觉样式单就可以增加网页的可访问性。请参见WAI以及unobtrusive JavaScript（非侵入式JavaScript）。

AJAX

异步JavaScript和XML。Jesse James Garrett创造了这个术语，它将脚本文档对象模型与基于XMLHttpRequest对象的Web页面同服务器端的异步后台通讯技术有机地结合在一起。

API

应用程序编程接口（Application Programming Interface），它通常是由一系列方法和属性组成的集合，程序员可以利用它来方便地调用更为复杂的计算机内部处理动作。在动态HTML中，面对程序员的设计目标与浏览器对元素定位的特定实现，通常会创建或使用一个自定义的API来作为它们之间的缓冲，这样一来，无论在哪种浏览器中都可以使用相同的编码模式。

匿名函数（anonymous function）

在定义函数时不需要明确给出函数名称的一种定义方式。通常会将它赋予一个对象属性，以便创建一个对象方法。

@规则（at-rule）

在样式单定义中使用的一种CSS命令。通常这种@规则命令会导入外部样式单或下载字体规范。每个@规则指令都会以一个@符号开始。

属性（attribute）

HTML（及XHTML）元素的一个特性就被称为它的一个属性。通过某些运算符可以为属性赋值，而在HTML中则使用等号（=）进行这一操作。在HTML中，有时无论具体的属性值是多少，只要属性名称出现就可以开启与该属性相关的一个功能；但在XHTML中，则需要为这些属性赋值。此外，HTML的属性名称并不需要区分大小写，而XHTML中的属性名则需要进行区分。

块级元素（block-level element）

这种HTML元素会自动在其前、后强制换行，从而保证在页面中的同一水平区域内不会有其他元素出现在它的周围。但绝对定位元素可以出现在它上方。h1元素就是块级元素之一。

border（边框）

在CSS中，它表示出现在块级元素的内容与填

充域外围的一片区域。即使其厚度为0，而且无法被用户看见，边框依旧存在。在页面中，边框实际上夹在边距与填充之间。

浏览器探查（browser sniffing）

一种用于设定全局变量以表示当前浏览器的品牌、版本、所在的操作系统及其他环境特性的脚本技术，此时通常会调用navigator对象的一些相关属性。在跨浏览器操作中，为防止出现浏览器兼容性问题，脚本可以使用这些变量来选择代码执行分支，以便适应不同浏览器的特定语法规范。不过这一技术正逐渐被对象检测技术所取代。

级联规则（cascading rule）

这是一个判断序列，当给定的元素存在多个适用的样式单规则时，能够执行CSS的浏览器将使用它来决定具体采用哪个样式单规则。每个级联规则都被赋予了一个特征评分值，以便决定元素所要采用的样式单规则及规则内的相关特性。

class

CSS中一个或多个元素组成的集合，之所以将这些元素归为一类，主要是为了在文档中使用同一个样式单来设置它们的样式，而这些元素的标签类型并不需要完全相同。通过class属性可以为元素指定一个类标识符（并且可以在一个样式单规则中使用该类选择符），这样就可以让网页作者在不依靠标签名称或ID的情况下创建元素分组。

集合（collection）

一组类型相同的脚本对象就被称为集合。通过标准的数字数组索引语法（collectionName[index]），脚本可以引用集合中的单个成员，而在集合的length属性的辅助下，还可以通过for循环来遍历访问集合中的所有成员。在新浏览器中，还可以通过数字索引来进行访问，如collectionName.item(index)。此外，如果已为对象指定了名称，那么也可以通过对象名来进行访问，例如，collectionName. namedItem("name")。IE浏览器还允许通过其集合标记符号来进行引用，如collectionName (index)。很多DOM属性和方法都会以集合的形式给出其返回值。

容器（container）

能够容纳其他类型元素的任意元素都称为容器。被包含的元素的标签会出现在容器元素的起始和结束标签之间。

情景选择器（上下文选择器，contextual selector）

在CSS中，通过这一途径，可以指定应该在怎样的包含环境下，将一个样式单规则应用于一个特定类型的元素。通过一个以空格进行分隔的列表，选择器指明了明确的包含层次，因此，“p em {color: red}”这个规则会将红色字体颜色应用于所有内嵌在p元素中的em元素，而位于一个li元素内的em元素则不会受这个样式单规则的影响。

CSS

这是Cascading Style Sheets的缩写形式，它是一种在万维网联合会（World Wide Web Consortium，W3C）资助下创建的推荐标准。在这个缩写后通常还会加上一个数字，以指明具体的版本号。例如，CSS2就是大家所熟知的CSS Level 2。从Level 2开始，CSS规范就被划分为几个独立的模块，如Core、HTML和Events等。

CSS-P

这是Cascading Style Sheets-Positioning的缩写形式。最初这部分工作从CSS中分离出来，但从CSS2开始这两个标准就融为一体，而且也不再使用CSS-P这个术语。

数据绑定（data binding）

这是微软的IE浏览器中的一种功能，它可以让页面内容动态连接一个数据源，如服务器上的一个数据库。例如，浏览器加载页面时，marquee元素就可以从一个数据库字段中获取最新的标题信息，并且滚动显示这些最新的标题内容。Windows版本的IE浏览器可以通过两种方式来访问多种数据源类型，但Macintosh系统中的IE浏览器只能与文本文件源进行数据绑定，而且文件中的信息必须以跳格或逗号进行分隔。

声明（declaration）

在CSS中，将一个属性名称、冒号运算符和一个属性值组合在一起就形成了一个声明。同一个样式单规则中的多个声明需要使用分号进行分隔。

不赞成使用（deprecated）

虽然某些网页标准和语言功能（如一个HTML元素或属性，以及对应的DOM等价物）得到了标准发布版的支持，但有时对应的文档并不建议继续使用它们。如果在发布版中已不赞成使用某项内容，那么在接下来的版本中通常会删除这项内容。为了保证后向兼容性，使得使用这些内容的现有网页能够继续正常工作，浏览器会在一定时期内继续支持这些不再建议使用的功能项。

DHTML

这是Dynamic Hypertext Markup Language（动态超文本标记语言）的缩写形式。DHTML是一种由很多标准组合而成的混合体，其中包括HTML（及XHTML）、CSS、DOM，以及ECMAScript等。实际上，通过这一技术，脚本在改变元素样式或文档树的组成结构时就不再需要从服务器端重新刷新页面内容。

DOM

这是W3C牵头的Document Object Model（文档对象模型）的缩写形式。这一术语中的所有字符通常都使用大写字母，特定的浏览器一般都会拥有一种特有的文档对象模型实现。

DTD

这是Document Type Definition（文档类型定义）的缩写形式，它十分详细地定义了元素、属性及属性值的类型，以便在SGML文档（亦即HTML或XML文档）中使用。用户并不会看到DTD文档，但在网页文档顶部的<!DOCTYPE>元素中通常会引用一个DTD文档，以便定义网页内容所要遵循的标记规则。当然，也可以自行定义新的DTD文档，但大多数HTML文档都遵循W3C发布的几个DTD之一。

动态内容（dynamic content）

文档加载完毕后，任何可以发生改变的HTML内容都称为动态内容。自Navigator 3及IE 4以后，不需要重新绘制页面就可以使用这些动态内容。可替换的img元素就是一个很好的例子。在IE 4和后续版本的IE浏览器，以及W3C DOM浏览器中，当内容发生改变，文档通过自动重排完成页面加载后，还允许对主体内容进行修改。

E4X

支持XML的ECMAScript（ECMAScript For XML），这是ECMAScript标准的一个扩展，设计它的目的是简化对XML文档要素的脚本访问操作。在主流浏览器中，Mozilla 1.8首先实现了E4X。

ECMA

一个位于瑞士的标准机构，以前被称为欧洲计算机制造商协会（European Computer Manufacturers Association）。

ECMAScript

这是基于JavaScript的脚本编程语言标准ECMA-262的常用名。这个标准定义了一种核心编程语言，其中未包含任何针对网页内容的特殊引用信息。ECMA-262第1版的功能大致上等价于Navigator 3中部署的JavaScript 1.1。其第2版纠正了第1版中的一些错误，而第3版则加入了一些与JavaScript 1.5所共有的新功能。

元素（element）

在文档中，由一个尖括号标签或标签对进行定义，具有特定上下文语境的部分就称之为元素。例如，<body>标签就在文档中创建了一个body元素。

事件绑定（event binding）

如果在某个对象上触发了事件，可以命令该对象对特定的事件类型进行处理，这种技术就称之为事件绑定。

事件冒泡（event bubbling）

IE 4及后续浏览器和W3C DOM使用这种事件模型机制，此时会让事件从目标元素开始沿着HTML元素的层次结构向上传播。当目标元素完成事件处理后，位于上层的事件处理程序还可以对该事件进行进一步的处理。通过cancelBubble属性或W3C DOM浏览器中的stopPropagation( )方法还可以在任一点上终止事件的传播。

事件处理程序（event handler）

这是一个与脚本有关的关键词，事件处理程序会截获一个事件动作（如一次鼠标点击），然后启动一个由一行或多行脚本指令组成的执行过程。在实际使用中，可以将它指定给HTML元素的一个属性，也可以作为元素对应对象的一个属性值，或者通过事件模型的特定方法（attachEvent( )或addEventListener( )）与对象联系在一起。每个元素都可以响应特定的事件集。

事件传播（event propagation）

对事件信息进行处理时，会一直沿着文档中元素或对象的层次结构完成这一操作。在新浏览器中，事件会从窗口、文档或主体元素（具体起始点由事件类型决定）内部向着目标元素传播。从这一点来看，事件传播完成了一次回转，并且沿着相同的包含层次路径向上浮动。当事件经过目标元素时，对于将要处理这个事件的对象而言，必须要让它能够捕获该事件。大多数事件类型都会自动向上传播。

滤镜（filter）

在Windows平台下，这是IE 4及后续浏览器的一种显示功能，它可以为文本内容增加排版效果，或者为视图切换加入动画过渡。通过CSS语法，就可以为元素指定一个滤镜效果。

HTML

这是Hypertext Markup Language（超文本标记语言）的缩写形式，它是SGML的一种简化版本，以便通过超文本传输协议（Hypertext Transfer Protocol，HTTP）在网络上发布信息。在W3C的资助下，HTML标准第4版扩展了将内容与形式分离的概念，它让HTML元素来定义内容上下文，而不是其特定的外观样式。对于XHTML而言，HTML 4.01是其初步规范的重要基础。

ID

ID是元素的一个标识符，在同一个文档内的所有元素中，应该保证ID的唯一性。通过id属性可以为一个元素指定其ID值，所有的HTML 4标签均支持这一做法。在很多位置都可以使用ID，例如，通过它可以将一个CSS样式单规则与文档中的某个元素联系起来，还可以简便地获取特定元素的一个脚本引用。

标识符（identifier）

为元素的id、class或name属性指定的名称，以及脚本编程语言中对象或变量的名称都被称为标识符。这些名称可以以任意的大写或小写英文字符开头，而后续的字符则可以使用字母、数字或下划线。

行内元素（inline element）

如果一个HTML元素与其周围的HTML内容均显示在同一个文本行内，那么就被称为行内元素。由于em元素中的内容并不会干扰文本内容的正常线性排列，因此这个能够强调段落内容的HTML元素就是一个行内元素。与之相对的是块级元素。

内在事件（intrinsic events）

HTML 4标准定义了一组事件，它们属于页面上实际显示的每个元素。这些事件主要是一些公用的鼠标和键盘事件。

JavaScript

这是Netscape公司的Brendan Eich为简化服务器端和客户端编程而发明的一种编程语言。在开发的最初阶段，它的名称是LiveScript。而在第1个商用脚本浏览器——Navigator 2发布之前，其名称变更为JavaScript。此后，JavaScript成为了ECMAScript的重要基础，而Microsoft公司所实现的JavaScript则称为JScript。

JavaScript样式单（JavaScript Style Sheets）

用于定义样式单规则的一种语法，但仅适用于Navigator 4。

JScript

从IE 3开始，在微软公司的IE浏览器中就内置了一种基于JavaScript的脚本编程语言，其正式名称就是JScript。它与ECMAScript和JavaScript均兼容。

JSON

这是JavaScript Object Notation的缩写形式，通过这种方式可以将JavaScript的数据类型转换为字符串。这种表示法会使用方括号来表示数组，并以大括号来表示对象，而其中的字符串则可以通过eval( )函数转换成对应的原始对象。

层（layer）

这个概念来自于Navigator 4为可定位元素设计的一种模型。从目前来看，如果将一个元素的CSS属性position设置为absolute、relative或fixed之一，那么该元素就属于这个范畴。每个已定位元素都位于其自身的透明层中，而这种透明层则位于文档主体之上。

边距（margin）

CSS中，元素边框向外的延伸区域就被称为边距。即使边距的厚度为零，每个元素四周也存在着边距。

媒介（media）

CSS中，它表示样式规则所请求的输出类型。虽然主流浏览器通常支持显示屏和打印机这两种媒介，但也存在着其他几种媒介形式，如投影、声音，以及手持设备的小显示屏等。

方法（method）

能够被脚本指令启动的一个脚本对象动作就称为方法。在JavaScript的语法中，方法名称后会跟着一对括号，这种引用形式很容易识别。在括号中，还可以包含零个或多个参数。根据具体的程序设计意图，方法可以返回一个值，当然，不返回任何值亦可。

修改键（modifier key）

修改键就是一个键盘按键，它通常与一个字符键或一个鼠标动作同时按下，以便启动一个特定的操作。所有的操作系统平台都有相同的修改键，其中包括Shift、Ctrl和Alt键。现代的微软键盘还包括Windows键，而Macintosh键盘则引入了Command键。当触发了任意一个字符键的事件时，还可以检查键盘和鼠标事件，以便获知此时按下了哪些修改键。

模块化（modularization）

最新的W3C标准已倾向于将庞大的标准分割为多个模块，即模块化。每个模块拥有一个特定的目标，例如，W3C DOM推荐标准中的事件模块。

基于Mozilla的浏览器（Mozilla-based browser）

由于Mozilla基金会管理了Gecko引擎的开发工作，因此所有使用这一引擎开发的网页浏览器都可以被划归为此类。目前流行的Mozilla浏览器包括Netscape 6及后续版本、Firefox和Camino。

节点（node）

在文档对象模型中，可以在文档的层次结构内进行引用的对象就被称为节点。某些类型的节点可以作为其他附加的、内嵌节点的容器，而有些节点则仅仅只包含文档的文本内容。

对象（object）

在脚本编程语言（如JavaScript）中，使用对象来表示HTML元素或其他可编程项。每个对象都拥有一定的属性和方法，以定义对象的行为和外观。通过读取或修改对象的属性，或者调用对象方法，脚本就可以改变对象的某些值或外观。浏览器的文档对象模型中的对象都对应于文档源代码中定义的HTML元素。例如，在最新的浏览器中，如果脚本为img对象的src属性指定了一个新的URL地址，那么在img元素所占据的页面矩形空间内，新的图像就会替代原有图像，其他类型的对象，如日期及字符串等，虽然并不会直接显示在屏幕上，但在脚本执行中时常会用到。

对象检测（object detection）

在脚本调用某个对象、属性或方法前，这种脚本编程技术可以验证所在的浏览器是否能够对它们提供相应的支持。目前这一技术已逐渐取代了浏览器探查技术。

填充（padding）

在CSS中，元素内容和边框之间的延伸区间就被称为填充。填充实际上就是在内容和边框之间提供一定的空余空间。即使填充的厚度为零，每个元素中也存在着填充。

父级（parent）

对HTML元素而言，在源代码顺序中临近的最外层元素就是其父级元素，例如，包围着td元素的tr元素就是它的父级元素。而对已定位的元素而言，父级元素决定了元素定位所用的坐标平面。对于脚本窗口对象而言，框架集文档就是一个框架的父级对象，它定义了加载当前文档的具体框架。

平台（platform）

为后续的产品开发提供基础的软件或硬件系统就被称为平台。对网页浏览器而言，这个术语可用于一个浏览器品牌或浏览器所在的操作系统，如Windows XP、Mac OS X和Solaris等。在本书中，“平台”通常用于浏览器品牌。

定位（positioning）

为页面中的某个元素指定精确的位置即定位。元素的定位方式有多种，包括相对定位、绝对定位及固定位置。

属性（property）

对象的一个特征就被称为属性，例如，对象的ID就是一个可以通过脚本获取的特征量。由于样式单属性可以作为style对象的属性进行引用，因此有时也可以将它们视为对象属性。CSS中的属性就是一个样式特征，如color或background-image等，通过样式单规则可以为它赋值。

伪类（pseudo-class）

伪类是一种指向HTML元素的特定状态或行为的样式单选择符，例如，如果设置为链接的a元素曾经被用户访问过，那么对应的伪类为a:visited。

伪元素（pseudo-element）

伪元素是一种指向HTML元素中特定部分的样式单选择符，例如，p:first-letter就表示一段文字的第1个字母。

quirks模式

为了后向兼容现有的HTML和CSS代码，现代浏览器在这种模式中会模拟早期浏览器的非标准行为。在IE 6、IE 7及基于Mozilla的浏览器中，可以根据<!DOCTYPE>标签中的不同设定来控制具体的运作模式（quirks模式及标准兼容模式）。

相对定位（relative positioning）

根据未进行定位时元素位置所确定的坐标系统，通过相对定位可以设定元素的准确位置。文档会保留为相对定位元素所指定的空白空间，这样一来，周围的HTML内容并不会挤入已定位元素留下的空白区域。

置换元素（replaced element）

置换元素既可以是一个块级元素也可以是一个行内元素，在替换其内容时并不需要对文档进行调整。例如，当页面加载完成后，通过脚本就可以替换img元素中的内容。

RGB

这是red-green-blue（红绿蓝）的缩写形式，它们是常用的颜色规范系统中的3个基准色，在HTML和CSS的颜色相关的属性中也使用这类颜色规范。每个颜色值的变化范围为0-255，设定时既可以使用十进制表示法也可以采用十六进制。

规则（rule）

在CSS中，与一个选择符相关的一系列样式声明就称为一个规则。通过为元素标签的style属

性赋值，也可以将一个规则内嵌在元素之中。

选择符（selector）

在CSS中，通过选择符，可以将元素的名称、ID、类或其他适当的元素分组与一个样式声明绑定在一起，而一旦将选择符和声明结合在一起就会创建一个样式单规则。

标准兼容模式（standards-compatible mode）

在这种模式中，浏览器会更为严格地遵循CSS规范。IE 6和IE 7使用这个模式来取消继承效果，即通常所说的quirks模式。在IE 6、IE 7，以及基于Mozilla的浏览器中，可以根据<!DOCTYPE>标签中的不同设定来控制具体的运作模式（quirks模式及标准兼容模式）。

样式单（style sheet）

在CSS中，一个或多个用于定义文档内容中特定分块的显示方式的规则就被称为样式单。样式单可以在外部文档中定义，也可以在style元素中定义，甚至可以通过style属性直接指定给某个元素。

过渡（transition）

在Windows系统下的IE浏览器中，隐藏并显示元素的一种可视效果就是过渡。通过滤镜可以对过渡效果进行控制。

非侵入式JavaScript（unobtrusive JavaScript）

这是一种针对Web页面的脚本设计方式，它能够在不开启脚本支持的情况下就执行一些关键任务及导航动作。在加载阶段，具有脚本感知功能的浏览器能够在访问者无法察觉的情况下对文档进行修改，从而为那些使用脚本浏览器的用户提供一些额外的增强功能。这种设计方式意味着那些使用特殊浏览器的访问者也不会错过Web站点或应用的主体内容。

验证（validation）

将源代码传入一个程序后，就可以将代码与检验标准进行比对，从而验证代码的语法准确性、结构完整性及与标准要求的符合度。

VBScript

这是Windows/IE中一种替代JScript的脚本编程语言。在同一个文档中，可以将VBScript代码和JScript代码结合在一起，甚至这两种代码还可以引用对方定义的变量和对象。

W3C

这是World Wide Web Consortium（万维网联合会）的缩写形式，详细信息请访问：*http://www.w3.org*。

WAI

这是W3C的Web Accessibility Initiative（网页无障碍倡议）行动的缩写形式。其目标是让残障用户也能够顺利地访问网页资源。详细信息请访问：*http://www.w3.org/WAI*。

Web 2.0

这实际上是一个市场术语，它表示通过富用户接口、访问者创建的内容或AJAX技术来加强Web站点和应用程序与访问者之间的交互性。

Web Forms 2.0

这是WHATWG所创建的网页应用程序1.0规范中的一部分，它的设计目标是在不借助脚本编程的情况下置入表单控件校验。

WHATWG

Web Hypertext Application Technology Working Group（Web超文本应用技术工作组，*http://whatwg.org*）是一个独立于W3C的组织，它旨在为万维网上部署各种应用而建立新的标准。其最主要的规范称之为Web Applications 1.0（网页应用程序1.0，也就是大家所熟知的HTML5），在这个规范中包括自校验表单（Web Forms 2.0）和其他重要功能。Opera 9是第1个实现Web Forms 2.0及声音对象的浏览器。

XHTML

这是W3C的Extensible Hypertext Markup Language（可扩展超文本标记语言）推荐的标

准缩写形式。它实际上是以XML应用的形式来实现HTML，详细信息请访问：*http://www.w3.org/MarkUp*。

XML

这是W3C的Extensible Markup Language（可扩展标记语言）推荐的标准缩写形式。它为促进结构化数据在互联网上的存储与传输提供了重要基础。详细信息请访问：*http://www.w3.org/XML*。

XMLHttpRequest

这是一个脚本对象的名称。通过这个对象，Web页面脚本可以与服务器进行异步通讯，以便在不干扰当前页面布局的情况下获取或提交XML数据。最初它是以一个ActiveX控件的形式出现在IE5中的，而现在，它已经成为了Mozilla、Safari、Opera和IE等主流浏览器的内建对象。

索引
Index

Symbol

A

We'd like to hear your suggestions for improving our indexes. Send email to *index@oreilly.com*.

B

C

D

E

F

G

H

I

K

L

M

N

O

P

Q

R

S

T

U

V

W

X

Y

Z