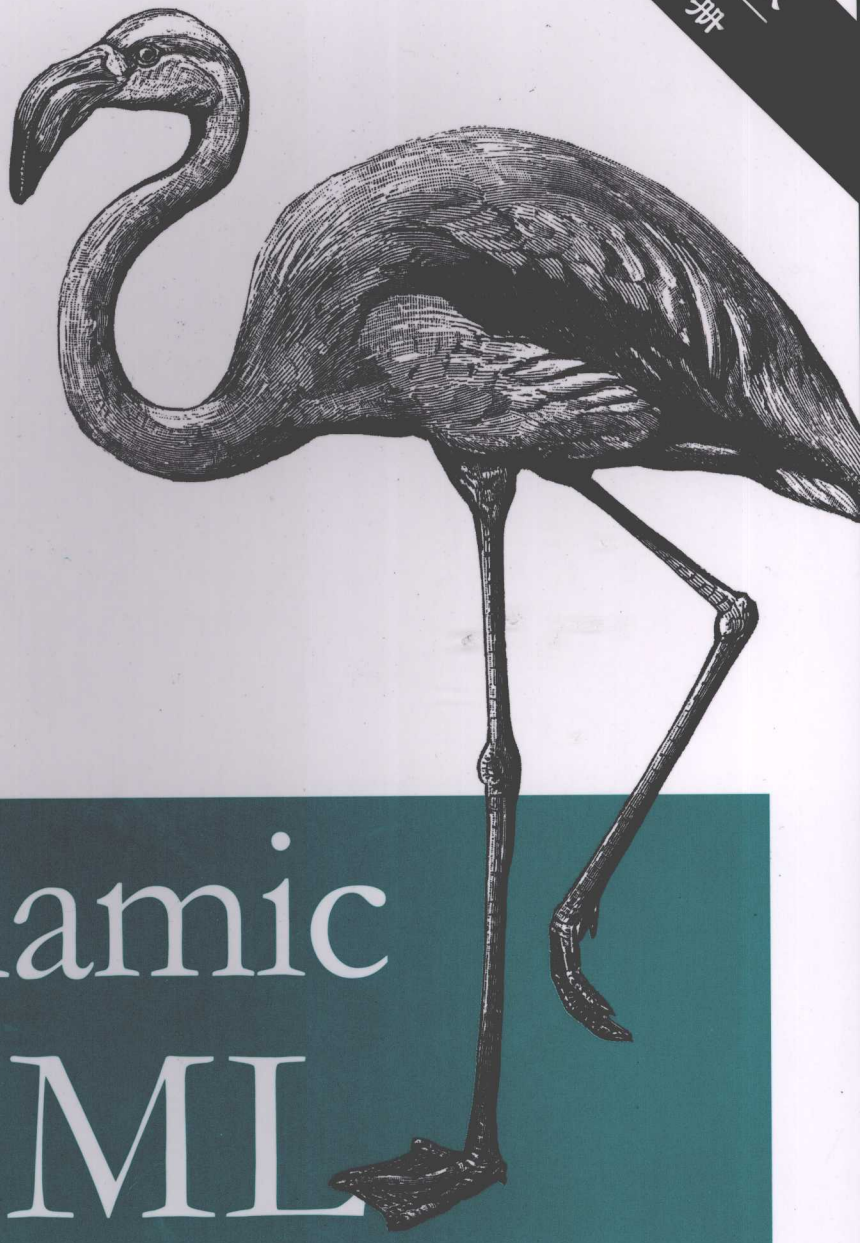
*Dynamic HTML*权威参考(影印版)

Updated for Ajax and Web 2.0

第三版

下册



Dynamic HTML

The Definitive Reference

O'REILLY®

東南大學出版社

Danny Goodman 著

第三版

Dynamic HTML权威参考(影印版)

Dynamic HTML: The Definitive Reference

下册

Danny Goodman

O'REILLY®

Beijing • Cambridge • Farnham • Köln • Paris • Sebastopol • Taipei • Tokyo

O'Reilly Media, Inc. 授权东南大学出版社出版

东南大学出版社

图书在版编目（CIP）数据

Dynamic HTML 权威参考：第3版：英文/（美）古德曼（Goodman, D.）著. —影印本.— 南京：东南大学出版社，2007.7

书名原文：Dynamic HTML: The Definitive Reference

ISBN 978-7-5641-0775-8

Ⅰ.D... Ⅱ.古... Ⅲ.超文本标记语言，HTML－程序设计－英文 Ⅳ.TP312

中国版本图书馆 CIP 数据核字（2007）第074176号

江苏省版权局著作权合同登记

图字：10-2007-104号

书　　名/ Dynamic HTML 权威参考（影印版）
责任编辑/ 张烨
封面设计/ Edie Freedman，张健
出版发行/ 东南大学出版社（press.seu.edu.cn）
地　　址/ 南京四牌楼2号（邮政编码210096）
印　　刷/ 扬中市印刷有限公司
开　　本/ 787毫米×980毫米　16开本　83.25印张
版　　次/ 2007年7月第1版　2007年7月第1次印刷
印　　数/ 0001-3000册
书　　号/ ISBN 978-7-5641-0775-8/TP·123
定　　价/ 128.00元（上下册）

Dynamic HTML 权威参考（影印版）

Dynamic HTML: The Definitive Reference

下册

O'Reilly Media, Inc.介绍

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二十世纪最重要的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。

出版说明

随着计算机技术的成熟和广泛应用，人类正在步入一个技术迅猛发展的新时期。计算机技术的发展给人们的工业生产、商业活动和日常生活都带来了巨大的影响。然而，计算机领域的技术更新速度之快也是众所周知的，为了帮助国内技术人员在第一时间了解国外最新的技术，东南大学出版社和美国 O'Reilly Meida, Inc.达成协议，将陆续引进该公司的代表前沿技术或者在某专项领域享有盛名的著作，以影印版或者简体中文版的形式呈献给读者。其中，影印版书籍力求与国外图书“同步”出版，并且“原汁原味”展现给读者。

我们真诚地希望，所引进的书籍能对国内相关行业的技术人员、科研机构的研究人员和高校师生的学习和工作有所帮助，对国内计算机技术的发展有所促进。也衷心期望读者提出宝贵的意见和建议。

最新出版的影印版图书，包括：

- 《深入浅出面向对象分析与设计》（影印版）
- 《Ajax on Rails》（影印版）
- 《Java 与 XML 第三版》（影印版）
- 《学习 MySQL》（影印版）
- 《Linux Kernel 技术手册》（影印版）
- 《Dynamic HTML 权威参考 第三版》（影印版）
- 《ActionScript 3.0 Cookbook》（影印版）
- 《CSS:The Missing Manual》（影印版）
- 《Linux 技术手册 第五版》（影印版）
- 《Ajax on Java》（影印版）
- 《WCF Service 编程》（影印版）
- 《JavaScript 权威指南 第五版》（影印版）
- 《CSS 权威指南 第三版》（影印版）
- 《嵌入式系统编程 第二版》（影印版）
- 《学习 JavaScript》（影印版）
- 《Rails Cookbook》（影印版）

About the Author

Danny Goodman has been writing about personal computers and consumer electronics since the late 1970s. In 2006, he celebrated 25 years as a freelance writer and programmer, having published hundreds of magazine articles, several commercial software products, and three dozen computer books. Through the years, his most popular book titles—on HyperCard, AppleScript, JavaScript, and Dynamic HTML—have covered programming environments that are accessible to nonprofessionals yet powerful enough to engage experts. He is the author of O'Reilly's popular *JavaScript and DHTML Cookbook*.

To keep up to date on the needs of web developers for his recent books, Danny is also a programming consultant to some of the industry's top intranet development groups and corporations. His expertise in implementing sensible cross-browser client-side scripting solutions is in high demand and allows him to, in his words, "get code under my fingernails while solving real-world problems."

Danny was born in Chicago, Illinois during the Truman Administration. He earned a B.A. and M.A. in Classical Antiquity from the University of Wisconsin, Madison. He moved to California in 1983 and lives in a small San Francisco–area coastal community, where he alternates views between computer screens and the Pacific Ocean.

Colophon

The animal on the cover of *Dynamic HTML: The Definitive Reference*, Third Edition, is a flamingo. Flamingos are easily identifiable by their long legs and neck, turned-down bill, and bright color, which ranges from white to pink to bright red. There are five living species of flamingo, encompassing the family *Phoenicopteridae*. Flamingos are found in Asia, Africa, Europe, South American, and the Caribbean islands. Although wild flamingos are sometimes seen in Florida, they do not naturally nest in the United States.

Flamingos feed on small crustaceans, algae, and other unicellular organisms. Their unusually shaped bills provide flamingos with a unique food-filtering system. A flamingo eats by placing its head upside down below the water surface and sucking in water and small food particles through the serrated edges of its bill. The flamingo then pushes its thick, fleshy tongue forward, forcing the water out but trapping the food particles on lamellae inside the beak.

In the wild, flamingos tend to live in remote, difficult-to-reach areas. In the suburbs, however, they stand guard over many a front lawn.

The cover image is from Dover Pictorial Archive. The cover font is Adobe ITC Garamond. The text font is Linotype Birka; the heading font is Adobe Myriad Condensed; and the code font is LucasFont's TheSans Mono Condensed.

Table of Contents

CHAPTER 3

Event Reference

The purpose of this chapter is to provide a list of every event type implemented in current mainstream browsers, as well as those specified in the W3C recommendation for the Events module of DOM Level 2. Events are listed alphabetically by their type names—the same format used by the W3C DOM Events modules. Event bindings using an element object property or the IE `attachEvent()` method require the "on" prefix to the event type name. So that you can readily see whether a particular entry applies to the browser(s) you must support, a version table accompanies each term listed in the following pages. This table tells you at a glance the version of Internet Explorer (IE), pre-Mozilla Netscape Navigator (NN), Mozilla (Moz), Apple Safari (Saf), Opera (starting from version 7), and W3C DOM specification in which the term was first introduced.

If a listing for IE signifies Win or Mac, it means that the event is supported only for the Windows or Macintosh operating system version. IE 5.5 or later is for Windows only. Note that a large number of event types are supported only in IE for Windows, and many of those apply only to data binding applications. If you are concerned with cross-browser deployment, pay very close attention to the browser compatibility charts to find the events that work on a broad array of browser brands and versions. Online Section VI contains many guidelines and examples for blending otherwise incompatible event mechanisms into routines that work on many browser types.

In the following listings below, the "Bubbles" category indicates whether the event follows event bubbling propagation (in browsers that support event bubbling, described in Online Section VI), while the "Cancelable" category means that the default action usually associated with the event (such as navigating to a new URL when clicking on an a element) can be canceled by script statements, thus averting the normal operation. The category named "Typical Targets" usually points to broad types of elements to which the event type may be applied. For more specific element support for each event type, consult Chapter 9.

Alphabetical Event Reference

abort

IE 3 NN 4 Moz all Saf all Op all DOM 2

Bubbles: No; Cancelable: No

Fires if an `img` element's content fails to complete loading due to user interruption (e.g., clicking **Stop** or rapidly navigating to another page) or other failure (e.g., timeout due to network traffic). The W3C DOM applies this event only to the `object` element, which, in the W3C standards view (but not yet widely supported in browsers), is the desired way to embed an image into a page.

Typical Targets The `img` element.

activate

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

Bubbles: Yes; Cancelable: No

Fires when an object becomes the active object. Giving an object focus makes it active, but a rendered element can be the active element without having focus. Only one element at a time may be active. See the `setActive()` method of shared objects in Chapter 2. If an element has received focus, the `activate` event fires before the `focusin` and then the `focus` events.

Typical Targets All rendered elements, plus the `document` and `window` objects.

afterprint, beforeprint

IE 5(Win) NN n/a Moz n/a Saf n/a Op n/a DOM n/a

Bubbles: No; Cancelable: No

Fires after the user clicks the **Print** button in the Print dialog box before content is assembled for the printer (`beforeprint`) and after the data has been sent to the printer (`afterprint`). You can use these events to trigger functions that modify a style sheet or other content rendering of a page (so that a potentially different-looking page reaches the printer) and then restore the page for viewing on the screen. This technique can work in lieu of style sheet media settings.

Typical Targets The `body` and `frameset` elements, plus the `window` object.

afterupdate

IE 4(Win) NN n/a Moz n/a Saf n/a Op n/a DOM n/a

Bubbles: Yes; Cancelable: No

Fires after data being sent to a writable data source object (through the IE data binding mechanism) has successfully updated the database.

Typical Targets Elements that accept data input and support data binding.

beforeactivate

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

Bubbles: Yes; Cancelable: Sometimes

Fires just before an object is to become the active object. Giving an object focus makes it active, but a rendered element can be the active element without having focus. Only one element at a time may be active. See the `setActive( )` method of shared objects in Chapter 2. If an element received focus, related events fire in the following sequence: `beforeactivate`, `activate`, `focusin`, and `focus`.

If you cancel the `beforeactivate` event, the element does not become active, nor does it receive focus, but only if the intended focus action occurs from explicit user action (clicking and tabbing). An element blocked from receiving focus causes the focus to go to another element: to the next focusable element in tabbing order (when the user tabs to the blocked element) or to the next outermost focusable parent element in the document tree (when a user clicks on the blocked element). Activating or giving focus to an element via the `setActive( )` or `focus( )` method cannot be blocked by canceling this event.

Typical Targets All rendered elements, plus the `document` and `window` objects.

beforecopy

IE 5(Win) NN n/a Moz n/a Saf 1.3/2 Op n/a DOM n/a

Bubbles: Yes; Cancelable: Yes

Fires just before a user-initiated **Copy** command (via the **Edit** menu, a keyboard shortcut, or a context menu) completes the task of moving the selected content to the system clipboard. At this point in the copy sequence, a function invoked by this event handler can perform additional or substitute processing for the normal system copy action. For example, additional information from the element, such as effective style information of the element containing selected text, can be preserved in the IE `clipboardData` object (see Chapter 2) for later processing with the help of the `beforepaste` event handler. Canceling the `beforecopy` event does not prevent user copying of a selection.

In Internet Explorer, the "before" events fire when the user displays the context menu (i.e., before any menu item is chosen). The events fire in the sequence `beforecut`, `beforecopy`, and `beforepaste`. Moreover, with the context menu approach, the three-event sequence fires twice.

Note that in Safari 1.3/2, you must pass `onbeforecopy` (rather than `beforecopy`) to `addEventListener( )` to bind the event.

Typical Targets Rendered elements except form controls.

beforecut

IE 5(Win) NN n/a Moz n/a Saf 1.3/2 Op n/a DOM n/a

Bubbles: Yes; Cancelable: Yes

Fires just before a user-initiated **Cut** command (via the **Edit** menu, a keyboard shortcut, or a context menu) completes the task of removing the content from its current location and moving the selected content to the system clipboard (assuming the browser is in edit mode

for body content). At this point in the cut sequence, a function invoked by this event handler can perform additional or substitute processing for the normal system cut action. For example, additional information from the element, such as effective style information of the element containing selected text, can be preserved in the IE `clipboardData` object (see Chapter 2) for later processing with the help of the `beforepaste` event handler. Canceling the `beforecut` event does not prevent user cutting of a selection.

See `beforecopy` for additional browser-specific notes.

Typical Targets All rendered elements.

beforedeactivate

IE *5.5* NN *n/a* Moz *n/a* Saf *n/a* Op *n/a* DOM *n/a*

Bubbles: Yes; Cancelable: Yes

Fires just before an object is about to yield activation to another object because the user clicked on another element, tabbed to another element, or because a script invoked the `setActive()` or `focus()` method of another element. If an element has focus and is the active element, the following event sequence fires en route to losing focus: `beforedeactivate`, `deactivate`, and `blur`. Because `beforedeactivate` is cancelable (but `deactivate` is not), cancel this event to prevent an element from deactivating or losing focus—provided you have a good reason to do this other than annoying your visitors.

Typical Targets All rendered elements, plus the `document` and `window` objects.

beforeeditfocus

IE *5(Win)* NN *n/a* Moz *n/a* Saf *n/a* Op *n/a* DOM *n/a*

Bubbles: Yes; Cancelable: Yes

Fires just before an editable element receives official focus by a user clicking or tabbing to the element. Editable elements include text-oriented form controls and body elements set to be editable (see the IE 5.5 `contentEditable` property of all elements in Chapter 2). A function invoked from this event handler can perform additional scripted actions, such as setting the color of the element text, before the user begins editing the content.

Typical Targets Text form controls; rendered elements in edit mode (IE 5.5 or later); content governed by the DHTML Editing ActiveX control (see *http://msdn.microsoft.com/workshop/browser/mshtml/*).

beforepaste

IE *5(Win)* NN *n/a* Moz *n/a* Saf *1.3/2* Op *n/a* DOM *n/a*

Bubbles: Yes; Cancelable: Yes

Fires just before a user-initiated **Paste** command (via the **Edit** menu, a keyboard shortcut, or a context menu) completes the task of pasting the content from the system clipboard to the current selection. If you are trying to paste custom information from the `clipboardData` object (saved there in an `beforecopy`, `copy`, `beforecut`, or `cut` event handler), you need to have the `beforepaste` and `paste` event handler functions working together. Set `event.returnValue` to

false in the beforepaste event handler so that the **Paste** item in the **Edit** (and context) menu is activated, even for a noneditable paste target. When the user selects the **Paste** menu choice, your paste event handler retrieves information from the clipboardData object and perhaps modifies the selected element's HTML content:

```
function handleBeforePaste( ) {
    event.returnValue = false;
}
function handlePaste( ) {
    if (event.srcElement.className == "OK2Paste") {
        event.srcElement.innerText = clipboardData.getData("Text");
    }
}
```

In the above paste operation, the system clipboard never plays a role because your scripts handle the entire data transfer—all without having to go into edit mode.

See beforecopy for additional browser-specific notes.

Typical Targets All rendered elements and the document object.

beforeprint

See afterprint.

beforeunload

IE *4(Win)/5(Mac)* NN *n/a* Moz *1.7* Saf *1.3/2* Op *n/a* DOM *n/a*

Bubbles: No; Cancelable: Yes

Fires just before the current document begins to unload due to impending navigation to a new page, form submission, or window closure. This event fires before the unload event, and gives your scripts and users a chance to cancel the unload action. Some of this activity is automatic to prevent nefarious scripts from trapping users on a page.

In the beforeunload event handler, assign a string to the event.returnValue property to force IE and Mozilla 1.8 or later to display a dialog box that lets the user choose whether the page should stay where it is, or whether the navigation or window closure action that the user requested continues as expected. The string assigned to the event property becomes part of the dialog box message (other text in the message is hard-wired by the browser and may not be removed or modified). The resulting action is controlled by the user's button choice in the dialog box.

Typical Targets The body and frameset elements, plus the window object.

beforeupdate

IE *4(Win)* NN *n/a* Moz *n/a* Saf *n/a* Op *n/a* DOM *n/a*

Bubbles: Yes; Cancelable: Yes

Fires just prior to sending data to a writable data source object (through the IE data binding mechanism). You can perform data validation and cancel the update.

Typical Targets Elements that accept data input and support data binding.

blur

IE 3 NN 2 Moz all Saf all Op all DOM 2

Bubbles: No; Cancelable: No

Fires after the current element loses focus (due to some other element receiving focus) or invoking the `blur( )` method of the current element. The `blur` event fires before the `focus` event in the other element.

Avoid using the `blur` event in text input fields to trigger form validation, especially if the validation routine displays an alert dialog box upon discovering an error. Interaction among the `blur` and `focus` events, along with the display and hiding of an alert dialog box can put you into an infinite loop. Use `change` instead.

Although the `blur` event has been supported for form controls and `window` objects since the early days of scriptable browsers, modern browsers can fire the event on virtually any other rendered element, provided the `tabindex` attribute is set for the element. Note that IE for Windows is known to omit firing the `blur` event on `window` objects.

Typical Targets For all browsers, `input` (of type text and password), `textarea`, `select`, and `window` objects; for IE 5 or later and W3C DOM browsers, add any rendered element for which the `tabindex` attribute is assigned a value.

bounce

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

Bubbles: No; Cancelable: Yes

Fires each time the text in a `marquee` element, whose behavior is set to `alternate`, touches a boundary and changes direction.

Typical Targets The `marquee` element.

cellchange

IE 5(Win) NN *n/a* Moz *n/a* Saf *n/a* Op *n/a* DOM *n/a*

Bubbles: Yes; Cancelable: No

Fires on the element hosting a data binding data source object (usually the `object` element) each time data in the remote database changes its value.

Typical Targets The `object` and `applet` elements.

change

IE 3 NN 2 Moz all Saf all Op all DOM 2

Bubbles: No (IE); Yes (Others); Cancelable: Yes (IE); No (Others)

Fires when a text-oriented form control or `select` element loses focus *and* its content or chosen item is different from what it was when the element most recently gained focus. Use this event in text-type `input` and `textarea` elements to validate an entry for that one field.

But also include form-wide validation with the `form` element's `submit` event handler. This event fires before the `blur` event.

Typical Targets Text-type `input`, `textarea`, and `select` elements.

click

IE 3 NN 2 Moz all Saf all Op all DOM 2

Bubbles: Yes; Cancelable: Yes

Fires after the user effects a mouse click or equivalent action. Click equivalents occur naturally on focusable elements (buttons and links for most browsers) by pressing the **Enter** key (and frequently the spacebar) when the item has focus. In modern browsers that support the `accesskey` attribute, typing the access key combination also triggers a click equivalent.

For mouse click actions, the `click` event fires only if the mouse button is pressed and released with the pointer atop the same element. In that case, the primary mouse events fire in this order: `mousedown`, `mouseup`, and `click`.

An `event` object created from a mouse event has numerous properties filled with details such as coordinates of the click and whether any modifier keys were held down during the event. Information about the button used is more reliably accessed through the `mousedown` or `mouseup` event. The event handler function can inspect these properties as needed.

Although the `click` event has been supported for button-oriented form controls and link objects since the early days of scriptable browsers, modern browsers can fire the event on virtually any other rendered element. Note that in Mozilla versions prior to 1.4 and Safari prior to 1.3, mouse events can fire on child text nodes of container-type elements, meaning that the `event` object's `target` property references the node, rather than the element. See Online Section VI for details about the impact of this behavior and cross-browser solutions.

Typical Targets For all browsers, `input` (of type `button`, `radio`, `checkbox`, `reset`, and `submit`), `a`, and `area` objects; Version 4 and later support the event for the `document` and `window` objects; for IE 4 or later and most W3C DOM browsers (not Opera 9), add any rendered element, as well as text nodes for Mozilla prior to version 1.4.

contextmenu

IE 5(Win) NN n/a Moz all Saf all Op n/a DOM n/a

Bubbles: Yes; Cancelable: Yes

Fires after the user clicks the right mouse button (or the button designated the secondary mouse button in the mouse control panel). This mouse button displays the context menu for the item beneath the pointer. For Mozilla prior to 1.4 and Safari prior to 1.3, the event target could be a text node. For all other supporting browsers and versions, the target (or IE `event.sourceElement`) is the containing element. To block the display of the context menu (and perhaps display a custom one of your own design via DHTML), set `event.returnValue` to `false` in the `contextmenu` event handler. While hiding the context menu may make it more difficult for users to view the source of a page or save an image (assuming you have already opened a document in a window bereft of the menubar), it is not a foolproof

way to guard against determined users capturing your page's content. Any scripted solution fails the instant the user disables scripting.

Typical Targets All rendered elements and the document object.

controlselect

IE *5.5* NN *n/a* Moz *n/a* Saf *n/a* Op *n/a* DOM *n/a*

Bubbles: Yes; Cancelable: Yes

Fires when the user selects an editable element (not its content) while the page is in edit mode. See move for a demonstration of this event.

Typical Targets All rendered elements and the document object.

copy

IE *5(Win)* NN *n/a* Moz *n/a* Saf *1.3/2* Op *n/a* DOM *n/a*

Bubbles: Yes; Cancelable: Yes

Fires after the user initiates the **Copy** command (via the **Edit** menu, a keyboard shortcut, or a context menu) to place a copy of the selected content into the system clipboard. An event handler function for this event can supplement the copy action by placing additional data of your choice into the clipboardData object (which the paste event handler can read and handle as needed).

To give users access to a **Copy** menu command for an otherwise uneditable element, set event.returnValue to false in the beforecopy event handler for the same object as the copy event handler. On the other hand, to prevent user copying of body content, set event.returnValue to false for the copy event handler. Just don't regard this tactic as a foolproof way to prevent users from copying your prized content.

Note that in Safari 1.3/2, you must pass oncopy (rather than copy) to addEventListener( ) to bind the event.

Typical Targets Rendered elements except form controls.

cut

IE *5(Win)* NN *n/a* Moz *n/a* Saf *1.3/2* Op *n/a* DOM *n/a*

Bubbles: Yes; Cancelable: Yes

Fires after the user initiates the **Cut** command (via the **Edit** menu, a keyboard shortcut, or a context menu) to place a copy of the selected content into the system clipboard. To cut body content, the containing element must be in edit mode (see the shared contendEditable property in Chapter 2). An event handler function for this event can supplement the cut action by placing additional data of your choice into the clipboardData object (which the paste event handler can read and handle as needed).

To give users access to a **Cut** menu command for an otherwise uneditable element, set event.returnValue to false in the beforecut event handler for the same object as the cut event handler. On the other hand, to prevent user cutting of body or form control content, set event.returnValue to false for the cut event handler.