



Internet Explorer 8 Technology Overview for Developers

Version 4.0, 18 March 2009

The information contained in this document represents the current view of Microsoft Corporation on the issues discussed as of the date of publication. Because Microsoft must respond to changing market conditions, it should not be interpreted to be a commitment on the part of Microsoft, and Microsoft cannot guarantee the accuracy of any information presented after the date of publication.

This White Paper is for informational purposes only. MICROSOFT MAKES NO WARRANTIES, EXPRESS, IMPLIED OR STATUTORY, AS TO THE INFORMATION IN THIS DOCUMENT.

Complying with all applicable copyright laws is the responsibility of the user. Without limiting the rights under copyright, no part of this document may be reproduced, stored in or introduced into a retrieval system, or transmitted in any form or by any means (electronic, mechanical, photocopying, recording, or otherwise), or for any purpose, without the express written permission of Microsoft Corporation.

Microsoft may have patents, patent applications, trademarks, copyrights, or other intellectual property rights covering subject matter in this document. Except as expressly provided in any written license agreement from Microsoft, the furnishing of this document does not give you any license to these patents, trademarks, copyrights, or other intellectual property.

Unless otherwise noted, the companies, organizations, products, domain names, e-mail addresses, logos, people, places, and events depicted in examples herein are fictitious. No association with any real company, organization, product, domain name, e-mail address, logo, person, place, or event is intended or should be inferred.

© 2009 Microsoft Corporation. All rights reserved.

Microsoft, ActiveX, Encarta, Internet Explorer, the Internet Explorer logo, MSDN, SmartScreen, Windows, Windows Live, Windows Server, Windows Vista, and the Windows logo are trademarks of the Microsoft group of companies.

All other trademarks are property of their respective owners.

Table of Contents

Executive Overview	1
Real World Interoperability and Compatibility	1
Standards Improvements	1
The Default Layout Mode.....	3
ActiveX.....	3
Adaptive Page Zoom.....	5
Improved Printing.....	6
W3C ARIA Support.....	7
File Upload Control.....	7
Faster and Easier	8
Integrated Developer Tools.....	8
CSS 2.1 Compliance	8
Acid2 Test Compliance	9
HTML and DOM Improvements	9
DOM Prototypes.....	10
Ajax Navigations	10
DOM Storage	11
Connection Events.....	11
Cross-domain Requests (XDR)	11
Cross-document Messaging (XDM)	12
Safer Mashups: HTML Sanitization.....	12
Safer Mashups: JSON Sanitization.....	12
Selectors API.....	13
Data URI Support.....	13
Improved Namespace Support.....	14

MIME-Handling Changes	14
Reach Beyond the Page	15
Accelerators	15
Web Slices	16
Enhanced Instant Search	17
End-user and IT Pro Features	17
Crash Recovery	17
Performance and Memory Improvements	18
Managing Add-ons	18
Domain Name Highlighting	21
Enhanced Add-on Management	21
Data Execution Prevention	22
Conclusion	22

Important Note For the most up-to-date developer information, see the [Windows® Internet Explorer® Developer Center on MSDN®](http://www.msdn.com/ie) [http://www.msdn.com/ie].

Executive Overview

The evolution of the Internet has facilitated new sources of rich information and more ways to access it. That growth has introduced a new set of opportunities, immersive experiences, online services, and standards to the Web. With this intensity and reliance, Web developers and designers face an evolving set of needs. To that end, the development process of Windows Internet Explorer 8 focused on three major themes:

- Provide **real-world interoperability** with other browsers and compatibility for existing sites
- Make Web development **faster and easier** thanks to built-in developer tools
- Enable experiences that **reach beyond the page** through new browser features that effortlessly connect users to innovative Web services

Internet Explorer 8 includes a host of new features and enhancements—all driven from real-world scenarios and customer feedback—that puts the Web at your service. This document provides a high-level overview of new and enhanced features in Internet Explorer 8, and is tailored for Web developers and designers like you.

Real World Interoperability and Compatibility

Each version of Internet Explorer has delivered better performance, improved reliability, and enhanced security over its predecessor. Internet Explorer 8 brings even more enrichments to the core platform and architecture, offering improved performance, safety, reliability, and compatibility.

Standards Improvements

With past versions of Internet Explorer, developers and designers have noted that Internet Explorer has had its own interpretation of some Web standards and the way the browser handles some HTML, Cascading Style Sheets (CSS), and scripting functionality. In many cases, interpretations were decided upon because Internet Explorer supported certain features before corresponding standards were finalized. If those standards changed as they were finalized, Internet Explorer's implementation may have varied from what the standard specifies.

The Internet Explorer team has stood behind our application compatibility commitments whenever possible because we know there are customers with critical dependencies on legacy behavior. For that reason, we have supported the legacy Internet Explorer model whenever feasible so that sites developed to it would continue to behave as expected in newer versions of Internet Explorer. Moving forward, we've placed the decision to support legacy behaviors versus strict standards into the hands of Web developers by enabling you to select the rendering mode on a page-by-page basis.

Internet Explorer 8 ships with multiple rendering modes that may be set by using the `X-UA-Compatible` header. The modes are summarized in Table 1.

Table 1. The compatibility modes in Internet Explorer 8.

Compatibility Mode Value	Render Behavior
IE=5	“Quirks” mode
IE=7	Internet Explorer 7 “Strict” mode
IE=EmulateIE7	Mode determined by the <code>!DOCTYPE</code> declaration: <ul style="list-style-type: none">• Quirks mode <code>!DOCTYPEs</code> result in Quirks mode• Standards mode <code>!DOCTYPEs</code> result in Internet Explorer 7 Strict mode
IE=8	Internet Explorer 8 Standards mode
IE=EmulateIE8	Mode determined by the <code>!DOCTYPE</code> declaration: <ul style="list-style-type: none">• Quirks mode <code>!DOCTYPEs</code> result in Quirks mode• Standards mode <code>!DOCTYPEs</code> result in Internet Explorer 8 Standards mode.
IE=edge	Uses latest standards that Internet Explorer 8 and any future versions of the browser support. Not recommended for production sites.

This value may be set as an HTTP response header or by adding it to the head tag, such as:

```
<meta http-equiv="X-UA-Compatible" content="IE=EmulateIE7" >
```

By default, Internet Explorer 8 renders content in Internet Explorer 8 Standards mode. However, you can have multiple values in the content attribute to support multiple standards. Developers who want to continue to have their sites render in Internet Explorer 7’s Strict mode should add the following to their head block:

```
<meta http-equiv="X-UA-Compatible" content="IE=EmulateIE7" >
```

Note that the setting in the document always overrides the HTTP header. For more information, see the following Microsoft® Help and Support article: “[Some Web sites may not be displayed correctly or work correctly in Windows Internet Explorer 8](http://go.microsoft.com/fwlink/?LinkId=146934)” [http://go.microsoft.com/fwlink/?LinkId=146934].

For more information on document compatibility, see “[Defining Document Compatibility](http://go.microsoft.com/fwlink/?LinkId=131990)” [http://go.microsoft.com/fwlink/?LinkId=131990].

The Default Layout Mode

Internet Explorer 8 supports multiple layout modes. The choice of layout modes is based on official Web standards and enables developers to select the most appropriate standard so that content will display correctly in the browser and applications will behave as expected. The behavior is as specified in Table 2.

Table 2. Internet Explorer 8 layout modes.

Page Content Declaration	Layout Mode
Known standards !DOCTYPEs and unknown !DOCTYPEs	IE8 Standards mode Note Content will display using Internet Explorer 7 Strict mode if <code>IE=7</code> or <code>IE=EmulateIE7</code> is declared.
<code>public identifier "-//W3C//DTD XHTML 1.0 Transitional//EN"</code> <code>public identifier "-//W3C//DTD XHTML 1.0 Frameset//EN"</code> <code>public identifier "-//W3C//DTD HTML 4.01 Transitional//EN", with a system identifier</code> <code>public identifier "-//W3C//DTD HTML 4.01 Frameset//EN", with a system identifier</code>	"Almost Standards" mode All content is interpreted as in Internet Explorer 8 Standards mode except for line heights; particularly important for legacy sites that contain images within table cells and expect no whitespace around the images. Note These !DOCTYPEs trigger Internet Explorer 7 Strict mode if <code>IE=7</code> or <code>IE=EmulateIE7</code> is declared.
Quirks mode !DOCTYPEs (includes the absence of a !DOCTYPE)	Quirks mode

Note For more information on !DOCTYPEs, see [the !DOCTYPE element reference article](http://msdn2.microsoft.com/en-us/library/ms535242.aspx) [http://msdn2.microsoft.com/en-us/library/ms535242.aspx].

Displaying Web pages in IE8 Standards mode brings developers closer to the point at which content will be written once and work consistently across modern browsers.

The `meta` compatibility tag attempts to solve backward compatibility problems by providing Web developers a way to signal the desired layout mode regardless of !DOCTYPE.

ActiveX

Internet Explorer 8 enables better management of Microsoft ActiveX® controls, such as where and how they can load, as well as which users can load them.

Internet Explorer 8 on Windows Vista® and Windows Server® 2008 introduces the ability to package ActiveX controls for installation to users' own profiles without administrator involvement. This solution addresses concerns raised by users and administrators of previous versions of Internet Explorer that the administration of controls was inconvenient. In the event that a user unknowingly installs a malicious

ActiveX control, the impact is limited to that user's profile. Additionally, most existing ActiveX controls will not have to be rewritten to benefit from this feature, although they will require repackaging.

ActiveX controls embedded as Web objects are presented to the user as add-ons that may be restricted for use on specific Web sites. When a Web site requests the loading of an add-on, the Information Bar provides users the ability to restrict the control to only the current Web site, or allow any Web site to use the control. Users can easily make changes to this behavior through the new Internet Explorer Add-ons Manager.

For more information on ActiveX improvements in Internet Explorer 8, see [this post](http://blogs.msdn.com/ie/archive/2008/05/07/ie8-security-part-ii-activex-improvements.aspx) [http://blogs.msdn.com/ie/archive/2008/05/07/ie8-security-part-ii-activex-improvements.aspx] on the Internet Explorer Team Blog.

Per-user (Non-admin) ActiveX

Running Internet Explorer 8 in Windows Vista, a standard user may install specially packaged ActiveX controls in their own user profile without requiring administrative privileges. This improvement makes it easier for an organization to realize the full benefit of User Account Control (UAC) by enabling standard users to install ActiveX controls used in their day-to-day browsing. If a user happens to install a malicious ActiveX control, the overall system will be unaffected, as the control was installed only under the user's account.

Per-user ActiveX was designed with compatibility in mind. Most existing ActiveX controls will not need to be rewritten to benefit from this feature; only a change to the ActiveX control's .INF file within the .CAB is required. As in Internet Explorer 7, when a Web page attempts to install a control, an Information Bar is displayed to the user. By clicking the Information Bar, users can choose to either install the control machine-wide, or install it only for their own user account. The available options depend on Group Policy settings for per-user ActiveX installations and whether the control has been packaged to allow per-user installation.

For more information, see "[Non-Admin ActiveX Controls](http://go.microsoft.com/fwlink/?LinkId=125781)" [http://go.microsoft.com/fwlink/?LinkId=125781].

Per-site ActiveX

When a user navigates to a Web site containing an ActiveX control, Internet Explorer 8 performs a number of checks, including a determination of where the control is permitted to run. This check is referred to as Per-site ActiveX, a defense mechanism to help prevent malicious repurposing of controls. Using Per-site ActiveX security hardens the browser. By default, an ActiveX control may only run from the domain in which it was installed. If a control is installed, but is not permitted to run on a specific Web site, an Information Bar appears asking the user whether or not the control should be permitted to run on the current Web site.

For more information, see "[Per-site ActiveX Controls](http://go.microsoft.com/fwlink/?LinkId=146958)" [http://go.microsoft.com/fwlink/?LinkId=146958].

Adaptive Page Zoom

Adaptive Page Zoom in Internet Explorer 8 enables users to enlarge or reduce the view of a Web page to improve readability. This feature is particularly useful on very large and very small displays because it allows for scaling of content while maintaining the intended layout of the page.

While the initial Internet Explorer 7 version of zoom was a “digital” page zoom, zoom in Internet Explorer 8 provides a higher quality, more predictable zooming experience known as an “adaptive” page zoom. While zooming, Internet Explorer 8 will resize the text and images and reflow the page to make it easier to read. This will eliminate horizontal scroll bars for the majority of scenarios and allows for persistent zoom states. As a developer, you may wish to understand the behavior of Adaptive Page Zoom and how it might affect your site’s design.

Internet Explorer 8 Adaptive Page Zoom is accomplished by scaling elements pre-layout. This is significantly different from Internet Explorer 7 Zoom behavior, which was analogous to “magnifying” the Web page—elements were scaled post layout, and then re-drawn on screen. Due to this important change, horizontal scrollbars are introduced only when the fixed width of the scaled content is greater than the width of the viewport. This is exactly like resizing regular layout on an un-zoomed Web page.

Text wrapping is also affected by this change. In Internet Explorer 7 Zoom, line lengths and line breaks were not recalculated as the zoom factor increased or decreased. This led to situations where text lines were either too small (resulting in too much white space) or too large (resulting in text runs that would go off the screen, requiring horizontal scrollbars). In Internet Explorer 8, line lengths are recalculated based on available space before the text is rendered on screen. Then, line breaks are inserted, taking the new lengths into account.

In addition, it is important to understand how common page elements and properties respond to zoom.

- **Fonts and text:** The glyph itself is not scaled. Rather, the font size is scaled and then the appropriate glyph is used. The important thing to note is that fonts do not scale linearly by design. For example, if text at 12pt is scaled to 110%, the resulting font size is calculated as 13.2pt. Since this font size does not exist, it is therefore rounded to the nearest available size, 13pt.
- **Fixed, auto, and relative sizing:** Dimension scaling is one of the most important improvements in Adaptive Page Zoom. Dimensions specified using absolute units (for example, in, cm, mm) or device and font dependent units (for example, px, ex, em) are scaled as per the zoom level. Therefore, at 200%, 100px becomes 200px and 20pt becomes 40pt. Content-dependent dimensions (percent and auto) are not scaled, as they are computed during layout. Therefore, at 200%, a width of 50% of the viewport does not become 100% of the same. This is a marked change from Zoom in Internet Explorer 7.
- **Positioning:** Positioned elements grow and shrink like in-flow elements. However, their new position is determined by the properties set and the offsets used. An absolutely positioned element, if offset to the left by 100px, will shift towards the right when zoomed in. It is possible

for the element to go off the screen. Similarly, floats will be positioned with respect to their container, per the normal rules of CSS. If the width of the container changes with zoom, then the position of the float will change. Zooming of adjacent floats is exactly like resizing the window—if the width of the viewport is not large enough to accommodate the floats, the last one in markup will drop to the next line.

- **DHTML properties:** In Internet Explorer 7 Zoom, some properties were treated as physical (such as mouse coordinates), while others were logical (offset). This implementation essentially required Web developers to be aware of or manually calculate the zoom state based on the property being used. With Internet Explorer 8 Adaptive Page Zoom, all DHTML properties are now assumed to be logical. This enables some key scenarios such as fly-out menus and “drag-and-drop.”

Getting Your Site Ready for Internet Explorer 8 Adaptive Page Zoom

Web developers should not expect to write any special code for Adaptive Page Zoom; all properties are logical, and scaling is purely internal. For developers who are interested in improving their site user experience with zoom, Microsoft recommends that you test the site with different zoom factors, resolutions, and window dimensions. Think about the following during testing:

- At what point do horizontal scrollbars appear? Does the user need to scroll to read a single line of text?
- Does content quickly go off screen because of fixed sizing and positioning?
- Does the `overflow:hidden` value on any elements make content inaccessible?
- Do fly-out menus adapt to available screen real estate, or do options go off the screen, making them inaccessible to the user?

For more information on zoom in Internet Explorer 8, see “[Making the Web Bigger: DPI Scaling and Internet Explorer 8](http://msdn.microsoft.com/en-us/library/cc849094(VS.85).aspx)” [http://msdn.microsoft.com/en-us/library/cc849094(VS.85).aspx].

Improved Printing

There are new printing features in Internet Explorer 8. Improved compliance with the CSS 2.1 specification in Internet Explorer 8 provides enhanced functionality and control in the print medium, giving developers greater control over how content is printed. In particular, support has been added for the following printing constructs:

- Correct behavior for `avoid` and `inherit` values for `page-break-after` and `page-break-before` properties
- `page-break-inside` property
- `widows` property
- `orphans` property

By using these CSS features, you can specify the margin area, portions of content that must be kept together, and much more, enabling you to greatly improve the readability of printed Web pages.

For more information on improved printing with CSS in Internet Explorer 8, see “[CSS How-to: Optimize Pages for Printing Using CSS](http://go.microsoft.com/fwlink/?LinkID=125786)” [http://go.microsoft.com/fwlink/?LinkID=125786].

W3C ARIA Support

Accessible Rich Internet Applications (ARIA) constitutes a syntax for defining how Web developers can mark up content with roles, states, and properties that browsers use when communicating with assistive technology. Thanks to ARIA, users with impairments can access Web sites with a rich interaction comparable to the originally intended experience.

Many third-party assistive technologies (ATs), such as Window-Eyes, implement Microsoft Active Accessibility (MSAA) application programming interfaces (APIs) to get and expose information about the user interface to end users. By exposing ARIA through MSAA in Internet Explorer 8, ATs that already use MSAA can also easily support ARIA. Note that only a portion of ARIA can be supported through MSAA’s functionality.

For more information on ARIA support in Internet Explorer 8, see “[Mapping ARIA Roles, States, and Properties to UI Automation](http://go.microsoft.com/fwlink/?LinkID=125799)” [http://go.microsoft.com/fwlink/?LinkID=125799].

File Upload Control

In the past, the HTML file upload control (`input type=file`) has been the source of a significant number of information disclosure vulnerabilities. To resolve these issues, two changes have been made to the behavior of the control. Internet Explorer 8 controls information exposed during file upload with its expanded file upload control and file locking on the Internet zone.

To block attacks that rely on “stealing” keystrokes to surreptitiously trick the user into typing a local file path into the control, the file path edit box is now read-only. The user must explicitly select a file for upload using the **File Browse** dialog box (shown in Figure 1).



Figure 1. The file path edit box is now read-only. Users must click the Browse button to select a file.

Additionally, the **Include local directory path when uploading files** URLAction has been set to **Disable** for the Internet zone. This change prevents leakage of potentially sensitive local file-system information to the Internet. For instance, rather than submitting the full path, such as the following:

C:\users\seanpurcell\documents\secret\image.png

Internet Explorer 8 will now submit only the filename:

image.png

Faster and Easier

Internet Explorer 8 delivers a better developer platform and the tools to back it up. With advancements in support for Ajax (Asynchronous JavaScript and XML) applications, as well as improvements that simplify the process of building cross-browser applications, Internet Explorer 8 enables developers to be more productive when building the most robust Web applications possible.

Integrated Developer Tools

Internet Explorer 7 supported a Developer Toolbar, which developers could download separately and run as an extension to the browser. Although many developers found it to be highly useful for testing, debugging, profiling, or rapidly prototyping a Web page, because it ran as an extension its performance was limited and it required a substantial memory footprint. Internet Explorer 8 replaces the Developer Toolbar with an integrated set of integrated developer tools that can be accessed by pressing F12 or by clicking **Developer Tools** on the **Tools** menu. Because the Developer Tools are an integral component of the browser, performance is improved and no memory is used when the tools are not running. The integrated Developer Tools make it faster and easier for developers to develop and troubleshoot rich content sites.

Developer Tools to edit and debug CSS and HTML, test and debug script, profile script performance, view or change the document object model (DOM), examine applied rules, and trace the origin of style values—all within a rich, visual environment that exposes the browser’s internal representation of a Web page *as it runs*, instead of just its source code. In this way, you can rapidly iterate within Internet Explorer 8—for example, changing an attribute or a style rule and immediately viewing the results—and then save the HTML tree and CSS files to disk as text files for integration back into the original application. The Developer Tools also make it easy to toggle between the different layout engines included in Internet Explorer 8, enabling you to quickly and easily identify changes that must be made for sites to work well with older versions of Internet Explorer.

By eliminating the need to tweak source code in one program, save it, and then refresh the browser to view the results, the Developer Tools enable Web developers to rapidly prototype and test a new page, debug a problem, or just learn more about Web development.

The Developer Tools contain new JavaScript debugging tools with advanced features that allow developers to set breakpoints, step in/over code, watch variables, check locals, enter commands in the immediate window, and so on. It is just like a traditional code debugger but with JavaScript.

For more information the Internet Explorer 8 Developer Tools, see “[Discovering the Internet Explorer 8 Developer Tools](http://go.microsoft.com/fwlink/?LinkID=125790)” [http://go.microsoft.com/fwlink/?LinkID=125790].

CSS 2.1 Compliance

Cascading Style Sheets (CSS) is a simple mechanism for adding styles (such as fonts, colors, spacing, and positioning) to Web documents. Before the introduction of CSS, style-related properties were set directly within HTML, which produced many challenges for Web authors. Today, CSS has evolved to

include a series of design patterns that allow authors to separate data from the display logic, thus improving markup organization and facilitating easier site maintenance.

Enabling Web developers and designers to write their pages once and have them render properly across all browsers is the main goal of CSS compliance. The CSS 2.1 Specification reached World Wide Web Consortium (W3C) Candidate Recommendation status as of July 17, 2007. The Internet Explorer 8 Standards mode layout engine was built with the CSS 2.1 spec in hand and full compliance in mind.

By shipping multiple engines and defaulting to Internet Explorer 8 Standards mode, Internet Explorer 8 will encourage developers to support the latest standards while allowing authors to fall back to Internet Explorer 7- compatible (or earlier) behavior if necessary.

Internet Explorer 8 is fully CSS 2.1 compliant. For more information on CSS compliance in Internet Explorer 8 and changes since Internet Explorer 7, see “[CSS Improvements in Internet Explorer 8](http://go.microsoft.com/fwlink/?LinkID=125787)” [http://go.microsoft.com/fwlink/?LinkID=125787].

Acid2 Test Compliance

Compliance with accepted standards in Internet Explorer 8 is evidenced by its passing the Web Standards Project’s Acid2 test, which uses a broad array of HTML and CSS commands to expose layout errors. However, in working to pass this test, Microsoft has encountered instances where the standards seem ambiguous or confusing. As part of its commitment to supporting Web standards, Microsoft has provided more than 1,000 certification test cases to the W3C for use by Web developers and other Web browser providers. The test cases are available for all to view in the “[Windows Internet Explorer Testing Center](http://go.microsoft.com/fwlink/?LinkID=110283)” [http://go.microsoft.com/fwlink/?LinkID=110283]. By sharing its interpretation of specifications for Web standards with the community in such a way, Microsoft is inviting discussion and advancing the goals of standardized tests and a fully standardized Web.

More information on Internet Explorer 8 and the Acid2 test can be found in the following post to the Internet Explorer Team Blog: “[Internet Explorer 8 and Acid2: A Milestone](http://blogs.msdn.com/ie/archive/2007/12/19/internet-explorer-8-and-acid2-a-milestone.aspx)” [http://blogs.msdn.com/ie/archive/2007/12/19/internet-explorer-8-and-acid2-a-milestone.aspx].

HTML and DOM Improvements

The HTML language uses elements to represent structure and meaning in a document. To assist developers in taking full advantage of the elements offered by HTML 4, Internet Explorer 8 provides upgraded support for several presentational elements. For example, the `q` element represents an inline quoted string, and the `object` element may now represent any “object,” including images. Through improved support for these and other HTML elements, Web developers can deliver more expressive and accessible HTML markup.

Some of the feedback regarding DOM functionality in Internet Explorer has been related to its attribute handling, which has not traditionally been compatible with the implementations used by other browsers. Internet Explorer 8 fixes many of the cross-browser inconsistencies, such as:

- Separate URL handling for attributes and properties such that relative URLs can be retrieved by using the attribute and absolute URLs can be retrieved by using the property.
- The element's attribute array modifiers have been fixed so that the **get/set/removeAttribute** implementations are now compatible with those of other browsers.
- The behavior of the **attributes** object reports the correct number of "attributes" associated with an element.
- Default attributes for HTML are supported so that they always exist on elements whether specified in the markup or not, which aligns with the XHTML 1.1 DTD.

For more information on improvements to HTML and DOM support in Internet Explorer 8, see "[HTML Enhancements in Internet Explorer 8](http://go.microsoft.com/fwlink/?LinkId=110271)" [http://go.microsoft.com/fwlink/?LinkId=110271].

DOM Prototypes

JavaScript, the principle scripting language used on the Web, is often used to code innovative new features that supplement HTML and CSS, cater to Web site-specific scenarios, normalize differences between browsers, and so on.

To further empower Web developers with the programming tools necessary to build new JavaScript scenarios that innovate, extend, and build upon the Web platform, Internet Explorer 8 offers a collection of features that extend some of JavaScript's advanced functionality into the Document Object Model (DOM), called DOM prototypes.

An accessor property, also called a getter/setter property, is a new type of JavaScript property available in Internet Explorer 8. By using accessor properties, Web developers can create or customize dynamic data-like properties that execute JavaScript code when their values are accessed or modified. The DOM prototype hierarchy defines all of its properties as built-in accessors. Web developers can also customize DOM built-in accessors to fine-tune the default behavior of the DOM.

For more information on DOM Prototypes in Internet Explorer 8, see "[Document Object Model Prototypes, Part 1: Introduction](http://go.microsoft.com/fwlink/?LinkId=128221)" [http://go.microsoft.com/fwlink/?LinkId=128221] and "[Document Object Model Prototypes, Part 2: Accessor \(getter/setter\) Support](http://go.microsoft.com/fwlink/?LinkId=139911)" [http://go.microsoft.com/fwlink/?LinkId=139911].

Ajax Navigations

One of the significant benefits of using Ajax is the ability to update page content without traditional page navigations. This can be problematic in some scenarios because components like the Address bar, **Back** and **Forward** buttons, and the travelog (browser history) only update after page navigation. As a result, Ajax content updates don't get saved as navigations, browser components don't get updated, and end users are sometimes left confused as to why browser features are stuck on older content. While some Web sites work around this limitation by navigating a hidden **iframe** element [http://msdn.microsoft.com/en-us/library/ms535258(VS.85).aspx] while updating content through Ajax, this can result in degraded performance.

Pages rendered in Internet Explorer 8 Standards mode can handle navigations within their Ajax applications and update the travelog and Address bar by taking advantage of functionality added to the [window.location.hash](http://msdn.microsoft.com/en-us/library/ms533775(VS.85).aspx) property [http://msdn.microsoft.com/en-us/library/ms533775(VS.85).aspx].

For more information on Ajax navigations, see “[Introducing Ajax Navigations](http://go.microsoft.com/fwlink/?LinkId=146959)” [http://go.microsoft.com/fwlink/?LinkId=146959], and download the [Ajax Hands-on Lab](http://go.microsoft.com/fwlink/?LinkId=128984) on MSDN Code Gallery [http://go.microsoft.com/fwlink/?LinkId=128984].

DOM Storage

Today, Web pages use the [document.cookie](http://msdn.microsoft.com/en-us/library/ms533693.aspx) property [http://msdn.microsoft.com/en-us/library/ms533693.aspx] to store data on the local machine. Cookies are limited in capability by the fact that only 50 key/value pairs can be stored per domain. Furthermore, the cookie programming model is cumbersome and requires that you to parse the entire cookie string for data.

Specified in the W3C’s HTML 5 Draft, DOM Storage objects provide a much simpler global and session storage model for structured data on the client side.

For more information on DOM Storage, see “[Introduction to DOM Storage](http://go.microsoft.com/fwlink/?LinkId=146960)” [http://go.microsoft.com/fwlink/?LinkId=146960], and download the [Ajax Hands-on Lab](http://go.microsoft.com/fwlink/?LinkId=128984) on MSDN Code Gallery [http://go.microsoft.com/fwlink/?LinkId=128984].

Connection Events

As outlined in the W3C’s HTML 5 Draft, connection events allow Web sites to check whether the user is connected to the network. These events can be useful to the developers of dynamic applications because they enable the seamless handling of network connection changes. For example, if the network connection is lost, the application can use local cached data and inform users that it is unable to connect to network sources. When network access is restored the application can smoothly apply data updates and refresh current data.

Connection events can be raised on the `body` element of a document. In addition, the status may be checked at any time in script.

For more information on connection events, see “[Connectivity Enhancements in Internet Explorer 8](http://go.microsoft.com/fwlink/?LinkId=146961)” [http://go.microsoft.com/fwlink/?LinkId=146961], and download the [Ajax Hands-on Lab](http://go.microsoft.com/fwlink/?LinkId=128984) on MSDN Code Gallery [http://go.microsoft.com/fwlink/?LinkId=128984].

Cross-domain Requests (XDR)

Cross-domain communication is an integral part of Ajax development and mashup applications. Making requests for data across domains can be challenging with today’s browsers, partly due to the prevailing interest of protecting users from cross-site attacks.

In Internet Explorer 8, Web pages can simply make a cross-domain data request (XDR) within the browser by using the new [XDomainRequest](http://msdn.microsoft.com/en-) object [http://msdn.microsoft.com/en-

us/library/cc288060(VS.85).aspx] and without server-to-server requests. XDRs require mutual consent between the Web page and the server. You can initiate a cross-domain request in your Web page by creating an **XDomainRequest** object off the **window** object and opening a connection to a particular domain. The browser will request data from the domain's server by sending an **Origin** header with the value of the origin. It will only complete the connection if the server responds with an **Access-Control-Allow-Origin: *** header. These semantics are part of the W3C's Web Application Working Group's draft framework on client side cross domain that **XDomainRequest** integrates with.

For more information on cross-domain requests, see "[Introducing Cross-domain Request \(XDR\)](http://go.microsoft.com/fwlink/?LinkID=144259)" [http://go.microsoft.com/fwlink/?LinkID=144259], the [Ajax Hands-on Lab](http://go.microsoft.com/fwlink/?LinkID=128984) on MSDN Code Gallery [http://go.microsoft.com/fwlink/?LinkID=128984], the reference page for the **XDomainRequest** object [http://msdn.microsoft.com/en-us/library/cc288060(VS.85).aspx], and "[Client-side Cross-domain Security](http://go.microsoft.com/fwlink/?LinkID=141677)" [http://go.microsoft.com/fwlink/?LinkID=141677].

Cross-document Messaging (XDM)

Cross-document Messaging (XDM) provides the **window.postMessage** method [http://msdn.microsoft.com/en-us/library/cc197015(VS.85).aspx], which allows different domains to communicate with each other given mutual consent. This feature can be used to help guarantee that application mashups are safer. XDM provides you with a simpler, more performant mechanism for two-way cross-domain communication than embedding `iframe` objects or hosting scripts from another domain.

For more information on cross-document messaging, see the reference page for the **postMessage** method [http://msdn.microsoft.com/en-us/library/cc197015(VS.85).aspx], and download the [Ajax Hands-on Lab](http://go.microsoft.com/fwlink/?LinkID=128984) on MSDN Code Gallery [http://go.microsoft.com/fwlink/?LinkID=128984].

Safer Mashups: HTML Sanitization

Internet Explorer 8 exposes a new method on the **window** object named **toStaticHTML** [http://msdn.microsoft.com/en-us/library/cc848922.aspx]. When a string of HTML is passed to this function, any potentially executable script constructs are removed before the string is returned. Internally, this function is based on the same technologies as the server-side Microsoft Anti-Cross Site Scripting Library.

For more information on safer mashups with JSON sanitization in Internet Explorer 8, see [this post](http://blogs.msdn.com/ie/archive/2008/07/02/ie8-security-part-v-comprehensive-protection.aspx) [http://blogs.msdn.com/ie/archive/2008/07/02/ie8-security-part-v-comprehensive-protection.aspx] on the Internet Explorer Team Blog, plus the reference page for the **toStaticHTML** method [http://msdn.microsoft.com/en-us/library/cc848922.aspx].

Safer Mashups: JSON Sanitization

JavaScript Object Notation (JSON) is a lightweight string-serialization of a JavaScript object that is often used to pass data between components of a mashup. Unfortunately, many mashups use JSON insecurely, relying on the JavaScript **eval** method to "revive" JSON strings back into JavaScript objects,

potentially executing script functions in the process. Security-conscious developers instead use a JSON parser to ensure that the JSON object does not contain executable script, but there's a performance penalty for this.

Internet Explorer 8 implements the ECMAScript 3.1 proposal for native JSON-handling functions (which uses Douglas Crockford's `json2.js` API). The **JSON.stringify** method accepts a script object and returns a JSON string, while the **JSON.parse** method accepts a string and safely revives it into a JavaScript object. The new native JSON methods are based on the same code used by the script engine itself, and thus have significantly improved performance over non-native implementations. If the resulting object contains strings bound for injection into the DOM, the previously described **toStaticHTML** function can be used to prevent script injection.

For more information on safer mashups with JSON sanitization in Internet Explorer 8, see [this post](http://blogs.msdn.com/ie/archive/2008/07/02/ie8-security-part-v-comprehensive-protection.aspx) [http://blogs.msdn.com/ie/archive/2008/07/02/ie8-security-part-v-comprehensive-protection.aspx] on the Internet Explorer Team Blog.

Selectors API

Selectors are a query language for searching and “selecting” tags (elements) within a Web page. They are most commonly seen in CSS to “select” a group of elements to which certain properties will be applied.

In Internet Explorer 7, there is no way of “executing” a selector independently of CSS. The Selectors API in Internet Explorer 8 puts the power and flexibility of CSS selectors into the hands of the Web developer for rapid element lookups.

The Selectors API provides increased performance for non-linear element traversals and filters, an optimal choice considering the alternative of independent element lookups and manual script traversal of the DOM. This API also significantly reduces the amount of code required for lookups.

Internet Explorer 8's implementation of the Selectors API is based on the W3C Working Draft for Selectors at <http://www.w3.org/TR/selectors-api>.

For more information on the Selectors API in Internet Explorer 8, see “[Selecting Objects with JavaScript](http://go.microsoft.com/fwlink/?LinkID=110273)” [http://go.microsoft.com/fwlink/?LinkID=110273].

Data URI Support

Data URIs offer Web developers the opportunity to embed small external resources (like CSS files or images) directly into a URL on a Web page.

The primary use case for data URIs is the encapsulation of a binary file inside of a reference URL for a style sheet or image. Because the binary file can be represented inline as a string, it may be saved to the local store and retrieved at a later time without needing to fetch anything from the network.

Data URIs also offer an alternative to traditional URIs in that they don't require a separate download. This is advantageous in scenarios where sub-downloads are blocked or unavailable. Data URIs may be up to 32kb in length and may not be used as the source of a `frame`, `iframe`, or `script` element.

For information on Data URI support in Internet Explorer 8, see the reference page for the [data](http://go.microsoft.com/fwlink/?LinkID=125789) protocol [http://go.microsoft.com/fwlink/?LinkID=125789].

Improved Namespace Support

Microsoft Internet Explorer 5 through Internet Explorer 7 provided limited element namespace support, but that was primarily used to remove an element from being treated as native HTML by the parser. Through element namespacing, Web developers have the ability to apply “behaviors” to those elements through special HTML markup known as HTC (HTML Components) and COM developers the ability to apply “binary behaviors,” which are a type of ActiveX control. Only elements within a namespace can have a binary behavior applied to them.

Internet Explorer 8 improves its namespace support by reducing the amount of code necessary to instantiate and use a binary behavior. Once a user installs a namespace handler provided by a vendor, the handlers can be appropriately invoked and displayed as designed.

For more information on improved namespace support in Internet Explorer 8, see “[Improved Namespace Support](http://go.microsoft.com/fwlink/?LinkID=125793)” [http://go.microsoft.com/fwlink/?LinkID=125793].

MIME-Handling Changes

Each type of file delivered from a Web server has an associated MIME type (also called a “content-type”) that describes the nature of the content (for instance, image, text, or application). For compatibility reasons, Internet Explorer has a MIME-sniffing feature that will attempt to determine the content-type for each downloaded resource. In some cases, Internet Explorer reports a MIME type different from the type specified by the Web server. For example, if Internet Explorer finds HTML content in a file delivered with the HTTP response header `Content-Type: text/plain`, Internet Explorer determines that the content should be rendered as HTML. Because of the number of legacy servers on the Web (such as those that serve all files as `text/plain`) MIME-sniffing is an important compatibility feature.

Unfortunately, MIME-sniffing also can lead to security problems for servers hosting untrusted content. Consider, for instance, the case of a picture-sharing Web service which hosts pictures uploaded by anonymous users. An attacker could upload a specially crafted JPEG file that contained script content, and then send a link to the file to unsuspecting victims. When the victims visited the server, the malicious file would be downloaded, the script would be detected, and it would run in the context of the picture-sharing site. This script could then steal the victim's cookies, generate a phony page, and so on.

To combat this problem, Internet Explorer 8 has a number of changes to the MIME-type determination code. These include restricting “upsniff” of files served with `image/*` content types into HTML and script, the ability to opt-out of MIME-sniffing, and a new mechanism to prevent untrusted content from compromising your site's security.

For more information on changes to MIME handling in Internet Explorer 8, see [this post](http://blogs.msdn.com/ie/archive/2008/07/02/ie8-security-part-v-comprehensive-protection.aspx) [http://blogs.msdn.com/ie/archive/2008/07/02/ie8-security-part-v-comprehensive-protection.aspx] on the Internet Explorer Team Blog.

Reach Beyond the Page

In addition to the improvements made to the underlying platform and developer experience, Internet Explorer 8 introduces new features that deliver new opportunities for Web developers to extend their experience beyond the Web page. These new features are enabled declaratively through community standards rather than building custom client code that is not interoperable across browsers.

Accelerators

Whether finding directions, posting blog entries, or performing other common actions, today's Web users often find themselves copying and pasting information from one Web page to another. While the experience has been "good enough" so far, Internet Explorer 8 brings these powerful Web services one step closer through a new feature called Accelerators.

Accelerators are contextual services that provide quick access to external site services from any Web page. They typically involve one of two types of actions:

- "Look up" information related to data in the current Web page.
- "Send" content from the current Web page to another application.

For example, a user may be interested in the location of a venue displayed on the current page. In the past, the user would need to copy the address and paste it into a new browser window, unless the site's developer had gone through the effort of integrating with a mapping service. By supporting the new "map" Accelerator, the user would get an in-place view of the map using his favorite mapping service on hover from any Web page, and can navigate to the Web page by clicking on the Accelerator. (See Figure 2 for an example.)

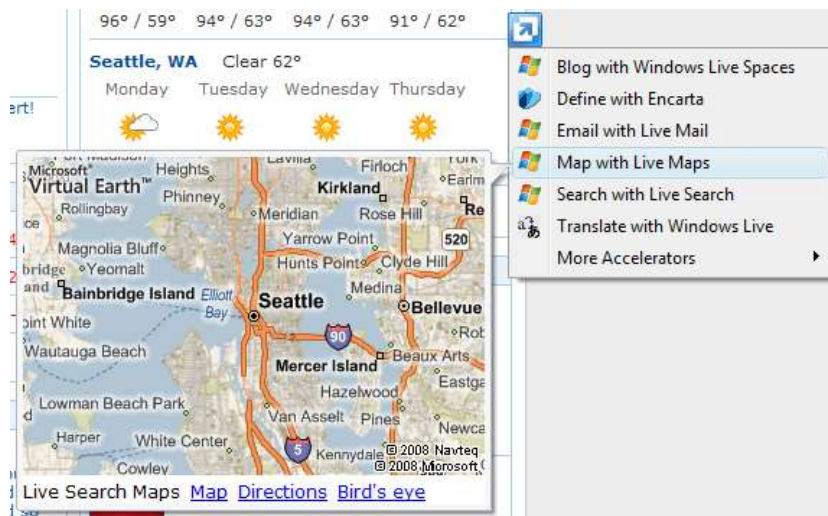


Figure 2. Image of an Accelerator menu in Internet Explorer 8.

An example of a “send” Accelerator could involve sending a segment of a news article to a blog application. After a user highlights a section she finds interesting, she can use the blog Accelerator, which navigates to her blog site with the selection already available in the edit field.

For more information on Accelerators in Internet Explorer 8, see the [OpenService Accelerators Developer Guide](http://go.microsoft.com/fwlink/?LinkID=111615) [http://go.microsoft.com/fwlink/?LinkID=111615].

Web Slices

Users commonly visit many Web sites several times a day to check for updates. The introduction of Really Simple Syndication (RSS) feeds can make this experience easier for users, although this requires a nontrivial amount of work on behalf of the developer.

Web Slices is a new feature for Web sites that enable users to subscribe to content directly within a Web page through simple HTML annotations, with prominent access to the Web page. Web Slices behave just like feeds in that users can subscribe to them and receive update notifications when the content changes. Sites are polled on an interval similar to how RSS feeds are handled. Site operators may also define the suggested interval for frequently updating content such as auctions or an e-mail inbox.

A Web Slice is a section within a Web page that is treated like a “subscribable” item, just like a feed. To enable Web Slices on your Web site, you annotate your Web page with class names for the title, description, and other “subscribable” properties. The Web Slice format uses a variation of the hAtom Microformat.

Users of Internet Explorer 8 can discover Web Slices within a Web page and add them to the Favorites bar, a dedicated row below the Address bar for easy access to links. Internet Explorer 8 subscribes to the Web page, detects changes in the Web Slice, and notifies the user when updates occur. Users can preview these updates directly from the Favorites bar and click through to the Web site to get more information.

For more information on creating Web Slices with Internet Explorer 8, see “[Subscribing to Content with Web Slices](http://go.microsoft.com/fwlink/?LinkID=125392)” [http://go.microsoft.com/fwlink/?LinkID=125392].

Enhanced Instant Search

Internet Explorer 7 has a built-in search box to the right of the Address bar. When the user enters a search term, that term is passed from the search box to the user’s preferred search engine and the user is taken to search results page from that provider.

The search box in Internet Explorer 8 looks similar, but it’s more helpful. As users type a search term, they will see real-time search suggestions from their chosen search provider, recommending common searches related to the text that is typed. Users can click on a suggestion at any time to immediately execute the search without having to type the entire word or phrase. Not only does this save time, but it increases the odds that the search results will be relevant.

Internet Explorer 8 also enables search providers to deliver direct results and “visual search” images that provide users with immediate answers. For example, if you enter a stock ticker symbol, the search provider could provide a stock quote and a corresponding chart directly in the search box drop-down list. Internet Explorer 8 provides the enabling technology, but the choice of what to show is made by the search provider.

For more information on enhanced search in Internet Explorer 8, see “[Search Provider Extensibility in Internet Explorer](http://go.microsoft.com/fwlink/?LinkID=105869)” [http://go.microsoft.com/fwlink/?LinkID=105869].

End-user and IT Pro Features

In addition to the features presented previously in this overview, Internet Explorer 8 introduces many features intended primarily for end-users or for IT pros (such as network administrators) that also have relevance for developers. Read on for details. The most current information for end-users will always be on the [Features](http://go.microsoft.com/fwlink/?LinkID=146962) [http://go.microsoft.com/fwlink/?LinkID=146962] page of the Internet Explorer 8 site, and the most current information for IT pros will always be on the [Internet Explorer TechCenter](http://go.microsoft.com/fwlink/?LinkID=117811) [http://go.microsoft.com/fwlink/?LinkID=117811].

Crash Recovery

With the advent of tabbed browsers, users typically have multiple browser tabs active within a single instance of the browser. Internet Explorer 8 has a new feature called LCIE (Loosely-coupled Internet Explorer), with which the window frame and the browser tabs run in different processes and potentially at different [Windows integrity levels](http://msdn.microsoft.com/en-us/library/bb625964.aspx) [http://msdn.microsoft.com/en-us/library/bb625964.aspx]. In Internet Explorer 8, thanks to the LCIE architecture, a crash in one browser tab does not cause the entire browser to crash, thereby helping to prevent the user from losing the pages they are browsing. In addition to saving users time, crash recovery in Internet Explorer 8 will also mean that IT professionals can spend less time helping users recover lost data. Recovering gracefully from crashes helps save time and money and can help increase user and IT staff productivity due to fewer interruptions.

In addition, Internet Explorer 8 has Automatic Crash Recovery. A tab that crashes will be restored automatically to its former state. In the unlikely event that the window frame itself crashes or closes unexpectedly, the entire session, including all tabs, will be restored. If a user was entering information on a Web page, such as writing an e-mail or filling out a form, and the page crashes, Internet Explorer 8 will attempt to recover the information entered when it returns the page, as shown in Figure 3.

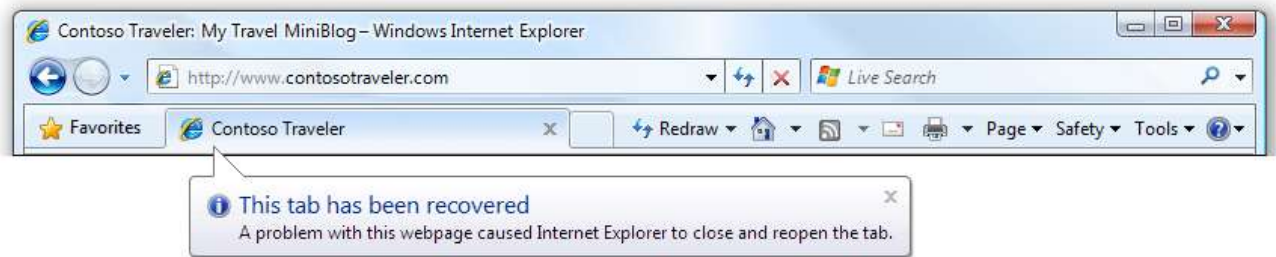


Figure 3. Image of crash recovery in Internet Explorer 8.

While perhaps less unnerving than a crash, accidental closure of a tab can be just as frustrating. Internet Explorer 8 reduces that frustration by letting users restore recently closed tabs from the New Tab page.

Performance and Memory Improvements

Internet Explorer 8 includes many performance enhancements, enabling Web developers to build richer, more interactive Web pages while still delivering a great user experience. The script engine is significantly faster, improving page load times and responsiveness for the vast majority of Web pages that are based on JavaScript or Ajax. Internet Explorer 8 also includes other revamped browser components that are faster, including the HTML parser, CSS rule processing, and markup tree manipulation.

Internet Explorer 8 includes many improvements to memory management—for example, it now mitigates memory leaks created by circular references between script objects and DOM objects, helping to deliver a more stable Web experience.

Managing Add-ons

Being able to manage add-ons running in the browser is a key component of keeping the browser and personally identifiable information (PII) protected. Consolidating all these management tasks into one area decreases developer and administrator effort and increases productivity. The ability to easily manage and remove suspicious add-ons is critical to building confidence in users who are concerned about malware. Most changes made in the **Manage Add-ons** window (shown in Figure 4) take effect immediately, although some (like disabling a toolbar or Explorer bar) might still require a restart of Internet Explorer.

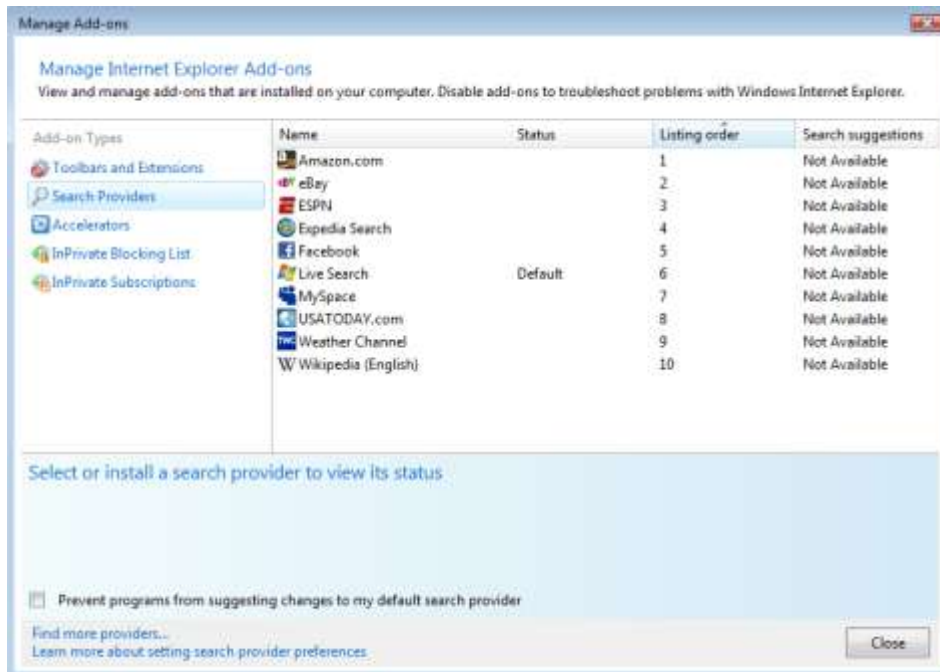


Figure 4. Image of Manage Add-ons window in Internet Explorer 8.

The **Manage Add-ons** window can be resized for different resolutions and users can choose custom columns, grouping, and sorting order. In addition, the **Manage Add-ons** window has the following capabilities:

- Select multiple add-ons from the list (CTRL+click or drag to multi-select)
- Support for context menu (right-click) actions
- Details about add-ons can be copied to the Clipboard and into e-mail, a document editor, or a spreadsheet so that information can be shared with administrators, technical support, or developers.

No Updates Required to Existing Controls

As a developer, you do not need to make changes to existing controls to continue to be managed in Internet Explorer 8. However, with the richer set of information and controls put in the hands of the user in Internet Explorer 8, control authors might wish to provide more detailed information with their controls. While the same set of information (such as publisher or version) is available in Internet Explorer 8 as was available in Internet Explorer 7, it is now easier for users to view it.

For more information, see [this post](http://blogs.msdn.com/ie/archive/2008/03/20/add-on-management-improvements-in-internet-explorer-8.aspx) [http://blogs.msdn.com/ie/archive/2008/03/20/add-on-management-improvements-in-internet-explorer-8.aspx] on the Internet Explorer Team Blog.

Easier to Find Add-on Information

More detailed information about installed add-ons is available at a glance with Internet Explorer 8. Links make it easy to accomplish the following common tasks:

- Find more add-ons with a single click. Just click **Find more add-ons...**
- Search online for information about a particular add-on by clicking **Search for this add-on via default search provider**.
- After selecting an add-on, the **More information** link appears. Clicking **More information** displays more detailed technical information about installed add-ons, including file names, versions, and other properties. You can even view or clear the list of Web sites that Microsoft ActiveX controls are allowed to run on for per-site installed ActiveX controls. For more information, see [Per-site ActiveX Controls](http://go.microsoft.com/fwlink/?LinkId=146958) [http://go.microsoft.com/fwlink/?LinkId=146958].
- Right-click any add-on for access to common actions like **Enable** or **Disable**.

New types of Add-ons to Manage

In Internet Explorer 8, the list of add-ons you can manage has been expanded to include Toolbars and Extensions, Search Providers, and Accelerators.

Toolbars and Extensions

An Explorer Bar, like a toolbar, is an extensibility type that is supported by previous versions of Internet Explorer, but was not listed in **Manage Add-ons** prior to Internet Explorer 8. With Internet Explorer 8, Explorer Bar information is more prominently displayed (see Figure 5) so users have more control over what's running in the browser.

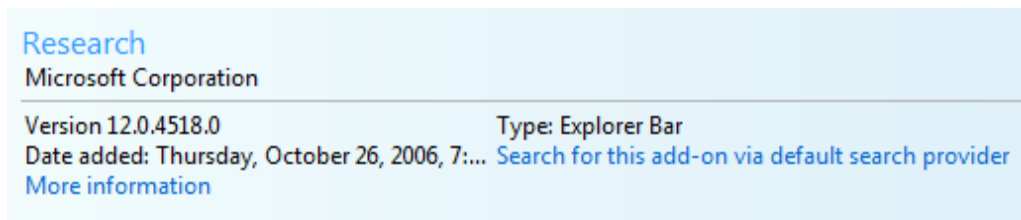


Figure 5. Image of Explorer Bar information in Internet Explorer 8.

Search Providers

Internet Explorer 7 added support for OpenSearch Search Providers, but the feature had its own separate management window. In Internet Explorer 8, management of search providers has been moved to **Manage Add-ons**. Internet Explorer 8 enables users to quickly see what search providers are installed, which provider is the default, and where each provider is sending information when a search is submitted. Additionally, users can change the order that search providers are listed (Internet Explorer 7 always sorted them alphabetically).

A new feature prevents programs from “suggesting changes” to the default search provider. This feature helps prevent users from losing their default search provider if another program attempts to change the search provider without user consent.

Internet Explorer 8, like Internet Explorer 7, continues to support the OpenSearch standard for search providers. For more information, see “[Search Provider Extensibility in Internet Explorer](http://go.microsoft.com/fwlink/?LinkId=105869)” [http://go.microsoft.com/fwlink/?LinkId=105869]

Domain Name Highlighting

Domain Name Highlighting (see Figure 6) aids users to more accurately and quickly ascertain that the site they are visiting is the intended site. This is particularly useful as many deceptive sites embed a domain name within the URL as an attempt to deceive users into believing they are visiting legitimate Web sites.

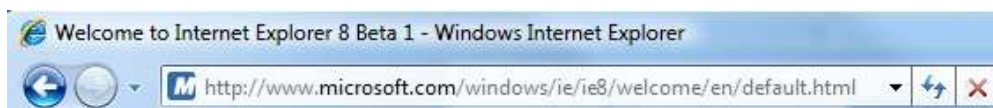


Figure 6. Image of domain name highlighting in Internet Explorer 8.

Figure 7 shows domain highlighting for a secured site with an Extended Validation (EV) SSL Certificate. Note that HTTPS and the domain name are highlighted. Domain highlighting builds on EV SSL present in Internet Explorer 7.

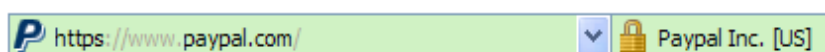


Figure 7. Image of domain highlighting for a secured site with an EV SSL Certificate in Internet Explorer 8.

Internet Explorer 8 builds on the Phishing Filter technology introduced in Internet Explorer 7—which was designed to warn users when they attempt to visit known phishing sites—and replaces it with the SmartScreen® Filter. The SmartScreen Filter helps protect against phishing Web sites, other deceptive sites, and sites known to distribute malware. Figure 8 is an example of the Address bar when the browser is at an unsafe Web site as determined by the SmartScreen Filter:

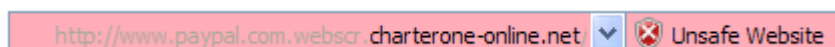


Figure 8. Image of the Address bar when Internet Explorer 8 is at an unsafe Web site as determined by the SmartScreen Filter.

Domain spoofing attacks are simple and fairly unsophisticated, but common. Domain name spoofing isn't always about banking scams; a sophisticated spoofing attack may spoof a B2B partner portal. If, for example, an attacker successfully spoofed a document sharing site, workers may inadvertently share confidential or sensitive documents exposing this information. Domain name highlighting helps block attacks to increase security and productivity.

Enhanced Add-on Management

Add-ons—specifically custom toolbars that users install—are the most frequent cause of browser crashes. Crashes due to add-ons can result in lost productivity and often require IT staff time to resolve

the issue. Internet Explorer 8 provides easier management of add-ons and enables users to manage their add-ons more effectively.

Being able to manage add-ons running in the browser is a key component of keeping the browser and personally identifiable information protected. The ability to easily manage and remove slow-performing or suspicious add-ons is critical to building confidence in users who are concerned about add-ons. Internet Explorer 8 makes it easier for users to disable an unwanted toolbar.

Data Execution Prevention

Internet Explorer 7 on Windows Vista introduced an off-by-default Internet Control Panel option, **Enable memory protection to help mitigate online attacks**.

This option is also referred to as Data Execution Prevention or No-Execute (DEP/NX).

DEP/NX helps to foil attacks by preventing code from running in memory that is marked non-executable, such as a virus disguised as a picture or video. DEP/NX, combined with other technologies like Address Space Layout Randomization (ASLR), make it harder for attackers to exploit certain types of memory-related vulnerabilities like buffer overruns. Best of all, the protection applies to both Internet Explorer *and* the add-ons it loads. No additional user interaction is required to provide this protection, and no new prompts are introduced.

The DEP/NX feature is enabled by default for Internet Explorer 8 on Windows XP Service Pack 3 (SP3), Windows Vista Service Pack 1 (SP1), Windows Server 2008, and later to help maintain security.

For more information on DEP/NX, see [this post](http://blogs.msdn.com/ie/archive/2008/04/08/ie8-security-part-I_3A00_-dep-nx-memory-protection.aspx) [http://blogs.msdn.com/ie/archive/2008/04/08/ie8-security-part-I_3A00_-dep-nx-memory-protection.aspx] on the Internet Explorer Team Blog.

Conclusion

Internet Explorer 8 includes a host of new features, enhancements, and other improvements driven by real-world scenarios and customer requests. Internet Explorer 8 is a platform for Web innovation, offering business value, developer efficiency, and better user safety.

Be sure to visit the following additional resources to learn more about Internet Explorer 8:

- Internet Explorer 8 Homepage: <http://www.microsoft.com/ie/ie8>
- Internet Explorer Team Blog: <http://blogs.msdn.com/ie/>
- Internet Explorer Developer Center: <http://msdn.microsoft.com/ie/>