Major Advances in Internet Explorer 8

- Developer Tools
- Great AJAX Support
- Improved Reliability and Privacy
“Your product line is outstanding and has taken our product lines to a new level. It is very rare to get exactly what you think you’re buying, never mind your tradition of delivering more and better than expected.”
Mark Hale - Simplicity Computer Solutions Inc.

“The DevExpress product line is top-notch and second-to-none. I say this because I have used other company’s products, but I must say I have not run across any set of components that are as mature, customizable, cross-platform compatible, fully thought out, and simply amazing as the DevExpress product line.”
Troy Zarger - Zarger Marketing Group

“I have found the Developer Express components to be the best there is. Let me explain why. Their components are developed with precision and expertise. Their documentation is exceptional and thorough. I find their appearance to be the best there is. With little effort a developer is able to create a professional looking application that far exceeds other after-market component suites.”
Victor Boba - BOCA Software Technologies, Inc.
"We used to bitterly complain about how buggy and featureless third-party controls were. After purchasing DXperience, we still complain, but this time it's about having to maintain the projects not using Developer Express' components."

P. Kline

"I have used just about every .NET component package and no one even comes close to Developer Express."

Dan Haligas - Gridistics, LLC

Download Your Free Evaluation Copy Today
www.devexpress.com/eval
Features

6 Welcome to CoDe Focus for Internet Explorer 8 Beta 2!
Chris Wilson

8 What’s New in Internet Explorer 8 Beta 2?
Doug describes the great feast of features and improvements in Internet Explorer 8 Beta 2.
Doug Stamper

14 Making Your Website Compatible Across Multiple Versions of Internet Explorer
Scott explains how to write content according to the latest Web standards and have it “just work” in many browsers.
Scott Dickens

18 Introducing Compatibility View
Internet Explorer 8 Beta 2 features a new Compatibility View, which is an end-user site compatibility mitigation option for gaining access to content built for older Web browsers.
Scott Dickens

21 Create Your Own Web Slices
A great new browsing feature in Internet Explorer 8 Beta 2, Web Slices let you subscribe to content directly within a Web page without requiring a separate feed file.
Seth McLaughlin

26 Accelerators in Internet Explorer 8 Beta 2
Ritika walks through how to create a codeless accelerator, which allows users to interact with Web services or applications and do more on your site.
Ritika Virmani

30 New Search Features in Internet Explorer 8 Beta 2
Sharon describes new features in Internet Explorer 8 Beta 2 that improve the search experience.
Sharon Cohen

36 Developer Tools in Internet Explorer 8 Beta 2
The Internet Explorer 8 Developer Tools let you edit your site within the browser and see changes take effect immediately, profile your site to help improve your JavaScript, enable better debugging, and more.
John Hrvatin, Deepak Jain, Sameer Chabungbam

43 Better AJAX Development with Internet Explorer 8 Beta 2
Sunava and Matt describe AJAX enhancements in Internet Explorer 8 Beta 2 including DOM Storage objects, connectivity enhancements, more connections per server, updates to the XMLHttpRequest object, the new XDomainRequest object, and features to enable cross-document messaging.
Sunava Dutta, Matthew Tevenan

53 Measuring AJAX Performance in Internet Explorer 8 Beta 2
In this article, Shreesh describes improvements in the AJAX subsystems of Internet Explorer 8 Beta 2 to measure and improve browser performance.
Shreesh Dubey

56 Performance Improvements in Internet Explorer 8 Beta 2
Internet Explorer 8 Beta 2 offers a better, faster JScript engine, improved download speed, faster navigation, native JSON support, and much more.
Christian Stockwell, Travis Leithead, Gaurav Seth

62 Reliability and Privacy with Internet Explorer 8 Beta 2
Andy discusses reliability improvements in Internet Explorer 8 Beta 2 and how they will help protect a user’s personally identifiable information.
Andy Zeigler

66 Secure Coding with Internet Explorer 8 Beta 2
Eric describes new Internet Explorer security features to help protect Web users and applications.
Eric Lawrence

Departments

7 Code Compilers

25 Advertiser Index

US subscriptions are US $29.99 for one year. Subscriptions outside the US pay US $44.99. Payments should be made in US dollars drawn on a US bank. American Express, MasterCard, Visa, and Discover credit cards are accepted. Bill me option is available only for US subscriptions. Back issues are available. For subscription information, email subscriptions@code-magazine.com or contact customer service at 832-717-4445 ext 10.

Subscribe online at www.code-magazine.com

CoDe Component Developer Magazine (ISSN # 1547-5166) is published bimonthly by EPS Software Corporation, 6605 Cypresswood Drive., Suite 300, Spring, TX 77379. POSTMASTER: Send address changes to CoDe Component Developer Magazine, 6605 Cypresswood Drive., Suite 300, Spring, TX 77379.
balance your work.

CounterSoft Gemini provides an intuitive way to manage software development projects: bugs, tasks, issues, requirements, and support tickets. A crisp browser-based product with integration into Microsoft Outlook and Visual Studio, so you can manage your bugs, tasks and issues as you please. Use our **FREE 5 User License** to get your team started.

call +44.20.7117.7804 or visit www.countersoft.com to see why people love our products.
Welcome to CoDe Focus for Internet Explorer 8!

In this issue, you'll find information on all things related to Internet Explorer 8 Beta 2.
From Web Slices to Developer Tools, from our improvements in performance to the great new features for building Web 2.0 applications, there's something here for everyone.

Since I joined the Internet Explorer team back in 1995, I've helped ship a lot of Internet Explorer releases, and Internet Explorer 8 is shaping up to be one of the most exciting, with a full set of new capabilities and improvements for users and developers.

We began Internet Explorer 8 by developing a strategy for protecting compatibility with the content of today, while simultaneously unleashing our ability to deliver great standards compliance. We then set the foundation for delivering standards compliance by building a brand-new layout engine, designed from the ground up for CSS 2.1 compliance. With that foundation in place, we started addressing other areas: not only have we improved security throughout, as well as building new capabilities to help Web developers build secure mashups (with features like cross-document messaging and cross-domain network requests), but we've also focused on performance throughout the Internet Explorer stack, particularly for today's (as well as tomorrow's) JavaScript-heavy Web applications.

Finally, we wanted to make the Web more efficient and approachable for users, and encourage users to reach beyond just a single Web page with Accelerators, Web Slices and new Search features. Not only does this let users navigate the resources of the Web more effectively, but it also provides new opportunities for Web services to integrate into the browsing experience, rather than just being a destination.

With this issue of CoDe Focus, several of my colleagues have written some great articles on these areas of Internet Explorer to help you start developing for Internet Explorer 8. Happy browsing!

-Chris Wilson
Internet Explorer Platform Architect

Visit DevMavens.com to learn what the most trusted community leaders are saying every day.

DevMavens.com
Feed your brain.

Are the other magazines leaving you with an empty feeling? Do you expect steak but end up with a mouthful of sawdust?

Satisfy your cravings by sinking your teeth into the next issue of CoDe Magazine. The meaty articles and juicy commentary will feed your brain with the best in-depth .NET coverage in town.

Each issue of CoDe Magazine is packed with practical information about the new Microsoft technologies, written from a developer’s perspective and giving you plenty to chew on.

Stock up!
Order a year’s supply now at www.code-magazine.com
What’s New in Internet Explorer 8 Beta 2?

For those of you who have been waiting to see what Microsoft has been cooking up for the standards-based Web, dinner is served! Released to the Web in August 2008, Internet Explorer 8 Beta 2 delivers much more than just a snack for today’s ravenous tech-savvy users. Internet Explorer 8 Beta 2 dispenses a four-course meal guaranteed to please even the most discriminating of palates.

All of this talk about food makes me hungry, so let’s dig in. Internet Explorer 8 Beta 2 is the latest installment of Microsoft’s Web browser and includes new features for end users and developers alike. From in-box Developer Tools, to Web Slices and Accelerators, to improved performance and a platform built for today’s Web 2.0 applications, there is plenty to feast on in Internet Explorer 8 Beta 2.

Internet Explorer 8 Beta 2 represents the third release of Internet Explorer that I have personally had the privilege to participate in and it outshines any browser we have shipped to date.

The Microsoft Internet Explorer team’s goals for the final Internet Explorer 8 Beta 2 release remain: to fully support CSS 2.1, to improve script performance, to enable new Web scenarios by including early HTML 5 support, to ship built-in developer tools, to integrate Web services into the user’s workflow through Web Slices and Accelerators, and to strengthen Internet Explorer’s foundation in the security and privacy realms.

This article briefly introduces the features included in Internet Explorer 8 Beta 2. For more in-depth discussion, see the other articles written by the Internet Explorer team in this issue of CoDe Focus.

Heaping Helpings for the Developer

We have unleashed the power of the standards-based Web through Cascading Style Sheets (CSS) 2.1 compliance, while simultaneously protecting compatibility with the content of today. Making your website compatible across multiple versions of Internet Explorer has been dramatically simplified with version targeting. Additionally, we have laid a great foundation for the development of secure mashups through cross-document messaging (XDM) and cross-domain requests (XDRs); which, when coupled with better script performance, dramatically improves the JavaScript-heavy Web applications experience.

Top this meal off with our integrated Developer Tools. Debugging and profiling your content or script has never been so easy.

Predictability: Interoperable Version Targeting

Improved standards support in Internet Explorer 8 Beta 2 makes developing sites that work across different browsers simpler and less time consuming. Internet Explorer 8 Beta 2 includes upgrades to the Web platform, compatibility, and functionality designed to keep you on point creating high-quality interactive experiences, not debugging browser interoperability issues.

Internet Explorer 8 Beta 2 interprets Web content in the most standards-compliant way it can. This simple statement turns out to have complex ramifications. Thankfully, Internet Explorer 8 Beta 2 delivers a model that both protects current content from client changes and enables adoption of new exciting features.

Compatibility Mode

In addition to the “Standards” and “Quirks” modes widely understood from past versions of Internet Explorer, Internet Explorer 8 Beta 2 ships with a third layout mode—Internet Explorer 7 Standards mode. With this layout mode, Internet Explorer 8 Beta 2 renders pages in the same way that Internet Explorer 7 renders its “Standards mode” pages. It’s also the quickest option for getting existing content to display correctly in Internet Explorer 8 Beta 2.

You tell Internet Explorer 8 Beta 2 to display a site or page in Internet Explorer 7 Standards mode by
simply adding a `meta` tag. No additional changes are required.

```
<meta http-equiv="X-UA-Compatible" content="IE=7"/>
```

Upon page load, Internet Explorer 8 Beta 2 reads the tag and switches the browser to interpret all content on the page as Internet Explorer 7 Standards instead of Internet Explorer 8 Standards. For more details, see “Making Your Website Compatible Across Multiple Versions of Internet Explorer” in this issue.

Power/Performance: AJAX Improvements

Internet Explorer 8 Beta 2 provides a programming model for Asynchronous JavaScript and XML (AJAX) development that is simpler yet more powerful than ever before, spanning browser, Web page, and server interaction. You'll be able to build pages that are faster and more functional, with better end-user experiences. Plus, the APIs we’ve added to Internet Explorer 8 Beta 2 are based on the W3C HTML 5.0 or Web Applications Working Group standards.

AJAX Navigations

One of the great benefits of implementing AJAX—and really, one of the main reasons it exists—is the ability to update page content without navigating to a new page. With this convenience, though, come drawbacks that can confuse users. On an AJAX-heavy page, the Address bar is not updated with each update. Subsequently, the browsing history isn’t updated either.

To enable AJAX navigations, Internet Explorer 8 Beta 2 in Internet Explorer 8 Standards mode treats updates to the `window.location.hash` property like traditional navigations. When the `hash` property is updated, the previous document URL (which may be from the previous hash fragment) is updated in the Address bar and the travelog. At the same time, a new `hashChanged` event is raised, and the hash URL fragment is saved before navigating away from the page.

On AJAX-enabled pages that take advantage of this new functionality, when AJAX content changes, navigation is as seamless as usual, but the user can back up and go forward as if the AJAX navigation was a traditional navigation.

DOM Storage

To store data on local machines, websites today often use the `document.cookie` property. However, cookies are limited in their capabilities; sites can only store fifty key/value pairs per domain, and the cookie programming model requires parsing the entire cookie string for data. DOM Storage objects—specified in the W3C’s HTML 5 Working Draft and implemented in Internet Explorer 8 Beta 2—provide a much simpler and more flexible global and session storage model for structured data on the client side.

DOM Storage offers essential differences from cookies. For one, it offers significantly more available disk space than cookies. In Internet Explorer, cookies can store 4 kilobytes (KB) of data, whereas DOM Storage provides about 10 megabytes (MB) for each storage area. Furthermore, DOM Storage doesn’t transmit values to the server with every request as cookies do, and data in a global store never expires. Unlike cookies, it's easy to access individual pieces of data using an interface that is supported in Internet Explorer 8 Beta 2 and other browsers, and sites can choose to store data for the life of a tab or until the site or the user clears the data.

More AJAX Enhancements

Internet Explorer 8 Beta 2 includes enhancements to connection events and connection scaling, as well as updates to the `XMLHttpRequest` (XHR) object. For more details, see “Better AJAX Development with Internet Explorer 8 Beta 2,” in this issue.

Productivity: Integrated Developer Tools

Great tools play a critical role in developer productivity, and while many Web development tools exist, they probably don’t meet your needs in all tasks.

To improve developer productivity in debugging JavaScript, CSS, and HTML, Internet Explorer 8 Beta 2 includes powerful, yet easy-to-use tools that have a few important characteristics:

Integrated and Simple to Use

Every installation of Internet Explorer 8 Beta 2 includes the Developer Tools. This enables debugging anywhere you find Internet Explorer 8 Beta 2, whether it’s your dev machine, test machine, or a client’s machine. In addition, by avoiding the use of an extension, we have limited the impact of the tools on browser performance to only the times when you open the tools. Internet Explorer 8 Beta 2 also includes on-the-fly script debugging, so you can enable debugging as needed for only the current Internet Explorer 8 Beta 2 process, rather than enabling debugging for all of Internet Explorer 8 Beta 2 and accepting degraded performance and regular script error dialog boxes.

Provides a Visual Interface to the Platform

Instead of reverse engineering how your site works or modifying your site to output debug information, the Developer Tools enable you to look into Internet Explorer 8 Beta 2 to view its representation of your site. This reduces the time you spend debugging dynamic sites where source inspection isn’t useful or investigating a behavior specific to Internet Explorer 8 Beta 2 where a generic authoring tool cannot help.

Enables Fast Experimentation

When prototyping a new design or testing fixes in previous versions of Internet Explorer, you likely
edit your source, save it, refresh your page in the browser, and repeat. The Internet Explorer 8 Developer Tools streamline this scenario by letting you edit your site within the browser and see changes take effect immediately.

**Helps Optimize Application Performance**

Identifying and fixing performance issues is usually an iterative approach done by focusing on one scenario at a time. With the Internet Explorer 8 Developer Tools script profiler, you can collect statistics like execution time and number of times a JavaScript function was called as you test your application and use the profile report to quickly identify and fix performance bottlenecks.

With these characteristics, the features of the Developer Tools will dramatically improve your productivity when developing in Internet Explorer 8 Beta 2. For more detailed information, see the article “Developer Tools in Internet Explorer 8 Beta 2” or visit the IE Developer Center on MSDN at [http://msdn.microsoft.com/ie](http://msdn.microsoft.com/ie).

Well, we’ve got the developer food groups covered, so now let’s move on to a balanced meal for the end users, shall we?

**Perfect Portions for the End User**

We cooked up even more goodness for end users to make the Web more efficient and approachable. Reach well beyond just a single Web page and expose the dynamic power of Web services. Web Slices and Accelerators take advantage of Web services in a convenient, easy-to-use experience.

Our improved search capabilities enable you to land that bigger data “fish” by casting a wider net that examines your Autocomplete (previously typed search terms), Search Suggestions (from your currently selected search provider), and your History, as well as the Internet.

Safety and reliability are paramount in today’s connected world. Safety and reliability join security in a protective journey for the modern browser. Internet Explorer 8 Beta 2 continues to push the innovative envelope with features like Loosely-Coupled Internet Explorer (LCIE), Automatic Tab Recovery and InPrivate™ Browsing.

And for dessert, we offer our Compatibility View feature. With millions of pages and sites in our globally connected world, not all pages will be ready simultaneously for strict standards-based content. While Internet Explorer 8 Beta 2 respects the intent of the Web author when known, Compatibility View also allows the user to fall back to previous rendering behavior should a site appear to be adversely affected.

Meal complete! Time to examine the recipes.

---

**Beyond a Single Web Page: Web Slices and Accelerators**

Probably the most anticipated end-user features of Internet Explorer 8 Beta 2 are Web Slices and Accelerators. Read on to whet your appetite, and then head to the main articles, later in this issue!

**Web Slices**

Web Slices are a new Internet Explorer 8 Beta 2 feature that allows you to subscribe to parts of a Web page and view updates directly from the Internet Explorer Favorites bar. Web Slices let users bring their favorite parts of the Web with them wherever they browse.

Web Slices allow the user to subscribe to a portion of a page and monitor the updates on the Favorites bar throughout their browsing experience. Any portion of a page that updates regularly is a reasonable candidate to become a Web Slice. News, stock quotes, sports scores, exchange rates, or even e-mail can be presented prominently in Internet Explorer as part of the Favorites bar. Updates to Web Slices are indicated visually in the item through bolded text.

There are two ways for users to discover a Web Slice: in document by hovering over a Web Slice region on a page, and through the feed discovery button on the command bar.

Once a Web Slice has been subscribed to, it can be accessed through the Favorites bar. For example, consider an eBay Web Slice that allows users to track an auction. Rather than clicking refresh on an auction page every few minutes, a user can subscribe to an auction and be visually alerted of changes to the auction automatically.

See “Creating Your Own Web Slices” in this CoDe Focus issue for a developer’s reference to Web Slices.

**Accelerators**

Accelerators, formerly known as Activities, are a new concept that allows users to interact with a Web service or application from the browsing experience. Users typically copy and paste text from one Web page to another for various reasons. Internet Explorer 8 Beta 2 Accelerators make this much easier to do.

The browser is becoming more than just a place to read content on the Web. Increasingly, users are interacting with Web content in several ways. Examples include pasting selected text in a blog post, mapping out an address, or checking the definition of a word. With Accelerators, users can reach beyond the page to various services for these common tasks.

Accelerators are easy for Web developers to create and can be written to plug into existing applications. They are great ways to attract users to your website through a single click!
What’s New in Internet Explorer 8 Beta 2?

Accelerators work on selected text, hyperlinks, and the current document. The user sees different options in the Accelerators menu depending upon the type of content. The most common content is selection; when a user makes a selection the Accelerator button appears. Clicking the button displays the set of Accelerators the user has installed that work for selected text.

Alternatively, users can also bring up the Accelerators menu through the context menu of the selected text or hyperlink. They can access the current page’s Accelerators from the context menu of the page or the Page button on the Internet Explorer Command bar.

For more details, see “Accelerators in Internet Explorer 8 Beta 2” in this issue.

Find the Data You Need: Improved Search

With Internet Explorer 8 Beta 2, we have taken some big steps that we hope will change the way you search. Our goal is to enable you to search for the right term on the right provider and get the right information as quickly as possible.

Search Suggestions

Search suggestions appear as you are typing and offer you ideas on things you might be searching for. In addition to highlighting interesting suggestions, visual suggestions can also be used to provide more useful information.

There are two additional sections you’ll notice under the search box. The first section, directly under the search box, contains your previously typed search terms. The other section you’ll notice is comprised of results from your History. This section is new to Internet Explorer 8 Beta 2. Using Windows Search, Internet Explorer 8 Beta 2 will search over your stored history.

In all three sections under the search box, we have added hit highlighting so that you can glance through the results and know exactly why each one was suggested to you. The search box dropdown is also consistent in look and feel to the Address bar. In normal use, the Address bar will display results from your Favorites, History, and feeds; however, you can also switch the address bar over to search mode by typing “?” a space, and then your search term. This will allow you to get search suggestions in the Address bar.

For all the details, see “New Search Features in Internet Explorer 8 Beta 2.”

InPrivate Browsing

If you’re using a shared PC, borrowing a laptop from a friend, or using a public PC, sometimes you don’t want other people to know where you’ve been on the Web. Internet Explorer 8 Beta 2 InPrivate Browsing lets you browse the Web without storing history, cookies, temporary Internet files, and other data.

Using InPrivate Browsing is as easy as launching a new InPrivate Browsing window. InPrivate Browsing can be found under Safety in the Command Bar, or in the Tools menu. When you are finished browsing, just close the window and Internet Explorer 8 Beta 2 will take care of the rest.

With InPrivate Browsing enabled, you browse just as you normally do, but with the following differences:

- New cookies are not stored
  - All new cookies become “session” cookies
  - Existing cookies can still be read
  - The new DOM storage feature behaves the same way
- New history entries will not be recorded
- New temporary Internet files will be deleted after the InPrivate Browsing window is closed
- Form data is not stored
- Passwords are not stored
- Addresses typed into the Address bar are not stored
- Queries entered into the search box are not stored
- Visited links are not stored

InPrivate Browsing makes shopping for jewelry for your spouse on a shared computer, for instance, as easy as pie! See the IE Team Blog for more details on InPrivate Browsing: http://blogs.msdn.com/ie/.

Automatic Tab Recovery

In the event of a crash, Automatic Crash Recovery is designed to get you back to browsing as quickly as possible. It uses Internet Explorer 8 Beta 2 tab isolation (Loosely-Coupled Internet Explorer, or LCIE) to help localize the failure to your tab. Should you encounter a crash, you will see a “balloon” telling you that a tab has been recovered. This is the “tab recovery experience”—the failure has been confined to just one tab. Your browser never goes away and you get back to your site quickly.

Behind the scenes, Internet Explorer Beta 2 keeps track of information about your tab. The following data about each tab is stored:

- Current URI
- The travelog (your back/forward history)
- Tab order
- Which tab was active
- Session cookies
- Form data
When you crash, Internet Explorer 8 Beta 2 will tear down the old tab process, create a new tab process, and recover the stored data back into the tab. For many websites this works well; however, there are other websites, such as sites with Web forms, or sites that you need to log in to, that cannot be successfully recovered.

Leveraging LCIE allows Automatic Crash Recovery to quickly restore the user to their browsing session without having to log back in to their sites or re-enter data into forms. Combined, LCIE and Automatic Crash Recovery provide an innovative and graceful way to recover from crashes.

See the IE Team Blog for more details on this feature.

Compatibility View

A lot of the existing pages on the Web—the majority, actually—were developed with either no concept of modern Web standards (developers just didn’t know or care) or were authored pre-standards (yes, there are pages that old out there). These older, non-standards-based pages work just great in Internet Explorer 8 Beta 2, since they depend on behavior that has been in Internet Explorer for quite some time and hasn’t changed.

As you might expect, there are also a large number of pages authored according to modern Web standards, and the trend is that standards-based pages will soon become the majority. Unfortunately, there are differences between modern browsers and older browsers in the level and type of standards support they provide. New standards have emerged; existing standards have been clarified; bugs have been fixed.

Most sites written to modern standards work in Internet Explorer 7 today. Some, but not all, of these will work in Internet Explorer 8 Beta 2 by default. But there is a small class that won’t. This is where the Compatibility View feature comes in. Compatibility View enables content designed for older Web browsers to still work well in Internet Explorer 8 Beta 2.

The feature is controlled by UI settings and Group Policy on the client. When enabled, Compatibility View performs the following changes:

- The Internet Explorer User Agent String denotes a version of ‘7’
- The Internet Explorer Version Vector, which is used in the evaluation of conditional comments, denotes a version of ‘7’
- All pages with !DOCTYPE directives that trigger Standards Mode map to Internet Explorer 7 Standards Mode

By default, Compatibility View is off for all sites in the Internet zone. It can be enabled on a per-domain basis. Also by default, Compatibility View is enabled for all intranet websites, thus ensuring greater compatibility between Internet Explorer 8 Beta 2 and existing line-of-business applications.

The feature is controlled by UI settings and Group Policy on the client. When enabled, Compatibility View performs the following changes:

- The Internet Explorer User Agent String denotes a version of ‘7’
- The Internet Explorer Version Vector, which is used in the evaluation of conditional comments, denotes a version of ‘7’
- All pages with !DOCTYPE directives that trigger Standards Mode map to Internet Explorer 7 Standards Mode

By default, Compatibility View is off for all sites in the Internet zone. It can be enabled on a per-domain basis. Also by default, Compatibility View is enabled for all intranet websites, thus ensuring greater compatibility between Internet Explorer 8 Beta 2 and existing line-of-business applications.

Internet Explorer 8: Feed the Need

Now that you have seen a brief overview of what to expect from Internet Explorer 8 Beta 2, it’s time to download a copy and get started. Visit http://www.microsoft.com/ie8 and feed your information hunger.

It has been a pleasure talking about food... er, I mean, features with you. Try this: close your eyes and imagine your favorite meal—all of the smells and the flavors. Now crank open your laptop, fire up Internet Explorer 8 Beta 2, and search for your favorite restaurants to get that reservation before somebody else gets your table!

Doug Stamper
Introducing
The right choice for developers!

Tim is a web developer, so he cares about having the best platform and tools for bringing his websites to life. An interoperable platform, with high levels of security and privacy, that helps him make the best possible websites for his customers.

That’s why Windows® is pleased to introduce the Windows Internet Explorer® 8 Beta. With substantial advancements like web standard compliance and a compatibility option to easily update existing sites, Tim spends more time innovating.

With full support for CSS 2.1, Tim can write his pages once to display properly across different browsers and take advantage of new navigation and cross-document/domain messaging implementation. He can use new storage features for AJAX applications and scripting performance enhancements so his applications run even faster.

A new suite of built-in developer tools enable Tim to quickly debug, test and performance tune HTML, CSS and JavaScript in a visual environment right from the browser, and he can drive traffic and customer engagement on his Web site with new, innovative features like Accelerators and Web Slices.

Try it for yourself today!
www.microsoft.com/ie8/developer
Making Your Website Compatible Across Multiple Versions of Internet Explorer

**Improved standards support in Internet Explorer 8 Beta 2 makes developing sites that work across browsers simpler and less time consuming.** The latest release of the world’s most popular Web browser includes upgrades to the Web platform, compatibility, and functionality designed to keep you on-point creating high-quality interactive experiences, not debugging browser interop issues. Let’s discuss how to best take advantage of these new features while ensuring your Web pages continue to work correctly in older browsers.

Internet Explorer 8 Beta 2 interprets Web content in the most standards-compliant way it can. This simple statement turns out to have complex overtones. With compatibility expectations growing, new features need to be balanced against costs associated with updating legacy content. Thankfully, Internet Explorer delivers a model that both protects current content from client changes and enables adoption of new exciting features.

Have you always wanted to write content according to the latest Web standards and have it “just work” in multiple browsers? With the improved standards support found in Internet Explorer 8 Beta 2, you can. The latest release now provides improved support for CSS 2.1, HTML 4.01, HTML 5, CSS 3, DOM L1 and L2 Core, and Aria.

Do you have existing content that “just works” in Internet Explorer 7 that you want to continue to display correctly in Internet Explorer 8 Beta 2? Don’t worry, that’s covered too.

### Choosing the Best Layout Mode

As a point of background knowledge, modern browsers ship with at least two layout modes, **Quirks** and **Standards** (most ship with more, but I’ll get into that later). Quirks mode denotes compatibility with older Web pages, in particular those written before modern standards. For instance, the box model, a general term describing sizing and positioning of elements on a page, is quite different in these older pages than those written according to the latest Web standards. Standards mode denotes compatibility with the latest and greatest Web standards.

Standards mode is, as you might expect, a moving target as the latest Web standards keep evolving and changing. For instance, Internet Explorer 8 Beta 2’s definition of Standards mode is different than Internet Explorer 7’s definition of Standards mode. Firefox 3’s definition of Standards mode is different than Firefox 2’s definition of Standards mode. And, Internet Explorer 8 Beta 2’s definition of Standards mode is different than Firefox 3’s definition of Standards mode. Differing interpretation of Standards is the cause of most cross-browser issues encountered in Web design.

So you might ask, why would I ever choose Standards mode if there is compatibility risk? For one, Standards bring new functionality. Internet Explorer 8 Beta 2 includes full support for CSS 2.1, which means a whole set of previously unavailable features are at your fingertips. Take, for example, the common case of dividing a page up into chapters and sections. By using the ‘counter’ property you can save yourself a great deal of effort in automatic numbering situations that require a bit more complexity than the standards ordered list. **Listing 1**

---

**Fast Facts**

| People spend more time surfing the Web than any other activity on the PC and Internet Explorer is the most popular gateway to that experience. The latest release of the browser includes a new Standards mode implementation that serves as a platform for rich new experiences. Included compatibility features allow you to take full advantage of new functionality while ensuring that your existing site works well in older versions of the browser. |

---

Scott Dickens
sdickens@microsoft.com
425-705-8865

Scott Dickens is a Lead Program Manager at Microsoft, based in Redmond, WA.

Scott works on several features of Internet Explorer including layout, versioning and modes, rendering, graphics and images, text, printing, zoom, and website compatibility.
has been coded not to W3C Standards explicitly but to Internet Explorer 7’s interpretation of Web standards. To allow such sites to continue to work in Internet Explorer 8 Beta 2 you have two options:

- Update content to support Internet Explorer 8 Beta 2’s interpretation of Web standards and special cases older versions of Internet Explorer as required (preferred).
- Tell Internet Explorer to interpret Standards mode content as content authored for Internet Explorer 7’s interpretation of Web standards.

Developing Standards Mode Pages

So you want to develop Web pages according to the latest Web standards? Me too! Here's what you need to do:

1. Author your page according to the latest Web standards as supported by Internet Explorer 8 Beta 2.
2. Include a Standards mode DOCTYPE.
3. Use conditional comments to provide fix-ups for legacy versions of Internet Explorer.

Step #3 is necessary only if your markup takes advantage of new functionality that didn’t exist in previous versions of the browser or in instances where behavior has been corrected or improved in the latest Internet Explorer version.

For those unfamiliar with the power of conditional comments, basically think of them as “if” statements within your HTML. The code contained within a conditional comment block is only exercised if the expression evaluates to TRUE. The expression generally takes the form of a declared Internet Explorer version value and a comparison is performed against Internet Explorer’s Version Vector registry key.

In the following example, note how using conditional comments allows you to direct different style sheets to Internet Explorer 8 Beta 2 and Internet Explorer 7 respectively:

DOCTYPEs and <META> Tags

Internet Explorer determines layout mode by examining the DOCTYPE tag declared in a Web page’s HTML. This action occurs during page load—the Web browser “sniffs” the DOCTYPE and then switches into the appropriate layout mode, Quirks or Standards, in order to display the page correctly (read: as the author likely intended). Visit MSDN to see a list of DOCTYPEs and what layout mode they trigger (http://msdn.microsoft.com/en-us/library/ms535242.aspx).

Recognize that a ton of existing content (on the Internet / various intranets / line of business apps / etc…) that uses a Standards mode DOCTYPE shows an example and Figure 1 shows expected output. Note that without counters you would have to number the chapters and sections in this example manually, and what a pain it would be to insert a new chapter between the first and second chapters!

For another, Standards provide consistency across browsers. Internet Explorer 8 Beta 2 also includes improved support for items defined in other Web standards such as HTML 4.01, HTML 5, CSS 3, DOM L1 and L2 Core, and ARIA.

All told, these improvements make up the bulk of new features found in the Internet Explorer 8 Standards mode.

**Listing 1: Using CSS 2.1 counters**

```html
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Strict//EN">
<html>
<head>
<style type="text/css">
body {counter-reset: chapter;}
  h1:before {content: "Chapter " counter(chapter) ";";}
  h1 {counter-reset: section -1; counter-increment: chapter;}
  h2:before {content: "Section " counter(chapter) "." 
            counter(section) " ";}
  h2 {counter-increment: section; }
</style>
</head>
<body>
<h1>First Chapter</h1>
<h1>Second Chapter</h1>
<h2>First Section of Second Chapter</h2>
<h2>Second Section of Second Chapter</h2>
<h1>Third Chapter</h1>
<h2>First Section of Third Chapter</h2>
</body>
</html>
```

**Figure 1:** Example rendering of counters sample code.
Almost Standards

Most modern browsers include a layout mode that can best be described as “almost standards.” In order to support certain legacy Web pages, a page’s content renders as Standards except for images in table cells, which are treated in a manner consistent with Quirks mode. Browsers toggle into “Almost Standards” mode when they encounter particular DOCTYPEs, most notably a HTML 4.01 Transitional DOCTYPE with a system identifier.

Internet Explorer 8 Beta 2 supports this paradigm as well, though there isn’t an official layout mode labeled as such available in the DOM or in the Developer Toolbar.

Or, better yet, take advantage of the complete CSS 2.1 support in Internet Explorer 8 Beta 2 to feed the same style sheet to all modern browsers and reserve conditional comments only for down-level browsers.

Some finer points to keep in mind. First, unless you are already an expert on Conditional Comments, it pays to review the sample code and syntax on MSDN (http://msdn.microsoft.com/en-us/library/ms537509.aspx). Second, if you want to feed Internet Explorer 8 Beta 2 and Internet Explorer 7 the same content, use a greater than or equal to comparison that targets both versions in a single declaration, such as:

```html
<!--[if gte IE 7]>
  <link rel="stylesheet" type="text/css" href="/stylesheets/standards.css" />
<!--[endif]-->
</head>
```

Compatibility Mode

In addition to the Standards and Quirks modes mentioned previously, Internet Explorer 8 Beta 2 ships with a third layout mode—Internet Explorer 7 Standards mode. This layout mode is Internet Explorer 8 Beta 2’s best interpretation of how Internet Explorer 7 used to handle Standards mode pages. It’s also the quickest option for getting existing content to display correctly in Internet Explorer 8 Beta 2.

You tell Internet Explorer 8 Beta 2 to display a site or page in Internet Explorer 7 Standards mode by simply adding a tag. No additional changes required.

One cool trick Internet Explorer 8 Beta 2 performs when displaying pages in Internet Explorer 7 Standards mode involves toggling the Version Vector value to reflect the layout mode. Stated another way, Internet Explorer 8 Beta 2 running in Internet Explorer 7 Standards mode evaluates conditional comments targeted at Internet Explorer 7, which ensures the page looks correct.

With the tag, you have two options for maximum flexibility:

- On a per-site basis, add a custom HTTP header
- On a per-page basis, add a special HTML tag

Let’s look at some examples.

Suppose you have an existing Web form that looks great in Internet Explorer 7 but displays incorrectly in Internet Explorer 8 Beta 2. Sure you could debug the page and update the content to ensure compatibility with Internet Explorer 8 Beta 2’s Standards mode implementation. But suppose you have a looming deadline and just don’t have the time to fully debug and redesign the page right now. It sounds like you’re in a perfect situation to take advantage of Internet Explorer 7 Standards mode. Add the following line of markup to the <HEAD> section of your page (after the Title, but before anything else):

```html
<meta http-equiv="X-UA-Compatible" content="IE=7"> 
```

On page load, Internet Explorer reads the tag and switches the browser to interpret all content on the page as Internet Explorer 7 Standards instead of Internet Explorer 8 Standards. The toolbar of the Internet Explorer 8 Developer Tools always tells you the layout mode of the document, so use it to check your results. For more information, see “Developer Tools in Internet Explorer 8 Beta 2,” in this issue.

A second case: Suppose you have an existing Web site that contains both Quirks mode and Standards mode documents. You know the site works in Internet Explorer 7, but are unsure whether it will continue to display correctly in Internet Explorer 8 Beta 2. Rather than update each individual page with the <META> tag as per above, use a custom HTTP header.

X-UA-Compatible: IE=EmulateIE7

Configuring this option on an IIS 7.0 server is fairly straightforward.

1. Start “Internet Information Services Manager” (location varies by OS version).
2. Select the desired node in the “Connections” panel. Your options are “server”, “default Web site”, specific “site”, or “folder”.
3. In the middle pane, select the entry for “HTTP Response Headers” in the “IIS” section. See Figure 2.
4. Select “Add…” from the Actions pane.
Making Your Web site Compatible Across Multiple Versions of Internet Explorer

When you visit a Web page, your browser sends a User Agent string to the server that hosts the content. This string indicates which browser you are using, its version number, and details about your system such as operating-system version. The Web server can use this information to provide content that is tailored to your specific browser. Additionally, Internet Explorer exposes the User Agent string as part of the DOM, which allows code (such as script) running on the client to obtain the browser version details.

Internet Explorer 8 Beta 2 ships with an updated User Agent string.

Mozilla/4.0 (compatible; MSIE 8.0; Windows NT 6.0)

On the plus side, the new string allows you to easily identify the latest version of Internet Explorer. Of course, if your browser detection code doesn’t specifically account for version ‘8.0’ of Internet Explorer, problems can ensue.

A correct usage case takes into account current and future versions. In pseudo-code:

```javascript
if ( ver > -1 )
{
    if ( ver >= 7.0 )
        msg = "You're using a recent copy of Internet Explorer."
    else
        msg = "You should upgrade your copy of Internet Explorer.";
}
```

Listing 2 shows how to correctly perform User Agent sniffing via JavaScript.

A quick word of caution on User Agent sniffing, if I may... At this point in time, every User Agent string from every browser has a chunk of text copied from some other browser. (As a point of reference note the ‘Mozilla’ string in the Internet Explorer User Agent.) Browsers spoof other browser’s User Agent strings in order to obtain or maintain compatibility. As a result, you can’t always trust what you get back from User Agent string detection. If you’re trying to determine the capabilities of a particular browser, do feature testing instead. You’re far more likely to get accurate browser detection results by figuring out if the browser has the exact capability you’re looking for than to make a general assumption based on browser name and version number.

**Conclusion**

Improvements to the Web platform in Internet Explorer 8 Beta 2 help to alleviate cross-browser problems and keep you productive. The new Internet Explorer 8 Standards mode provides many features that allow for innovative, creative, and powerful site design. And, best of all, enabling these experiences is simple and straightforward—just use a DOCTYPE that triggers Standards mode. On the other end of the spectrum, the compatibility features in Internet Explorer 7 Standards mode allow existing content to display correctly in the newest edition of the browser without extensive site rewrites. Now is the perfect time to update your Web pages to support Internet Explorer 8 Beta 2.

Scott Dickens

www.code-magazine.com
Introducing Compatibility View

Compatibility View allows access to content designed for older Web browsers. When a website says that it supports modern Web standards, Internet Explorer 8 Beta 2 respects that and displays the site using its most standards compliant mechanism. In the majority of cases, this works out just fine. However, every once in a while, a page that says “display me using modern standards” really means “display me like Internet Explorer 7 used to display modern standards pages”. This is where Compatibility View comes in.

T
to set the stage, let me begin with a quick, overly simplified view of Web content. There are a set of rules which determine how websites should be displayed. Internet Explorer 8 Beta 2 follows a set of rules as defined in modern Web standards.

A ton of existing pages on the Web—the majority, actually—were developed with either no concept of these modern Web standards (developers just didn’t know or care) or were authored pre-standards (yes, there are plenty of pages that old still in use today). These older, non-standard pages work just great in Internet Explorer 8 Beta 2 as they depend on behavior that has been included in the browser for quite some time and hasn’t dramatically changed release to release.

As you might expect, there’s also a large number of pages authored according to modern Web standards and the trend is that standards mode pages will soon become the majority. Unfortunately, there are differences between modern browsers and older browsers in the level and type of standards support they provide. New standards have emerged; existing standards have been clarified; bugs have been filed.

Think of this as no different than the gasoline you pump into your car each week. At one point in time, leaded fuel was the standard. Then came unleaded. Diesel is the standard in trucks, but is also used in some cars. And, going forward, who knows—maybe alternative fuels? In any case, an original Ford Model-T supports a different fuel type than modern day Ford models.

It follows then that there are differences in standards support between browser versions. For example, Internet Explorer 8 Beta 2’s support for modern Web standards is different than Internet Explorer 7’s support for modern Web standards. Thankfully, just about every site written according to modern standards works in Internet Explorer 7 today. Some, but not all, of these will work in Internet Explorer 8 Beta 2 by default. But, there is a small class that won’t. Compatibility View allows access to this content by making the browser emulate Internet Explorer 7 behaviors when displaying Web pages.

Fast Facts

| Local files such as ‘C:\Test.htm’ display in Internet Explorer 8 Standards mode (Compatibility View off). |

Internet Explorer 7 Compatibility View

Three elements make it possible for Internet Explorer 8 Beta 2 to look and act like Internet Explorer 7 when displaying Web content.

The first of these is Layout mode, the set of HTML, CSS, and DOM rules used by the browser when evaluating markup and placing elements on the screen. Internet Explorer 8 Beta 2 ships with a Layout mode equivalent to Internet Explorer 7 Standards mode. When in Compatibility View, Internet Explorer evaluates pages that would normally trigger Internet Explorer 8 Standards mode, the latest and most standards compliant Layout mode, as mapping to Internet Explorer 7 Standards mode.

The second is User Agent string. When you visit a Web page, your browser sends a User Agent string to the server that hosts the content. This string indicates which browser you are using, its version number, and details about your system such as operating-system version. The Web server can use this information to provide content that is tailored to your specific browser. In Compatibility View, Internet Explorer 8
Beta 2 sends a User Agent string indicating a browser version of ‘7’ –
Mozilla/4.0 (compatible; MSIE 7.0; Windows NT 6.0; Trident 4.0;)

A new token indicates the underlying Trident engine version and serves as a signal that Internet Explorer 8 Beta 2 is running in Internet Explorer 7 Compatibility View.

The final element is version vector, which is used in the evaluation of Conditional Comments. Conditional Comments are “if” statements within your HTML. The code contained within a conditional comment block is only exercised if the expression evaluates to TRUE. The expression generally takes the form of a declared Internet Explorer version value and a comparison is performed against Internet Explorer’s Version Vector registry key.

To recap, when Compatibility View is enabled:

- Standards mode pages map to Internet Explorer 7 Standards mode pages (Quirks pages still map to Quirks mode)
- The User Agent string indicates a version of ‘7’
- The Version Vector, used in the evaluation of Conditional Comments, indicates a version of ‘7’

**Putting the User in Control**

A new UI button controls the Compatibility View feature. The button is located in the navigation bar just to the right of the Address bar (next to the refresh button, see Figure 1) and it only displays when toggling into Compatibility View makes sense such as when viewing Standards mode pages. In all other cases, such as when viewing Quirks mode pages, viewing intranet sites (they’re already being displayed in Compatibility View as explained later), or viewing sites with a <META> tag / HTTP header indicating Internet Explorer 7 or Internet Explorer 8 Standards, the button is hidden.

Depending on the speed of your machine, you may see the page refresh when the Compatibility View button is selected. In any case, a balloon tip lets you know that the site is now running in Compatibility View. Additionally, the Compatibility View icon shows a “pressed” state so that you can know what view you’re running in after the balloon tip disappears.

The “scope” of emulation is limited to the domain you are viewing when you press the button, not some other mechanism like the life of the process or the tab. And, Internet Explorer remembers your button press so that the next time you visit the site you don’t have to press the button again.

**Compatibility View and Local Intranet**

A large number of line-of-business websites located on corporate intranets are Internet Explorer 7 capable today, but may not be Internet Explorer 8 Beta 2 capable by default. In order to preserve compatibility, Internet Explorer 8 Beta 2 ships with smart defaults based on zone evaluation. In the default state, all Internet websites display in Internet Explorer 8 Standards mode (Compatibility View off) and all intranet websites display in Internet Explorer 7 Standards mode (Compatibility View on).

**And, of course, all of the feature’s settings, including the advanced configuration options, are configurable via Group Policy and the Internet Explorer Administration Kit (IEAK).**

Let’s look at some examples.

If you navigate to sites on the Internet like www.msn.com and www.live.com, Compatibility View is off by default. Internet Explorer 8 Beta 2 identifies itself with a User Agent string of ‘8’, Version Vector of ‘8’ and displays Web pages that trigger standards mode in Internet Explorer 8 Standards mode. The same is also true if you navigate by IP address, such as http://192.168.0.1. As Internet Explorer can’t tell offhand whether the IP address is internal or external, it assumes the latter. Use Compatibility View to fix problems with websites in this category.

If you navigate to sites on the intranet like http://myPortal and http://sharepoint/sites/mySite Internet Explorer 8 Beta 2 identifies itself with a User Agent string of ‘8’, Version Vector of ‘7’, and displays Web pages that trigger Standards mode in Internet Explorer 7 Standards mode.

A new entry in the Tools menu allows for advanced configuration of the feature. You can make it so

---

**Figure 1: Compatibility View button.**
that all intranet sites display in Internet Explorer 8 Standards mode. You can also configure policy such that every site is viewed in Internet Explorer 7 Compatibility View (identical to the behavior from Internet Explorer 8 Beta 1 when the ‘Emulate IE7’ button was pressed). Lastly, you can pre-populate a list of sites (Figure 2) that should always be viewed in Compatibility View and/or edit the current entries.

Making Your Pages Work with Internet Explorer 8 Beta 2

The best way to ensure users have a great experience with your website, and thus don’t have to use the Compatibility View feature at all for Internet sites, is to test your site using Internet Explorer 8 Beta 2 and update it as necessary.

1. Navigate to your website using Internet Explorer 8 Beta 2.

2. If the pages work great, add the ‘EmulateIE8’ <META> tag / HTTP header. Doing so ensures that new site visitors will never see the Compatibility View button when viewing your site. Additionally, the <META> tag / HTTP header has the added benefit of “bringing back” those users who may have selected Compatibility View for your website before you had a chance to test and add the <META> tag / HTTP header. The tag / header forces Compatibility View off.

3. If pages look differently than they did in Internet Explorer 7 or don’t work the same way, enable Compatibility View.

4. If switching on Compatibility View makes your pages render and function as they do in Internet Explorer 7, then add the ‘EmulateIE7’ <META> tag / HTTP header to immediately get your site working.

5. Long term, update your site to support Internet Explorer 8 Standards mode by authoring your page according to modern Web standards supported by Internet Explorer and including a Standards mode DOCTYPE.

For more information on using the <META> tag / HTTP header, see the article “Making Your Website Compatible Across Multiple Versions of Internet Explorer”.

Conclusion

Compatibility View is an end-user site compatibility mitigation option for gaining access to content built for older Web browsers. When a user leverages the feature for a particular website, components of the Internet Explorer subsystem—the User Agent String, the Version Vector, and Layout mode—change to resemble the way things were in Internet Explorer 7.

Unfortunately, most users will only discover the Compatibility View feature after they’ve had a less than stellar experience with a particular Internet facing website. You can prevent such an event from happening on your website by testing your pages in Internet Explorer 8 Beta 2.

Scott Dickens

![Compatibility View Settings](image1.png)

Figure 2: Compatibility View Settings.
Create Your Own Web Slices

Web Slices are a new Internet Explorer 8 Beta 2 feature that allow you to subscribe to parts of a Web page and view updates directly from the Internet Explorer Favorites bar. Web Slices let users bring their favorite parts of the Web with them wherever they browse. This article will explain how Web Slices work, why they are cool, and how you can start creating Web Slices of your own.

Web Slices are a new concept that allows the user to subscribe to a portion of a page and monitor the updates on the status bar throughout their browsing experience. Any portion of a page that updates regularly is a reasonable candidate to become a Web Slice. News, stock quotes, sports scores, exchange rates or even e-mail can be presented prominently in Internet Explorer as part of the Favorites bar. Updates to Web Slices are indicated visually in the item through bolded text.

There are four main behaviors for a Web Slice:

- **Discovery:** Users discover the Web Slice within a Web page and subscribe to it, adding the Web Slice to the Favorites bar.
- **Update:** Internet Explorer uses the Feed Download Engine to periodically check for changes to the Web Slice.
- **Notify:** When the Web Slice changes, it is bolded on the Favorites bar.
- **Display:** The user can view the update to the Web Slice by clicking on the Web Slice on the Favorites bar.

There are two ways for users to discover a Web Slice: in document by hovering over a Web Slice region on a page (Figure 1), and through the feed discovery button on the command bar (Figure 2).

Once a Web Slice has been subscribed to it can be accessed through the Favorites bar. Figure 3 shows an eBay Web Slice that allows users to track an auction. Rather than clicking refresh on an auction

### Fast Facts

Web Slices are specialized forms of feeds that can be updated by the site author and pushed to users.

Seth McLaughlin
seth.mclaughlin@microsoft.com

Seth is a Program Manager on the Internet Explorer User Experience team.

Figure 1: In page discovery of a Web Slice.

Figure 2: Discovery of a Web Slice through the feed button on the command bar.
The id attribute

The id value is used to identify the Web Slice in a URL, similar to the way a named anchor (bookmark) is used. The id attribute must be present on the same element that uses the hslice class.

The id should be unique. If two Web Slices on the page share the same id value, Internet Explorer will ignore one of them.

Note: If the id value changes, Internet Explorer will no longer be able to identify the Web Slice.

entry-title class

At least one child of the hslice must specify an entry-title class name. The value of this property maps to the Web Slice button text and the advertised name on the Feed Discovery button.

hslice class

To detect a Web slice on the page, Internet Explorer looks for elements that have a class of hslice. The hslice element describes a section of the Web page that the browser can subscribe to. All required and optional properties of a Web Slice must be contained within the hslice element.

Web Slice class names do not require a CSS rule in the style sheet as do traditional style classes. You can combine Web Slice class names with other classes using a space-delimited list.

Figure 3: Sample eBay Web Slice.
In this example, Internet Explorer displays “Seattle Weather 62°” when the Web Slice is updated.

### Optional Elements

These elements for a Web Slice in code are optional.

- entry-content class
- rel=feedurl
- ttl
- endtime

#### entry-content class

The **entry-content** element is the part of the Web Slice that Internet Explorer brings to the Favorites bar. The entire element is displayed in the Web Slice preview window when the user clicks the Web Slice button.

The default width and height of the preview window is determined by the size of the container designated as **entry-content**.

Like the **entry-title** class, the Web Slice can contain more than one **entry-content** element. All elements are combined to create a single preview.

*Note: For security reasons, the Web Slice preview does not allow script or Microsoft ActiveX controls. Consequently, HTML forms and buttons are not supported. Instead, you can use links that contain URL parameters to perform related actions. Navigations occur within the currently selected tab. To get around this limitation, consider using an alternative display source, described below.*

The inline styles used in the HTML of the **entry-content** element and the style rules of the original Web page are used to style the HTML in the preview window. Inline styles from parent elements are not inherited by the preview. For more information, see the [Applying Styles to Web Slices](#) section below.

#### rel=feedurl

Any link within the Web Slice that specifies a **feedurl** rel attribute will be treated as an alternate source for updates. The alternate source can be another Web page with Web Slice annotations or a single feed item. When present, Internet Explorer subscribes to the alternate update source and no longer uses the original page content to update the Web Slice.

### Web Slice in the Browser

Once a Web Slice has been subscribed to, it can be accessed through the Favorites bar. Each time there is an update to the Web Slice’s content, the Web Slice will gleam to alert the user.

#### Alternative Update Source

The alternate update source is useful to manage the background updates from the client on a separate server. This is done by providing an alternative Web page or feed, using the **feedurl** property (discussed above). Figure 4 demonstrates how this process works. In the diagram, you can see that there are two Web pages that represent the same Web Slice. **Basic.html** is the original Web page that the user discovers and subscribes to the Web Slice. **Update.html** is linked to from Basic.html, and serves as the alternate update Web page. You can see this in code below.

![Alternative Update Source](chart.png)

**Figure 4: Using an alternative update source for a Web Slice.**
Create Your First Web Slice

To demonstrate the concepts discussed above, I’ll show you how to create a simple Web Slice. Base 9 is a (fictional) local jazz band. They use their website to promote upcoming shows and want to create a Web Slice to notify users when a new show is scheduled.

The first step is to decide exactly what content will be displayed in the Web Slice and how it will be laid out on the page. The Internet Explorer 8 Beta 2 Web Slice Style Guide provides best practices for the design of Web Slices. Refer to the style guide to ensure that the content and layout of your Web Slices are optimal.

The Web Slice will contain five items:

1. A title
2. A picture of the band
3. The location of the show
4. The date and time of the show
5. A link to buy tickets

Using the elements discussed above, it is straightforward to put together the code for this Web Slice:

```
<div class="hslice" id="upcoming_show" style="width: 320px; height: 240px">
  <div class="entry-content" style="width: 320px; height: 240px">
    <h2 class="entry-title" style="text-align: center">Base 9 show on 7/9</h2>
    <img src="band.jpg" style="width: 100px; height: 100px; float:left; margin-right: 20px">
    <ul>
      <li>Where: Hotel Sierra</li>
      <li>When: Friday 7/9/08 @ 5pm</li>
      <li><a href="http://buytickets.com">Buy tickets for the show</a></li>
    </ul>
  </div>
</div>
```

While the example here uses inline styles, it is also possible to use styles defined within the <style> block, or through an external style sheet. To attach the class myclass to the hslice element, separate the two class names with a space:

```
Basic.html:

...<div class="hslice" id="auction" myclass="myclass">
...<h2 class="entry-title">Base 9 show on 7/9</h2>
...<img src="band.jpg" style="width: 100px; height: 100px; float:left; margin-right: 20px">
...<ul>
...<li>Where: Hotel Sierra</li>
...<li>When: Friday 7/9/08 @ 5pm</li>
...<li><a href="http://buytickets.com">Buy tickets for the show</a></li>
...</ul>
</div>
```
It is important to note that the page containing your Web Slice must be hosted on a Web server (not accessed locally) for Internet Explorer to detect the content and allow the user to subscribe to the Web Slice.

Once a user adds the slice, it appears on the Favorites bar as shown in Figure 6.

This HTML code can then be updated (manually or through a script) to reflect upcoming concert dates for the Base 9 band. It is a quick and easy way for Base 9 fans to keep tabs on when their favorite band will be performing!

**Conclusion**

Creating a Web Slice is very easy and is a great way to promote your site’s content to users directly from the Internet Explorer Favorites bar.

For more information on creating Web Slices, please visit the Internet Explorer Developer Center at [http://www.msdn.com/ie](http://www.msdn.com/ie). There you’ll find the Internet Explorer 8 Beta 2 Web Slice Style Guide, which provides best practices for the visual design of Web Slice content. To explore Web Slices other developers have created, check out the new Web Slice Gallery.

*Seth McLaughlin*

---

**Advertisers Index**

<table>
<thead>
<tr>
<th>Advertiser</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Active Software Professionals</td>
<td>12</td>
</tr>
<tr>
<td><a href="http://www.aspalliance.com">www.aspalliance.com</a></td>
<td></td>
</tr>
<tr>
<td>Applied Innovations</td>
<td>76</td>
</tr>
<tr>
<td><a href="http://www.appliedi.net">www.appliedi.net</a></td>
<td></td>
</tr>
<tr>
<td>Aspose</td>
<td>75</td>
</tr>
<tr>
<td><a href="http://www.aspose.com">www.aspose.com</a></td>
<td></td>
</tr>
<tr>
<td>Code Magazine</td>
<td>7</td>
</tr>
<tr>
<td><a href="http://www.code-magazine.com">www.code-magazine.com</a></td>
<td></td>
</tr>
<tr>
<td>Countersoft</td>
<td>5</td>
</tr>
<tr>
<td><a href="http://www.countersoft.com">www.countersoft.com</a></td>
<td></td>
</tr>
<tr>
<td>Developer Express</td>
<td>2</td>
</tr>
<tr>
<td><a href="http://www.devexpress.com">www.devexpress.com</a></td>
<td></td>
</tr>
<tr>
<td>DevMavens</td>
<td>6</td>
</tr>
<tr>
<td><a href="http://www.devmavens.com">www.devmavens.com</a></td>
<td></td>
</tr>
<tr>
<td>EPS Software Corp.</td>
<td>38</td>
</tr>
<tr>
<td><a href="http://www.eps-software.com">www.eps-software.com</a></td>
<td></td>
</tr>
<tr>
<td>Microsoft</td>
<td>13</td>
</tr>
<tr>
<td><a href="http://www.microsoft.com/ie8/developer">www.microsoft.com/ie8/developer</a></td>
<td></td>
</tr>
<tr>
<td>West Wind Technologies</td>
<td>61</td>
</tr>
<tr>
<td><a href="http://www.west-wind.com">www.west-wind.com</a></td>
<td></td>
</tr>
<tr>
<td>Xiine</td>
<td>71</td>
</tr>
<tr>
<td><a href="http://www.xiine.com">www.xiine.com</a></td>
<td></td>
</tr>
</tbody>
</table>

This listing is provided as a courtesy to our readers and advertisers. The publisher assumes no responsibility for errors or omissions.
Accelerators in Internet Explorer 8 Beta 2

Connect the user to your Web application with Accelerators!

Accelerators provide an easier and faster way for users to access your Web application while seamlessly plugging in to your existing infrastructure.

Accelerators, formerly known as Activities, are a new concept that allows users to interact with a Web service or application from the browsing experience. Users typically copy and paste text from one Web page to another for various reasons. Internet Explorer 8 Beta 2 Accelerators make this much easier to do.

The browser is becoming more than just a place to read content on the Web. Increasingly, users are interacting with Web content in several different ways. Examples include pasting selected text in a blog post, mapping out an address, or checking the definition of a word. With Accelerators, users can reach beyond the page to various services for these common tasks.

Accelerators are easy to create for Web developers and can be written to plug in to existing applications. They are great ways to attract users to your website through a single click!

Select → Search, Map, Blog!

Accelerators work on selected text, hyperlinks and the current document. The user sees different options in the Accelerators menu depending upon the type of content. The most common content is selection, when a user makes a selection the Accelerator button appears. Clicking the button displays the set of Accelerators the user has installed that work for selected text.

Alternatively, users can also bring up the Accelerators menu through the context menu of the selected text or hyperlink. They can access the current page’s Accelerators from the context menu of the page or the Page button on the Internet Explorer Command bar.

Accelerators typically involve two types of scenarios: “look up” of information within a Web page, and “sending” of content to a Web application.

Consider the example of a user who is interested in a restaurant and wants to see its location. Figure 1 illustrates this form of a “look-up” Accelerator, where the user selects the address and sees a quick preview of the map with an Accelerator.

In an example of a “send” Accelerator, a user reads an interesting article and wants to include it in a blog post. The user can select a portion of the article and use an Accelerator which navigates to the user’s blog site with the selection already available in the edit field.

Web Developers, Write Your Own Accelerators Easily

Accelerators plug in easily with existing HTTP GET/POST architecture. Building an Accelerator is as easy as specifying a set of parameters (Codeless Accelerators) or by using a COM object (Code-based Accelerators) for local application integration.

Fast Facts

Accelerators enable typical “look up” or “send” scenarios for selected text, hyperlinks and the current page. Web developers can easily write an Accelerator that plugs into their existing Web application.

Figure 1: Preview of a restaurant’s location using Accelerators.

What Happened to the Name “Activities”?

“Activities” was the name of the feature in the Internet Explorer 8 Beta 1 release. The naming process involves brainstorming, direct user research/rating, linguistics evaluations, legal searches, etc. and the final name “Accelerators” aligns with the overall Internet Explorer 8 brand.

Ritika Virmani
ritika.virmani@microsoft.com

Ritika is a Program Manager on the Internet Explorer Services Integration team and works on Navigation in Internet Explorer.

Ritika Virmani
ritika.virmani@microsoft.com

Ritika is a Program Manager on the Internet Explorer Services Integration team and works on Navigation in Internet Explorer.

Accelerators in
Internet Explorer 8 Beta 2

Connect the user to your Web application with Accelerators!

Accelerators provide an easier and faster way for users to access your Web application while seamlessly plugging in to your existing infrastructure.

Accelerators, formerly known as Activities, are a new concept that allows users to interact with a Web service or application from the browsing experience. Users typically copy and paste text from one Web page to another for various reasons. Internet Explorer 8 Beta 2 Accelerators make this much easier to do.

The browser is becoming more than just a place to read content on the Web. Increasingly, users are interacting with Web content in several different ways. Examples include pasting selected text in a blog post, mapping out an address, or checking the definition of a word. With Accelerators, users can reach beyond the page to various services for these common tasks.

Accelerators are easy to create for Web developers and can be written to plug in to existing applications. They are great ways to attract users to your website through a single click!

Select → Search, Map, Blog!

Accelerators work on selected text, hyperlinks and the current document. The user sees different options in the Accelerators menu depending upon the type of content. The most common content is selection, when a user makes a selection the Accelerator button appears. Clicking the button displays the set of Accelerators the user has installed that work for selected text.

Alternatively, users can also bring up the Accelerators menu through the context menu of the selected text or hyperlink. They can access the current page’s Accelerators from the context menu of the page or the Page button on the Internet Explorer Command bar.

Accelerators typically involve two types of scenarios: “look up” of information within a Web page, and “sending” of content to a Web application.

Consider the example of a user who is interested in a restaurant and wants to see its location. Figure 1 illustrates this form of a “look-up” Accelerator, where the user selects the address and sees a quick preview of the map with an Accelerator.

In an example of a “send” Accelerator, a user reads an interesting article and wants to include it in a blog post. The user can select a portion of the article and use an Accelerator which navigates to the user’s blog site with the selection already available in the edit field.

Web Developers, Write Your Own Accelerators Easily

Accelerators plug in easily with existing HTTP GET/POST architecture. Building an Accelerator is as easy as specifying a set of parameters (Codeless Accelerators) or by using a COM object (Code-based Accelerators) for local application integration.

Fast Facts

Accelerators enable typical “look up” or “send” scenarios for selected text, hyperlinks and the current page. Web developers can easily write an Accelerator that plugs into their existing Web application.

Figure 1: Preview of a restaurant’s location using Accelerators.

What Happened to the Name “Activities”?

“Activities” was the name of the feature in the Internet Explorer 8 Beta 1 release. The naming process involves brainstorming, direct user research/rating, linguistics evaluations, legal searches, etc. and the final name “Accelerators” aligns with the overall Internet Explorer 8 brand.

Ritika Virmani
ritika.virmani@microsoft.com

Ritika is a Program Manager on the Internet Explorer Services Integration team and works on Navigation in Internet Explorer.

Accelerators in
Internet Explorer 8 Beta 2

Connect the user to your Web application with Accelerators!

Accelerators provide an easier and faster way for users to access your Web application while seamlessly plugging in to your existing infrastructure.

Accelerators, formerly known as Activities, are a new concept that allows users to interact with a Web service or application from the browsing experience. Users typically copy and paste text from one Web page to another for various reasons. Internet Explorer 8 Beta 2 Accelerators make this much easier to do.

The browser is becoming more than just a place to read content on the Web. Increasingly, users are interacting with Web content in several different ways. Examples include pasting selected text in a blog post, mapping out an address, or checking the definition of a word. With Accelerators, users can reach beyond the page to various services for these common tasks.

Accelerators are easy to create for Web developers and can be written to plug in to existing applications. They are great ways to attract users to your website through a single click!

Select → Search, Map, Blog!

Accelerators work on selected text, hyperlinks and the current document. The user sees different options in the Accelerators menu depending upon the type of content. The most common content is selection, when a user makes a selection the Accelerator button appears. Clicking the button displays the set of Accelerators the user has installed that work for selected text.

Alternatively, users can also bring up the Accelerators menu through the context menu of the selected text or hyperlink. They can access the current page’s Accelerators from the context menu of the page or the Page button on the Internet Explorer Command bar.

Accelerators typically involve two types of scenarios: “look up” of information within a Web page, and “sending” of content to a Web application.

Consider the example of a user who is interested in a restaurant and wants to see its location. Figure 1 illustrates this form of a “look-up” Accelerator, where the user selects the address and sees a quick preview of the map with an Accelerator.

In an example of a “send” Accelerator, a user reads an interesting article and wants to include it in a blog post. The user can select a portion of the article and use an Accelerator which navigates to the user’s blog site with the selection already available in the edit field.

Web Developers, Write Your Own Accelerators Easily

Accelerators plug in easily with existing HTTP GET/POST architecture. Building an Accelerator is as easy as specifying a set of parameters (Codeless Accelerators) or by using a COM object (Code-based Accelerators) for local application integration.

Fast Facts

Accelerators enable typical “look up” or “send” scenarios for selected text, hyperlinks and the current page. Web developers can easily write an Accelerator that plugs into their existing Web application.

Figure 1: Preview of a restaurant’s location using Accelerators.

What Happened to the Name “Activities”?

“Activities” was the name of the feature in the Internet Explorer 8 Beta 1 release. The naming process involves brainstorming, direct user research/rating, linguistics evaluations, legal searches, etc. and the final name “Accelerators” aligns with the overall Internet Explorer 8 brand.

Ritika Virmani
ritika.virmani@microsoft.com

Ritika is a Program Manager on the Internet Explorer Services Integration team and works on Navigation in Internet Explorer.
This article focuses on writing Codeless Accelerators for existing Web applications. Enabling Codeless Accelerators allows existing services to easily integrate with the browser without having to change the service implementation. They are secure and easy to install.

OpenService Description Format

The OpenService Description Format provides a structured way to describe an Accelerator to the browser. Internet Explorer 8 Beta 2 uses this XML file to know how to display the Accelerator to the user and how to send content to the service.

The example of an Accelerator in Listing 1 illustrates the OpenService Description Format.

The XML namespace for Accelerators is http://www.microsoft.com/schemas/openservicedescription/1.0

The main components of the format are:

- homepageUrl
- display
- activity

homepageUrl element

The homepage URL along with the category is the identity of the Accelerator and is a required parameter. This is the URL where the user can access the Web application through browsing.

All the URLs expressed in the file must match the same domain as the homepage URL.

display element

These elements govern how the user sees the Accelerator. These include the name of the Accelerator that is displayed in the menu and an optional icon and short description. The name is required and should typically be in the “verb with service” format. Example: “Search with LiveSearch”

activity element

The elements contained within the activity element describe the functionality of the Accelerator.

These are some of the options you can specify for your Accelerator:

- category
- content type it works on
- HTTP method it uses to send the content to your service
- URI template for HTTP submission
- encoding and character set information

The example in Listing 1 illustrates how a few of these can be set by using the OpenService Description Format.

Choose a Category for Your Accelerator

Accelerators are grouped by category when displayed to the user for users to find a service based on its functionality. The user can select a default Accelerator for each category and these are displayed in the top level Accelerator menu. All other Accelerators are listed in the second level menu as shown in Figure 2.

Listing 1: Accelerators example

```xml
<?xml version="1.0" encoding="UTF-8 Beta 2"?>
<openServiceDescription xmlns="http://www.microsoft.com/schemas/openservicedescription/1.0">
  <!-- Identity of the Accelerator -->
  <homepageUrl>http://maps.live.com</homepageUrl>

  <!-- How the Accelerator is displayed to the user -->
  <display>
    <name>Map with Live Maps</name>
    <icon>http://www.live.com/favicon.ico</icon>
  </display>

  <!-- Functionality of the Accelerator -->
  <activity category="map">
    <activityAction context="selection">
      <preview action="http://maps.live.com/geotager.aspx" />
      <parameter name="b" value="{selection}" />
      <parameter name="clean" value="true" />
      <parameter name="w" value="320" />
      <parameter name="h" value="240" />
      <parameter name="format" value="full" />
    </preview>
    <execute action="http://maps.live.com/default.aspx">
      <parameter name="where1" value="{selection}" />
      <type="text" />
    </execute>
  </activityAction>
</activity>
</openServiceDescription>
```
Building an Accelerator is as easy as specifying a set of parameters (XML-based Codeless Accelerators) or using a COM object (Code-based Accelerators).

Table 1 shows some of the categories that Internet Explorer promotes, along with examples of existing services.

Choose the category that best fits your Accelerator. If your category is unique and none of the recommended categories align with your Accelerator, you can define your own. The following example defines the category "share".

```
<activity category="share">
    <!-- Your implementation here -->
</activity>
```

Your category should meet the following criteria:

- It should be a verb that users can recognize
- It should be generic enough so that other Accelerators can share it
- It should not be tied to a specific brand or application

How It All Works

Accelerators are displayed in the Accelerators menu based on the content type (the context attribute of the Accelerator). Internet Explorer queries the installed Accelerators to get the available Accelerators that can run on this content type.

The content (selected text/hyperlink/document) is submitted to the service to handle using GET/POST submissions specified in the Accelerator. The Accelerator expresses the data to send using form-based or inline parameters.

Cookies can be used to store state and user credentials as part of the HTTP request.

A variable is enclosed in brackets, as in {documentUrl}. A variable can be specified as optional by using a "?" after the variable name; for example, {documentUrl}?

Inline Parameters: These specify the inputs to a service through the URI template.

Form-Based Parameters: These specify the inputs to a service through name-value pairs, similar to HTTP form submissions. This is useful for making HTTP POST requests, or if the HTTP GET request could be longer than the URL character limit.

```
<execute method="get"
```

Using the Preview Window

Accelerators provide an optional HTML preview window that displays when the user hovers over an Accelerator menu item. The preview window is a great place for quick actions like viewing an in-place definition, translation or even a map.

This functionality is optional and should only be used for lightweight interaction. The idea is to provide users with a quick preview and encourage them to go to your website by clicking on a hyperlink within the preview or by clicking on the Accelerator menu item.

Preview functionality is described through the Accelerator XML within the preview element. During the hover, Internet Explorer makes an HTTP GET/POST request and displays the resulting HTML within the HTML preview window.

The preview window supports JavaScript sandboxed to the domain of the Accelerator. It also supports ActiveX controls that are already installed and opted-in by the user for the given domain.

Localization for Accelerators

Each Accelerator XML file represents a single locale. If your service works for multiple languages, you will need multiple XML files. After identifying the user's preferred language you can provide the user the appropriate file to install.

```
<execute method="post"
    action="http://example.com/done.aspx">
    <parameter name="url" value="{documentUrl}" />

    <parameter name="title" value="{documentTitle}" />
</execute>
```
Following is an example of an Accelerator in Spanish where the display and category have been localized.

```xml
<!-- Localize the text displayed to the user -->
<display>
  <name>Mapa con Live Maps</name>
  <icon>http://www.live.com/favicon.ico</icon>
</display>

<!-- Localize the category. Map is mapa in Spanish -->
<activity category="mapa">
  ...
</activity>
```

**Getting the User to Install Your Accelerator**

You can promote your Accelerators from your website. Internet Explorer has a browser API that is accessible by Web pages and supports easy installs through the click of a button.

The following example checks if an Accelerator is installed:

```javascript
window.external.isServiceInstalled
  ("http://maps.live.com/GetMap.xml", "map"); <!-- map is the category -->
```

The following example displays a button that installs an Accelerator:

```html
<button onclick="window.external.AddService
  ("http://maps.live.com/GetMap.xml")">
  Add map Activity</button>
```

**Putting It All Together**

Through this article, you can see that Accelerators are a great way to attract users to your website and very easy to write for Web developers. Once you get used to Accelerators, copying text, opening a new tab, navigating to a website and pasting text seems very old school.

For Codeless Accelerators, in essence you need to specify the display elements, the category, the content type of the input and the parameters that need to be sent to your service. You can optionally use the preview window to display HTML content returned from your service or the user can navigate to your website by clicking on the Accelerator menu item.

Listing 2 is a simple checklist to make sure that your service is consistent with other Accelerators and takes advantages of the Internet Explorer 8 Beta 2 functionality.

**Listing 2: Accelerators checklist**

1. **Display Name – make it descriptive**
   The display name should lead with the verb, article, and then the product/service. The verb should be the category. This should not exceed 50 characters. Here are some examples:
   - Map with Live Maps (category="map")
   - Blog with Windows Live Spaces (category="blog")

2. **Categories – use a recommended category**
   Categories are a way to group similar Accelerators so that users can quickly find what they are looking for. IE allows Web developers to define their own category, but suggest using a recommended category.

   - Blog: A blog service that creates a new blog post based on a link or user-selection
   - Bookmark: A service adds a link to the user’s personal bookmarks on the Web
   - Define: A service that provides definitions based on selected text
   - Email: A service that can create a new email message
   - Find news: A service that finds related articles within the scope of the site
   - Find product: A service that finds related products within the scope of the site
   - Find: A service that finds related content within the scope of the site
   - Map: A service that provides maps for a selected address
   - Send: A service that converts Web data into application data
   - Share: A service that shares a link with the site community or network
   - Translate: A service that translates the current Web page or selection to another language

3. **Preview – entice users to your site**
   The preview is the window that pops up when users are hovering over theAccelerator. Use it as a visual way to attract users to your site. These should be simple views that load quickly (under 500ms) and do not require any interactivity as this creates an undesirable user model (form fields, tabs to switch to another view, etc.)

4. **Content – not just selection**
   Accelerators are available on text selection, links, and the current Web page. You can also create one Accelerator that uses all three contexts.

5. **Tests – make sure it works**
   Test your Accelerator on different types of content so that it works properly.
   a. Blank content – what happens when blank content is sent to your service?
   b. Multi-line content – does your service handle line-breaks?
   c. Script – Some user selection may have JavaScript associated to it. If you specify HTML selection, then your service should be filtering this script on the server.

**Does the Name Change to “Accelerators” Affect the XML and APIs?**

No. The OpenService XML format will remain the same as in Beta 1 and the APIs will continue to use “Activities.” This is done to ensure that the transition from Beta 1 to Beta 2 does not require sites to update their content.

**Ritika Virmani**
New Search Features in Internet Explorer 8 Beta 2

Can you imagine a world without search? I don’t think a single day goes by when I don’t do at least one Internet search. Internet search has gotten leaps and bounds better than what it started out as years ago; yet aside from adding a search box to the browser chrome, browsers, as of yet, have not significantly changed the way that users search online.

With Internet Explorer 8 Beta 2, Microsoft is taking some big steps that we hope will change the way you search. Our goal is to enable you to search for the right term on the right provider and get the right information as quickly as possible. With contributions from your favorite search providers, we are now providing suggestions in Internet Explorer 8 Beta 2 that can help you formulate great search terms. In addition, in Internet Explorer 8 Beta 2 it is now very easy to search with any search provider. You can easily install multiple providers and switching over to use them is simple. We know you don’t just search on Google or Yahoo or Windows Live; Search with Wikipedia, eBay, or your favorite online store is just as easy.

What Can You Do with Search in Internet Explorer 8 Beta 2?

Let’s imagine that you want to use Internet Explorer 8 Beta 2 to conduct some Internet searches to plan a night out on the town (in this example, Seattle). You might want to start out by searching for “seattle restaurants”, so you start typing in the search box. As you are typing, one of the first things you’ll notice is Internet Explorer 8 Beta 2’s new search suggestions as seen in Figure 1.

Search suggestions appear as you are typing and offer you some ideas on things you might like to search for. In fact, as I type “seattle” I see “seattle restaurants” as one of the suggestions. Clicking on the suggestion brings me right to the results page. This is great for me as restaurant is one of those words that I chronically misspell. (I’ve used Word’s spelling correction feature twice already as I’m typing this article so not having to type out the full word in the search box is great for me.)

Figure 1: Internet Explorer’s new search box dropdown. At the top of the dropdown are previously typed search terms. The next section contains the search suggestions, after which are history results, and finally at the bottom is the QuickPick menu.

Figure 2: The search box showing some visual suggestions. These visual suggestions highlight particularly valuable suggestions like the Space Needle, the Mariners and the Seahawks.

Our goal is to enable you to search for the right term on the right provider and get the right information as quickly as possible.

Almost any website that has a search box can be used as an Internet Explorer search provider. Creating an OpenSearch description file for your website only takes a few minutes and once your users add your website to their search box they can access it at any time.
As you type more, you will notice that some suggestions include small images. These are called visual suggestions. Visual suggestions may cue you in to particularly important suggestions such as in Figure 2 where “seattle space needle,” “seattle mariners” and “seattle seahawks” have been highlighted with visual suggestions.

In addition to highlighting interesting suggestions, visual suggestions can also be used to provide some bit of information such as in Figure 3 where I’ve typed “seattle weather”.

There are two additional sections you’ll notice under the search box. The first section, directly under the search box contains your previously typed search terms. From observation, we know that people tend to search for the same things over and over again. Maybe you read an article last week that you want to go back to, or maybe like me, there are sites that you visit infrequently enough that you can’t remember the URL, you don’t have in your Favorites folder and you just know how to get to because every time you search for “seattle metro transit” (for example), the first result is the King County Metro website.

The other section you’ll notice under the search box is a section of results from your History. This section is new to Internet Explorer 8 Beta 2. Using Windows Search, Internet Explorer 8 Beta 2 will search over your stored history. The search results are based on the pages’ title and URL. Perhaps you didn’t actually do a search for that interesting news article that you read last week. You might have just been browsing your favorite news site. With history results under the search box, you can find the page you were looking for without ever doing a search.

In all three sections under the search box, we have added hit highlighting so that you can glance through the results and know exactly why each one was suggested to you. The search box dropdown is also consistent in look and feel to the address bar. In normal use, the address bar will display results from your Favorites, History, and Feeds, however, you can also switch the address bar over to search mode by typing “?” followed by a space and then your search term. This will allow you to get search suggestions in the address bar.

Now that you’ve seen what’s available under the search box, let’s go back to planning a night out on the town. Having issued the query for “seattle restaurants” we are now looking at a search results page. One of the results that came up is Wild Ginger, which sounds like a good choice. However, thinking about it more, it would be a really good idea to read what the Seattle Post-Intelligencer newspaper (Seattle P-I) has to say about Wild Ginger since they frequently do restaurant reviews. With the Seattle P-I installed as a search provider, you can just select the name of the restaurant and click on the Accelerators icon. There you will find all your installed search providers and clicking on one you can issue a search. This is just like how Accelerators work. In Figure 4 you can see how your search providers are available as accelerators.

Any installed search provider will automatically behave like an Accelerator. Your default search provider will be listed as the default in the Accelerator list and your secondary providers will be listed in the over-
flow menu. Search providers don’t have to do any extra work to enable Accelerator support—this is done automatically whenever you install a search provider. Search providers can even support Accelerator previews if they want, although this does require some new work on the part of the search provider.

After issuing your search via the Accelerator menu, you will now be viewing a search results page for Wild Ginger. After reading the review of Wild Ginger, you decide that you aren’t in the mood for Asian food tonight and would rather go for Italian. So you search for Italian restaurants on the Seattle P-I site. If you are paying attention, you’ll notice that the text “Italian restaurants” has now been placed in the search box. This is because Internet Explorer 8 Beta 2 detects searches that you do from within the websites of your installed search providers.

One of the biggest advantages of the search box is that it allows you to initiate a search from anywhere. You don’t have to first go to your provider’s home page. But we know that once users are already at their search provider’s website they may change or refine their query on the site. By keeping track of your most recent search query string, we have made it very easy to switch over to a different search provider if you decide that a different provider is more appropriate to handle your query. For example, you could decide that instead of searching for restaurants on one site, you would rather search on another. In which case, your most recent search term is all ready for you in the search box. Speaking of which...

Switching search providers in Internet Explorer 8 Beta 2 is much improved from how it was in Internet Explorer 7. Internet Explorer 8 Beta 2 has a nifty little menu under the search box called the QuickPick menu. Here you will see the favicons of all your installed search providers. Clicking on one of these icons lets you execute a search with that provider. The QuickPick menu will appear when you first click into the search box and it will remain accessible as you begin typing. Once you start typing the QuickPick menu moves to the bottom of the dropdown below your history results. We really hope that this will make secondary providers more accessible and encourage everyone to use the search box for all the sites they use, not just for search sites.

Within the QuickPick menu, you also have the option to access the Internet Explorer Find on Page feature. Find on Page in Internet Explorer 8 Beta 2 is much improved from what was available in Internet Explorer 7. Find on Page is now a single line bar that drops down from the Internet Explorer chrome. It supports highlighting of all the hits within the website and highlights the hits as you are typing instead of having to press enter when you are finished. In addition, Find on Page will display to you the number of hits within the page. Figure 5 shows IE8’s new Find on Page functionality.

If you have any search features on your website that worked with Internet Explorer 7, those will continue to work with Internet Explorer 8 Beta 2.

Figure 5: Internet Explorer 8 Beta 2’s new Find on Page bar doesn’t block any of the page content. It will highlight all the matches within the page as you type your query.
How Do You Create Search Features for Internet Explorer 8 Beta 2?

Having seen all the great search features available in Internet Explorer 8 Beta 2, the question you may have as a website owner is how your users can start using these great features with your site. Well, I have some great news to start with off. If you have any search features on your website which worked with Internet Explorer 7, those will continue to work with Internet Explorer 8 Beta 2.

The first step in creating search features for Internet Explorer 8 Beta 2 is to enable users to install your website as a search provider in the search box. To do this, there are two steps you will take. The first step is to create an OpenSearch description file of your search provider and the second step is to expose the description file on your website so that users can install it.

Creating OpenSearch Description Files

OpenSearch is a collaborative community started originally by Amazon and A9 that has created a specification for how to describe search providers and search results. The great thing about using OpenSearch is that it is used by Internet Explorer 7, Internet Explorer 8 Beta 2, Firefox 2, and Firefox 3. You can find more details about the organization and the full specification at [http://www.opensearch.org](http://www.opensearch.org).

OpenSearch description files are small XML files that describe your search provider to a client, in this case to Internet Explorer. Within an OpenSearch description several bits of information can be found:

- **Search Provider Name**: The name that will be displayed in the search box when your provider is selected.
- **Search URL**: This is the URL where searches for your search provider are sent to.
- **Icon**: A favicon file for your search provider. This icon is displayed next to your provider name in the search box.

All OpenSearch description files must include the search-provider name and search URL. Without these elements, Internet Explorer cannot install your search provider. We also highly recommend that you include an icon within the OpenSearch description file, since if Internet Explorer can’t find an icon for your provider either in your description file or at the root of your website, it will use a generic icon to represent your search provider in the Internet Explorer search box drop-down menu. The advantage of using this install method is that you don’t have to change the layout of your page at all to support this. The disadvantage, however, is that not all users will discover that you have a search provider available as not everyone will notice the change in color in the search box or know to click on it.

To add an embedded provider to your website, you must include a special link element within the head of your Web page.

Here is a sample OpenSearch Description file which contains the search providers name, URL, and icon:

```xml
<?xml version="1.0" encoding="UTF-8"?>
<OpenSearchDescription xmlns="http://a9.com/-/spec/opensearch/1.1/">
  <ShortName>My Search Provider</ShortName>
  <Url type="text/html" template="http://example.com/?q={searchTerms}&source=IE"/>
  <Image height="16" width="16" type="image/icon" href="/websearch.ico"/>
</OpenSearchDescription>
```

The ShortName element contains the search provider name that is displayed in the Internet Explorer search box. The Url element contains the URL where searches will be sent for this search provider. One thing that people sometimes forget to do is to escape the ampersands within their URLs; otherwise your description file breaks. In the example above there is an escaped ampersand. The Image element contains the favicon file for this search provider.

Within the search URL provided for this search provider, you can see the parameter “q={searchTerms}”. When Internet Explorer navigates to this provider to get search results, {searchTerms} will be replaced by the query string that the user typed into the search box. You should make sure that this text is found somewhere within your search URL, otherwise you will not know what your user was looking for.

You can find further details on OpenSearch description files at [http://www.opensearch.org](http://www.opensearch.org).

Installing Your Search Provider

Once you have created your OpenSearch description file, there are two ways that users can install it. Users can install via an embedded link on your website, or with a link or button that you create and promote on your website.

If your page contains an embedded link, when Internet Explorer displays your page, the search box dropdown arrow will light up orange as seen in [Figure 6](#). Users can then install your provider via the Internet Explorer search box drop-down menu. The advantage of using this install method is that you don’t have to change the layout of your page at all to support this. The disadvantage, however, is that not all users will discover that you have a search provider available as not everyone will notice the change in color in the search box or know to click on it.

Why Create an OpenSearch Description File for Your Website?

OpenSearch description files are small XML files that describe exactly how to execute a search on your website. Once you create this file, your users can install your website as one of their Internet Explorer search providers. However, in addition, since OpenSearch is accepted as a standard, Firefox users will also be able to install your search provider with the same file. Not all your users will make heavy use of the search box but for those that do, why not make it easy to include your website as one of their providers?
The rel attribute of the link must have the value “search” and the type attribute must have the value “application/opensearchdescription+xml”. The value of the href attribute is an absolute or relative URL that specifies the location of the OpenSearch description file. The title attribute is displayed in the search box dropdown menu to let people know what search provider they are installing. The title should match the title of your search provider in the OpenSearch description file otherwise the search box will always light up orange when your customers visit your website, regardless of whether they have your search provider installed or not.

Alternatively, instead of relying on a hidden link for the user to install your search provider, you could also create a link or button from which users can install your search provider. The Windows Search Guide (http://www.microsoft.com/windows/ie/search-guide/en-en/default.mspx) is an example of a page that uses links to allow the user to install search providers.

You can use the JavaScript method AddSearchProvider to create a button or link from which your search provider can be installed. For example, here is a button which adds a search provider:

```html
<INPUT TYPE="button" VALUE="Add Search Provider" onClick='window.external.AddSearchProvider( "http://www.example.com/Provider.xml");'>
```

When users click on your link or button, they will see a dialog that will let them install the search provider. If the user chooses to, they can make your search provider the default provider by selecting a check box on the install dialog.

Creating Search Suggestions

I told you previously that Internet Explorer 8 Beta 2 now supports search suggestions in the search box. As a user types in the search box, Internet Explorer can send a request to your search provider with the current query string and you can supply suggestions which might enable the user to create a better search term. Internet Explorer 8 Beta 2 supports two formats of search suggestions—a JSON format and an XML format. The JSON search-suggestion format lets you send text-based search suggestions, which can be augmented with descriptions and URLs. The XML search-suggestion format lets you send all the content available with JSON suggestions, as well as visual suggestions and section titles.

JSON search suggestions are returned as a JavaScript array of arrays. The array contains four values:
- Query String (required). The search term that was requested. If the browser requested suggestions for “seattle” then the query string is “seattle”. Returning the query string allows the client that requested the suggestion to check that the results returned are for the correct query.
- Completions (required). These are the suggestions for the given query string.
- Descriptions (optional). Alternate strings that provide additional information or context about a given completion.
- Query URLs (optional). A list of URLs that should be used if the user selects a given suggestion.

The following is an example JSON response.

```
["seattle",
 ["seattle weather", "seattle seahawks", "seattle mariners"],
 ["rainy all the time", "football team", "baseball team"],
```

This JSON response will produce the search box dropdown seen in Figure 7. In the JSON response, the first element in the first array, plus the first element in the second array, plus the first element in the third array, create a single search suggestion.

Descriptions and query URLs are optional elements in the JSON response. If you do not wish to include these, you can simply leave out the arrays entirely as seen in the example below.

```
["seattle",
 ["seattle weather", "seattle seahawks", "seattle mariners"]
```

The JSON suggestion format does not let you specify the title that you want your suggestions section to have; if you are using JSON suggestions, Internet Explorer will title the section with the name of your search provider plus the word “Suggestions.” If your search provider was titled “Joe’s Search” then your suggestions would be titled “Joe’s Search Suggestions”.

XML search suggestions support all of the same features as JSON search suggestions, along with a few
additional features that are not available in JSON. These additional features include the ability to:

- Name the title of the suggestions section.
- Split suggestions across multiple sections.
- Include visual suggestions.

The Basic XML suggestion looks like the following:

```xml
<?xml version="1.0"?>
<SearchSuggestion>
  <Query>seattle</Query>
  <Section>
    <Item>
      <Text>seattle weather</Text>
      <Description>rainy all the time</Description>
      <Url>http://www.weather.com/seattle</Url>
    </Item>
    <Item>
      <Text>seattle seahawks</Text>
      <Description>football team</Description>
      <Url>http://www.nfl.com/seahawks</Url>
    </Item>
    <Item>
      <Text>seattle mariners</Text>
      <Description>baseball team</Description>
      <Url>http://www.mlb.com/mariners</Url>
    </Item>
  </Section>
</SearchSuggestion>
```

This XML response is equivalent to the previous JSON response and will produce exactly the same output including the same section name, as shown in Figure 7. If you do not wish to include the descriptions or query URLs with your suggestions, you simply remove those elements from the items.

If you do not specify a title for your suggestions section, Internet Explorer will title your suggestions with the name of your search provider followed by “Suggestions”; this is the same as how JSON suggestions are titled. If you wish to specify the title of your suggestions, you can add a title attribute to the <section> tag, as in the following example.

```xml
<?xml version="1.0"?>
<SearchSuggestion>
  <Query>seattle</Query>
  <Section title="My Custom Suggestions">
    ...
  </Section>
</SearchSuggestion>
```

To create multiple sections, you can insert a <separator> tag into your XML. You must include at least one item before and after your separator. If you include a title attribute in the tag, then the separator will be titled; if none is included, the separator will just be a line drawn across the dropdown.

If you wish to add images to your suggestions, you just need to include an <image> tag within an item, as shown in the following item.

```xml
<Item>
  <Text>seattle mariners</Text>
  <Image source="http://www.search.com/mariners.jpg" alt="Seattle Mariners" width="69" height="70" />
</Item>
```

By default, if you include text with an image, the text will be vertically centered next to the image. However, you can change the text alignment by an align attribute on the image element. By setting the align attribute value to “top”, “middle”, or “bottom”, you can specify the text alignment.

You can create items with text and an image, or you can create items which only display an image. If you do not include text you must include a URL so that Internet Explorer knows where to navigate a user to when they click on that suggestion.

It is highly recommended that you include the height and width attributes of your image when sending a visual suggestion. However, if you do not include these values, Internet Explorer will allocate a default amount of space to the image and scale the image to fit within that space. The default amount of space is the vertical equivalent of one line of text.

As I alluded to previously, if you support search suggestions in either JSON or XML format, you need to specify your suggestions URL in your OpenSearch description file. You can do this by adding additional URL elements to the file. If you are supporting JSON suggestions then add a URL of type “application/x-suggestions+json”. If you are supporting XML suggestions then add a URL of type “application/x-suggestions+xml”. These URLs should be added after your main search URL to maintain compatibility with Internet Explorer 7. Use the same {searchTerms} parameter to indicate where in the URL to include the query string.

**Call to Action**

I hope you’ve been intrigued by what you’ve read here today. If you have not yet done so, install Internet Explorer 8 Beta 2 and check out some of the search features that you’ve read about here. I hope that they will make a difference in the way that you search and enable you to search more efficiently. If you own a website and don’t already advertise an OpenSearch description of your site, add one and then consider if search suggestions would be appropriate for your site as well. Happy searching!

*Sharon Cohen*
Developer Tools in Internet Explorer 8 Beta 2

Great tools play a critical role in developer productivity, and while many web development tools exist, they likely don’t meet your needs in all tasks. For example, you may want to quickly debug JavaScript, investigate a behavior specific to Internet Explorer, or iterate rapidly to prototype a new design or try solutions to a problem.

In order to improve developer productivity in these and similar scenarios, Internet Explorer 8 Beta 2 includes powerful, yet easy-to-use tools that have a few important characteristics.

Integrated and Simple to Use

Every installation of Internet Explorer 8 Beta 2 includes the Developer Tools. This enables debugging anywhere you find Internet Explorer 8 Beta 2, whether it’s your dev machine, test machine, or a client’s machine. In addition, by avoiding the use of an extension we limit the impact of the tools on browser performance to only the times when you open the tools. Internet Explorer 8 Beta 2 also includes on-the-fly script debugging so you can enable debugging as needed for only the current Internet Explorer process rather than enabling debugging for all of Internet Explorer and accepting degraded performance and regular script error dialog boxes.

Optimize Application Performance

Identifying and fixing performance issues is usually an iterative approach done by focusing on one scenario at a time. With the Internet Explorer 8 Developer Tools script profiler, you can collect statistics like execution time and number of times a JavaScript function was called as you test your application and use the profile report to quickly identify and fix performance bottlenecks.

With these characteristics, the features of the Developer Tools will dramatically improve your productivity when developing in Internet Explorer. The rest of this article explains many of these features in more detail. For complete documentation of all features, including keyboard shortcuts, visit the Internet Explorer Developer Center on MSDN at http://msdn.microsoft.com/ie.

Getting Started

Getting started with the tools is simple: press F12 or click Developer Tools on the Tools menu of either the menu bar or the command bar. You can also add the Developer Tools icon to the command bar for quick access by right-clicking on the command bar, selecting Customize, and then Add or Remove Commands... Find the Developer Tools icon in the list on the left and click Add. Click Close and the new button will appear in the command bar.

Once open, the tools exist in their own window, each one connected to a single tab in Internet Explorer. If you prefer to decrease the number of open windows, pin the tools within a tab by clicking the “Pin” button or pressing CTRL+P.

Some features of the tools don’t need the complete tools interface. In that case, click the Minimize button or press CTRL+M when the tools are pinned. The tools become a row at the bottom of the win-

Attention Keyboard Users!
The Internet Explorer 8 Developer Tools have extensive keyboard support to make it easier to accomplish tasks. Use standard Windows conventions like F2 to edit and F6 to navigate tab groups, and check out the Internet Explorer 8 Developer Tools Quick Reference at http://go.microsoft.com/fwlink/?LinkID=123648 for a full list of keyboard shortcuts.

Every installation of Internet Explorer 8 Beta 2 includes the Developer Tools.
dow, providing access to all menu items and the document and browser modes without filling all the screen space used by the full user interface.

**Inspect and Edit HTML and CSS**

The tools give you visibility into the browser so you can inspect your site’s HTML and CSS as it exists inside Internet Explorer instead of just the source. This is particularly helpful on dynamic sites, complex sites, and sites that use frameworks like ASP or PHP.

The primary view in the tools displays the site’s Document Object Model (DOM) tree, which reflects the one maintained by Internet Explorer to represent the site in memory. You can navigate the tree with the mouse or the keyboard, but a quick way to find the node for a specific element on the page is turning on “Select Element by Click” and clicking the desired element on the page. The tools will automatically select the corresponding tree node.

If a site uses script to modify the tree, as many do, you’ll want to view the latest tree so click the Refresh button in the tools to acquire the latest tree from Internet Explorer.

Once you’ve selected an element in the DOM tree, the right-hand pane exposes more information about the element.

When debugging the positioning of an element, use the Layout pane to find box model information such as the element’s offset, height, and padding.

The Style pane improves CSS debugging by providing a list of all rules that apply to the selected element. The rules display in precedence order so those that apply last appear at the bottom and any property overridden by another is struck through, allowing you to quickly understand how CSS affects the current element without manually matching selectors.

If you’re looking for information about a specific property, switch to Trace Styles. This pane contains the same information as Styles except it groups it by property, as shown in Figure 1. Simply find the property that interests you, click the plus, and view a list of all rules that set that property, again in precedence order.

---

**Figure 1:** Trace Styles helps you find a specific style property.
We make it look so easy you’d swear
we were cheating. Milos.

The Milos® Solution Platform is advanced component technology used by EPS Software to develop custom applications for clients.

To find out how your project can benefit from the Milos Solution Platform, visit us online at:

www.eps-cs.com/milos
To further inspect CSS, switch to the CSS tab for a list of all CSS files for the current site, as shown in Figure 2.

**Live Editing**

After using the tools to gather important information about your site, you need to be able to take action quickly. Instead of editing your source, saving, and refreshing, edit within the tools by clicking on HTML attributes and values and CSS rules and properties to make them editable. Just click, type the new value, and press ENTER. The change takes effect immediately so you can quickly test changes. In HTML, you can also press the Edit button to make the entire tree editable and then add, remove, or edit complete elements. In either case, the Developer Tools enable quick editing without modifying your source.

**Save Changes**

All changes made in the tools only exist in Internet Explorer’s internal representation of the site. Therefore, refreshing the page or navigating away brings back the original site. However, in some cases you may want to save the changes. In the HTML and CSS tabs, click the Save button to save the current HTML or CSS, respectively, to a file. Remember that not only will the areas of your site that you modified differ from your source, but other parts may also differ since the tools display your site as it exists in Internet Explorer rather than in your source. To prevent accidentally overwriting source, the tools save the output as text and add a leading comment to the file.

**Debug JavaScript**

While CSS makes it easier to build dynamic sites without JavaScript, most complex sites still need at least some script. And as with any programming language, a good debugger is critical to being productive and the Developer Tools (Figure 3) provide just that.

**Simple to Begin Debugging**

Debugging JavaScript in Internet Explorer 8 Beta 2 is simple. On any site you want to debug, open the Script tab and click Start Debugging. That’s all there is to it; there is no need to launch a separate application or attach to a process. In previous versions of Internet Explorer, you had to enable script debugging for all of Internet Explorer by changing a setting in the Internet Options Control Panel. However, enabling script debugging reduces browser performance and generates debugging dialogs on any site with a script error, which is frustrating when not debugging. In Internet Explorer 8 Beta 2, Developer Tools includes on-the-fly script debugging, so you can avoid these problems by leaving script debugging disabled and letting the Developer Tools enable debugging for only the current Internet Explorer process. Any new Internet Explorer window will not have debugging enabled until you start debugging in it.

Once started, you have all the functionality you expect from a debugger.

**Execution Control**

You can pause execution at a specified place by setting a breakpoint. Click a line number or right-
click the source and choose Insert Breakpoint. You can set breakpoints within inline functions or event handlers and use conditional breakpoints.

If you’re not sure where to set a breakpoint, click Break All to pause execution before the next JavaScript statement runs or Break On Error to pause execution when Internet Explorer encounters a script error. After pausing execution, step through script, including anonymous functions, with Step Into, Step Over, and Step Out.

**Variable Inspection**

As you step through code, it’s helpful to inspect the current state of variables. Within the Script tab, use the Locals pane to view local variables and the Watches pane to monitor a custom list of variables. You can add variables to this list by right-clicking it in the source and clicking Add Watch or typing the variable name in the Watches pane.

**Navigating Call Stack**

As you debug your application by stepping through the JavaScript code, you can view the functions in the current call stack. Clicking on a function in the call stack window displays the source code in the source view window.

**JavaScript Console**

As with HTML and CSS, the Developer Tools not only give you the information needed to debug your site, but they also make it easy to use that data. Instead of editing your source to test a statement, type the JavaScript in the Console pane to execute it. Click Multi Line Mode to enter large amounts of script. Internet Explorer executes the code immediately so you can test the outcome of adding that code at the same location as the current break. You can use Console even when you are not debugging. To make it easy for you to fix errors on your page, Console also logs all the script errors.

**Eliminate “Alert Debugging” with Console**

The Developer Tools offer a powerful feature to help you debug: console.log. Instead of using window.alert and generating countless dialogs, call console.log to output strings to the Console pane. Add variables to the output and use console.info, console.warn, and console.error to vary the formatting of entries in the Console window. Since the Developer Tools Console functionality is similar to that available in Firebug, a Web developer tool for Firefox, debugging notifications will work across browsers without any changes to your script.

**Profile JavaScript**

Debugging JavaScript helps perfect the behavior of your site, but the JavaScript profiler can help take your site to the next level by improving its performance. The profiler gives you data on the amount of time spent in each of your site’s JavaScript methods and even built-in JavaScript functions like string concatenation. Since you can start and stop the JavaScript profiler at any time during application execution, you can collect profile data for the specific scenario that interests you. Figure 4 shows a screenshot of the profiler.

**Starting a Profiling Session**

As with the debugger, getting started is simple: open the Profiler tab and click Start Profiling to begin a session. Perform the scenario you want to profile and then click Stop Profiling to view the data.

**Viewing Data**

By default, the data appears in a function view which lists all the functions used. Click the Current View

![Figure 4](image-url): Using the Profiler tab in Internet Explorer Developer Tools you can collect information about your JavaScript methods and functions, giving you information you need to improve performance of your JavaScript.
Internet Explorer 8 Beta 2 has the ability to render pages and report version information as Internet Explorer 7.

Find the Performance Hot Spots

Use the Call Tree view in the Internet Explorer 8 Developer Tools Profiler to quickly find the performance hot spot in your application. Open your profile report in the Call Tree view and sort on the Inclusive Time column. Continue expanding the node which takes the highest inclusive time until you find your performance hot spot.

More Information

For more information about the Internet Explorer Developer Tools, check out these resources:

- Internet Explorer Developer Center: [http://msdn.microsoft.com/ie](http://msdn.microsoft.com/ie)

dropdown menu and select Call Tree to view a tree representing the order of calls made so you can walk through the code path and find hot spots.

In both views, you can add, remove, arrange, and sort the columns to find the information you're looking for:

- Right-click a column header and select Add / Remove Columns to customize the view to include exactly the data you're interested in.
- Right-click and select Sort By or simply click a column header to sort the data by that field.
- Drag and drop columns to rearrange them.

The profile report gives you information on the URL and line number of the function to help you locate the code in your application. Click on the function name to bring up the source code for the function in the source view window of the Script tab.

Running Multiple Sessions and Exporting Data

Click Start Profiling again to begin a new profiling session. When you click Stop Profiling, the new data displays. To view the data from previous profiling sessions, click the file dropdown and choose another report. The profile report persists for the current Internet Explorer process, but you can save the profile data in CSV format by clicking Export and use any other application for later investigation.

Test in Different Browser Modes

Internet Explorer 8 Beta 2 has the ability to render pages and report version information as Internet Explorer 7. Web developers and end users can use this ability to ensure sites continue to work even if not built for Internet Explorer 8 Beta 2, but you can also use it to test how your site will look to Internet Explorer 7 users.

Browser Mode

The Browser Mode menu lets you choose how Internet Explorer should report three important properties:

- **User agent string**: The value Internet Explorer sends to Web servers to identify itself.
- **Version vector**: The value used when evaluating conditional comments.
- **Document mode**: The value used to determine whether Internet Explorer uses the most recent behavior for CSS, DOM, and JavaScript operations or emulates a previous version of Internet Explorer for compatibility.

Three Browser Mode options exist, each modifying these values in different ways:

- **Internet Explorer 7**: In this mode, Internet Explorer reports a user agent, version vector, and document mode identical to those used by Internet Explorer 7. Use this mode to test how Internet Explorer 7 users experience your site.
- **Internet Explorer 8**: In this mode, Internet Explorer reports a user agent, version vector, and document mode to match default Internet Explorer 8 Beta 2 behavior which is the most standards-compliant available in Internet Explorer 8 Beta 2. Use this mode if you want to test how Internet Explorer 8 Beta 2 users experience your site.
- **IES Compatibility View**: In this mode, Internet Explorer reports a user agent string indicating Internet Explorer 7, but an additional token appended to the string serves as an indication to web servers that the browser is Internet Explorer 8 Beta 2-capable. The version vector and document mode handling are identical to those used by Internet Explorer 7. Use this mode to test how Internet Explorer 8 Beta 2 users experience your site if they've chosen the Compatibility View option.

Document Mode

The Document Mode defines how Internet Explorer renders your page but has no effect on the version vector or user agent string. By using this option in conjunction with the browser mode, you can quickly test which document mode you should use for your site. Three options exist:

- **Quirks**: This behavior matches that of Internet Explorer when rendering a document with no doctype or a quirks doctype. It's similar to the behavior of Internet Explorer 5 and the quirks mode behavior of Internet Explorer 6, and the same as the quirks mode of Internet Explorer 7.
- **Internet Explorer 7 Standards Mode**: This behavior matches that of Internet Explorer 7 rendering a document with a strict or unknown doctype.
- **Internet Explorer 8 Standards Mode**: This behavior is the latest standards-compliant available in Internet Explorer 8 Beta 2 and is the mode used by default in Internet Explorer 8 Beta 2 when rendering a document with a strict or unknown doctype.

There's More!

The Internet Explorer 8 Developer Tools include many other features to help you develop sites in Internet Explorer. Some let you view important information like element class name or image size on the page, while others help with common tasks like matching a color on the screen or measuring a distance. Experiment with the tools yourself to learn more about these and others. Visit the Internet Explorer Developer Center at [http://msdn.microsoft.com/ie](http://msdn.microsoft.com/ie) for complete documentation. And don't forget to follow the Internet Explorer Blog at [http://blogs.msdn.com/ie/](http://blogs.msdn.com/ie/) to keep up to date on the latest news from the Internet Explorer team!

John Hreatin
Deepak Jain
Sameer Chabungbam
Better AJAX Development with Internet Explorer 8 Beta 2

Internet Explorer 8 Beta 2 provides a programming model for AJAX (Asynchronous JavaScript and XML) development that is simpler yet more powerful than ever before, spanning browser, Web page, and server interaction. You’ll be able to build pages that are faster and more functional, with better end-user experiences. Plus, the APIs Microsoft has added to Internet Explorer 8 Beta 2 are based on the World Wide Web Consortium (W3C) HTML 5.0 or Web Applications Working Group standards.

Inside this article, you’ll find information on all of the AJAX enhancements available in Internet Explorer 8 Beta 2.

These include tools to improve performance and client-side state management: local storage APIs, AJAX navigations, increased connections per host (plus a way to detect the number of connections on a host), and a scriptable connectivity event.

We’ve also worked on enhancing our communication objects and protocols. For example, safer and easier cross-domain communication is now possible through the XDomainRequest object and the W3C’s Access Control draft, as well as HTML 5.0’s cross-document messaging.

For up-to-the minute information on AJAX and other technologies in Internet Explorer, visit the Internet Explorer Developer Center at http://www.msdn.com/ie.

AJAX Navigations

One of the great benefits of implementing AJAX—indeed, one of the main reasons for its existence—is the ability to update page content without navigating to a new page. With this convenience, though, come drawbacks that can confuse users. On an AJAX-heavy page, the Address bar is not updated with each update. Subsequently, the “travelog” or browsing history, isn’t updated either.

As an example, consider a mapping Web site such as Windows Live™ Search Maps or Google Maps. When using the AJAX-enabled features of those applications—such as pan and zoom—neither the Address bar nor the travelog is updated. Users who are used to the Address bar changing with each new page they navigate to or who rely on the browser’s Back and Forward buttons to navigate back and forth a page at a time may find this jarring. While some Web sites work around this limitation by navigating a hidden IFrame when updating content through AJAX, this technique can decrease performance.

To enable AJAX navigations, Internet Explorer 8 Beta 2 in IE8 mode (the default compatibility mode) treats updates to the window.location.hash property like individual, “traditional” navigations. When the hash property is updated, the previous document URL (which may be from the previous hash fragment) is updated in the Address bar and the travelog (and therefore the Back button). At the same time, a new hashChanged event is raised and the hash URL fragment is saved before navigating away from the page.

On AJAX-enabled pages that take advantage of this new functionality, when AJAX content changes, navigation is as seamless as usual, but the user can back up and go forward as if the AJAX navigation was a traditional navigation. The markup sample in Listing 1 demonstrates a use for this new functionality. In this case, the hash property is set when the onendzoom event of the Microsoft® Virtual Earth™ map control occurs. In other words, every time the user zooms in or out, the Address bar and travelog are both updated, enabling the user to navigate back and forth between zoom levels by using the Back and Forward buttons.

Every time the user zooms in or out, the fragment identifier in the Address bar is updated, as shown in Figure 1. On this page, that simply means the integer after the number sign (#) is incremented.

![Figure 1: With each AJAX navigation, the fragment identifier (the integer at the end) is incremented.](http://localhost/ajax/ajax-navigation.html#6)
Better AJAX Development with Internet Explorer 8 Beta 2

The following JavaScript snippet introduces the `DOM Storage` property `localStorage` by using it to create a store and then setting (`setItem` method), getting (`getItem` method), and removing an entry (`removeItem` method), plus clearing the store entirely (`clear` method).

```javascript
// Store a key-value pair.
localStorage.setItem("Sean","Purcell");

// Retrieve value string for a given key.
var storedItem = localStorage.getItem("Sean");

// Remove item from store.
localStorage.removeItem("Sean");

// Clear the store.
localStorage.clear();
```

Now let's see DOM Storage in action. **Listing 2** is a code sample that will present you with a page (shown in **Figure 2**) with a text box and two buttons. Type something in the text box and then click the “Send to Store” button, and then close the window. Navigate back to the page, and what you typed will still be there. That’s the DOM store in action.

**DOM Storage**

To store data on local machines, Web sites today often use the `document.cookie` property. However, cookies are limited in their capabilities; sites can only store fifty key/value pairs per domain, and the cookie programming model requires parsing the entire cookie string for data. DOM Storage objects—specified in the W3C’s HTML 5 Working Draft and implemented in Internet Explorer 8 Beta 2—provide a much simpler and more flexible global and session storage model for structured data on the client side.

Consider this scenario: A user is trying to find the best pair of tickets for a popular concert on an online ticketing site. For that purpose, the user opens several windows to make multiple requests, just to see the best seats she can get. If the site’s application is using cookies to store its session state, information could “leak” from one transaction into the other, potentially causing the user to purchase seats that she didn’t want without noticing. The potential for this sort of information “leak” becomes even more widespread as applications become more capable of offline behaviors, such as storing values locally for later return to the server, and as people start using more tabs to connect to the same Web site in parallel.

DOM Storage offers essential differences from cookies. For one, it offers significantly more available disk space than cookies. In Internet Explorer, cookies can store 4 kilobytes (KB) of data, whereas DOM Storage provides about 10 megabytes (MB) for each storage area. Furthermore, DOM Storage doesn’t transmit values to the server with every request as cookies do, and data in a global store never expires. Unlike cookies, it’s easy to access individual pieces of data using an interface that is supported in Internet Explorer 8 Beta 2 and other browsers, and sites can choose to store data for the life of a tab or until the site or the user clears the data.

Now let’s see DOM Storage in action. **Listing 2** is a code sample that will present you with a page (shown in **Figure 2**) with a text box and two buttons. Type something in the text box and then click the “Send to Store” button. This serializes the string to the DOM store.

To see what happens without the DOM store, remove the `body onload` event to observe the past behavior.

It’s important to note that DOM Storage is just a mechanism for Web applications to simply store data, and that there is no database behind it. For example, you can’t perform complex queries like search by value on the DOM store.

**Connectivity Enhancements**

Internet Explorer 8 Beta 2 introduces several connectivity enhancements, including more accurate connectivity status information and an increase in the default number of concurrent connections to a single server.

Matthew Tevenan
mtevenan@microsoft.com

Matthew Tevenan is a technical writer with the Internet Explorer Developer Content team, and has been with Microsoft for over five years now. He previously worked in the Windows Client SDK and Windows Digital Media Division as a technical editor. Matthew has a Master’s degree in Technical Communication from the University of Washington, and a Bachelor’s degree in French from the University of San Francisco. His previous work experience as a technical communicator includes stints at Dantz Development Corporation (now EMC) and Apple Inc. He spends his free time studying graphic design, playing the cello, and being perturbed by his two bad cats.
Listing 1: AJAX Navigations sample markup

```html
<html>
<head>
<meta content="IE=8" http-equiv="X-UA-Compatible">
<title>AJAX Map</title>
</head>

<!-- Load the Virtual Earth map control. -->
<script src="http://dev.virtualearth.net/mapcontrol/mapcontrol.ashx?v=6" type="text/javascript"></script>
<script type="text/javascript">
var oMap = null;
var iZoomLevel = 0;

function GetMap()
{
    oMap = new VEMap('myMap');
oMap.LoadMap();
oMap.AttachEvent("onendzoom", ZoomHandler);
iZoomLevel = oMap.GetZoomLevel();
window.location.hash = iZoomLevel;
}

function ZoomHandler(e)
{
iZoomLevel = oMap.GetZoomLevel();
window.location.hash = iZoomLevel;
}

function HashChangeHandler()
{
var hash = window.location.hash;
var iNewZoomLevel = hash.substr(1);
if (iNewZoomLevel != iZoomLevel)
{
iZoomLevel = iNewZoomLevel;
oMap.SetZoomLevel(iNewZoomLevel);
}
}
</script>

<!-- Attaching the event handler to a new onhashchange event allows
the page to detect when the hash has changed and an AJAX
navigation has occurred. -->
<body onhashchange="HashChangeHandler();" onload="GetMap();"style="overflow:scroll;height:100%">
<div id="myMap" style="position:relative;width:500px;height:500px;vertical-align:middle"></div>
</body>
</html>
```

Listing 2: DOM Storage sample markup

```html
<html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">
<head>
<meta content="IE=8" http-equiv="X-UA-Compatible"></head>
<title>HTML 5.0 DOM Store</title>
<script type="text/javascript">
var key1 = 'defaultkey';

function getValue()
{
    var storeditem = localStorage.getItem(key1);
    if(storeditem != null)
    {
        document.getElementById("textinput").value = storeditem;
    }
}

function setValue()
{
    var itemindex = document.getElementById('textinput').value;
alert(itemindex);
    localStorage.setItem(key1,itemindex);
}

function clearText()
{
    document.getElementById('textinput').value = "";
}

</script>

<body onload="getValue();">
<h1>Type into the text area and retrieve your text from the local store in Internet Explorer 8.</h1>
<div>
Enter Text:
<input id="textinput" onclick="clear_text();" size="60"
type="text" value="Your message here">
<input onclick="setValue();" type="submit" value="Send to Store" />
</div>
<button onclick="clearItems();">Clear Stored selection</button>
</body>
</html>
```
ers initiated by choosing Work Offline from the File menu. This was not useful in determining whether the user truly had an active connection. In Internet Explorer 8 Beta 2, however, the onLine property indicates whether the system is actually connected to the network.

Internet Explorer 8 Beta 2 also introduces new simple events. When the window.navigator.onLine property changes after a page is loaded, a simple event is raised on the page's body element as follows.

- If onLine changes from true to false, the offline simple event is raised on the body element.
- If onLine changes from false to true, the online simple event is raised on the body element.

Finally, Internet Explorer 8 Beta 2 introduces new callback handlers to indicate a change in the status of a connection. The onoffline handler is raised when the user's computer loses its network connection—for example, when the network cable is unplugged or the network adapter is disabled. The ononline handler is raised when the computer's network connection is restored. Of course, these handlers are also raised when the user toggles the Work Offline setting on or off.

Let's see these new connectivity status enhancements at work. Listing 3 is a sample that creates a page (shown in Figure 4) displaying the current connectivity status. It will automatically update as your network connection is enabled and disabled.

More Connections per Server

If you're an Internet Explorer and AJAX expert, you may already know that with versions of Internet Explorer prior to Internet Explorer 8, you were limited to two concurrent connections to a single HTTP 1.1 server. Things were a bit better with HTTP 1.0 servers, as the limit was 4, but HTTP 1.1 connections are far more common today.

This two-connection limit was due to a mandate in the Internet Engineering Task Force’s RFC 2616. In 1999, when the spec was drafted, this limit was appropriate, considering the dominance of dial-up Internet connections, the uncommonness of script-heavy Web applications, and the relative scarcity of broadband connections. Today, however, broadband is much more commonplace. The need for more per-host connections is especially strong considering the increase in popularity of AJAX applications and pages, many of which send large amounts of data asynchronously. That and other improvements to general Web infrastructure have led to an increase in the maximum number of concurrent connections in Internet Explorer 8.

In Internet Explorer 8 Beta 2, the maximum number of concurrent connections from a single host process connecting via broadband to a single server has been increased from 2 to 6.
over dial-up (with a modem over a telephone line) to a single server remains the same as for Internet Explorer 7 and earlier.

If you've implemented workarounds to circumvent the previous connection limitations, you may have compatibility concerns with this new behavior. In that case, and to avoid overwhelming your Web server, you may want to tailor content delivery based on the number of connections each client computer is capable of having open concurrently. For that reason, Internet Explorer 8 Beta 2 includes two new read-only properties of the window object that enable your server to determine the number of available connections on the client computer. Table 1 lists the new properties and their return values.

<table>
<thead>
<tr>
<th>Server Type</th>
<th>Scriptable Property</th>
<th>Return Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>HTTP 1.1</td>
<td>window.maxConnectionsPerServer</td>
<td>MaxConnectionsPerServer</td>
</tr>
<tr>
<td>HTTP 1.0</td>
<td>window.maxConnectionsPer1_0Server</td>
<td>MaxConnectionsPer1_0Server</td>
</tr>
</tbody>
</table>

Table 1: New scriptable properties for determining the number of connections on the client computer.

Depending on the value the property returns, you can choose to parallelize downloads or to change the priority of content delivery.

The sample in Listing 3 shows you these new properties in action. In Internet Explorer 8 Beta 2, this page will display the number of maximum connections available to your computer. If you are connected via broadband to the Internet, it will display “6” for each value. If you are connected via dial-up, the values will be “2” for HTTP 1.1 servers and “4” for HTTP 1.0 servers.

**XMLHttpRequest Enhancements**

The XMLHttpRequest object was invented nearly a decade ago, primarily to enable Microsoft Outlook® Web Access to display e-mails without requiring a page refresh. Since its creation, however, its adoption has scaled up beyond the scope of its original design, while its functional design has not changed. Your advanced AJAX application may need finer control over AJAX requests than has typically been available.

Listing 3: Connectivity Status enhancements sample markup

```
<DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
 * "http://www.w3.org/TR/html4/loose.dtd">
<html>
<head>
<title>Connection Events Manual Test Page</title>
<meta content="text/html; charset=UTF-8" http-equiv="Content-Type">
<script type="text/javascript">
  function onli()
  {
    connectivity.innerText = "Online";
    connectivity.style.color = "red";
  }
  function onof()
  {
    connectivity.innerText = "Offline";
    connectivity.style.color = "blue";
  }
  function Start()
  {
// Check whether currently connected to a network.
    if (window.navigator.onLine)
    {
      onli();
    }
    else
    {
      onof();
    }
</script>
</head>
<body onload="Start();" onoffline="onof();" ononline="onli();"></body>
</html>
```

Listing 3: Connectivity Status enhancements sample markup

XMLHttpRequest timeout support enables you to preset a period of time after which the request will abort and make a new connection available.
since the timeout is set to 0.02 seconds, you will quickly see a “timed out” alert. If you want to cycle through more alerts, increase the timeout value.

**Cross-domain Request ("XDR")**

Web browsers have a security policy called the same site origin policy, which blocks Web pages from accessing data from another domain. Web sites often work around this policy by having their server request content from another site’s server in the backend, thus circumventing the check within the browser. Figure 4 illustrates this process for a typical mashup site.

In Internet Explorer 8 Beta 2, Web pages can simply make a cross-domain data request within the browser using the new XDomainRequest object, and without server-to-server requests. Figure 5 illustrates a cross-domain request (XDR). 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. This behavior is part of the W3C’s Web Application Working Group’s draft framework on client-side cross-domain communication that the XDomainRequest object integrates with.

It’s important to note that to protect user data, cross-domain requests are anonymous, which means that servers cannot easily find out who is requesting data. As a result, you only want to request and respond with cross-domain data that is not sensitive or personally identifiable.

The following JavaScript snippet introduces the XDomainRequest object and its events:

```javascript
// Creates a new XDR object.
xdr = new XDomainRequest();

// Indicates there is an error and the request cannot be completed.
xdr.onerror = alert_error;

// The request has reached its timeout.
xdr.ontimeout = alert_timeout;

// The object has started returning data.
xdr.onprogress = alert_progress;

// The object is complete.
xdr.onload = alert_loaded;
```

This snippet introduces the XDomainRequest properties and methods.

Since the timeout is set to 0.02 seconds, you will quickly see a “timed out” alert. If you want to cycle through more alerts, increase the timeout value.

**For More Information**

Get the latest information on AJAX and other Internet Explorer technologies in the Internet Explorer Developer Center, at [http://www.msdn.com/ie](http://www.msdn.com/ie).
As mentioned previously, XDR has two components: a client side that makes a request for data to a URL across domains, and a server side that responds with the Access-Control-Allow-Origin: * header and the data, which Internet Explorer then makes available to the requesting domain after performing security checks.

The code sample in Listing 6 demonstrates the client-side component of XDR. The sample refers to a fictional server, so you should feel free to use your local machine or a sandbox server to try this out.

Listing 4: Number of maximum connections status sample markup

```html
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
  "http://www.w3.org/TR/html4/loose.dtd">
<html>
<head>
<meta content="text/html; charset=UTF-8" http-equiv="Content-Type">
<title>More Connections Per Host</title>
<script type="text/javascript">
function Start()
{
  // Get the value of the maximum number of connections per
  // HTTP 1.1 server.
  http_1_1.innerText = window.maxConnectionsPerServer;
  // Get the value of the maximum number of connections per HTTP
  // 1.0 server.
  http_1_0.innerText = window.maxConnectionsPer1_0Server;
}
</script>
</head>
<body onload="Start();">
<h3>Taking Advantage of Six Connections Per Host on Broadband</h3>
<div style="border: double 5px black; margin: 4px 4px 4px 4px;
font-size: 40pt;">
This is for HTTP 1.1 servers: 
<s id="http_1_1" style="color: red;"></s>
This is for HTTP 1.0 servers: 
<s id="http_1_0" style="color: blue;"></s>
</div>
</body>
</html>
```

Listing 5: New XMLHttpRequest timeout sample markup

```html
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01//EN"
  "http://www.w3.org/TR/html4/strict.dtd">
<html>
<head>
<meta content="text/html; charset=UTF-8" http-equiv="Content-Type">
<title>XMLHttpRequest Enhancements</title>
</head>
<body onload="run_test();">
<h1>Test starts onload</h1>
<script type="text/javascript">
var xhr;

function run_test()
{
  // Create a new XMLHttpRequest object.
  xhr = new XMLHttpRequest();
  xhr.onreadystatechange = alert_readystate;
  xhr.ontimeout = alert_timeout;

  // The URL is preset in the text area.
  // This is passed in the open call with a GET
  // request.
  xhr.open("GET", "http://www.fourthcoffee.com/index.php", true);
  // The timeout is set to .02 seconds.
  // You must set the timeout after open.
  xhr.timeout = 20;

  // The request is sent to the server.
  xhr.send(null);
}

function alert_readystate()
{
  alert(xhr.readyState);
}

function alert_timeout()
{
  alert("timed out");
}
</script>
</body>
</html>
```
Listing 6: Cross-domain request sample markup

```html
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN" "http://www.w3.org/TR/html4/loose.dtd">
<html><head>
<meta content="text/html; charset=utf-8" http-equiv="Content-Type">
<title>Internet Explorer 8 - XDomainRequest</title>
<script type="text/javascript">
var xdr;

function read_data()
{
    var output = document.getElementById('text_response');
    if(output)
    {
        // To view the responseText on the page, click the Read button.
        output.innerText = xdr.responseText;
    }
    // The Read button also displays the content type and length of response in alerts.
    alert("Content-Type: " + xdr.contentType);
    alert("Length: " + xdr.responseText.length);
}

function alert_error()
{
    alert("XDR onerror");
}

function alert_timeout()
{
    alert("XDR ontimeout");
}

function alert_loaded()
{
    alert("XDR onload");
    alert("Got: " + xdr.responseText);
}

function alert_progress()
{
    alert("XDR onprogress");
    alert("Got: " + xdr.responseText);
}

function req_abort()
{
    if(xhr)
    {
        xdr.abort(); // Abort XDR if the Stop button is pressed.
    }
}

function req_init()
{
    var url = document.getElementById('input_url').value;
    var timeout = document.getElementById('input_timeout').value;
    if (window.XDomainRequest) // Check whether the browser supports XDR.
    {
        xdr = new XDomainRequest(); // Create a new XDR object.
        if (xhr)
        {
            // There is an error and the request cannot be completed.
            // For example, the network is not available.
            xdr.onerror = alert_error;
            xdr.ontimeout = alert_timeout;
            // This event is raised when the request reaches its timeout.
            xdr.onload = alert_loaded;
            xdr.timeout = timeout;
            // The URL is preset in the text area. This is passed in the open call with a get request.
            xdr.open("get", url);
            // The request is then sent to the server.
            xdr.send();
            else
            {
                alert('Failed to create new XDR object.');
                };
                else
                {
                    alert('XDR does not exist.');
                }
        }
    }
}
</script>
</head>
<body>
<h1>XDomainRequest</h1>
<form action="">
<!-- Assign URL and timeout values from their text boxes to variables. -->
<input id="input_url" type="text" value="http://www.fourthcoffee.com/thedata.txt" />
<br>
<input id="input_timeout" type="text" value="10000" />
<br>
<input onclick="req_init()" type="button" value="Get" />
<input onclick="req_abort()" type="button" value="Abort" />
<input onclick="read_data()" type="button" value="Read" />
</form>
<div id="text_response">
</div>
</body>
</html>
```
Listing 7: Cross-document messaging sample origin page markup

```html
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
 "http://www.w3.org/TR/html4/loose.dtd">
<html>
<head>
<title>Internet Explorer 8 - Cross Document Messaging</title>
<meta content="text/html; charset=UTF-8" http-equiv="Content-Type">
<script type="text/javascript">
// This function runs when you click "Send to Frame" and
// posts content to the IFrame using Cross-document Messaging.
function post_to_frame()
{
    var text_sent = document.getElementById('finput').value;
    if (text_sent != '' && text_sent !=
        'Type text you want to send here')
    {
        var target_iframe =
            document.getElementById('remote_iframe');
        // Post message to secure receiver document. The
        // message will only be sent if the target IFrame has
        // the same protocol and host as the specified target
        target_iframe.contentWindow.postMessage(text_sent,
            "http://www.fourthcoffee.com");
    }
}
function clear_text()
{
    document.getElementById('finput').value = '';
}
</script>
<style type="text/css">
#remote_iframe {
    width: 800px;
    height: 300px;
}
</style>
</head>
<body>
<h1>Post a Message to an IFrame using HTML 5.0 Cross-document Messaging</h1>
<div>
    Enter Content:
    <input id="finput" onclick="clear_text();" size="32"
        type="text" value="Type text you want to send here">
    <input onclick="post_to_frame();" type="submit" value="Send To Frame">
</div>
<iframe id="remote_iframe" src="Test_IFrame.htm"></iframe>
</body>
</html>
```

Listing 8: Cross-document messaging sample receiving page markup.

```html
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
 "http://www.w3.org/TR/html4/loose.dtd">
<html>
<head>
<title>XDM Receiver Frame</title>
<meta content="no-cache" http-equiv="pragma">
<script type="text/javascript">
// Runs the receiver function when the onmessage event is
// raised (a message is received).
if (window.attachEvent)
{
    window.attachEvent("onmessage", receiver); // IE
}
else
{
    window.addEventListener("message", receiver, false);
    // Other browsers
}
// The input to the receiver function is the message.
function receiver(e)
{
// Check to make sure message is not blank
// and origin is expected (http://www.contoso.com).
if (e.data != '' && e.origin == "http://www.contoso.com")
{
    // Assign values to spans for display within the frame.
    document.getElementById('fdomain').innerHTML =
        e.origin;
    document.getElementById('fdata').innerHTML = e.data;
    document.getElementById('fcurrentpage').innerHTML =
        window.location.href;
}
</script>
</head>
<body>
<div>
    Remote Message Receiver<br>
    IFrame HREF: <span id="fcurrentpage"></span><br>
</div>
<br>
<div>Incoming Message</div>
<br>
<span>Received From: </span><span id="fdomain"></span><br>
<span>Received Data: </span><span id="fdata"></span><br>
</body>
</html>
```
On the server side, the browser will request data from it by sending an Access-Control-Allow-Origin: * header. It will only complete the connection if the server responds with an Access-Control-Allow-Origin with a value of *.

**Cross-document Messaging (“XDM”)**

As you discovered in the section of this article on XDR, the browser’s same site origin policy blocks Web pages from getting data from other domains. This means that a richer experience is prevented since different domains on a single Web page can’t communicate with each other. Web sites work around this policy by creating nested IFrames and retrieving data transmitted through URLs. Another way Web sites work around this policy is by directly hosting script and other resource files from other domains. This second workaround only allows one-way communication. It is also a security risk because embedded script and resources run with the same privileges as the host Web site and have access to the user’s data like that stored in cookies.

Cross-document messaging (“XDM”) provides a postMessage() method off of the document object. The postMessage method, along with the onmessage event, allows different domains to communicate with each other, given mutual consent. XDM provides a much simpler, more efficient mechanism for two-way cross-domain communication than the workarounds mentioned previously. The following JavaScript snippet introduces both the postMessage method (used in the sender) and the onmessage event (used in the receiver):

```javascript
// Posts the message data (text_sent) to the
// target URI (*http://www.fourthcoffee.com*).
target iframe.contentWindow.postMessage(text_sent, *http://www.fourthcoffee.com*);

// Runs myFunction once a message is received.
window.addEventListener("onmessage", myFunction);
```

Take a look at XDM in action to better understand how it works. **Listing 7** is a page that sends a message to an IFrame (Listing 8) within the page. In our example, the page in Listing 7 resides on the domain contoso.com and the IFrame in Listing 8 resides on the domain fourthcoffee.com.

You can try this out on your own locally. Just replace both http://www.fourthcoffee.com and http://www.contoso.com with http://localhost (of course, you’ll need to enable IIS (Internet Information Services) first) and place both files in the inetpub folder on your hard drive. What you’ll see looks a lot like Figure 6.

**Conclusion**

We hope you’ve enjoyed learning about the ways you can develop AJAX applications in a way that’s better and smarter than ever before. If you haven’t downloaded Internet Explorer 8 Beta 2 yet, now’s a great time to do so, and try out the new functionality!

Sunava Dutta
Matthew Tveenan
Measuring AJAX Performance in Internet Explorer 8 Beta 2

Growing popularity of AJAX has led to increased use of JavaScript in creating dynamic and rich browser experiences. JavaScript has become the lingua franca of the Web and its performance has become an emerging differentiator in how the industry compares browsers. These attempts have so far been centered on the use of JavaScript micro-benchmarks. While this is a good first step, it is not sufficient in measuring end-user perceivable performance of AJAX applications. We used a more holistic methodology for measuring AJAX performance in Internet Explorer 8 Beta 2.

The AJAX Subsystems

In AJAX applications, data is retrieved asynchronously using the XMLHttpRequest object. An AJAX scenario can be visualized as an activity passing through a pipeline of logical subsystems within the browser stack. In order to understand the performance characteristics of AJAX applications, we need to understand what each of these subsystems do and how they interact with each other. Figure 1 shows the AJAX subsystems for Internet Explorer 8 Beta 2:

- **Network**: Whenever a user types in a URL to load a webpage, the browser communicates with the server over the network, and waits for a response from the server. The network is also responsible for asynchronous data exchange between the Web client and the server.
- **Parsers**: When data is received from the server, it reads, analyzes, and converts the data (HTML, CSS, XML, etc.) into their native object model formats.
- **Layout**: Internet Explorer's layout subsystem takes input from the parsers and computes the layout of the various components, which form the Web page.
- **Rendering**: Internet Explorer's rendering engine does the final painting of the page (and any subsequent updates that are required).
- **Native OM (or DOM)**: The DOM is the object representation of the website's HTML and CSS content. The DOM also acts as a layer for communication between different browser components.
- **JScript Engine**: The JScript engine represents Microsoft's implementation of the ECMAScript language based on ECMA-262 3rd Edition standard. It contains the basic primitives (functions, objects, types, etc.) for performing various language operations.

Interpreting Micro-benchmarks

There are several micro-benchmarks (for instance, SunSpider, Celtic Kane, RockStarApps, ZIMBRA) cited to compare browser performance on AJAX applications today. These micro-benchmarks typically measure two things:

1. Pure JavaScript language performance
2. DOM/IE performance (speed of performing typical DOM operations from JavaScript)
Micro-benchmarks are simple to create, easy to run and provide a quick way for developers to run regressions before making check-ins. But, the use of micro-benchmarks has limitations:

1. They consist of simple operations that run several thousand iterations, which is generally not representative of real world applications.
2. They can be written to exaggerate a particular behavior in a browser and browsers can be tuned to run certain micro-benchmarks very well.
3. Micro-benchmarks insert timers in the JavaScript code to report the elapsed time. Using timers in user level JavaScript code is not the most accurate way to measure CPU time.

Microsoft's goal in Internet Explorer 8 Beta 2 was to improve end-user perceivable performance so we were careful not to use micro-benchmarks as a singular metric to focus our engineering effort. Instead, we took a more balanced approach and used real world code in addition to micro-benchmarks to drive end-user visible AJAX performance improvement.

Measurement Methodology

We use three different measurements to represent a more holistic view of AJAX performance:

1. AJAX subsystem measurements
2. Real World Code (RWC) measurements
3. Micro-benchmark measurements

Figure 2: CPU times per site/per AJAX subsystem (June 10 2008 Internet Explorer 8 build).
Measuring real-world code is challenging in many ways. The scenarios are hard to automate and replicate results consistently. Isolating a scenario is difficult and it is very easy to get “noise” in the measurements that can lead to misleading performance data. Investment needs to be made to add the right level of instrumentation to get consistent and accurate measurements. Working closely with large AJAX customers has been our way to solve some of these challenges and we continue to develop and hone this process.

### Micro-benchmarks

Several micro-benchmarks have emerged (ZIMBRA, SunSpider, Celtic Kane and RockStarApps) and SunSpider seems to have become the de facto JavaScript micro-benchmark. In our tests, SunSpider benchmark shows pure JScript performance improvement of over 400% compared to Internet Explorer 7.

### Conclusion

AJAX performance measurement is a complex problem and we adopted a structured approach in Internet Explorer 8 Beta 2 to drive targeted performance improvements. Be sure to read the follow-up article in this issue, “Performance Improvements in Internet Explorer 8” that describes the actual changes.

---

**Micro-benchmarks Links**

The micro-benchmarks referenced in this article are listed below:

- ZIMBRA: http://www.zimbra.com/
- SunSpider: http://webkit.org/perf/sunspider-0.9/sunspider.html
- RockStarApps: http://www.rockstarapps.com/samples/performance/

---

**AJAX Subsystem Measurements**

We measured AJAX subsystems time profile using the ETW (Event Tracing for Windows) infrastructure in Internet Explorer 8 Beta 2. Using ETW events that mark off AJAX subsystems, we can accurately measure time spent within each AJAX subsystem, including CPU and elapsed time. This data can be collected while running any scenario. A select set of thirty popular websites were run to collect this data. The process in each case is to clear the cache, launch Internet Explorer 8 Beta 2 with a blank page, navigate to the site, wait for 45 seconds (to let any animation/layout settle down) and exit. Each test is run three times and the subsystem profiles are averaged and the run that shows the time closest to the average is used for the analysis. Figure 2 shows the subsystem times for each of these 30 sites.

This data shows that the majority of the time spent in the scenarios we measure today is dominated by the layout and rendering subsystems in Internet Explorer 8 Beta 2. This analysis clearly shows why the current set of micro-benchmarks, which concentrate on measuring how fast JavaScript executes, is not a good measure of end-user perceived performance for today’s class of AJAX applications. Note that browser architectures are different, so AJAX subsystems in Internet Explorer 8 Beta 2 (and hence the time profiles) would not be comparable across browsers.

**Real-World-Code Measurements**

As part of the early access program to get customer feedback, we have been working with a few customers that develop AJAX applications. Part of this engagement includes getting comparative data on performance scenarios on their applications. We worked closely with the Google Gmail product team and focused on making engineering fixes that directly resulted in improved end-user performance. As a result of these efforts, we were able to directly impact commonly used Gmail operations between 15% - 25% compared to Internet Explorer 7. We believe Gmail is quite representative of the current generation of AJAX applications in how they exercise the AJAX subsystems, so we think other AJAX applications will see similar improvements.

---

Shreesh Dubey

---

Microsoft’s goal in Internet Explorer 8 Beta 2 was to improve end-user perceivable performance so we were careful not to use micro-benchmarks as a singular metric to focus our engineering effort.
Performance Improvements in Internet Explorer 8 Beta 2

Great performance is one of many things being delivered in Internet Explorer 8 Beta 2, the latest version of the popular browser. Beyond a much faster JScript engine, Internet Explorer 8 Beta 2 includes profound performance improvements and exciting new developer features that make it one of the most exciting browser releases in years.

Internet Explorer 8 Beta 2 streamlines the browsing experience and ensures that users and developers can be more productive than in any other browser. Find out why the community is buzzing as I highlight some of the most significant performance improvements and great new features being released as part of Internet Explorer 8 Beta 2.

Faster JScript

As Shreesh Dubey discusses in the article “AJAX Performance Measurement Methodology”, the JScript engine is an important component affecting the speed of the AJAX pipeline used in many websites today. Internet Explorer 8 Beta 2 includes several enhancements that significantly improve those sites. We have worked hard in several broad areas to make sites faster and to reduce the amount of memory Internet Explorer 8 Beta 2 needs on your favorite sites. I describe a few of them below.

Circular Memory Leak Mitigation

A key factor influencing the performance of Web browsers today is their ability to minimize the amount of memory they require. We have made significant improvements to how the JScript engine manages memory in Internet Explorer 8 Beta 2.

The JScript garbage collector in previous versions of Internet Explorer manages the lifetime of JScript objects but not of DOM objects. As a result, the JScript garbage collector cannot break circular references between DOM objects and JScript objects, and memory leaks can result. In Internet Explorer 6, these circular references are broken when the Internet Explorer process terminates. In Internet Explorer 7, these circular references are broken when users navigate away from a page that contains the leaks.

Here are four simplified examples of patterns that would leak memory in previous versions of Internet Explorer:

**Direct Circular References**

The DOM object refers to itself and an object not in the tree:

```javascript
function leakTest1()
{
    var elem1 = document.createElement("DIV");
    elem1.thing = elem1;
}
```

Circular references between DOM objects not in tree:

```javascript
function leakTest2()
{
    var elem1 = document.createElement("DIV");
    var elem2 = document.createElement("DIV");
    elem1.thing = elem2;
    elem2.item = elem1;
}
```

**Circular References Created by Closures**

Closures with element created dynamically:

```javascript
function leak3()
{
    var elem = document.createElement("DIV");
    elem.onload = function ()
    {
        var y = elem;
    }
}
```

Leaks caused by function pointers:

```javascript
function leak4()
{
    var elem = document.createElement("DIV");
    elem.thing = elem.setAttribute;
}
```
In Internet Explorer 8 Beta 2, the JScript garbage collector treats DOM objects referenced by JScript objects as any other JScript object. Rather than wait until page navigation as in Internet Explorer 7 or process termination as in Internet Explorer 6, the garbage collector manages the lifetime of these DOM objects, and breaks circular references whenever possible throughout the lifetime of the site.

**JScript Runtime Improvements**

In addition to eliminating many sources of circular reference memory leaks, the Internet Explorer 8 Beta 2 JScript engine bakes in impressive optimizations to directly improve scripting speed.

**Optimized String Operations**

One of the most common JScript operations in use today is the concatenation of strings. Performance of string concatenation has significantly improved over previous versions of Internet Explorer. In some cases these changes have improved the speed of the + and += operators (as well as the built-in String.concat method) by several orders of magnitude. Coupled with further improvements to other string methods, these enhancements should significantly improve methods that use string concatenation, and obviate the need for developers to devise complex alternative solutions.

**Optimized Array Operations**

In addition to the great work to improve string operations, array storage and access within the JScript engine has resulted in order-of-magnitude performance gains for many built-in array methods. JScript authors implementing their own array methods (such as Array.push and Array.pop) as a faster replacement for the built-in array methods should consider using the built-in methods to improve code performance.

**Optimized Execution Engine**

Apart from optimizations to strings and array operations, the JScript execution engine shipped within Internet Explorer 8 Beta 2 also optimizes and improves the performance of function calls, object creation, accessing object prototype properties and specific lookup patterns for JScript variables scoped to the window or this objects.

The improvements to the Internet Explorer 8 Beta 2 scripting engine combine to improve pure JScript performance by 400% on the SunSpider benchmarking suite and speed up Google Gmail between 15% - 25% compared to Internet Explorer 7.

What this means for users is that they will have a faster and more reliable browsing experience on their favorite sites. Developers, in turn, can once again create great new content without having to expend effort managing object lifetime.

**Unblocked Downloads**

Beyond improvements to the Internet Explorer 8 Beta 2 scripting engine, another important aspect of site performance is the ability to quickly download page elements. When navigating to a Web page, Internet Explorer typically attempts to parallelize the download of page elements. In the case where Internet Explorer encounters an external script to download, however, previous browser versions would block parallel downloads until the script download was complete.

This approach was sensible when bandwidth was constrained, processing power was expensive, and sites were simple. Since the scripts being downloaded could alter the makeup of the page (and thus determine what should be downloaded) this approach could save valuable processing power that would otherwise be wasted. With CPU cycles relatively cheap and broadband widely available, with Internet Explorer 8 Beta 2 Microsoft recognized an opportunity to speed up browsing by taking advantage of these resources.

As a result, Internet Explorer 8 Beta 2 no longer blocks parallel downloads in the presence of external scripts. What this means is that Internet Explorer 8 Beta 2 can more efficiently use available bandwidth to make sites faster for users like you and me. This is especially true as sites become increasingly complex. Even better, developers do not need to change their sites whatsoever to take advantage of this improvement.

"Internet Explorer 8 Beta 2 streamlines the browsing experience and ensures that users and developers can be more productive than in any other browser."

**Faster Startup and Navigation**

In addition to the vast improvements to the JScript engine and the improved parallel download behavior, Internet Explorer 8 Beta 2 features numerous enhancements to reduce the amount of time it takes to start the browser and to navigate to your favorite sites.

That work has involved streamlining the Safety Filter, which now keeps users safe at minimal cost. Microsoft has also worked hard to reduce memory fragmentation throughout the browser, significant-
ly reduced registry accesses by up to 25% when navigating to some sites, and streamlined aspects of the Internet Explorer user interface. These changes and others have contributed to a much faster Internet Explorer.

**Great New Features**

In Internet Explorer 8 Beta 2 Microsoft is introducing several new features that developers can take advantage of to improve their site performance. Several of those features have been discussed in other articles in this issue.

Sunava Dutta introduced the Microsoft implementation of the HTML5 specification’s DOM Storage, Cross-Document Request (XDR), and Connectivity Events in his article, “Better AJAX Development with Internet Explorer 8 Beta 2”. Web developers can take advantage of these great cross-platform features to speed up their sites.

Beyond these three key features, John Hrvatin has introduced the JScript profiler built into the Internet Explorer 8 Beta 2 developer tools. A crucial addition to any Web developer’s toolbox, the profiler gives site authors the tools they need to understand and improve the speed of their sites.

In addition to these great enhancements, Internet Explorer 8 Beta 2 includes support for several new standards that developers can exploit to build lighting fast sites. The Internet Explorer 8 Beta 2 implementation of the ECMAScript 3.1 JSON recommendation, support for mutable DOM prototypes, and native support for the Selectors API are all cutting edge features that will form the basis for the next generation of Web sites and frameworks.

One of the things that Web developers have told us is that they need a way to extend the DOM with cutting-edge new features they can deliver as rapidly as their business demands.

**Native JSON support**

JavaScript Object Notation (JSON) is a lightweight data-interchange format based on a subset of the object literal notation of the JScript language and is often used to pass data between components of AJAX websites. Unfortunately, many sites don’t use JSON securely and rely on the JavaScript eval method to “revive” JSON strings back into JavaScript objects.

Security-conscious developers instead use a more secure JSON parser to ensure that the JSON object does not contain executable script, often at the price of a significant performance penalty.

The JScript engine that ships with Internet Explorer 8 Beta 2 implements the ECMAScript 3.1 JSON proposal. 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. I am excited at this addition, as it is not only secure but is also much faster than equivalent framework implementations.

To illustrate, here is an example of serializing and deserializing data using the JSON.stringify and JSON.parse methods:

```javascript
// Create a new object from the string
var parsedValue = JSON.parse(stringifiedValue);

// Convert the object to a string
var stringifiedValue = JSON.stringify(testObj);
```

**Streamlined DOM Searching**

Do you find yourself caught in the trap of assigning an ID to every element in your source code in order to use the ubiquitous getElementsByTagName API to retrieve what you are looking for? Perhaps you are a veteran user of getElementsByTagNames? Whatever element lookup technique you employ, you have probably run into those scenarios where one of these APIs just does not stretch as far as you need it to.

Once upon a time I too contributed to those geeky Web pages with animations that followed your cursor around the screen; I would write the layout of the page in CSS, and then use JavaScript to perform the animation by judicious use of setTimeout and [element].style.left/top.

No matter the scenario, you may find yourself searching for a better searching API, or you are turning to JavaScript frameworks that include tailored element lookup routines for you (prototype.js and jQuery, for example). Thankfully, with Internet Explorer 8 Beta 2 there is now a better way! You should acquaint yourself with querySelector and querySelectorAll APIs. They will likely become some of the most frequently used APIs in your repertoire. They plug directly into the improved Internet Explorer 8 Beta 2 CSS 2.1 selector support so that you can leverage the selector-matching engine at any time using JavaScript.

To experiment with the syntax, try replacing a call to getElementsBy('id') with querySelector('*id')—the results are the same. The following example uses querySelectorAll to grab all relevant elements by class and turn off their collective display status. Listing 1 illustrates the original unoptimized code demonstrating that the use of the Selectors API results in significantly less-code to download and execute on the client.
the fact that you can harness the native implementation to speed up your code dramatically. Just go to http://webkit.org/perf/slickspeed/ in Internet Explorer 8 Beta 2 to see for yourself!

Mutable DOM Prototypes

As with the support for the Selectors APIs, the support for mutable DOM prototypes gives developers the building blocks to create efficient and compact abstractions.

Like many programming languages, JavaScript objects derive from well-defined and documented types, which are discoverable through reflection. Unlike many programming languages, however, JavaScript makes these types directly available and (strangely enough) mutable through its support for prototype inheritance! To illustrate the feature, consider this example:

```javascript
function useSelectors(clsName)
{
    // Hide everything...
    var unset = document.querySelectorAll('h3, p, ul, code');
    for (var i = 0; i < unset.length; i++)
    {
        unset[i].style.display = 'none';
    }
    var set = document.querySelectorAll('.' + clsName);
    for (var i = 0; i < set.length; i++)
    {
        set[i].style.display = 'block';
    }
}

function showHideContentAreaContent(clsName)
{
    // ***********************************************
    // Begin: Hiding elements that should not be shown
    // Hide all the h3's (topic headers for sub-articles)...
    var h3Collection = document.getElementsByTagName('h3');
    for (var i = 0; i < h3Collection.length; i++)
    {
        // ...unless this element has a class that matches.
        if (h3Collection[i].className.indexOf(clsName) != -1)
        {
            h3Collection[i].style.display = 'block';
        } else
        {
            h3Collection[i].style.display = 'none';
        }
    }
    // Hide the p tags that have a className assigned...
    var pCollection = document.getElementsByTagName('p');
    for (var j = 0; j < pCollection.length; j++)
    {
        // ...unless this element has a class that matches.
        if (pCollection[j].className.indexOf(clsName) != -1)
        {
            pCollection[j].style.display = 'block';
        } else
        {
            pCollection[j].style.display = 'none';
        }
    }
    // Hide any UL elements that may be part of an article...
    var ulCollection = document.getElementsByTagName('ul');
    for (var k = 0; k < ulCollection.length; k++)
    {
        // ...unless this element has a class that matches.
        if (ulCollection[k].className.indexOf(clsName) != -1)
        {
            ulCollection[k].style.display = 'block';
        } else
        {
            ulCollection[k].style.display = 'none';
        }
    }
    // Finally, hide the CODE element on one article unless
    // it needs to be shown.
    var codeColl = document.getElementsByTagName('code');
    for (var m = 0; m < codeColl.length; m++)
    {
        if (codeColl[m].className.indexOf(clsName) != -1)
        {
            codeColl[m].style.display = 'block';
        } else
        {
            codeColl[m].style.display = 'none';
        }
    }
    // END: Hiding elements
    // ************************************************
}

Listing 1: Original [unoptimized] code to hide articles except those matching a certain class. Without the Selectors API, this code is almost 500% larger.

The Selectors APIs are available at two different scopes: document and element. If you want to find the first result of a particular selector, then use querySelector (which returns the found element directly). Alternatively, if you want to find all matching elements use querySelectorAll. Using element scope can be important when you want to narrow your results to a particular subtree of the DOM instead of collecting results from everywhere. Interestingly enough, the provided selector-string parameter is first checked against the entire document, however the elements returned must be within the proper subtree of the element on which the API was called.

The folks over at WebKit have written a test that allows you to compare the Internet Explorer 8 Beta 2 native implementation of the Selectors API to common JavaScript implementations—highlighting
How to Use Data URIs

In some circumstances you can exploit Data URIs to avoid expensive round trips to download content.

Keep in mind, however, that the base64 encoding incurs additional processing overhead and that Data URIs cannot be cached independently of their containing document, script, or style sheet.

To get started with Data URIs follow these best practices:

- Include Data URIs in a cacheable script or style sheet
- Use Data URIs only for small images
- Don’t embed Data URIs directly in HTML or other non-shared resources

Even with this simplified example, you can see that if xyz is a feature that enhances any and all array accesses, then the inclusion of it on the Array prototype is the right place to get the most impact per line of code. Thanks to our community feedback, Microsoft is pleased to bring the power and flexibility of JavaScript prototypes to the IE DOM as well. You may be thinking that I am getting far too excited by a feature that does not seem particularly remarkable, but bear with me for just a moment longer and I’m certain that you too will be a fan.

Compared to frameworks released by Web developers, features provided by a browser release at a slower pace and cater to a more general audience. One of the things that Web developers have told us is that they need a way to extend the DOM with cutting-edge new features—features only they can deliver as rapidly as their business demands. Essentially, developers need to be able to fill functionality gaps long before a native implementation shows up in any browser. DOM prototypes give Web developers the basic building blocks that they can supplement the DOM directly with extra functionality or replace existing implementations with their own.

Drawing from the previous example, if I want to add a feature to all image elements in the DOM I could add it to the HTMLImageElement prototype:

```
// Apply the change to the prototype.
HTMLImageElement.prototype.grow = function()
```

The alternative approach is to add the feature API manually to each individual image element separately—crippling performance. You can see how mutable prototypes provide a much simpler and efficient approach to this common problem.

In addition to providing a staging ground, DOM prototypes enable more efficient JavaScript by providing a single, logical place for element-related helper functions. For many scenarios the Element prototype is the most reasonable location for these additions, as it is a parent to all of the elements in the Internet Explorer DOM. As a result, changes to the Element prototype apply to all elements. A good example of how you can use this approach is to add the getElementsByClassName API to all elements; the draft spec of HTML 5 recently defined this API and Internet Explorer 8 Beta 2 does not implement it natively. You can define this method to immediately treat elements as if they supported it natively. Best of all, that method will be faster than more complex alternatives offered as part of generalized frameworks. See Listing 2 for the implementation of _MSHTML5_getElementsByClassName_Elem.

```
Listing 2: Implementation of the getElementsByClassName JavaScript function.

function _MSHTML5_getElementsByClassName_Elem(classList)
{
    var tokens = classList.split(" ");
    if (tokens.length == 0)
    {
        return null;
        // Per HTML5: "If there are no tokens specified in the
        // argument, then the method must return an empty
        // NodeList."
    }
    // Pre-fill the list with the results of the first token
    // searched.
    var staticNodeList = this.querySelectorAll("." + tokens[0]);
    // Start the iterator at 1 because the first match is already
    // collected.
    for (var i = 1; i < tokens.length; i++)
    {
        // Search for each token independently.
        var tempList = this.querySelectorAll("." + tokens[i]);
        // Collects the "keepers" between loop iterations.
        var resultList = new Array();
        for (var finalIter = 0; finalIter < staticNodeList.length; finalIter++)
        {
            // checking...
            resultList.push(staticNodeList[finalIter]);
            // This element was in both lists; it should
            // continue into the next round of token
            // checking...
        }
        // Copy the AND results for the next token.
        staticNodeList = resultList;

        // collected.
        for (var tempIter = 0; tempIter < tempList.length; tempIter++)
        {
            if (found)
            {
                // This element in both lists; it should
                // continue into the next round of token
                // checking...
            }
            // checking...
            resultList.push(tempList[tempIter]);
            // This element was in both lists; it should
            // continue into the next round of token
            // checking...
        }
        // This element was in both lists; it should
        // continue into the next round of token
        // checking...
    }
    return staticNodeList;
}
```
With that in mind, I hope you are eager to see how Microsoft’s investment to improve scripting speed, download pages faster, and reduce browser overhead enables you to browse faster.

The full benefit of Internet Explorer 8 Beta 2 is not limited to these immediate gains. Not only is Internet Explorer 8 Beta 2 speeding up the Web as it exists today, but it is delivering groundbreaking features that will let you build the next generation of lighting fast websites.

With Internet Explorer 8 Beta 2, one thing is clear. It’s a great time to be online.

---

**Data URI**

If you are like me, you have at one point likely built a website that required at least a few small images. In some cases you can use CSS sprites to combine images and avoid network round trips. In some cases, however, I had to resign myself to the inevitable and incur the cost of an extra round trip. No longer!

Normally, URLs act as pointers to network locations that contain data. For instance, http://www.example.com/ie.gif instructs the Web browser to download the image file named “ie.gif” from the HTTP server running at www.example.com.

Internet Explorer 8 Beta 2 introduces support for Data URIs. Rather than acting as a resource locator, a Data URI contains the resource data directly within the URI string. In order to embed binary data within the string, the data is first encoded using base64 encoding.

For instance, here is a Data URI that represents a 10x10 blue dot:

```
data:image/png;base64,iVBORw0KGgoAAAANSUhEUgAAAAoAAAAKAQMAAAC3/F3+AAAACXBIWXMAAA7DAAAOwwHHb6hkAAAAXNSR0IArs4c6QAAAARnQU1BAACxjwv8YQUAAAAgY0hSTQAAeiYAAICEAAD6AAAAgOgAAHUwAADqYAAAOpgAABdwnLpRPAAAAANQTFRFALfvPEv6TAAAAAtJREFUCB1jYMAHAAAeAAEBGNgTr6YAAAIkAAAAAElFTkSuQmCC
```

Internet Explorer 8 Beta 2 permits Data URIs of up to 32kb in length. Careful use of Data URIs in place of URLs can help you avoid costly network round trips and improve the overall performance of your sites.

**The Big Picture**

Now that I have completed my whirlwind tour of some of the great features that Microsoft will deliver in Internet Explorer 8 Beta 2, I think it’s important to take a step back to see the big picture. The focus of the performance improvements in Internet Explorer 8 Beta 2 center on building a platform that allows developers to get more done when they build and browse sites.
Reliability and Privacy with Internet Explorer 8 Beta 2

Reliability and privacy are two must-have features for every Web user. To that end, Internet Explorer 8 Beta 2 introduces powerful and easy-to-use features that improve the dependability of your browsing experience and the security of your personally identifiable information. Read on for all the details.

Reliability

For Internet Explorer, reliability means that the browser should always start quickly, perform well, connect to the Internet, and show websites without crashing or hanging. Most users want their browser to work, recover smoothly after a crash, and display the Web correctly. Users are not as concerned with what causes the problem, whether that be a poorly functioning add-on or poorly performing website. As part of Microsoft’s ongoing commitment to improve reliability, we have done a great deal of work in Internet Explorer 8 Beta 2 to make the browser more robust in all of these areas: performance, recovery and display. In particular, this article will discuss:

- **Loosely-coupled Internet Explorer:** An architectural feature that helps isolate different parts of the browser from each other for smoother browsing performance.
- **Automatic Crash Recovery:** A feature that is designed to get users back to browsing as quickly as possible after a crash.

![Figure 1: A tab in Internet Explorer 8 Beta 2 with the separate processes dimmed for illustration.](image-url)
Loosely-Coupled Internet Explorer

One of the Internet Explorer team’s most significant investments is in a feature called Loosely-Coupled Internet Explorer (“LCIE”), which is an architectural attribute that helps isolate different parts of the browser from each other, most notably, the frames from the tabs. LCIE is the foundation on which we have built several of our features, including Automatic Crash Recovery.

In Internet Explorer 8 Beta 1, we isolated the frame window, which roughly corresponds to the “chrome,” from the tabs by putting them in their own separate processes so that a tab can now crash without bringing down the rest of your browser. Visually, this separation would look like Figure 1, with the frame area highlighted and the tab area dimmed.

Building on Internet Explorer 8 Beta 1, we have continued to develop LCIE in Internet Explorer 8 Beta 2 to further improve reliability and performance. For Beta 2, we added the following changes:

Frame Process Merging

To help improve startup performance, we have reduced the number of processes that Internet Explorer starts. Instead of firing up two processes every time you launch the browser (one for the frame and one for your tabs), we now only fire up one frame process the first time you launch Internet Explorer. Subsequent launches will only start a new tab process or make a new tab in an existing tab process.

For users that are accustomed to browsing websites in multiple “sessions”—for example if you want to log in to multiple e-mail sites simultaneously—you can specify the “-nomerge” command line option to disable this feature.

More Tab Processes

It turns out that the vast majority of all Internet Explorer sessions contain three or fewer tabs. Accordingly, in Internet Explorer 8 Beta 2 we try to give users three efficient tab processes. This is contingent on the user’s computer capabilities, but the more capable a computer is, the more processes Internet Explorer will use, up to a point. Adding more processes gives users much better isolation in the event of a failure. If each tab is in its own process, websites are completely isolated from each other.

Virtual Tabs

We have also added the internal capability to “hot swap” the process from underneath a tab. Previously, Protected Mode worked on a per-process basis. For example, say you add a website to your trusted sites in Internet Explorer 7. If that site links to another site that is not in your trusted sites, it will cause you to switch browser windows when you click the link.

We improved this in Internet Explorer 8 Beta 1 with LCIE when we split the frame from the tabs. With the split, Internet Explorer can create a new tab in the same window and switch you to that tab as opposed to being “punted” to a new window.

Virtual tabs lets you navigate across Protected Mode in the same tab since Internet Explorer just switches the process under the tab to the correct integrity level. This is really just “UI-sugar”—virtual tabs do not impact security or protected mode in any way, other than to make it more convenient to transition between Protected Mode on and off.

LCIE’s ability to isolate different parts of the browser, coupled with more tab processes and virtual tabs, helps improve Internet Explorer’s performance and overall reliability.

Automatic Crash Recovery

In the event of a crash, Automatic Crash Recovery is designed to get you back to browsing as quickly as possible. It uses LCIE’s tab isolation to help localize the failure to your tab. If you experienced a crash in Internet Explorer 8 Beta 1, you may have noticed the bubble shown in Figure 2.

This is the “tab recovery experience.” The failure has been confined to your tab. Your browser never goes away and Internet Explorer gets you back to the site quickly.

What’s happening behind the scenes is that Internet Explorer is keeping track of an array of information about your tab. In Internet Explorer 8 Beta 1, the following data about each tab was stored:

- Current URI
- The travel log (your back/forward history)
- Tab order
- Which tab was active

Figure 2: The “tab recovery experience” in Internet Explorer 8 Beta 2.
When you crash, Internet Explorer tears down the old tab process, creates a new tab process, and recovers the stored data back into the tab. For many websites this works well; however, there are other websites, such as sites with Web forms, or sites that you need to log in to, that Internet Explorer 8 Beta 1 didn’t recover successfully.

In Internet Explorer 8 Beta 2, we improved this further by adding session cookie and form data recovery.

Session cookies are often used for authenticating the user to a website. Session cookies are temporary cookies that only persist for the lifetime of your browsing session. When you log in to a website, they usually give you a session cookie that contains a unique token that identifies you while you are logged in. As you navigate around the website, Internet Explorer sends your session cookie to the site, and the site can examine this token and determine that you are authenticated. Unlike persistent cookies, they are not written and retained on your hard disk.

In Internet Explorer 8 Beta 2, we recover your session cookies too, and still do not write them to disk! Internet Explorer stores copies of your session cookies in the frame process. When your tab crashes, Internet Explorer just copies them back from the frame into the tab, and the user is automatically logged back into the site they were using (Web-based e-mail, blog sites, social networking sites, etc.).

Note that session cookie recovery only takes place for tab crashes. If the whole browser crashes, the session cookies are lost; however, we do expect that the overwhelming majority of crashes will be isolated to the tabs as most crashes are caused by malfunctioning add-ons, which are now isolated to a tab process.

In addition to keeping you logged in with session cookies, Internet Explorer can now recover your form data. If you typed information, such as an e-mail, blog post, comments, into an HTML form, Internet Explorer can now recover that information.

Leveraging LCIE’s tab isolation allows Automatic Crash Recovery to quickly restore the user to their browsing session without having to log back in to their sites or re-enter new data into forms. Combined, LCIE and Automatic Crash Recovery provide an innovative and graceful way to recovery from crashes.

Privacy

Have you ever wanted to take your Web browsing “off the record”? Perhaps you’re using someone else’s computer and you don’t want them to know which sites you visited. Maybe you need to buy a gift for a loved one without ruining the surprise. Either way, your browsing history is your data, and you should be able to easily control it.

Delete Browsing History

In Internet Explorer 7, we added a feature called Delete Browsing History, which lets you delete all of the information that Internet Explorer saves in one click. This is a useful tool that has become a standard feature across all browsers. If there are things in your Web browsing past that you want to erase, you can do that easily.

The problem is that usually you don’t want to delete everything! Cookies, in particular, are really useful for storing preferences on websites that you use frequently. Many sites have a “remember me” option. Other sites, particularly financial websites, will store a cookie on each computer that you use to eliminate extra challenge questions (for instance, “What was your high school mascot?”).

Internet Explorer 8 Beta 2 solves this problem by adding an option that lets you keep cookies and temporary Internet files from websites in your Favorites list, as shown in Figure 3.

To avoid having your favorite sites “forget you,” simply add them to your Favorites, and make sure the “Preserve Favorites website data” checkbox is selected. Internet Explorer will preserve any cookies or cache files that were created by websites in your favorites.

Oh, by the way, we heard your feedback about checkboxes! Now Delete Browsing History will remember your preferences. We also added a “Delete
Reliability and Privacy with Internet Explorer 8 Beta 2

Existing cookies can still be read

The new DOM storage feature behaves the same way

New history entries will not be recorded

New temporary Internet files will be deleted after the InPrivate Browsing window is closed

Form data is not stored

Passwords are not stored

Addresses typed into the address bar are not stored

Queries entered into the search box are not stored

Visited links are not stored

Whether you know ahead of time that you don’t want your history recorded, or you just want to clean up your history without wiping out your preferences, Internet Explorer 8 Beta 2 puts you in control.

Andy Zeigler

Figure 4: InPrivate Browsing in Internet Explorer 8 Beta 2.
Secure Coding with Internet Explorer 8 Beta 2

The Internet Explorer team has made significant investments to ensure that Internet Explorer 8 Beta 2 is the most secure version to date. Many of these improvements (like the SmartScreen anti-phishing/anti-malware filter) operate automatically and require no changes to Web pages or add-ons. However, other security improvements will impact Web applications and browser add-ons. This article describes how to take advantage of these new Internet Explorer security features to help protect Web users and applications.

As we designed Internet Explorer 8 Beta 2, Microsoft security teams researched the common attacks in the wild and the trends that suggest where attackers will be focusing their attention next. For each class of attack, we developed a set of layered mitigations to provide defense-in-depth protection against exploits. Broadly speaking, there are two classes of threat that developers need to be concerned about: threats to Web applications, and threats to users’ computers.

Web Application Threats

As more and more applications and user data migrate to the Web, attackers are ever more interested in attacking Web applications. While attacks against server-side code (for instance, buffer overflows, SQL injection, etc) remain popular, cross-site scripting (XSS) attacks have become the most common class of software vulnerability. XSS attacks exploit vulnerabilities in Web applications in order to steal cookies or other data, deface pages, steal credentials, or launch more exotic attacks.

The XSS Filter

Internet Explorer 8 Beta 2 helps to mitigate the threat of XSS attacks by blocking the most common form of XSS attack (called a reflection attack). In a reflection attack, data from a HTTP request (e.g. a query string parameter or POST body) is “reflected” back in the HTTP response without proper output-encoding. That reflected script runs in the context of the returned page, leading to a script injection exploit.

The new XSS Filter operates as an Internet Explorer 8 Beta 2 component with visibility into all requests and responses flowing through the browser. When the filter’s heuristics discover script being sent in a cross-site request, it identifies and neuters the script if it is subsequently replayed in the server’s HTML response.

Figure 1 shows a case where the XSS Filter has identified a cross-site scripting attack in the URL. It has neutered this attack as the identified script was...
replayed back into the response page. In this way, the filter effectively blocks the attack without modifying the initial request to the server or completely blocking the entire response.

In the unlikely event that you wish to disable the filter for your pages, you can do so by setting a HTTP response header:

```
X-XSS-Protection: 0
```

The XSS Filter helps block the most common XSS attacks, but it cannot possibly mitigate all XSS vulnerabilities. It’s important that Web developers provide additional defense-in-depth and work to eliminate XSS vulnerabilities in their sites. Preventing XSS on the server-side is much easier than catching it at the browser; simply never trust user input! Most Web platform technologies offer one or more sanitization technologies—developers using ASP.NET should consider using the Microsoft Anti-Cross Site Scripting Library. To further mitigate the threat of XSS cookie theft, sensitive cookies (especially those used for authentication) should be protected with the `HttpOnly` attribute.

### Safer Mashups

While the XSS Filter helps mitigate reflected scripting attacks when navigating between two servers, in the Web 2.0 world, Web applications are increasingly built using client-side mashup techniques.

Unfortunately, many mashups are built unsafely, relying SCRIPT SRC techniques that simply merge scripting from a third-party directly into the mashup page, providing the third-party full access to the DOM and non-`HttpOnly` cookies.

```
<html>
<head><title>MyMashup</title>
<script language="javascript"
       src="http://untrusted.example.com"></script>
</head>
<body>This page includes a script file from another domain. That script obtains full access to this document and its cookies.
</body>
</html>
```

To help developers build more secure mashups, Internet Explorer 8 Beta 2 includes support for the HTML 5 cross-document messaging feature. The new `postMessage` call enables IFRAMES to communicate more securely while maintaining DOM isolation. The frames communicating via `postMessage` do not get direct DOM access, but can only send string messages to each other. Each message clearly identifies its origin, and the `varTargetUri` parameter helps ensure that messages are not misdirected.

```
// PostMessage will only deliver the 'Hello'
// message to frame 'o' if it is currently
// at recipient.example.com
o.contentWindow.postMessage('Hello',
   'http://recipient.example.com');
```

Internet Explorer 8 Beta 2 also introduces the `XDomainRequest` object to permit secure cross-domain retrieval of “public” data using HTTP. For more information, see “Better AJAX Development with Internet Explorer 8 Beta 2,” in this issue.

While HTML 5 Cross-Document Messaging and `XDomainRequest` both help to build secure mashups, a critical threat remains. When using either technique, the string data retrieved from the third-party frame or server could contain malicious script. If the calling page blindly injects the string into its own DOM, a script injection attack will occur. To help eliminate this threat, two new technologies can be used in concert with these cross-domain communication mechanisms to mitigate script-injection attacks.

### Safer Mashups: HTML Sanitization

Internet Explorer 8 Beta 2 exposes a new method on the window object named `toStaticHTML`. 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 mentioned previously. So, for example, you can use `toStaticHTML` to help ensure that HTML received from an HTML 5 `postMessage` call cannot execute script, but can take advantage of basic formatting:

```
document.attachEvent('onmessage',function(e) {
  if (e.domain == 'weather.example.com') {
    spnWeather.innerHTML =
    window.toStaticHTML(e.data);
  }
});
```

Calling:

```
window.toStaticHTML("This is some <b>HTML</b> with embedded script following...
                        <script>alert('bang!');</script>!");
```

will return:

```
This is some <b>HTML</b> with embedded script following... !
```

The sanitized string can be safely injected into the DOM without the possibility of script execution.

```
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 Beta 2 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.

Listing 1 uses both JSON and HTML sanitization to prevent script injection, even if the weather service returns a malicious response containing script:

HTTP/1.1 200 OK
Content-Type: application/json
Access-Control-Allow-Origin: allow *

{"Weather":{"City":"Seattle","Zip":98052,
"Forecast":{"Today":"Sunny","Tonight":"<script defer>alert('bang!')</script>Dark","Tomorrow":"Sunny"}}

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, application, etc).

For legacy 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 than the type specified by the Web server. For instance, 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. Due to the prevalence of legacy servers on the Web (for instance, those that serve all files as `text/plain`) MIME-sniffing remains 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 to view the “picture”, 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, etc.

To combat this threat, we’ve made a number of changes to the Internet Explorer 8 Beta 2 MIME-type determination code.

MIME-Handling: Restricting Images

First, Internet Explorer 8 Beta 2 prevents “up-sniff” of files served with `image/*` content types into HTML+script. Even if a file contains script, if the server declares that it is an image, IE will not run the embedded script. This change mitigates the picture-sharing attack vector—with no code changes on the part of the server. We were able to make this change by default with minimal compatibility impact only because servers rarely knowingly send HTML or script with an `image/*` content type.

Listing 1: Use JSON and HTML Sanitization to prevent script-injection

```html
<html>
<head>
<title>XDR+JSON Test Page</title>
<script>
if (window.XDomainRequest){
    var xdr = new XDomainRequest();

    xdr.onload = function(){
        var objWeather = JSON.parse(xdr.responseText);
        var oSpan = window.document.getElementById("spnWeather");
        oSpan.innerHTML = window.toStaticHTML("Tonight it will be <b>" + objWeather.Weather.Forecast.Tonight + "</b> in <u>" + objWeather.Weather.City + "</u>.");
    }
    xdr.open("POST", "http://evil.example.com/getweather.aspx");
    xdr.send("98052");
}
</script>
</head>
<body>
<span id="spnWeather"></span>
</body>
</html>
```
MIME-Handling: Sniffing Opt-Out

Next, we’ve provided Web applications with the ability to opt-out of MIME-sniffing. Sending the new X-Content-Type-Options response header with a value of nosniff prevents Internet Explorer from MIME-sniffing a response away from the declared content-type. For example, consider the following HTTP-response:

HTTP/1.1 200 OK
Content-Length:108
Content-Type:text/plain
X-Content-Type-Options: nosniff

<html><body bgcolor="#AA0000"> This page renders as HTML source code (text) in Internet Explorer 8 Beta 2. </body></html>

In Internet Explorer 7, the text is interpreted as HTML, as shown in Figure 2. In Internet Explorer 8 Beta 2, thanks to the nosniff option, the page is rendered as plaintext, as shown in Figure 3.

The nosniff option puts servers in control of MIME-sniffing, enabling servers that host untrusted content to reliably control IE’s interpretation of that content, helping to prevent script-injection attacks.

MIME-Handling: Force Save

Lastly, for Web applications that need to serve untrusted HTML files, we have introduced a mechanism to help prevent the untrusted content from compromising your site’s security. When the new X-Download-Options header is present with the value noopen, Internet Explorer prevents the user from opening a file download directly; instead, they must first save the file locally. When the locally saved file is later opened, it no longer executes in the security context of your site, helping to prevent script injection.

Taken together, the new cross-domain communication APIs, content-sanitization APIs, and MIME-sniffing improvements enable the construction of significantly more secure Web applications.

Threats to Users’ Computers

While Web application attacks are becoming more common, attackers are always interested in compromising ordinary users’ local computers. In order to allow the browser to effectively enforce security policy to protect Web applications, personal information, and local resources, attacks against the browser must be prevented.

Internet Explorer 7 made major investments in this space, including Protected Mode, ActiveX Opt-in, and Zone Lockdowns. In response to the harden-
Secure Coding with Internet Explorer 8 Beta 2

...ing of the browser itself, attackers are increasingly focusing on compromising vulnerable browser add-ons. For Internet Explorer 8 Beta 2, we’ve made a number of investments to improve add-on security, reduce attack surface, and improve developer and user experience.

"DEP/NX helps to foil attacks by preventing code from running in memory that is marked non-executable.

Data Execution Prevention (DEP/NX)

Internet Explorer 7 on Windows Vista introduced an off-by-default Internet Control Panel option to “Enable memory protection to help mitigate online attacks.” This option is also referred to as Data Execution Prevention (DEP) or No-Execute (NX).

We have enabled this option by default for Internet Explorer 8 Beta 2 on Windows Server 2008, Windows Vista SP1, and Windows XP SP3.

DEP/NX helps to foil attacks by preventing code from running in memory that is marked non-executable. 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.

For Internet Explorer 7, DEP/NX was disabled by default for compatibility reasons. Several popular add-ons were not compatible with DEP/NX and would crash when Internet Explorer loaded them with DEP/NX enabled. The most common problem was that these add-ons were built using an older version of the ATL library. Before version 7.1 SP1, ATL relied upon dynamically generated code in a way not compatible with DEP/NX.

While developers of many popular add-ons have since released updated extensions compatible with DEP/NX, some add-ons may not be updated before Internet Explorer 8 Beta 2 becomes available. Fortunately, new DEP/NX APIs have been added to Windows Server 2008 and recent Windows Service Packs to enable use of DEP/NX while retaining compatibility with older ATL versions. These new APIs allow Internet Explorer to opt-in to DEP/NX without causing add-ons built with older versions of ATL to crash.

Local Administrators can control DEP/NX by running Internet Explorer as an Administrator and unchecking the Tools > Internet Options > Advanced > “Enable memory protection to help mitigate online attacks” option.

You can see which processes are protected by DEP/NX on Windows Vista Task Manager’s Process tab; on earlier versions of Windows, you can use Process Explorer. In either case, ensure that the “Data Execution Prevention box” is checked in the View > Select Columns menu.

If you build Internet Explorer add-ons, you can help ensure users enjoy a smooth upgrade to Internet Explorer 8 Beta 2 by taking the following steps:

1. If your code depends on older versions of ATL, please rebuild it with ATL v7.1 SP1 or later (Visual Studio 2005 includes ATL 8.0).
2. Set the /NXCompat linker option to indicate that your extension is compatible with DEP/NX.
3. Test your code with DEP/NX enabled using Internet Explorer 8 Beta 2 on Windows Vista SP1 or Windows XP SP3.

Figure 4: The Per-Site ActiveX information bar allows users to choose where an add-on may run.
Have You Xiine It?

Experience the next generation in digital reading FREE.
Go to www.xiine.com to read CoDe Magazine and other great publications.

All You Can Read. Digitally.

Experience the next generation in digital reading FREE.
Go to www.xiine.com to read CoDe Magazine and other great publications.
4. Opt your code into other available defenses like stack defense (/GS), safe exception handling (/SafeSEH), and ASLR (/DynamicBase).

Application Protocol Prompting

While ActiveX controls are a prime target for attackers, some enterprising attackers have started to look for exploitable attack surfaces in processes that can be automatically launched by the browser.

Application Protocol handlers enable third-party applications (such as streaming media players and Internet telephony applications) to directly launch from within the browser or other programs in Windows. Unfortunately, while this functionality is quite powerful, it presents a significant amount of attack surface, because some applications registered as protocol handlers may contain vulnerabilities that could be triggered from untrusted content from the Internet.

Applications handling URL protocols must be robust in the face of malicious data. Because handler applications receive data from untrusted sources, the URL and other parameter values passed to the application may contain malicious data attempting to exploit the handling application. For this reason, handling applications that could initiate unwanted actions based on external data must first confirm those actions with the user.

To help ensure that the user remains in control of their browsing experience, Internet Explorer 8 Beta 2 will now prompt before launching application protocols, as shown in Figure 5.

To provide defense-in-depth, Application Protocol developers should ensure that they follow the Best Practices described on MSDN to ensure that handling applications are robust even when presented with malicious URLs.

Protected Mode Overview

Introduced in Internet Explorer 7 on Windows Vista, Protected Mode helps reduce the severity of threats to both Internet Explorer and extensions running in Internet Explorer by helping to

Per-Site ActiveX

A key attack surface reduction we made for Internet Explorer 8 Beta 2 is “Per-Site ActiveX,” a defense mechanism to help prevent malicious repurposing of controls.

When a user navigates to a Web site containing an ActiveX control, Internet Explorer 8 Beta 2 performs a number of checks, including a determination of where a control is permitted to run.

If a control is installed, but is not permitted to run on a specific website, an Information Bar appears (Figure 4) asking the user whether or not the control should be permitted to run on the current website.

End-users may adjust their per-site ActiveX settings using the Manage Add-ons dialog box. IT Professionals administering a system of computers running Internet Explorer 8 Beta 2 may choose to preset allowed controls and their associated domains. Such settings can be configured using Group Policy.

If your ActiveX control is designed for use only on your website and you do not wish to allow user-override, you should use the Microsoft SiteLock ATL Template to prevent your control from being used on other websites. Using the SiteLock template helps ensure that your control cannot be maliciously repurposed by other websites.

If an ActiveX control meets the requirements to place itself on the PreApproved list and has done so by listing the CLSID of the control within the following registry key, then the Per-Site ActiveX restriction is not applied to that control.

HKLM\SOFTWARE\Microsoft\Windows\CurrentVersion\Ext\PreApproved

Figure 5: Internet Explorer now prompts before launching application protocol handlers.
Protected Mode Improvements: Cookie-Sharing

Protected Mode restricts file writes (including cookies) to low integrity locations—this means that attempts to use the InternetGetCookie or InternetSetCookie APIs from a medium integrity application will fail to get or set cookies from Protected Mode Internet Explorer. In Internet Explorer 8 Beta 2, medium integrity applications can easily access low integrity cookies by using the new IGetProtectedModeCookie and ISetProtectedModeCookie APIs. As al-

Figure 6: Internet Explorer 7 introduced the Protected Mode architecture to help prevent silent installation of malicious code.

Figure 7: Redirection of navigations into Protected Mode make it difficult to monitor Internet Explorer instances.
File Upload Control

Historically, the HTML File Upload Control (<input type=file>) has been the source of a number of information disclosure vulnerabilities. To resolve these issues, two changes were made to the behavior of the control.

To block attacks that rely on “stealing” keystrokes to surreptitiously trick the user into typing a local file path into the control (which is later surreptitiously submitted), the File Path edit box is now read-only (Figure 9). The user must explicitly select a file for upload using the File Browse dialog.

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 filesystem information to the Internet. For instance, rather than submitting the full path:

```
C:\users\ericlaw\documents\secret\image.png
```

Internet Explorer 8 Beta 2 will now submit only the filename

```
image.png
```

Conclusion

Security is a core characteristic of trustworthy browsing, and Internet Explorer 8 Beta 2 includes major improvements to address the evolving Web security landscape. The Internet Explorer team is working to help protect users and provide new ways to enhance Web application security.

Thanks for your help in securing the Web!

Eric Lawrence
Do you need robust file support in your applications?

Aspose is your ONE Shot Solution!

OPEN CREATE MODIFY CONVERT & SAVE

- DOC
- DOCX
- WordML
- OOXML
- ODF
- RTF
- XLS
- XLSX
- CSV
- TAB
- PDF
- FO
- PPT
- POT
- PPS
- SWF
- XSN

With Aspose file management components developers can create applications which can open, create, modify, convert and save some of the most popular file formats in business, including: Word Documents, Excel Spreadsheets, PowerPoint Presentations, PDF Documents, Flash Presentations and Project Files. In addition, Aspose also offers an excellent line of products for data visualization including Aspose.Chart and Aspose.Grid, as well as components for spell checking, recurrence patterns, network protocols, barcode management and ad-hoc query generation.

All Aspose products can be purchased individually or in one complete suite, Aspose.Total. There are three separate Aspose.Total suites to accommodate the platform of your choice; .NET, SSRS and Java. Download your free evaluation copy today at www.aspose.com
As a web developer you have enough things to worry about... your choice of web hosting shouldn’t be one of them.

Appliedi.net offers a full range of worry-free WINDOWS HOSTING options to help you grow your business.

- Specializes in Shared, VPS & Managed Dedicated Windows Hosting Solutions
- 24x7 Email, Livechat & Toll Free Phone Support
- Microsoft Gold Certified Partner
- Secure & Reliable Network
- 99.99% Guaranteed Uptime SLA
- And always 100% Satisfaction Guaranteed!

Save 30% on WINDOWS HOSTING use coupon code: codemag1

So stop worrying, sign up today at www.appliedi.net/codemag1 & leave the hosting to us!