CHAPTER 1

Understanding Rich Internet Applications

The Web has changed. What began as a text-based system for scientists and academics to share information has grown into a vital part of life for many people. No longer can businesses afford to ignore it; for many, a simple collection of static HTML documents will no longer suffice. Users have come to expect a deeper, more involving Web experience; your users and your company or clients want Rich Internet Applications.

The idea of Rich Internet Applications, or *RIAs*, first came about in the early part of this century. One of the early proponents of the idea was a company named Macromedia. Macromedia developed its Flash Player technology — already the most installed piece of software in history, to provide designers and developers with a set of tools that could be used to create RIAs effectively. More important, Flash applications had the advantage of being completely cross-platform and cross-browser, unlike many other RIA technologies such as Ajax. Flash apps could be written and designed without regard to how they might perform on other machines or different browsers.

IN THIS CHAPTER

Moving Beyond HTML

Web pages are for the most part written in Hypertext Markup Language (HTML). HTML is a text-based language that allows developers to "mark up" text on their Web page with instructions as to how the Web browser should display text and other Web page items.

A brief history of the Web

HTML was invented in 1990 by Tim Berners-Lee, a scientist at the European Particle Physics Laboratory (CERN). Berners-Lee was seeking to develop a

means by which CERN's visiting scientists could more easily and effectively exchange information, even across otherwise-incompatible computer systems. The fruits of his labor, first used by Berners-Lee and a colleague on Christmas Day of that year, was Hypertext Transfer Protocol (HTTP), HTML, and the first browser/editor, which he dubbed WorldWideWeb.

Over the next several years, these three inventions were expanded upon by many other Web pioneers. However, the technologies would not see widespread use until 1994, the year the Netscape Corporation released its Web browser, Navigator. Navigator was the first Web browser widely available to the public. That year and the following saw the founding of such companies as Yahoo! and Amazon and also the first hints of what the Web might become.

Unfortunately, those early days also saw the first hints of what in many ways remains the biggest challenge facing Web developers around the world: browser incompatibility.

After Netscape rejected its licensing offers, Microsoft developed its own browser, Internet Explorer (IE). Microsoft placed IE on the desktop of its soon-to-be ubiquitous Windows 9x operating system. Microsoft and Netscape would battle for the next few years for market dominance in what has become known as the Browser Wars.

This period marked a low point for Web developers, as both companies released version after version of their browsers. With each new release there were features not supported by the other company's browser. An early effort at providing basic animation on Web pages led Netscape to introduce the ill-fated `blink` tag. Microsoft responded with the equally ill-conceived `marquee` tag. As neither browser supported the other's innovation, neither tag took hold.

For some time during this period, Web design books and tutorials went so far as to advocate that developers create two separate sites; one that would display correctly in Navigator, the other in IE. Others focused on the non-too-reliable JavaScript solution of *browser sniffing* — using JavaScript to attempt to determine the user's browser and display browser-specific code.

All of the early versions of HTML shared one thing in common: a complete lack of capability to provide any interactivity for the users. As the Web grew in popularity and profitability, companies began demanding interactivity. This would include animation, games and ads or other areas of pages that could meaningfully respond to users. All of those capabilities existed for the development of desktop applications; why then, clients would ask, were they lacking on the Web?

Dynamic HTML

An early and for a time popular solution to this lack of interactivity was Dynamic HTML (*DHTML*). DHTML pages contain basic HTML markup, which can then be manipulated at runtime with JavaScript. DHTML offered designers a lot of freedom to create the kinds of interactivity their clients and bosses were demanding. For the first time, Web photo galleries could be created that allowed users to simply move a mouse over a thumbnail of an image and have a larger version of that image appear automatically.

Unfortunately, DHTML suffered from the beginning from the very problem facing everyone who tried to develop Web pages: varying browser support. If Netscape and Microsoft were at odds with their support for various HTML tags, it was nothing compared to the way they handled JavaScript.

In fact, JavaScript had been developed by Netscape, and for a time was not supported at all in Internet Explorer. Microsoft had developed a competing language known not-that-creatively as *Jscript*, which of course was not, in turn, supported by Netscape. Even after Microsoft tacitly acknowledged that JavaScript would win that battle and introduced support for it in its browser (it would eventually abandon Jscript), it continued to support JavaScript in a slightly different form than what was used by Netscape. As a result DHTML added lots of impressive features into pages, but development of it became a nightmare. At times, a developer had to write two complete scripts, one for Navigator, the other for IE, then use complicated browser-sniffing to determine which script should be run. These issues would ultimately doom DHTML.

Ajax

By the early part of this century, Netscape could no longer hold its position against Microsoft's market dominance with IE. DHTML eventually gave way to easier, more stable solutions. In large part, this was due to the end of the Browser Wars. In the late 1990s, Netscape was bought out by one-time Internet giant AOL. By the middle of the first decade of the 21st century, AOL would officially dissolve Netscape as a separate corporate entity. Navigator survives, but as little more than the official AOL browser; few if any users who are not AOL customers use Navigator.

The second factor that brought about the end of the Browser Wars was the establishment of Web standards.

Driven by a grass-roots effort of Web designers who were tired of having to deal with browser incompatibilities and having to develop and maintain separate versions of their pages, the Web standards movement advocated for a *code once* theory. HTML designers agreed that if one was willing to ignore browser-specific variations, it was possible to develop a page that would display near-perfectly across all browsers.

The advent of Cascading Style Sheets (CSS) provided a more powerful formatting platform for Web pages than had previously existed in HTML. JavaScript was also accepted as an open standard. These two progressions meant that it was at long last possible to develop pages guaranteed to display the same in the growing market of *standards-based browsers*: Minor fixes were still at times required for proper display in IE, but they were just that — minor — and more recent releases of IE promised ever-growing standards support.

At about the same time, developers noticed a little-used JavaScript method, *HTTPRequest*. They discovered that HTTPRequest could use JavaScript to make requests to Web servers. This enabled a designer to create pages that could, dynamically at runtime, retrieve data to update a small portion of a page. Prior to this, pages had to be completely refreshed every time they needed new data, a slow and inefficient process. As the data being retrieved with JavaScript was most often formatted using Extensible Markup Language(XML), this new method of feeding data into pages was termed *AJAX*, or Asynchronous JavaScript and XML.

In order to simplify the development of pages, advanced developers created Ajax libraries: code bases that enabled designers without a deep understanding of JavaScript to develop pages that used the technology. In time, libraries began to develop that had nothing to do with XML or even with retrieving data from the server, but instead merely relied on JavaScript, standards-based HTML, and CSS to dynamically redraw pages. Thus Ajax became the logical (and much more successful) successor to DHTML. Eventually, its name would be changed to reflect this reality.

Today the term Ajax is not an acronym for anything. Instead it is simply the name for this idea of using XHTML, CSS, JavaScript and XML to create interactive Web pages and for creating Rich Internet Applications.

Understanding the Flash Platform

In 1995, a small company developed *FutureSplash*, a set of tools that allowed designers to create vector-based animations. Shortly thereafter, Macromedia — at the time itself a fairly small, San Francisco-based software company — bought FutureSplash. Macromedia quickly changed the name of the animation tool to *Flash*, and began development of two separate but closely linked applications: Flash Player, a free browser plug-in to play Flash movies; and an authoring tool, known then simply as Flash, for creating those movies. In 2006, Adobe Systems purchased Macromedia bringing Flash under its umbrella.

Flash platform overview

What began as a simple animation tool has now grown into a complete platform of tools for creating animation, embedding video, and developing complete Rich Internet Applications for the browser and the desktop alike.

Flash Player

At the heart of the Flash platform is Flash Player. Still free, it remains the most-installed piece of software on Earth. As of June 2009, Flash Player is installed on more than 99 percent of all Internet-connected computers and it has started to expand into other devices as well. Adobe plans to offer support for Flash Player on smart phones by 2010 and is working on bringing it to televisions and any other device that can display Flash Player's content. Already, some digital cameras use Flash Player for their menu interface.

Flash Player relies on the ActionScript *Virtual Machine*. This technology allows Flash Player to deliver its content in exactly the same way across all browsers and operating systems. Those creating content for Flash Player never have to worry about browser and operating system incompatibilities: If a version of Flash Player exists for a platform, your content will run on it, and it will look and behave exactly the same as whichever platform you used when you created it.

Flash Player's release cycle is tied to that of Flash Professional. Thus, Flash Player 10 was released in October 2008, alongside Flash Professional CS4; Flash Player 11 is scheduled to be released in the spring of 2010 with Flash Professional CS5.

Adobe Integrated Runtime

In 2007, Adobe released Adobe Integrated Runtime (AIR). AIR allows developers to leverage existing skills and technologies to develop desktop applications. While it is possible to use a variety of languages to develop AIR applications, even including HTML and CSS, most AIR development is done with tools in the Flash platform.

Similar to Flash Player, AIR applications run within a virtual machine, thus they are not subject to operating system variations. Therefore, a developer using a Macintosh computer can create an AIR application and rest assured that it will work precisely the same way on a friend's Windows-based machine (so long as that user has AIR installed). If not, AIR can quickly be download and installed from Adobe's Web site for free. While the focus of this book is on developing Web applications, AIR development is covered in later sections.

Flash Professional CS5

Flash Professional has long been the primary design tool in the Flash platform. Originally, it was simply Flash. When version 6 (known as *MX*) was released, Macromedia began marketing both a Standard and Professional version. Two releases later, the Standard version was dropped, but the program has since maintained the Professional moniker.

Flash Professional is a full-featured, vector-based design tool. It includes a powerful set of drawing tools, as well as a timeline to create animation. Designers can import assets from other applications, including images from Adobe Photoshop or other imaging software and vector drawings from Adobe Illustrator, and then use these assets to create rich interactive environments. As of version 7, designers can also import video into Flash movies. In fact, over the last few years Flash video has become the de facto format for delivering video on the Web.

The Flex framework

As the idea of Rich Internet Applications began to take hold, Macromedia realized that Flash Professional had a problem. It was built around the idea of creating animation, not complete Web sites. While Macromedia briefly flirted with the idea of turning Flash Professional into a more developer- and code-centric tool, doing so would require sacrificing that core designer market for Flash. Therefore, Macromedia instead developed a new tool set for creating Rich Internet Applications, which would become known as *Flex*.

Flex is in many ways nothing more than an alternate way to create Flash content. Using Flash Professional, you can draw assets and animate them on a timeline. While coding support exists in Flash Professional, it is somewhat limited and no method exists to completely describe a site's assets in code. Flex, on the other hand, is entirely code-based. Flex applications are created using a combination of two languages. Visual assets are built using MXML, an XML-based markup language developed specifically for Flex. (MXML does not actually stand for anything.) Interactions, server data retrieval, and other dynamic aspects of a Flex application are built using ActionScript.

The Flex framework is open source, so anyone can download the Software Development Kit (SDK) from `http://opensource.adobe.com` and develop Flex applications for free. Anyone can

also contribute bug reports and assist in the future development of the framework. The Flex framework is by necessity a key focus of this book, as Flash Catalyst is designed as a tool for the framework.

Flash Builder 4

While the Flex framework is open source and development of Flex applications is technically free, Adobe does offer a commercial Integrated Development Environment (IDE) for Flex. Originally known as Flex Builder, it was rebranded in early 2009 as *Flash Builder* to better fit the overall branding of the Flash Platform. Flash Builder is the tool of choice for professional Flex developers and indeed even for many developers creating applications in Flash Professional, due to the later tool's limited coding capabilities.

Flash Builder is built on the Eclipse toolset. Eclipse is itself an open-source IDE, used primarily by Java developers.

Cross-Reference

Flash Builder is covered in detail in Part IV.

Summary

As the Web has grown and matured so too have the needs of Web developers. Today's Web users expect more from the pages they visit, and Rich Internet Applications aim to satisfy that expectation with the help of such tools as Ajax and the Flash platform. In this chapter, you explored:

- A brief history of the Web
- The move from Dynamic HTML to Ajax
- The Flash Platform and its various tools and runtimes