

ENTERPRISE AJAX

STRATEGIES FOR BUILDING HIGH
PERFORMANCE WEB APPLICATIONS

"This book does an excellent job of covering advanced Ajax topics such as performance, scaling, Object Oriented JavaScript, usability, and accessibility. There's dozens of basic Ajax books on the market, but if you want to build mission critical business apps, this is the book to own."

—DUANE NICKULL, Sr. Technical Evangelist—
Adobe Systems, Inc. & Chair, OASIS SOA Reference Model Technical Committee

DAVID JOHNSON

ALEXEI WHITE

ANDRE CHARLAND

ENTERPRISE AJAX

This page intentionally left blank

ENTERPRISE AJAX

Strategies for Building High Performance Web Applications

***Dave Johnson, Alexei White, and
Andre Charland***



PRENTICE
HALL

Upper Saddle River, NJ • Boston • Indianapolis • San Francisco
New York • Toronto • Montreal • London • Munich • Paris • Madrid
Cape Town • Sydney • Tokyo • Singapore • Mexico City

Many of the designations used by manufacturers and sellers to distinguish their products are claimed as trademarks. Where those designations appear in this book, and the publisher was aware of a trademark claim, the designations have been printed with initial capital letters or in all capitals.

The authors and publisher have taken care in the preparation of this book, but make no expressed or implied warranty of any kind and assume no responsibility for errors or omissions. No liability is assumed for incidental or consequential damages in connection with or arising out of the use of the information or programs contained herein.

The publisher offers excellent discounts on this book when ordered in quantity for bulk purchases or special sales, which may include electronic versions and/or custom covers and content particular to your business, training goals, marketing focus, and branding interests. For more information, please contact:

U.S. Corporate and Government Sales
(800) 382-3419
corpsales@pearsontechgroup.com

For sales outside the United States, please contact:

International Sales
international@pearsoned.com

Editor-in-Chief:

Mark L. Taub

Managing Editor:

Gina Kanouse

Production: Deadline

Driven Publishing

Indexer: Angie Bess

Publishing

Coordinator:

Noreen Regina

Cover Designer:

Alan Clements

Composition:

Tolman Creek

Library of Congress Cataloging-in-Publication Data:

Johnson, Dave

Enterprise AJAX: Strategies for Building High Performance Web Applications / Dave Johnson, Alexei White, Andre Charland.

p. cm.

ISBN-13: 978-0-13-224206-6 (pbk. : alk. paper) 1. AJAX (Computer programming language) 2. Web sites—Authoring programs. 3. Web site development.

I. White, Alexei. II. Charland, Andre. III. Title.

TK5105.S885.A52J64 2007

006.7'86--dc22 2007015974

Copyright © 2008 Dave Johnson, Alexei White, Andre Charland.

All rights reserved. Printed in the United States of America. This publication is protected by copyright, and permission must be obtained from the publisher prior to any prohibited reproduction, storage in a retrieval system, or transmission in any form or by any means, electronic, mechanical, photocopying, recording, or likewise. For information regarding permissions, write to:

Pearson Education, Inc.
Rights and Contracts Department
501 Boylston Street, Suite 900
Boston, MA 02116
Fax: (617) 671-3447

This material may be distributed only subject to the terms and conditions set forth in the Open Publication License, v1.0 or later (the latest version is presently available at <http://www.opencontent.org/openpub/>).

ISBN-13: 978-0-13-224206-6

ISBN-10: 0-13-224206-0

Text printed in the United States on recycled paper at Courier Stoughton, Inc., Stoughton, Massachusetts.

First printing, July 2007

CONTENTS

Prefacexiii
Acknowledgmentsxvii
About the Authorsxviii

Chapter 1 AJAX and Rich Internet Applications 1

The Changing Web	3
Sore Points of Traditional Web Applications	4
AJAX Painkillers	6
AJAX in the Enterprise	9
Drivers for AJAX Adoption	10
Usability	10
Network Utilization	14
Data Centricity	15
Incremental Skills, Tools, and Technologies Upgrade	16
Server Agnosticism	16
What About the Application?	17
AJAX Technologies	18
Programming Patterns	19
AJAX Alternatives	20
XUL	20
XAML	20
Java Applets and Web Start	21
Adobe Flash, Flex, and Apollo	21
OpenLaszlo	22
Summary	22
Resources	24

Chapter 2 AJAX Building Blocks 25

JavaScript	25
JavaScript Types	26
Closures	28

Object-Oriented JavaScript	29
Prototype Property	33
OOP and Inheritance	34
Mutability	38
Threading	39
Error Handling	40
Namespacing	41
Document Object Model	42
Fundamentals	43
Manipulating the DOM	46
Cascading StyleSheets	48
Inheritance and the Cascade	49
Inline Styles	50
StyleSheets	51
Dynamic Styles	55
Events	59
Event Flow	60
Event Binding	62
Cross-Browser Events	65
The Event Object	68
Client-Server Messaging	69
XMLHttpRequest Basics	70
Dealing with Data	78
Summary	81
Resources	82

Chapter 3 AJAX in the Web Browser83

Incremental AJAX	85
Impact on the Server	85
HTML Standards	86
Document Type Definitions	87
Box Models	89
Bootstrapping AJAX Components	91
The Onload Event	91
Browser Tricks	96
Model—View—Controller	100
View	101
Controller	104
Model	105
AJAX MVC	107
AJAX Model	107
AJAX View	116

AJAX Controller	121
Aspect Oriented JavaScript	131
Summary	133
Resources	133

Chapter 4 AJAX Components135

Imperative Components	135
Declarative Components	139
Server-Side Declarative Programming	140
Declarative Google Map	142
Alternative Approaches	147
Custom Declarative Component	148
Behavior Component	152
Declarative Component	156
The Declaration	163
Building the Component	168
Basic Functionality	168
Connecting to the Server	173
Closing the Loop	177
Summary	180
Resources	181

Chapter 5 Design to Deployment183

Modeling AJAX	184
Applying the Model-View-Controller Pattern	185
Preempt Performance Problems	186
Prototyping	188
Wireframing	189
Verifying Design Decisions	196
Testing	208
Test-Driven Development	209
Debugging	226
Deployment	232
JavaScript Compression	232
Image Merging	238
Protecting Intellectual Property	240
Documentation	240
Summary	243
Resources	244

Chapter 6 AJAX Architecture247

- Asynchronous Messaging249
- Polling250
- Server Push251
 - Comet252
- Tracking Requests253
- Caching: Approaching Data255
- Basic Caching256
- Caching in the Component257
- Caching in the Browser261
- Caching on the Server265
- Caching in the Database269
 - MySQL269
 - MS SQL Server270
 - Oracle270
- Updating the Server Model: Concurrency270
 - Pessimistic Locking271
 - Read-Only Locking271
 - Optimistic Locking272
 - Conflict Identification272
 - Conflict Resolution275
 - Automated Conflict Resolution275
- Throttling276
 - Client276
 - Server278
- Scaling278
 - Load Balancing and Clustering280
 - AJAX Scaling Issues281
- Offline AJAX282
- Firefox Offline Storage284
- Internet Explorer userData Offline Storage287
- Using Flash Client Storage288
- Offline AJAX and Concurrency292
- Summary293
- Resources293

Chapter 7 Web Services and Security295

- Web Services295
- Web Service Protocols296
 - Representational State Transfer296
 - XML Remote Procedure Call297

Web Services	298
Choosing the Right Tool	300
SOAP on the Client	302
IBM Web Services JavaScript Library	303
Firefox	305
Internet Explorer	307
Cross Domain Web Services	309
Server Proxy	310
URL Fragment Identifiers	312
Flash Cross Domain XML	315
Script Injection	315
Security	317
Security Concerns with AJAX	318
Cross-Domain Vulnerabilities	319
Cross-Site Scripting	319
Cross-Site Request Forgery	325
JavaScript Hijacking	327
SQL Injection	330
Prepared Statements	331
Stored Procedures	332
XPath Injection	333
Data Encryption and Privacy	334
Firewalls	336
Summary	337
Resources	337

Chapter 8 AJAX Usability **339**

Common Problems	340
The Back Button and Bookmarking	340
Page Weight	352
Auto-Commit	357
Accessibility	358
Identifying Users with Accessibility Needs	359
JavaScript and Web Accessibility	360
Screen Readers and Accessibility	360
What Not to Do for Screen Readers	360
A JAWS-Compliant AJAX Interaction	361
Keyboard Accessibility	364
Usability Testing	367
Quick-and-Dirty Testing	367
Recruiting Participants	368
Designing and Running Tests	368
Software-Assisted Testing	369

- Tools for Testing Usability 369
- General Tips for Software-Assisted Testing 370
- Summary 371
- Resources 371
 - The Back Button 371
 - Usability Testing 372

Chapter 9 User Interface Patterns 373

- Display Patterns 373
 - Animation Patterns 374
- Interactivity Patterns 384
 - Basic Interactivity Patterns 384
- Summary 396
- Resources 396
 - Drag-and-Drop Resources 396
 - Progress Bar Resources 397
 - Activity Indicator Resources 397
 - Color-Fade Resources 397
 - In-Place Editing Resources 397
 - Drill-Down Resources 397
 - Live-Searching Resources 397
 - Live-Forms Resources 398

Chapter 10 Risk and Best Practices 399

- Sources of Risk 400
 - Technical Risks 400
 - Cultural/Political Risks 400
 - Marketing Risks 400
- Technical Risks 401
 - Reach 401
 - Browser Capabilities 403
 - Maintenance 404
 - Forward-Compatibility 405
 - Third-Party Tools Support and Obsolescence 407
- Cultural and Political Risks 407
 - End Users' Expectations 407
 - Trainability 408
 - Legal 409
- Marketing Risks 410
 - Search Engine Accessibility 410
 - Reach 412
 - Monetization 413

Risk Assessment and Best Practices	413
Use a Specialized AJAX Framework or Component	414
Progressive Enhancement and Unobtrusive JavaScript	414
Google Sitemaps	417
Visual Cues and Affordances	418
Avoid Gold Plating	419
Plan for Maintenance	420
Adopt a Revenue Model the Works	420
Include Training as Part of the Application	421
Summary	422
Resources	423
Search Engine Optimization	423
Statistics	423
Roadmaps	423
Screen Capture Tools	423

Chapter 11 Case Studies **425**

U.S. Department of Defense Re-Arms with Web 2.0	425
Background	425
The Challenge	426
The Solution	427
Technologies Used	427
The Outcome	428
Agrium Integrates AJAX into Operations	429
Background	429
The Challenge	429
The Solution	430
Technologies Used	432
The Outcome	433
AJAX Aides International Transportation and Logistics Firm	433
Background	434
The Challenge	434
The Solution	436
Technologies Used	438
The Outcome	439
Summary	440
Resources	441

Appendix A The OpenAjax Hub **443**

The Key Feature: The Hub's Publish/Subscribe Manager	444
An Example	444
Future Toolkit Support for the OpenAjax Hub	446

Index **447**

This page intentionally left blank

PREFACE

If you are like many of the talented developers we meet, you're interested in AJAX and how you can use it to improve your web applications. You may have even done some preliminary research online, checked out Ajaxian.com, or read a beginner's book to AJAX development. If not, then you're like an even larger group of talented people who just want to break into AJAX and want to get started. In one respect or another, we've all been there. The good news is that as a community of developers, we're finally starting to figure this thing out. In the end, it's not that hard.

We decided to write this book because we were frustrated that there was little information available for the more advanced topics in AJAX development. This was mainly because people in the industry were still "writing the book" on some of these topics, and despite a couple of years in mainstream use, AJAX was just creeping into the enterprise software stack. We wanted to create a resource of information that would be of interest to enterprise developers in this space. To that end, we have tried to bring together current development approaches with JavaScript and the other technologies that comprise AJAX and present it in a way that would be familiar and amenable to any enterprise developer.

WHY DO YOU NEED THIS BOOK?

Most of this content has been derived from our years of first-hand experience in building AJAX applications and user-interface components at Nitobi (www.nitobi.com). We feel that this represents an excellent cross-section of the knowledge we acquired during this time and should serve as a useful resource for developers hoping to include AJAX in their development projects. If you are interested in becoming more skilled in the areas of JavaScript development, troubleshooting Ajax quirks and performance problems, and in designing usable software from the ground up, this book should serve as an excellent resource.

We've given a considerable amount of time to discussing how to write JavaScript code in a way that should be familiar to Java or C# developers to get you up and running quickly. In doing so, we describe AJAX development with familiar software design patterns at the forefront of our minds and include information on some of the hottest topics in AJAX development, such as security and offline storage. We also present real solutions to building high-performance AJAX applications, not only through code optimization, but also through taking advantage of Internet infrastructure mainstays, such as caching.

This book takes a slightly different approach than other AJAX books in that we try to present a well-rounded discussion—one that includes (of course) a lot of advice about programming and a fair amount of discussion on issues such as application usability, accessibility, and internationalization. It also includes a framework for assessing risk in an AJAX development project, and it spotlights some developers who use AJAX in real enterprise applications to see what can be learned from their experiences.

WHO IS THIS BOOK FOR?

Enterprise AJAX has been written with intermediate-to-advanced server-side developers in mind (Java, object-oriented PHP, or ASP.NET). Many of the concepts in the book have been adopted from the time honored software engineering patterns introduced by the “gang of four” (that is, Erich Gamma, Richard Helm, Ralph Johnson, and John Vlissides, authors of *Design Patterns: Elements of Reusable Object-Oriented Software* [Addison-Wesley Professional]). Readers would benefit from a basic understanding of software design patterns, or at least an interest in learning more about them because they are applied throughout the book. We hope that delivering AJAX in a familiar way using patterns can help the more experienced developer understand the concepts and ideas more easily.

Perhaps more important than understanding patterns, you should ideally have at least a basic knowledge of JavaScript, HTML, and CSS. Even some understanding of XML, XSLT, or JSON can be helpful, although not essential. Furthermore, we expect that you are experienced with server-side programming in an object-oriented language such as Java, C#, or PHP.

After reading this book, developers should be familiar with the constituent parts that make up the AJAX technology stack and be familiar with

object-oriented JavaScript development. Similarly, you will have a good knowledge of the tools available to aid in developing AJAX applications and a good knowledge of various AJAX issues, such as security, usability, and accessibility.

WHAT'S IN STORE

We begin in Chapter 1, “AJAX and Rich Internet Applications,” by covering the basics of what an AJAX application is and how the pieces all fit together. We also discuss the evolution of the web application and some of the reasons that AJAX is becoming the preferred solution for web-based applications.

In Chapter 2, “AJAX Building Blocks,” we dive into the AJAX technology stack. This includes critical information about the right way to program JavaScript, and we pay special attention to object-oriented JavaScript development, the Document Object Model, cascading stylesheets, events, and XMLHttpRequest object, as well as other issues relating to transferring data from the client to the server.

Chapter 3, “AJAX in the Web Browser,” builds on Chapter 2 and lays a foundation for understanding the major browser differences, and with that knowledge, it looks at how one can build AJAX applications using the Model-View-Controller design pattern. In particular, you see how to write a client-side Model in JavaScript, how to generate HTML views from data, and how to connect the Model and View using a JavaScript-based Controller that relies on a publish-subscribe event system.

In Chapter 4, “AJAX Components,” we are ready to look at how you can build an AJAX user-interface component for use in a web application. In particular, we examine the differences between an imperative and declarative approach, and we look at some of the caveats of a declarative approach while presenting a complete example of building an AJAX-based data-grid component.

At this point in the book, we look at some of the overarching goals and problems with AJAX development. Chapter 5, “Design to Deployment,” specifically looks at issues throughout the software development lifecycle that are unique to AJAX, from application design to testing to deployment. You should leave this chapter with a good idea of various AJAX performance problems as well as many of the tools that are useful from the start to end of any AJAX development project.

Chapter 6, “AJAX Architecture,” introduces the reader to various architectural issues surrounding AJAX development. This includes investigating asynchronous messaging patterns and approaches to server communication such as server push, caching, scaling, and offline AJAX. Although many of these are common to any web-based application, we approach these issues with a unique AJAX perspective.

Building on Chapter 6, Chapter 7, “Web Services and Security,” discusses how AJAX can fit into a service-oriented architecture using Web Services in the web browser, as well as the different security problems that can arise when building an AJAX web application.

Chapter 8, “AJAX Usability,” starts the final section of the book by exploring some pertinent topics in usability, specifically where they apply to building AJAX applications for everyday users. Of interest in Chapter 8 are complete solutions to common problems such as the Back-button problem and approaches to addressing accessibility and internationalization.

Chapter 9, “User Interface Patterns,” is a hands-on exploration of some powerful AJAX user-interface patterns including in-place editing, master-detail, live forms, and drag and drop. These are some of the core user-interface design patterns that developers should be aware of when building almost any AJAX application.

In Chapter 10, “Risk and Best Practices,” we shift gears and explore sources of risk in developing scalable enterprise-grade AJAX applications. This is likely the least explored topic in AJAX books but is equally important to the technology itself when considering building a new application.

To wrap things up, in Chapter 11, “Case Studies,” we look at some actual AJAX implementations in demanding enterprise environments. We speak to the developers and hear what they did right or wrong and what they would do differently next time.

All in all, we hope this gives you a new perspective on AJAX development, and most of all, that you come away with some new skills to bring to your development projects.

Support/Feedback

We tried, of course, to keep all the information in this book as current and correct as possible, but errors are bound to slip through. We apologize in advance for any inaccuracies. Please see the book website <http://www.enterpriseajax.com> for any errata.

In addition, you will find all the source code from this book on the website for convenient download. All the source code is available under a GPL license.

We're also eager to get feedback on the book, code samples, and so-on for the next edition. Please direct this feedback to enterpriseajax@nitobi.com.

This page intentionally left blank

ACKNOWLEDGMENTS

This manuscript would not have been possible without the generous support of many people behind the scenes. We would like to thank our publisher Prentice Hall, and especially Mark Taub who kept the whole thing on the rails. Very useful feedback on the book came from Brent Ashley, Uche Ogbuji, and John Peterson; it was much appreciated. We'd also like to thank our supportive team at Nitobi who picked up the slack when we were off writing chapters and who contributed technical and editorial know-how: James Douma, Jake Devine, Joel Gerard, Mike Han, and Brian Leroux.

Dave Johnson: Of course, I would like to thank Alexei and Andre for their help on getting this project complete as well as a few other people who help us behind the scenes, such as Jordan Frank. Kristin, of course, has been monumental in keeping me sane, and I have been sure to always trust the words of Jack.

Alexei White: In addition to the people already mentioned, I'd really like to thank my co-authors, Dave and Andre, and the many other contributors to this project who all lent their expertise in one way or another. These include Bill Scott, Christian Van Eeden, Dave Huffman, Mike Hornby-Smith, Bob Regan, Gez Lemon, and Luke Wroblewski. I also want to thank Lara for encouraging me to sit down and work when all I wanted to do was play Frisbee.

Andre Charland: I'd first like to thank Dave Johnson and Alexei, my co-authors, for allowing me to help with the book. It's been an honor and very rewarding. I'd like to thank my Mom and Dad and Jonny for pushing me through when I wanted to quit.

ABOUT THE AUTHORS



Dave Johnson Dave is the co-founder and CTO of Nitobi Software, a Vancouver-based AJAX component vendor and consulting firm. Dave spends most of his time on architecting and building high performance AJAX components for use in web-based applications. A core focus of Nitobi is building AJAX components and user interfaces that deliver real value to customers through increased productivity and higher efficiency. Dave has spoken around the world about AJAX and web development, including AJAXWorld 2006, XTech 2007, and JavaOne 2007. Dave has a bachelor of science degree in electrical engineering from the University of British Columbia and is completing his Ph.D. at Imperial College London.



Alexei White Alexei is a developer, designer, and user-experience advocate. As product manager for component tools at Nitobi and a long-time developer of AJAX components and applications, he tries to find ways to build web applications that are faster, cheaper, and that users love. He is the primary architect of RobotReplay (www.robotreplay.com), a next-generation web-analytics tool by Nitobi and SayZu (www.sayzu.com), an AJAX-driven, up-and-coming web survey service. At Nitobi, he has been involved in the design and development of many mission-critical and large-scale web applications with an emphasis on rich, AJAX-driven interfaces. Alexei has a bachelor's degree in commerce from the University of British Columbia, and he lives in Vancouver.



Andre Charland Andre Charland co-founded Nitobi in 1998 after working for several other Internet startups. As president and CEO, he is directly involved in software development and has successfully executed more than 100 development projects. He was also an early proponent of the building blocks of AJAX. Andre has spoken widely on AJAX, blogging, and web usability. He has been quoted internationally in the media on blogging for business and maintains his own blog at <http://captainajax.com>. Charland is on the board of BALLE BC and co-founder of the Social Tech Brewing Co.

AJAX AND RICH INTERNET APPLICATIONS

Traditional web-based applications are common place in today's enterprise. They are used for everything from customer relationship management (CRM) to enterprise resource planning (ERP). Although useful, these applications are, for the most part, built largely depending on traditional web-application stalwarts of HTML forms and whatever preferred server-side programming to do the heavy lifting. In these traditional web applications, the user interface (UI) is commonly rigid and noninteractive with any data entered by the user requiring a complete web page refresh to have that data submitted to the server. The combination of an unfamiliar HTML forms-based UI with the lengthy delay associated with refreshing the entire web page—data, style, structure, and all—can result in a thoroughly tedious experience for the end user.

This is where Asynchronous JavaScript and XML (AJAX) can be a useful tool in improving web application usability. It's spawning a new breed of web applications that can expand the possibilities of what users can accomplish inside a web browser. AJAX is not only improving upon stale and archaic web architectures, but it also enables web-based applications to rival or surpass the importance of desktop applications in terms of usability and user experience. AJAX even allows powerful new application workflows and visualizations that currently have no desktop software-based equivalent—not necessarily because of a technological shortfall on the part of desktop developers but certainly because AJAX has put Rich Internet Applications (RIA) within reach of most web developers. From that perspective, AJAX has already changed and will continue to change the way users view traditional web and desktop applications alike.

Although AJAX recently garnered widespread acclaim from its use in the popular Google web applications such as GMail and Google Maps, it has actually been around, along with the constituent technologies that

comprise the AJAX acronym, for nearly a decade. AJAX is primarily just a renaming of dynamic HTML (DHTML), which in the past was shunned by the developer community yet today has become a hot ticket. Most of the technologies and techniques associated with AJAX are well understood. Although AJAX is particularly en vogue in public web application development, it is also starting to make headway in the enterprise setting. This book introduces AJAX to developers who are accustomed to working with traditional web applications in the enterprise, be it anything from CRM to e-commerce application development. We present AJAX techniques giving a firm grounding in the technical details that can enable you to build advanced AJAX applications that improve application usability and, therefore, impact the business bottom line.

The question begs to be asked, however, “What place does a rich-client technology like AJAX have in the enterprise?” You can think of the benefits in at least three ways:

- AJAX can improve and empower the user experience for end users, making them more effective and satisfied.
- AJAX can reduce the demands on network and server infrastructure, saving money by reducing maintenance and even bandwidth, and improve quality of service for all users.
- AJAX can create the possibility for new kinds of functionality not possible or practical in a traditional application model, giving users new tools to achieve their goals.

To understand why all this can be true, you need to appreciate how incredibly limiting the traditional web application model is and how AJAX makes more from the sum of its parts. The opportunity to innovate with web experiences drives the use of XMLHttpRequest, JavaScript, and Cascading Style Sheets (CSS) and creates new opportunities for the enterprise.

There’s no question that the enterprise AJAX marketing machine is in top gear. Enterprise vendors are supporting AJAX in many forms. IBM has initiated the Open AJAX Alliance, and Dojo dominates web-development discussion boards. Microsoft released ASP.Net AJAX, and Google has its Web Toolkit (GWT) targeted at Java developers. Oracle has ADF, a set of AJAX components for Java Server Faces, and Yahoo released the Yahoo User Interface library (YUI). Adobe supports Flex and AJAX integration through the FA Bridge and has released an open-source AJAX framework

called Spry. Underneath it all, however, is a genuine and compelling need to improve the way enterprise web applications are designed.

The Changing Web

Microsoft first introduced the core piece of technology required for AJAX functionality, the XMLHttpRequest (XHR) object, at the end of the '90s in Internet Explorer 5. At the same time, it introduced Outlook Web Access (OWA), which was quite an impressive AJAX interface and far ahead of its time. The main drawback at that time was that it was not possible to use XHR in any other browser, and there was strong reluctance in the community to locking into yet another Microsoft tool or platform. This is evidenced by the slow adoption of XHR in mainstream development until recently.

With the eventual introduction of XHR remote scripting in Firefox and Safari, it became possible to construct rich asynchronous communication in a cross-browser fashion. Implicitly, this meant that XHR could be deployed to wide and diverse audiences without much risk. When combined with JavaScript, DHTML, and CSS, it became possible to build rich client applications without the annoying refresh that characterized web applications. Unlike many other rich client techniques or technologies, discussed in a later section, AJAX is based on open standards supported by different browsers and operating systems—virtually eliminating the fear of vendor lock-in and increasing the opportunities for portability.

Everything in a traditional application revolves around the web page being a static view into an application that is based entirely on a web server. The only possible user interaction is to enter data into a web form or click a link—both of which result in the entire page being refreshed whether it was to update an entire customer record in a CRM application or to change the application state between viewing a customer record to editing it. In some respects, the traditional web application leaves much to be desired—such as when entering large amounts of data. At the same time, there are many situations in which the traditional web application excels; applications such as search engines or document repositories have long been useful and successful examples of traditional web applications. Furthermore, the strengths of the traditional web, for example, the HTTP protocol and resource caching, are strengths that are also used by AJAX-based applications.

Unlike popular AJAX mapping or email applications, most enterprise web applications are built around data entry, data editing, or data reporting. The most common data entry applications consist of a list of data such as customer records or sales information in a CRM application where items can be added to the list, deleted, or edited. Let's look at how the user interaction might play out in a traditional and an AJAX-based web application when a hotshot salesman is asked to use the new, but painfully slow, online CRM tool to track his meetings, contacts, and progress in the sales process.

Sore Points of Traditional Web Applications

As the salesman logs in to the application, he's confronted with a web page containing a list of ten potential customer records. In most traditional web applications, this sort of functionality would be achieved with a static HTML `<table>` listing each of the data records, beside which would be buttons that link to edit and delete pages. The salesman now wants to update a record with some new information. The first task is to locate the record. If it's not in the first ten items, he will have to search, which involves navigating through the data in the list by paging to the next ten items and waiting for a page to refresh. When he locates the record, he clicks the Edit button. Clicking the Edit button sends a request to the server; then, a new page is sent up to the web browser with a number of form fields on a page. Most of the form fields are text fields; some provide check boxes, drop down lists, or simple data validation (like checking to ensure a local telephone number has seven digits). On the data edit form, there would be little in the way of keyboard shortcuts, aside from the traditional Tab and Shift + Tab functionality to move between edit fields. After the data is edited, the user clicks the Save button at the bottom of the page, which sends the data to the server so that it can validate the data and commit it to the database. Another page is sent back to the browser to confirm the save. If an error occurs in the data, the user gets a visual cue on the form page that needs to be sent back to the browser, makes the appropriate edit, and clicks the Submit button again. A fairly slow and tedious process if you have to do this same operation many times a day.

Rather than having a separate form for editing data, the data listing web page can be updated to an editing page where each data record can be edited at once. After all the changes are made, they can be submitted to the server to be saved. In the name of usability, this is the type of UI that many traditional web applications might use rather than the single record editing scenario previously described. When the user decides to save the

data, it must all be saved at once rather than incrementally as it is entered or updated by the user. This means that all the data must be sent to the server in one large batch, which can result in one of several possible outcomes:

- Concurrency or validation issues force all the data to be redisplayed in a busy and hard-to-understand format prompting the user to fix multiple problems with the data at once.
- Momentary network or server errors can cause the data to be corrupted or lost completely with little aid for the end user to resubmit the data.
- User authentication fails and all the changes are lost.

Whatever the outcome, it generally results in several, long page refreshes as the server persists the data to the database and redirects to a new web page causing a great deal of frustration and anguish for the end user. The interactions between the user and the application are illustrated in the sequence diagram in Figure 1.1. Of particular note are the regions where the user sits idle waiting for the response from the server. (This time is often spent playing Solitaire.)

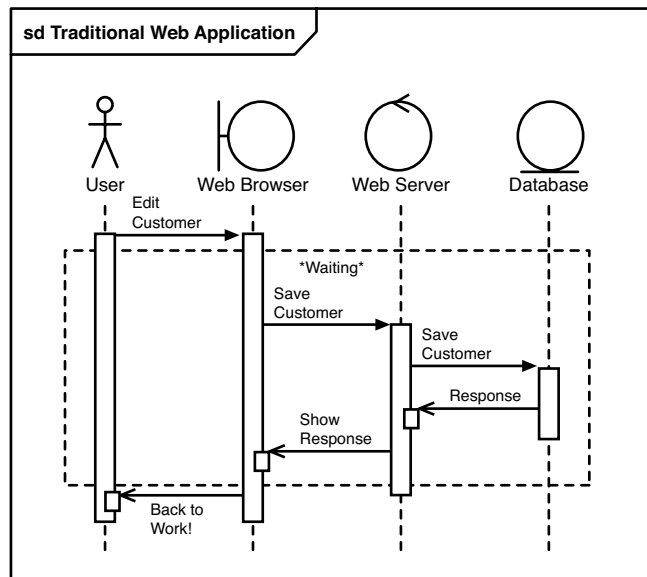


Figure 1.1 Sequence Diagram of Traditional Web Application Data Editing Workflow—The Dashed Boxes Represent Times When the End User Is Forced to Wait While Processing Is Performed on the Server

HTML forms do make sense for certain types of data, especially for novice users or infrequent operations; however, for applications with lots of complex data that has to be navigated through quickly and edited on-the-fly, they are painful. If a user needs to copy data from a spreadsheet or email into the application, it means retyping everything or copy and pasting each individual piece of data. Usability experts sometimes refer to this as “swivel chair integration,” and it doesn’t take a usability expert to figure out that this is not an efficient way of working and is a tedious experience.

AJAX Painkillers

Unlike the traditional web forms approach to a high-volume data entry application, an effective application needs to be responsive and intuitive. To that end, the impact on the users’ workflows should be minimal; for example, the users need to scroll through thousands of prospective customer records as though the data was accessed right from their local computer, as opposed to paging through ten records at a time. They also need to continue entering data into the application while data is saved to the server. And the UI conventions and interactions of the application must be as close to desktop applications as possible, reducing time spent as the user switches their thought process from desktop to web. An ideal interface for rapid data entry needs to be something that resembles a spreadsheet but has each column bound to a particular field in a database table. Although like the traditional application, the data would be listed in a simple HTML `<table>`, the data for any record in the interface would immediately become editable when clicked and saved to the server when the users press the Enter key—as is the case in most spreadsheet applications. If errors occur during the saving process due to concurrency issues in the database, this information would be dynamically displayed in the interface showing which data was in error as the errors occur. Similarly, after editing the data and pressing the Enter key, the focus would automatically move to the next record, which immediately could be edited by pressing any keyboard key, again as one expects from desktop spreadsheet applications, as shown in Figure 1.2. You can see that by using AJAX, there is no time that the user is forced to sit idle while waiting for the server to respond. Instead, the user can continue to edit data before the response from the save operation returns to the browser.

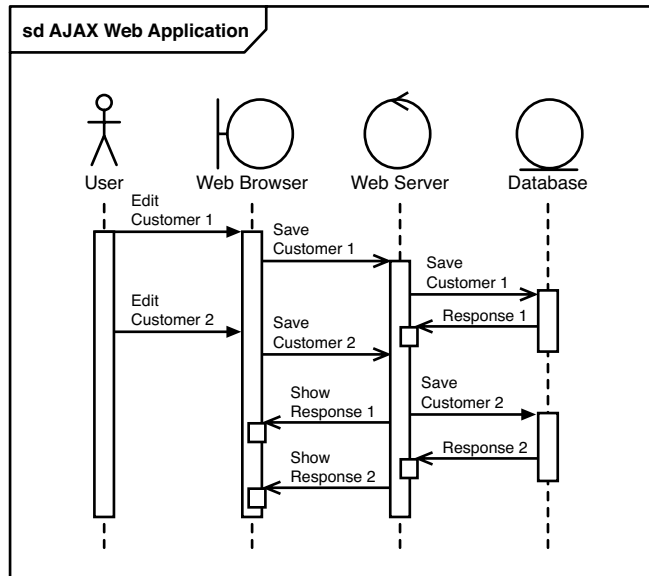



Figure 1.2 Sequence Diagram of AJAX Web Application Data Editing Workflow—The Asynchronous Nature of AJAX Enables the End User to Continue Working While the Server Processes the Requests

The key to this AJAX-based user interaction is that it is focused on sending small pieces of data, not a rendered HTML web page, to and from the server rather than a monolithic web page assembled completely by the server. This is what enables the user to edit a customer record on in the web browser without any requests to the server until the data is ready to be saved. Even then, the web page is not refreshed because only the edited data is sent to the server behind the scenes, asynchronously, using AJAX functionality.

Other hot keys also need to work in the example application, such as Ctrl + N to create a new record and Ctrl + V to paste data from either text documents or spreadsheets directly into the web application. (See Figure 1.3.) Furthermore, server-side data validation can be used so the user can get real-time feedback on the availability of a username or email address in the database and, therefore, further reduce the number of page refreshes.



ContactName	ContactEmail	JobTitle	CompanyName	PhoneNumber	Address	Country
Charles Frost	cfrost@canopy.org	Developer	WashOnline Ltd	(425) 965-2239	75 Barland Drive,	USA
Hale Hancock	hahancock@source	Project Manager	Design Johnson	(545) 237-4635	5138 Langley	USA
Tyler Bryan	tybryan@spadecol	Accountant	Silver Ridge Ltd	(120) 746-3725	1377 Harper Court,	USA
Debra Dale	ddale@bfc.ca	VP Public Relations	Magna Lion Inc	(249) 227-2723	85 Country Woods	USA
Jahana Harmon	jaharmon@demal	Media Buyer	U.S. Eagle Inc.	(335) 767-8635	2411 Santa Maria	USA
Edith Sanchez	edithsanch@ibc	Branch Manager	Volume	(88-6) 235-4885	24 Kaemisson	USA
Leia Shelton	leishelton@quack	Director of	IS Investigations	(860) 244-2632	94 Kermessid	USA
Christopher Best	chrisbest@duke	Accountant	Inter Eastern Ltd	(745) 725-8255	743 Airport Way,	USA
Quincy Conway	quincyconway@blu	Marketing	Press Byte	(920) 367-7467	5 12th Ave, New	USA
Myang Macdonald	mmacdo@supercite	Bookkeeper	Thermalbox Inc.	(479) 578-4795	65 Brentwood	USA
Chet Fisher	cfisher@bthetdata	Human Resources	Fund Products Ltd	(773) 755-5436	7 Eastbrook Drive	USA

Figure 1.3 Screenshot of an AJAX Data Grid Application Exhibiting Desktop Spreadsheet Functionality, Such as Data Selection with the Mouse-Enabling Features (Such as Data Copy and Paste)

Protecting users from themselves and the effects of networks is another benefit in usability that you can take advantage of in an AJAX architecture. It can be frustrating to spend time filling out a long HTML form only to lose your connection and then not being able to commit your actions or data entry back to the server or database. With AJAX, you can constantly send data back to server asynchronously. This also allows you to keep the server side and client side data in sync at all times. Although you wouldn't want to unnecessarily commit changes to a data base on every keystroke, you can push data up to the server or even store it locally to protect the user from losing the data due to network outages or client system problems.

AJAX in the Enterprise

Until recently, the widespread use of JavaScript was limited at best. JavaScript itself has a history of being banned in some cases from use in corporate web development because of irregular support among browsers and security concerns. The modernization of JavaScript in Firefox and Internet Explorer finally gave developers a reliable platform on which to create rich applications, and the coining of the term AJAX gave a common vernacular. A survey by BZ Research in September 2006 (see Figure 1.4) found that 18.9 percent of respondents said that their companies had deployed production systems using AJAX.¹ Another 12.1 percent said that they were developing their first AJAX production systems but haven't deployed yet, and 14.2 percent are developing pilot systems. In addition, 37.7 percent were actively researching

¹<http://www.sdtimes.com/article/story-20060901-12.html>

the technology. A mere 9.5 percent said that neither they nor their company has plans to use AJAX (7.6 percent said that they didn't know).

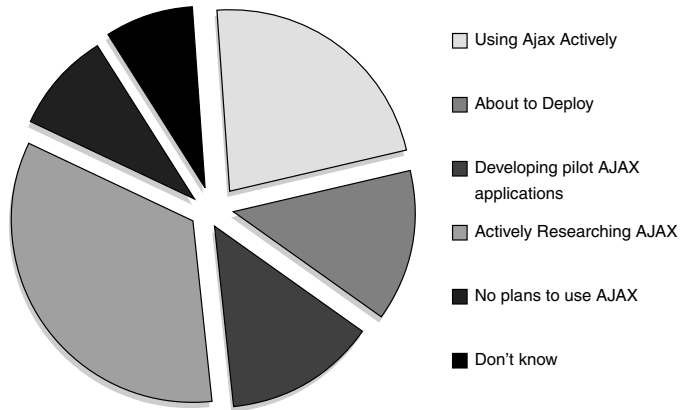


Figure 1.4 AJAX Use in the Enterprise 2006 (Source: SD Times)

Looking at the demand for qualified developers, the sheer number of new job postings related to AJAX is astounding. In Figure 1.5, you can see the growth in job postings that require AJAX skills.



Figure 1.5 AJAX Job Trends (Source www.indeed.com)

This demand is driven by organizations that feel pressure to modernize their approach to application development for sound economic reasons. These drivers include the need for greater usability, improved use of network infrastructure, and better data architectures.

Drivers for AJAX Adoption

Enterprises application development has no room for superfluous use of risky or unnecessary technology. Development is centered on helping employees do their jobs better and helping enterprises reduce costs, gain a competitive advantage, or simply to make money. Any investment in a new technology must be justified along these guidelines. As developers, you need to be mindful of the drivers for adoption of AJAX in the enterprise if you are to be successful at developing enterprise-quality software.

Usability

Although it's clear that AJAX is first and foremost a tool for user experience, we still haven't explained why this is so important. Does it actually matter if our software is nicer to use? How large or small is the user experience opportunity?

"I estimate that low intranet usability costs the world economy \$100 billion per year in lost employee productivity"—Dr. Jakob Nielsen, *Information Architecture for the World Wide Web*, Second Edition

Some of the benefits associated with good user interfaces are qualitative and difficult to measure precisely. This isn't to imply they are not of financial value, but many business benefits are hard to quantify, and seasoned IT managers know intuitively they can translate into significant bottom-line savings. When we look at streamlining a user interface, we can measure success by looking at factors like the following:

- **Steps to complete a task**—Reducing the number of steps has implications for the amount of time consumed but also for the number of opportunities for error. Fewer errors mean cost savings down the road when these errors would have to be manually corrected.

- **Benefits of a familiar user interface**—Often, Web-based applications replace desktop applications that had superior user interfaces. The benefits of offering users a similar or even a familiar user interface to what they use on the desktop means lower training costs, fewer errors, and greater out-of-the-gate productivity.
- **Improved application responsiveness**—More responsive applications can improve productivity not only by reducing “wait,” but also by promoting a more fluid, uninterrupted workflow. In a responsive application, users can move rapidly from one action to another as quickly as they can visualize the workflow. Less responsive applications can defeat the users’ workflow visualization by forcing them to continually wait for information to be displayed.
- **Impact of better information**—A rich client application provides better information to the users by giving visual feedback and hints as to the functionality of the user interface. (Buttons light up when hovered, menus reveal themselves when selected, and so on.) Forms can validate themselves against data instantly without having to wait for lengthy page refreshes, and users can see early on when mistakes have been made, *when they are made*, which helps to maintain a linear workflow and avoid mistakes.
- **Direct data visualization**—Offloading much of the data process to the client along with client-side technologies such as Dynamic HTML and Scaling Vector Graphics means that data can be visualized in new dynamic and intuitive ways reducing the conceptual work for the end user of the application.
- **Support volume**—If usability has increased, there should be an impact on the number of support requests for the impacted applications. This is something that IT managers need to watch closely and use as a barometer of user-interface usability.

Usability is often referred to as a reason for AJAX adoption but rarely well defined. When evaluating AJAX application usability, it is important to look at the issue both quantitatively and qualitatively and compare results to traditional web applications. Improved usability in an AJAX application reveals itself quantitatively through reduced user error or increased productivity and qualitatively through user feedback and preferences.

Productivity in economic terms is generally a measurement of output for hours worked. So, if you can increase the output of workers in large

enterprises, there's clearly value in the usability investments. Enterprise work forces spend a significant part of their time using web-based applications, which means that improving these tools has tremendous overall benefits. The productivity gains from an AJAX UI can be significant. In applications with high data throughput where hundreds or thousands of employees are performing data entry on a daily or hourly basis, clear quantitative measurements can be made about productivity. Using a web application with an AJAX UI can certainly save several seconds per mouse click removing costly web page refreshes that can quickly add up when applied to every step in a transaction across an entire business every day.

By thinking about a few simple user interactions with enterprise web applications, we follow some general guidelines throughout this book that can result in improved usability of your web applications.

Fire and Forget

The most basic thing to consider in the interests of usability is the power of asynchronous server interactions. Asynchronous interactions mean that the user can interact with the application and continue to achieve workflow goals at the same time as the server is dealing with previous actions, such as persisting edited data to the database. We call this fire and forget because the request to the server is sent, and application control is *immediately* returned to the user who can then proceed to work. When the response from the server is finally received by the web browser, the application can then ensure that no errors occurred on the server, and the user is not required to take any action—although, they might see some small change in the UI to indicate that the operation was successful. In the rare cases where an error occurs on the server, the user will be interrupted and notified of the error so that they can take action to correct it.

These asynchronous requests to the server occur behind the scenes through JavaScript, which is equally important because it enables the application to fire and forget requests and removes the need for costly page refreshes. Quite simply, AJAX enables faster and more responsive UIs, so users spend less time waiting and more time working.

In addition to client-side efficiency improvements, we show you how using AJAX server requests can improve server performance through caching, and much of the application business logic can even be moved to JavaScript code in the web browser, further reducing server workload. The knock on effect is that there is also reduced network traffic and ultimately

latency for the end user. A great example of this effect is how Macrumors.com used AJAX to reduce bandwidth by a factor of six and needed only half as many servers.²

Virtual Desktop

A major benefit of the AJAX web application over a standard desktop application is that it's built from the beginning to consume data and not documents; this data can come from a central web server or an external web service. AJAX applications revolve around accessing and displaying data. Unlike desktop applications, which are typically confined to local data storage or use the network access occasionally for updates, AJAX-based applications are consumers of web accessible data. Using the browser to orchestrate and consume data from multiple sources in one interface is powerful and opens a whole new range of applications that we explore later. We have already seen this shift in how we consume and sometimes produce web pages versus documents. Now, we have the ability to push further and consume and update different types of data merely with the browser.

In the corporate setting, giving the users the impression that the data is actually on their local computer is relevant in many scenarios. In some instances, there is no choice, such as in many financial systems where live market and trading data needs to be delivered to the client in real time. Currently, many of these applications are delivered through heavier technologies that are not as platform-independent or as easy to distribute as an AJAX web app through a web browser.

Giving the user the impression that the data is right on their desktop goes a long way to improving the application speed and, therefore, the users productivity and efficiency.

Context Switching

AJAX brings to the web interactivity and efficiencies we've become accustomed to in the desktop environment. We have already mentioned how a large difference between AJAX and traditional web applications is that AJAX applications often carry over desktop user interaction patterns to the

²<http://www.macrumors.com/events/mwssf2006-stats.php>

web application. The result of this is that users find web applications more usable due to simple time-saving mechanisms such as copying data from a desktop spreadsheet to a web-based spreadsheet and that when switching between using a desktop application and a web application through the course of a day, the users need not change their mental model of the application in terms of what actions they can and cannot take. Other key factors when considering how to reduce the amount of time spent transforming from a desktop to a web approach to working include keyboard shortcuts, mouse movements, and general UI design patterns. Developers need to consider these factors when building web applications with high usability in mind.

At the same time, we need to be aware that other techniques are achievable through AJAX that, with some user training, can actually be more efficient for end users. Some of those use cases include operations such as drag and drop where providing better affordances to users to indicate what objects can be dragged and where they can be dropped can go a long way in improving application usability.

Network Utilization

In addition to the qualitative user experience (UX) concerns, we can look to quantitative metrics to evaluate the potential for cost savings. A recent study on Developer.com³ found that AJAX had the potential to reduce the number of bytes transferred over the network by 73 percent, and total transmission time by 32 percent. In a sample application, users experienced the following measurable benefits:

- **Reduced time spent waiting for data to be transmitted**—Time is money. Over many repetitions, the time employees spend waiting for the page to load can add up to significant costs.
- **Time spent completing a particular task**—Increased efficiency in the user interface can often mean that time is saved at the task level, offering opportunities for concrete cost savings.

³<http://www.developer.com/xml/article.php/3554271>