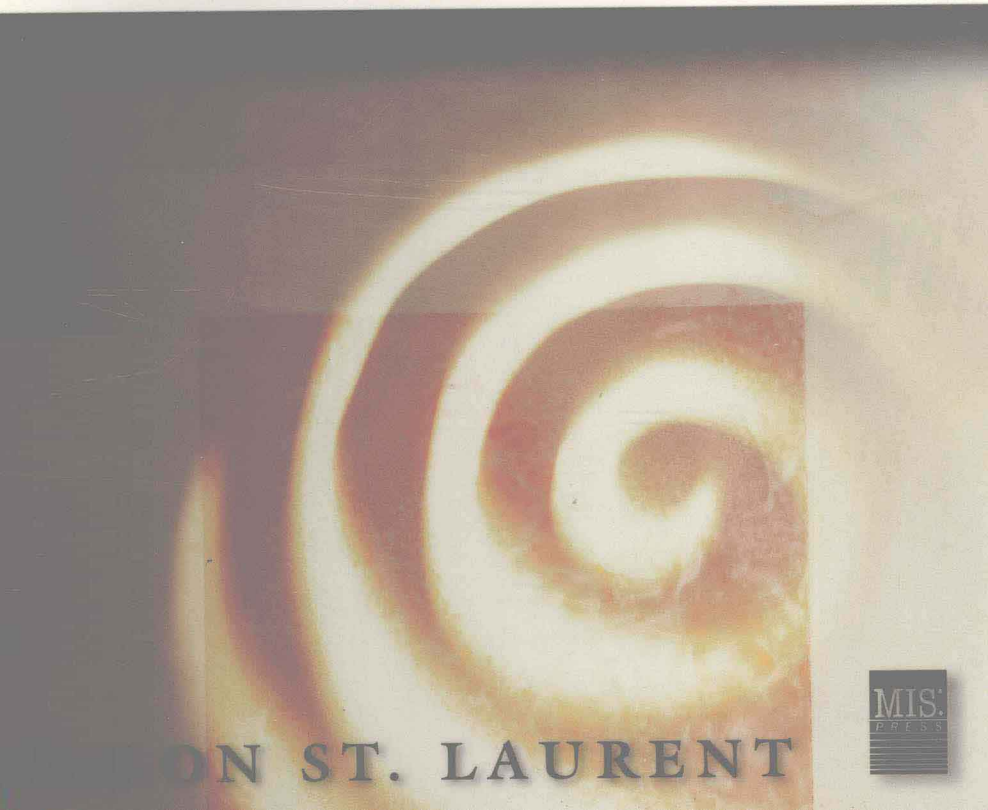
DYNAMIC HTML

A PRIMER



ON ST. LAURENT

MIS: PRESS

DYNAMIC HTML: A PRIMER

Simon St. Laurent

A Subsidiary of
Henry Holt and Co., Inc.

MIS:Press
A Subsidiary of Henry Holt and Company, Inc.
115 West 18th Street
New York, New York 10011
http://www.mispress.com

Limits of Liability and Disclaimer of Warranty

The Author and Publisher of this book have used their best efforts in preparing the book and the programs contained in it. These efforts include the development, research, and testing of the theories and programs to determine their effectiveness.

The Author and Publisher make no warranty of any kind, expressed or implied, with regard to these programs or the documentation contained in this book. The Author and Publisher shall not be liable in any event for incidental or consequential damages in connection with, or arising out of, the furnishing, performance, or use of these programs.

First Edition—1997

Library of Congress Cataloging-in-Publication Data

St Laurent, Simon
Dynamic HTML : A Primer / by Simon St. Laurent
p. cm.
ISBN 1-55828-569-5
1. HTML (Document markup language)
QA76.76.H94S7 1997
005.7'2—dc21
97-28076
CIP

MIS:Press and M&T Books are available at special discounts for bulk purchases for sales promotions, premiums, and fundraising. Special editions or book excerpts can also be created to specification.

For details contact: Special Sales Director
MIS:Press and M&T Books
Subsidiaries of Henry Holt and Company, Inc.
115 West 18th Street
New York, New York 10011

10 9 8 7 6 5 4 3 2 1

Associate Publisher: *Paul Farrell*

Managing Editor: *Shari Chappell*
Editor: *Ann C. Lush*
Copy Edit Manager: *Karen Tongish*
Production Editor: *Gay Nichols*
Technical Editor: *Bebo White*
Copy Editor: *Betsy Hardinger*

DYNAMIC HTML:

A PRIMER

For Tracey, who makes my sweetest dreams come true

Acknowledgements

I would love to thank Tracey Cranston for being her usual amazing self and helping me stay calm when it didn't seem like anything worked. My parents, who've helped me finance too many of my computing ventures for the last 15 years, provided constant support and encouragement.

Michael Sprague got me started with this book, converting me quite successfully from a Dynamic HTML skeptic to a thorough believer. Ann Lush did a great job of keeping me on track, listening patiently to my complaints and always pointing out the bright side of authoring. Bebo White provided great feedback on the book and the examples, and hopefully some of his comments have taken root in the pages that follow. I'd also like to thank Paul Farrell and Nettie Aljian at MIS:Press for teaching me a lot about the publishing business and the strange but exciting world of computer books.

INTRODUCTION

Break It Down to Make It Stronger

When HTML was created in 1991, it dramatically changed the way people access information. By providing an intelligible, easy-to-learn standard for hypertext documents, HTML and the other World Wide Web standards made it possible for ordinary people to create complicated hypertext documents and distribute them easily. Although HTML tags were often cryptic, learning HTML wasn't much harder than learning a word processor, at least at first. The structures are simple, even elegant, and it's not difficult to create readable documents that provide a lot of information. Over time, HTML documents acquired new design frills and a measure of interactivity, but the underlying structure didn't change very much.

It's time for a second revolution. Although HTML gives designers a basic set of structures, the structures lack flexibility. Once an HTML document is loaded, it can't be changed. Applets and plug-ins provide interactivity, but the basic HTML code can't change its appearance in response to user needs. Although HTML has come a long way in designability, it's time for it to move forward in programmability.

The Web is on the verge of a change nearly as dramatic as the appearance of the first graphical browsers. One of its fundamental elements is about to be broken and

rebuilt to be stronger, better, and faster. The browser will finally achieve full status as an interface, much as Netscape has hyped it recently. Interactive documents—documents that easily transform parts of themselves in "live" ways that seem as transparent as the traditional desktop GUI—are on the way. Best of all, you don't have to be a Java expert or C++ guru to work with them.

The source of all this promise is Dynamic HTML. A proposed standard developed by the W3 committee (http://www.w3.org), Dynamic HTML was first implemented in Microsoft Internet Explorer 4. The basic concept is simple: break the document into containers of information that can be separately addressed and modified—live. The document is no longer static, because Dynamic HTML allows scripts, applets, and ActiveX objects to modify the HTML code directly. The browser will keep up with changes to the HTML and will redisplay the document without having to go back to the server or rebuild the page from scratch. HTML is no longer just a way to present information; instead, it has become an interface that gives users a much more powerful way to interact with the information. HTML developers will be able to build powerful interfaces without needing to resort to heavy-duty Java or ActiveX programming, although both kinds of programming will have their uses in conjunction with Dynamic HTML.

Like every other Web standard, Dynamic HTML is affected by the browser wars. After Netscape and Microsoft began submitting their varying standards to open committees, including members of both sides, a few false prophets predicted the end of the browser wars. But the companies will probably always find new things to fight about. In this battle, I think Microsoft has much more to offer, but I hope to give you a fair perspective on the advantages and disadvantages of the models and tools offered by both sides. Revolutions have a way of

fragmenting, and computing revolutions have proven to be no purer than any other. Despite the tumult, new tools are emerging that will give developers (including ordinary Web creators) an easy way to create rich documents with much more depth and programmability.

One word of warning: taking advantage of these tools will require some knowledge of scripting. I've tried to write this book to benefit HTML developers as well as programmers and scripters, but advanced techniques inevitably require advanced knowledge. I've included information along the way to help non-scripters make immediate use of the power of Dynamic HTML, but as the book progresses there's simply too much to explain. If you've programmed in anything resembling C (for JavaScript) or Basic (for VBScript), you'll probably be fine and can skip some of the sections devoted to elementary programming concepts. If you have no programming experience, read this book as far as you can. Try the examples and make small changes. Then visit your local bookstore (or the Netscape and Microsoft Web sites) for a good tutorial on scripting. I suspect you'll be better off learning JavaScript. Most of this book uses JavaScript for the scripting, although I've also provided some illustrations in VBScript as well as one long project that's based on a previously developed Basic program. I have serious doubts about the viability of Visual Basic, but if you're working in a Microsoft shop you may be able to ignore them.

Contents in Brief

Contents

CHAPTER 1

The Problem of Interactivity

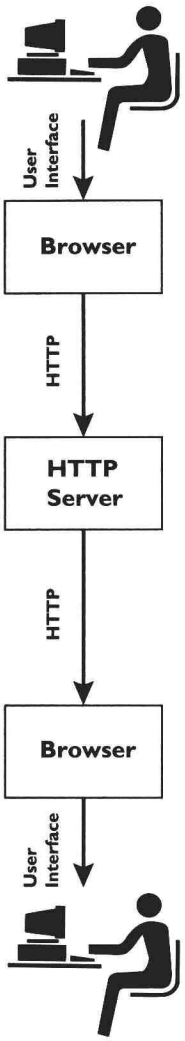
At the inception, it was possible to learn HTML and Web page design in an hour. It wasn't really programming, just a collection of mark-up tags that produced approximately the same output every time a page was opened. Almost as soon as the Web sprang into existence, users demanded a more sophisticated type of interactivity. They wanted documents capable of changing every time a page opened. To meet this need, a succession of techniques were developed for creating live pages, first on the server and eventually on the client. These approaches demand a much more sophisticated level of programming than HTML does. Some of them work on the server and others on the reader's browser, but all of them have drawbacks, primarily in ease of use and performance.

Because of the limitations of browsers, most of these solutions work on the Web server. Using them requires more than simple HTML; you need a thorough understanding of the process of retrieving Web documents. Because you'll probably need to use these techniques in conjunction with HTML, it's worthwhile to understand the process. The examples in this chapter barely begin to take advantage of the power of these techniques but should give you a grounding to help you understand how Dynamic HTML goes well beyond their capabilities.

Normally, a browser on a computer attached to the Internet sends a request to a server. The URL for a document contains three key pieces of information: the protocol to use to transfer data, the server where the data is stored, and a path to the document. If that document is a static file, the process is simple and looks like Figure 1.1. The browser interprets the URL, figures out which server to contact, and sends a request with some simple information. The server processes the request, finds the file, and sends it to the browser. If the document needs more files to display (such as graphics), the browser issues new requests for each needed file and displays the document as the files arrive.

Server-Side Includes: Parsing Documents

If a document needs to include information that changes, the situation is a little more complicated. If the changes are simple, a server-side include might do the trick. Instead of pulling up the file and transmitting it, the server pauses to examine the document and make additions as requested by tags in the document. The overall process is similar to a standard HTTP request, but the server takes a few more steps (see Figure 1.2)



The user enters a URL: **http://www.dhtml.com/index.html** in a browser or clicks on a hyperlink.

The browser interprets the URL as an HTTP request for the server at **www.dhtml.com** asking for the file **index.html**.

The browser sends an http `GET` request to the server for the file over the network.

The server receives and processes the request, pulling up **index.html** from its local file system.

The server returns an HTTP response to the browser, including headers that provide file and file-type (Content-type: text/html) information as well as the document itself.

The browser examines the response and processes the data according to the content-type. This file is HTML, so the browser formats the document, displays it, and waits for further user input.

The user can read the document and click on hyperlinks, starting the HTTP cycle again.

Figure 1.1 Retrieving an HTML document using HTTP.