

TOM NEGRINO
DORI SMITH



VISUAL
QUICKSTART
GUIDE

JAVASCRIPT

FOR THE WORLD WIDE WEB

*Teach yourself JavaScript the
quick and easy way! This
Visual QuickStart Guide uses
pictures rather than lengthy
explanations. You'll be up
and running in no time!*

2nd
EDITION

VISUAL QUICKSTART GUIDE

JAVASCRIPT

FOR THE WORLD WIDE WEB

2ND EDITION

Tom Negrino
Dori Smith

Visual QuickStart Guide

JavaScript for the World Wide Web, 2nd Edition

Tom Negrino and Dori Smith

Web site for this book: <http://www.chalcedony.com/javascript/>

Peachpit Press

1249 Eighth Street

Berkeley, CA 94710

(800) 283-9444

(510) 524-2178

(510) 524-2221 (fax)

Find us on the World Wide Web at: <http://www.peachpit.com>

Peachpit Press is a division of Addison Wesley Longman

Copyright © 1998 by Tom Negrino and Dori Smith

Editor: Nancy Davis

Production: David Van Ness

Inhouse production: Amy Changar

Cover design: The Visual Group

Notice of rights

All rights reserved. No part of this book may be reproduced or transmitted in any form or by any means, electronic, mechanical, photocopying, recording, or otherwise, without prior written permission of the publisher. For more information on getting permission for reprints and excerpts, contact Trish Booth at Peachpit Press.

Notice of liability

The information in this book is distributed on an "As is" basis, without warranty. While every precaution has been taken in the preparation of this book, neither the author nor Peachpit Press shall have any liability to any person or entity with respect to any loss or damage caused or alleged to be caused directly or indirectly by the instructions contained in this book or by the computer software and hardware products described herein.

ISBN: 0-201-69648-7

0 9 8 7 6 5

Printed and bound in the United States of America



INTRODUCTION

Welcome to JavaScript! Using this easy-to-learn programming language, you'll be able to add pizzazz to your Web pages and make them more useful for you and for your site's visitors. We've written this book as a painless introduction to JavaScript, so you don't have to be a geek or a nerd to write a script. Pocket protectors will not be necessary at any time. As a friend of ours says, "We're geeky, so you don't have to be!"

We wrote this book for you

We figure that if you're interested in JavaScript, then you've already got some experience in creating HTML pages and Web sites, and you want to take the next step and add some interactivity to your sites. We don't assume that you know anything about programming or scripting. We also don't assume that you are an HTML expert (though if you are, that's just fine). We do assume that you've got at least the basics of building Web pages down, and that you have some familiarity with common HTML like links, images, forms, and frames.

How to use this book

Throughout the book, we've used some devices that should make it easier for you to work with both the book and with JavaScript itself.

In the step-by-step instructions that make up most of the book, we've used a special type style to denote either HTML or JavaScript code, like this:

```
<BODY>
window.status
```

You'll also notice that we show the HTML in uppercase, and the JavaScript in lowercase, which makes it easier to distinguish between the two on a Web page. Whenever you see a quote mark in a JavaScript, it is always a straight quote (like ' or "), never curly quotes (aka "smart" quotes, like ' or ").

In the illustrations accompanying the step-by-step instructions, we've highlighted the part of the scripts that we're discussing in red, so you can quickly find what we're talking about. We often also highlight parts of the screen shots of Web browser windows in red, to indicate the most important part of the picture.

Because book pages are narrower than computer screens, some of the lines of JavaScript code are too long to fit on the page. When this happens, we've broken the line of code up into one or more segments, inserted this gray arrow → to indicate that it's a continued line, and indented the rest of the line. Here's an example of how we show really long lines in scripts.

```
document.write("You're running Netscape
→ Navigator, a fine JavaScript-enabled browser.")
```

Don't type that code!

Some books give you program listings, and expect you to type in the examples. We think that's way too retro for this day and age. It was tough enough for us to do all that typing, and there's no reason you should have to repeat that work. So we've prepared a companion Web site for this book, one that includes all of the scripts in the book, ready for you to just copy and paste into your own Web pages. The site also includes additional tips and scripts. If we discover any mistakes in the book that got through the editing process, we'll list the updates on the site, too. You can find our companion site at:

<http://www.chalcedony.com/javascript/>

If for some reason you do plan to type in some script examples, you might find that the examples don't seem to work, because you don't have the supporting files that we used to create the examples. For example, in an example where a sound plays when you click an on-screen button, you'll need a sound file. No problem. We've put all of those files up on the book's Web site, nicely packaged for you to download. You'll find one downloadable file per chapter, containing all of the scripts, HTML files, and the sound and image files we used in that chapter.

You can contact us via e-mail at:
javascript-vqs@chalcedony.com

Time to get started

One of the best things about JavaScript is that it's easy to start with a simple script that makes cool things happen on your Web page, then add more complicated stuff as you need it. You don't have to learn a whole book's worth of information before you can start improving your Web pages.

Of course, every journey begins with the first step, and if you've read this far, your journey into JavaScript has already begun. Thanks for joining us, and please keep your hands and feet inside the moving vehicle. And please, no flash photography.

TABLE OF CONTENTS

Introduction	ix
Chapter 1: Getting Acquainted with JavaScript	1
What JavaScript is	2
What JavaScript can do	3
JavaScript isn't Java	4
The snap-together language	5
Handling Events	8
Values and variables.	9
Assignments and comparisons	10
What tools to use?	11
Chapter 2: Start Me Up!	13
Where to put your scripts	14
Hiding scripts from old browsers	15
Putting comments in scripts	16
Alerting the user	17
Redirecting the user	18
Browser detection and conditionals	19
Plug-in detection	21
Around and around with loops.	22
Functions	25
Scrolling status bars.	27
Status bar messages	30
Chapter 3: Fun with Images	31
Creating rollovers	32
Creating more effective rollovers	34
Triggering rollovers from a link	36
Creating cycling banners.	37
Making the banner cycling wait for the user.	39
Building slide shows.	40
Displaying a random image.	42
Combining a rollover with an image map.	44
Automatically changing background colors	48

Chapter 4: Frames, Frames, and More Frames	51
Keeping a page out of a frame	52
Forcing a page into a frame.	53
Loading a frame	54
Creating and loading a dynamic frame	55
Sharing functions between frames	56
Storing information in frames	58
 Chapter 5: Verifying Forms	 59
Validating Email addresses.	60
Selecting menu items	63
Working with radio buttons	64
Setting one field with another.	66
Validating Zip codes	68
Verifying passwords	69
 Chapter 6: Working with Browser Windows	 73
Opening a new window	74
Scrolling a window	76
Bringing windows to the front	78
Updating one window from another	80
Creating new pages with JavaScript.	82
Closing a window	84
Creating a control panel	85
Displaying alerts when a window is loaded	87
 Chapter 7: Dynamically Updated Pages	 89
Putting the current date into a Web page.	90
Working with days	92
Customizing your message for the time of day.	93
Converting military time to AM/PM.	94
Creating a countdown.	98
Working with referrer pages	100
 Chapter 8: JavaScript and Cookies	 101
Baking your first cookie	103
Reading a cookie.	106
Using cookies as counters.	107
Deleting cookies	109
Handling multiple cookies	111

Chapter 9: Java and Plug-ins	113
Checking if Java is enabled	114
Determining a user's IP address	116
Getting the user's monitor size	119
Using Java to display text	121
Playing sounds using a plug-in	123
Playing a sound on a rollover	125
Chapter 10: JavaScript and Cascading Style Sheets	127
Moving an object	128
Moving CSS text (Netscape)	130
Moving CSS text (Internet Explorer)	131
Modifying a CSS drop shadow	132
Rotating a CSS shadow	133
Modifying a CSS glow	134
Chapter 11: Debugging Your JavaScript	135
Netscape's built-in debugger	136
JavaScript debuggers	137
Common errors	138
Following variables while a script is running	143
Viewing values in another window	144
Writing error messages to another window	145
Chapter 12: Where to Learn More	147
Finding Help on the Web	148
Usenet newsgroups	153
Books	154
Chapter 13: Real World JavaScript	155
Starting Out	156
Giving users feedback with JavaScript	158
Sharing a .js file	164
Appendix A: JavaScript Genealogy and Reference	169
JavaScript versions	170
ECMAScript	172
Browsers and JavaScript	173
Object flowchart	174
JavaScript object table	175

Appendix B: JavaScript Reserved Words	183
JavaScript reserved words	184
Java keywords reserved by JavaScript.	184
Additional words reserved by ECMAScript	184
Additional reserved words proposed to be used by ECMAScript	184
Other identifiers to avoid	185
Index	187

GETTING ACQUAINTED WITH JAVASCRIPT

1

For Web site creators, the evolution of HTML has been a mixed blessing. In the early days of the World Wide Web, HTML was fairly simple, and it was easy to learn most everything you needed to learn about putting together Web pages.

As the Web grew, page designers' aspirations grew as well, and their demand for greater control over the look of the page forced HTML to change and become more complex.

Because the Web is a dynamic medium, page designers also wanted their pages to interact with the user, and it soon became obvious that HTML was insufficient to handle the demand. Netscape invented JavaScript as a way to control the browser and add flash and interactivity to Web pages.

In this chapter, you'll learn what JavaScript is (and what it isn't); what it can do (and what it can't); and some of the basics of the JavaScript language.

What JavaScript is

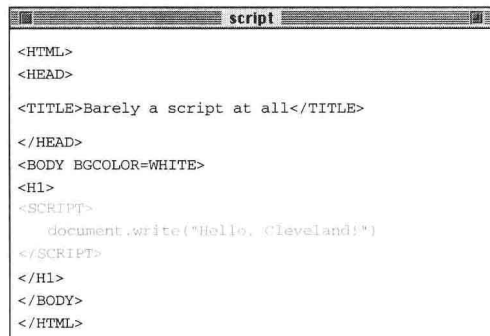
JavaScript is a programming language that you can use to add interactivity to your Web pages. But if you're not a programmer, don't panic at the term "programming language"; there are lots of JavaScripts available on the Web that you can copy and modify for your own use with a minimum of effort. In fact, standing on the shoulders of other programmers in this way is a great technique for getting comfortable with JavaScript, and if you do enough of it (and read this book), you'll soon be creating your own scripts from scratch.

To make it easier for you to get up and running with JavaScript, the authors have set up a Web site that supplements this book. We've included all the scripts in the book (so you don't have to type them in yourself!), as well as additional notes, addendums, and updates. You can find our site at <http://www.chalcedony.com/javascript/>.

You'll often see JavaScript referred to as a "scripting language," with the implication that it is somehow easier to script than to program. It's a distinction without a difference, in this case.

A JavaScript script is a program that is included on an HTML page. Because it is enclosed in the `<SCRIPT>` tag, the text of the script doesn't appear on the user's screen, and the Web browser knows to run the JavaScript program. The `<SCRIPT>` tag is most often found within the `<HEAD>` section of the HTML page, though you can, if you wish, have scripts in the `<BODY>` section. Scripts that write text to the screen or that write HTML are best put in the `<BODY>` section, as in Script 1.1. If you're unfamiliar with these HTML concepts and you need more information about HTML, we suggest that you check out Elizabeth Castro's *HTML for the World Wide Web: Visual Quickstart Guide*, also available from Peachpit Press.

Script 1.1 This very simple script types "Hello, Cleveland!" into the browser window.



```
<HTML>
<HEAD>

<TITLE>Barely a script at all</TITLE>

</HEAD>
<BODY BGCOLOR=WHITE>
<H1>
<SCRIPT>
    document.write("Hello, Cleveland!")
</SCRIPT>
</H1>
</BODY>
</HTML>
```



Figure 1.1 A rollover is an image that changes when you move the mouse pointer over it.

What JavaScript can do

There are many things that you can do with JavaScript to make your Web pages more interactive and provide your site's users with a better, more exciting experience.

JavaScript lets you create an active user interface, giving the users feedback as they navigate your pages. For example, you've probably seen sites that have buttons that highlight as you move the mouse pointer over them. That's done with JavaScript, using a technique called a rollover (Figure 1.1).

You can use JavaScript to make sure that your users enter valid information in forms, which can save time and money. If your forms require calculations, you can do them in JavaScript on the user's machine without needing to use a complex server CGI (a program that runs on a Web server and extends the server's functions).

With JavaScript, you have the ability to create custom HTML pages on the fly, depending on actions that the user takes. Let's say that you are running a travel site, and the user clicks on Hawaii as a destination. You can have the latest Hawaii travel deals appear in a new window.

JavaScript controls the browser, so you can open up new windows, display alert boxes, and put custom messages in the status bar of the browser window.

Because JavaScript has a set of date and time features, you can generate clocks, calendars, and timestamp documents.

In Netscape Navigator 3.0 and Microsoft Internet Explorer 3.0 and later, you can use JavaScript to test for the presence of browser plug-ins, and direct the user to a different page if they don't have the plug-in needed to view your page.

JavaScript isn't Java

Despite the name, JavaScript and Java have almost nothing to do with one another. Java is a full-featured programming language developed and marketed by Sun Microsystems. With Java, a descendant of the C and C++ programming languages, programmers can create entire applications and control consumer electronic devices, much as C and C++ are used. Unlike those languages, Java holds out the promise of cross-platform compatibility, that is, a programmer should be able to write one Java program that could then run on any kind of machine, whether that machine is running Windows 95 or NT, the Mac OS, or Unix. In practice, Java hasn't fully realized that dream, due in no small part to bickering between Sun and Microsoft as to the direction of the language. Microsoft got involved because they want to integrate Java into Windows in their own way (a way that Sun says would make Java work one way on Windows, and another way on other machines, thereby defeating Java's main purpose).

Java's main use is to create *applets*, small programs that download over the Internet and run inside Web browsers. Because of Java's cross-platform nature, these applets should run identically on any Java-enabled browser (a browser that has the Java engine built-in).

You embed Java applets in your Web pages using the `APPLET` HTML tag. When the browser sees the `APPLET` tag, it downloads the Java applet from the server, and the applet then runs in the area of the screen specified in the tag (Figure 1.2). When the user moves on to another Web page, the applet is flushed from the computer's memory.

JavaScript can interact with a Java applet on a Web page, as you'll see in Chapter 9. The combination of JavaScript and Java enables you to provide a powerful and interesting experience to your site's users.

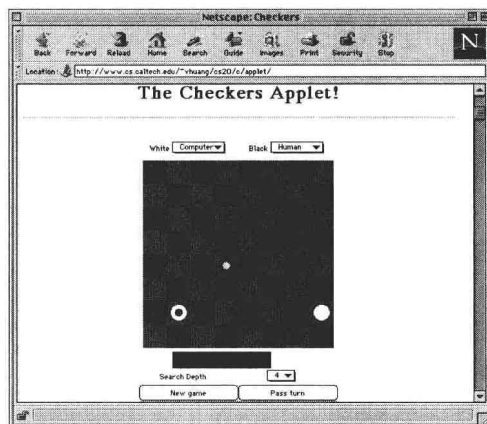


Figure 1.2 This Java applet plays a mean game of checkers, and is something you can't create with JavaScript. The applet takes up the space from the pop-up menus for player color, down to the buttons at the bottom of the screen.



Figure 1.3 The cat object (this one's name is Pixel).



Figure 1.4 This pop-up menu is a browser object, which can be manipulated by JavaScript.

The snap-together language

Here's another buzzword that we should get out of the way: JavaScript is an *object-oriented* language. So what does that mean?

Objects

First, let's think about objects. An *object* is some kind of a thing. A cat, a computer, and a bicycle are all objects (Figure 1.3) in the physical world. To JavaScript, there are objects it deals with in Web browsers, such as windows, forms, and the elements of the form, such as buttons and check boxes (Figure 1.4).

Because you can have more than one cat, or more than one window, it makes sense to give them names. While you could refer to your pets as Cat #1 and Cat #2, it's a bad idea for two reasons: first, it's easier to tell the cats apart if they have unique names, and second, it's just plain impolite. In the same way, all the examples in this book will give objects their own unique names.

✓ Tip

- Be aware that scripts you might see on the Internet will refer to objects like `window[0]` and `form[1]`. This is poor style for the reasons given above, and you'll find that it's much easier for you to keep track of the different objects in your scripts if you give them names instead of numbers.

Properties

Objects have *properties*. A cat has fur, the computer has a keyboard, and the bicycle has wheels. In the JavaScript world, a window has a title, and a form has a check box.

Properties can modify objects, and the same property can apply to completely different objects. Let's say that you have a property called `empty`. It's okay to use `empty` wherever it applies, so you could say that both the cat's tummy is empty and the cat's bowl is empty.

Note that the computer's keyboard and the bicycle's wheels aren't only properties; they are also objects in their own right, which can have their own properties. So objects can have sub-objects.

Methods

The things that objects can do are called *methods*. Cats purr, computers crash, and bicycles roll. JavaScript objects also have methods: buttons `click()`, windows `open()`, and text can be `selected()`. The parentheses signal that we're referring to a method, rather than a property.

✓ Tip

- It might help to think of objects and properties as nouns, and methods as verbs. The former are things, and the latter are actions that those things can do, or have done to them.