Quick Solutions to
Common CSS Problems

CSS
Cookbook

Christopher Schmitt
Foreword by Dan Cederholm

CSS Cookbook

Learn how to solve the *real* problems you face with CSS. This cookbook offers hundreds of practical examples for using CSS to format your web pages, and includes code samples you can use right away. You'll find exactly what you need, from the basics to complex hacks and workarounds.

Each recipe explains how to customize a solution to meet your needs, and each chapter features a sample design that showcases the topics discussed. You'll learn about the behavior of the latest browsers—including IE 8, Firefox 3, Safari 4, and Google Chrome—and how you can resolve differences in the ways they display your web pages. Arranged in a convenient format for quick reference, this third edition is a valuable companion for anyone working with CSS.

- Learn the basics, including CSS rule structure
- Work with web typography and page layout
- Create effects for images and other page elements
- Learn techniques for configuring lists, forms, and tables
- Design effective web navigation and create custom links
- Get creative by combining CSS with JavaScript
- Learn useful troubleshooting techniques
- Explore features of HTML5 and CSS3

"Christopher's fantastic cookbook will give you solutions to pretty much all of the CSS problems you'll come up against in your day-to-day web design work, saving you bags of time and frustration. This guy is one of the industry's brightest minds—he really knows his stuff."

— **Chris Mills**
Opera Software

Christopher Schmitt has been working with the Web since 1993. He is the author of several books on web design and digital imaging, including earlier editions of *CSS Cookbook*, and is a co-lead of Adobe Task Force and member of the Education Task Force for the Web Standards Project.

INTRODUCTORY   INTERMEDIATE   ADVANCED

Familiarity with HTML and web interfaces is recommended.

O'REILLY®

oreilly.com

US $49.99   CAN $62.99
ISBN: 978-0-596-15593-3

Safari Books Online

Free online edition
for 45 days with purchase of this book. Details on last page.

Praise for *CSS Cookbook*, Third Edition

"There's a lot to know about Cascading Style Sheets, but sometimes you just want a quick answer to a specific problem. In *CSS Cookbook*, Christopher Schmitt delivers clear, expert solutions to the most important CSS design tasks while also promoting web standards, demonstrating current professional techniques, and providing useful information about the latest CSS standards."

> —Dave McFarland, author of *JavaScript: The Missing Manual*

"Whether you're a seasoned web professional or creating your very first site, *CSS Cookbook* deserves a prominent place on your desk—it's a fantastic reference and an indispensable time-saver."

> —Dan Rubin, author of *Web Standards Creativity* and
> *Pro CSS Techniques*

"Using straightforward and approachable language, Christopher Schmitt's *CSS Cookbook* delves directly into the *how* of web design, offering designers practical, accessible tips for improving their work."

> —Ethan Marcotte, interactive design director at Happy Cog, and
> coauthor of *Designing with Web Standards* and *Handcrafted CSS*

CSS Cookbook

CSS Cookbook

Christopher Schmitt
foreword by Dan Cederholm

CSS Cookbook, Third Edition

by Christopher Schmitt

Printing History:

August 2004:	First Edition.
October 2006:	Second Edition.
December 2009:	Third Edition.

Table of Contents

Foreword

Any great chef will tell you that the key to creating good food is using quality ingredients. Author Christopher Schmitt has just gone shopping for you. By compiling hundreds of CSS recipes into this single book, he's giving you a one-stop shop where you can pick up the ingredients to create stylish, flexible web pages.

When I was first learning the wonders of CSS, trial and error prevailed as my primary means for discovering its creative powers: "Hmm, I'd like to turn this list into a horizontal navigation bar," or "I need to stylize the components of a form using CSS for a client." Several hours (or days) would go by after plugging in various CSS rules, removing some, and experimenting with endless combinations. This hit-or-miss approach worked (at times), and although a curious person like me may even consider it "fun," it sure ate up a lot of time in the process.

I wish I'd had this book. Instead of stumbling upon the solution for styling every element of the page, I could have just thumbed through *CSS Cookbook*, grabbed the recipe, and started baking. The guesswork would've been eliminated, and I could have instead spent my time doing what I love to do best: creating.

The modular nature of this book makes it an indispensable reference for designers and developers of any caliber. Posed with problems from how best to handle typography, links, and navigation to even entire page layouts, Christopher clearly explains not only the styles necessary to complete the task, but also the caveats that may be attached for certain browsers. By additionally explaining the helpful workarounds to everyday CSS problems, he's arming you with the critical knowledge you need to be a successful CSS designer.

For example, a recent article told of a common usability problem: when posed with a Submit button at the end of a form, some users just can't shake their double-clicking habits. The button may get clicked twice, with the results of the form getting duplicated. What to do? A solution wasn't offered in the aforementioned article. However, unsurprisingly, there's a recipe in this very book that'll solve this little problem using CSS and a dash of JavaScript.

And that's the heart of this book's purpose: real problems and the goods that will deliver real results. You've heard about how CSS will simplify your life, making pages lighter and easier to maintain. Now it's time to start *using* it, and with this book, you'll have one less excuse not to.

So, my advice is to clear off a space on your desk because *CSS Cookbook* will take up permanent residency in the corner. Hopefully for you, that spot will be easily within arm's reach.

—Dan Cederholm
Founder, SimpleBits (*http://www.simplebits.com*)
Salem, Massachusetts

Preface

Every book tells a story—even books on web design tips and techniques.

This book is about Cascading Style Sheets, or CSS as it's commonly abbreviated. CSS is a simple standardized syntax that gives designers extensive control over the presentation of their web pages and is an essential component of web design today.

Compared to 1990s-era development techniques, CSS gives web designers greater control over their designs so that they can spend less time editing and maintaining their websites. CSS also extends beyond traditional web design to designing and controlling the look of a web page when it is printed.

You don't need any special hardware or software to design web pages. The basic requirements are a computer, a modern browser such as Firefox, Safari, or Internet Explorer for Windows (to name a few), and your favorite web page editor. A web page editor can be anything from a simple text editor such as Notepad (for Windows) or TextEdit (for the Mac), to a full-fledged WYSIWYG tool such as Adobe Dreamweaver set in code view.

Now you know what the book is about. Let me tell you its story, its history.

Some would say web design officially began when Tim Berners-Lee, inventor of the World Wide Web, put together the first set of web pages. Others would say it began when the `center` tag came about due to Netscape's own extension of HTML.

Though it might seem ironic, I happen to believe that this new media really got started with books. The books that helped lead the way to the dot-com boom in the 1990s started with Lynda Weinman's first full-color book about web graphics, *Designing Web Graphics* (Pearson), which was published in January 1996, and then David Siegel's *Creating Killer Web Sites* (Hayden), published several months later that same year. These two books helped to kick off the web revolution as much as those who invented the technologies that made the Web possible.

However, the methods written in those books, although cutting edge for their time, are out of date in today's context. As I write these pages, it has been 13 years since those initial books were published; the same year Weinman's and Siegel's first books about web design came out describing how to use `font` tags, nested tables, and single-pixel GIFs was the same year CSS was first introduced.

CSS has come a long way since then. With more than 13 years of development put into it, it's only now—with the advent of Internet Explorer 8 for Windows reaching a large audience—that web designers, developers, and everyday users of browsers can use CSS2 to its intended potential.

In addition to IE8, other browsers are making their presence known, and are often ahead of Internet Explorer in supporting new features. Browsers such as Firefox, Safari, Chrome, and Opera are implementing the latest specifications of CSS3 and HTML5 as quickly as the World Wide Web Consortium (W3C) Working Groups' members are bandying them about.

If you are serious about building today's usable and cutting-edge websites, use CSS and *CSS Cookbook*, a collection of CSS-based solutions to common web design problems. Together they can help you create your own bit of web design history.

Audience

This book is for web designers and developers struggling with the problems of designing with CSS. With this book, web builders can solve common problems associated with CSS-enabled web page designs.

CSS Cookbook is ideal for people who have wanted to use CSS for web projects but have shied away from learning a new technology. If you are this type of reader, use the solutions in the book one or a few at a time. Use it as a guidebook and come back to it when you are ready or need to learn another technique or trick.

Even if you consider yourself an expert in CSS, but not in basic design knowledge, this book is useful to have next to your computer. It covers elements of design from web typography to page layouts, and even includes a chapter on designing with CSS to get you motivated.

Assumptions This Book Makes

This book makes several assumptions about you, dear reader.

One assumption is that you possess some web design or development experience either as a hobbyist, a student, or a professional.

Since *CSS Cookbook* is neither an introduction to CSS nor a book that goes into great detail on how CSS should work in browsers, people at the start of their web design or

development education might find this book a bit more challenging than a general or complete book on the theory of CSS.

If you are looking for a book that delves into such topics about the CSS specification, you should look into *CSS: The Definitive Guide*, Third Edition, by Eric A. Meyer (O'Reilly), which serves as a solid complement to this book.

If you use a program such as Adobe Dreamweaver only in its WYSIWYG or design mode and rarely, if ever, touch the markup in code view, you might have trouble getting the most out of this book right away. To get an introduction to handcoding HTML, look into *Learning Web Design* by Jennifer Niederst Robbins (O'Reilly).

Although WYSIWYG tools allow for CSS-enabled designs, some of the tools have not caught up with some of the unorthodox approaches recommended in this book and might cause some trouble if you attempt to implement them by editing solely in WYSIWYG mode.

To benefit from this book, you must be able to edit HTML and CSS by hand. Some of the code in this book can be re-created using dialog-box-driven web page building applications, but you may run into some problems along the way trying to click tabs and enter CSS values into said tabs.

Another assumption is that web designers and developers practicing their craft with HTML table-based layouts, `font` tags, and single-pixel GIFs will find this book both helpful *and* frustrating.

Web designers who are practicing or are more familiar with these old production methods are going to find CSS challenging. The "browser hell" often associated with cross-browser development still exists, as browser vendors tended to interpret the CSS specification differently or didn't implement the CSS specification completely. This frustration is a natural part of the learning process. You should approach the process of learning how to design with CSS with patience and a good sense of humor.

The good news is that the major browser vendors seem to have solved the problem. The recent version releases of browsers appear to have implemented CSS correctly; however, attempting cross-browser support for the older or less-popular browsers may still be a challenging exercise.

Yet the benefits of CSS, including greater control over the look and feel of web pages and easier maintenance over multipage websites, outweigh the hardships associated with browser hell.

A handful of solutions within this book use JavaScript and the JavaScript framework, jQuery. This book assumes that you have a general knowledge of the scripting language as well as the ability to successfully include JavaScript code into a web document.

If this is a hurdle, I recommend that you download the code from the O'Reilly website to get a firsthand look at a working example. On the other hand, if you were looking for a solution-focused book that deals with recipes where CSS plays a minor

role compared to JavaScript, that book would be *JavaScript & DHTML Cookbook* by Danny Goodman (O'Reilly).

The final assumption is that you desire a resource that provides fast answers to common CSS-based web design problems. The solutions in this book, covering everything from web-based typography to multicolumn layouts, are geared for modern browsers with version numbers later than or equal to 5, with the exception of Safari and Chrome.

Whenever possible, I mention when a technique might cause problems in modern browsers. Although there is a chapter on hacks and workarounds to hide stylesheets from browsers with poor implementations of the complete CSS specification, this book makes no assurances that you are going to create pixel-perfect designs in every browser. Even with traditional web design methods from the 1990s, this has never been the case (see *http://dowebsitesneedtolookexactlythesameineverybrowser.com/* for more information).

Contents of This Book

For me, the best use for a book such as this is to crack it open from time to time when trying to solve a particular problem, which I did with the first edition of the book to refresh my memory while writing this edition. To that end, this book will serve you well on or near your desk—always within reach to resolve a problem about CSS or web design. However, feel free to read the book from its first page to its last.

The following paragraphs review the contents of each chapter and the appendixes.

Chapter 1, *Using HTML Basics*, goes over semantic markup solutions on content.

Chapter 2, *CSS Basics*, discusses the general concepts of CSS as well as some techniques associated with best practices in development.

Chapter 3, *Web Typography*, discusses how to use CSS to specify fonts in web pages, headings, pull quotes, and indents within paragraphs as well as other solutions.

Chapter 4, *Images*, discusses CSS techniques directly associated with manipulating styles and properties related to web graphics.

Chapter 5, *Page Elements*, covers a loose collection of items that don't necessarily fit in every chapter, but that all carry a theme of affecting the design of the overall page. Solutions in this chapter cover the topics of centering elements, setting a background image, placing a border on a page, and other techniques.

Chapter 6, *Lists*, describes how to style basic list items in various ways. Solutions include cross-browser indentation, making hanging indents, inserting custom images for list markers, and more.

Chapter 7, *Links and Navigation*, shows how to use CSS to control the presentation of a link and sets of links. Solutions range from the basic, such as removing an underline from links, to the more complex, such as creating a dynamic visual menu.

Chapter 8, *Forms*, discusses how to work around the basic ways browsers render forms. You'll learn how to set styles to specific form elements, set a submit-once-only button, and style a login form, among other things.

Chapter 9, *Tables*, shows how to style HTML tables. Although CSS can help you eliminate HTML table-based designs, sometimes you may need to style tabular data such as calendars and statistical data. This chapter includes solutions for setting cell padding, removing gaps in table cells with images, and styling a calendar.

Chapter 10, *Designing Web Pages for Printing*, talks about how you can use CSS to engineer layouts. The solutions in this chapter include methods for designing one-column layouts as well as multicolumn layouts.

Chapter 11, *Page Layouts*, provides information on how to set styles that are used when printing web pages. Solutions discuss how to add a separate print stylesheet to a web page, set styles for web forms, and insert URLs after links.

Chapter 12, *Hacks, Workarounds, and Troubleshooting*, provides solutions that enable you to hide stylesheets that certain browsers cannot handle. Recipes include hiding stylesheets for browsers such as Netscape Navigator 4, Internet Explorer 5 for Windows, and others.

Chapter 13, *Designing with CSS*, is an inspirational chapter. Focusing on the notion that CSS is merely a tool that implements design, this chapter covers topics such as playing with enlarging type sizes, working with contrast, and building a panoramic presentation.

Chapter 14, *Interacting with JavaScript*, demonstrates how to use the JavaScript framework, jQuery, in conjunction with CSS for more advanced effects.

Appendix A is a collection of links and websites you can access to learn more about CSS.

Appendix B is a listing of CSS 2.1 properties that can help you define the look and feel of, or, in some cases, the sound of HTML elements on a web page.

Appendix C is a listing of selectors, pseudo-classes, and pseudo-elements available within CSS 2.1.

Appendix D is a listing of selectors and pseudo-classes available from the new CSS3 specification.

Appendix E takes a look at how various modern browsers handle the display of form elements. The print book version contains an introduction to this appendix, as well as information on how you can access the full version. The online version of this appendix contains lookup tables that allow you to quickly check out which CSS properties are supported, as well as the entire form element review that contains screenshots of every test.

Conventions Used in This Book

The following typographical conventions are used in this book:

Italic

> Indicates new terms, URLs, email addresses, filenames, file extensions, pathnames, directories, and Unix utilities

`Constant width`

> Indicates commands, options, switches, variables, attributes, keys, functions, types, classes, namespaces, methods, modules, properties, parameters, values, objects, events, event handlers, XML tags, HTML tags, macros, the contents of files, or the output from commands

`Constant width bold`

> Shows commands or other text that should be typed literally by the user

`Constant width italic`

> Shows text that should be replaced with user-supplied values

> This icon signifies a tip, suggestion, or general note.
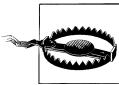
> This icon indicates a warning or caution.

Using Code Examples

This book is here to help you get your job done. In general, you may use the code in this book in your web pages and design. You do not need to contact us for permission unless you're reproducing a significant portion of the code. For example, writing a program that uses several chunks of code from this book does not require permission. Selling or distributing a CD-ROM of examples from O'Reilly books does require permission. Answering a question by citing this book and quoting example code does not require permission. Incorporating a significant amount of example code from this book into your product's documentation does require permission.

We appreciate, but do not require, attribution. An attribution usually includes the title, author, publisher, and ISBN. For example: "*CSS Cookbook*, Third Edition, by Christopher Schmitt. Copyright 2010 O'Reilly Media, Inc., 978-0-596-15593-3."

If you feel your use of code examples falls outside fair use or the permission given here, feel free to contact us at *permissions@oreilly.com*.

We'd Like to Hear from You

Please address comments and questions concerning this book to the publisher:

O'Reilly Media, Inc.
1005 Gravenstein Highway North
Sebastopol, CA 95472
800-998-9938 (in the United States or Canada)
707-829-0515 (international or local)
707-829-0104 (fax)

We have a web page for this book, where we list errata, examples, and any additional information. You can access this page at:

http://www.oreilly.com/catalog/9780596155933

This book also has another website:

http://csscookbook.com

To comment or ask technical questions about this book, send email to:

bookquestions@oreilly.com

For more information about our books, conferences, Resource Centers, and the O'Reilly Network, see our website at:

http://www.oreilly.com

Safari® Books Online

Safari Books Online is an on-demand digital library that lets you easily search over 7,500 technology and creative reference books and videos to find the answers you need quickly.

With a subscription, you can read any page and watch any video from our library online. Read books on your cell phone and mobile devices. Access new titles before they are available for print, and get exclusive access to manuscripts in development and post feedback for the authors. Copy and paste code samples, organize your favorites, download chapters, bookmark key sections, create notes, print out pages, and benefit from tons of other time-saving features.

O'Reilly Media has uploaded this book to the Safari Books Online service. To have full digital access to this book and others on similar topics from O'Reilly and other publishers, sign up for free at *http://my.safaribooksonline.com*.

Acknowledgments

First, thanks to David Siegel and Lynda Weinman for their inspiration and support from the beginning of web design.

I wouldn't be writing any books for an industry I love so very much without the support and friendship of Molly Holzschlag.

I'd like to acknowledge my appreciation and respect for the following fellow web builders for pushing CSS-enabled web designs forward: Douglas Bowman, Tantek Çelik, Dan Cederholm, Mike Davidson, Ethan Marcotte, Eric A. Meyer, Mark Newhouse, Dave Shea, Nicole Sullivan, Stephanie Sullivan, and Jeffrey Zeldman.

Special thanks go to the technical editors, Opera Web Evangelist Bruce Lawson, Shelley Powers, and Edd Dumbill, as well as copyeditor Audrey Doyle, for their time, expertise, and patience.

Special thanks also go to Tatiana Diaz, my editor for the previous edition of this book.

Simon St.Laurent took over for Tatiana in the role of editor for this edition. His calm demeanor and ability to guide this book through the production process made the metallic bladelike swooshing sounds of deadlines bearable.

Thanks to my friends who know me as the web geek I truly am, and who are OK with me not mentioning them all by name.

Thanks to Jessica, who made me a chocolate cake with homemade chocolate icing and chocolate chips to celebrate my birthday and the release of the previous edition. I enjoyed it immensely, and my dentist appreciated the extra work. I'm not expecting another cake, but I did put you in my acknowledgments.

Thanks to my family for their love and appreciation. Your support through good times and bad has been a rock. As always, I'm looking forward to our next reunion.

Thanks to Ari Stiles for being OK with me taking time out to work on this book. I love you.

And to my dad, I dedicate this book once again. Thanks for being the best dad ever.

—Christopher Schmitt
Fall 2009
http://christopherschmitt.com/
http://twitter.com/teleject

Using HTML Basics

1.0 Introduction

Using CSS effectively requires using HTML effectively. To set you on the right path with HTML, this chapter runs through the basics of using HTML well. You'll explore basic but critical techniques for creating an HTML page, validating the markup to make sure it's free of any typos and errors, and taking advantage of new possibilities for adding video and audio with HTML5.

> If you feel you're an old hand at this, feel free to skim the chapter. Even a review of the chapter should help you build some good habits that will ease your work.

Structuring Documents

To build a design for your web pages, first there must be content in a web document, usually a simple text file. That content within a text file needs to be tagged with what is called *HyperText Markup Language*, more commonly referred to as *HTML*. HTML provides *structure* to documents through the use of *elements*.

When you wrap these elements with tags, such as p for paragraphs and h2 for headings, throughout the content, the web page starts to form an inherent HTML document structure.

The browser then applies its own stylesheet to render what is known as the default rendering of the web page onto this document structure.

This default look and feel won't win any design awards. It's a starting point that allows the *presentation* or design to be associated through Cascading Style Sheets (CSS) and JavaScript more cleanly to provide appearance and movement to the web page.

Semantic Markup

This chapter is a primer on how to code semantic HTML. Semantic markup is the "radical" notion that we use the appropriate HTML element for its respective content.

For example, to denote a paragraph, we use the simple p tag at the beginning and end of the paragraph text:

```
<p>Lorem ipsum dolor sit amet, consectetuer adipiscing elit, sed diam nonummy
nibh euismod tincidunt ut laoreet dolore magna aliquam erat volutpat.</p>
```

Avoiding Old-Tag Soup

The semantic approach to HTML isn't common on the Web. Since various HTML elements look different when they appear in a browser, web designers occasionally brew often-strange concoctions of HTML elements into what is commonly referred to as *tag soup* to achieve the desired look and feel.

To gain control of this look and feel, designers might add presentational HTML tags to otherwise semantically marked-up content, like so:

```
<p><font face="Georgia, Times, serif" size="+2">Lorem ipsum dolor</font>
<font face="Arial, Helvetica, sans-serif" size="1">sit amet, consectetuer adipiscing
elit, sed diam <b>nonummy</b> nibh euismod tincidunt ut laoreet dolore magna
aliquam erat volutpat.</font></p>
```

Those additional HTML tags are there to control the look and feel of just *one* paragraph.

When you use traditional HTML coding, every single element in a site's HTML page would therefore need to be coded with additional elements to create the specific colors, fonts, alignment, and layout that a designer wants; such a process is extremely tedious and prone to errors.

Imagine you were designing a website that consisted of 20 pages, and you wanted to add certain design elements such as colors, fonts, sizing, and alignment to the site. Now imagine maintaining a 1,000-page website. How about a 1,000,000-page website?

A site's HTML documents quickly become bogged down with additional code that makes both the content and the code all but unmanageable.

HTML Is Document Structure

So, it's important to get the document structure right as much as possible with HTML. Through the use of semantic, lean coding, web developers save time in terms of maintenance while also allowing the framework on which stylesheets can be applied.

> If you feel knowledgeable enough about HTML and HTML5 already, the information in this chapter might already be in your domain. If that's the case, you might want to skip through this chapter.

1.1 Picking a Text Editor

Problem

You want to choose a text editor for marking up content with HTML.

Solution

Numerous software applications are geared toward coding HTML. Some are free and some require payment.

Some basic text editors that come preinstalled with operating systems include:

- Notepad (Windows OS)
- TextEdit (Mac OS)
- gedit (Linux OS)

Here are some other free text editors that have more features:

- Notepad++ (Windows OS; *http://notepad-plus.sourceforge.net/uk/site.htm*)
- TextWrangler (Mac OS; *http://www.barebones.com/products/TextWrangler/*)
- jEdit (Windows OS, Mac OS, and Linux OS; *http://www.jedit.org/*)

For more professional-level, commercial integrated development environments (IDEs), try one of the following:

- Adobe Dreamweaver (Windows OS and Mac OS; *http://www.adobe.com/products/ dreamweaver/*)
- Panic Software's Coda (Mac OS; *http://www.panic.com/coda/*)

Discussion

For editing HTML, some applications come bundled with common operating systems such as Mac OS X and Windows. They are TextEdit and Notepad, respectively.
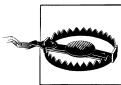
> Do not use word processing programs for working with HTML. Although these programs are ideal for creating common documents that you need to print, they add extraneous formatting to your text that you don't want or need.

Before using TextEdit, go to File→Preferences and check "Plain text" as the format option. Otherwise, the text editor might strip out the HTML elements.

If you use Notepad, select Format→WordWrap. This option allows long lines to be wrapped within the application window, making it easier to edit.

For both TextEdit and Notepad, make sure to save the HTML file with an *.html* file extension. Do not append an additional *.html* extension onto the file. For example, *example.txt.html* or *example.html.txt* only leads to heartbreak.

Even though these code editors—which are free and already installed in the operating system—do not offer many options, many web designers rely on them for working with HTML.

More robust, still free

Another text editing option that is also free is TextWrangler from Bare Bones Software. TextWrangler is not as full-featured as the company's flagship product, BBEdit, but it might suit your needs just the same. TextWrangler and BBEdit are Mac-only applications.

For Windows, there are options such as Notepad++ and TextPad (see *http://www.text pad.com/*).

If you use Unix, there are the vi/vim and Emacs editors. Another potential text editor is jEdit, which is also available for Mac and Windows.

IDE solutions

More full-featured products often cost more, but they provide a complete solution for dealing with almost every aspect of building websites. Popular products in this realm include Adobe Dreamweaver and Panic Software's Coda.

See Also

http://www.notepad.org/logo.htm, to get a "Made with Notepad" graphical banner to place on your web page

1.2 Coding a Basic HTML Page

Problem

You want to create your first HTML page.

Solution

Start with basic content, such as the following:

```
My Basic Web Page
Epsum factorial non deposit quid pro quo hic escorol.
```

Next, add an `html` element around the entire document:

```
<html>
My Basic Web Page
Epsum factorial non deposit quid pro quo hic escorol.
</html>
```

Then place the `head` and `body` elements in the document, like so:

```
<html>
 <head>
 </head>
 <body>
My Basic Web Page
Epsum factorial non deposit quid pro quo hic escorol.
 </body>
</html>
```

Insert a `title` element in the `head` element:

```
<html>
 <head>
  <title>CSS Cookbook</title>
 </head>
 <body>
My Basic Web Page
Sed quis custodiet ipsos custodes?
 </body>
</html>
```

The heading (`h1`) and paragraph (`p`) elements go inside the `body` element, and the page should render as shown in Figure 1-1:

```
<html>
 <head>
  <title>CSS Cookbook</title>
 </head>
 <body>
  <h1>My Basic Web Page</h1>
  <p>Sed quis custodiet ipsos custodes?</p>
 </body>
</html>
```

Discussion

Every web page needs to have an HTML element wrapping the entire document. Within each HTML element are two required elements: `head` and `body`.

The `head` element contains the information about the document, often called meta information. The `head` element needs to have the `title` element within it. This text is usually set in the top portion of the browser window and is used when creating bookmarks. It's important to be concise and to avoid long descriptions when using the `title` tag.
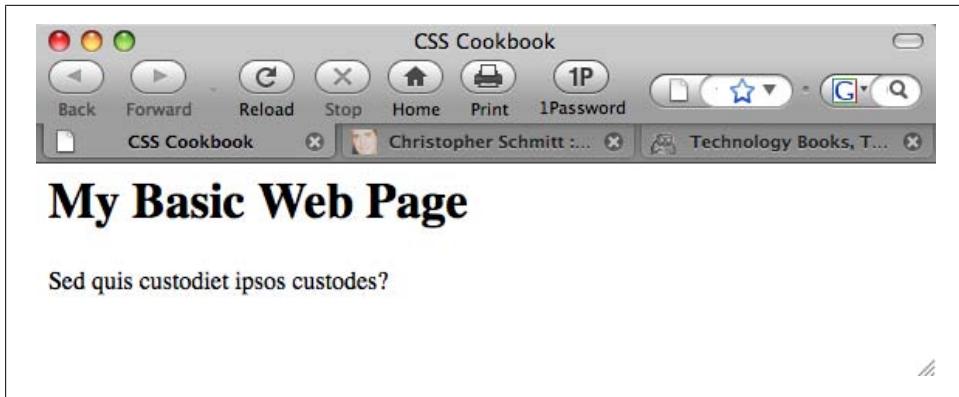
Figure 1-1. The default rendering of a basic HTML web page

If the `title` element contains no text, browsers will use either the filename or the first few words of the document instead.

> Only text is allowed within the `title` element. Other HTML elements aren't allowed.

The content of a web document is placed within the `body` element. If you need to edit or revise a web page, most of the time it is within this element.

For this example, the heading was set with an `h1` element along with the standard `p` element for the paragraph.

See Also

Recipe 1.1 for choosing a text editor

1.3 Understanding DOCTYPEs and Effects on Browser Layout

Problem

You want to make your web page standards compliant and valid.

Solution

HTML 4.01 has three document types: *Strict*, *Transitional*, and *Frameset*.

Both HTML5 and XHTML 1.1 have one document type, but XHTML 1.0, like HTML 4.01, has three.

Only *one* document type definition (DTD) appears in the HTML document. Use any one of the following DOCTYPEs that best fits your project needs.

HTML 4.01 Strict DTD:

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01//EN"
"http://www.w3.org/TR/html4/strict.dtd">
```

HTML 4.01 Transitional DTD:

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
"http://www.w3.org/TR/1999/REC-html401-19991224/loose.dtd">
```

HTML 4.01 Frameset DTD:

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Frameset//EN"
" http://www.w3.org/TR/1999/REC-html401-19991224/frameset.dtd">
```

HTML5 DTD:

```
<!DOCTYPE html>
```

XHTML 1.0 Strict DTD:

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
```

XHTML 1.0 Transitional DTD:

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
```

XHTML 1.0 Frameset DTD:

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Frameset//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-frameset.dtd">
```

XHTML 1.1 DTD:

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.1//EN"
"http://www.w3.org/TR/xhtml11/DTD/xhtml11.dtd">
```

Here's a basic page with the HTML5 DTD and the required `head`, `body`, and `html` elements:

```
<!DOCTYPE html>
<html>
 <head>
  <title>CSS Cookbook</title>
 </head>
 <body>
  <h1>My Basic Web Page</h1>
  <p>Epsum factorial non deposit quid pro quo hic escorol.</p>
 </body>
</html>
```

Discussion

A DOCTYPE, short for *document type definition*, defines an HTML or XHTML document's building blocks and tells the browsers and validators which version of HTML or XHTML your document uses.

The DOCTYPE declaration must appear at the beginning of every web page document, before the `html` element, to ensure that your markup and CSS are standards compliant and that browsers handle the pages based on the appropriate DTDs.

Quirks mode

XHTML requires a valid DOCTYPE at the top of the document; otherwise, the pages won't validate and the browsers will fall back into what is known as *quirks mode*.

Quirks mode occurs when a browser treats a web page as "buggy." As a result, such pages are treated as though they were written in invalid markup, and therefore will be improperly rendered in modern browsers even if the XHTML and CSS are coded perfectly.

A web page that is without a DOCTYPE, with an older DOCTYPE, or with a typo-riddled DOCTYPE triggers quirks mode. So, when coding pages, make sure to check that the DOCTYPE is both added to the page and typed correctly to ensure that browsers do not render pages in quirks mode.

> If a web page has an HTML5 DOCTYPE, modern browsers will trigger standards mode, even though the actual markup isn't coded with HTML5 elements. Internet Explorer for Windows 6 and 7 ignores HTML5 features.

Figures 1-2 and 1-3 show how a table contained within a `div` with a width of 100% goes into quirks mode in Internet Explorer 6, and how the page should look in standards mode.

Why not use the latest DOCTYPE?

Using newer DOCTYPEs such as HTML5 is an option. However, it's not the only option. Unlike software application releases, newer DOCTYPEs don't make older DOCTYPEs moot.

For example, you would be hard-pressed to install, much less run, Photoshop 4 on today's computers. However, you can still use HTML4 syntax and DOCTYPEs without fear of browsers *not* rendering your content.
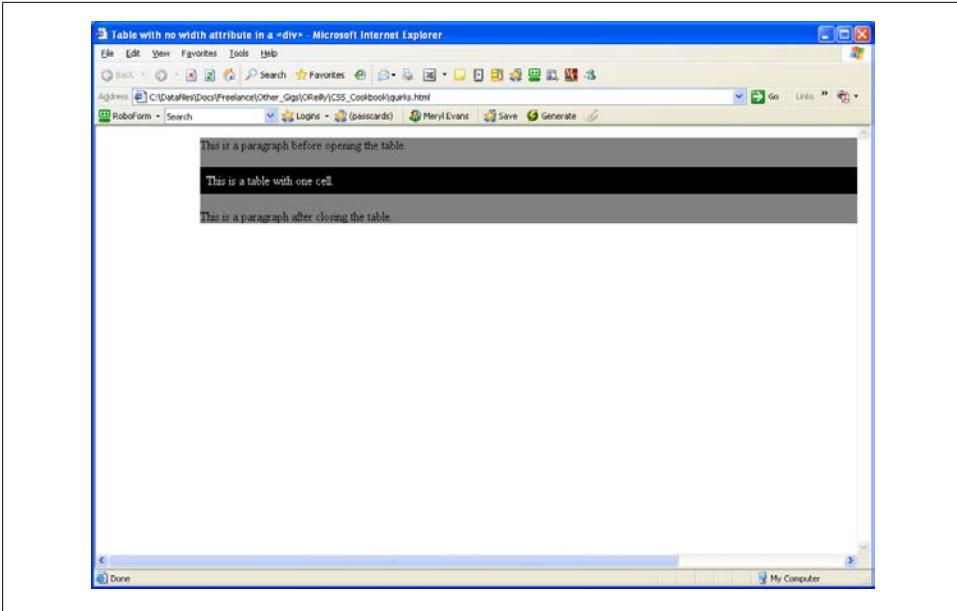
Figure 1-2. Table width in Internet Explorer 6 in quirks mode with no DOCTYPE included
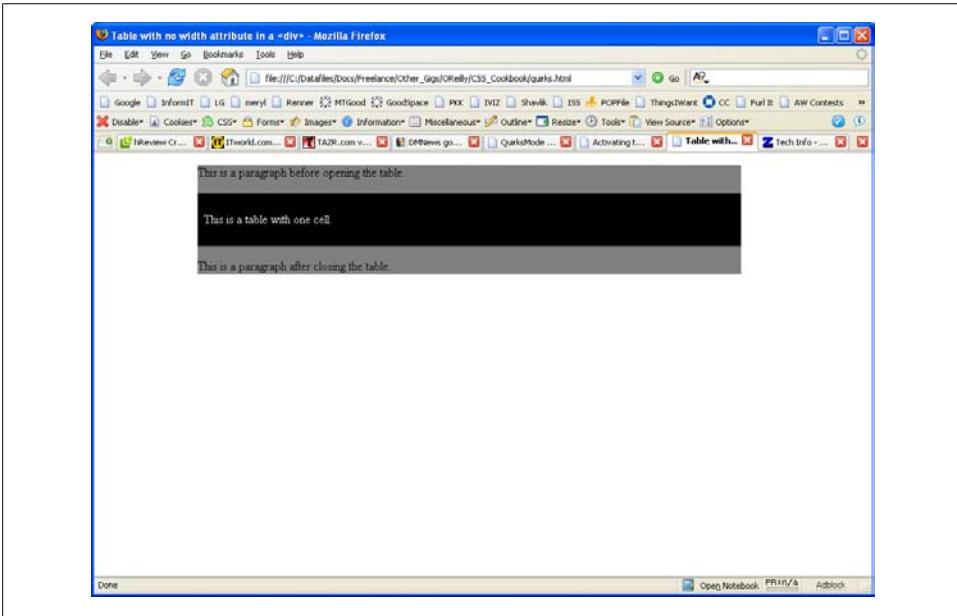
Figure 1-3. Table width in Firefox 1.5 in standards mode with HTML 4.01 Strict DOCTYPE

The smallest web page ever

The Solution provides an example of a relatively short HTML5 page. However, an even shorter *and* valid example can be made:

```
<!DOCTYPE html>
<title>Small HTML5</title>
<p>Hello world</p
```

These three HTML elements validate for HTML5 by checking out the page at *http://validator.w3.org/check?uri=http%3A%2F%2Fjsbin.com%2Fowata&ss=1*.

See Also

HTML5 specification for DTD at *http://dev.w3.org/html5/spec/Overview.html#the-doctype*; HTML 4.01 specification for DTD at *http://www.w3.org/TR/html401/intro/sgmltut.html#h-3.3*; W3C validators at *http://www.w3.org/QA/Tools/#validators*; DOCTYPEs article from A List Apart at *http://www.alistapart.com/articles/doctype/*; Article from QuirksMode at *http://www.quirksmode.org/index.html?css/quirksmode.html*; Mozilla's information on quirks mode, which explains the differences between the rendering modes and how it handles quirks mode, at *https://developer.mozilla.org/en/Mozilla's_Quirks_Mode*; Opera's DOCTYPE page at *http://www.opera.com/docs/specs/doctype/*

1.4 Marking Up Headers

Problem

You want to differentiate the importance of headings within the same document.

Solution

Use one of the six available headings, h1 through h6, as shown in Figure 1-4:

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html>
 <head>
  <title>CSS Cookbook</title>
 </head>
 <body>
  <h1>My Basic Web Page</h1>
  <p>Epsum factorial non deposit quid pro quo hic escorol.</p>

  <h2>Secondary Heading</h2>
  <p>Feles mala! cur cista non uteris? stramentum novum in ea posui.</p>

  <h3>Tertiary Heading</h3>
  <p>Por scientie, musica, sport etc., li tot Europa usa li sam
  vocabularium.</p>
```

```
<h4>Quaternary Heading</h4>
<p>Lex clavatoris designati rescindenda est.</p>

<h5>Quinary Heading</h5>
<p>Ire fortiter quo nemo ante iit.</p>

<h6>Senary Heading</h6>
<p>Interdum feror cupidine partium magnarum europe vincendarum.</p>

  </body>
</html>
```
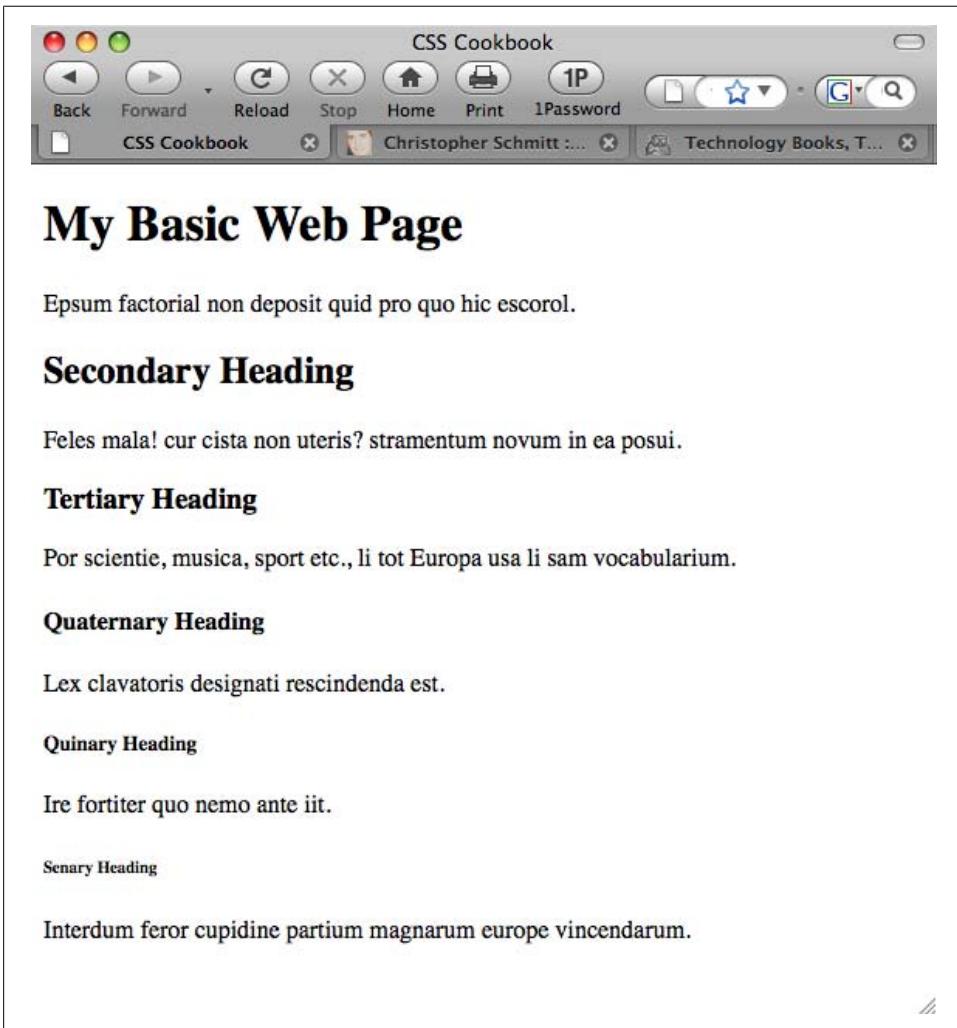

Figure 1-4. The default rendering of six heading levels

Discussion

You can choose from among six different levels of headings when marking up titles for a document.

When marking up content, be sure to use the headings in order. For example, if you use the h2 element, the header underneath it should be wrapped in the h3 element (not h4 or h5). The title of the page should not be wrapped in the h2 element (use the h1 element). In short, don't skip header tags!

It's not important to use *all* of the headers when creating a document. However, be sure not to overuse the h1 element, as that might lower your search engine ranking. Use the h1 element once for the unique title of your blog post or page; then use h2 and h3 for the other portions of the document.

> If you need to use h4, h5, and h6 elements in your document, break up the content into separate pages or investigate the document structure. A document requiring six different heading levels might be so loaded down with content that it will fail to hold an average person's attention span.

Also, if you are concerned about the look of the headings, do not worry. Through the power of CSS, the design of the headings (along with the rest of the page) can be modified.

> Using headers appropriately in a document benefits people using screen readers. For a demonstration, see the video at *http://www.youtube.com/watch?v=AmUPhEVWu_E*.

See Also

Chapter 3 for modifying headers and other common type treatments

1.5 Making Appropriate Quotations

Problem

You want to cite quotations with HTML, as shown in Figure 1-5.

Solution

Use the blockquote element when quoting a large amount of text:

```
<blockquote cite="http://www.example.com/">
  <p>Si fallatis officium, quaestor infitias eat se quicquam scire de factis
```

```
    vestris.</p>
  </blockquote>
```
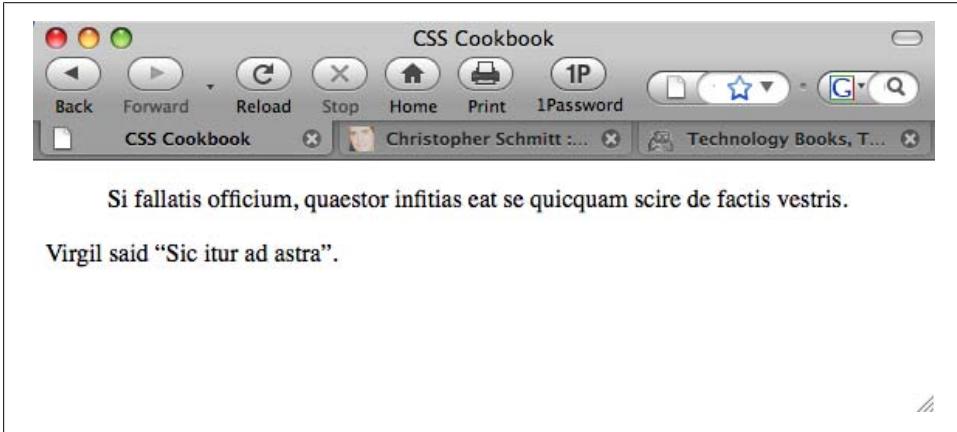

Figure 1-5. The default rendering of quotations

For citing phrases, use the q element:

```
<p>Virgil said <q>Sic itur ad astra</q>.</p>
```

Discussion

The `blockquote` element is a *block-level* element. This means that text tagged with a `blockquote` element separates itself from the rest of the text by forcing a line break above and below itself.

The `q` element is an *inline element*, which does not force a line break. Inline elements are useful for quoting small portions of text within a paragraph element.

> The `q` element is typically rendered with quotation marks around the text it envelops. However, these quotation marks do not appear in Internet Explorer for Windows.

The `cite` attribute is optional for both the `blockquote` and `q` elements. The value of a `cite` attribute is a URI where the source of the quote originated.

See Also

Chapter 3 for other common type treatments

1.6 Adding an Image

Problem

You want to add an image to a web page, as shown in Figure 1-6.
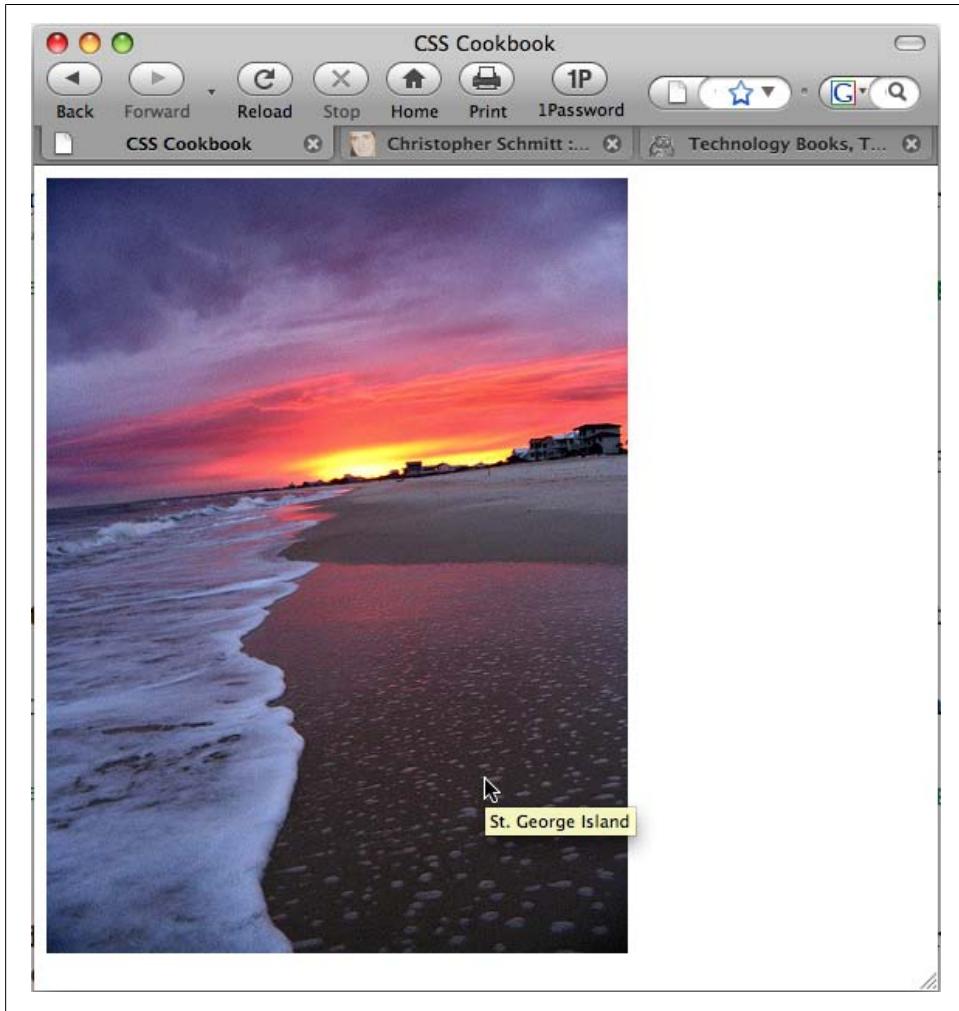
Figure 1-6. An image placed within a document

Solution

Use the `img` element to specify the location of the image file:

```
<img src="saint-george-island.jpg" />
```

Add an `alt` attribute within the `img` element to provide alternative text in case images are turned off or people are surfing with an assistive technology such as a screen reader:

```
<img src="saint-george-island.jpg" alt="St. George Island beach at sunset
with an approaching storm"  />
```

Discussion

The `img` element does not address content within the web document itself. It merely defines the location of its placement within the document and specifies its location relative to the HTML document.

Additional tips

Even though a picture is worth a thousand words, the value of the `alt` attribute should be a relatively short description.

As shown in Figure 1-6, some browsers display text next to a cursor, called a tool tip, within the `title` attribute of an image:

```
<img src="saint-george-island.jpg" alt="St. George Island beach at sunset
with an approaching storm" title="St. George Island beach" />
```

File formats

Common image formats supported by browsers include GIF and JPEG. Both formats have their own pros and cons in terms of which types of images are best for each.

Based on the *compression scheme*, which is the method with which an image's file size is reduced, GIFs are better at areas of flat color and fewer gradients, and JPEGs are good for photos and subtle color changes.

All browsers support the PNG file format; however, alpha transparency is only now supported in Internet Explorer 8 for Windows. Alpha transparency allows for opacity or levels of transparency within an image, unlike the GIF format, which can assign only one color to be transparent. If an older version of IE renders a PNG image with alpha transparency, the transparent portions usually turn into blocks of solid white.

Character case sensitivity

When specifying an image file within HTML, make sure the filename does not contain spaces and the lower- and uppercase characters match. Although your computer OS might be OK with a difference in cases, chances are the web server hosting your web files will not, and may keep images from appearing in the browser.

See Also

Chapter 4 for designing web pages with images

1.7 Adding Audio with HTML5

Problem

You want to add audio to a web page with HTML5.

Solution

Use the `audio` element to specify an audio file, as shown in Figure 1-7:

```
<!DOCTYPE html>
<html lang="en">
 <head>
  <meta charset="utf-8">
  <title>CSS Cookbook</title>
 </head>
 <body>
  <h1>Audio Example</h1>
   <audio src="html5audio.ogg" autoplay controls>
    <a href="html5test.ogg">Download audio</a>
   </audio>
 </body>
</html>
```


Figure 1-7. Audio added to a web page

Discussion

The `audio` element has five attributes associated with it: `src`, `autobuffer`, `autoplay`, `loop`, and `controls`. If you don't have the `controls` attribute, the audio player disappears.

Audio compatibility

At the time of this writing, no one audio file type plays across all the browsers that support the HTML5 `audio` element, as shown in Table 1-1.

Table 1-1. Audio file format support in HTML5

	Firefox 3.5	Safari 4	Chrome 3 beta	Opera 10
Ogg Vorbis	Y		Y	
MP3		Y	Y	
WAV	Y	Y		Y

To create a cross-browser solution, use the `audio` element along with the `source` element that cites both OGG and MP3 files. Then include Flash Player `embed` and `object` code afterward:

```
<audio controls autobuffer>
  <source src="html5audio.ogg" />
  <source src="html5audio.mp3" />
  <!-- include Adobe Flash player EMBED and OBJECT code here -->
</audio>
```

> If you do insert audio, setting the file to `autoplay` is not recommended, as it interferes with the experience for web surfers using screen readers.

See Also

Recipe 1.8 for adding video to web pages

1.8 Incorporating Video with HTML5

Problem

You want to add video to HTML5.

Solution

Use the HTML5 `video` element, as shown in Figure 1-8:

```
<!DOCTYPE html>
<html lang="en">
 <head>
  <meta charset="utf-8">
  <title>CSS Cookbook</title>
 </head>
 <body>
  <h1>Video Example</h1>
```

```
      <video src="html5video.ogg" width="320" height="240"
   controls poster="html5video.jpg">
       <a href="html5video.ogg">Download movie</a>
     </video>
    </body>
   </html>
```
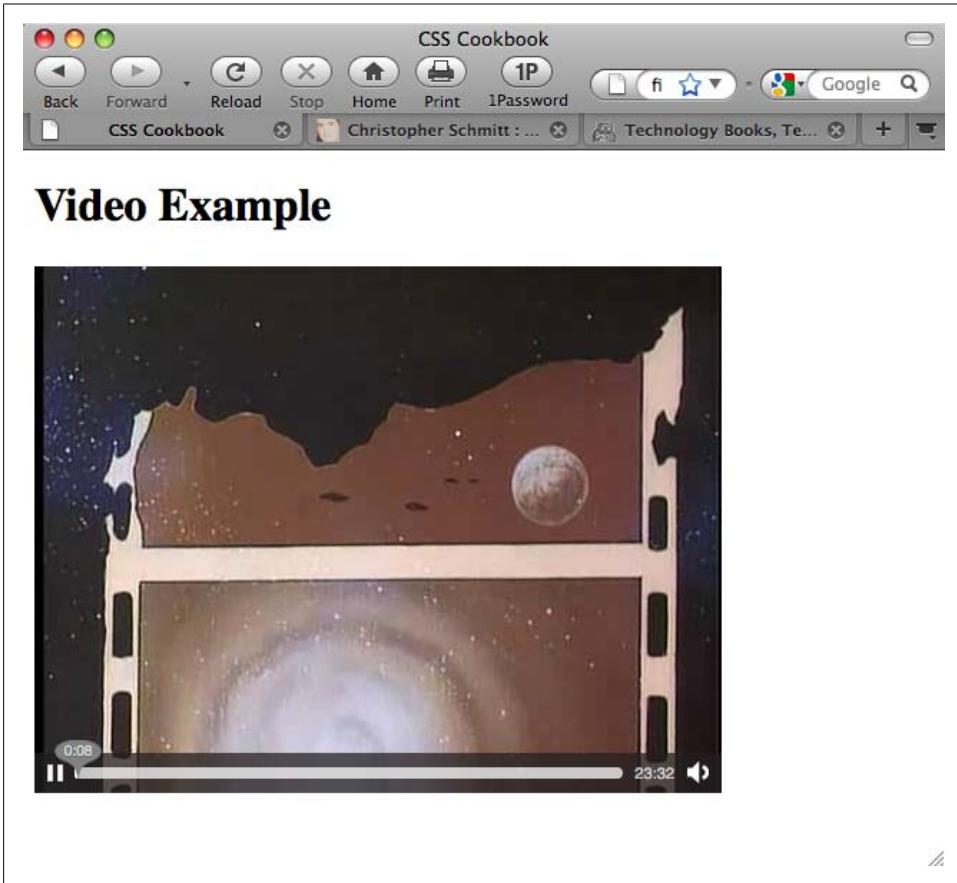

Figure 1-8. Video added to a web page

Discussion

You do not have to specify the width and height of the video element. If you do not set the video element with its respective attributes, the movie will play to the default values of the video file itself.

A video file might have its own *poster*, which is a static image that represents the video as a whole, similar to a thumbnail. However, you can override this poster by using the poster attribute. The poster image can be any file type the browser supports (e.g., GIF, JPEG, or PNG).

Although the `controls` attribute is optional, for the sake of usability I suggest using it so as not to offend your site's visitors.

You can place alternative text in between the `video` tags, including a link to download the video file, for browsers that do not recognize the `video` element. This method allows website visitors a method to view the content with third-party solutions other than browsers.

At the time of this writing, Safari 3.1 and later, Firefox 3.5 and later, Opera 10 beta, and Chrome 3 beta support the `video` element.

See Also

http://www.videolan.org/ for information on the export tools in the VLC software application, which you can use to convert common video files to OGG format (supported by Firefox and Opera)

1.9 Using strong and em Effectively

Problem

You want to emphasize certain words or phrases in a paragraph, as shown in Figure 1-9.
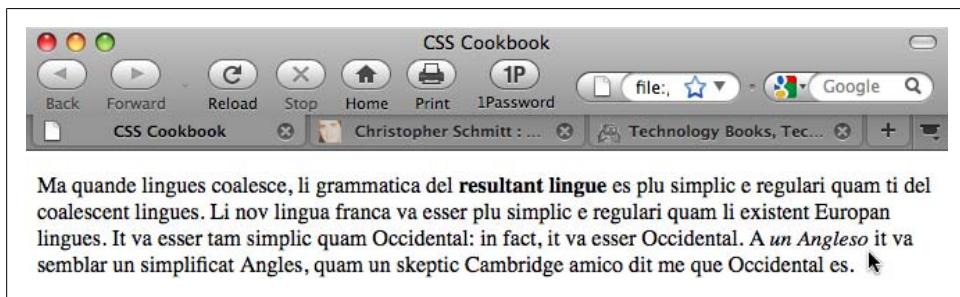
Figure 1-9. The default rendering of highlighted text

Solution

Use the `strong` and `em` elements to denote emphasis within a document:

```
<p>Ma quande lingues coalesce, li grammatica del <strong>resultant lingue</strong>
es plu simplic e regulari quam ti del coalescent lingues. Li nov lingua franca
va esser plu simplic e regulari quam li existent Europan lingues. It va esser
tam simplic quam Occidental: in fact, it va esser Occidental. A <em>un
Angleso</em> it va semblar un simplificat Angles, quam un skeptic Cambridge amico
dit me que Occidental es.</p>
```

Discussion

The `strong` element's default rendering is to make text bold, while the `em` element sets text in italics.

You would use `em` to draw attention to or contrast one or more words from the rest of a sentence. For example:

- *Darth Vader* translates loosely as *Dark Father* in Dutch.
- There are, not 57, but *50* states in the United States of America.
- If you join him, he will *complete* your training.

`Strong` is an alternative element to `em` to bring attention to words or phrases.

Although the use of `em` and `strong` helps to break up the monotony of text, be sure to use these elements sparingly as well as consistently so that you do not overuse or abuse their importance.

See Also

Chapter 3 for other common type treatments

1.10 Creating Lists

Problem

You want to create a list of items within a web page, as shown in Figure 1-10.
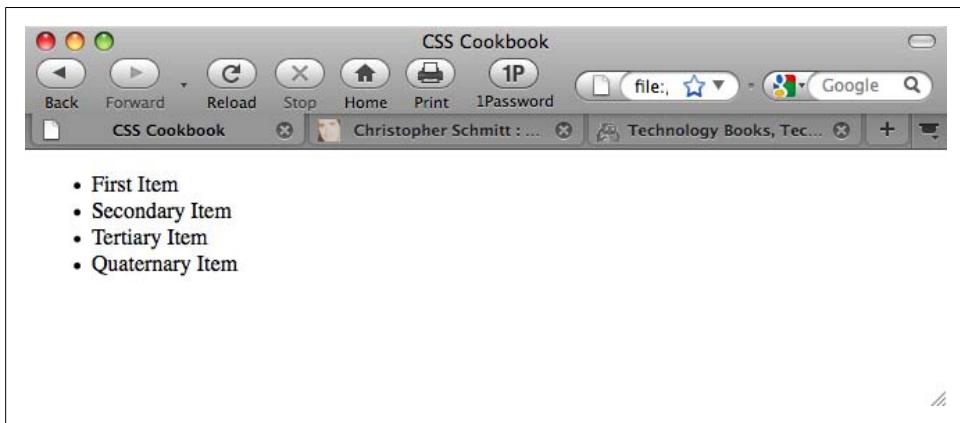
Figure 1-10. The default rendering of an unordered list

Solution

Use the `ul` element to wrap around a list of phrases:

```
<ul>
 First Item
 Secondary Item
 Tertiary Item
 Quaternary Item
</ul>
```

Then use the `li` element to wrap around each item within the list:

```
<ul>
<li>First Item</li>
<li>Secondary Item</li>
<li>Tertiary Item</li>
<li>Quaternary Item</li>
</ul>
```

Discussion

There are three types of lists in HTML: *unordered*, *ordered*, and *definition* lists.

Marking up unordered lists and ordered lists is fairly straightforward. Use two elements, `ul` and `li`, to mark up a series of items for an unordered list, which typically results in a circle appended to the left side of each list item.

 An unordered list is typically used to create the base of a navigation menu.

Ordered lists, which use an `ol` element instead of a `ul` element, have a numeral in sequential order prepended to the list.

As shown in Figure 1-11, definition lists, which are used to define terms, work a little bit differently from unordered and ordered lists. Each item is broken down into two parts: the term (`dt`) and the definition (`dd`).

```
<dl>
<dt>First Term</dt>
<dd>Seriatim</dd>
<dt>Secondary Term</dt>
<dd>Sequentia</dd>
<dt>Tertiary Term<dt>
<dd>Sequens mirabitur aetas</dd>
</dl>
```
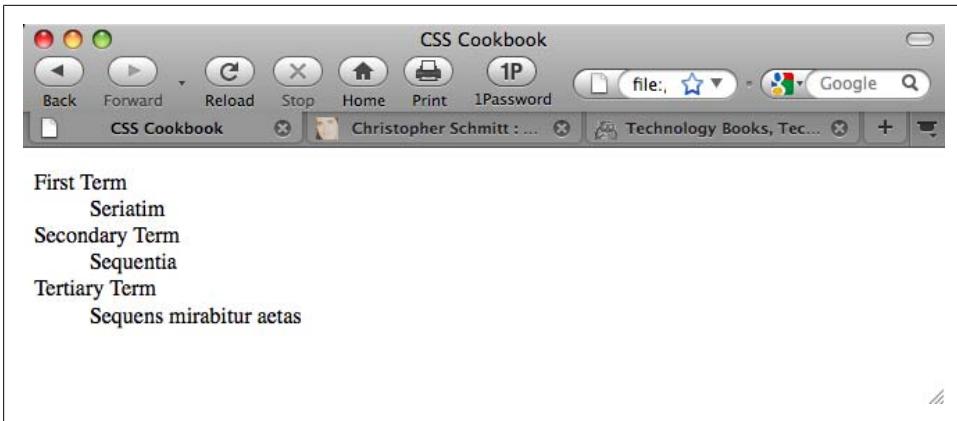
Figure 1-11. The default rendering of a definition list

See Also

Chapter 6 on lists and Chapter 7 on links and navigation

1.11 Making a Link to a Web Page

Problem

You want to link to another web page.

Solution

Using the anchor link:

```
<p>This book's <a href="http://www.csscookbook.com/">Web site</a> contains
links to download more materials.</p>
```

to link to another page in the same website, link to its file:

```
<p>Check out the <a href="about.html">About page</a> for more information.</p>
```

Discussion

Along with the `img` element (see Recipe 1.6), some browsers display a tool tip if a title attribute and value are present within the anchor link, as shown in Figure 1-12:

```
<p>This book's <a href="http://www.csscookbook.com/" title="Link to the book
site">Web site</a> contains links to download more material.</p>
```
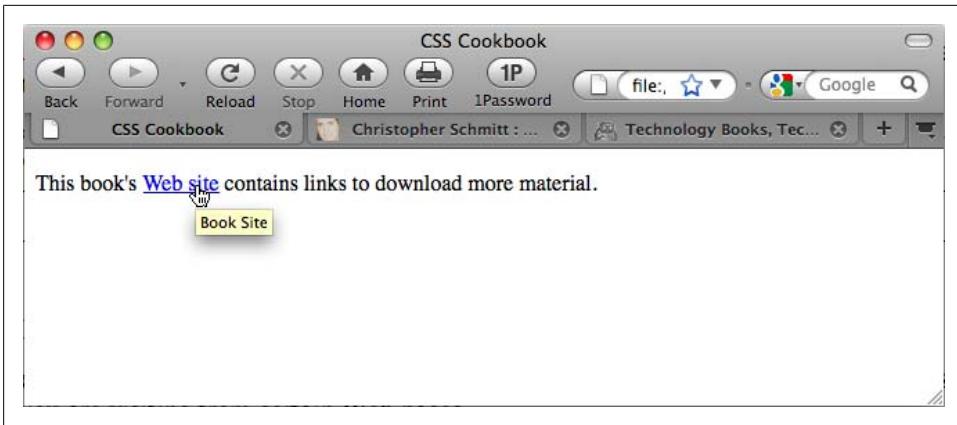
Figure 1-12. A tool tip displayed over a link

Linking to another web page on the same site

When you are creating links within the same site, use *relative links* instead of anchor links. Relative links are addresses that are valid only if you are visiting from certain web pages.

For example, suppose you have a website composed of four pages within the same *root folder*, the main directory that contains the website files, as shown in Figure 1-13:

- *httpdocs/*
  - —*index.html*
  - —*aboutus.html*
  - —*contactus.html*
  - —*services.html*
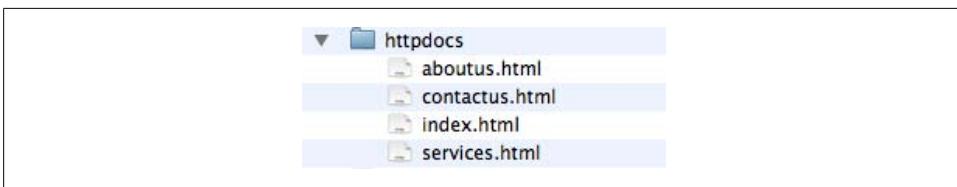
Figure 1-13. Sample directory structure

Including everything that is needed to point a web browser to a location in a link means that you created an *absolute link*, which looks like this:

```
<a href="http://www.csscookbook.com/services.html">Services Page</a>
```

If you want to create a link from the index page to another page on the same website, use a *relative link*. A relative link is a little bit leaner than an absolute link and, as in this example, can cite just the filename itself within the `href` attribute:

```
<a href="services.html">Services Page</a>
```

Relative links contain neither the full *http://* protocol nor the domain name.

When a browser navigates to a relative link, it uses the domain name of the page it is currently viewing to assemble the link to where it should go next.

Moving up folders

Just as your personal computer probably contains numerous folders holding numerous files for a project, websites are also composed of folder sets and files. To link from one document to another document within the same website, use relative links.

For example, say you have a main technical specs page within a *specs* folder, which itself is in a *widget* folder. The organization of the files on the server might look something like this:

- *products/*
  —*widget/*
    —*specs/*
      —*specs.html*

To provide a link to the main widget page from the technical specs page, use `../` to tell the browser to go up to the parent directory:

```
<a href="../widget.html">Widget Page</a>
```

If you want to go up *two* parent directories and link to the main products page from the technical specs page, you would format the link like so:

```
<a href="../../products.html">Product Page</a>
```

Using the root relative link

The process for using relative links to move between the folders of a large website can sometimes be tricky, if not convoluted. Another type of link to use in such a case is a *root relative* link.

Here is how you would use a root relative link to code the link from the technical specs page to the main product page in the preceding example:

```
<a href="/products/products.html">Product Page</a>
```

The forward slash signifies the protocol and domain name of the URI, a sort of short-hand for links.

Linking to certain elements within a web page

You can also link to certain elements within an HTML document by creating anchors. You can create an anchor by assigning an `id` attribute to an HTML element:

```
<h2 id="hireme">Hire Me</h2>
```

Then, *link* to that anchor by prefacing the `id` name with a hash symbol (#):

```
<a href="#hireme">Hire Me</a>
```

When clicked, the browser navigates to the part of the document that has the corresponding `id` name.

> If a document is *not* longer than the browser's viewport or window, there won't be any noticeable change that the browser has skipped to an anchored link.

Designers use anchors to create a table of contents at the top of a web page that lets you quickly navigate to other parts of the document. This approach is particularly useful on web pages with a large amount of content to help users avoid excessive scrolling.

See Also

Chapter 7 on links and navigation

1.12 Coding Tables

Problem

You want to create a simple HTML table, as shown in Figure 1-14.
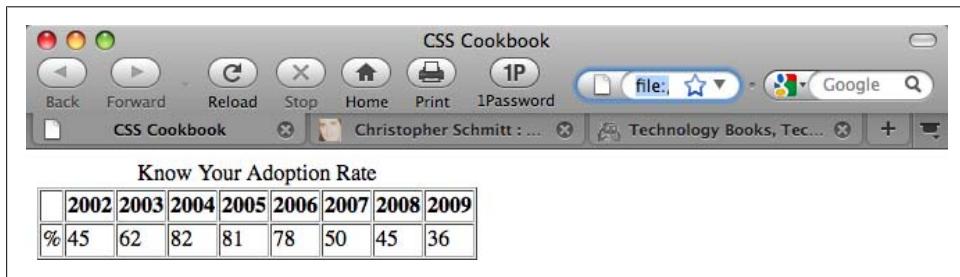
Figure 1-14. The default rendering of a basic HTML table

Solution

Use specific elements related to marking up tabular data:

```
<table border="1" cellspacing="1" cellpadding="1">
  <caption>
    Know Your IE6 Adoption Rate
  </caption>
  <tr>
    <th> </th>
    <th>2002</th>
    <th>2003</th>
    <th>2004</th>
    <th>2005</th>
    <th>2006</th>
    <th>2007</th>
    <th>2008</th>
    <th>2009</th>
  </tr>
  <tr>
    <td>%</td>
    <td>45</td>
    <td>62</td>
    <td>82</td>
    <td>81</td>
    <td>78</td>
    <td>50</td>
    <td>45</td>
    <td>36</td>
  </tr>
</table>
```

Discussion

First, add a `table` tag at the beginning and end of the tabular data. The `table` tag defines the table as a whole.

The optional `caption` element is for the summary of the tabular data and appears immediately after the opening `table` element.

Then, if your table has a header, add the `thead` tag to one or more rows as the table header. Use the `tbody` tag to wrap the table body so that it is distinct from the table header.

Next, add `tr` table row tags to mark off each table row. This element wraps groups of individual table cells. First you define a row, and then you add the enclosed cells.

> No tag exists for a table column. Only through building successive table rows do columns emerge.

After that, use the `th` tag for each cell you want to designate as a table header cell, which includes years and percentages in the Solution. You should enclose the specific cell content in the tag. By default, browsers make the text in header cells boldface.

Use the `td` tag to mark out individual cells in a table. Like the `th` tag, the `td` tag wraps specific cell content.

> For a simple, web-based HTML table generator to bypass handcrafting numerous table cells, try *http://www.askthecssguy.com/kotatsu/index .html*.

See Also

Chapter 9 on tables

1.13 Creating an HTML vCard (hCard)

Problem

You want to include in a web page contact information such as that found on a business card, as shown in Figure 1-15.

Figure 1-15. The default rendering of an hCard

Solution

Use `class` attributes with specific attributes listed in the hCard microformat specification (see *http://microformats.org/wiki/hcard*):

```
<div class="vcard">
  <span class="fn n">Josiah Bartlet</span>
```

```
    <div class="org">White House</div>
    <div class="adr">
      <div class="street-address">1600 Pennsylvania Avenue NW</div>
      <span class="locality">Washington</span>,
      <span class="region">DC</span>,
      <span class="postal-code">20500</span>
    </div>
</div>
```
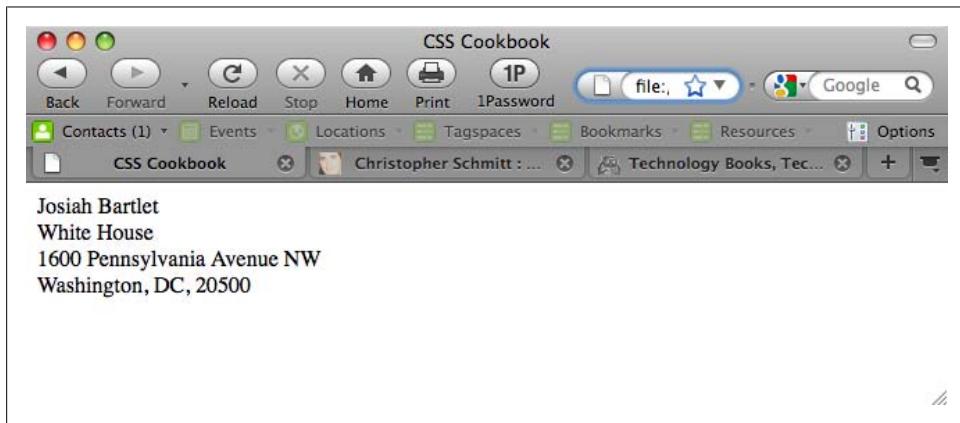
Discussion

The hCard microformat gives you a way to represent contact information, including people, organizations, and places, using XHTML class attributes. It is one of many standards detailed in the Microformats Project (see *http://microformats.org/*), the aim of which is to provide standards for coding machine-readable information into web pages using semantic HTML. Similar to a design pattern, an hCard standardizes the way in which information is represented, which allows third-party software to glean the information and put it to all kinds of good uses.

To save time and avoid typos, use the hCard Creator (see *http://microformats.org/code/hcard/creator*) to generate the HTML syntax.

Extending hCards

The H2VX web service (see *http://http://h2vx.com/vcf/*), which is available to use on the site and as a favelet, crawls the markup within a web page looking for hCard data from a web address. If it finds an hCard or hCards, it prompts the site visitor to download the data as a vCard.

The site visitor can then import the vCard into his favorite address book application, such as Outlook (Windows) or Address Book (Mac OS X).

Operator (see *https://addons.mozilla.org/en-US/firefox/addon/4106*) is a Firefox add-on that detects microformatted text on a web page and then provides you with options to do various things with the data, depending on the type of microformat used.

A similar plug-in is available for Safari at *http://zappatic.net/safarimicroformats/*.

See Also

The hCard validator at *http://en.hcard.geekhood.net/*; Recipe 1.14 for using HTML to mark up an event

1.14 Marking Up an Event (hCalendar)

Problem

You want to use HTML to mark up an event.

Solution

Use `class` and `title` attributes with specific attributes listed in the hCard microformat specification (see *http://microformats.org/wiki/hcalendar*):

```
<div class="vevent" id="hcalendar-The-CSS-Summit">
 <a class="url" href="http://csssummit.com/">
  <abbr class="dtstart" title="2009-07-18T09:00-04:0000">July 18,
2009  9</abbr>
  - <abbr class="dtend" title="2009-07-18T18:00-04:00">6pm</abbr>
  :  <span class="summary">The CSS Summit</span>
 at <span class="location">Online Conference</span></a>
</div>
```

Discussion

Based on the iCalendar file format used to exchange event data, the hCard microformat uses standardized HTML to encode event time and place information into a web document.

Each separate event is designated with the `vevent` class. This specifies the content as an hCalendar entry.

The beginning time of the event, `dtstart` and `summary`, is *required* for every hCalendar event, whereas the end-time `dtend` and `location` properties are optional.

An hCalendar cheat sheet, available at *http://microformats.org/wiki/hcalendar-cheat sheet*, provides a list of optional properties.

See Also

The hCalendar Creator (*http://microformats.org/code/hcalendar/creator*) and the Conference Schedule Creator (*http://dmitry.baranovskiy.com/work/csc/*) to easily create your own hCalendar; Recipe 1.13 for including contact information in a web page

1.15 Validating HTML

Problem

You want to make sure the HTML on your web page is properly coded.

Solution

Use the W3C validator (see *http://validator.w3.org/*) to input the URI of a web document to test its HTML validity, as shown in Figure 1-16.

Alternatively, you can enter code for testing by uploading a CSS file or by entering the CSS rules.
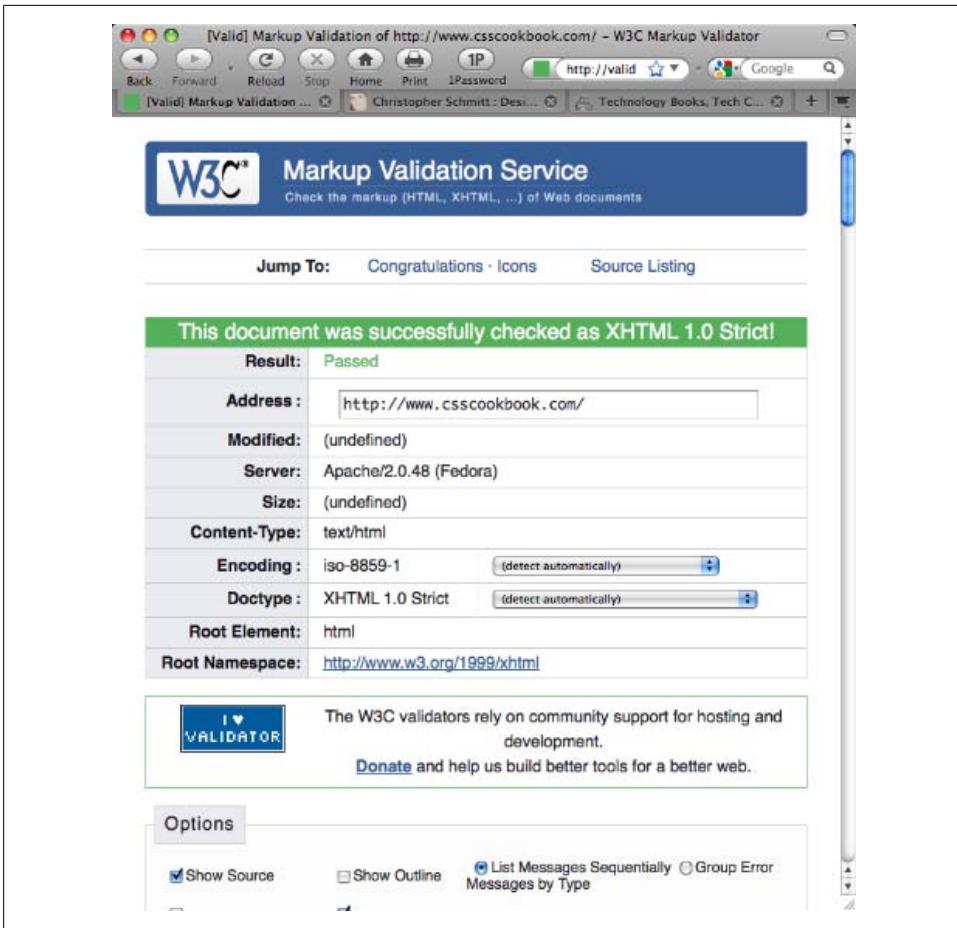
Figure 1-16. Validating a web page

Discussion

The W3C hosts a robust HTML checker on its website. However, sometimes the output can be hard to understand. When validating, make sure to select More Options→Verbose Output.

This feedback option provides more background information regarding errors within your code, giving you a better chance at troubleshooting problems.

Creating an HTML validator bookmarklet

Take any page you visit on the Web directly to the W3C's HTML validator through a bookmarklet. A *bookmarklet* is a tiny piece of JavaScript tucked away in the Address portion of a bookmark.

Create a new bookmark, name it "HTML Validator," and then replace whatever is in the address field with this line:

```
javascript:void(document.location='http://validator.w3.org/check?
charset=%28detect+automatically%29&doctype=Inline&ss=1&group=0&
verbose=1&uri='+escape(document.location))
```

When visiting another site, clicking on the bookmarklet takes the page currently loaded in the browser and runs it through the CSS validator.

See Also

Recipe 2.27 for validating CSS rules

CSS Basics

2.0 Introduction

Cascading Style Sheets (CSS) provide a simple way to style the content on your web pages. CSS may look complicated to first-time users, but this chapter shows how easy it is to use CSS.

Here's an exercise with the traditional "Hello, world!" example. First, open a text editor or a favorite web page editor tool and enter the following:

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html>
 <head>
  <title>CSS Cookbook</title>
 <head>
 <body>
  <p>Hello, world!</p>
 </body>
</html>
```

Save the file and view it in your web browser. There is nothing special about this line, as shown in Figure 2-1.
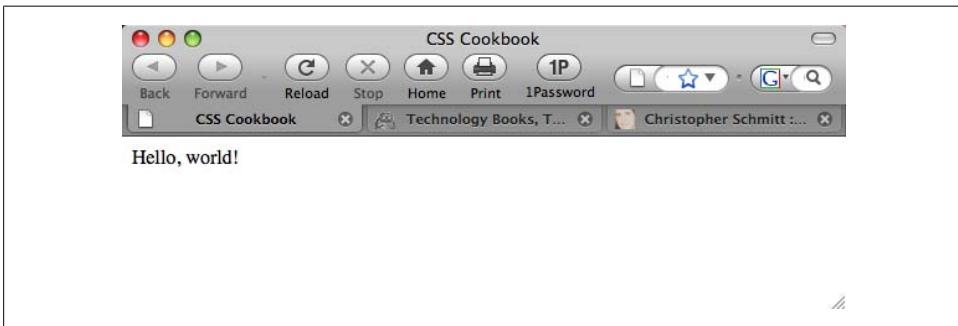
Figure 2-1. The default rendering of HTML text without CSS

To change the style of the HTML text to sans serif, add the following CSS, as shown in Figure 2-2:

```
<p style="font-family: sans-serif;">Hello, world!</p>
```
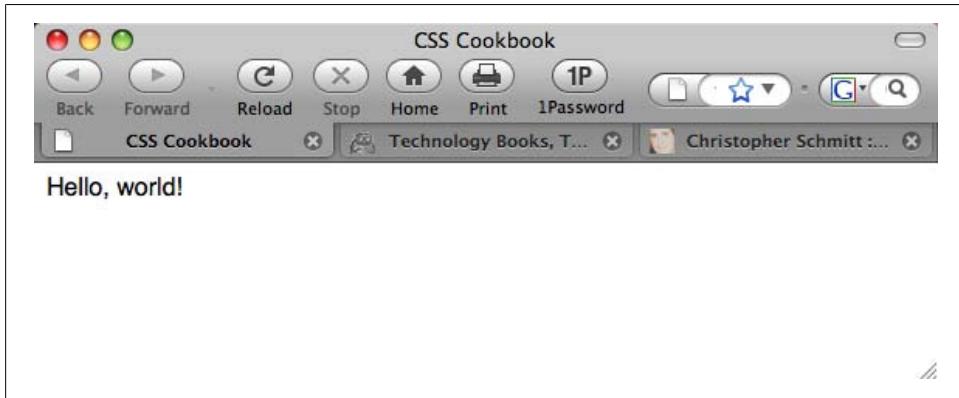

Figure 2-2. The font, changed to sans serif through CSS

To keep the default font but change the font size to 150%, use the following code, as shown in Figure 2-3:

```
<p style="font-size: 150%">Hello, world!</p>
```
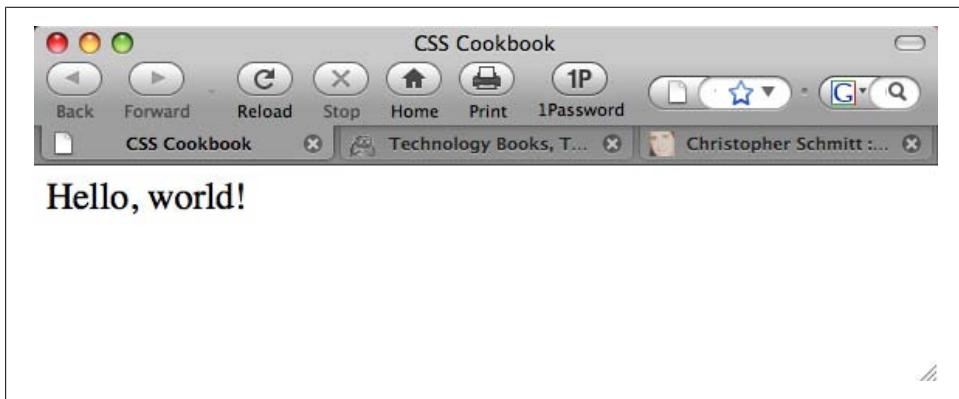

Figure 2-3. Increasing the size of the text

In this chapter, you'll learn about selectors and properties, organizing stylesheets, and positioning. These general recipes will prepare you for fancier recipes in upcoming chapters.

2.1 Applying CSS Rules to a Web Page

Problem

You want to use CSS rules to dictate the design of your web page.

Solution

Start with a blank page in Notepad, your favorite text processor, or HTML development software such as Adobe Dreamweaver or Microsoft Expression.

> If you use a basic text editor, make sure the preferences are set to save as Plain Text (and not Rich Text).

Then add the following HTML between the `body` tags, and save the file as *cookbook.html*:

```
<html>
 <head>
  <title>CSS Cookbook</title>
 </head>
 <body>
  <h1>Title of Page</h1>
  <p>This is a sample paragraph with a
<a href="http://csscookbook.com">link</a>.</p>
 </body>
</html>
```

Now add the following code changes (shown in boldface) to redefine the style for links, bulleted lists, and headers, as shown in Figure 2-4:

```
<html>
 <head>
  <title>CSS Cookbook</title>
  <style type="text/css">
  <!--
  body {
   font-family: verdana, arial, sans-serif;
  }
  h1 {
   font-size: 120%;
  }
  a {
   text-decoration: none;
  }
  p {
   font-size: 90%;
  }
  -->
  </style>
```

```
  </head>
  <body>
   <h1>Title of Page</h1>
   <p>This is a sample paragraph with a
<a href="http://csscookbook.com">link</a>.</p>
  </body>
  </html>
```
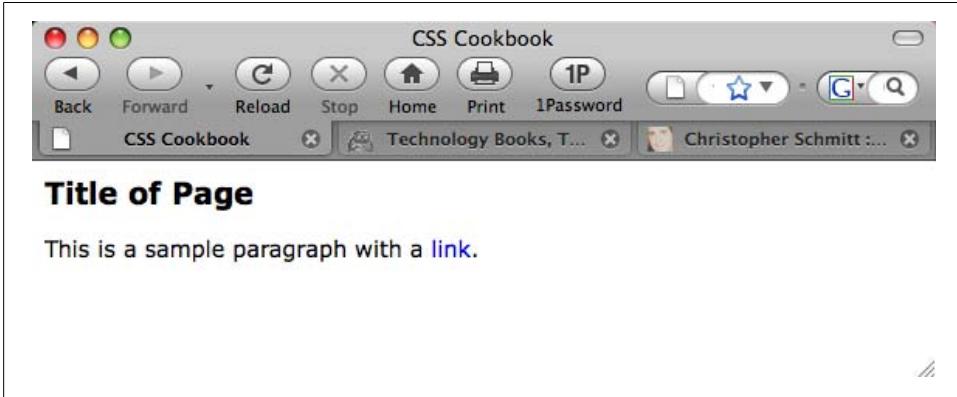

Figure 2-4. Content rendered differently after adding CSS

Discussion

CSS contains rules with two parts: selectors and properties.

A *selector* identifies what portion of your web page gets styled. Within a selector are one or more properties and their values.

The *property* tells the browser what to change, and the *value* lets the browser know what that change should be.

For instance, in the following declaration block example, the selector tells the browser to style the content marked up with h1 elements in the web page to 120% of the default size:

```
h1 {
 font-size: 120%;
}
```

Table 2-1 shows a breakdown of the selectors, properties, and values in the Solution. The "Result" column explains what happens when you apply the property and value to the selector.

Table 2-1. Breakdown of selectors, properties, and values in the Solution

Selector	Property	Value	Result
h1	font-size	120%	Text size larger than default size
p	font-size	90%	Text size smaller than default size

The standard for writing CSS syntax includes the selector, which is normally the tag you want to style, followed by properties and values enclosed within curly braces:

```
selector { property: value; }
```

However, most designers use the following format to improve readability:

```
selector {
 property: value;
}
```

The addition of whitespace and line breaks helps make the CSS more readable. Both are valid approaches to writing CSS. Use whatever method is more comfortable for you.

Also, CSS allows selectors to take on more than one property at a time, to create more complex visual presentations. To assign multiple properties within a selector, use a semicolon to separate the properties, as shown in the following code. Note the use of the semicolon following the last property in the list, though there are no other properties following it. This ensures that we can quickly add new items, without the potential of adding errors by forgetting the separator:

```
selector {
 property: value;
 property: value, value, value;
 property: value value value value;
}
selector, selector {
 property: value;
}
```

Wrapping the CSS rules

For internal stylesheets (see Recipe 2.11), the CSS rules are wrapped within the HTML `style` element:

```
<style type="text/css">
  <!--
  -->
</style>
```

The `style` element informs the browser that the content inside the element comprises formatted CSS rules and that the browser should be prepared to process the content. The HTML comment is there to shield older browsers that do not know how to render CSS rules appropriately. For most modern browsers, the HTML comment is no longer needed.

See Also

Recipe 2.2 for more information about CSS selectors; Appendixes C and D for lists of selectors

2.2 Using Basic Selectors to Apply Styles

Problem

You want to use basic selectors to associate styles to a web page.

Solution

Use different kinds of selectors to target different portions of web pages to style, as shown in Figure 2-5:

```
<html>
 <head>
  <title>CSS Cookbook</title>
  <style type="text/css">
  <!--
  * {
   font-family: verdana, arial, sans-serif;
  }
  h1 {
   font-size: 120%;
  }
  #navigation {
   border: 1px solid black;
   padding: 40px;
  }
  li a {
   text-decoration: none;
  }
  p {
   font-size: 90%;
  }
  -->
  </style>
 </head>
 <body>
  <h1>Title of Page</h1>
  <p>This is a sample paragraph with a
<a href="http://csscookbook.com">link</a>. Lorem ipsum dolor sit amet,
consectetuer adipiscing elit, sed diam nonummy nibh euismod tincidunt ut
laoreet dolore magna <em class="warning">aliquam erat volutpat</em>. Ut
wisi enim ad minim veniam, quis nostrud exerci tation ullamcorper suscipit
lobortis nisl ut aliquip ex ea commodo consequat.<p>
   <ul id="navigation">
    <li><a href="http://csscookbook.com">Apples</a></li>
    <li><a href="http://csscookbook.com">Bananas</a></li>
    <li><a href="http://csscookbook.com">Cherries</a></li>
   </ul>
 </body>
</html>
```
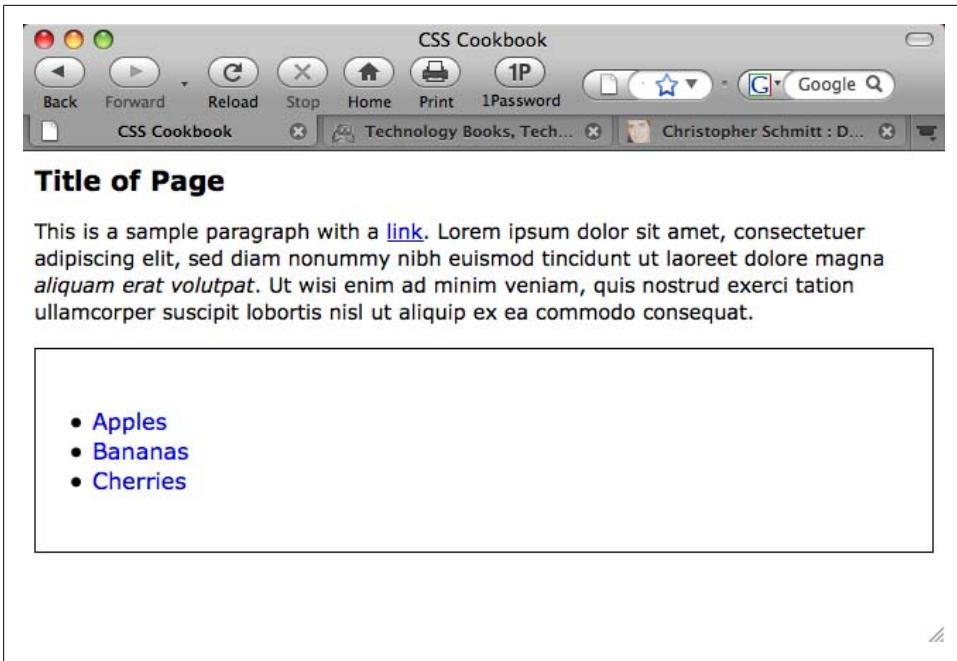
Figure 2-5. Web page with CSS styles

Discussion

CSS allows for many, and sometimes ingenious, ways to pinpoint which parts of a web page should be styled.

To better understand how to pick out portions of a web page using selectors, a developer needs to recognize that content marked up with HTML creates a structure.

Although the elements used in the HTML in the Solution might look like a jumbled order, as shown in Figure 2-6, they do follow a certain structure.

This structure might be invisible to the visitor visiting the web page, but it's a crucial part of the rendering process a browser goes through.

When a browser pulls a web page from the server and begins to display the page, the elements of the page are placed in a structure that the browser software assembles.

Although this process of placing the elements in an organizational structure is more programming oriented, a good visual representation would be to view the structure much like an organizational chart at a company.

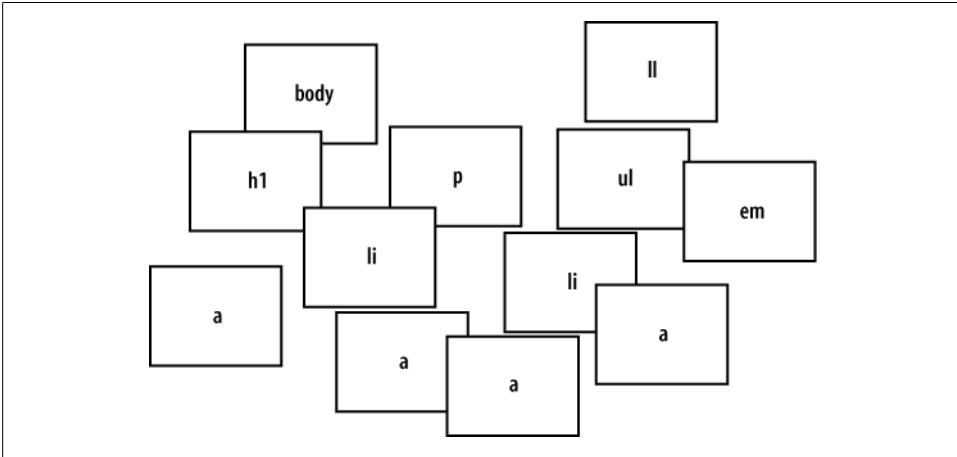Based on the HTML used in the Solution, the organizational chart would look like Figure 2-7.

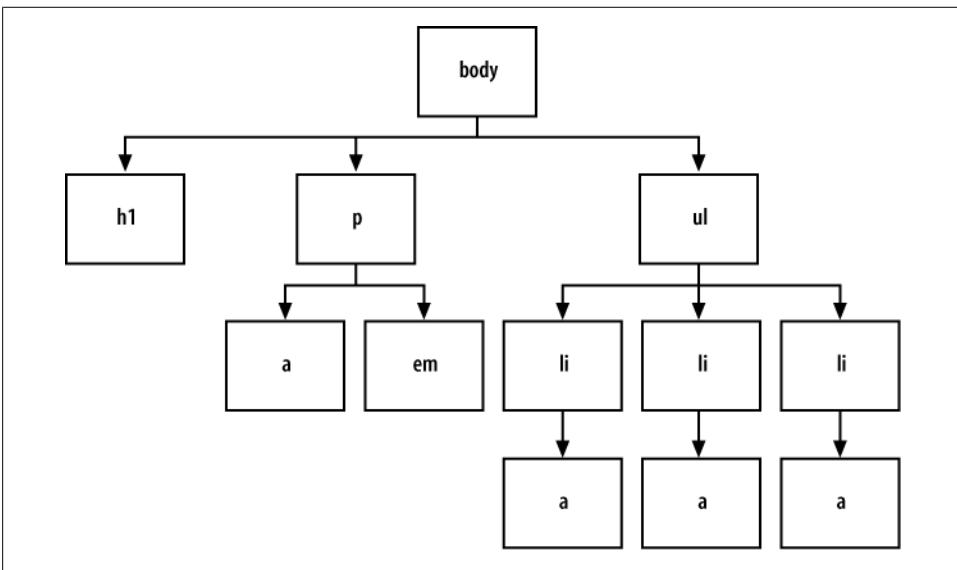*Figure 2-6. Elements used in the Solution*

Figure 2-7. Elements used in the web page arranged in a structure

Type selectors

Type selectors are selectors that name the element or HTML tag to style. The following rules apply font styles to the h1 and p elements within a web page, as shown in Figure 2-8:

```
h1 {
 font-size: 120%;
}
p {
 color: blue;
}
```
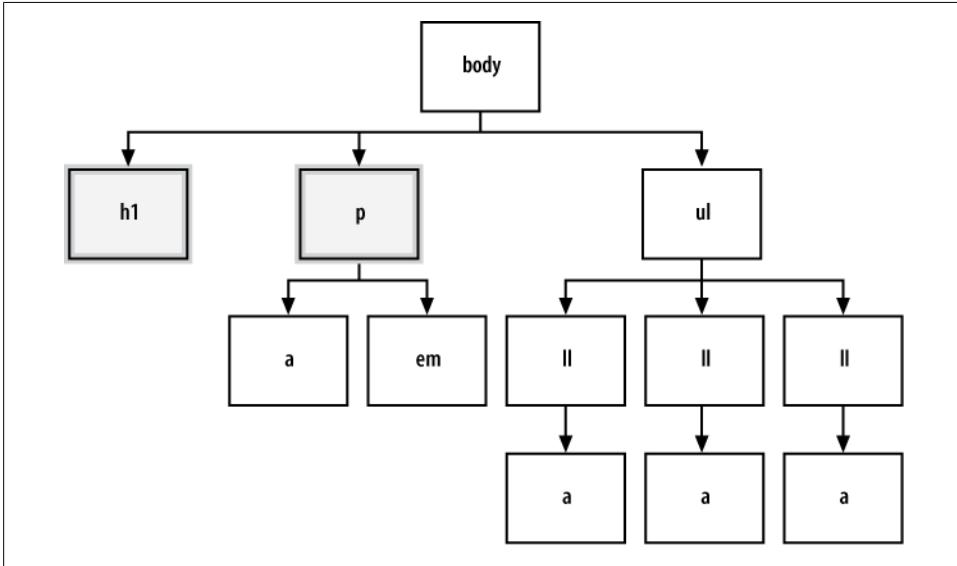

Figure 2-8. The elements selected from the CSS rules

Class selectors

When you want to apply the same CSS rule on different elements, you can use a *class selector*.

For example, you can use class selectors to identify warnings with boldface text in a paragraph as well as a list item.

First, create a `warning` class selector preceded by a period (`.`), which is also known as a full stop:

```
<html>
 <head>
  <title>CSS Cookbook</title>
  <style type="text/css">
  <!--
  * {
   font-family: verdana, arial, sans-serif;
  }
  body {
  }
  h1 {
   font-size: 120%;
```

```
    }
    #navigation {
     border: 1px solid black;
     padding: 40px;
    }
    li a {
     text-decoration: none;
    }
    p {
     font-size: 90%;
    }
    .warning {
     font-weight: bold;
    }
    -->
    </style>
   </head>
   <body>
    <h1>Title of Page</h1>
    <p>This is a sample paragraph with a
<a href="http://csscookbook.com">link</a>. Lorem ipsum dolor sit amet,
consectetuer adipiscing elit, sed diam nonummy nibh euismod tincidunt
ut laoreet dolore magna <em class="warning">aliquam erat volutpat</em>.
Ut wisi enim ad minim veniam, quis nostrud exerci tation ullamcorper suscipit
lobortis nisl ut aliquip ex ea commodo consequat.<p>
     <ul id="navigation">
      <li><a href="http://csscookbook.com">Apples</a></li>
      <li><a href="http://csscookbook.com">Bananas</a></li>
      <li><a href="http://csscookbook.com">Cherries</a></li>
     </ul>
   </body>
  </html>
```

Then add the `class` attribute to a link and a list item to style those elements, as shown in Figure 2-9:

```
   <html>
    <head>
     <title>CSS Cookbook</title>
     <style type="text/css">
     <!--
     * {
      font-family: verdana, arial, sans-serif;
     }
   h1 {
      font-size: 120%;
     }
     #navigation {
      border: 1px solid black;
      padding: 40px;
     }
     li a {
      text-decoration: none;
     }
     p {
      font-size: 90%;
```

```
    }
    .warning {
     font-weight: bold;
    }
    -->
    </style>
   </head>
   <body>
    <h1>Title of Page</h1>
    <p>This is a sample paragraph with a
<a href="http://csscookbook.com" class="warning">link</a>. Lorem ipsum dolor
sit amet, consectetuer adipiscing elit, sed diam nonummy nibh euismod tincidunt
ut laoreet dolore magna <em class="warning">aliquam erat volutpat</em>. Ut wisi
enim ad minim veniam, quis nostrud exerci tation ullamcorper suscipit lobortis
nisl ut aliquip ex ea commodo consequat.<p>
     <ul id="navigation">
      <li class="warning"><a href="http://csscookbook.com">Apples</a></li>
      <li><a href="http://csscookbook.com">Bananas</a></li>
      <li><a href="http://csscookbook.com">Cherries</a></li>
     </ul>
   </body>
  </html>
```
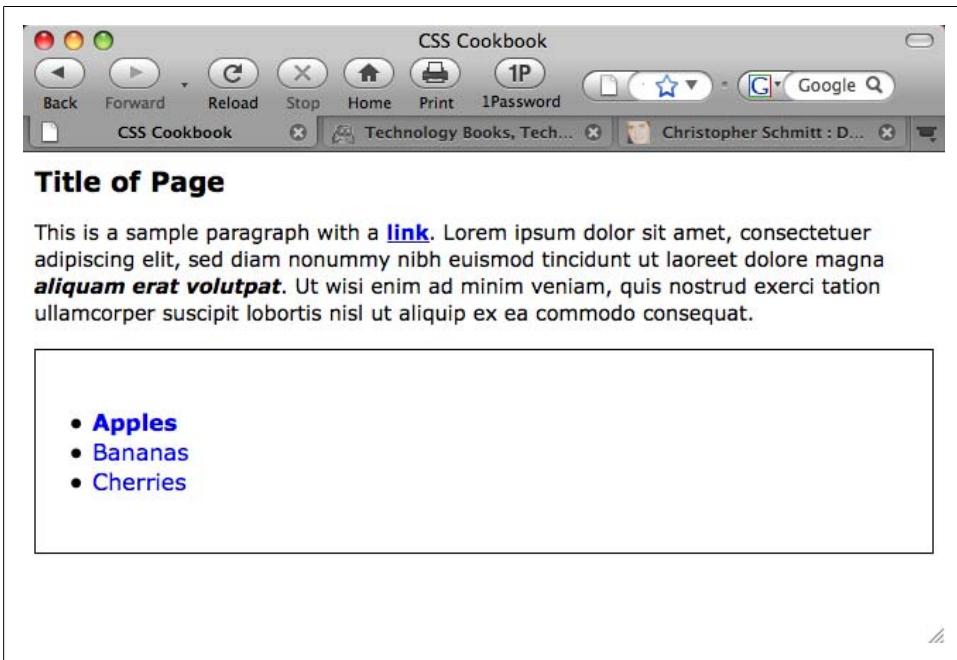

Figure 2-9. The CSS class selectors modifying the look of the web page

Figure 2-10 shows which portions of the document are selected with this class selector.
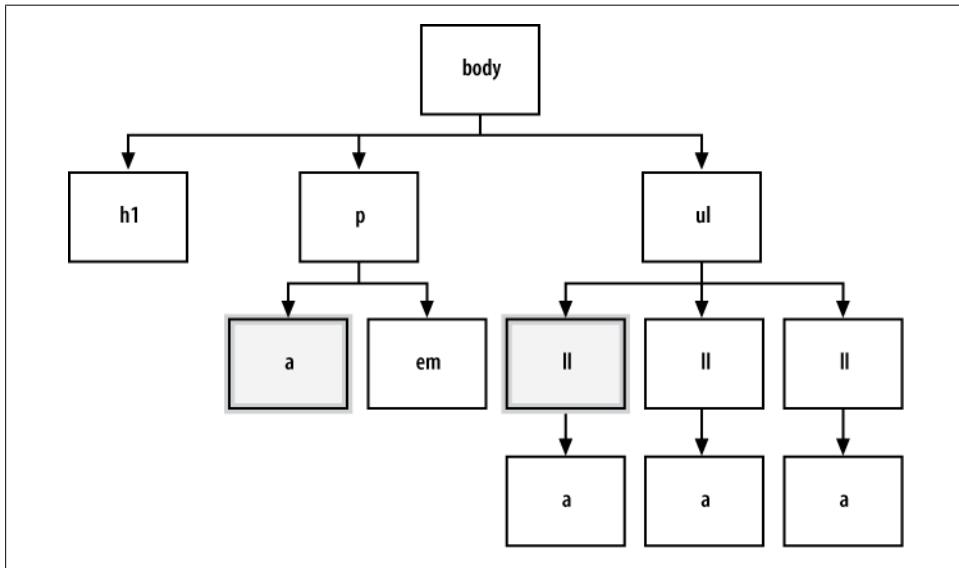

Figure 2-10. The styled elements within the page structure

ID selectors

ID selectors resemble class selectors except they appear once in the HTML document. An ID selector can appear multiple times in a CSS document, but the element an ID selector refers to appears only once in an HTML document.

Often, ID selectors appear in a `div` to mark major divisions within a document, but you can use them elsewhere.

To create an ID selector, use the hash symbol (#), followed immediately by a label or name:

```
#navigation {
   border: 1px solid black;
   padding: 40px;
   }
```

Then add an `id` attribute with a value of `navigation`, as shown in Figure 2-11:

```
<ul id="navigation">
 <li class="warning"><a href="http://csscookbook.com">Apples</a></li>
 <li><a href="http://csscookbook.com">Bananas</a></li>
 <li><a href="http://csscookbook.com">Cherries</a></li>
</ul>
```
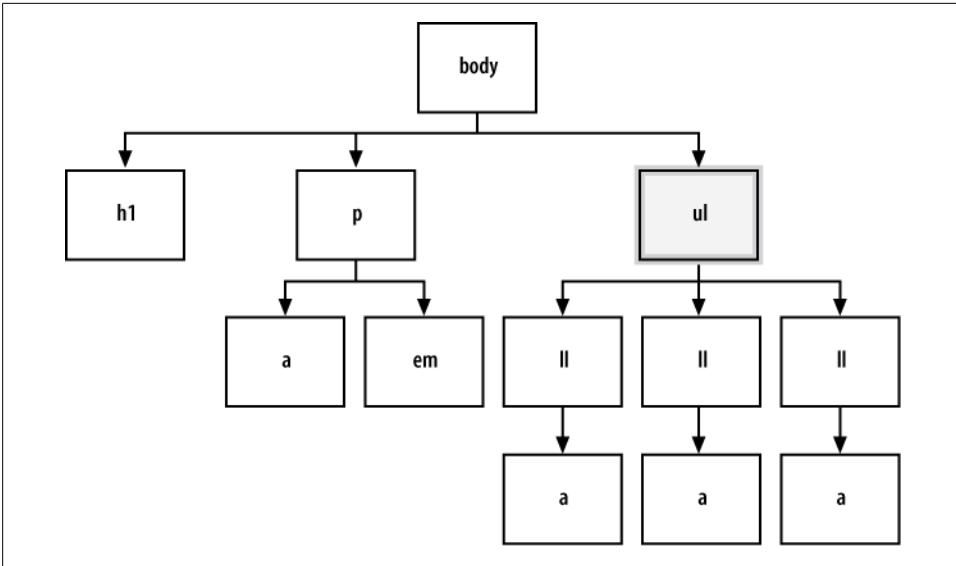
Figure 2-11. An unordered list element, styled

Descendant selectors

Descendant selectors allow for more granular control in picking parts of a web page than type and class selectors. Descendant selectors typically have two elements, with the second element being a descendant of the first:

```
li a {
 text-decoration: none;
}
```

The following code adds the HTML in which `a` appears within `li`, as shown in Figure 2-12:

```
<ul id="navigation">
 <li class="warning"><a href="http://csscookbook.com">Apples</a></li>
 <li><a href="http://csscookbook.com">Bananas</a></li>
 <li><a href="http://csscookbook.com">Cherries</a></li>
</ul>
```

In this example, every time there is a link or `a` element within a list item or `li` element, this CSS rule is applied.

Universal selectors

The *universal selector* is represented with an asterisk (*) and is applied to all elements, as shown in Figure 2-13.

In the following code, every element containing HTML text would be styled with Verdana, Arial, or some other sans serif font:

```
* {
 font-family: Verdana, Arial, sans-serif;
}
```
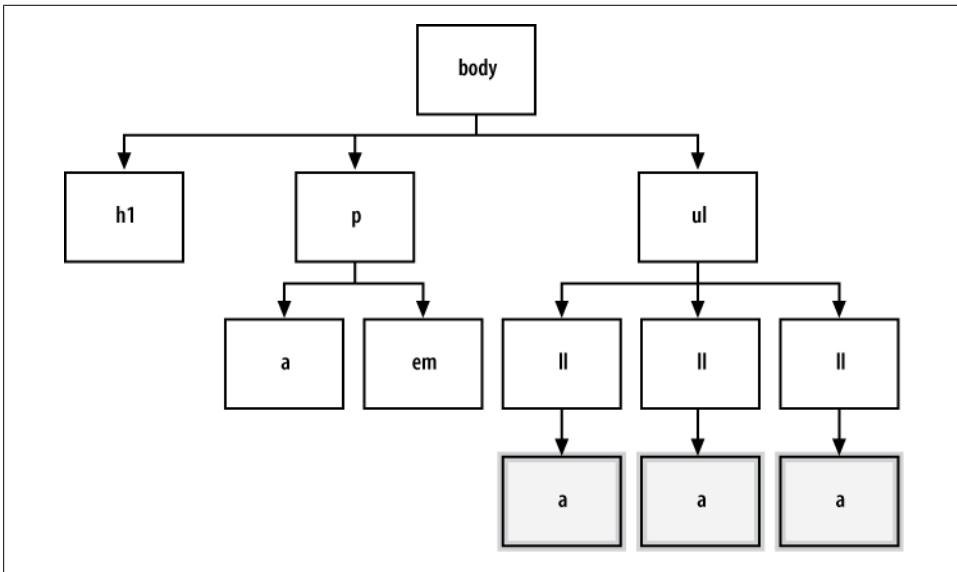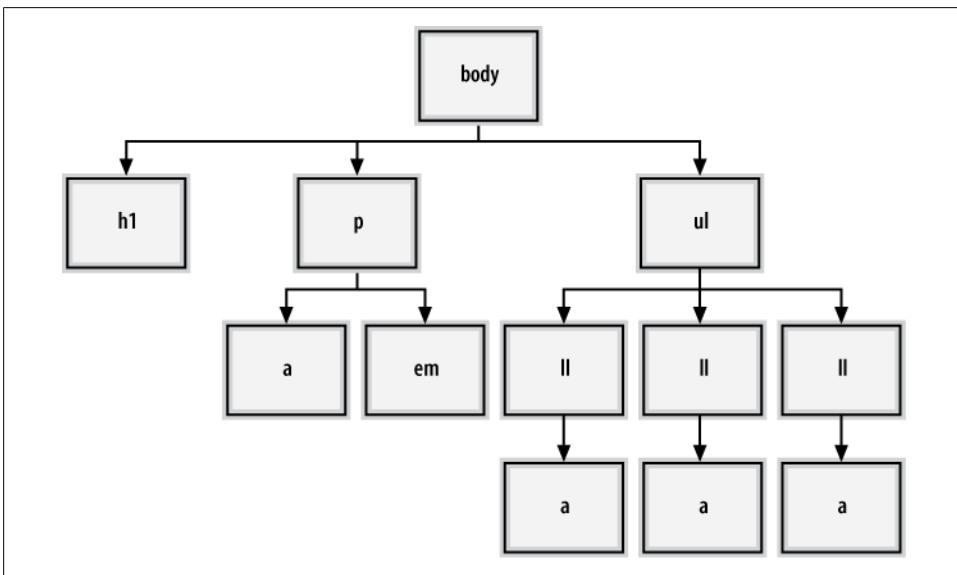

Figure 2-12. The links within the list items selected

Figure 2-13. Every element styled with the universal selector

See Also

The CSS 2.1 specification for selectors at *http://www.w3.org/TR/CSS21/selector.html*; Selectutorial, a tutorial of CSS selectors, at *http://css.maxdesign.com.au/selectutorial/*; the browser selector support guide from Westciv at *http://westciv.com/style_master/ academy/browser_support/selectors.html*; Chapter 3 for more on web typography; Appendix C for a list of selectors

2.3 Applying Child Selectors

Problem

You want to style descendant selectors, but only child elements that are one level from their parent element.

Solution

Use a child selector, which you signify by a right-angled bracket often set between two type selectors, as shown in the following code:

```
strong {
 text-decoration: underline;
}
div > strong {
 text-decoration: none;
}
```

Discussion

With a child selector, an element is styled if it is the *direct* descendant of its parent element.

Only the `strong` element that isn't contained within another element, the `div` element in this case, is not underlined, as shown in Figure 2-14:

```
Nothing happens to this part of the sentence because this
<strong>strong</strong> isn't the direct child of div.
<div>
 However, this <strong>strong</strong> is the child of div.
Therefore, it receives the style dictated in the CSS rule.
</div>
```

To see which elements are affected by this CSS rule in an organizational chart, take a look at Figure 2-15.

As shown in Figures 2-14 and 2-15, the first `strong` element was not underlined because it was placed within the `div` element.
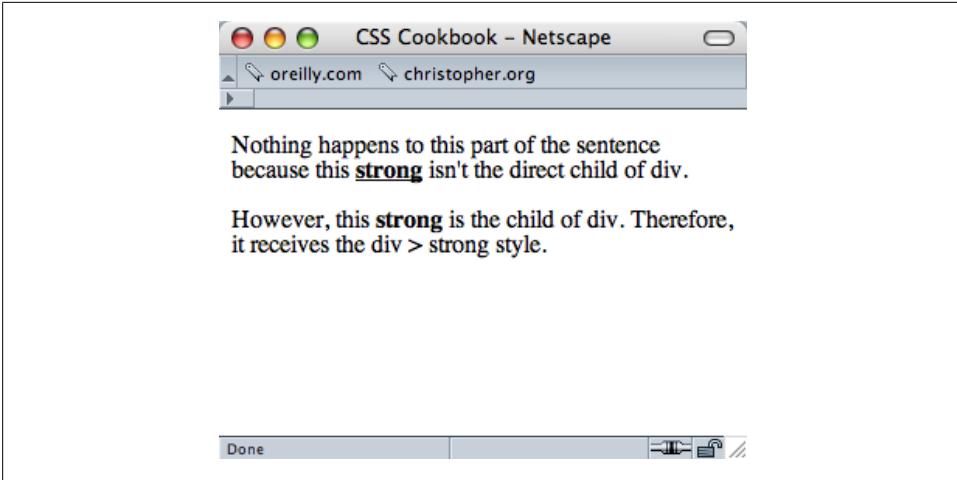
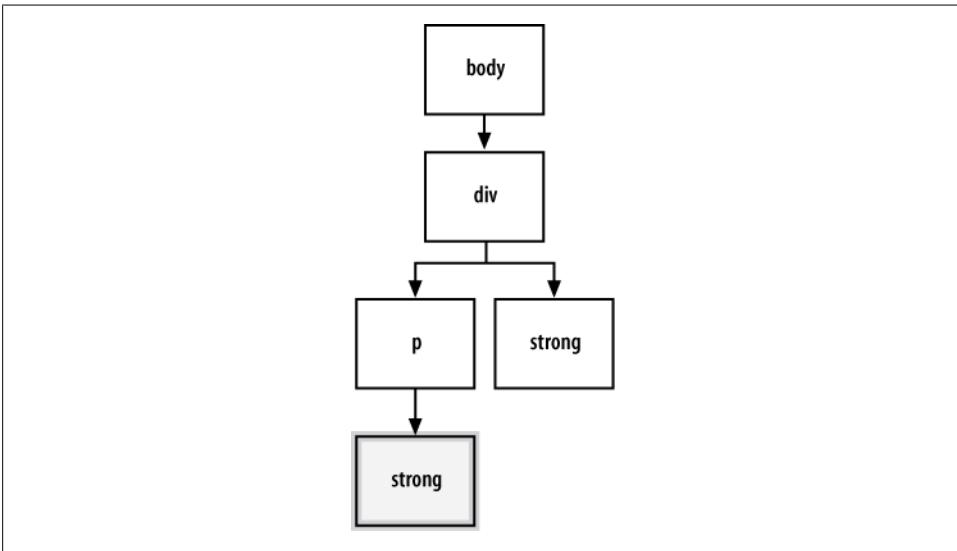*Figure 2-14. The effect of the child selector rule*

Figure 2-15. The child selector highlighted in the markup structure

If the direct parent-to-child relationship is not present, the style won't hold. This is an easy but powerful difference between a child selector and a descendant selector.

Child selectors are not supported in Internet Explorer 6 and earlier.

See Also

The CSS 2.1 specification for child selectors at *http://www.w3.org/TR/CSS2/selector
.html#child-selectors*

2.4 Applying Adjacent Selectors

Problem

You want to assign styles to an element when it's next to another, specific element.

Solution

Use an adjacent sibling, which is formed by a plus sign between two selectors, as shown
in the following code:

```
li + li {
  font-size: 200%;
}
```

Discussion

Adjacent siblings describe the relationship between two elements that are placed side
by side within the flow of a web page's markup.

Figure 2-16 shows the effect of this adjacent sibling rule. Notice that only the second
and third list items are styled, since the second and third list items are placed side by
side. The first item is not styled because it does not meet the requirements of having a
list item come before it.

To see which elements are affected by this CSS rule showcasing adjacent sibling selectors in an organizational chart, take a look at Figure 2-17.

> Adjacent selectors are not supported in Internet Explorer 6 and earlier.

See Also

The CSS 2.1 specification for adjacent selectors at *http://www.w3.org/TR/CSS2/selector
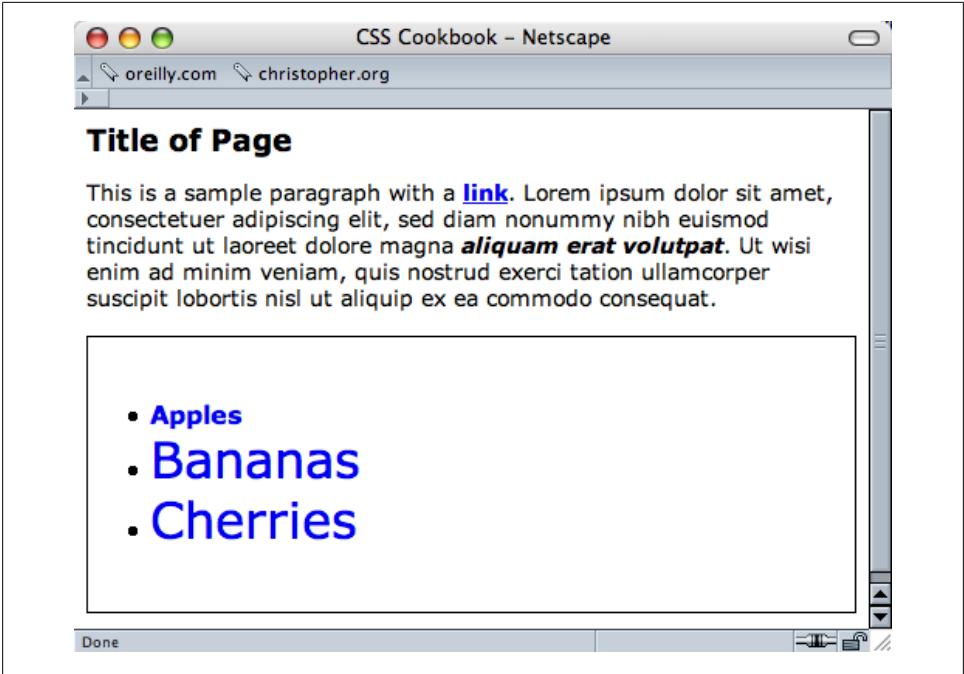.html#adjacent-selectors*

Figure 2-16. Adjacent sibling selectors, which affect the ordered list because it appears after the unordered list
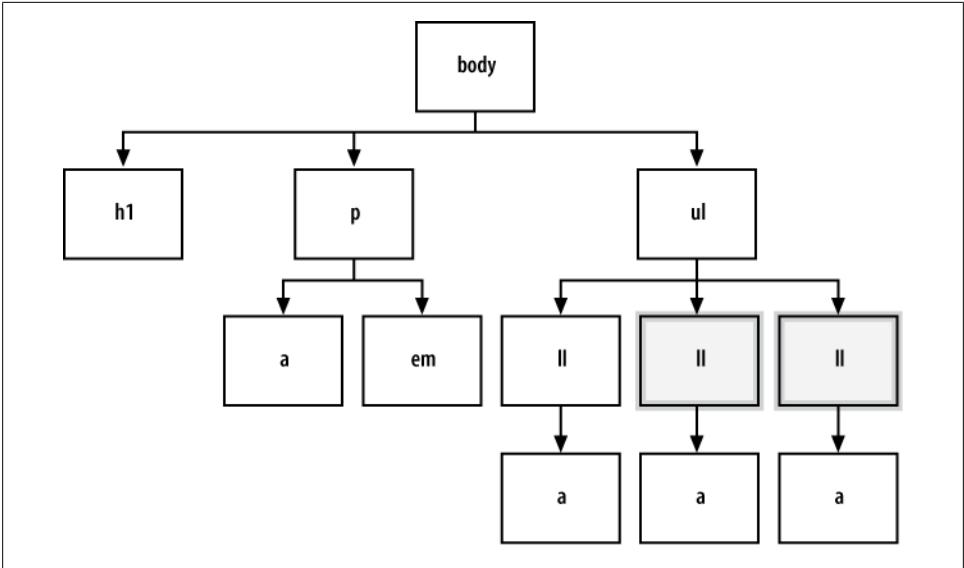

Figure 2-17. Showing which elements are being styled

2.5 Applying Attribute Selectors

Problem

You want to use style elements based on preexisting attributes of HTML elements, rather than adding an additional `class` attribute.

Solution

CSS2 attribute selectors have the following four main options for finding an element:

`[attribute]`
> Searches for matches based on the attribute. For example:
>
> ```
> a[href] {
>  text-decoration: none;
> }
> ```
>
> As a result of the preceding code, whenever the `href` attribute appears within an `a` element in the HTML, the link won't have an underline.

`[attribute=val]`
> Searches for matches based on the value. For example:
>
> ```
> a[href="csscookbook.com"] {
>  text-decoration: none;
> }
> ```
>
> As a result of the preceding code, whenever a link that points to `csscookbook.com` appears in the HTML, the link won't have an underline.

`[attribute~=val]`
> Searches for matches that contain the space-separated attribute somewhere in the value. For example:
>
> ```
> a[title~="tv hd digital"] {
>  text-decoration: none;
> }
> ```
>
> As a result of the preceding code, whenever the word *digital* appears in the `title` attribute of an anchor element, the link won't have an underline.

`[attribute|=val]`
> Searches for matches that contain the attribute with a hyphen. For example:
>
> ```
> a[title|="anti"] {
>  color: red;
> }
> ```
>
> As a result of the preceding code, whenever the word *anti* appears in the `title` attribute of an anchor element, the link is colored red.

Discussion

Although CSS2 selectors enjoy support in major browsers (except for Internet Explorer 6 and earlier), the following new additions to attribute selectors in the CSS3 specification, called *substring matching attribute selectors*, are just beginning to be adopted:

`[attribute^=val]`

Searches for matches where the attribute's value begins with `val`. For example:

```
a[href^="mailto:"] {
 padding-right: 15px;
 background: url(icon-email.png) no-repeat right;
}
```

As a result of the preceding code, whenever a link contains `mailto:`, an email icon is assigned at the end of that link.

`[attribute$=val]`

Searches for matches where the attribute's value ends with `val`. For example:

```
a[href$='.rss'], a[href$='.atom'] {
   padding-right: 15px;
   background: url(icon-rss.png) no-repeat right;
}
```

As a result of the preceding code, whenever a link contains a reference to a syndication feed, an RSS icon is inserted at the end of the link.

`[attribute*=val]`

Searches for matches where the attribute value is anywhere within `val`. For example:

```
a[href *="username"] {
   padding-right: 15px;
   background: url(icons-star.png) no-repeat right;
}
```

As a result of the preceding code, whenever a specific username appears in a link on a social media site, a star icon is added to the right of the link.

See Also

The CSS2 specification for attribute selectors at *http://www.w3.org/TR/CSS2/selector .html#attribute-selectors*; the CSS3 specification for attribute selectors at *http://www .w3.org/TR/css3-selectors/#attribute-selectors*; the Opera Developer Community article on CSS3 selectors at *http://dev.opera.com/articles/view/css-3-attribute-selectors/*

2.6 Using Pseudo-Classes

Problem

You want to add styles to items that are not (typically) based on elements' names, attributes, or content.

Solution

Create a pseudo-class. Here is an example of a pseudo-class that creates a common rollover effect on HTML links:

```
a:link {
 color: blue;
a:visited {
 color: purple;
}
a:hover {
 color: red;
}
a:active {
 color: gray;
}
```

Discussion

In this use of a pseudo-class, a basic link appears in blue. As soon as the mouse pointer hovers over the link, the link changes to red. While the link is being clicked, the link appears gray. When returning to the page with the link after visiting, the link appears purple.

Three other CSS2 pseudo-classes include `:first-child` (which selects the first child element), `:focus` (see Recipe 7.4), and `:lang(n)`.

CSS3 pseudo-classes

The CSS3 specification introduces a new slate of pseudo-classes. Although Internet Explorer does not support these new selectors, browser support is growing for them, as shown in Table 2-2.

Table 2-2. Browser support for CSS3 pseudo-classes

Selector	Firefox 2	Firefox 3.5	Opera 9	Opera 10	Safari 3.1	Safari 4	Chrome
`:target`	Y	Y	Y	Y	Y	Y	Y
`:enabled`	Y	Y	Y	Y	Y	Y	Y
`:disabled`	Y	Y	Y	Y	Y	Y	Y
`:checked`	Y	Y	Y	Y	Y	Y	Y
`:default`		Y	Y	Y			

Selector	Firefox 2	Firefox 3.5	Opera 9	Opera 10	Safari 3.1	Safari 4	Chrome
`:valid`	Y	Y	Y	Y			
`:invalid`	Y	Y	Y	Y			
`:in-range`	Y	Y	Y	Y			
`:out-of-range`	Y	Y	Y	Y			
`:required`			Y	Y			
`:root`	Y	Y	Y	Y	Y	Y	Y
`:not()`	Y	Y		Y	Y	Y	Y
`:nth-child()`				Y	Y	Y	Y
`:nth-last-child()`				Y	Y	Y	Y
`:nth-of-type()`				Y	Y	Y	Y
`:nth-last-of-type()`				Y	Y	Y	Y
`:last-child`		Y		Y	Y	Y	Y
`:first-of-type`				Y	Y	Y	Y
`:last-of-type`				Y	Y	Y	Y
`:only-child`		Y		Y	Y	Y	Y
`:only-of-type`				Y	Y	Y	Y
`:empty`		Y		Y	Y	Y	Y

See Also

The CSS2 specification for pseudo-classes at *http://www.w3.org/TR/CSS2/selector.html #pseudo-class-selectors*; the CSS3 specification for pseudo-classes at *http://www.w3 .org/TR/css3-selectors/#pseudo-classes*

2.7 Using Pseudo-Elements

Problem

You want to style certain aspects of an element without introducing new markup such as a `span` element.

Solution

Use a pseudo-element. You can see an example of the `::first-letter` pseudo-element in Figure 2-18:

```
p::first-letter {
 font-size: 200%;
 font-weight: bold;
 }
```
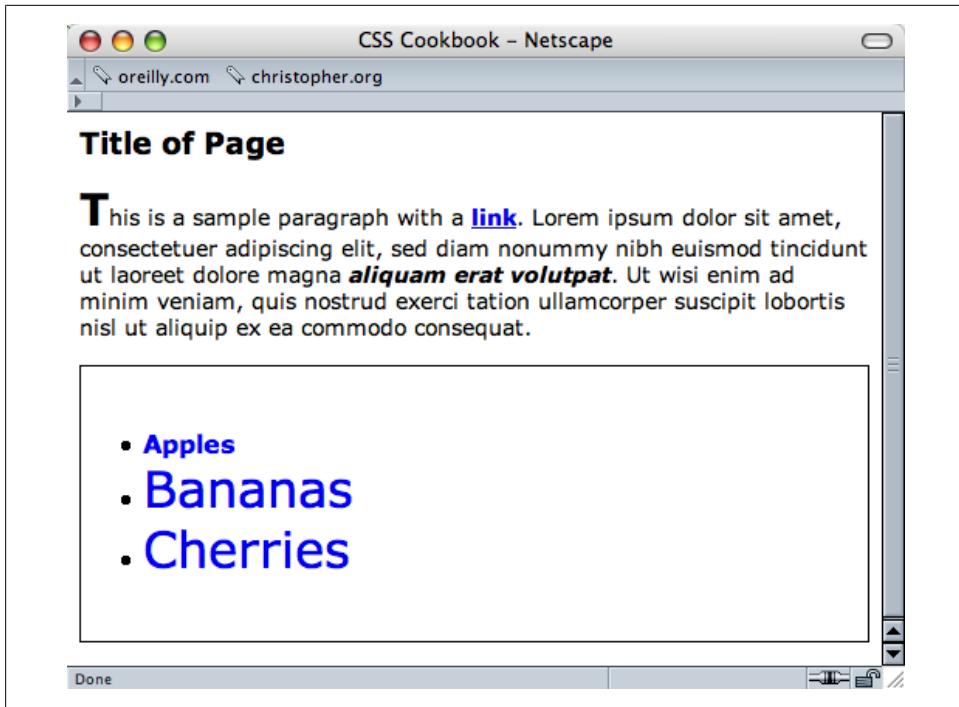
Figure 2-18. The first letter, styled

> A double-colon identifier was added in CSS3, but to add support for Internet Explorer 8, you need to recopy the CSS rules with the original single colon for cross-browser support.

Or you can use `::first-line` (as shown in Figure 2-19) to style the entire first line. If the first line isn't a complete sentence or includes the start of a second sentence, `::first-line` still impacts only the first line:

```
p::first-line {
  font-size: 200%;
  font-weight: bold;
}
```

Discussion

With most selectors, a developer makes use of elements and their arrangement within a web document to style a document.

However, sometimes developers can style an item that's not marked up by elements through the use of pseudo-elements. CSS2 pseudo-elements consist of `::first-letter`, `::first-line`, `::before`, and `::after`.
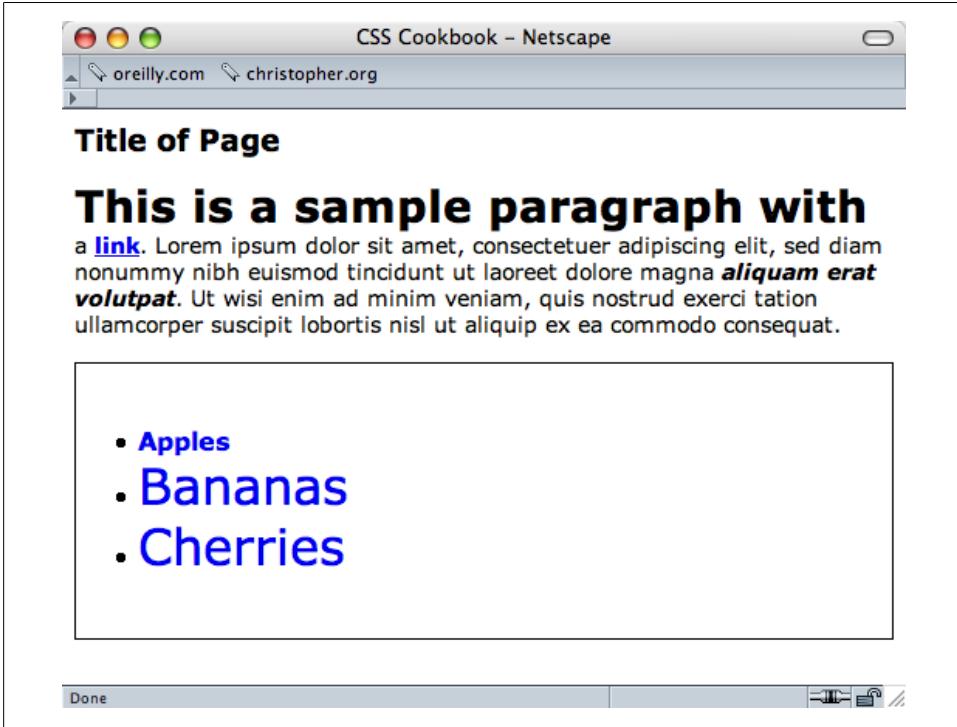
Figure 2-19. The first line, styled

See Also

The CSS 2.1 specification for pseudo-elements at *http://www.w3.org/TR/CSS2/selector .html#pseudo-element-selectors*; the CSS3 specification for pseudo-elements at *http:// www.w3.org/TR/css3-selectors/#pseudo-elements*

2.8 Determining When to Use Class and ID Selectors

Problem

You want to determine the best use for class and ID selectors.

Solution

Use class selectors when you need to apply a style multiple times within a document, and ID selectors for one-time-only appearances within a document.

In the following stylesheet, `#banner` and `#content` are ID selectors and `.title` and `.content` are class selectors:

```
body {
 margin: 0;
 font-family: Verdana, Arial, Helvetica, sans-serif;
 font-size: .75em;
 padding: 0;
}
#banner {
 margin-top: 0;
 margin-bottom: 0;
 background-color: #900;
 border-bottom: solid 1px #000;
 padding: 5px 5px 5px 10px;
 line-height: 75%;
 color: #fff;
}
#sub_banner {
 background-color: #ccc;
 border-bottom: solid 1px #999;
 font-size: .8em;
 font-style: italic;
 padding: 3px 0 3px 10px;
}
#content {
 position: absolute;
 margin-left: 18%;
 width: 40%;
 top: 100px;
 padding: 5px;
}
#nav1 {
 position: absolute;
 width: 30%;
 left: 60%;
 top: 100px;
 padding: 5px;
}
#nav2 {
 position: absolute;
 padding: 5px 5px 5px 10px;
 top: 100px;
 width: 15%;
}
#footer {
 text-align: center;
 padding-top: 7em;
}
.warning {
 font-weight: bold;
 color: red;
}
.title {
 font-size: 120%;
}
.content {
 font-family: Verdana, Arial, sans-serif;
```

```
   margin-left: 20px;
   margin-right: 20px;
  }
  .footer {
   font-size: 75%;
  }
```

Here are the ID and class selectors in the HTML code:

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html>
 <head>
  <title>CSS Cookbook</title>
  <link href="1-2.css" rel="stylesheet" type="text/css" />
 </head>
 <body>
 <div id="header">
  <h1>CSS Collection</h1>
  <h2>Showcase of CSS Web Sites</h2>
 </div>
 <div id="content">
  <h3>Content Page Title</h3>
  <p class="title">Content Item Title</p>
  <p class="content">Content goes here.</p>
 </div>
 <div id="navigation">
  <h3>List Stuff</h3>
  <a href="http://csscookbook.com/">Submit a site</a><br />
  <a href="http://csscookbook.com/">CSS resources</a><br />
  <a href="http://csscookbook.com/">RSS</a><br />
  <h3>CSS Cookbook Stuff</h3>
  <a href="http://csscookbook.com/">Home</a><br />
  <a href="http://csscookbook.com/">About</a><br />
  <a href="http://csscookbook.com/">Blog</a><br />
  <a href="http://csscookbook.com/">Services</a><br />
 </div>
 <div id="blipverts">
  <h3>Ads go here.</h3>
 </div>
 <div id="siteinfo">
   <p class="footer">Copyright 2006</p>
  </div>
 </body>
</html>
```

Discussion

ID selectors identify unique attributes that have one instance in the document tree,
whereas class selectors can be used frequently throughout the web page. Remember
that ID selectors use a hash symbol (#) and class selectors begin with a period (.).

Typically, web developers will use ID selectors to mark off unique sections of a web page. In the Solution, notice that the page is divided into the following sections:

- Header
- Content
- Navigation
- Blipverts
- Siteinfo

By assigning these sections their own ID selector, designers are able to apply customized styles to those areas of the page, while keeping those same styles away from the other sections. This is accomplished through the combination of descendant selectors and ID selectors.

In the following example, the different h3 elements get different CSS rules:

```
#content h3 {
 font-size: 2em;
 font-weight: bold;
}
#navigation h3 {
 font-size: 0.8em;
 font-weight: normal;
 text-decoration: underline;
}
```

HTML5 divisions

Although still a Working Draft at this stage, the HTML5 specification at the time of this writing creates new elements that replace common division in an HTML document with the div element. Some of these new HTML5 elements are:

- header
- nav
- section
- article
- aside
- footer

Instead of writing divisions in the HTML like so:

```
<div id="header">
 ...
</div>
<div id="content">
 ...
</div>
<div id="navigation">
 ...
```

```
    </div>
    <div id="blipverts">
     ...
    </div>
    <div id="siteinfo">
     ...
    </div>
```

you write them within the HTML5 document as follows, resulting in cleaner markup:

```
    <header>
     ...
    </header>
    <section>
     ...
    </section>
    <nav>
     ...
    </nav>
    <aside>
     ...
    </aside>
    <footer>
     ...
    </footer>
```

You can start using HTML5 now, but there are a few caveats.

First you need to use the new DOCTYPE for HTML5, which is easy to memorize in comparison to XHTML's DOCTYPE:

```
    <!DOCTYPE html>
```

Then you need to use JavaScript to get Internet Explorer to treat the new elements like block-level elements:

```
    <script type="text/javascript">
    document.createElement("header");
    document.createElement("section");
    document.createElement("nav");
    document.createElement("aside");
    document.createElement("footer");
    </script>
```

> Although you might rely on JavaScript to enforce a block-level element in Internet Explorer, some web designers have decided to take one step back: they still use div elements, but set the values of the id attributes to those of HTML5 elements.
>
> Through this technique, they are preparing themselves for when HTML5 has gained wider acceptance in browsers. At that time, they can do a simple search and replace through their code to convert a page to HTML5 elements.

Also, when styling the elements be sure to set the elements as block level:

```
header, section, nav, aside, footer {
  display: block;
}
```

See Also

A clickable list of HTML5 elements at *http://simon.html5.org/html5-elements*; the CSS 2.1 specification for ID selectors at *http://www.w3.org/TR/CSS21/selector.html#id-se lectors*; the CSS 2.1 specification for class selectors at *http://www.w3.org/TR/CSS21/ selector.html#class-html*

2.9 Understanding CSS Properties

Problem

You want to learn more about CSS properties.

Solution

Recipes in this chapter cook up popular properties such as `color`, `font-family`, `font-size`, and `text-decoration`. Properties fall between the brackets and their values immediately follow, as shown in the following generic example:

```
selector {
  property: value;
}
```

A real-world example might look like this:

```
li {
  list-style-type: circle;
}
```

Anytime `li` appears in the document, the bullet appears as a circle rather than as a traditional bullet.

Discussion

Selectors identify what should be styled, whereas properties identify how the selectors should be modified.

For example, the `color` property means the element's color will change, but it doesn't indicate *what* color it will change to. That's the job for `value`. Table 2-3 showcases a few more properties and values, and what they do.

Table 2-3. A short list of CSS properties

Property	Value	Result
font-weight	bold	Adds bold to text
border-color	Color name or color hexadecimal HTML value (e.g., #000000 for black and #ffffff for white)	Adds color to a border
border-style	solid	Adds a solid line
	dotted	Adds a dotted line
	dashed	Adds a dashed line
	double	Adds two lines
text-align	left	Aligns text to the left
	center	Aligns text in the center
	right	Aligns text to the right
	justify	Fully expands text from left to right

Learning a new language, even one not as complex as CSS, can be daunting if you cannot grasp what effects or features are available. If you are new to CSS, take some time and code as many properties listed in Appendix B as you can. The more familiar you are with CSS properties, the easier it will be to code web pages.

See Also

The W3C full property table at *http://www.w3.org/TR/CSS21/propidx.html*; the HTML Dog CSS Properties at *http://www.htmldog.com/reference/cssproperties/*; a detailed look at the `border` property in Recipe 4.4; the complete listing of CSS properties in Appendix B

2.10 Understanding the Box Model

Problem

You want to better understand the box model and how margins, borders, and padding work around content.

Solution

Every block-level element, such as a `p` or `div` element, contains a top, right, bottom, and left edge. These sides of a block element are composed of three layers surrounding the content, as shown in Figure 2-20.
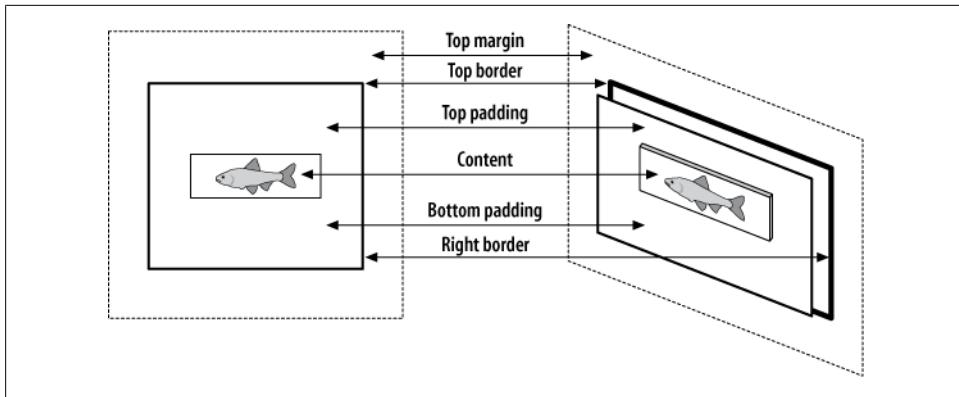
Figure 2-20. Box model viewed straight on and off to the side

Therefore, each `block` element contains four sections:

Content
Actual content such as text, images, Java applets, and other objects. The content area is in the middle of the box.

Padding
Surrounds the content area.

Border
The next-outer layer that surrounds the padding and makes up the box border.

Margin
The transparent box that begins at the edge of the border and expands beyond.

The default *margin* value is 0, which lines up with the edge of the *border*. A border with a value of 0 lines up with the *padding* edge.

Obviously, a padding value of 0 lies flush against the content. Values above 0 expand the boxes.

Discussion

For a mental image of the box model, picture a cardboard box on the floor.

Looking down at the box you see its four sides: top, right, bottom, and left. The box can be as big or as small as you want because you can modify the size of the box through the `height` and `width` properties:

```
div {
 height: 150px;
 width: 150px;
}
```

Add books into the box until you fill the box with books:

```
<div>
<li>Moby Dick</li>
<li>The Red Badge of Courage</li>
<li>The Catcher in the Rye</li>
</div>
```

To help see the edges of the box, place a thin border around the box, as shown in Figure 2-21:

```
div {
 border: thin solid #000000;
 height: 150px;
 width: 150px;
}
```
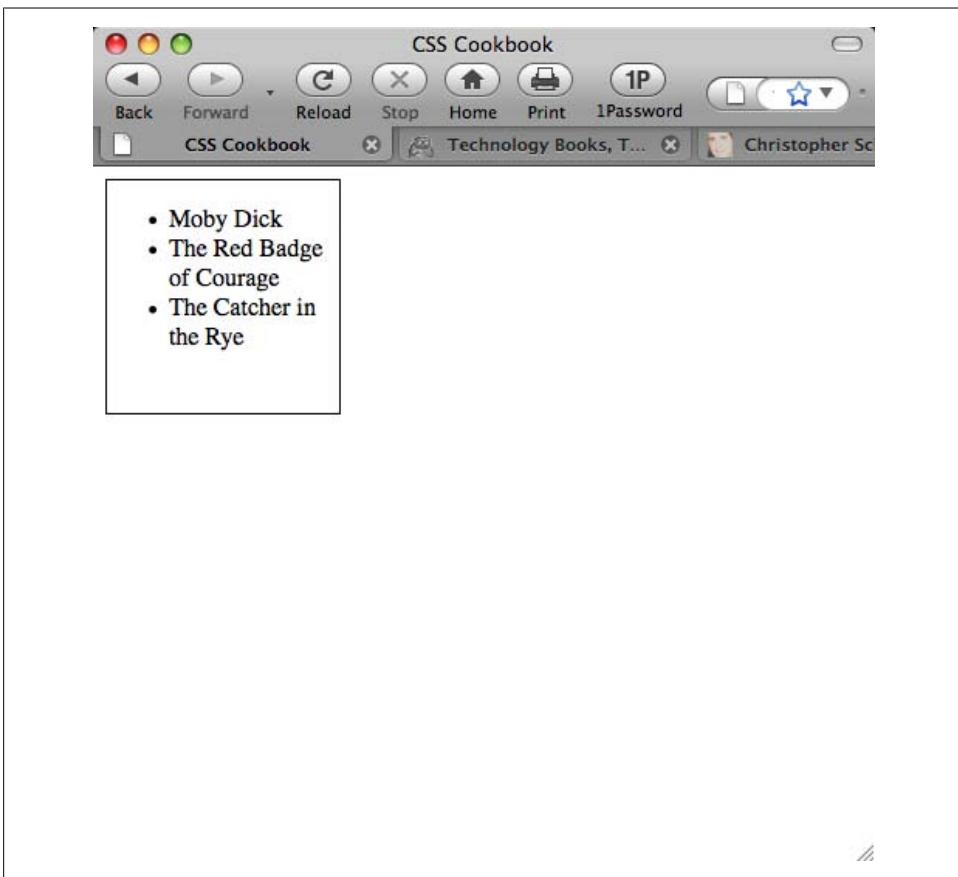

Figure 2-21. A border placed around the content

The books overlap or sit next to each other, and that's not good for books, especially since in this example they're collector's items.

So, add *padding* between the books and the box with the `padding` property for a little breathing room and protection. As you use more padding, you also reduce the number of books you can place into the box. Some padding has been added to the example shown in Figure 2-22:

```
div {
  border: thin solid #000000;
  height: 150px;
  width: 150px;
  padding: 10px;
}
```
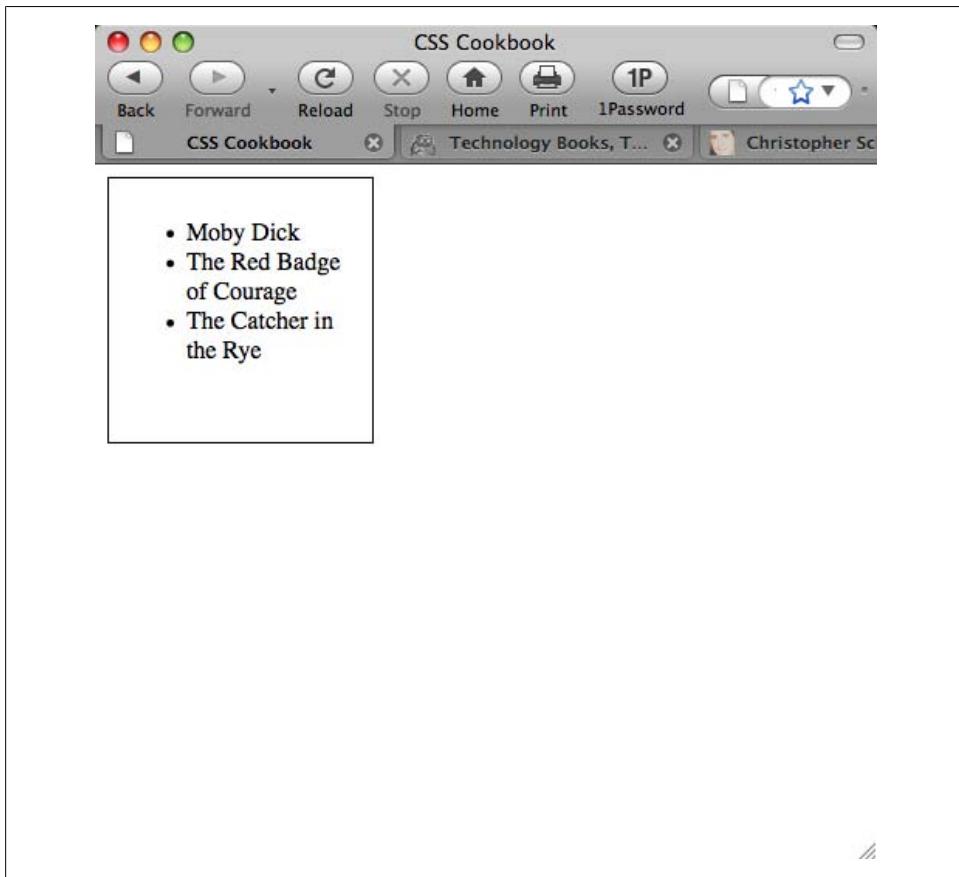

Figure 2-22. Padding added

Adding padding changes the overall size of the box, despite it being set to a width and height of 150 pixels. With the addition of the padding on all sides of the box, the new width is 170 pixels (a padding of 10 pixels is placed on both the right and left sides). Also, the height is now 170 pixels.

You need another box to hold the books that didn't fit in the first box. So, create another box, and enter the rest of the books. Put the new box next to the original below it, as shown in Figure 2-23:

```
<div>
 <li>Moby Dick</li>
 <li>The Red Badge of Courage<li>
 <li>The Catcher in the Rye</li>
</div>
<div>
 <li>The Red Queen</li>
 <li>The Awakening</li>
 <li>The Scarlet Letter</li>
</div>
```
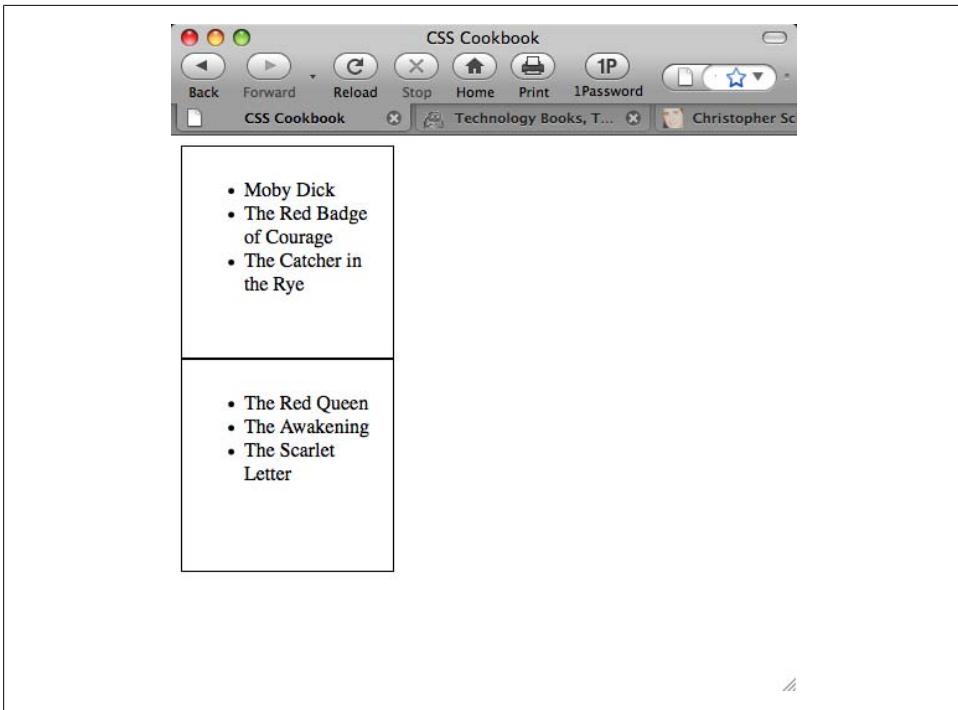

Figure 2-23. An additional listing of books added

However, you want to space out the boxes so that they aren't on top of each other. So, modify the space between the boxes by using the `margin` property, as shown in Figure 2-24:

```
div {
  border: thin solid #000000;
  height: 150px;
  width: 150px;
  padding: 10px;
  margin: 25px;
}
```
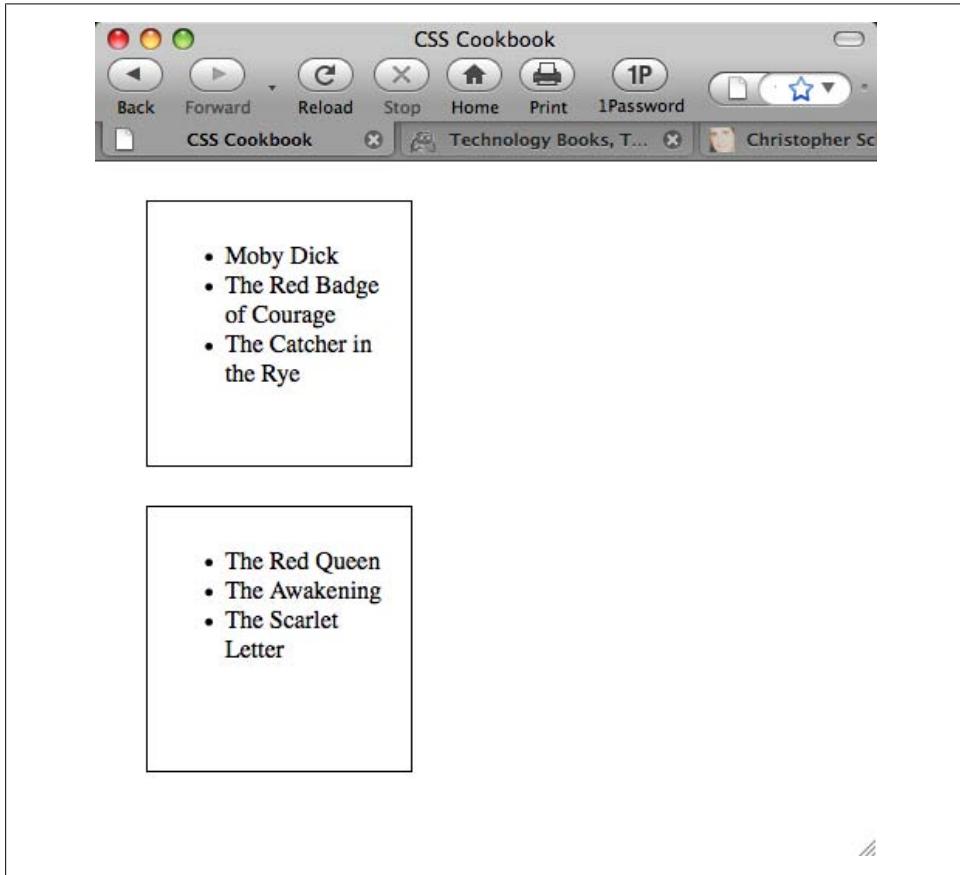

Figure 2-24. Adding a margin to the block-level elements

To help you distinguish the two boxes, modify the `border` property. Like the margin and padding, the border can be as thick or thin as you like. In Figure 2-25, the border was increased to 5 pixels:

```
div {
  border: 5px double #000000;
  height: 150px;
  width: 150px;
  padding: 10px;
  margin: 0px;
}
```
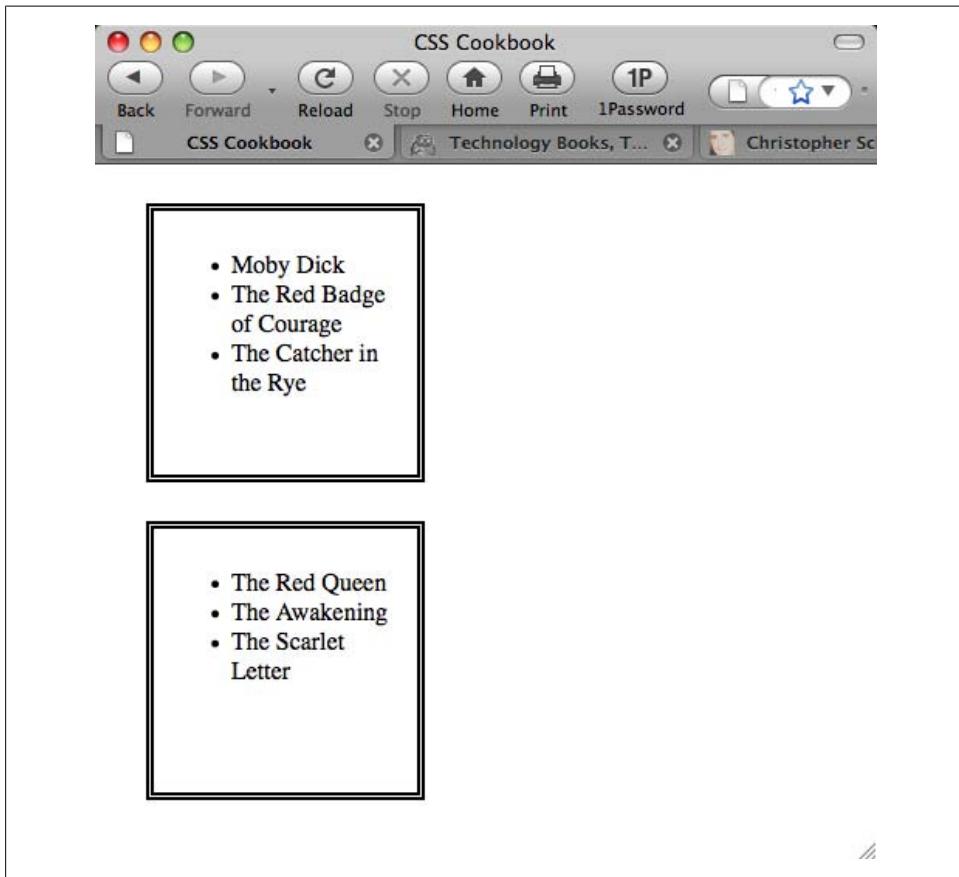

Figure 2-25. Border increased to 5 pixels

At this point, you've modified the box model fairly consistently across two elements. You've adjusted the margin, padding, and borders around each side. However, you can also modify specific edges of the box model.

For example, if you want to adjust the right side of the div element, but keep the same values for the other sides, the code could look something like the following (see Figure 2-26):

```
div {
  border: 5px solid #000000;
  height: 150px;
  width: 150px;
  padding: 10px;
  margin: 0px;
  border-right: 1px solid #000000;
  padding-right: 1px;
  margin-right: 1px;
}
```
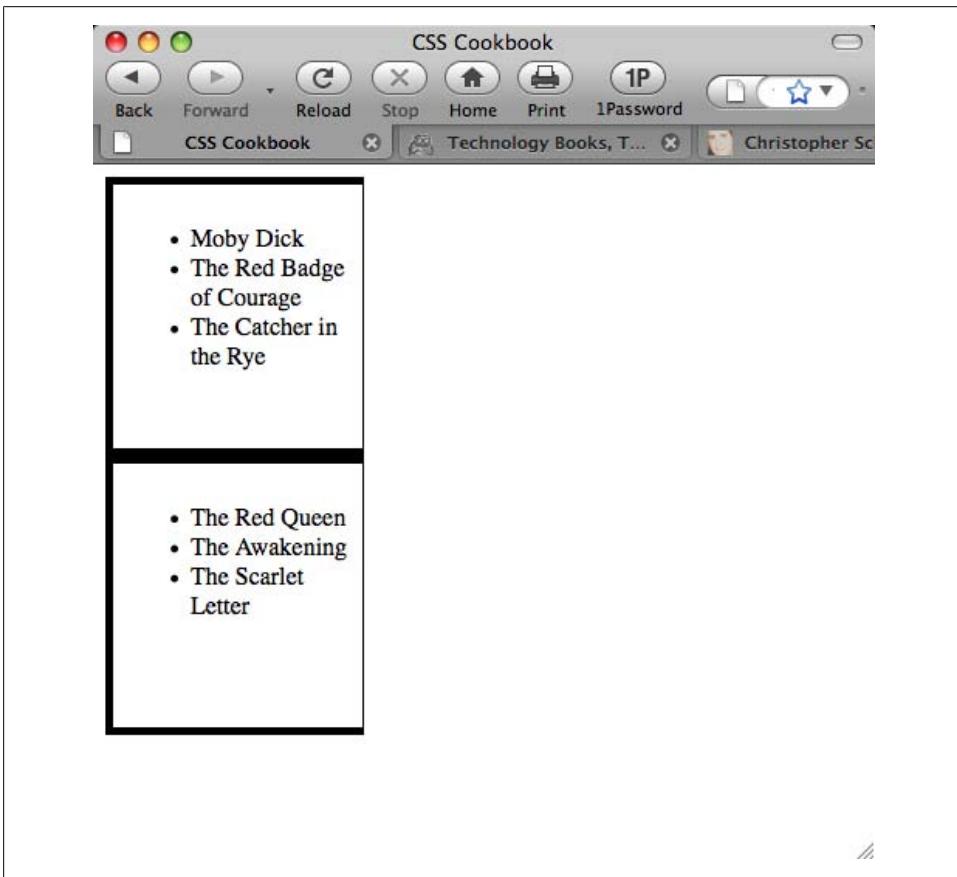

Figure 2-26. Adjustments to the right side of the box

You could also modify the other sides specifically as well. For example, using the `margin` property, the code might look like the following:

```
div {
  margin-top: 1px;
  margin-right: 1px;
  margin-bottom: 1px;
  margin-left: 1px;
}
```

By adjusting the sides and different properties of the box model, developers are able to better format the presentation of their web pages.

See Also

The CSS 2.1 box model at *http://www.w3.org/TR/CSS21/box.html*; the Brain Jar box model at *http://www.brainjar.com/css/positioning/default.asp*; the interactive CSS Box Model demo at *http://www.redmelon.net/tstme/box_model/*

2.11 Associating Styles to a Web Page

Problem

You want to know about the different ways to add styles to a web page.

Solution

You can apply styles in three ways: externally, internally, and inline. An internal style-sheet appears near the top of the HTML document, within the `head`:

```
<style type="text/css">
<!--
#header {
 width: 100%;
 height: 100px;
 font-size: 150%;
}
#content {
 font-family: Verdana, Arial, sans-serif;
 margin-left: 20px;
 margin-right: 20px
}
.title {
 font-size: 120%;
}
-->
</style>
```

Note the use of HTML comments immediately after the `style` element. Those are placed there to prevent the CSS content from showing up in the web page layout or being rendered by the browser in some unwanted fashion.

External stylesheets are stored in a separate file, which gets associated with the HTML file through linking. The following code is saved in its own file:

```css
/* CSS Document */
h1 {
 font-size: 150%;
}
h2 {
 font-size: 120%;
}
p {
 font-family: Verdana, Arial, Helvetica, sans-serif;
}
```
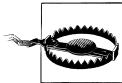
Notice that the `style` element is not present in the external stylesheet.

In the web page, add the following line between the `head` tags to link to the external stylesheet that contains the preceding styles:

```html
<link href="screen.css" rel="stylesheet" type="text/css" media="screen" />
```

Inline styles work similarly to `font` in that they appear with the markup they affect:

```html
<h1 style="font-family: verdana, arial, sans-serif;
font-size: 150%; color: blue;">Page Title</h1>

<p style="font-family: sans-serif; font-size: 90%;">Hello, world!</p>
```

Discussion

The three different types of stylesheets are:

External
All web pages link to the external stylesheet that contains nothing but CSS styles. If you want to change the font color on all pages linked to this stylesheet, just update the external stylesheet. Link to the stylesheet with the `link` tag.

Internal
A unique web page might have its own stylesheet so that styles affect only that page and not all web pages. Define internal styles within the `style` tags.

Inline

Inline styles work similarly to `font` with the style information applied to a specific tag within a web page. Designers rarely apply inline styles and do so when they know there is only one occurrence of a specific style.

External and internal stylesheets save time in terms of website maintenance compared to inline styles. Skipping the use of `font` for every text item needing styling keeps the file slim and trim.

For example, say you inherit a web page where all the text is blue and you use `font` to control the size of the text. You receive orders to change the text to black, so you search for every instance of `<p>` to change the color value from blue to black, as in the following:

```
<p><font size="2" color="blue">Text goes here</font></p>
```

To change all `p` elements from blue to black in an *external* stylesheet requires two steps: open the CSS file and change the color:

```
p {
 color: black;
}
```

In an *internal* stylesheet, you can change the text from blue to black by searching for the style at the top of the page and replacing `blue` with `black`:

```
<style type="text/css">
<!--
p {
 color: black;
}
-->
</style>
```

When to use inline styles

With inline styles, changing the color takes as much time as fixing the original file with the `font` tag:

```
<p style="font-color: blue">Test goes here.</p>
```

Why would anyone want to use inline styles, considering it's time-consuming to make changes? It's rare, but you may have content that appears once in the whole website but that needs a special style. Rather than cluttering the external stylesheet with the style for one item, you'd use inline styles.

When to use internal stylesheets

As for internal and external stylesheets, most sites use external stylesheets. However, when writing the CSS code for a web page design, it's best to start out with an internal stylesheet. When you reach the point where the design is complete or starts to get a little unwieldy, move the stylesheet to a separate file. Then make edits to the external stylesheet as needed.

Also, you may have a special page that's not related to the website or that uses a special style. In this case, an internal stylesheet could be easier to use as opposed to adding more clutter to the external stylesheet.

See Also

The "Style Sheets" section in the HTML 4.01 specification at *http://www.w3.org/TR/ html401/present/styles.html*; W3Schools' "CSS: How to Insert a Style Sheet" at *http:// www.w3schools.com/css/css_howto.asp*

2.12 Understanding the Origin

Problem

You want to know how many ways a CSS rule can be associated to a document.

Solution

You can apply styles to a document in the following ways:

- Via the browser's or user agent's own internal stylesheet
- Via the user's stylesheet (if the user has created one)
- Via your (the author's) stylesheet, which can be one of the following:
  —Inline stylesheet
  —Embedded stylesheet
    —Imported stylesheet
  —Linked or external stylesheet

Discussion

The higher up the list the CSS rules appear, the more prominence they have over other CSS rules that originate elsewhere. Understanding this list is helpful when troubleshooting potential problems in web designs.

See Also

Recipe 2.13 for information on sort order within CSS; Chapter 11 for hacks, workarounds, and troubleshooting tips

2.13 Understanding the Sort Order Within CSS

Problem

You want to know how a browser handles the application of CSS rules.

Solution

The basic rule of thumb is "any CSS rule that is closest to the content wins" over any other CSS rule.

Discussion

With so many ways CSS can be associated to a web document (see Recipe 2.12), there needs to be a way for the browser to handle potential conflicts if the same or a similar rule appears from two different sources.

Follow this guideline when trying to determine how to resolve conflicts within your CSS rules:

- The user's own styles take priority over browser styles.
- The author's (your) styles take priority over user styles.
- Embedded styles take priority over linked or imported styles.
- Inline styles take priority over embedded, linked, or imported styles.

For example, say we have a series of paragraphs, all set to a sans serif font, as shown in Figure 2-27:

```
p {
  font-family: "Gill Sans", Trebuchet, Calibri, sans-serif;
}
```
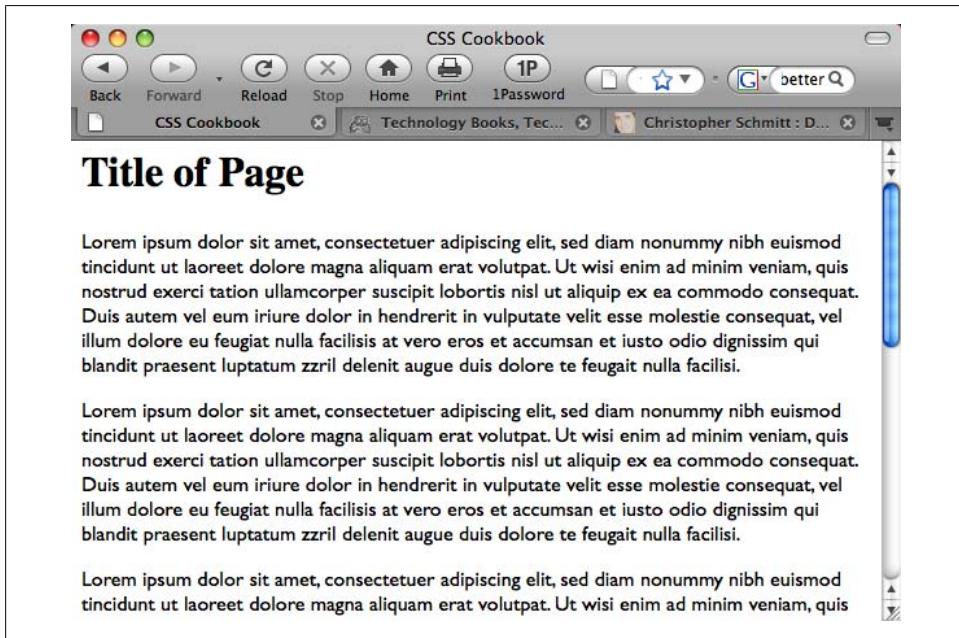

Figure 2-27. Paragraphs set to a sans serif typeface

But when we bring in another rule to style the paragraphs with a serif font and place this new rule before the previous rule, as shown in the following code, the paragraphs remain unchanged:

```
p {
font-family: Garamond, "Hoefler Text", "Times New Roman", Times, serif;
}
p {
 font-family: "Gill Sans", Trebuchet, Calibri, sans-serif;
}
```

Only when we place the serif font rule for the paragraphs after the sans serif font rule does the change in the browser take place, as shown in Figure 2-28:

```
p {
 font-family: "Gill Sans", Trebuchet, Calibri, sans-serif;
}
p {
font-family: Garamond, "Hoefler Text", "Times New Roman", Times, serif;
}
```
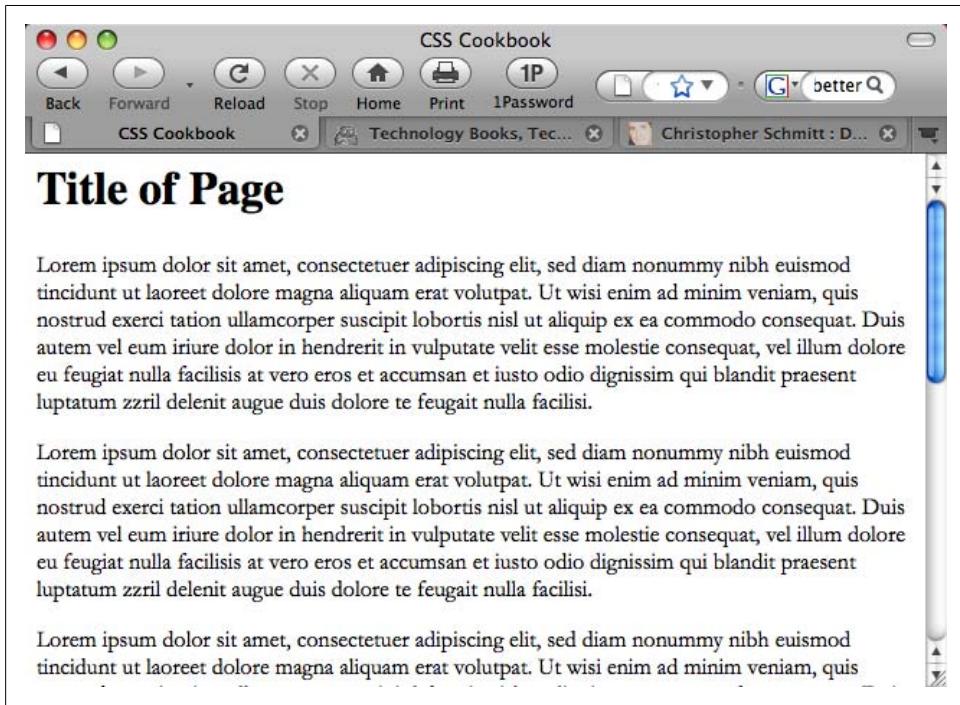

Figure 2-28. Paragraphs set to a serif typeface

Again, this occurrence follows the rule of thumb that "any CSS rule that is closest to the content wins."

However, there is an exception to this rule—and that's where specificity (Recipe 2.15) comes into play.

See Also

Recipe 2.12 for information on how many ways a CSS rule can be associated to a document; Recipe 2.15 for information on how to clarify specificity

2.14 Using !important to Override Certain CSS Rules

Problem

You want to make certain CSS rules more important than others.

Solution

Use the `!important` declaration to override another CSS rule:

```
p {
 font-size: 12px !important;
}
```

Discussion

The `!important` rule consists of an exclamation point (!) followed immediately by the word *important*.

In some browsers, a user can have a stylesheet set up for browsing the Web that enables him to set font sizes or other CSS properties to his liking.

However, as a designer of a web document, you might want to make sure your designs render in the manner you planned. The `!important` rule gives you (very) little insurance that your designs remain intact.

The user controls his experience

The nature of the Web means that designs are never precise or "pixel-perfect" from one display to another. Therefore, the `!important` declaration doesn't ensure that your own styles are what you expect to show up on the user's browser. The user has ultimate control of how a page is viewed on his browser.

Also, although you as the web designer write the `!important` CSS rules, the user also can write these rules in his own stylesheet.

In the CSS2 specification, `!important` rules that the user may wish to write override any `!important` rules the designer writes.

See Also

The CSS 2.1 specification on `!important` rules at *http://www.w3.org/TR/CSS21/cascade* *.html#important-rules*

2.15 Clarifying Specificity

Problem

You want to understand how potential conflicts within CSS are resolved, if origin and sorting order for a CSS rule are the same.

Solution

Each CSS rule carries information that lets the browser (and us) know its weight or specificity.

Consider the following three CSS rules:

```
#header p.big {
 font-family: Impact, Haettenschweiler, "Arial Narrow Bold", sans-serif;
}
p.big {
 font-family: Futura, "Century Gothic", AppleGothic, sans-serif;
}
p {
 font-family: "Gill Sans", Trebuchet, Calibri, sans-serif;
}
```

The higher the specificity a CSS rule possesses, the greater the chance that the CSS rule will win out over another rule. However, when viewed in the browser, the first CSS rule (with the Impact font) wins out, as shown in Figure 2-29.

To determine why the first rule wins, determine the CSS rule's specificity. Follow Table 2-4 when trying to determine a CSS rule's specificity.

Table 2-4. A guide for determining specificity

Selector example	Inline style	Number of ID selectors	Number of class selectors	Number of elements
p	0	0	0	1
p.big	0	0	1	1
#header p.big	0	1	1	1

According to Table 2-4:

- The p selector has a specificity value of 0,0,0,1.
- The p.big selector has a specificity value of 0,0,1,1 because of the class selector.

- The `#header p.big` selector has a specificity value of 0,1,1,1 because of the class and ID selectors.

In these examples, the last selector has a greater specificity, and therefore wins in a conflict.
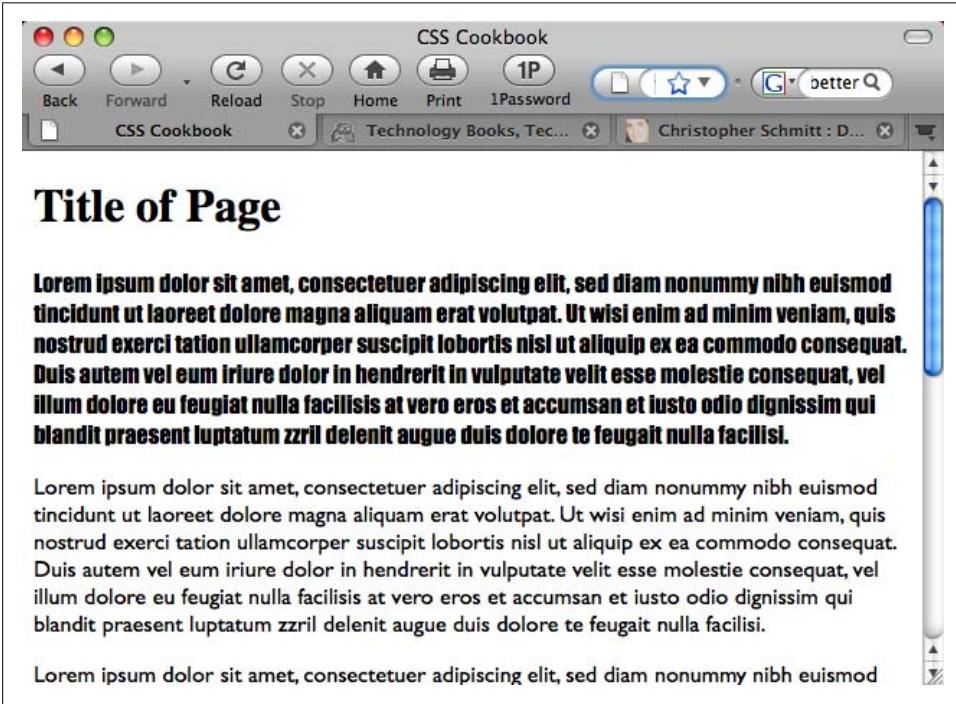
Figure 2-29. The winning CSS rule

Discussion

The origin and sorting order of CSS help a browser to determine which rules win out over others (and the `!important` declaration allows certain rules to override others). When those methods of determining which CSS rules should win fail, there is a conflict. CSS has in place a way to deal with those conflicts: the specificity of the CSS rule itself.

The higher the specificity of a CSS rule, the greater the likelihood that the CSS wins.

> The universal selector carries a specificity of 0,0,0,0. Inherited values do not have specificity.

Several CSS specificity calculators are available online to help you determine the specificity of rules. One such calculator is available at *http://www.suzyit.com/tools/specificity.php*.

See Also

Eric Meyer's post on specificity at *http://meyerweb.com/eric/css/link-specificity.html*; Molly Holzschlag's post about CSS2 and CSS 2.1 specificity at *http://www.molly.com/2005/10/06/css2-and-css21-specificity-clarified/*

2.16 Setting Up Different Types of Stylesheets

Problem

You want to provide stylesheets for different media types such as aural, print, and handheld.

Solution

Create separate external stylesheets for the different media and name them by their media, such as *print.css*, *screen.css*, and *handheld.css*. Then use the `link` element with the media type in the web page to link to these styles. Another option is to use the `@media` rule.

Here's *print.css*:

```
body {
  font: 10pt Times, Georgia, serif;
  line-height: 120%;
}
```

Here's a new file called *screen.css*:

```
body {
  font: 12px verdana, arial, sans-serif;
  line-height: 120%;
}
```

And finally, here's another file called *projection.css*:

```
body {
  font: 14px;
  line-height: 120%;
}
```

Now link to the three files from the web page, with the following lines within the `head` section. Each link has a different media type:

```
<link rel="stylesheet" type="text/css" href="/css/print.css" media="print" />
<link rel="stylesheet" type="text/css" href="/css/screen.css" media="screen" />
<link rel="stylesheet" type="text/css" href="/css/projection.css"
media="projection" />
```

You could use the `@media` rule instead to specify the different media rules within the same stylesheet:

```
<style type="text/css">
<!--
@media print {
 body {
  font: 10pt Times, Georgia, serif;
 }
}

@media screen {
 body {
  font: 12pt Verdana, Arial, sans-serif;
 }
}

@media projection {
 body {
  font-size: 14pt;
 }
}

@media screen, print, projection {
 body {
  line-height: 120%;
 }
}
-->
</style>
```

Discussion

When creating styles for printing, add them to *print.css* and only these styles will be applied during printing. This ensures that the page prints without wasting space or ink by printing images. Only devices supporting the specific media type will see their related media CSS styles. The media stylesheets don't affect the appearance of other media or the web page itself.

The `@media` rule allows you to put all the media in one stylesheet.

Figure 2-30 shows how the web page looks in its original screen format. Users don't need to print the side items, so copy the *screen.css* stylesheet and save it as a new one called *print.css*. Rather than starting from scratch, modify *screen.css* to optimize the web page for printing. The following items in *screen.css* have been changed in *print.css*:

```
#sub_banner {
 background-color: #ccc;
 border-bottom: solid 1px #999;
 font-size:.8em;
 font-style: italic;
 padding: 3px 0 3px 5px;
}
#nav1 {
```

```
  position: absolute;
  width: 30%;
  left: 60%;
  top: 100px;
  padding: 5px 5px px 5px 0;
}
#nav2 {
  position: absolute;
  width: 15%;
  left: 1%;
  top: 100px;
  padding: 5px 5px px 5px 0;
}
h1 {
  text-align: left;
  color: #fff;
  font-size: 1.2em;
  text-align: left;
  margin-bottom: 5px;
  margin-top: 5px;
}
.entry {
  padding-bottom: 20px;
  padding: 5px;
  border: solid 1px #999;
  background-color: #fcfcfc;
  margin-bottom: 25px;
}
```


Figure 2-30. How the page would look if printed without print.css

Figure 2-31 shows how the page looks with *print.css*:

```
#sub_banner {
 display: none;
}
#nav1 {
 display: none;
}
#nav2 {
 display: none;
}
h1 {
 display: none;
}
.entry {
 padding: 5px;
}
```
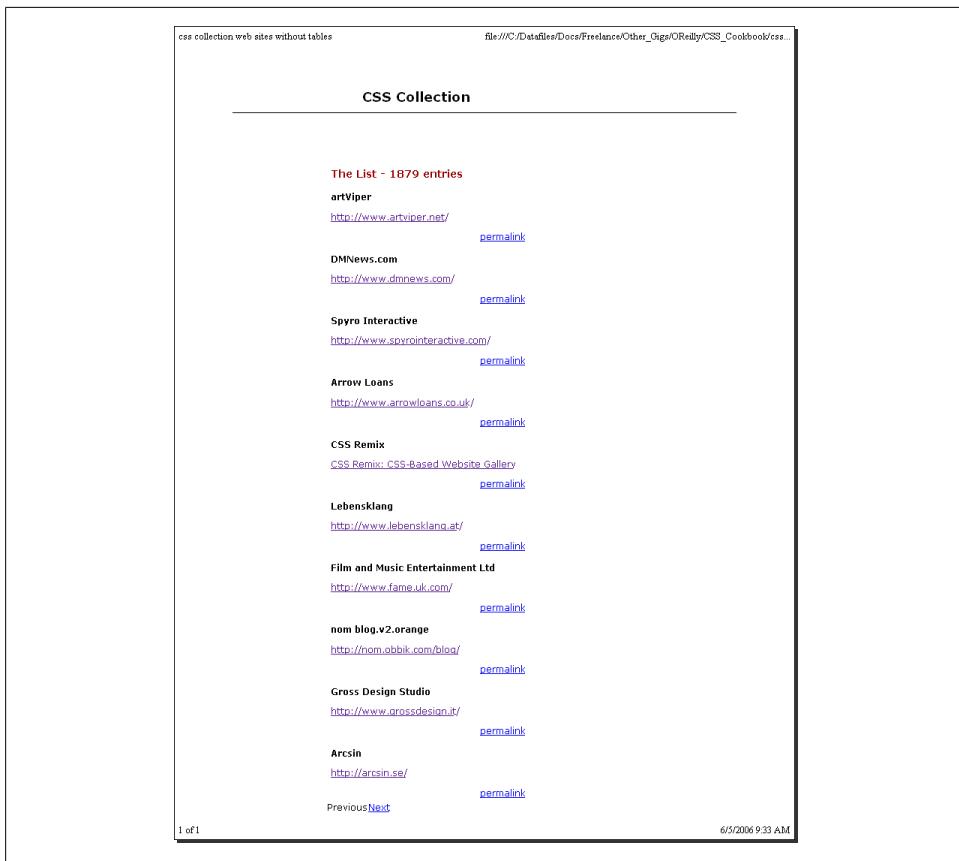

Figure 2-31. Creating print.css and adding a link to the stylesheet results in a printer-friendly web page

This takes out the sub_banner with the tagline and hides the two navigation columns. The h1 element wasn't necessary to have, and removing it saved space at the top. The entries have a light gray box, a big waste of ink, so they've been simplified to show padding only between entries.

Remember to add the link element in the HTML page:

```
<link rel="stylesheet" type="text/css" href="/css/print.css" media="print" />
<link rel="stylesheet" type="text/css" href="/css/screen.css" media="screen" />
```

That's all there is to it. CSS simplifies many things, including design for different media. Table 2-5 lists the current media types that appear in the CSS 2.1 specification.

Table 2-5. List of media types

Media type	Devices
all	Used for all devices
aural	Used for speech and sound synthesizers
braille	Used for Braille tactile feedback devices
embossed	Used for Braille printers
handheld	Used for handheld or small devices such as PDAs and smartphones
print	Used for printers and print previews
projection	Used for projected presentations
screen	Used for color monitors
tty	Used for fixed-pitch character grids such as teletypes, terminals, and portable devices with limited characters
tv	Used for television and WebTV

See Also

Chapter 10 for setting up styles for printing; the section "Media types" of the CSS 2.1 specification at *http://www.w3.org/TR/CSS21/media.html*; A List Apart's "ALA's New Print Styles" at *http://www.alistapart.com/articles/alaprintstyles*; A List Apart's "Pocket-Sized Design: Taking Your Website to the Small Screen" at *http://www.alistapart.com/articles/pocket*

2.17 Adding Comments Within Stylesheets

Problem

You want to organize and keep track of the CSS with comments.

Solution

Add /* and */ anywhere in the styles to show the start and end of a comment:

```
/* This is a comment */
a {
 text-decoration: none;
}
/* This is also a comment */
h1, h2 {
 font-size: 100%; /* This is also a comment, too */
 color: #666666;
}
```

Discussion

You might look at old code and not remember why you took certain steps with the code. Comments can explain and organize code so that you can better understand it if you review it at a later time. Comments also help those who didn't create the original code to understand its purpose. Browsers ignore content that appears between /* and */.

As you break your code into sections, comments come in handy in terms of identifying each section, such as the header, footer, primary navigation, subnavigation, and so on. Comments provide a great way to test your web pages. If you're not sure whether a style works or how it affects the page, add a comment around the style to turn it off:

```
/*
a {
 text-decoration: none;
}
*/
```

In the preceding code, the comments around `text-decoration` ensure that the text decoration (including underlining) will not take effect. Unless there are other styles for `a`, the underline appears under links until the comment is removed.

See Also

The CSS 2.1 specification on comments at *http://www.w3.org/TR/CSS21/syndata.html #comments*

2.18 Organizing the Contents of a Stylesheet

Problem

You want to know how to effectively organize contents within a stylesheet for easier management.

Solution

You can manage CSS by grouping the common visual elements of a web page together. The following list suggests the order of items grouped in a stylesheet:

1. Elements (`h1` through `h6`, `p`, `a`, `list`, `links`, `images`)
2. Typography
3. Page layout (header, content, navigation, global navigation, subnavigation, sidebar, footer)
4. Form tags (form, fieldset, label, legend)
5. Content (post, events, news)

Here are the comments from three stylesheets, with each one organizing the CSS differently:

```
/* Typography & Colors
---------------------------------- */
[css code ]

/* Structure
---------------------------------- */
[css code ]

/* Headers
---------------------------------- */
[css code ]

/* Images
---------------------------------- */
[css code ]

/* Lists
---------------------------------- */
[css code ]

/* Form Elements
---------------------------------- */
[css code ]

/* Comments
---------------------------------- */
[css code ]

/* Sidebar
---------------------------------- */
[css code ]

/* Common Elements
---------------------------------- */
[css code ]
```

Discussion

What works for one person may not work for another. The setup in the Solution is a recommendation based on a combination of experience and best practices that should work well for small to medium-size websites.

For different projects and your own personal preference, you might find a way that works better for you. Visit your favorite websites and review their stylesheets to study how they're organized.

See Also

Doug Bowman's "CSS Organization Tip 1: Flags," a method for finding rules in your CSS files, at *http://www.stopdesign.com/log/2005/05/03/css-tip-flags.html*

2.19 Working with Shorthand Properties

Problem

You want to use shorthand properties in stylesheets.

Solution

Begin with a properly marked up section:

```
<h3>Shorthand Property</h3>
<p>Combine properties with shorthand and save time, typing, and a
few bytes. Your stylesheets will also be easier to read.</p>
```

Then use just one instance of the `font` property instead of using `font-style`, `font-size`, and `font-family`:

```
h3 {
 font: italic 18pt verdana, arial, sans-serif;
}
p {
 border: 2pt solid black;
}
```

Discussion

You can toss several CSS properties in favor of shorthand properties.

The `border` property is a shorthand property that combines three properties into one. The `border` property can cover the values from the following properties:

- `border-color`
- `border-width`
- `border-style`

The `font` property is a shorthand property that combines five properties into one. The `font` property can cover the values from the following properties:

- `font-style`
- `font-size/line-height`

- font-family
- font-weight
- font-variant

Enter the values just as you would with any other property, except for `font-family` and `font-size/line height`. With `font-family`, enter the fonts in the priority you wish them to have and use a comma between each.

If you use both `font-size` and `line-height`, separate their values with a forward slash:

```
h3 {
  font: italic 18pt/20pt verdana, arial, sans-serif
}
```

For a rundown of the shorthand properties available to web developers, see Table 2-6.

Table 2-6. Shorthand properties

Property	Values	Example
background	background-color	background: url(book.gif) #999 no-repeat top;
	background-image	
	background-repeat	
	background-attachment	
	background-position	
border	border-width	border: thin solid #000;
border-left	border-style	
border-right	border-color	
border-top		
border-bottom		
font	font-style	font: 14px italic Verdana, Arial, sans-serif;
	font-variant	
	font-weight	
	font-size/line-height	
	font-family	
	caption	
	icon	
	menu	
	message-box	
	small-caption	
	status-bar	

Property	Values	Example
list-style	list-style-type	list-style: circle inside;
	list-style-position	
	list-style-image	
margin	margin-top	margin: 5px 0px 5px 10px;
	margin-right	margin: 15px 0;
	margin-bottom	margin: 5px;
	margin-left	
padding	padding-top	padding: 5px 10% 15px 5%;
	padding-right	padding: 7px 13px;
	padding-bottom	padding: 6px;
	padding-left	

See Also

The CSS 2.1 specification for border shorthand properties at *http://www.w3.org/TR/CSS21/box.html#border-shorthand-properties* and font shorthand properties at *http://www.w3.org/TR/CSS21/about.html#shorthand*; Appendix B for a full list of CSS properties

2.20 Setting Up an Alternate Stylesheet

Problem

You want to provide other style options for users who might want larger text or a different color scheme.

Solution

Use the link element with a title and link it to the alternate stylesheets. The title lets the user see what options are available when viewing the list of available styles. In Firefox, select View→Page Styles to see the list.

```
<link href="default.css" rel="stylesheet" title="default styles"
type="text/css" media="screen" />
<link href="green.css" rel="stylesheet" title="green style"
type="text/css" media="screen" />
<link href="blue.css" rel="stylesheet" title="blue style"
type="text/css" media="screen" />
```

Unfortunately, this doesn't work in Internet Explorer 6.0 or Safari.

Discussion

Alternate stylesheets work similarly to the media type stylesheets in Recipe 2.16. But instead of creating styles for media, you're providing users with multiple choices of styles for the screen. Furthermore, this technique doesn't require use of JavaScript. Some users have disabled JavaScript, which would affect a stylesheet switcher.

All you have to do is make a copy of your default stylesheet and rename it. Make the changes to the stylesheet and add the `link` element with a `title`, as shown in Figure 2-32.
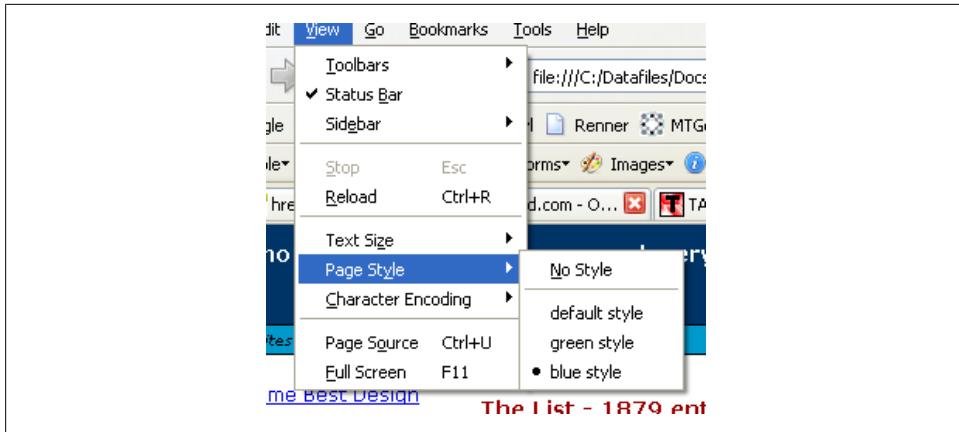
Figure 2-32. Switching stylesheets within the browser options

See Also

A List Apart's article "Invasion of the Body Switchers" by Andy Clarke and James Edwards, which shows how to create a JavaScript style switcher, at *http://www.alistapart.com/articles/bodyswitchers*; the Amit Ghaste CSS Style Switcher tutorial at *http://ghaste.com/pubs/styleswitcher.html*

2.21 Using Floats

Problem

You want to place an image on the left or right side, with text wrapping around the image instead of appearing above or below the image, as shown in Figure 2-33.
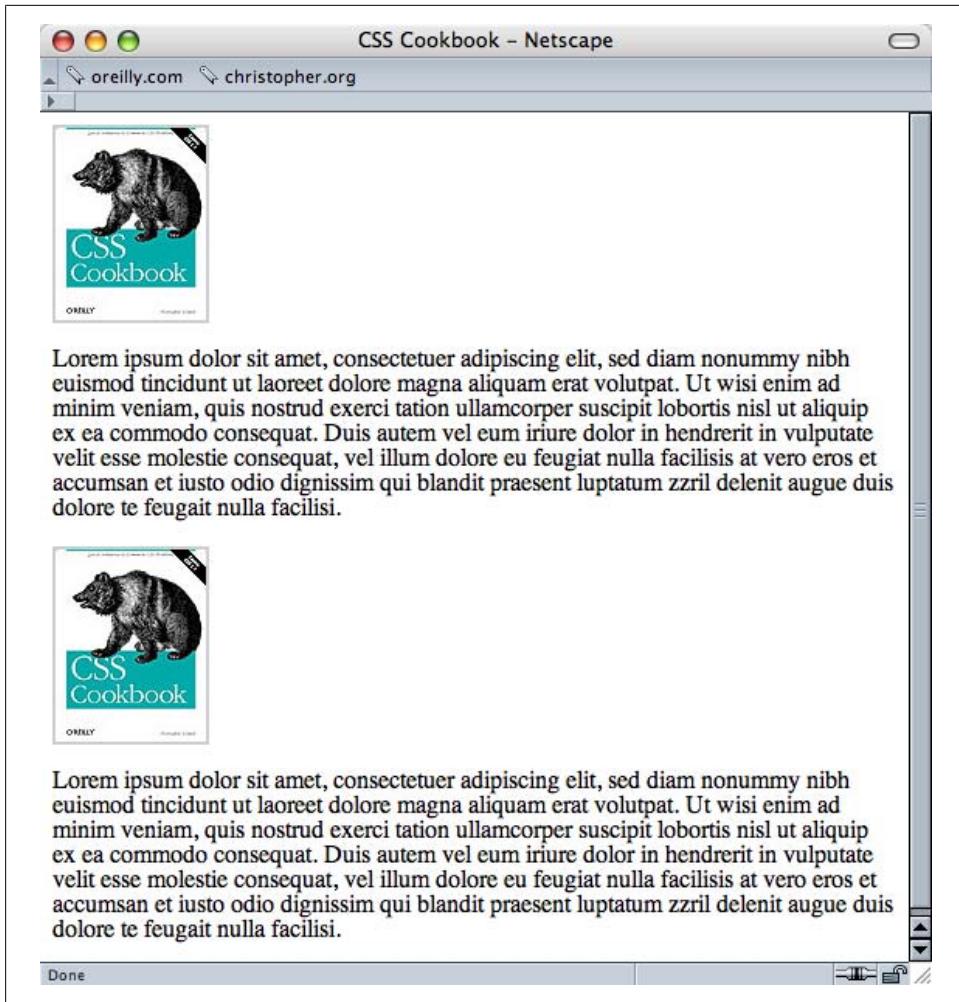
Figure 2-33. Images not wrapping around the text by default

Solution

First create class selectors for the image:

```
.leftFloat {
 float: left
}
.rightFloat {
 float: right
}
```

> Using class names that describe the presentation, as I did in this Solution, is not recommended. This is for demonstration purposes only.

Then add the class selector to the markup (see Figure 2-34):

```
<img src="csscookbook.gif" class="leftFloat" alt="cover" />
<p>This is the book cover for the <em>CSS Cookbook</em>.</p>
<img src="csscookbook.gif" class="rightFloat" alt="cover" />
<p>This is the book cover for the <em>CSS Cookbook</em>.</p>
```
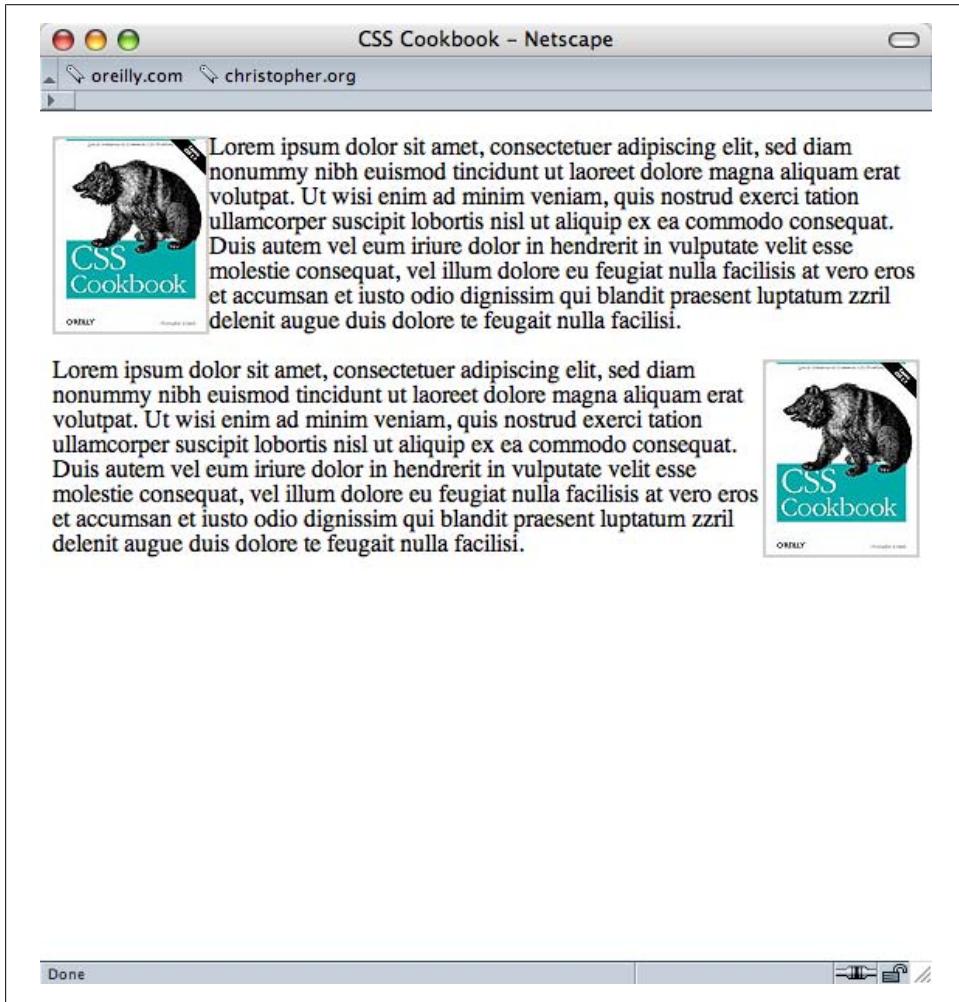

Figure 2-34. Text wrapping around the images, thanks to float

Discussion

Before standards compliance was recommended, designers used the `align` attribute with the `img` element to move images to the side with text wrapping. The W3C deprecated `align` and now recommends using `float` instead.

You can use `float`s with elements other than images to shift an item left or right from its original placement.

In Figure 2-34, the second image overlaps the paragraph referencing the first image. This looks confusing and needs to be fixed. To work around that, use `clear`:

```
p {
  clear: left;
}
```

The `clear` property tells the paragraph to appear after the end of the image flow. At the second `img`, the `clear` property pushes the image down to the first line after the previous line ends. Instead of lining up with the second `p` element, the image waits for a new line before showing up.

See Also

The W3C 2.1 specification on floats at *http://www.w3.org/TR/CSS21/visuren.html #floats*; Chapter 8, which provides recipes for using `float` with page columns; Eric Meyer's CSS/edge, which covers floats, at *http://meyerweb.com/eric/css/edge/*

2.22 Using Self-Clearing Floated Elements

Problem

You want to stop a floated element from overlapping other content, but without any reliance on other HTML elements.

Solution

First, examine a situation where a `float` is overlapping part of a layout, as shown in Figure 2-35:

```
<div>
 <img src="schmitt-csscookbook.jpg" alt="cover" />
 <p>Lorem ipsum dolor sit amet, consectetuer adipiscing elit,
    sed diam nonummy nibh euismod tincidunt ut laoreet dolore
    magna aliquam erat volutpat...
 </p>
</div>
```

Then set up the CSS rules for the sample:

```
div {
 border: 1px solid black;
```

```
 padding: 25px;
}
img {
 border-right: 1px solid #999;
 border-bottom: 1px solid #999;
 float: left;
 padding: 1px;
}
p {
 float: right;
 width: 87%;
}
```
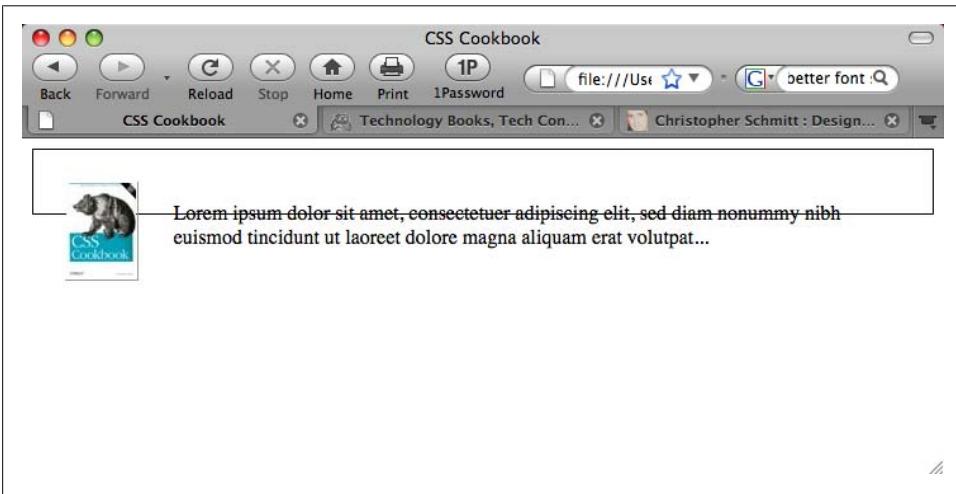

Figure 2-35. The image and paragraph overlapping the border

To force the border of the `div` element to encapsulate the floated elements, use the self-clearing float technique.

First, set up the CSS rules:

```
.clearfix:after {
 content: ".";
 display: block;
 height: 0;
 clear: both;
 visibility: hidden;
}
/* CSS rule for IE6 */
* html .clearfix {
 height: 1%;
}
/* CSS rule for IE7 */
*:first-child+html .clearfix {
 min-height: 1px;
}
```

Then add a `class` selector to the parent `div` element with the value of `clearfix`, as shown in Figure 2-36:

```
<div class="clearfix">
 <img src="schmitt-csscookbook.jpg" alt="cover" />
 <p>Lorem ipsum dolor sit amet, consectetuer adipiscing elit,
    sed diam nonummy nibh euismod tincidunt ut laoreet dolore
    magna aliquam erat volutpat...
 </p>
</div>
```
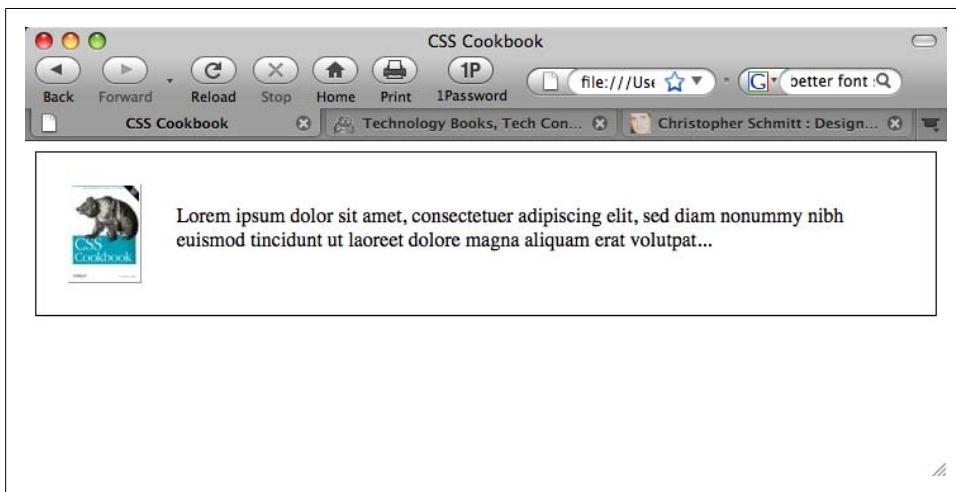

Figure 2-36. The floated elements, now cleared

Discussion

The clearing method discussed in Recipe 2.21 relies on the presence of an additional element coming right after a floated element.

Another method that web developers use is to place a `div` or `br` element after a floated element in the markup, and then set that element's `clear` property:

```
<div>
 <img src="schmitt-csscookbook.jpg" alt="cover" />
 <p>Lorem ipsum dolor sit amet, consectetuer adipiscing elit,
    sed diam nonummy nibh euismod tincidunt ut laoreet dolore
    magna aliquam erat volutpat...
 </p>
 <div style="clear: both;"></div>
</div>
```

When many hands are often touching a web document or documents, it's impractical to make sure that a wedge like this is going to be consistently used by everyone.

Self-clearing floats

The self-clearing float technique, originally published by Position is Everything (see *http://positioniseverything.net/easyclearing.html*), showed a way to clear floated elements without the additional markup.

However, Internet Explorer 7 and earlier can't execute auto-generated content through `:after` pseudo-elements.

To get around the limitations of the browser, two CSS rules are needed—one for IE7 and another for IE6—to trick the respective browsers into clearing the floated elements.

> You can tuck away these CSS rules using conditional comments so that only IE browsers see them.

Using overflow

Another method for clearing floats is to use an uncommon CSS property, `overflow`:

```
div {
 border: 1px solid black;
 padding: 25px;
 overflow: hidden;
 zoom: 1
}
```

The `overflow` property makes sure the element clears all the floats that are inside it. (The `zoom` property is for IE6, if you need it. If not, you can get rid of it.)

See Also

Recipe 2.21 for information on using floats; *http://www.sitepoint.com/blogs/2005/02/26/simple-clearing-of-floats/* for other ways to clear a float

2.23 Using Absolute Positioning

Problem

You want to position an element based on the window rather than its default position.

Solution

Use the `position` property with the `absolute` value in the stylesheet. Also use `bottom`, `left`, or both `bottom` and `left` to indicate where to position an element:

```
.absolute {
 position: absolute;
 bottom: 50px;
```

```
    left: 100px;
    }
```

Discussion

The `absolute` value places the content *out of the natural flow of the page layout* and puts it exactly where the CSS properties tell it to go within the current box or window. The sample code used in the Solution tells the browser to position the element with the `absolute` class exactly 40 pixels down from the top and 20 pixels over from the left edge of the window.

Let's look at the natural flow of an image and a paragraph, as shown in Figure 2-37.
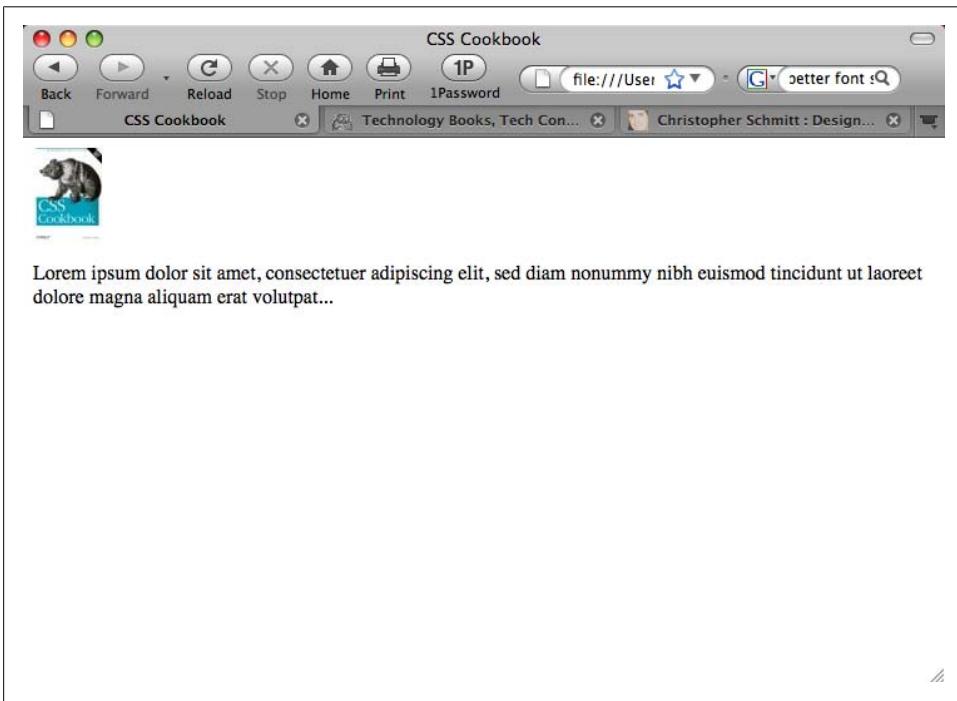
Figure 2-37. Default rendering of the content

Apply the absolute positioning to the `div` that encompasses the content by adding the `class` attribute and the `absolute` value, as shown in Figure 2-38:

```
<div class="absolute">
 <img src="csscookbook.gif" alt="cover" />
<p>Lorem ipsum dolor sit amet, consectetuer adipiscing elit,
sed diam nonummy nibh euismod tincidunt ut laoreet dolore
magna aliquam erat volutpat...
 </p>
</div>
```
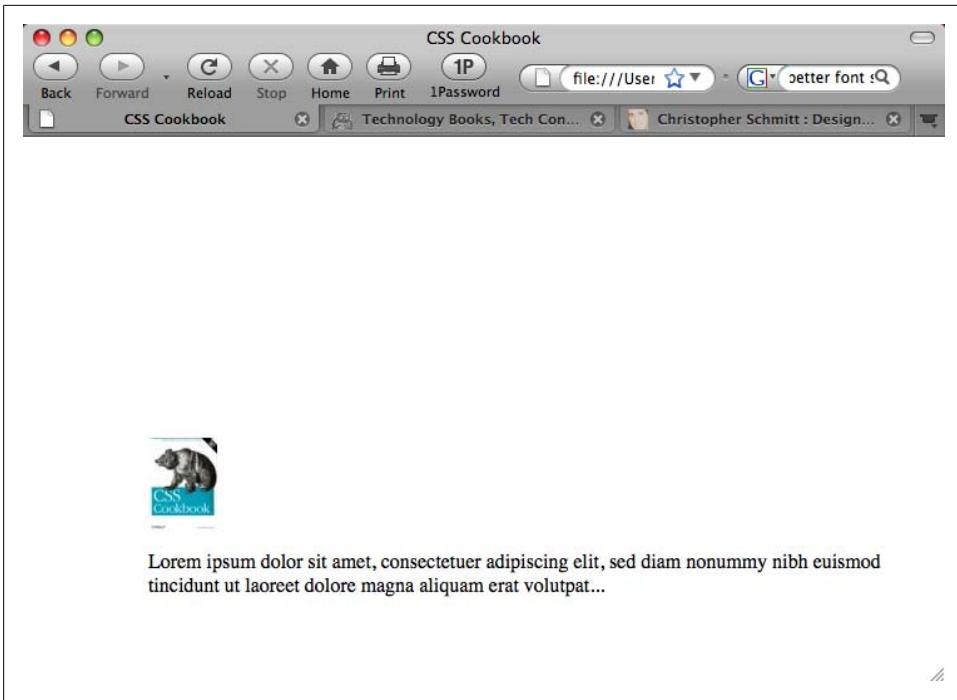
Figure 2-38. Absolute positioning, which places an element based on its location within a window

You can also use the `right` and `bottom` properties to change the absolute position. `Bottom` represents the bottom of the window, no matter how big or small you make the window.

> Here we used absolute positioning of elements to shift a block of content around to demonstrate how it works. However, you need to be careful when doing absolute positioning because absolutely positioned elements will remain in place even as flexible web page layouts change due to flexible browser and/or text resizes.

See Also

The W3C 2.1 specification on absolute positioning at *http://www.w3.org/TR/CSS21/ visuren.html#absolute-positioning*; W3Schools' tutorial on positioning at *http://www .w3schools.com/css/css_positioning.asp*

2.24 Using Relative Positioning

Problem

You want to place content based on its position in the document. In other words, the element's position is modified relative to its natural position as rendered by the browser.

Solution

Use the `position` property with the `relative` value in the stylesheet. Also add `top`, `left`, or both `top` and `left` to indicate where to position the element.

Using the following CSS rule on the image, the image was able to move over the paragraph content, as shown in Figure 2-39:

```
.relative {
 position: relative;
 top: 100px;
 left: 20px;
}
```
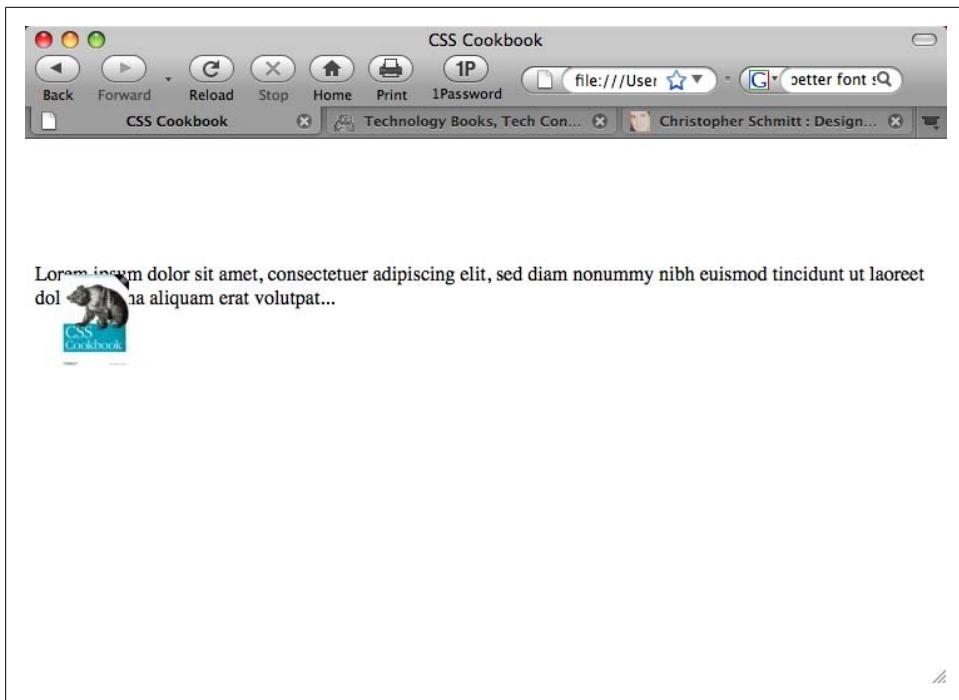

Figure 2-39. Relative positioning, which places an element based on its location within the document's natural flow

Discussion

Unlike absolute positioning, the sample code doesn't start at the top and left edges of the window. Instead, it begins where p would be if left alone.

The code tells the browser to position the paragraph 100 pixels down from the top and 20 pixels over from the left edge of the original paragraph's position instead of the edge.

With absolute positioning, the content is placed exactly where the properties state it should go from the edges in the current box.

See Also

The W3C 2.1 specification on relative positioning at *http://www.w3.org/TR/CSS21/visuren.html#relative-positioning*; W3Schools' tutorial on positioning at *http://www.w3schools.com/css/css_positioning.asp*

2.25 Using Shackling Positioning

Problem

You want to move an element within the constraints of another element's dimensions. For example, you want to place the image of the book cover within the confines of the shaded box and not the upper-lefthand corner of the browser's viewport, as shown in Figure 2-40.
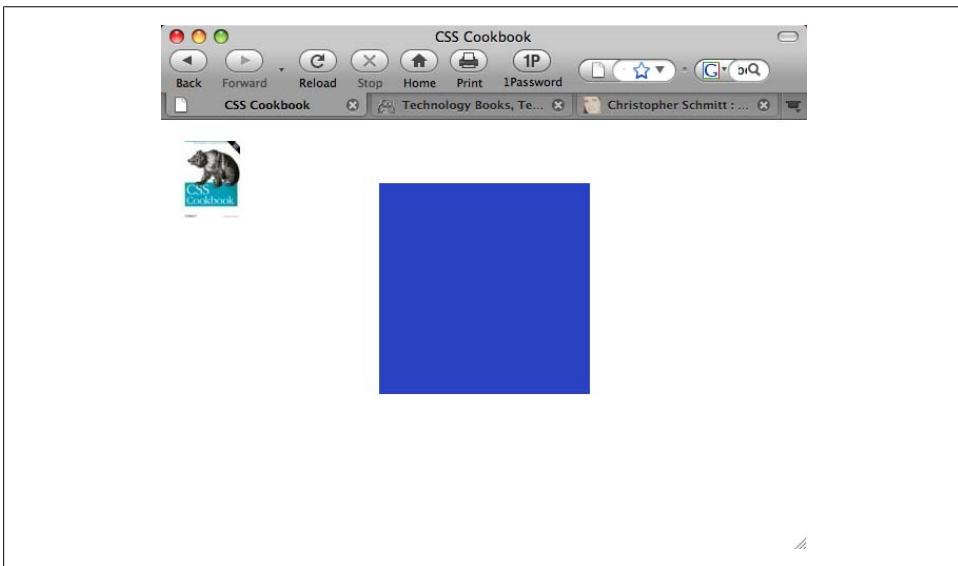
Figure 2-40. An image positioned absolutely to the upper-left corner of the browser's viewport

Solution

First set the `position` property to a value of `relative` for the parent element:

```
#content {
 position: relative;
 width: 200px;
 height: 200px;
 margin: 10% auto;
 background: #2942c4;
}
```

Then set the child element to be positioned absolutely using the offset properties `top`, `right`, `bottom`, and `left`, to move the element within the confines of the parent element, as shown in Figure 2-41:

```
#positioned {
 position: absolute;
 top: 20px;
 left: 20px;
}
```
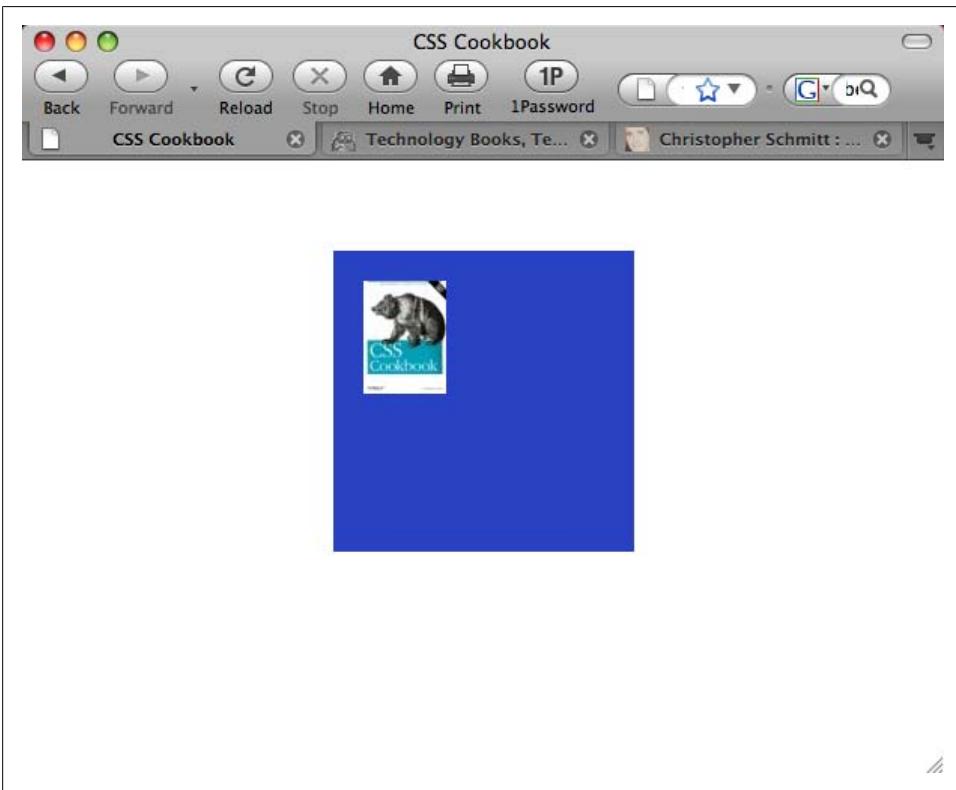

Figure 2-41. The image now shackled to the dimensions of its immediate parent element

Discussion

When an element is absolutely positioned, it's taken out of the normal flow and positioned according to its containing element. In most cases, this is going to be the base, or root element, in the web document. That's typically going to be the `html` element.

However, the context of that containing element can change.

If a parent element is also positioned, the absolutely positioned element doesn't get affixed to the root element (typically the upper-left corner of the viewport, if no offset properties are set). This effect is called *changing the context of the parent element*. I call it *shackling* because it's shorter and I have a life to live.

See Also

Doug Bowman's article, "Making the absolute, relative," at *http://stopdesign.com/archive/2003/09/03/absolute.html*

2.26 Stacking Elements with z-index

Problem

You have a positioned element overlapping another element, blocking it from view.

Solution

Use the `z-index` property in conjunction with a `position` property set to `absolute`, `relative`, or `fixed`:

```
div.image {
  position: relative;
  z-index: 20;
  width: 13px;
  height: 14px;
  background-image: url(star.gif);
  background-repeat: no-repeat;
}
```

Discussion

Digital images are composed of layers. The layer on top hides whatever is on the layers below it. This analogy also holds true for the `z-index` property. An element with a higher `z-index` value overlaps an element with a lower `z-index` value.

The `z-index` property works when the element is positioned with a value of `absolute`, `relative`, or `fixed`. Without the appropriate position property, `z-index` is not applied.

When you're using more than one element with the `z-index` property, try to use values factored by 10 (e.g., 10, 20, 30) instead of 1, 2, 3, and so on. This approach allows you to fit in other, unplanned elements in the stacking order without having to reset their values.

See Also

The CSS2 specification for the `z-index` property at *http://www.w3.org/TR/CSS2/visuren .html#z-index*

2.27 Validating CSS Rules

Problem

You want to make sure your CSS rules aren't maligned with typos.

Solution

Go to *http://jigsaw.w3.org/css-validator/*, as shown in Figure 2-42, and enter the URI of the page to be validated.

You can enter code for testing via two additional methods: by uploading a CSS file or by entering the CSS rules.

Discussion

Validating CSS is different from validating HTML in that you don't declare what kind of DOCTYPE is being used.

Although numerous tools on the market have built-in validators (e.g., Adobe Dreamweaver), the W3C CSS Validator is the one that is usually up-to-date and provides better feedback, especially with the CSS3 specification.

> If CSS3 rules are being used in the stylesheet, be sure to select "CSS level 3" from the profile select menu. As of this writing, CSS rules are checked against only the CSS 2.1 specification by default.

Creating a CSS validator bookmarklet

Take any page you visit on the Web directly to the W3C's CSS Validator through a bookmarklet. A *bookmarklet* is a tiny piece of JavaScript tucked away in the Address portion of a bookmark.

Figure 2-42. Entering a web address for CSS validation

Create a new bookmark, name it "CSS Validator," and then replace whatever is in the address field with this line:

```
javascript:void(document.location='http://jigsaw.w3.org/css-
validator/validator?profile=css21&usermedium=all&warning=1&lang=en&uri=
'+escape(document.location))
```

When you visit another site, clicking on the bookmarklet runs the page currently loaded in the browser through the CSS Validator.

See Also

A Firefox extension for passing a currently viewed page to the W3C CSS Validator into a new browser tab, available at *https://addons.mozilla.org/en-US/firefox/addon/2289*

Web Typography

3.0 Introduction

Before CSS, web developers used `font` tags to set the color, size, and style of text on different parts of a web page:

```
<font face="Verdana, Arial, sans-serif" size="+1" color="blue">
 Hello, World!
</font>
```

Although this method was effective for changing the appearance of type, the technique was limiting.

Using multiple `font` tags across many, many pages resulted in time-consuming updates, inflated the overall file size of the web document, and increased the likelihood that errors would occur in the markup. CSS helps to eliminate these design and maintenance problems.

First set content within a `p` element:

```
<p>Hello, World!</p>
```

Then set styles in the head of the document to dictate the look of the paragraph:

```
<style type="text/css" media="all">
 p {
  color: blue;
  font-size: small;
  font-family: Verdana, Arial, sans-serif;
 }
</style>
```

Through this technique, the paragraph's structure and its visual presentation are separated. Because of this separation, the process of editing and maintaining a website's design, including typography, is simplified immensely. You can modify the style in a stylesheet without having to make changes at the content level.

In addition, web developers get more editing capabilities over previous techniques, as well as control over typography. Besides setting the color, style, and size of fonts, this chapter also covers techniques for setting initial caps, creating visually compelling pull quotes, modifying leading, and more.

3.1 Specifying Fonts

Problem

You want to set the typeface of text on a web page.

Solution

Use the `font-family` property:

```
body {
 font-family: Georgia, Times, "Times New Roman", serif;
}
```

Discussion

You can specify the fonts you want the browser to render on a web page by writing a comma-delimited list for the value of the `font-family` property. If the browser can't find the first font on the list, it tries to find the next font, and so on, until it finds a font.

If the font name contains spaces, enclose the name with single or double quotation marks.

At the end of the list of font choices, you should insert a generic font family. CSS offers five font family values to choose from, as shown in Table 3-1.

Table 3-1. Font family values and examples

Generic font family values	Font examples
serif	Georgia, Times, "Times New Roman", Garamond, "Century Schoolbook"
sans-serif	Verdana, Arial, Helvetica, Trebuchet, Tahoma
monospace	Courier, "MS Courier New", Prestige
cursive	"Lucida Handwriting", "Zapf-Chancery"
fantasy	Comic Sans, Whimsy, Critter, Cottonwood

All web browsers contain a list of fonts that fall into the five families shown in Table 3-1. If a font is neither chosen via a CSS rule nor available on the user's computer, the browser uses a font from one of these font families.

Problem finding fonts

The most problematic generic font value is `fantasy` because this value is a catchall for any font that doesn't fall into the other four categories. Designers rarely use this font because they can't know what symbols will be displayed!

Another problematic generic value is `cursive` because some systems can't display a cursive font. If a browser can't use a cursive font, it uses another default font in its place. Because text marked as `cursive` may not actually be displayed in a cursive font, designers often avoid this generic font value as well.

If you want to use an unusual font that might not be installed on most people's machines, the rule of thumb is to set the last value for the `font-family` property to `serif`, `sans-serif`, or `monospace`. This approach maintains at least some legibility for the user viewing the web document.

Inheriting fonts throughout a web page

You don't have to set the same properties for every tag you use. A child element *inherits*, or has the same property values of, its parent element if the CSS specification that defines a given property can be inherited. For example, if you set the `font-family` property to show a serif font in a paragraph that contains an `em` element as a child, the text in the `em` element is also set in a serif font:

```
<p style="font-family: serif;">The water fountain
with the broken sign on it is <em>indeed</em> broken.</p>
```

Inheritance doesn't occur under *two* circumstances.

One is built into the CSS specification and concerns elements that can generate a box. Elements such as `h2` and `p` are referred to as *block-level elements* and can have other properties such as margins, borders, padding, and backgrounds, as shown in Figure 3-1.
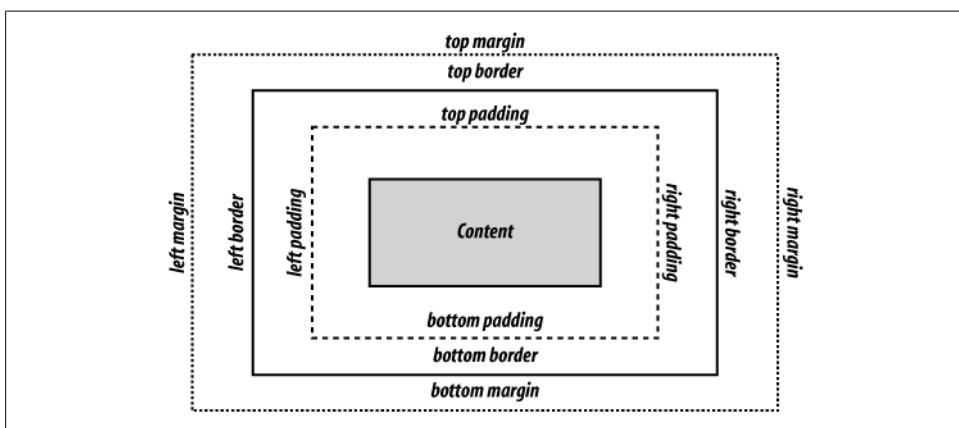
Figure 3-1. The box model for a block-level element

Because these properties aren't passed to child block-level elements, you don't have to write additional rules to counter the visual effects that would occur if they were passed. For example, if you applied a margin of 15% to a body element, that rule would be applied to every h2 and p element that is a child of that body element. If these properties were inherited, the page would look like that shown in Figure 3-2.
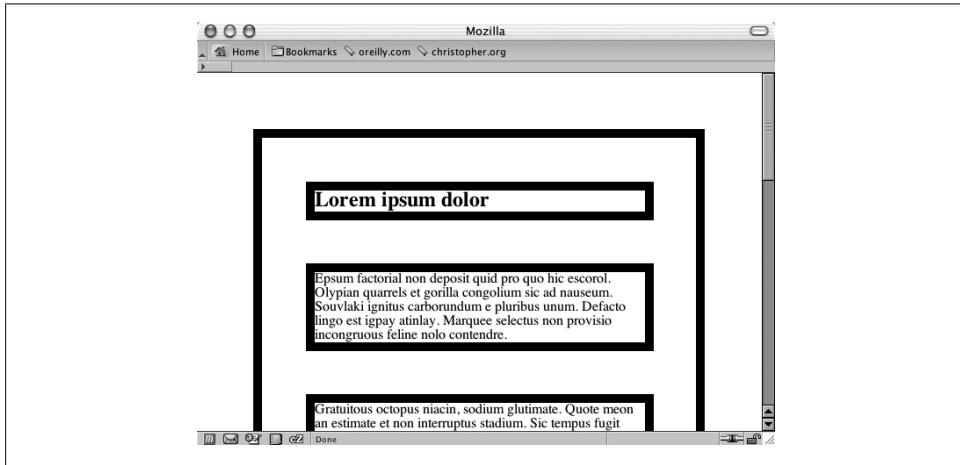
Figure 3-2. Hypothetical mock-up of margins and border properties being inherited

Because certain properties are defined to be inheritable and others aren't, the page actually looks like that shown in Figure 3-3 in a modern CSS-compliant browser.

Figure 3-3. How the page looks when block-level elements don't inherit certain properties

The other circumstance under which inheritance doesn't work is, of course, if your browser doesn't follow the CSS specification. Thankfully, this hasn't happened in any recent browser releases, as the most notable example of this came from the old Netscape Navigator 4 browser.

See Also

The CSS 2.1 specification for inheritance at *http://www.w3.org/TR/CSS21/cascade.html#inheritance*; the CSS 2.1 specification for `font-family` values at *http://www.w3.org/TR/CSS21/fonts.html#propdef-font-family*; more about CSS and Netscape 4 issues at *http://www.mako4css.com/cssfont.htm*

3.2 Using Web-Safe Fonts

Problem

You want to specify fonts that are on most of your site visitors' machines.

Solution

Use what are commonly referred as *web-safe fonts*, which are type files that are preinstalled on Macintosh and Windows operating systems.

> If you use Linux, you can install Microsoft TrueType fonts by installing the *msttcorefonts* package. For more information, see *http://embraceubuntu.com/2005/09/09/installing-microsoft-fonts/*.

Here are examples of sans serif web-safe font stacks:

```
font-family: Verdana, Geneva, sans-serif;
font-family: Arial, Helvetica, sans-serif;
font-family: Tahoma, Geneva, sans-serif;
font-family: "Trebuchet MS", Area, Helvetica, sans-serif;
font-family: "Lucida Sans Unicode", "Lucida Grande", sans-serif;
```

Here are examples of serif web-safe font stacks:

```
font-family: Georgia, "Times New Roman", Times, serif;
font-family: "Palatino Linotype", "Book Antigua", Palatino, serif;
font-family: "MS Serif", New York, serif;
```

The following are monospace web-safe font stacks:

```
font-family: "Courier New", Courier, monospace;
font-family: "Lucida Console", Monaco, monospace;
```

This is a cursive web-safe font stack:

```
font-family: "Comic Sans MS", cursive;
```

Discussion

You can find approximately 13 fonts on both Windows and Macintosh operating systems, as shown in Table 3-2.

Table 3-2. Cross-platform fonts

Windows/Mac OS font	Font family	Example
Arial	Sans serif	ABCDEFGHIJKLMNOPQRSTUVWXYZ abcdefghijklmnopqrstuvwxyz 0123456789
Arial Black	Sans serif	ABCDEFGHIJKLMNOPQRSTUVWXYZ abcdefghijklmnopqrstuvwxyz 0123456789
Comic Sans MS	Cursive	ABCDEFGHIJKLMNOPQRSTUVWXYZ abcdefghijklmnopqrstuvwxyz 0123456789
Courier New	Monospace	ABCDEFGHIJKLMNOPQRSTUVWXYZ abcdefghijklmnopqrstuvwxyz 0123456789
Georgia	Serif	ABCDEFGHIJKLMNOPQRSTUVWXYZ abcdefghijklmnopqrstuvwxyz 0123456789
Helvetica	Sans serif	ABCDEFGHIJKLMNOPQRSTUVWXYZ abcdefghijklmnopqrstuvwxyz 0123456789
Impact	Sans serif	ABCDEFGHIJKLMNOPQRSTUVWXYZ abcdefghijklmnopqrstuvwxyz 0123456789
Tahoma	Sans serif	ABCDEFGHIJKLMNOPQRSTUVWXYZ abcdefghijklmnopqrstuvwxyz 0123456789
Times	Serif	ABCDEFGHIJKLMNOPQRSTUVWXYZ abcdefghijklmnopqrstuvwxyz 0123456789
Times New Roman	Serif	ABCDEFGHIJKLMNOPQRSTUVWXYZ abcdefghijklmnopqrstuvwxyz 0123456789
Trebuchet MS	Sans serif	ABCDEFGHIJKLMNOPQRSTUVWXYZ abcdefghijklmnopqrstuvwxyz 0123456789
Verdana	Sans serif	ABCDEFGHIJKLMNOPQRSTUVWXYZ abcdefghijklmnopqrstuvwxyz 0123456789

Windows/Mac OS font	Font family	Example
Wingdings	(symbol)	✂✁✄✆✇✈✉✌✍✎✏☺☹☻❀❁❂❃❄❅❆❇❈❉

Courier, Helvetica, and Times are installed on most X11 Unix/Linux systems. The other fonts listed as web safe for both Windows and Mac OS X in Table 3-2 do not commonly appear.

Extending web-safe font listings

The popular productivity software applications Microsoft Office and Apple iWork install additional font files. Assuming a large number of computer users have one of these software applications installed on their machines (depending on the operating system), it is possible to extend the web-safe font list.

Web designer Jason Cranford Teague did just that. Researching the font listings for the software applications, he composed a directory listing extended web-safe fonts sortable by font name, weight, OS, or rank (the likelihood it's installed on a user's machine), as shown in Figure 3-4. To view the list, see *http://tr.im/xGGi*.
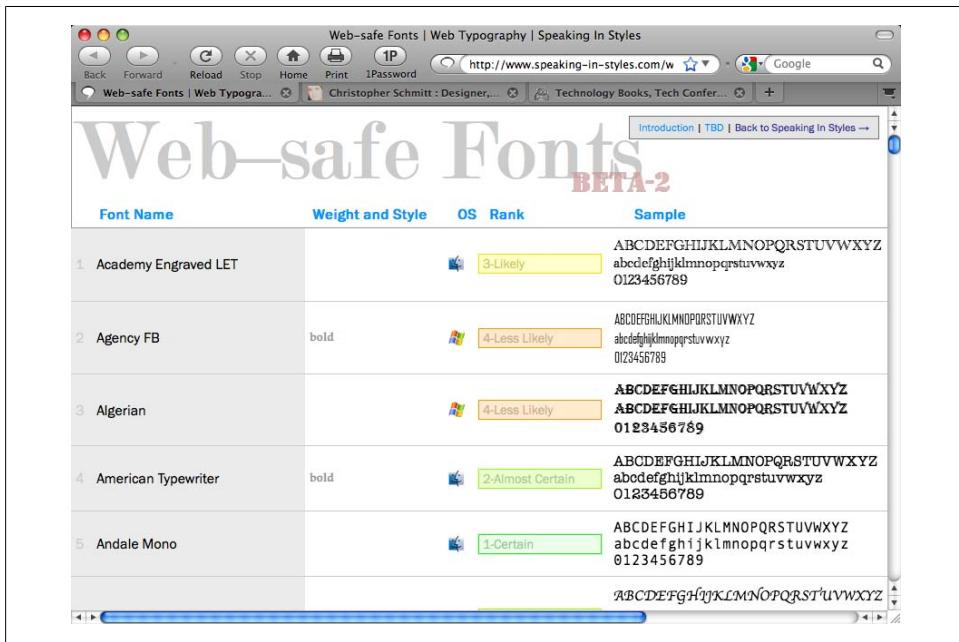
Figure 3-4. Directory of extended web-safe fonts

More robust stacking

Although simply stating the web fonts we know are on people's machines is a good solution for cross-platform development, the `font-family` property allows web designers to select fonts beyond just the basics. So, don't limit web page designs to a handful of typefaces.

For example, Gill Sans is an excellent sans serif font; however, it's not commonly installed on computers. To create a font stack that takes into account a desire to have Gill Sans in the web page design, but provide alternatives, use this CSS code:

```
p {
  font-family: "Gill Sans", Trebuchet, Calibri, sans-serif;
}
```

Design Strategist Nathan Ford explores this approach and offers more potential font stacks in his blog post "Better CSS Font Stacks" (see *http://unitinteractive.com/blog/2008/06/26/better-css-font-stacks/*).

See Also

The Web Safe Fonts Preview at *http://www.fonttester.com/web_safe_fonts.html*

3.3 Setting an Ampersand Flourish

Problem

You want a stylish ampersand for a heading instead of the default web-safe font ampersand.

Solution

First apply a `span` element around the ampersand within the heading:

```
<h1>The Lorem Ipsum <span class="amp">&amp;</span> Dolor</h1>
```

Then set the font stack for the class selector to include fonts with stylish ampersand characters, as shown in Figure 3-5:

```
span.amp {
  font-family: "Goudy Old Style", "Palatino", "Book Antiqua", serif;
  font-style: italic;
  font-weight: normal;
}
```

Discussion

To type an ampersand within the text of a web page, use its HTML entity name, `&`. HTML entities are coded variations of special characters, such as the less-than (<) or greater-than (>) signs, to keep the browser from rendering the characters like markup.
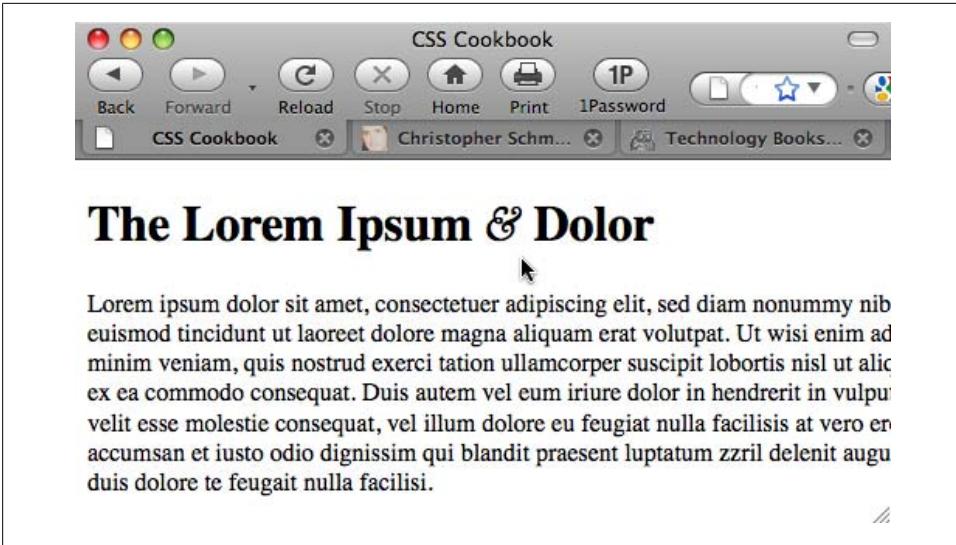
Figure 3-5. The ampersand style changes to a more distinguished look

> To add a less-than and greater-than sign in the text of a web document, use `<` and `>`, respectively.

Styling ampersands

Typographer Robert Bringhurst suggests in his book, *The Elements of Typographic Style* (Hartley and Marks), to use the best possible ampersand available when working with text. He also states that often the italic versions of typefaces contain better ampersand forms than the normal or roman counterpart.

Web designer Dan Cederholm approached this tenet for web design and even researched ampersands in various typefaces found on both Windows and Macintosh operating systems (see *http://simplebits.com/notebook/2008/08/14/ampersands.html*), as shown in Figure 3-6.

See Also

http://htmlhelp.com/reference/html40/entities/latin1.html for a listing of HTML entities; Richard Rutter's *The Elements of Typographic Style Applied to the Web* at *http://webtypography.net/*
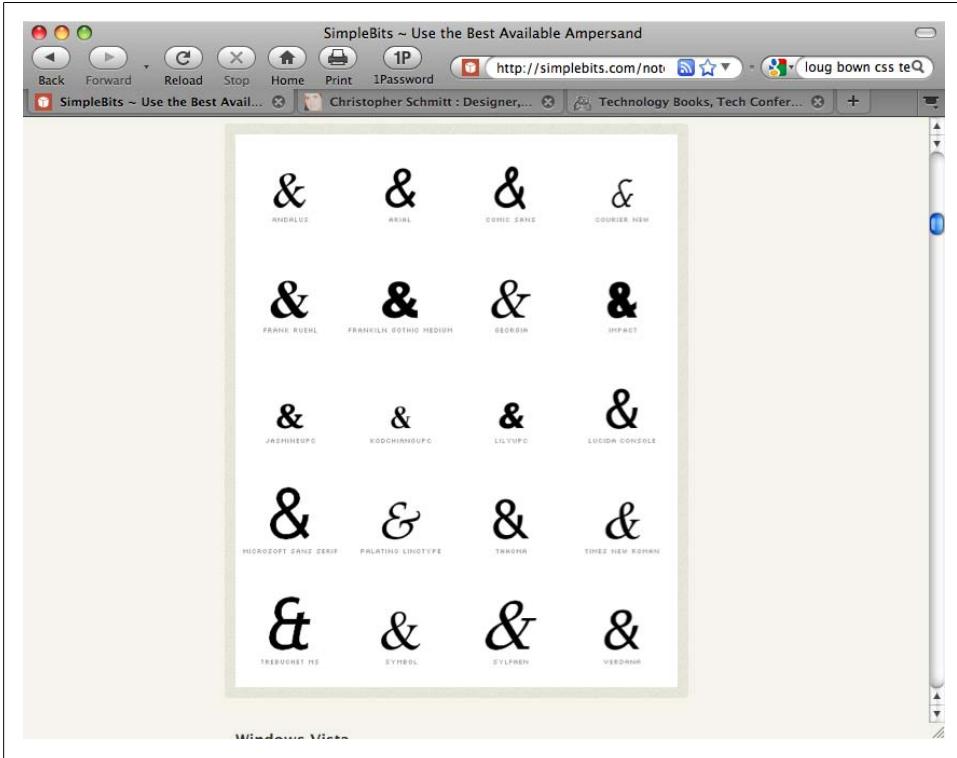
Figure 3-6. A directory of fonts with stylized fonts

3.4 Embedding Font Files

Problem

You want to use a font file in your web page, as shown Figure 3-7, with the Museo typeface.

Solution

Use the `@font-face` rule to assign a `font-family` name:

```
@font-face {
 font-family: "Museo 300";
}
```

Then associate the font file and file type:

```
@font-face {
 font-family: "Museo 300";
 font-style: normal;
 font-weight: normal;
```

```
    src: url("fonts/Museo300-Regular.otf") format("opentype");
}
```

Next, place the embedded font's `font-family` value at the start of the font stack:

```
h2 {
  font-family: "Museo 300", Verdana, Geneva, sans-serif;
  font-weight: normal;
}
```
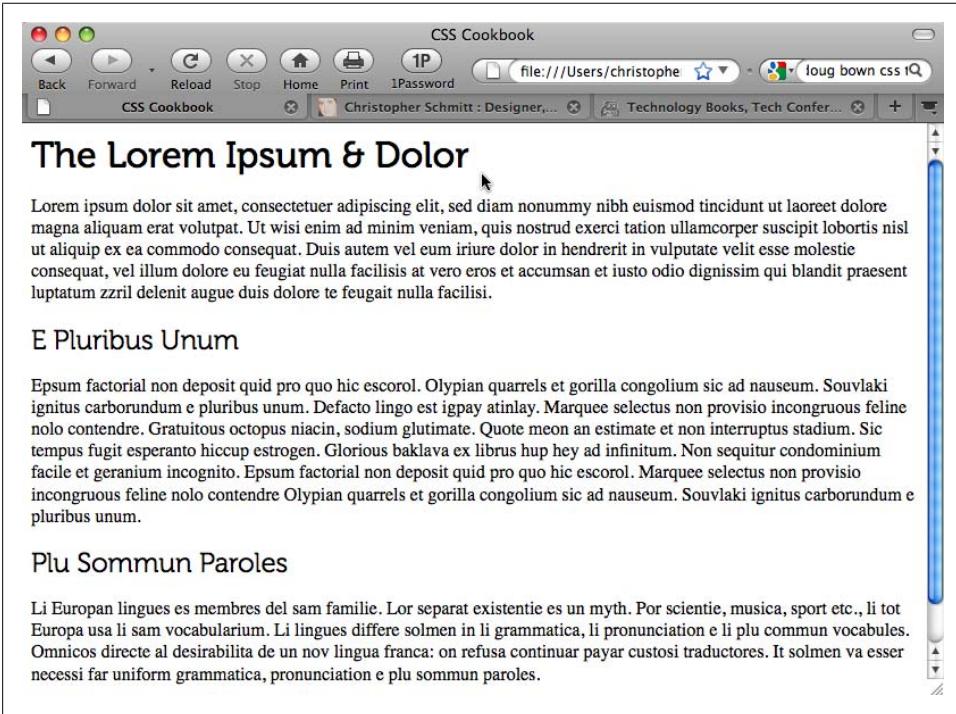

Figure 3-7. Stylized ampersand cited through a font stack

Discussion

The specification for font embedding has been part of the CSS2 specification since 1998. Internet Explorer for Windows has supported `@font-face` since version 4, but the IE browser supports only the Embedded OpenType Font format (*.eot*), which contains Digital Rights Management (DRM) code.

Other open file types for font embedding are supported in Safari 3.1 and later, Opera 10 and later, and Firefox 3.5 and later for the OpenType Face (*.otf*) and TrueType Format (*.ttf*), as shown in .

Table 3-3. Browser file type support

	.ttf	.otf	.eot
Safari 3.1 and later	Y	Y	
Opera 10 and later	Y	Y	
Firefox 3.5 and later	Y	Y	
IE4 and later			Y

> There is a new file format, Web Open Font Format (WOFF), that shows some promise. Support is included in Firefox 3.6. For more information, see *http://hacks.mozilla.org/2009/10/woff/*.

Creating cross-browser embedding

To convert a font file to an *.eot* file for cross-browser support, Microsoft provides an application called Web Embedding Fonts Tool, or WEFT (see *http://www.microsoft .com/typography/WEFT.mspx*). However, although the tool works, it has not been updated in some time. Be sure to read the tutorial closely.

To code for cross-browser font embedding, the `@font-face` rule allows for referencing multiple files:

```
@font-face {
  font-family: "Fontin Sans";
  src: url("fonts/font-file.otf")format("opentype"),
    url("fonts/font-file.eot") format("embedded-opentype");
}
```

This method also allows for linking to alternative locations in case one web server goes down:

```
@font-face {
  font-family: "Museo 300";
  font-style: normal;
  font-weight: normal;
  src: url("http://example.com/fonts/font-file.otf")format("opentype"),
    url("http://example.com/fonts/font-file.eot") format("embedded-opentype"),
    url("http://csscookbook.com/fonts/font-file.otf")format("opentype"),
    url("http://csscookbook.com/fonts/font-file.eot")
  format("embedded-opentype");
}
```

The problem with embedded fonts

As of this writing, a number of vendors that sell fonts do not license their files for embedding in web pages. If they do sell a license for the font, the cost is relatively prohibitive. (Embedding fonts is different from making an image with type set in it and placing that image on a web page. That is still legal to do, if you bought the fonts you are using to create the images in the first place.)

Although the *.eot* format was supposed to allow typographers to help control their digital rights with work, embedding type has not taken off yet.

The typographers' concerns are based on the fact that copying fonts from the embedding technique is relatively easy and takes away from their livelihood, which are true and valid points—especially since this is the type of behavior the Recording Industry Association of America (RIAA) has been battling since Napster, and photographers have been battling since the Mosaic browser introduced the `img` element.

Some typographers are finding a way to sell fonts and still allow their fonts to be available for embedding. For example, typographer Jos Buiveng, whose font is used in the Solution, releases a few fonts in a font family for free (see *http://www.josbuivenga.demon .nl/*). To obtain the additional weights to complete the set, you pay a small fee. Some other typographers, such as Fonthead Design (see *http://fonthead.com/*), allow for embedding simply as part of the typical license when buying their fonts.

For a list of free fonts available for embedding, see *http://www.fontsquir rel.com/*.

Third-party workaround

A number of third-party solutions allow font embedding to occur without people stealing the files. Web designer Richard Rutter proposed such a solution in July 2008 (see *http://clagnut.com/blog/2166/*):

> [D]esigners do not necessarily have to upload the font file to their own web server. They can link to a font file on another server. And this is where the real opportunity lies.
>
> When you embed a Google map on your web page, you don't download a bunch of map images from Google and stick them on your server, you link to Google which then serves up the maps to registered domains. The same approach can be applied to fonts. Font foundries could license their fonts for embedding and serve those fonts only to registered websites, using their own hosted system or via a trusted third party.

New services such as Typekit (see *http://blog.typekit.com/2009/07/21/serving-and-pro tecting-fonts-on-the-web/*) and Fontdeck (see *http://fontdeck.com/*) aim to do just that. For a small recurring fee you can have a professionally crafted typeface on your website that appeases the type vendors as well as makes font embedding easy to do.

Other techniques

Other alternatives to placing different typefaces into web page designs include Flash and images.

sIFR 3 is the name for a type workaround that uses Flash and JavaScript to include fonts without embedding. For more information, see *http://wiki.novemberborn.net/ sifr3/How+to+use*.

Another solution is to set custom fonts in pages, to replace HTML text with images. For more information, see Recipe 4.20.

See Also

The `@font-face` rule in the CSS specification at *http://www.w3.org/TR/2008/REC-CSS2 -20080411/fonts.html#font-descriptions*; Paul Irish's "Bulletproof @font-face syntax" blog post at *http://tr.im/Gxhf*

3.5 Forcing a Break on Really Long Words

Problem

You want to force a word break on a long word (or a long string of characters).

Solution

Use the `word-wrap` property with a value of `break-word`, as shown in Figure 3-8:

```
p {
 border: 1px solid black;
 width: 150px;
 padding: 12px;
}
p.break {
 word-wrap: break-word;
}
```
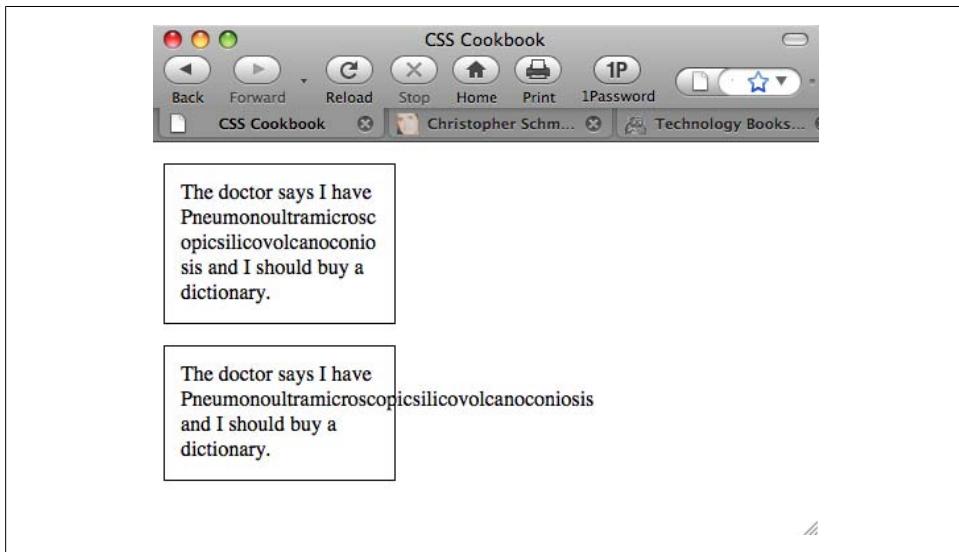

Figure 3-8. The longest word in the dictionary, split and wrapped within a border

Discussion

Appearing in the CSS3 specification, the `word-wrap` property was first used in Internet Explorer. Safari and Firefox 3.5 have since adopted it.

The default value of `word-wrap` is `default`, which would allow the normal behavior of a long word to break the confines of the box.

See Also

The CSS3 specification for `word-wrap` at *http://www.w3.org/TR/css3-text/#word-wrap*; Recipe 3.11 for clipping long passages of text

3.6 Specifying Font Measurements and Sizes

Problem

You want to set the size of type used on a web page.

Solution

Set the values of fonts using the `font-size` property:

```
p {
 font-size: 0.9em;
}
```

Discussion

The `font-size` property can take on different values and several units. In the Solution, I used em units. Other units are also available, such as percentages.

Setting the size of the font with percentages causes the browser to calculate the size of the font based on the size of the parent element. For example, if the font size for the body is set to 12 pixels and the font size for the `p` element is set to 125%, the font size for the text in paragraphs is 15 pixels.

You can use percentages, length units, and `font-size` keywords to set type size.

Length units

Length units fall into two categories: absolute and relative. Absolute length units include the following:

- Inches (in)
- Centimeters (cm)
- Millimeters (mm)

- Points (pt)
- Picas (pc)

A point, in terms of the CSS specification, is equal to 1/72 of an inch, and a pica is equal to 12 points.

A negative length value such as `-25cm` for the `font-size` property is not allowed.

Relative units

Relative units set the length of a property based on the value of another length property. Relative length units include the following:

- Em
- X-height (ex)
- Pixels (px)

Em units refer to the default font size set in the preference of the user's browser, and *x-height* (ex) refers to the height of the lowercase letter *x* in the font.

A *pixel* is the smallest dot that can be made on a computer screen.

Setting the size of fonts to 0 or a negative value

The CSS specification doesn't dictate how browser vendors should treat text when the `font-size` property is set to a value of 0. Therefore, different browsers interpret the value unpredictably.

For example, such text isn't visible in the Firefox or Mozilla browser. In Internet Explorer for Macintosh and Safari, the text isn't hidden, but rather is displayed at the default value of the font size. The Opera browser displays the text at a smaller but still legible size. And Safari 4 for Macintosh sets the type size to a small, illegible, but still visible line of text that appears to be equal to the size of 0.1 em, as shown in Figure 3-9.

If you want to make text invisible, use the `visibility` or `display` CSS property instead of setting the size of fonts to zero:

```
p {
 display: none;
}
```

A negative value for `length`, such as `-25cm`, for the `font-size` property isn't allowed.

See Also

The CSS 2.1 specification for `font-size` at *http://www.w3.org/TR/CSS21/fonts.html #font-size-props*
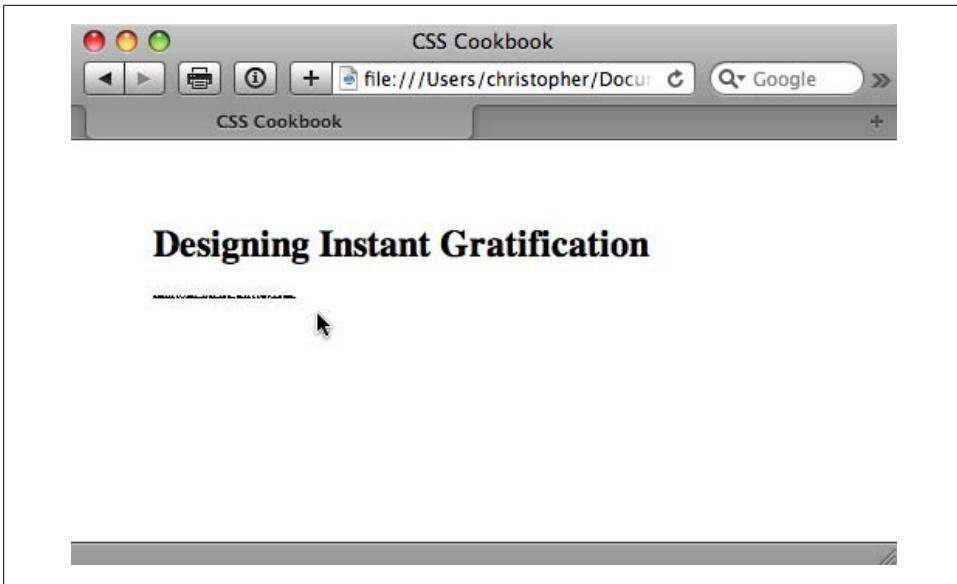
Figure 3-9. Safari 4 for Macintosh showing illegible type when the font size is set to zero

3.7 Gaining More Cross-Browser Consistency with Font Sizes

Problem

You want the size of type to be consistent across different browsers and operating systems.

Solution

Set the `font-size` in the `body` element to 62.5%:

```
body {
 font-size: 62.5%;
}
```

Then set the `font-size` in the inherited form and table elements to `1em` for Internet Explorer for Windows:

```
input, select, th, td {
 font-size: 1em;
}
```

Now the font sizes in your document will be equivalent to 10 pixels for each 1 em unit. For example, if you add the body declaration in the first part of the Solution, this rule sets the font size for a paragraph to 19 pixels:

```
p {
 font-size: 1.9em /* displays text as 19 pixels */
}
```

Discussion

Because browser displays vary due to different operating systems and video settings, setting type in a *fixed* (or *absolute*) value doesn't make much sense. In fact, it's best to avoid absolute measurements for web documents, unless you're styling documents for fixed output. For example, when you create a stylesheet to print a web document, absolute length units are preferred. For more on creating stylesheets for printing, see Chapter 11.

Using pixels

Although pixels appear to consistently control the size of typography in a web document across most platforms and browsers, it's not a good idea to use pixels when designing for web typography.

The main issue in regard to setting type size in pixels isn't one of accurate sizing, but of accessibility. People with poor vision might want to resize the type to better read the document.

However, if you use pixels to set the type on your web page, people using Internet Explorer 7 will be unable to resize the type. Because Internet Explorer for Windows is the most commonly used browser on the planet, the use of pixels to set type size becomes a problem for most users who need to resize the type in their browsers.

> Internet Explorer 8 and all other browsers have zooming features that expand fonts (even those set in pixels) and images.

If you do require an absolute size measurement, you should use pixels rather than points, even though print designers are more accustomed to point measurements. The reason is that Macintosh and Windows operating systems render point sizes differently, but pixel size typically stays the same.

> Even though pixels are technically a relative unit, designers refer to pixels as absolute units. A pixel is relative in terms of its actual physical size, but it is absolute in terms of its size ratio on a web page, which is what is important to a designer.

If accessibility is a concern, switch to em units. In the Solution, we set the text in the paragraph to 0.9 em units. This value is equivalent to setting the font size to 90% of the default font size set in the browser's preference.

However, the use of em units raises another concern. This time the problem pertains to usability. Although you might be able to resize the type in a web page, if you set a

font to a size that is smaller than the default text size of the browser (e.g., to 0.7 em), Internet Explorer for Windows will display small, almost illegible lines of text, as shown in Figure 3-10. So, the lesson here is be careful with relative sizes, as it is easy to make text illegible.

Figure 3-10. Almost illegible type set with em units

Using font keywords

This brings up the possibility of another solution: the use of `font-size` keywords. The CSS 2.1 specification has seven font keywords for absolute sizes that you can use to set type size (see Figure 3-11): `xx-small`, `x-small`, `small`, `medium`, `large`, `x-large`, and `xx-large`.

There are two other `font-size` keywords for relative measurements: `larger` and `smaller`. If a child element is set to `larger`, the browser can interpret the parent's `font-size` value of `small` and increase the text inside the child element to `medium`.

`Font-size` keywords provide two benefits: they make it easy to enlarge or reduce the size of the text in most browsers, and the font sizes in browsers never go smaller than 9 pixels, ensuring that the text is legible. If you do set text to a small size, use a sans serif font such as Verdana to increase the chances for legibility.

Using em units to control type

Although using font keywords allows for general control over the size of the typography, designers typically want more choices than the several that keywords provide. The Solution offered in this recipe, developed by Richard Rutter (*http://www.clagnut .com/*), delivers this kind of control.

Browsers set the default value of 16 pixels for web typography, which is equal to the `medium` keyword. By setting the `font-size` in the `body` element to 62.5%, the default value of 16 pixels reduces to 10 pixels:

```
(16 pixels)62.5% = 10 pixels
```
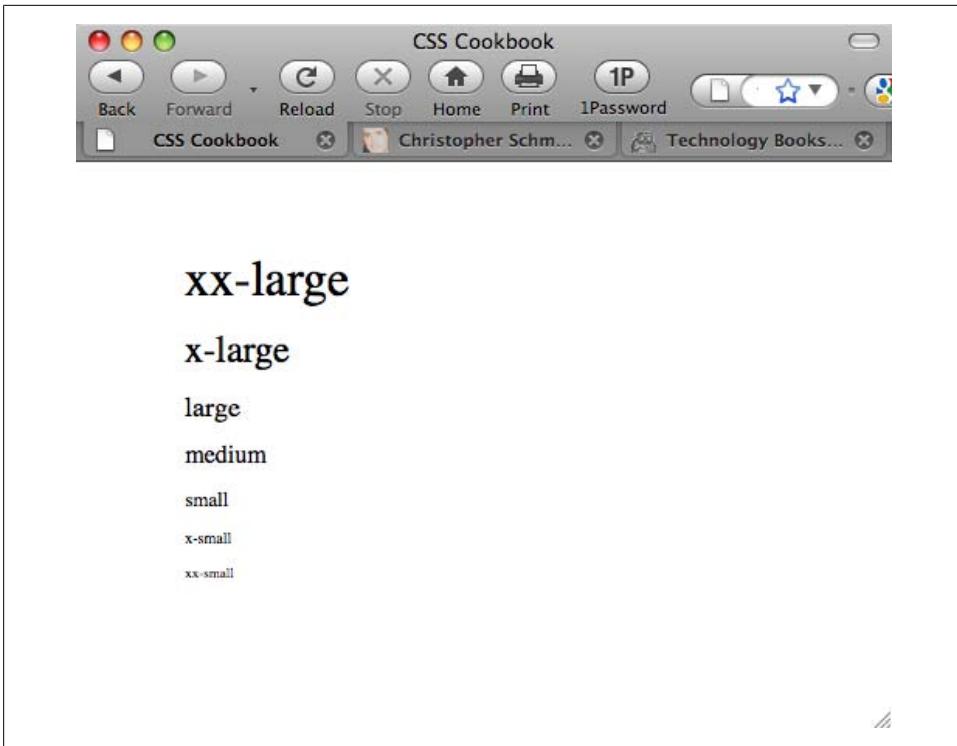
Figure 3-11. The font-size keywords on display

As we discussed earlier, an em unit is the default font size of the user's browser. With the manipulation of the default font size on the **body** element, 1 em unit is now set to 10 pixels:

```
1em = 10px
```

This Solution then allows the web developer pixel-size control over her fonts without the browser limitations manifested in the use of pixels as a value.

For example, if a web developer wants to set the size of a heading to 24 pixels and the text in a paragraph to 15 pixels, the rule sets based on this Solution would look like the following:

```
body {
 font-size: 62.5%;
}
input, select, th, td {
 font-size: 1em;
}
h2 {
 font-size: 2.4em;
}
p {
```

```
    font-size: 1.5em;
  }
```

See Also

The original article by Richard Rutter detailing the Solution at *http://www.clagnut.com/blog/348/*; the article "CSS Design: Size Matters," written by Todd Fahrner (an invited member to the W3C CSS Working Group), available at *http://www.alistapart.com/articles/sizematters/*; the CSS 2.1 specification at *http://www.w3.org/TR/CSS21/cascade.html#q1* for more on how a browser determines values; the CSS2 specification for length units at *http://www.w3.org/TR/REC-CSS2/syndata.html#length-units*; the "Font Size" section in Chapter 5 of *CSS: The Definitive Guide* by Eric A. Meyer (O'Reilly)

3.8 Setting Hyphens, Em Dashes, and En Dashes

Problem

You want to use em and/or en dashes instead of a hyphen, as shown in Figure 3-12.
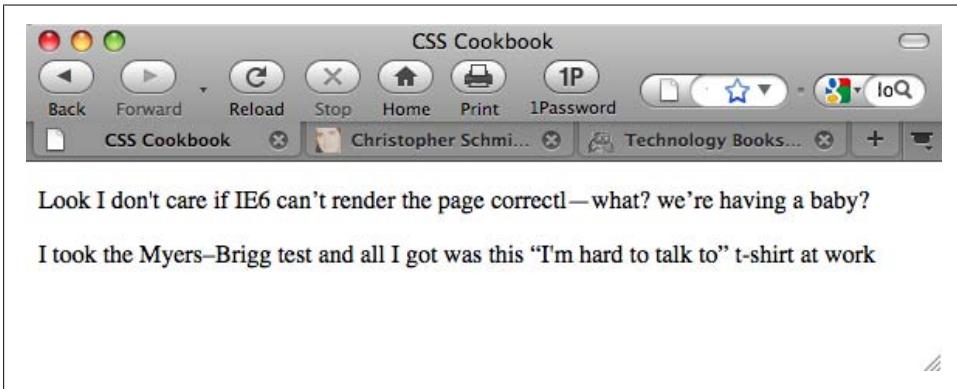
Figure 3-12. Using em and en dashes

Solution

Use the em dash with the decimal representation —:

```
<p>Look I don't care if IE6 can&#8217;t render the page
correctl&#8212;what? we&#8217;re having a baby?</p>
```

For the en dash, use the decimal representation –:

```
<p>I took the Myers&#8211;Brigg test and all I got was this
&#8220;I'm hard to talk to&#8221; t-shirt at work</p>
```

Discussion

A common way to represent em and en dashes is through their HTML entities, `&em;` and `&en;`, respectively. However, for improved cross-browser and cross-platform support, it's better to use the decimal values instead.

See Also

A breakdown of em and en dashes at *http://www.alistapart.com/articles/emen/*

3.9 Centering Text

Problem

You want to center text within a paragraph or a heading.

Solution

Use the `text-align` property with the value set to `center`:

```
h3 {
 text-align: center;
}
p {
 text-align: center;
}
```

Discussion

The `center` value for the `text-align` property is designed to control the alignment of inline content within a block element.

See Also

The CSS 2.1 specification for `text-align` at *http://www.w3.org/TR/CSS21/text.html #alignment-prop*; Recipe 4.3 for centering various items in a web page

3.10 Setting Text to Be Justified

Problem

You want to align text to be justified on both the left and right sides, as shown in Figure 3-13.
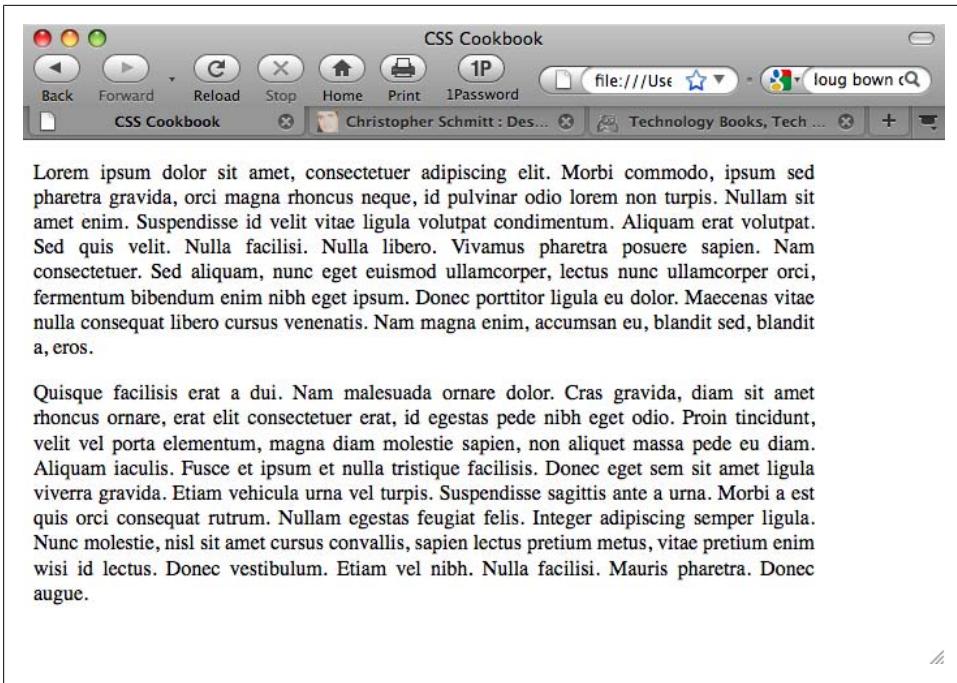
Figure 3-13. A paragraph justified on both sides

Solution

Use the `text-align` property:

```
P {
  width: 600px;
  text-align: justify;
}
```

Discussion

How well does web-based text justification work? According to the CSS 2.1 specification, it depends on the algorithms developed by the engineers who made the browser being used to view the web page. Because there isn't an agreed-upon algorithm for justifying text, the look of the text varies from browser to browser, even though the browser vendor technically supports justification.

Browser support for the property is good in Internet Explorer, Safari, Firefox, Chrome, and Opera. In those browsers, justified text looks pleasing to the eye. In other browsers, justified text may look bad; for example, it might have a lot of whitespace between words.

Justified text is difficult for dyslexics to read. For more information on designing for dyslexia, see *http://www.thepickards.co.uk/index.php/200512/designing-for-dyslexia/*.

See Also

The CSS 2.1 specification for `text-align` at *http://www.w3.org/TR/REC-CSS2/text.html#alignment-prop*

3.11 Indicating an Overflow of Text with an Ellipsis

Problem

You want to keep from expanding beyond the desired boundaries of a parent element, as shown in Figure 3-14.
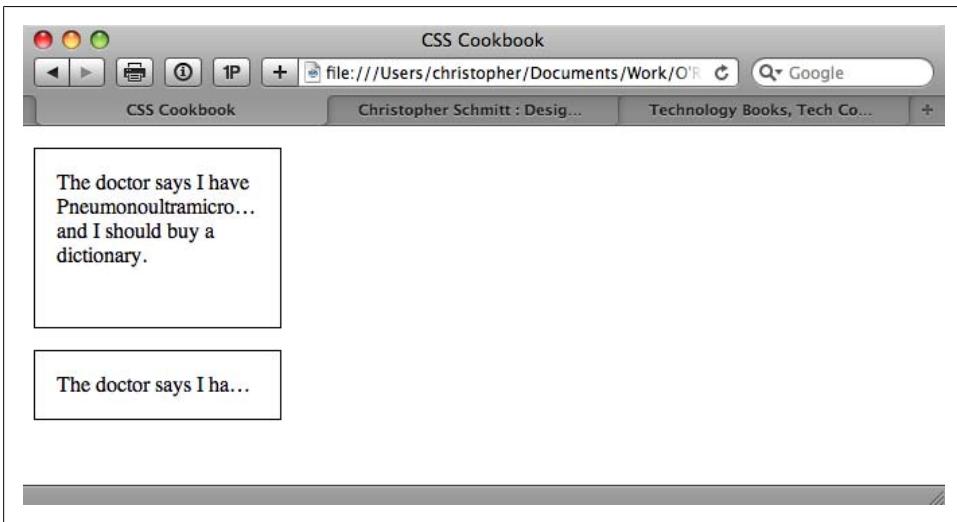
Figure 3-14. Additional text marked with an ellipsis

Solution

Use the `text-overflow` property (along with Opera's proprietary `-o-text-overflow` property):

```
p {
 border: 1px solid black;
 width: 150px;
 height: 100px;
 padding: 12px;
```

```
 border: 1px solid black;
 overflow: hidden;
 padding: 1em;
 text-overflow: ellipsis;
 -o-text-overflow: ellipsis;
}
p.nowrap {
 white-space: nowrap;
 height: auto;
}
```

Discussion

Currently, Safari and Opera support `text-overflow` for the clipping text and substituting ellipsis (`...`).

See Also

The CSS3 specification for `text-overflow` at *http://www.w3.org/TR/2003/CR-css3-text-20030514/#text-overflow*

3.12 Removing Space Between Headings and Paragraphs

Problem

You want to reduce the space between a heading and a paragraph.

Solution

Set the `margin` and `padding` for both the heading and paragraph to 0:

```
h2 + p {
 margin-top: 0;
 padding-top: 0;
}
h2 {
 margin-bottom: 0;
 padding-bottom: 0;
}
p {
 margin: 1em 0 0 0;
 padding: 0;
}
```

Discussion

By using an attribute selector, you are setting the margin and padding between a paragraph and a heading to 0.

Browsers have their own internal stylesheets that dictate the default values for HTML elements. These styles include predetermined values for margin and padding of elements for headings and paragraphs.

These default values make it easy for people to read nonstyled documents, but are often undesired by web developers.

See Also

The CSS 2.1 specification's default stylesheet for HTML4 at *http://www.w3.org/TR/CSS21/sample.html*

3.13 Setting a Simple Initial Cap

Problem

You want a paragraph to begin with an initial cap.

Solution

Mark up the paragraph of content with a `p` element:

```
<p>Online, activity of exchanging ideas is sped up. The
distribution of messages from the selling of propaganda to the
giving away of disinformation takes place at a blindingly fast
pace thanks to the state of technology &hellip;</p>
```

Use the `:first-letter` pseudo-element to stylize the first letter of the paragraph, as shown in Figure 3-15:

```
p:first-letter {
 font-size: 1.2em;
 background-color: black;
 color: white;
}
```
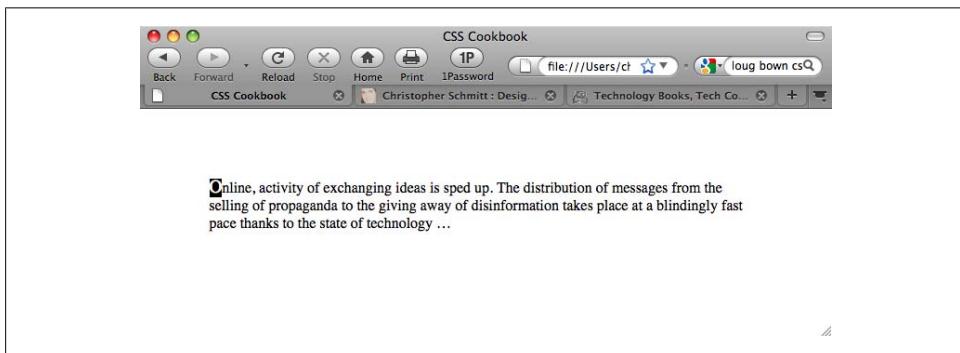

Figure 3-15. A simple initial cap

Discussion

The CSS specification offers an easy way to stylize the first letter in a paragraph as a traditional initial or drop cap: use the `:first-letter` pseudo-element.

`:first-letter` has gained support in modern browsers, but another solution is needed to support older versions of Internet Explorer.

Wrap a `span` element with a `class` attribute around the first letter of the first sentence of the first paragraph:

```
<p><span class="initcap">O</span>nline, activity of exchanging ideas is sped
up. The distribution of messages from the selling of propaganda
to the giving away of disinformation takes place at a blindingly
fast pace thanks to the state of technology &hellip;</p>
```

Then set the style for the initial cap:

```
p .initcap {
 font-size: 1.2em;
 background-color: black;
 color: white;
}
```

Initial caps, also known as *versals*, traditionally are enlarged in print to anything from a few points to three lines of text.

See Also

The CSS 2.1 specification for `:first-letter` at *http://www.w3.org/TR/CSS21/selector .html#x52*

3.14 Setting a Larger, Centered Initial Cap

Problem

You want to place a large initial cap in the center of a paragraph.

Solution

Create the decoration that sets the text indent for the paragraph (see Figure 3-16):

```
p {
 text-indent: 37%;
 line-height: 1em;
}
p:first-letter {
 font-size: 6em;
 line-height: 0.6em;
 font-weight: bold;
}
```
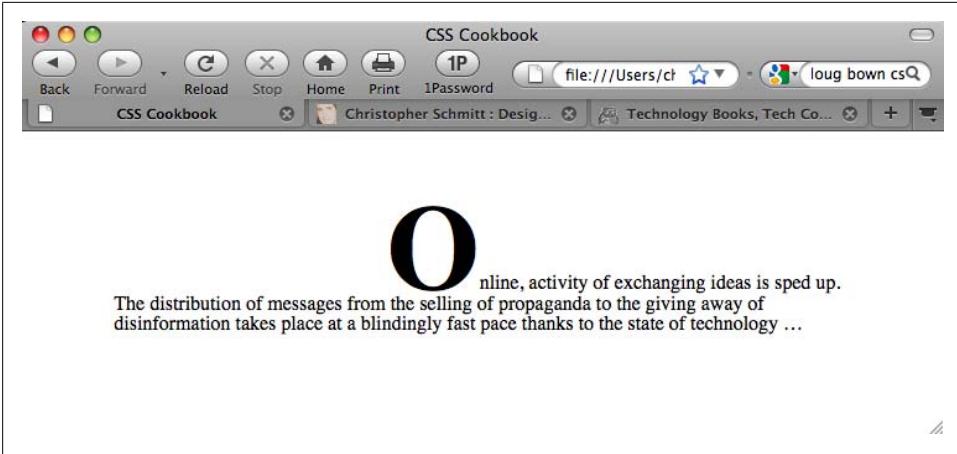
Figure 3-16. A larger, centered initial cap

Discussion

This Solution works due to interaction through the use of the `text-indent` property. The `text-indent` property moves the first line toward the middle of the paragraph.

The value is set to 37%, which is a little bit more than one-third the distance from the left side of the paragraph, as shown in Figure 3-17, but not enough to "center" the initial cap.
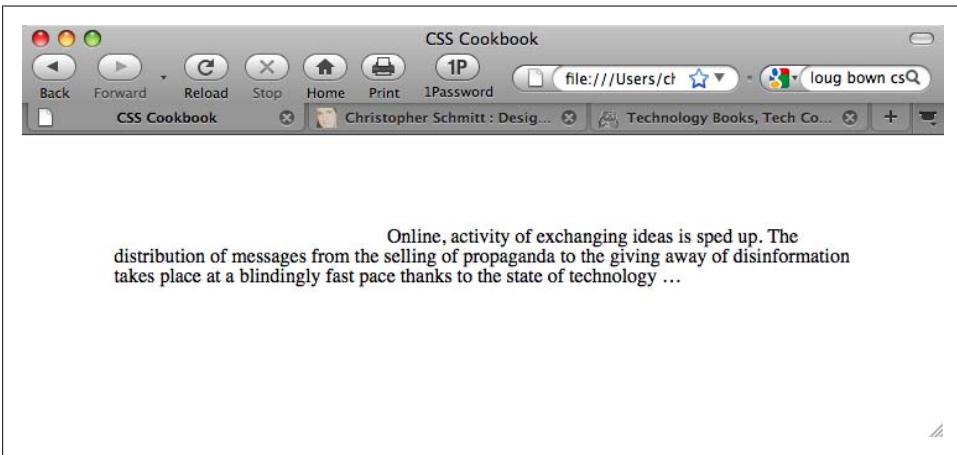
Figure 3-17. The indented text

Note that this recipe for centering the initial cap works, technically, when the character's width is equal to 26% of the paragraph's width. In other words, if the letter for the initial cap or the width of the paragraph is different for your own work, adjustments to the values in the CSS rules are necessary to move the initial cap to the center.

See Also

Recipe 3.30 for adjusting leading with line height; the CSS 2.1 specification for `text-indent` at *http://www.w3.org/TR/CSS21/text.html#propdef-text-indent*

3.15 Setting an Initial Cap with Decoration (Imagery)

Problem

You want to use an image for an initial cap.

Solution

Wrap a `span` element around the first letter of the first sentence of the first paragraph:

```
<p><span class="initcap">O</span>nline, activity of exchanging
ideas is sped up. The distribution of messages from the selling of
propaganda to the giving away of disinformation takes place at a
blindingly fast pace thanks to the state of technology&hellip;</p>
```

Set the contents inside the `span` to be hidden:

```
span.initcap {
 display: none;
}
```

Then set an image to be used as the initial cap in the background of the paragraph (see Figure 3-18):

```
p {
  line-height: 1em;
  background-image: url(initcap-o.gif);
  background-repeat: no-repeat;
  text-indent: 35px;
  padding-top: 45px;
}
```

Discussion

The first step of this Solution is to create an image for use as the initial cap. Once you have created the image, make a note of its width and height. In this example, the image of the letter measures 55 × 58 pixels (see Figure 3-19).
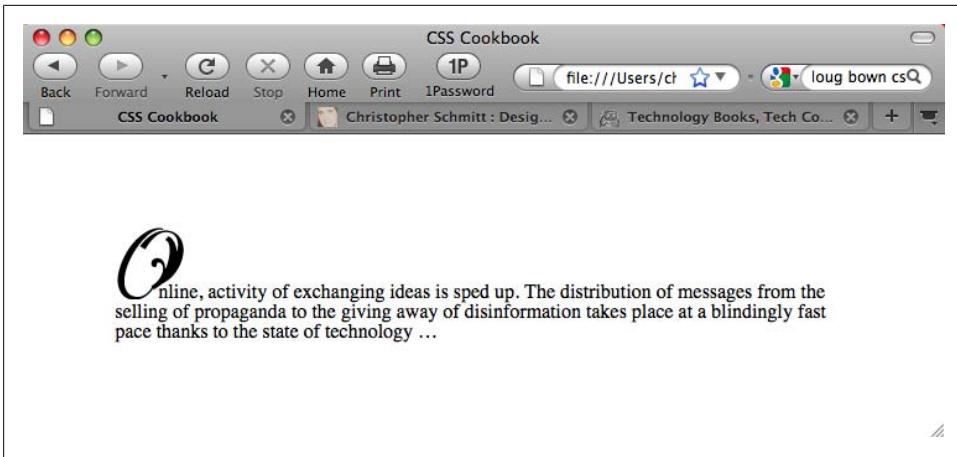
Figure 3-18. An image used as an initial cap
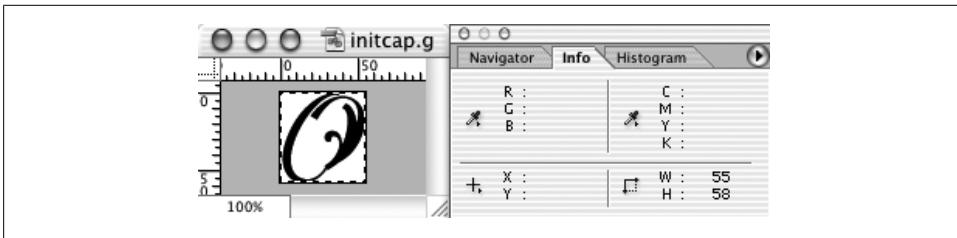
Figure 3-19. The image of the initial cap

Next, hide the first letter of the HTML text by setting the `display` property to `none`. Then put the image in the background of the paragraph, making sure that the image doesn't repeat by setting the value of `background-repeat` to `no-repeat`:

```
background-image: url(initcap-o.gif);
background-repeat: no-repeat;
```

With the measurements already known, set the width of the image as the value for `text-indent` and the height of the image as the padding for the top of the paragraph (see Figure 3-20):

```
text-indent: 55px;
padding-top: 58px;
```

Then change the `text-indent` and `padding-top` values so that the initial cap appears to rest on the baseline, as was shown in Figure 3-18.
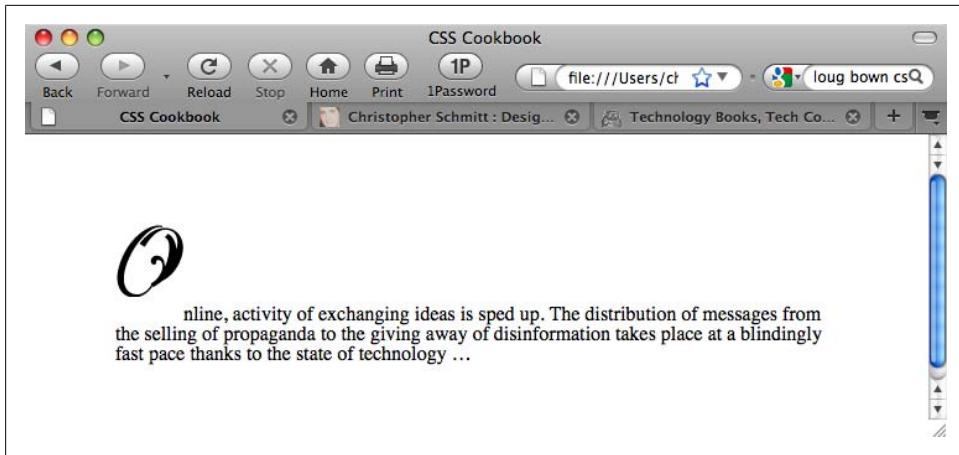
Figure 3-20. Adjusting the space for the initial cap

Allow for accessibility

Note that users with images turned off aren't able to see the initial cap, especially since the Solution doesn't allow for an `alt` attribute for the image. If you want to use an image but still have an `alt` attribute show when a user turns off images, use an image to replace the HTML character:

```
<p><img src="initcap-o.gif" alt="O" />nline, activity of exchanging
ideas is sped up. The distribution of messages from the selling
of propaganda to the giving away of disinformation takes place at
a blindingly fast pace thanks to the state of technology&hellip;</p>
```

Note that although the `alt` attribute is displayed in this Solution, the ability to kern the space between the initial cap and the HTML text is lost. The HTML text begins exactly at the right side of the image and can't be moved closer to the letter being displayed in the graphic itself.

See Also

Recipe 3.13 for setting a simple initial cap

3.16 Creating a Heading with Stylized Text

Problem

You want to use CSS properties to design a heading that is different from the default. For example, you want to put the heading in Figure 3-21 into italics, as shown in Figure 3-22.
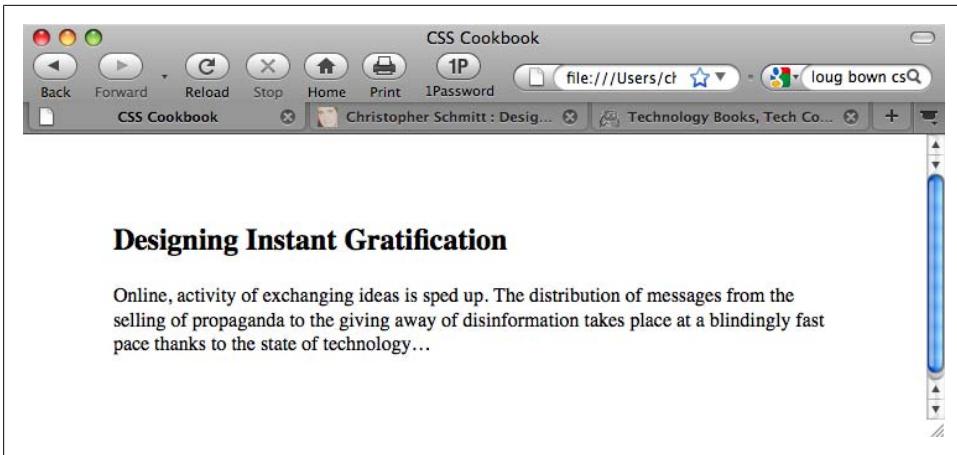
Figure 3-21. The default rendering of a heading
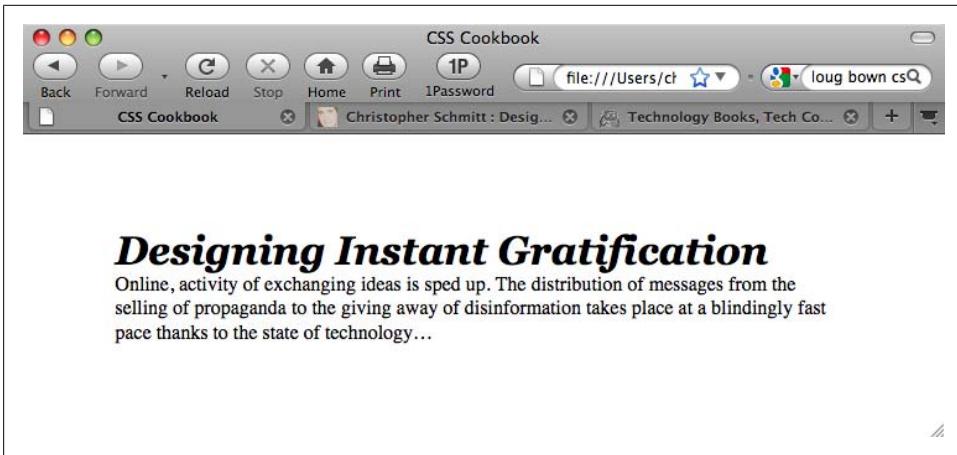
Figure 3-22. The stylized text of a heading

Solution

First, properly mark up the heading:

```
<h2>Designing Instant Gratification</h2>
<p>Online, activity of exchanging ideas is sped up. The
distribution of messages from the selling of propaganda to the
giving away of disinformation takes place at a blindingly fast
pace thanks to the state of technology&hellip;</p>
```

Then, use the `font` shorthand property to easily change the style of the heading:

```
h2 {
  font: bold italic 2em  Georgia, Times, "Times New Roman", serif;
  margin: 0;
```

```
  padding: 0;
}
p {
 margin: 0;
 padding: 0;
}
```

Discussion

A *shorthand property* combines several properties into one. The `font` property is just one of these timesavers. One `font` property can represent the following values:

- `font-style`
- `font-variant`
- `font-weight`
- `font-size/line-height`
- `font-family`

The first three values can be placed in any order; the others need to be in the order shown.

When you want to include the `line-height` value, put a forward slash between the `font-size` value and the `line-height` value:

```
p {
 font: 1em/1.5em Verdana, Arial, sans-serif;
}
```

When setting the style headings, remember that browsers have their own default values for padding and margins of paragraphs and heading tags. These default values are generally based on mathematics, not aesthetics, so don't hesitate to adjust them to further enhance the look of your web document.

See Also

The CSS 2.1 specification for the `font` shorthand property at *http://www.w3.org/TR/CSS21/fonts.html#propdef-font*

3.17 Creating a Heading with Stylized Text and Borders

Problem

You want to stylize the borders on the top and bottom of a heading, as shown in Figure 3-23.
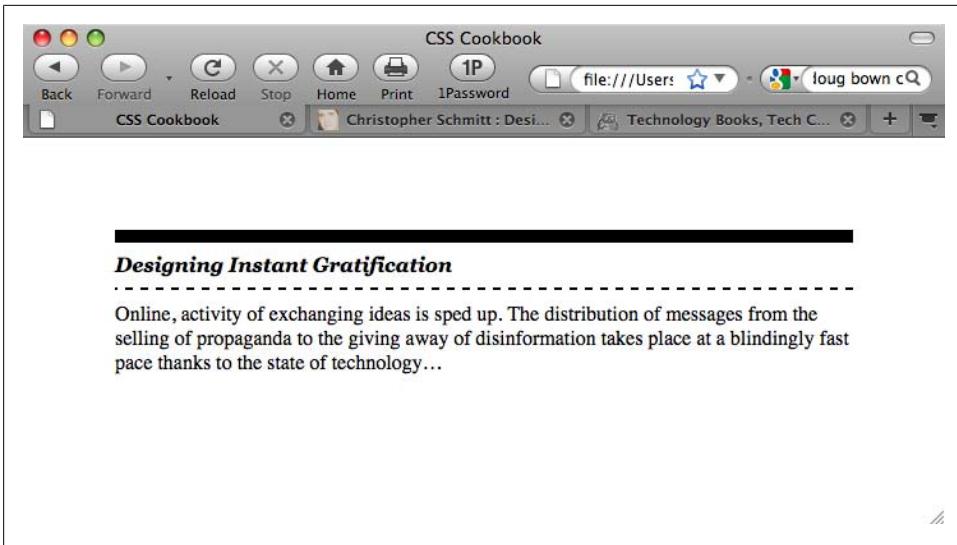
Figure 3-23. A heading stylized with borders

Solution

Use the `border-top` and `border-bottom` properties when setting the style for the heading:

```
h2 {
  font: bold italic 2em Georgia, Times, "Times New Roman", serif;
  border-bottom: 2px dashed black;
  border-top: 10px solid black;
  margin: 0;
  padding: 0.5em 0 0.5em 0;
  font-size: 1em;
}
p {
  margin: 0;
  padding: 10px 0 0 0;
}
```

Discussion

In addition to top and bottom borders, a block-level element also can have a border on the left and right sides via the `border-left` and `border-right` properties, respectively. The `border-top`, `border-bottom`, `border-left`, and `border-right` properties are shorthand properties that enable developers to set the width, style, and color of each side of a border.

Without the two shorthand border declarations in the Solution, the CSS rule for the heading would be expanded by four extra declarations:

```
h2 {
  font: bold italic 2em Georgia, Times, "Times New Roman", serif;
  border-bottom-width: 2px;
  border-bottom-style: dashed;
  border-bottom-color: black;
  border-top-width: 10px;
  border-top-style: solid;
  border-top-color: black;
  margin: 0;
  padding: 0.5em 0 0.5em 0;
  font-size: 1em;
}
```

Also available is a shorthand property for the `top`, `bottom`, `left`, and `right` shorthand properties: `border`. The `border` property sets the same style for the width, style, and color of the border on each side of an element:

```
h2 {
  border: 3px dotted #33333;
}
```

When setting the borders, make sure to adjust the padding to put enough whitespace between the borders and the text of the heading. This aids in readability. Without enough whitespace on a heading element, the text of the heading can appear cramped.

See Also

Recipe 5.5 for more information on styles of borders and the shorthand `border` property

3.18 Stylizing a Heading with Text and an Image

Problem

You want to place a repeating image at the bottom of a heading, like the grass in Figure 3-24.

Solution

Use the `background-image`, `background-repeat`, and `background-position` properties:

```
h2 {
  font: bold italic 2em Georgia, Times, "Times New Roman", serif;
  background-image: url(tall_grass.jpg);
  background-repeat: repeat-x;
  background-position: bottom;
  border-bottom: 10px solid #666;
  margin: 10px 0 0 0;
  padding: 0.5em 0 60px 0;
}
```
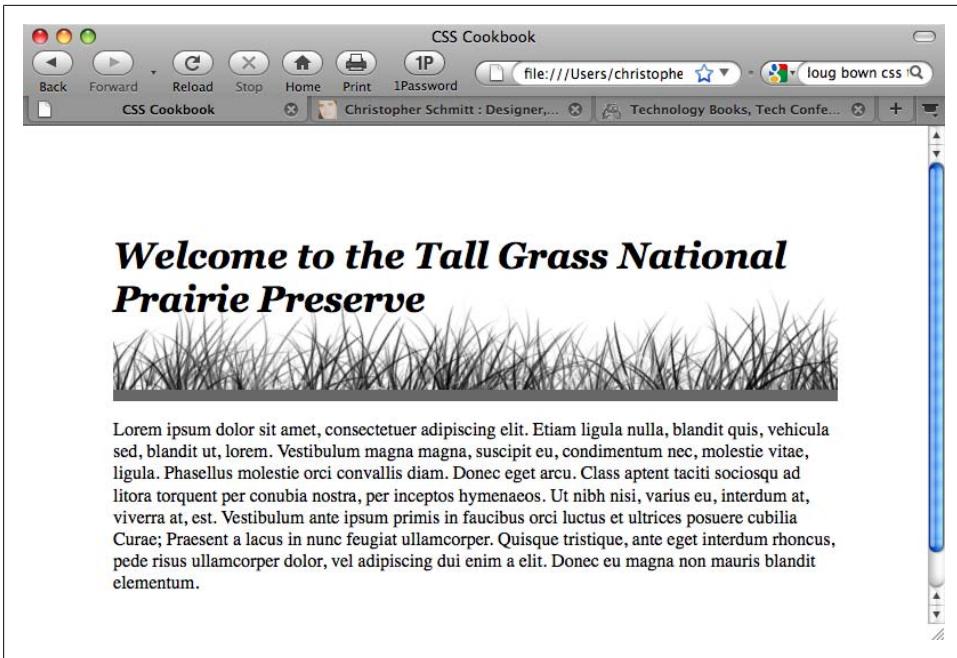
Figure 3-24. A background image used with a heading

Discussion

Make a note of the height of the image used for the background. In this example, the height of the image is 100 pixels (see Figure 3-25).

Figure 3-25. An image of tall grass

Set the `background-repeat` property to a value of `repeat-x`, which will cause the image to repeat horizontally:

```
background-image: url(tall_grass.jpg);
background-repeat: repeat-x;
```

> The image's location for the value of `url()` is relative to its position to the stylesheet and *not* the HTML document.

Next, set the `background-position` property to `bottom`:

```
background-position: bottom;
```

The `background-position` property can take up to two values corresponding to the horizontal and vertical axes. Values for `background-position` can be a length unit (such as pixels), a percentage, or a keyword. To position an element on the x-axis, use the keyword value `left`, `center`, or `right`. For the y-axis, use the keyword value `top`, `center`, or `bottom`.

When the location of the other axis isn't present, the image is placed in the center of that axis, as shown in Figure 3-26:

```
background-position: bottom;
```
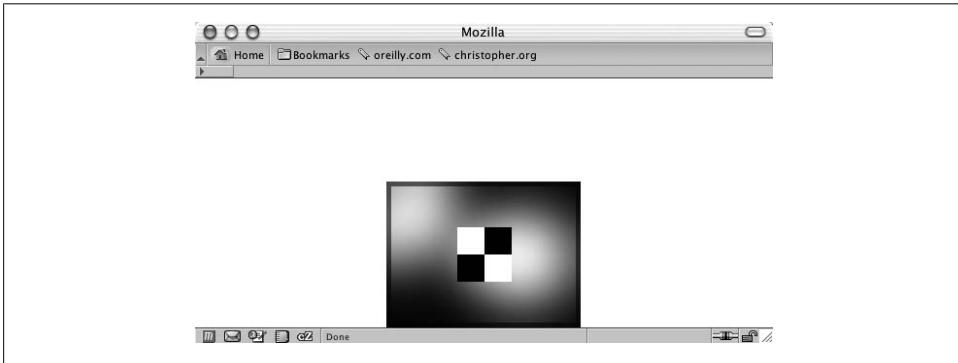

Figure 3-26. The image aligned on the bottom of the y-axis and in the middle of the x-axis

So, in this Solution, the image is placed at the bottom of the y-axis but repeats along the x-axis.

See Also

Recipe 4.5 for setting a background image in an entire web page

3.19 Creating a Pull Quote with HTML Text

Problem

You want to stylize the text for a pull quote so that it is different from the default. Undifferentiated quotes aren't obviously from another writer, whereas stylized quotes are (see Figure 3-27).
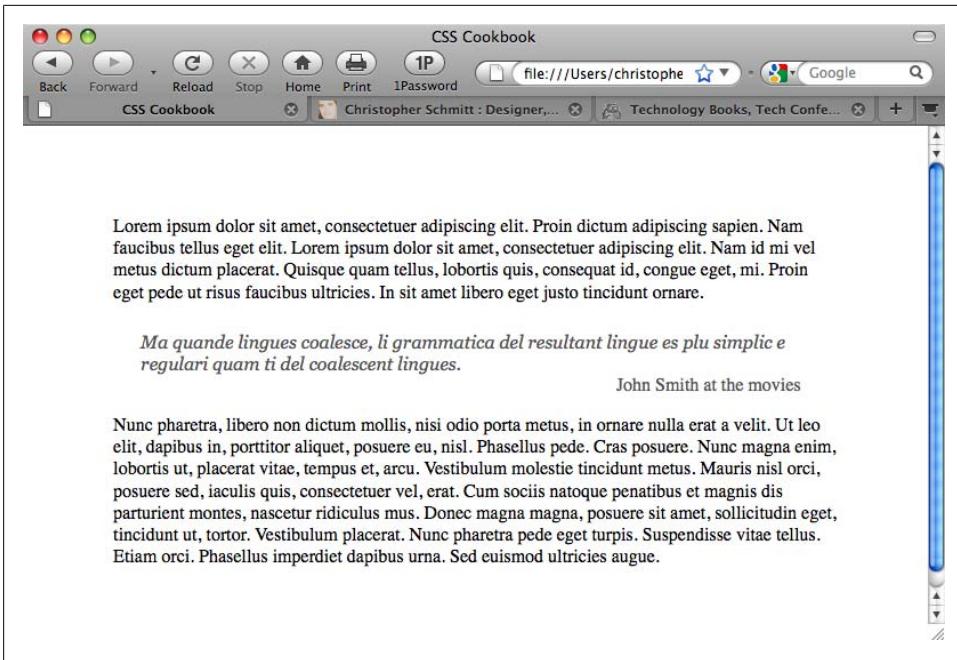
Figure 3-27. A stylized pull quote

Solution

Use the `blockquote` element to indicate the pull quote semantically in the markup:

```
<blockquote>
 <p>Ma quande lingues coalesce, li grammatica del resultant
 lingue es plu simplic e regulari quam ti del coalescent
lingues.</p>
 <div class="source">John Smith at the movies</div>
</blockquote>
```

With CSS, apply the margin, padding, and color values to the `blockquote` element:

```
blockquote {
 margin: 0;
 padding: 0;
 color: #555;
}
```

Next, set the style for the `p` and `div` elements nested in the `blockquote` element:

```
blockquote p {
 font: italic 1em Georgia, Times, "Times New Roman", serif;
 font-size: 1em;
 margin: 1.5em 2em 0 1.5em;
 padding: 0;
}
blockquote .source {
```

```
   text-align: right;
   font-style: normal;
   margin-right: 2em;
 }
```

Discussion

A pull quote is used in design to grab a reader's attention so that he will stick around and read more. One easy way to create a pull quote is to change the color of a portion of the main text.

Improve on this by adding contrast: change the pull quote's generic font family so that it is different from that of the main text. For example, if the main text of a web document is set in sans serif, set the pull quote text to a serif font.

See Also

Recipes 3.21 and 3.22 for more information on designing pull quotes with CSS

3.20 Placing a Pull Quote to the Side of a Column

Problem

You want to place a pull quote to the side of a main passage of text.

Solution

Apply padding to the left side of the text:

```
#content {
 padding-left: 200px;
}
```

Then use the `float` property to let the content wrap around the pull quote:

```
blockquote {
 padding: 0;
 margin: 0;
 float: left;
 width: 180px;
 text-align: right;
 color: #666;
}
```

Next, set a negative margin value to pull the pull quote in the padding area on the left side of the text, as shown in Figure 3-28:

```
blockquote {
 padding: 0;
 margin: 0;
 float: left;
 width: 180px;
```

```
    margin-left: -200px;
    text-align: right;
    color: #666;
}
```
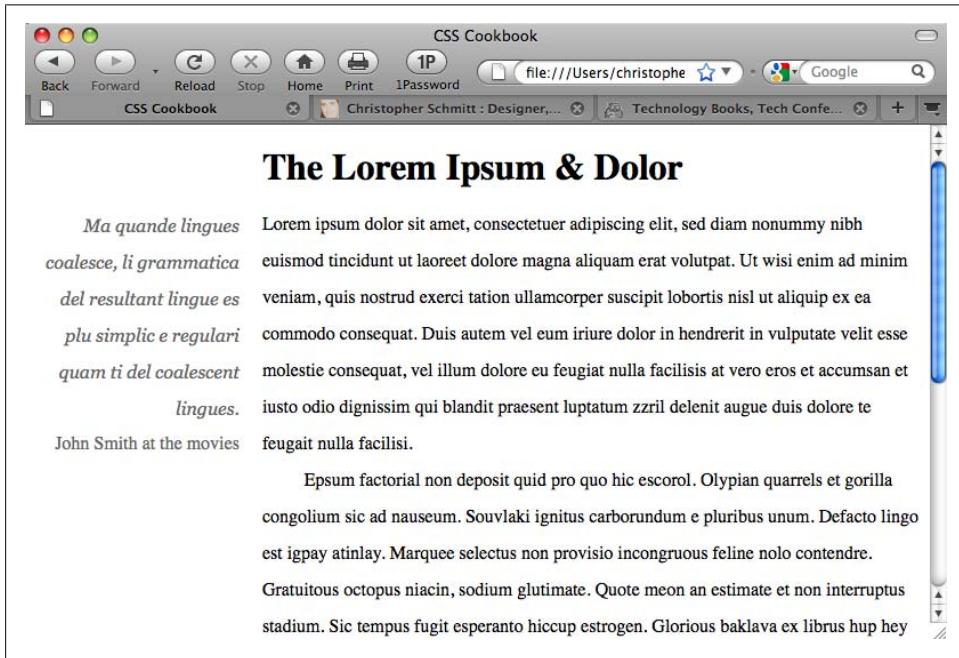

Figure 3-28. A pull quote to the left of a column

Discussion

Setting the pull quote to the left side of the text is a two-step process.

First, set enough room for the pull quote through the use of padding on the element that contains the entire passage. Then set a negative value for the `blockquote` on a floated pull quote to pull it out of the passage of text completely.

This technique is not limited to pull quotes, but is also useful for placing photos to the left of text to reinforce the content.

See Also

Chapter 10 for more ways to flow text in a web page

3.21 Creating a Pull Quote with Borders

Problem

You want to stylize a pull quote with borders on the top and bottom, as in Figure 3-29.
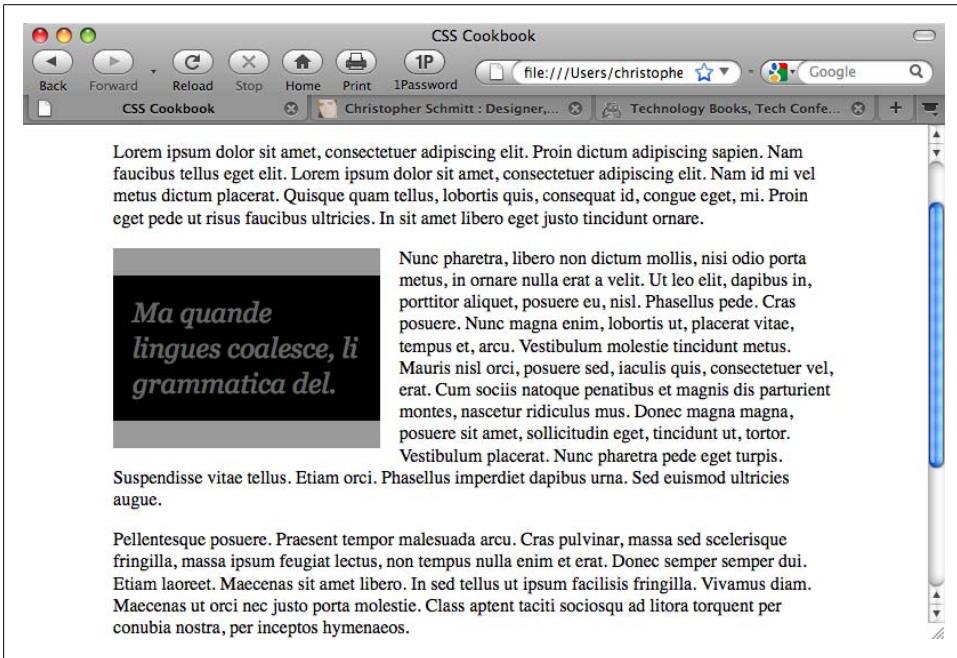
Figure 3-29. A stylized pull quote using borders

Solution

To put borders on the left and right instead of the top and bottom, use the `border-left` and `border-right` properties:

```
border-left: 1em solid #999;
border-right: 1em solid #999;
```

Use the `blockquote` element to mark up the pull quote content:

```
<blockquote>
 <p>&laquo;Ma quande lingues coalesce, li
grammatica del.&raquo;</p>
</blockquote>
```

Next, set the CSS rules for the border and text within the pull quote:

```
blockquote {
  float: left;
  width: 200px;
```

```
      margin: 0 0.7em 0 0;
      padding: 0.7em;
      color: #666;
      background-color: black;
      font-family: Georgia, Times, "Times New Roman", serif;
      font-size: 1.5em;
      font-style: italic;
      border-top: 1em solid #999;
      border-bottom: 1em solid #999;
    }
    blockquote p {
     margin: 0;
     padding: 0;
     text-align: left;
     line-height: 1.3em;
    }
```

Discussion

Set the `float` property as well as the `width` property for the `blockquote` element. These two CSS properties allow the main content to wrap around the pull quote:

```
float: left;
width: 200px;
```

Contrast the pull quote with the surrounding text by changing the quote's foreground and background colors:

```
color: #666;
background-color: black;
```

Use the `border-top` and `border-bottom` properties to match the color of the text in the pull quote:

```
border-top: 1em solid #999;
border-bottom: 1em solid #999;
```

See Also

Chapter 7 for several page-layout techniques that take advantage of the `float` property; Recipe 3.17 for styling headings with borders; Recipes 13.3 and 13.4 for more on designing with contrast

3.22 Creating a Pull Quote with Images

Problem

You want to stylize a pull quote with images on either side, such as the curly braces in Figure 3-30.
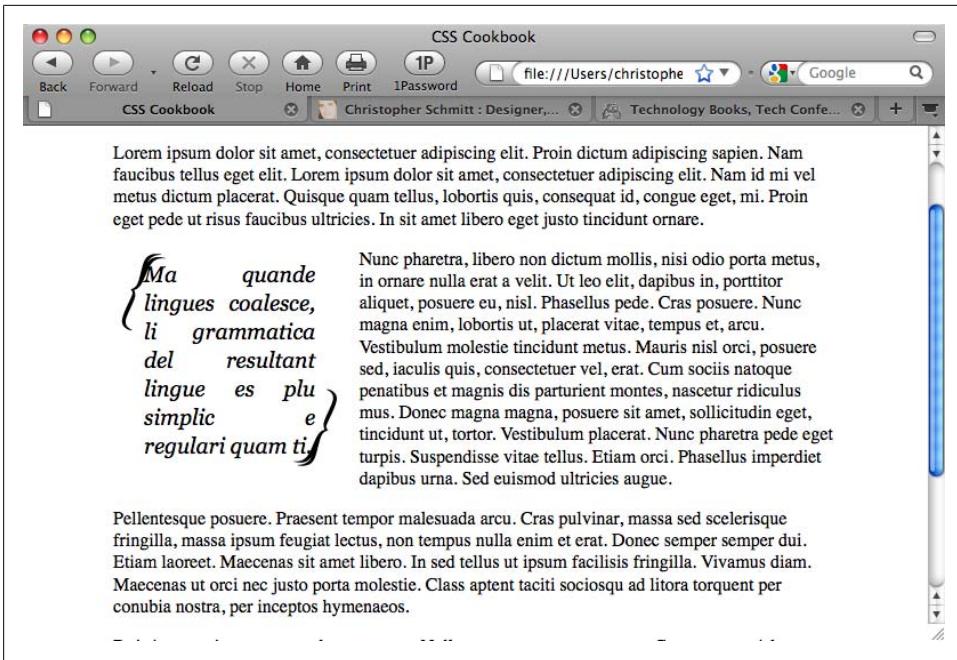
Figure 3-30. A pull quote with images

Solution

Use the `blockquote` element to mark up the pull quote content:

```
<blockquote>
 <p>Ma quande lingues coalesce, li grammatica del resultant
lingue es plu simplic e regulari quam ti.</p>
</blockquote>
```

Then set the style for the pull quote, placing one image in the background of the `blockquote` element and another in the background of the `p` element:

```
blockquote {
 background-image: url(bracket_left.gif);
 background-repeat: no-repeat;
 float: left;
 width: 175px;
 margin: 0 0.7em 0 0;
 padding: 10px 0 0 27px;
 font-family: Georgia, Times, "Times New Roman", serif;
 font-size: 1.2em;
 font-style: italic;
 color: black;
}
blockquote p {
 margin: 0;
 padding: 0 22px 10px 0;
 width:150px;
```

```
      text-align: justify;
      line-height: 1.3em;
      background-image: url(bracket_right.gif);
      background-repeat: no-repeat;
      background-position: bottom right;
   }
```

Discussion

For this Solution, the images for the pull quote come in a pair, with one at the upper-left corner and the other at the bottom-right corner. Through CSS, you can assign only one background image per block-level element.

The workaround is to give these images the proper placement; put one image in the background of the blockquote element and the other in the p element that is a child of the blockquote element:

```
   blockquote {
      background-image: url(bracket_left.gif);
      background-repeat: no-repeat;
      float: left;
      width: 175px;
   }
   blockquote p {
      background-image: url(bracket_right.gif);
      background-repeat: no-repeat;
      background-position: bottom right;
   }
```

Then adjust the padding, margin, and width of the blockquote and p elements so that you have an unobstructed view of the images:

```
   blockquote {
      background-image: url(bracket_left.gif);
      background-repeat: no-repeat;
      float: left;
      width: 175px;
      margin: 0 0.7em 0 0;
      padding: 10px 0 0 27px;
   }
   blockquote p {
      margin: 0;
      padding: 0 22px 10px 0;
      width: 150px;
      background-image: url(bracket_right.gif);
      background-repeat: no-repeat;
      background-position: bottom right;
   }
```

A benefit of this Solution is that if the text is resized, as shown in Figure 3-31, the images (braces) reposition themselves.
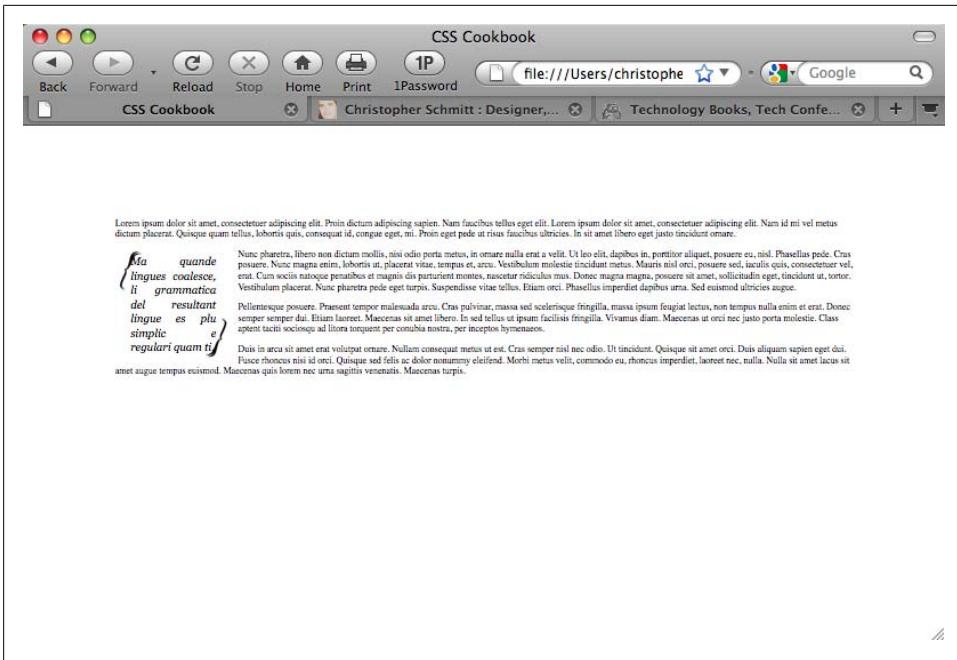
Figure 3-31. The background images staying in the corners as the text is resized

See Also

Recipe 7.20

3.23 Setting the Indent in the First Line of a Paragraph

Problem

You want to place an indent in the first line of each paragraph, as shown in Figure 3-32.

Solution

Use the `text-indent` property to create the indent:

```
p {
  text-indent: 2.5em;
  margin: 0 0 0.5em 0;
  padding: 0;
}
```
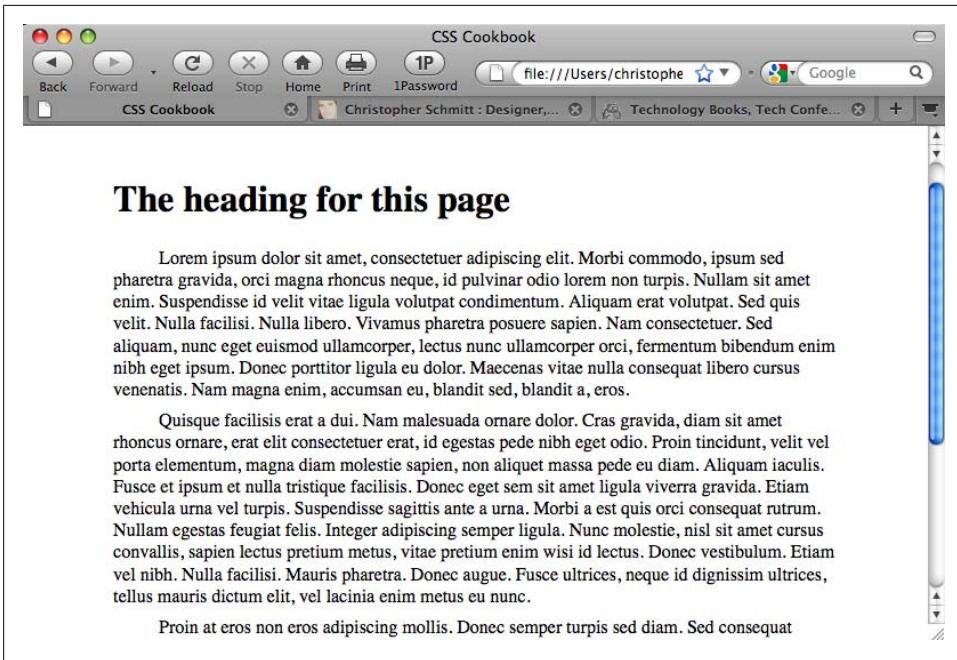
Figure 3-32. Paragraphs with first lines indented

Discussion

The `text-indent` property can take absolute and relative length units as well as percentages. If you use percentages, the percentage refers to the element's width and not the total width of the page. In other words, if the indent is set to 35% of a paragraph that is set to a width of 200 pixels, the width of the indent is 70 pixels.

See Also

The CSS 2.1 specification for `text-indent` at *http://www.w3.org/TR/CSS21/text.html #propdef-text-indent*

3.24 Setting the Indent of Entire Paragraphs

Problem

You want to indent entire paragraphs, as shown in Figure 3-33.
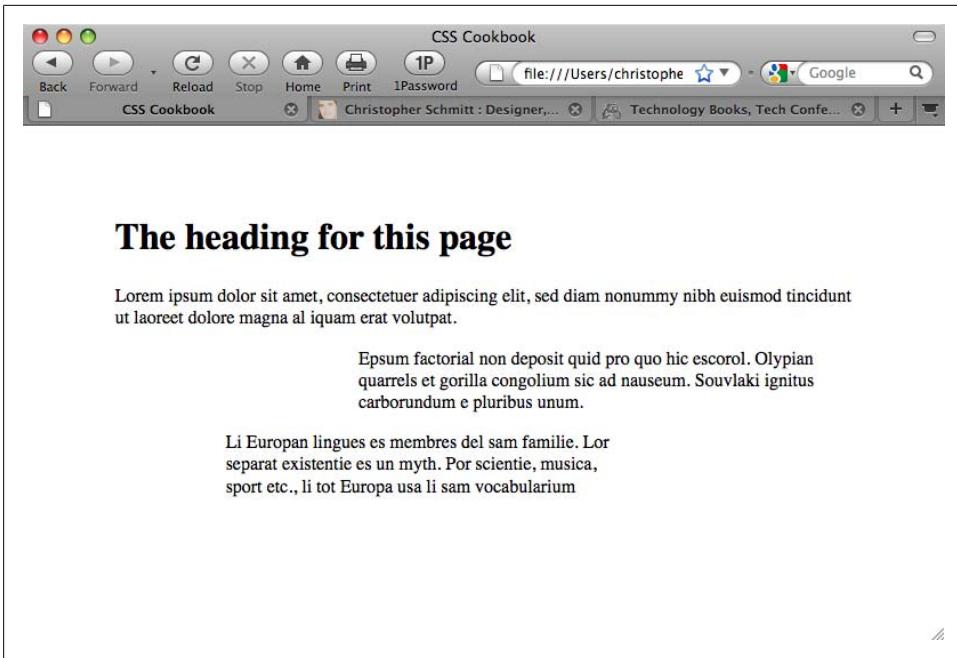
Figure 3-33. Indented paragraphs

Solution

To achieve the desired effect, use class selectors:

```
p.normal {
 padding: 0;
 margin-left: 0;
 margin-right: 0;
}
p.large {
 margin-left: 33%;
 margin-right: 5%;
}
p.medium {
 margin-left: 15%;
 margin-right: 33%;
}
```

Then place the appropriate attribute in the markup:

```
<p class="normal">Lorem ipsum dolor sit amet, consectetuer
adipiscing elit, sed diam nonummy nibh euismod tincidunt ut
laoreet dolore magna al iquam erat volutpat.</p>
<p class="large">Epsum factorial non deposit quid pro quo hic
escorol. Olypian quarrels et gorilla congolium sic ad nauseum.
Souvlaki ignitus carborundum e pluribus unum.</p>
<p class="medium ">Li Europan lingues es membres del sam
```

```
    familie. Lor separat existentie es un myth. Por scientie, musica,
    sport etc., li tot Europa usa li sam vocabularium</p>
```

Discussion

Class selectors pick any HTML element that uses the `class` attribute. The difference between class and type selectors is that type selectors pick out every instance of the HTML element. In the following two CSS rules, the first selector is a type selector that signifies that all content marked as `h2` be displayed as red, and the second selector is a class selector that sets the padding of an element to 33%:

```
h2 {
   color: red;
}
.largeIndent {
   padding-left: 33%;
}
```

Combining both type and class selectors on one element provides greater specificity over the styling of elements. In the following markup, the third element is set to red and also has padding on the left set to 33%:

```
<h2>This is red.</h2>
<h3 class="largeIndent">This has a rather large indent.</h3>
<h2 class="largeIndent">This is both red and indented.</h2>
```

An alternative solution to class selectors is to apply the indent using margins and then use adjacent sibling selectors to apply the style to the paragraphs:

```
p, p+p+p+p {
 padding: 0;
 margin-left: 0;
 margin-right: 0;
}
p+p, p+p+p+p+p {
 margin-left: 33%;
 margin-right: 5%;
}
p+p+p, p+p+p+p+p+p {
 margin-left: 15%;
 margin-right: 33%;
}
```

This method takes advantage of the adjacent sibling selectors, which are represented by two or more regular selectors separated by plus sign(s). For example, the `h2+p` selector stylizes the paragraph *immediately following* an `h2` element.

For this recipe, we want to stylize certain paragraphs in the order in which they appear on-screen. For example, `p+p` selects the paragraph element that follows another paragraph. However, when there are more than two paragraphs, the third paragraph (as well as others after the third paragraph) is rendered in the same style as the second paragraph. This occurs because the third paragraph is immediately followed by a paragraph.

To separate the styles from the second and third paragraphs, set up another CSS rule for the third paragraph that selects three paragraphs that follow each other:

```
p+p+p {
 margin-left: 15%;
 margin-right: 33%;
}
```

Then, build off of these CSS rules by *grouping* the selectors. Instead of writing two CSS rules to stylize the third and sixth paragraphs, separate the selectors by a comma and a space:

```
p+p+p, p+p+p+p+p+p {
 margin-left: 15%;
 margin-right: 33%;
}
```

The main problem with adjacent sibling selectors is that they aren't supported by all versions of Internet Explorer for Windows. Therefore, these users will not see the paragraphs indented. Adjacent sibling selectors are supported in Safari, Firefox, Chrome, and Opera. Internet Explorer 8 has almost complete support.

> Instead of using attribute selectors, another way to approach this Solution is to use the `:nth-child()` selector to pinpoint which paragraphs will be applied. However, attribute selectors enjoy more browser support than `:nth-child()` at the time of this writing.

See Also

The CSS 2.1 specification for class selectors at *http://www.w3.org/TR/CSS21/selector.html#class-html*; the CSS 2.1 specification for adjacent sibling selectors at *http://www.w3.org/TR/CSS21/selector.html#adjacent-selectors*

3.25 Creating a Hanging Indent

Problem

You want to create a hanging indent.

Solution

Use a negative value for the `text-indent` property:

```
p.hanging {
 text-indent: -5em;
}
```

Discussion

The typographic treatment of a hanging indent is already commonplace in most browsers in definition lists. With this simple code, a series of hanging indents (see Figure 3-34) is created without breaking a proverbial sweat:

```
<dl>
<dt>Hanging Indent</dt>
<dd>A common typographic effect where the first line of a paragraph is aligned
with the left margin while the proceeding lines are indented. The technique
creates the visual effect where the first line is left hanging over other lines
of text.</dd>
</dl>
```
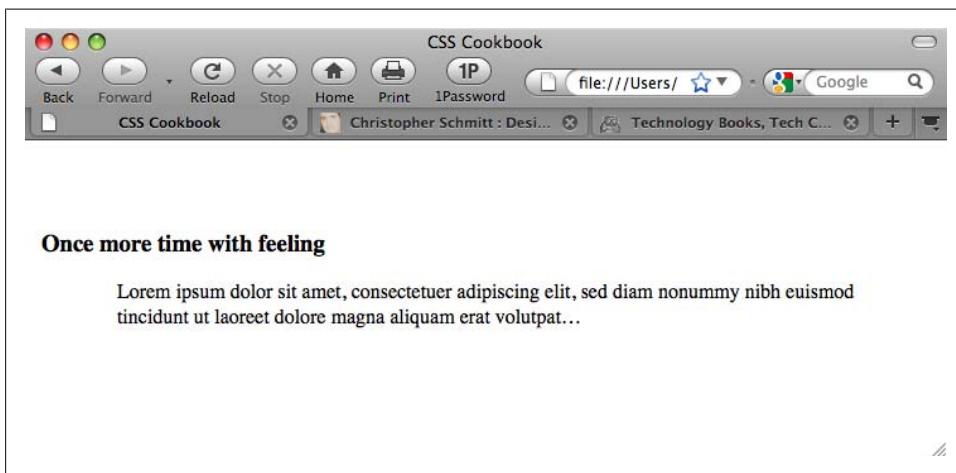

Figure 3-34. Definition lists that render hanging indents by default

When you want a hanging indent on just a paragraph (not a list), use of the definition list markup will not suffice. The straightforward approach shown in the Solution involves the use of the `text-indent` property in CSS.

Hanging indents safely

Before putting the `text-indent` property into a stylesheet, make sure the code is implemented the right way. For example, if you put just the `text-indent` property into a CSS rule along with some basic font styling properties, that hanging indent could cause a legibility issue.

In Figure 3-35, notice that the hanging indent extends to the left of the viewport. Readers might be able to determine the words being cropped off through the context of the rest of the paragraph; however, that's simply an unneeded burden to place on them.

To work around this situation, apply a value equal to the indent to the left margin of the paragraph. The hanging indent then extends over the area already made clear by the margin, ensuring that the text in the hanging indent remains visible:

```css
p.hanging {
  text-indent: -5em;
  margin-left: 5em;
}
```
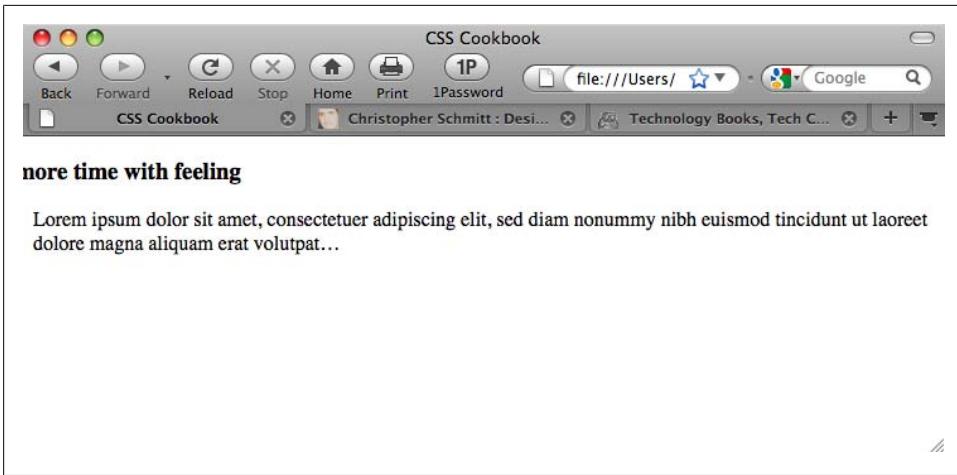

Figure 3-35. The hanging indent, exiting stage left

The paired hanging indent

In addition to having just the first line indent, moving a heading to the left as well results in a paired hanging indent:

```css
#content p.hanging {
  text-indent: -60px;
  margin: 0 0 0 60px;
  padding: 0;
}
#content h3 {
  text-indent: -60px;
  margin: 0 0 0 60px;
  padding: 0;
}
```

The HTML markup for this effect follows:

```html
<div id="content">
<h3>One more time with feeling</h3>
<p class="hanging">
Lorem ipsum dolor sit amet, consectetuer adipiscing elit, sed diam nonummy nibh
euismod tincidunt ut laoreet dolore magna aliquam erat volutpat&hellip;</p>
</div>
```

Or with some slight adjustment, have only the heading become the hanging indent:

```
#content p {
 margin: 0;
 padding: 0 0 0 60px;
}
#content h3 {
 text-indent: -60px;
 margin: 0 0 0 60px;
 padding: 0;
}
```

The refined HTML markup follows:

```
<div id="content">
 <h3>One more time with feeling</h3>
 <p>Lorem ipsum dolor sit amet, consectetuer adipiscing elit, sed diam nonummy
nibh euismod tincidunt ut laoreet dolore magna aliquam erat
volutpat&hellip;</p>
</div>
```

See Also

The CSS 2.1 specification for `text-indent` at *http://www.w3.org/TR/CSS21/text.html#propdef-text-indent*

3.26 Styling the First Line of a Paragraph

Problem

You want to set the first line of a paragraph in boldface, as in Figure 3-36.

Solution

Use the `:first-line` pseudo-element to set the style of the first line:

```
p:first-line {
 font-weight: bold;
}
```

Discussion

Just like a class selector, a *pseudo-element* enables you to manipulate the style of parts of a web document. Unlike a class selector, however, resizing a browser window or changing the size of the font can change the area marked by a pseudo-element. In this Solution, the amount of text in the first line can change if the browser is resized, as shown in Figure 3-37.
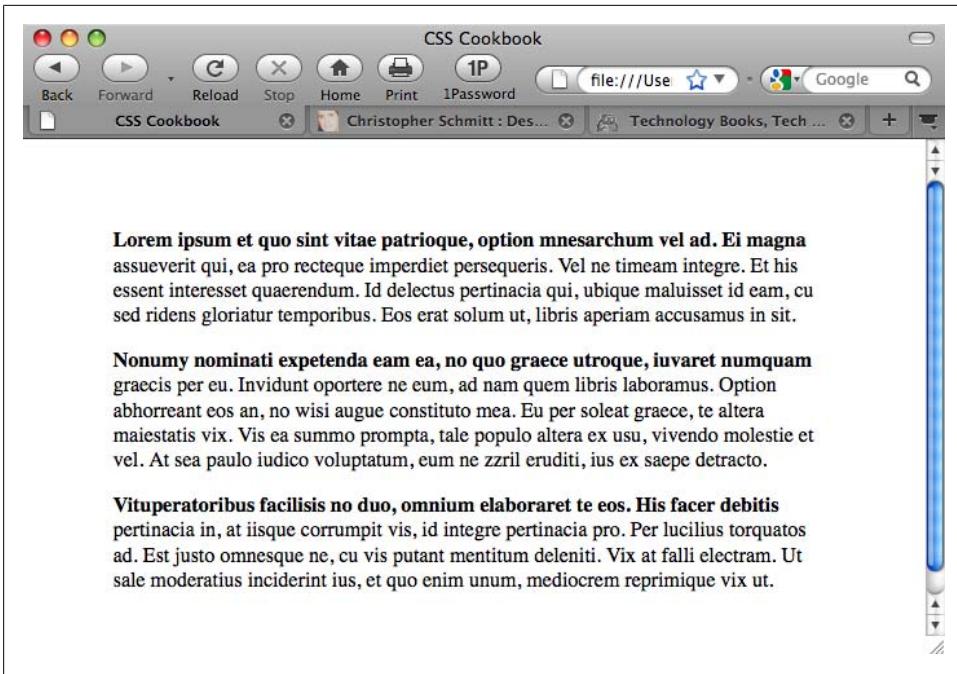
Figure 3-36. The first line set to bold
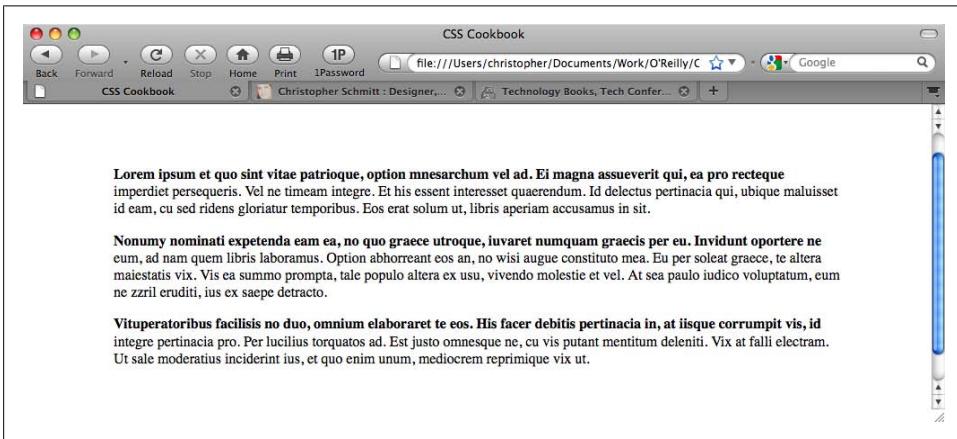
Figure 3-37. The amount of text changing when the browser is resized

See Also

The CSS 2.1 specification for `:first-line` at *http://www.w3.org/TR/CSS21/selector .html#first-line-pseudo*

3.27 Styling the First Line of a Paragraph with an Image

Problem

You want to stylize the first line of a paragraph and include an image, as shown in Figure 3-38.

Figure 3-38. The first line with a background image

Solution

Use the `background-image` property within the `:first-line` pseudo-element:

```
p:first-line {
 font-size: 2em;
 background-image: url(background.gif);
}
```

Discussion

With the `:first-line` pseudo-element, you can apply styles only to the first line of text of an element, and not the width of the element itself.

In addition to the `background-image` property, the `:first-line` pseudo-element also supports the following properties, allowing for greater design control:

- `font`
- `color`
- `background`
- `word-spacing`
- `letter-spacing`

- text-decoration
- vertical-align
- text-transform
- text-shadow
- line-height
- clear

See Also

The CSS 2.1 specification for :first-line at *http://www.w3.org/TR/CSS21/selector .html#first-line-pseudo*

3.28 Creating a Highlighted Text Effect

Problem

You want to highlight a portion of the text in a paragraph, as in Figure 3-39.
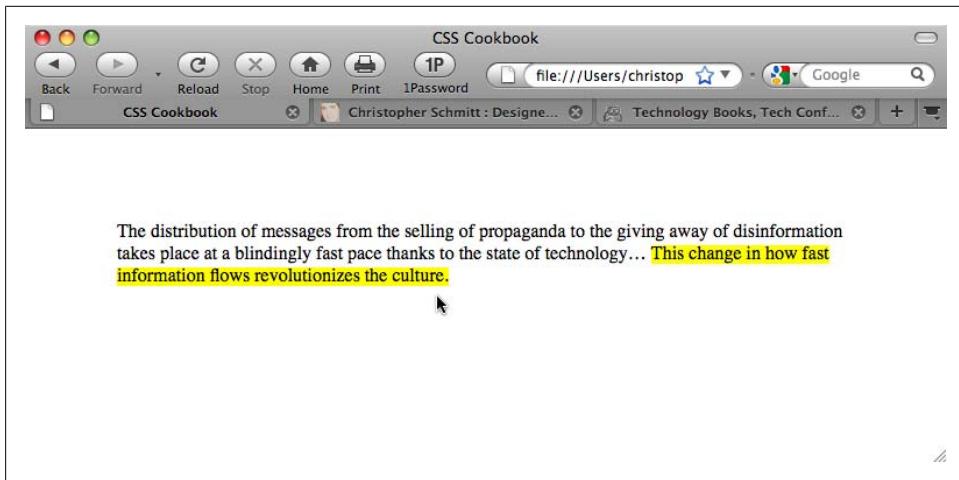
Figure 3-39. Highlighted text

Solution

Use the strong element to mark up the portions of text you want to highlight:

```
<p>The distribution of messages from the selling of propaganda
to the giving away of disinformation takes place at a blindingly
fast pace thanks to the state of technology&hellip; <strong>This
change in how fast information flows revolutionizes the
culture.</strong></p>
```

Then set the CSS rule to set the highlighted text through the `background-color` property:

```
strong {
 font-weight: normal;
 background-color: yellow;
}
```

Discussion

Although the `strong` element is used in this Solution, you also can use the `em` element instead of the `strong` element to mark highlighted text. The HTML 4.01 specification states that you should use `em` for marking *emphasized* text, and use `strong` to indicate "stronger emphasis."

Once the text has been marked, set the highlighter color with the `background-color` property. Because some browsers apply a bold weight to text marked as `strong`, set the `font-weight` to `normal`. When using the `em` element, be sure to set the `font-style` to `normal` as this keeps browsers from setting the type in italic, as shown in the following code listing:

```
em {
 font-style: normal;
 background-color: #ff00ff;
}
```

See Also

The HTML specification for `strong` and `em` at *http://www.w3.org/TR/html401/struct/text.html#edef-STRONG*

3.29 Changing the Text Selection Color

Problem

You want to set the color of highlighted text when it is selected, as shown in Figure 3-40.

Solution

Use the `::selection` pseudo-element to set both the color and the background color of text:

```
::selection {
 color: #90c;
 background: #cf0;
}
```

Discussion

At the time of this writing, the only browser supporting the `::selection` pseudo-element is Safari. However, Firefox has its own proprietary CSS selector.
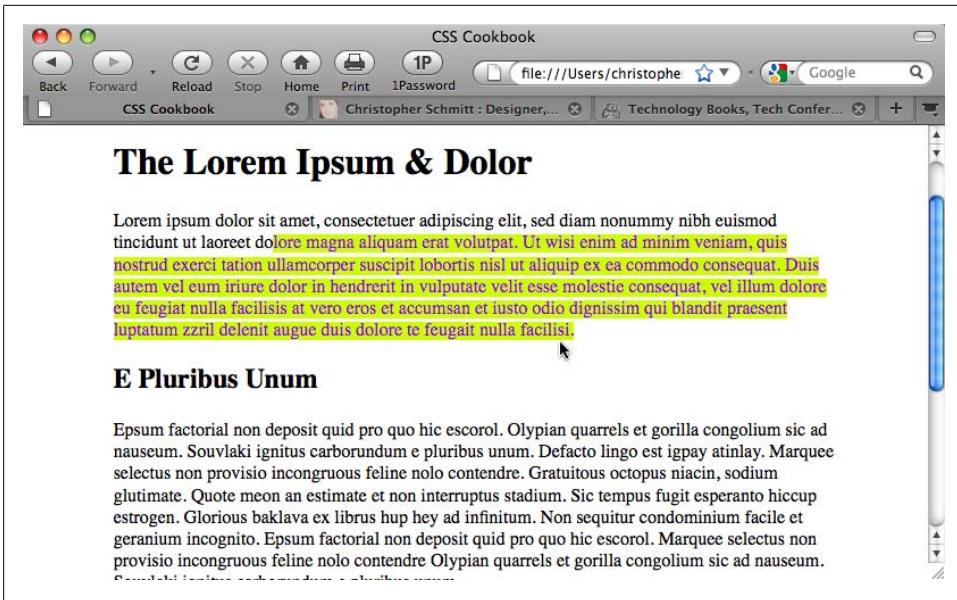
Figure 3-40. Color set when selecting a passage of text with the mouse

To include support for Firefox in conjunction with Safari, duplicate the `::selection` CSS rule for the `::-moz-selection` property:

```
::selection {
 color: #90c;
 background: #cf0;
}
::-moz-selection {
 color: #90c;
 background: #cf0;
}
```

See Also

The CSS3 specification for UI element fragments at *http://www.w3.org/TR/2001/CR -css3-selectors-20011113/#UIfragments*

3.30 Changing Line Spacing

Problem

You want to leave more or less space between lines. Figure 3-41 shows the browser default, and Figure 3-42 shows paragraphs with more space between lines.
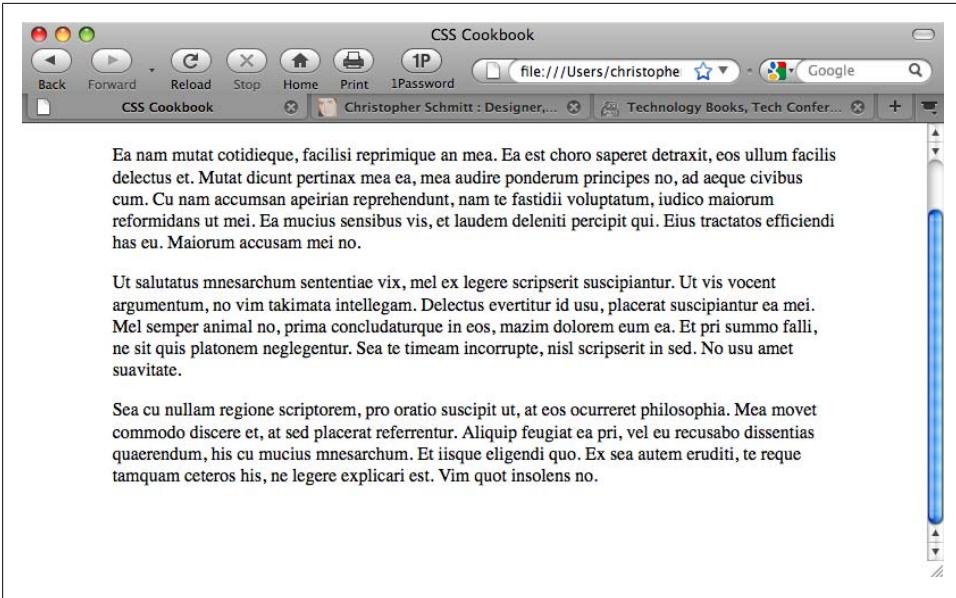
Figure 3-41. The default leading of a paragraph
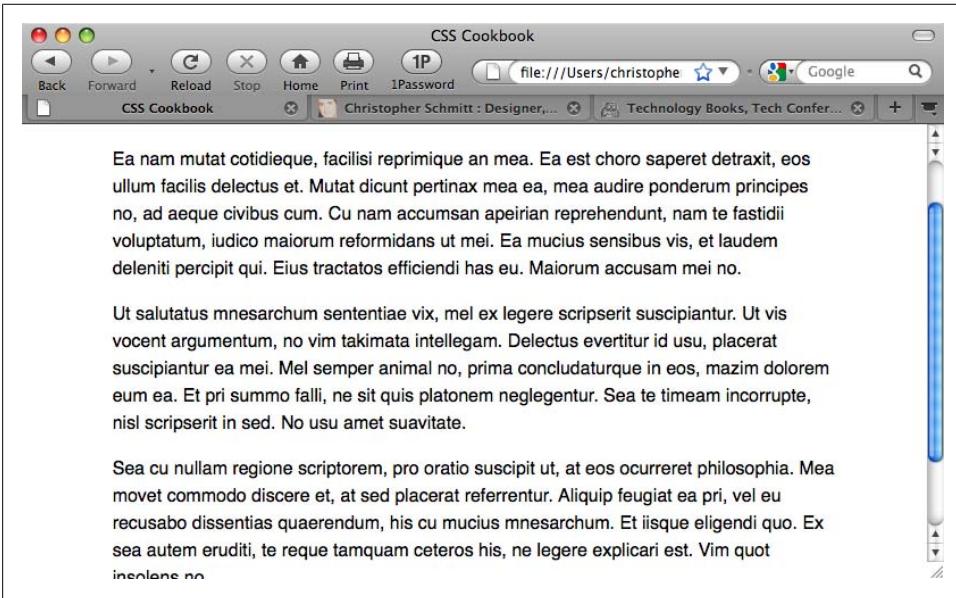
Figure 3-42. Increased leading between the lines of text

Solution

Use the `line-height` property:

```
p {
  line-height: 1.5em;
}
```

Discussion

As the `line-height` value increases, the distance between the lines of text grows. As the value decreases, the distance between the lines of text shrinks, and eventually the lines overlap each other. Designers notice a similarity to line height and *leading*.

A `line-height` value can be a number and a unit such as points, just a number, or a number and a percent symbol. If the `line-height` value is just a number, that value is used as a percentage or a scale unit for the element itself as well as for child elements. Negative values aren't allowed for `line-height`.

The following example effectively sets the `font-size` to 12 pixels and the `line-height` to 14.4 pixels [(10px * 1.2) * 1.2px = 14.4px]:

```
body {
  font-size: 10px;
}
p {
  font-size: 1.2em;
  line-height: 1.2;
}
```

You also can set the `line-height` property with the shorthand `font` property when paired with a `font-size` value. The following line transforms any text in a `p` element to have a font size of 1 em, to have a `line-height` of 1.5 em, and to display in a sans serif typeface:

```
p {
  font: 1em/1.5em sans-serif;
}
```

See Also

The CSS 2.1 specification for `line-height` at *http://www.w3.org/TR/CSS21/visudet.html #propdef-line-height*; Recipe 3.15 for more information on the `font` property

3.31 Adding a Graphic Treatment to HTML Text

Problem

You want to apply a repeating graphic treatment on top of HTML text—for example, worn edges or stripes—as shown in Figure 3-43.
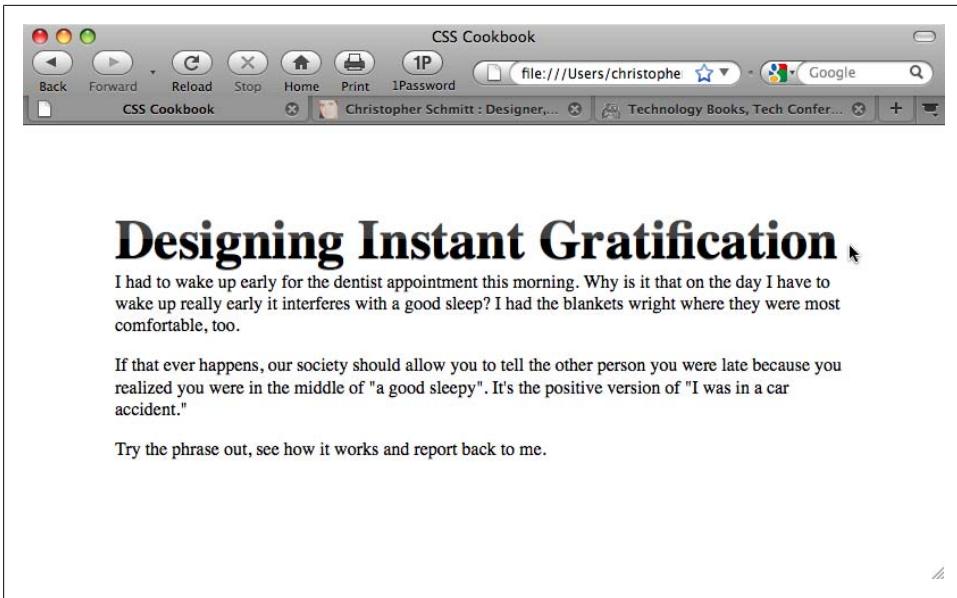
Figure 3-43. A PNG image repeating over the top half of the HTML text to create a glassy appearance

Solution

Place a span element after the opening tag of a heading element, but before the HTML text:

```
<h2><span></span>Designing Instant Gratification</h2>
```

Next, use a version of the Gilder/Levin image replacement technique (*http://www.mez zoblue.com/tests/revised-image-replacement/#gilderlevin*) to place a PNG file with a seamless pattern over the HTML text:

```
h2 {
 font:3em/1em Times, serif;
 font-weight: bold;
 margin:0;
 position: relative;
 overflow: hidden;
 float: left;
 text-shadow: 0 1px  0 rgba(153,153,153,.8);
}
h2 span {
 position: absolute;
 width: 100%;
 height: 5em;
 background: url(title-glass.png);
}
p {
 clear: left;
}
```

Discussion

The text within the heading element is set to float to the left. This technique is designed to allow the background image, placed in the `span` element, to be placed over the HTML text through absolute positioning.

Normally, when floating an element the heading would move to the left and the content would wrap on the right side. However, the `clear` property placed on the paragraph stops this from happening.

The `height` property is set to 5 em and the `overflow` property is set to a value of `hidden` to keep the background image from spilling out of the heading element and onto the other portions of the web document, as in the preceding paragraph.

See Also

http://www.mezzoblue.com/tests/revised-image-replacement/#gilderlevin for additional information on the Gilder/Levin image replacement technique

3.32 Placing a Shadow Behind Text

Problem

You want to place a shadow behind the text in a heading, as shown in Figure 3-44.
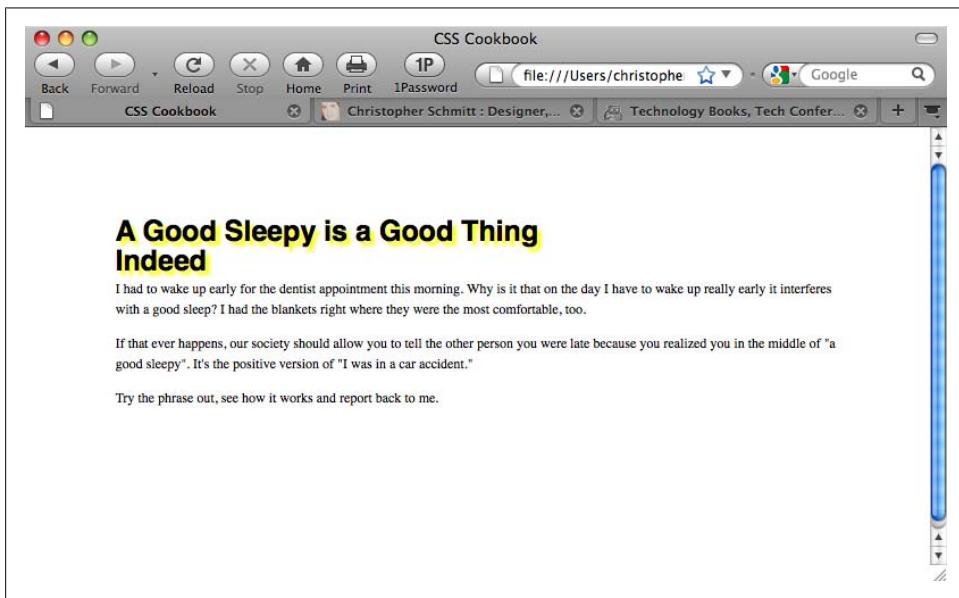
Figure 3-44. Instant drop shadows on HTML text

Solution

Use the `text-shadow` property to set the color and placement of the shadow:

```
h1 {
 font-size: 2.5em;
 font-family: Myriad, Helvetica, Arial, sans-serif;
 width: 66.6%;
 text-shadow: yellow .15em .15em .15em;
 margin: 0 0 0.1em 0;
}
```

Discussion

The first value of the `text-shadow` property sets the color. The first length unit value, `.15em`, moves the shadow on the x-axis relative to the position of the HTML text. The next value moves the shadow on the y-axis. The last value is the blur radius of the shadow. The larger the value the more disperse the shadow.

Setting the opacity of the shadow

By setting the color of the shadow using RGBA, you can set the color to a level of opacity. This would allow the shadow color to blend better into the background:

```
body {
 Background-color: #000;
}
h1 {
 font-size: 2.5em;
 font-family: Myriad, Helvetica, Arial, sans-serif;
 width: 66.6%;
 text-shadow: rgba(205, 205, 0, .7) .15em .15em .15em;
 margin: 0 0 0.1em 0;
}
```

Creating a bevel look

By setting the distance of the shadow to one pixel off to the left along with 60% opacity, you can accomplish a simple bevel effect with the `text-shadow` property, as shown in Figure 3-45:

```
body {
 background-color: #999;
}
h1 {
 text-shadow: 0 1px 0 rgba(255,255,255,.6);
}
```
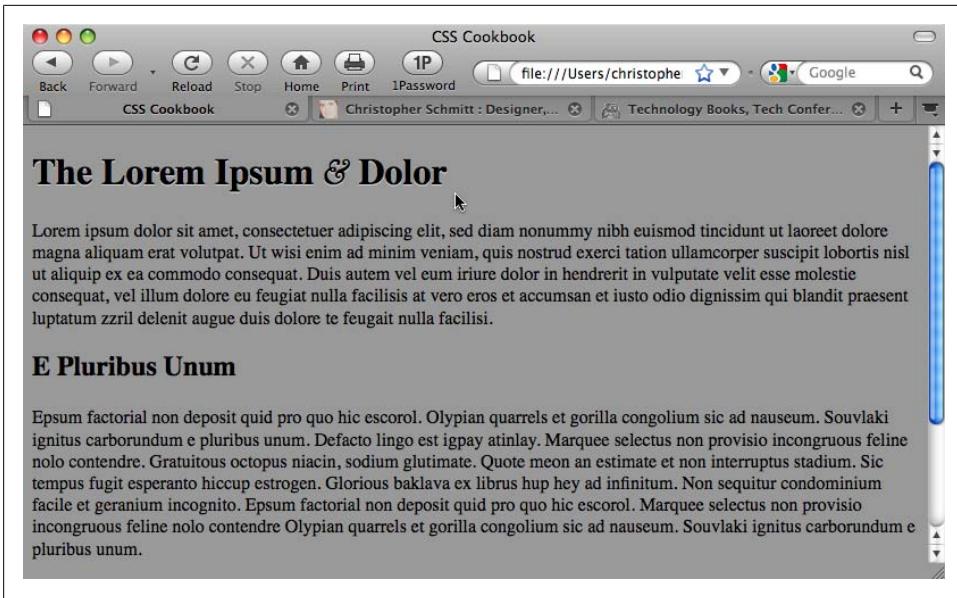
Figure 3-45. Bevel look with a text-shadow

Add a red flame to the top of text

The `text-shadow` property can take more than one value (with each value separated by a comma). This technique can allow you to create interesting effects (depending on your point of view), such as a red flame on top of a heading, as shown in Figure 3-46:

```
h1 {
 color: red;
 text-shadow: rgba(0, 0, 0, .9) 0px 0px 1px,
  rgba(255, 255, 51, .9) 0px -5px 5px,
  rgba(255, 204, 51, .7) 2px -10px 7px,
  rgba(255, 153, 0, .6) -2px -15px 10px;
}
```

Known support

The only known browsers that support the `text-shadow` property are Firefox 3.5 and later, Opera 9.5 and later, and Safari.

Text shadow for Internet Explorer

To set a text shadow for Internet Explorer 6 and later, use the proprietary `filter` property:

```
h2 {
 filter:shadow(color=#999999,direction=270, strength=1);
}
```
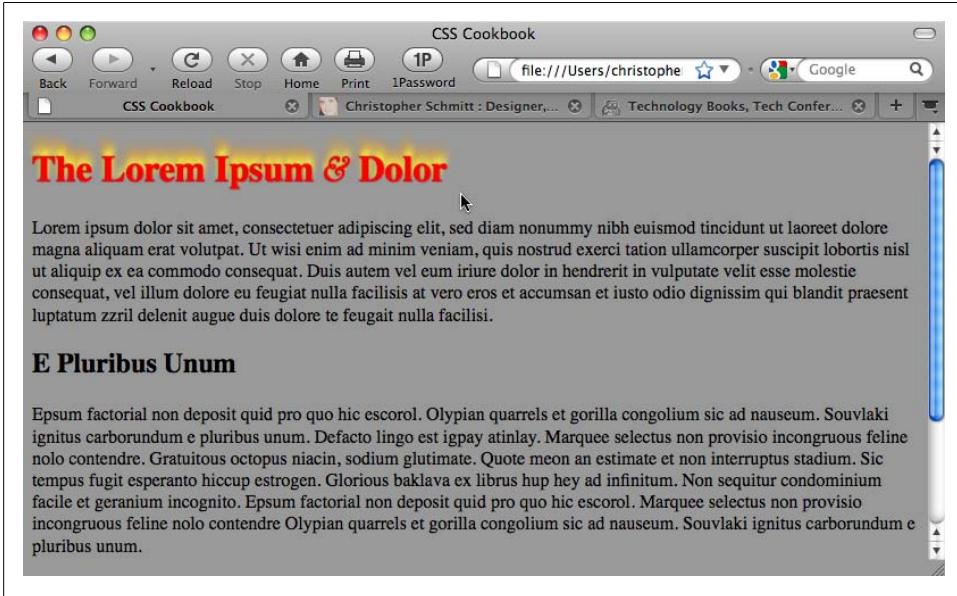
Figure 3-46. Adding a red flame to text

In the preceding code, the `color` property is set with a hexadecimal value, `direction` is a value between 0 and 360, and `strength` is the length of the shadow set in pixels.

> In IE8, Microsoft is transitioning `filter` and other properties to use CSS vendor extensions. See *http://blogs.msdn.com/ie/archive/2008/09/08/microsoft-css-vendor-extensions.aspx* for more information.

See Also

The CSS 2.1 specification for `text-shadow` at *http://www.w3.org/TR/REC-CSS2/text.html#text-shadow-props*

3.33 Adjusting the Space Between Letters and Words

Problem

You want to adjust the space between letters and words within HTML text.

Solution

To adjust the space between letters, use the `letter-spacing` property, as shown in Figure 3-47:

```
h2 {
 font: bold italic 2em "Helvetica Nue", serif;
 margin: 0;
 padding: 0;
 letter-spacing: -0.1em;
}
```
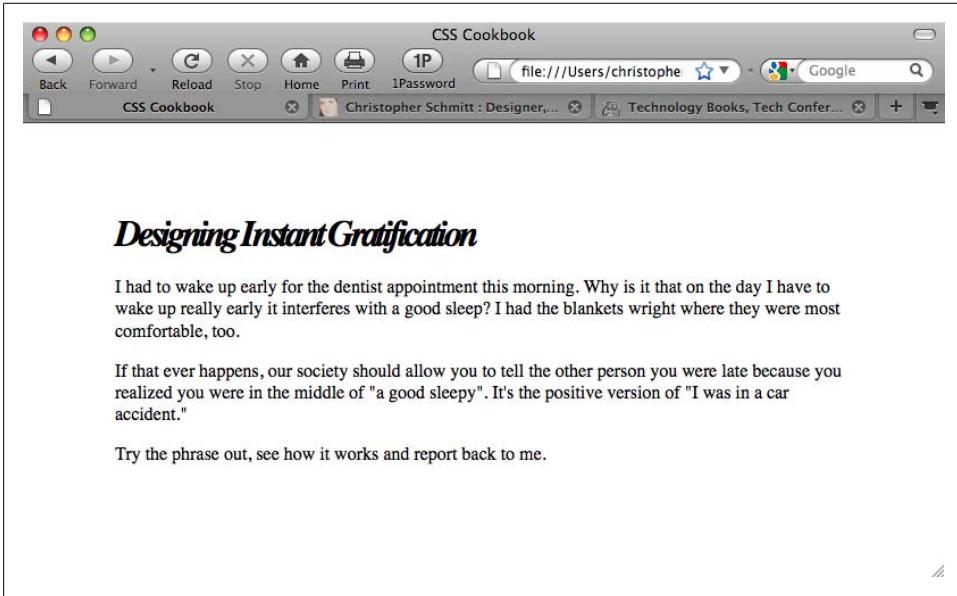

Figure 3-47. The styled letter spacing of the text in the heading

To adjust the space between words, use the `word-spacing` property, as shown in Figure 3-48:

```
h2 {
 font: bold italic 2em "Helvetica Nue", serif;
 margin: 0;
 padding: 0;
 word-spacing: 0.33em;
}
```

Discussion

One of the main strengths of CSS is how the technology handles web typography. Web designers and developers no longer have to use a puzzling array of nested fonts, b elements, and single-pixel GIF tricks to create compelling text treatments. An effect such as adjusting the space between two letters or separating whole words within a paragraph is exactly something that CSS can render with ease.
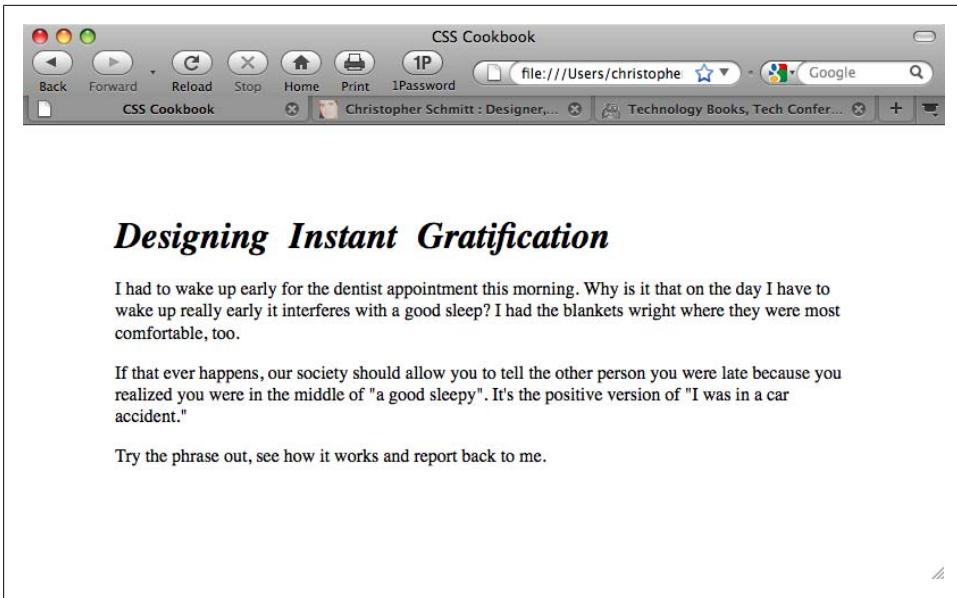
Figure 3-48. Words in the heading spaced farther apart

Kerning and tracking

Adjusting the space between letters to create a better aesthetic is an old tradition in graphic design. Two terms describe how the change in space is adjusted: *kerning* and *tracking*.

Kerning is a design term used to describe the process of changing the space between a pair of letters to create a better visual effect. An example of kerning is adjusting just the space between an uppercase letter *T* and a lowercase letter *i*. *Tracking* is defined as involving more than a pair of letters to the point of adjusting the space between letters to large amounts of text.

The `word-spacing` property is supported in Firefox, Internet Explorer 6 for Windows and later, Opera 3.5 and later, and Safari.

Best practices

A best practice is to set the values of `letter-spacing` and `word-spacing` in relative unit sizes instead of absolute length units. Since users can redefine the font sizes of their browsers, a fixed width value of 5 points originally intended for a font size of 12 pixels will still be 5 points, even if the user resizes the text to a larger value. In other words, the 5-point spacing between letters is barely going to be noticeable when the font size is set to 72 pixels or larger. With relative units such as em, however, a value of `1.5em` for the `letter-spacing` property scales along with the resizing of the text.

Also, it's best to employ text effects so that the text being styled is still legible. If communication is important to you or your client, a subtle effect is better than creating esoteric text elements. As the text becomes illegible, you might annoy the very same people you are trying to reach.

See Also

The CSS 2.1 specification for `letter-spacing` at *http://www.w3.org/TR/CSS2/text.html #propdef-letter-spacing* and for `word-spacing` at *http://www.w3.org/TR/CSS2/text.html #propdef-word-spacing*; *http://desktoppub.about.com/cs/typespacing/a/kerningtracking .htm* for more on kerning and tracking

3.34 Applying Baseline Rhythm on Web Typography

Problem

You want to set two columns of text on the same baseline, as shown in Figure 3-49.
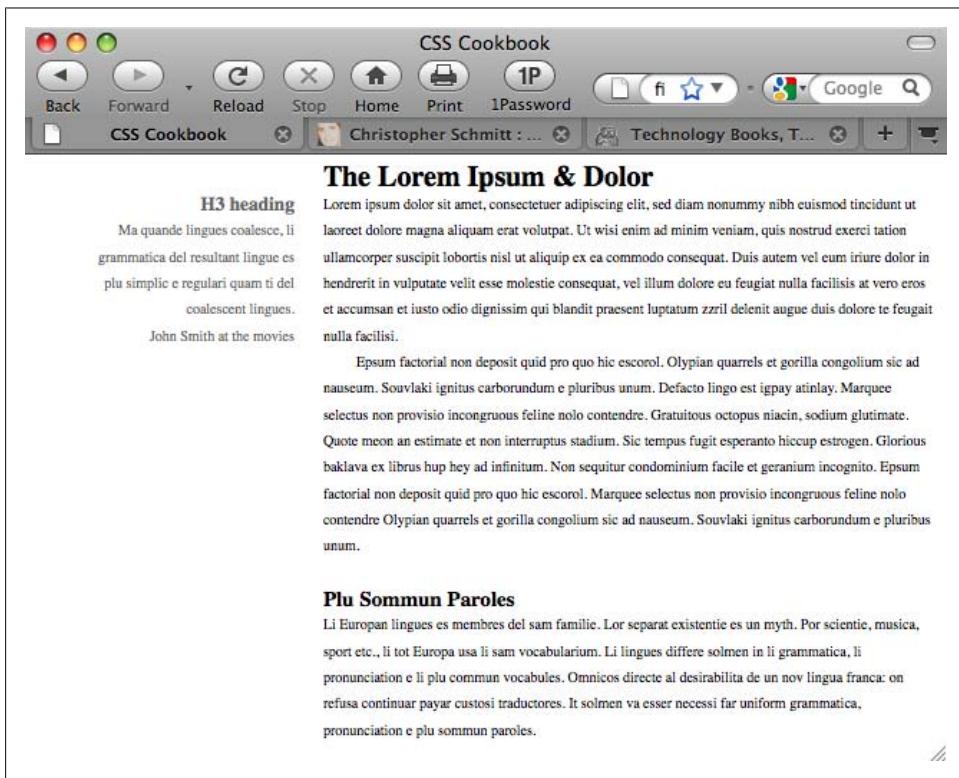
Figure 3-49. Column text lined up on the same baseline (with lines added for emphasis)

Solution

As stated in Recipe 3.6, set the `font-size` on the `body` selector to 62.5%:

```
body {
 font-size: 62.5%
}
```

Next, set the `line-height` (or leading), as discussed in Recipe 3.30:

```
body {
 font-size: 62.5%;
 line-height: 1.83em;
}
```

Determine the `line-height` of the other type-related HTML elements using the following formula:

(body `line-height` / `font-size` of the HTML element) = HTML element's `line-height` in em units

For the `h2` element with a `font-size` of `1.5em`, the quotient is `1.2em`:

```
(1.83em / 1.5em ) = 1.2em
```

Update the CSS rules to include this new `line-height` value for the `h2` element:

```
body {
 font-size: 62.5%;
 line-height: 1.83em;
}
h2 {
 margin: 0;
 font-size: 1.5em;
 line-height: 1.2em;
}
```

To ensure that the margins of the `h2` element stay in tune with the `line-height` property, apply the same value:

```
body {
 font-size: 62.5%;
 line-height: 1.83em;
}
h2 {
 margin: 0;
 font-size: 1.5em;
 line-height: 1.2em;
 margin-bottom: 1.2em;
}
```

Do the same calculation and setup for the rest of the type-related elements.

Discussion

Although the effect of lining up text in two or more columns along the same baseline parlays a sense of professional craftsmanship often lacking in most web pages, its requirement for detail-oriented calculations could make even the most patient web designer a little frustrated, especially if that designer or her client requests changes in the `font-size` value. That seemingly simple request results in a new set of calculations.

To help with that approach, web designer Geoffrey Grosenbach created a Baseline Rhythm Calculator (see *http://topfunky.com/baseline-rhythm-calculator/*) to alleviate people's suffering.

See Also

Richard Rutter's article on vertical rhythm at *http://24ways.org/2006/compose-to-a-vertical-rhythm*

3.35 Styling Superscripts and Subscripts Without Messing the Text Baseline

Problem

You want to add superscripts and subscripts without adjusting the baseline of the text, as shown in Figure 3-50.

Solution

Use the HTML elements `sup` and `sub` to set superscripts and subscripts, respectively:

```
<p>Lorem ipsum dolor sit amet, consectetuer adipiscing elit
<sup><a href="#footnote1">1</a></sup>, sed diam nonummy nibh euismod tincidunt
ut laoreet dolore magna aliquam erat volutpat. Ut wisi enim ad minim
veniam<sup><a href="#footnote2">2</a></sup>, quis nostrud exerci tation
ullamcorper suscipit lobortis nisl ut aliquip ex ea commodo consequat.
Duis autem vel eum iriure dolor in hendrerit in vulputate velit esse molestie
consequat, vel illum dolore eu feugiat nulla facilisis at vero eros et accumsan
et H<sub>2</sub>O iusto odio dignissim qui blandit praesent luptatum zzril
delenit augue duis dolore te feugait nulla facilisi.<p>
```

Then adjust the alignment of the text within the `sup` and `sub` elements:

```
sup, sub {
 vertical-align: baseline;
 position: relative;
 top: -0.4em;
}
sub {
 top: 0.4em;
}
```

file:///Users/christopher/Documents/Work/O

Q▾ Google

Bookmarketlets ▾   In Control Conference ▾   CSS Prism   ChoiceTweets   Twitshirt This!

CSS Cookbook   Christopher Schmitt : Desi...   Technology Books, Tech C...

The Lorem Ipsum & Dolor

Ma quande lingues coalesce, li grammatica del resultant lingue es plu simplic e regulari quam ti del coalescent lingues.

John Smith at the movies

Lorem ipsum dolor sit amet, consectetuer adipiscing elit[1], sed diam nonummy nibh euismod tincidunt ut laoreet dolore magna aliquam erat volutpat. Ut wisi enim ad minim veniam[2], quis nostrud exerci tation ullamcorper suscipit lobortis nisl ut aliquip ex ea commodo consequat. Duis autem vel eum iriure dolor in hendrerit in vulputate velit esse molestie consequat, vel illum dolore eu feugiat nulla facilisis at vero eros et accumsan et H_2O iusto odio dignissim qui blandit praesent luptatum zzril delenit augue duis dolore te

Figure 3-50. The baseline shifting in the Safari browser with the addition of superscripts and subscripts

Discussion

The Solution works by snapping the text within the sup and sub elements to the baseline just like the rest of the text. Then you can position the text off of the baseline through the use of relative positioning (see Recipe 2.24) to re-create the desired appearance of superscript and subscript.

See Also

http://paularmstrongdesigns.com/weblog/stop-superscripts-from-breaking-line-heights -once-and-for-all for web designer Paul Armstrong's blog post about this technique

3.36 Setting Up Multiple Columns of Text

Problem

You want to set a long passage of text into multiple columns, as shown in Figure 3-51.
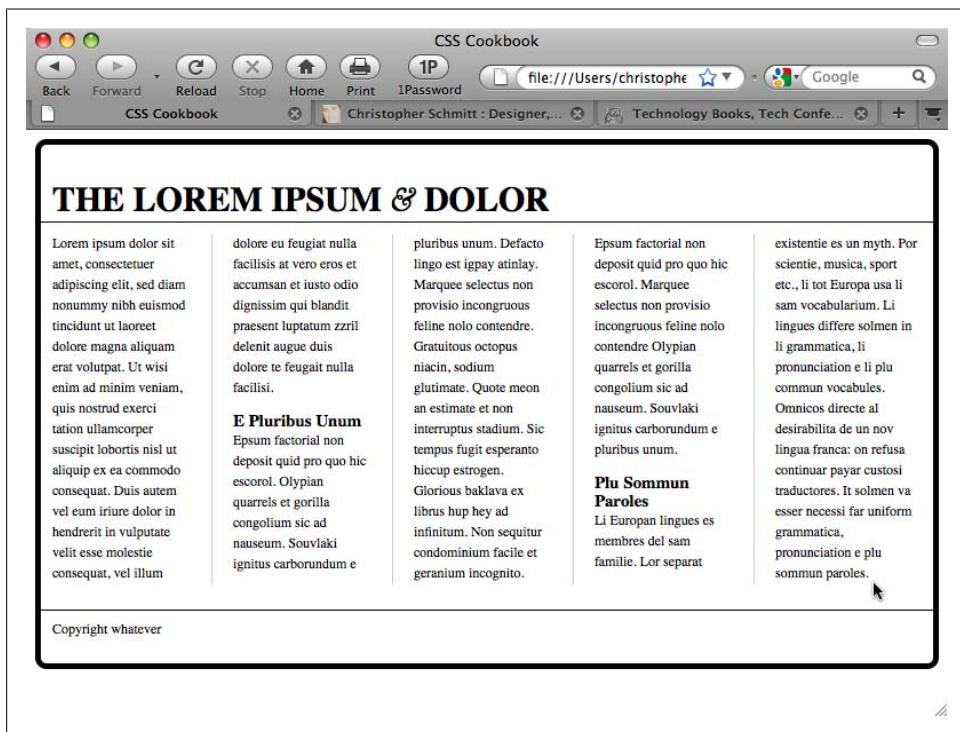
Figure 3-51. Words in the heading spaced farther apart

Solution

Wrap a `div` element around the content passage to set it in columns:

```
<div id="column">
 <p>...<p>
 <h2>...</h2>
 <p>...<p>
 <h2>...</h2>
 <p>...<p>
</div>
```

Use proprietary `column-gap` and `column-width` tags:

```
#column {
 -moz-column-gap: 3em;
 -moz-column-width: 11em;
 -webkit-column-gap: 3em;
 -webkit-column-width: 11em;
 padding: 10px;
}
```

Then set line rules using the proprietary `-column-rule` properties:

```
#column {
 -moz-column-gap: 3em;
 -moz-column-width: 11em;
 -moz-column-rule: 1px solid #ccc;
 -webkit-column-gap: 3em;
 -webkit-column-width: 11em;
 -webkit-column-rule: 1px solid #ccc;
 padding: 10px;
}
```

Discussion

The use of the column properties saves web designers time as setting columns of text is a laborious process.

To achieve the column effect for a design, web designers would need to count the number of words for each column to make sure each column had an equal number of words; set each equal number of words with their own `div` element; and individually position or float those `div` elements into place.

Known issues

The CSS3 column properties make the process of setting columns easy and automatic for web designers. The main problem is that they are supported only through proprietary CSS extensions in Firefox and Safari.

A JavaScript solution through a jQuery plug-in provides an alternative that avoids the use of proprietary CSS properties (see *http://welcome.totheinter.net/2008/07/22/multi-column-layout-with-css-and-jquery/*).

For techniques on how to set up column layouts, see Chapter 10.

See Also

The Peter-Paul Koch test column properties at *http://www.quirksmode.org/css/multicolumn.html*

Images

4.0 Introduction

When Marc Andreessen's first browser allowed for the inline display of images back in the early 1990s, it helped to kick-start a visually engaging aspect of surfing the Web. Shared documents no longer were just text-laden academic papers, allowing designers the initial foothold to begin the field of web design.

Since those early days, designers have been using GIFs, JPEGs, and PNGs to enhance websites beyond the placement of one or two images on a web page.

In this chapter, we'll discuss many recipes regarding CSS interactions with images. Recipes include dealing with borders, manipulating background images, rounding corners on boxes, replacing HTML text with images, and much more.

4.1 Transforming Color Images to Black and White in IE with CSS

Problem

You want to convert color images in a web page to grayscale versions in Internet Explorer.

Solution

Use the proprietary `filter` CSS property to automatically convert images to grayscale:

```
img {
 filter: gray;
}
```

In IE8, Microsoft is transitioning `filter` and other properties to use CSS vendor extensions. For more information, see *http://blogs.msdn.com/ie/ archive/2008/09/08/microsoft-css-vendor-extensions.aspx*.

Discussion

Although not the most useful CSS property, `filter` does have its uses.

One example is to set images to gray for print stylesheets (see Chapter 11). This approach saves your user money, as color inks are more expensive than black ink.

Another example is to craft custom stylesheets for older versions of Internet Explorer with conditional comments (see Recipe 12.7), setting all the imagery to be black and white. This approach is what web designer Andy Clarke did with his site redesign (see *http://stuffandnonsense.co.uk/blog/about/hello/*).

See Also

MSDN documentation on the grayscale `filter` property at *http://msdn.microsoft.com/ en-us/library/ms533003(VS.85).aspx*

4.2 Setting a Border Around an Image

Problem

You want to place a border around an image.

Solution

Use the `border` property on the `img` element, as shown in Figure 4-1:

```
img {
 width: 300px;
 border: 2px solid #666;
 background: #fff;
 padding: 2px;
}
```

Discussion

If you make an image a link, you can create a more complex presentation with the `border` property.

Using the `:hover` pseudo-class, you can change the style of the border when a user rolls his mouse cursor over the image, as shown on the right side of Figure 4-1:

```
img {
 width: 300px;
 border: 2px solid #666;
```

```
  background: #fff;
  padding: 2px;
}
a:hover img {
  border-style: solid;
  background: #999;
}
```
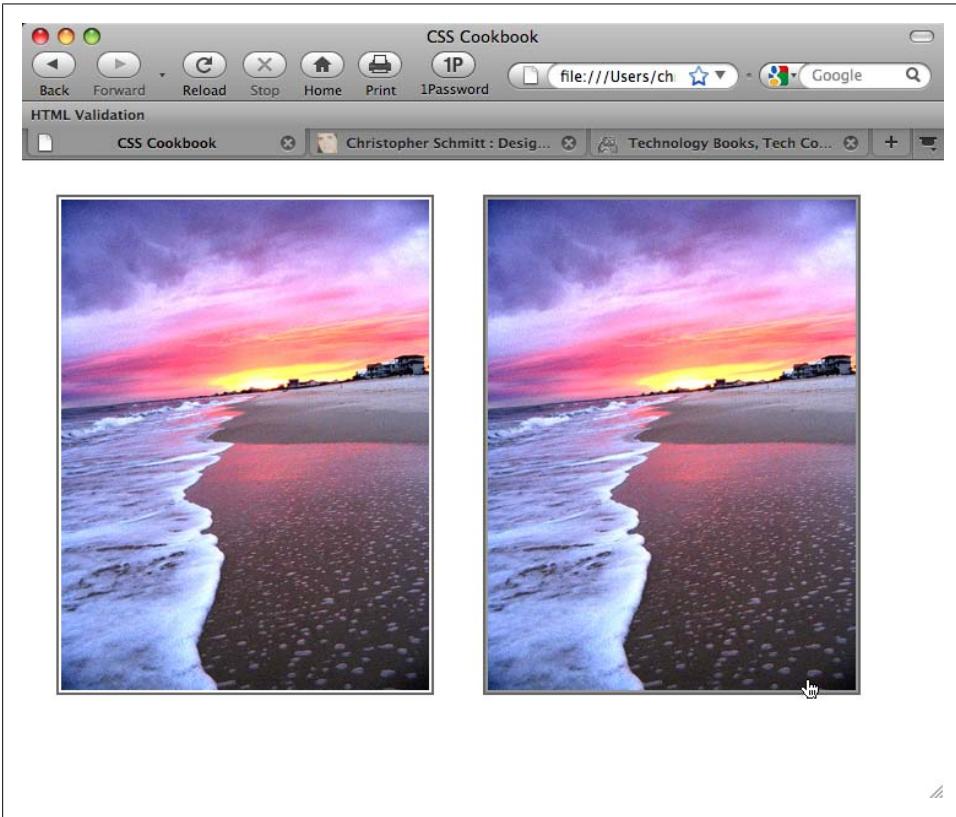

Figure 4-1. A border placed around an image

Although the border acts like a frame around the image, you can change the border style and color when a user rolls his mouse cursor over the image. The padding of `2px` set in the `img` declaration block allows for color changes inside this frame as well. So, a simple move of rolling over an image creates a rich visual with only two declaration blocks.

See Also

Recipe 4.4 for removing borders from images

4.3 Setting a Rounded Border Around an Image

Problem

You want to round the right-angle corners of an image border.

Solution

Set the border value and then use the CSS3 `border-radius` property along with its browser-specific `border-radius` properties, as shown in the right side of Figure 4-2:

```
div{
 background-image: url(beach.jpg);
 width: 375px;
 height: 500px;
 border: 8px solid #666;
 border-radius: 40px;
 -moz-border-radius: 40px;
 -webkit-border-radius: 40px;
}
```
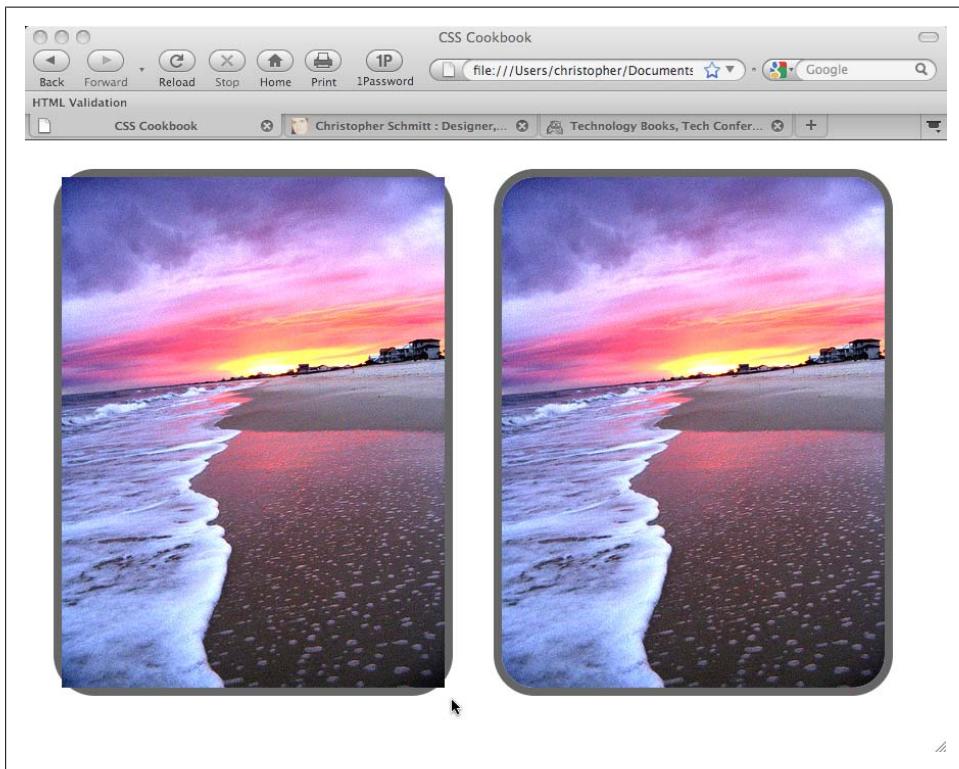

Figure 4-2. Rounded borders on the right side

Discussion

The radius is half the distance of a circle's diameter and is used to set the amount of curvature on the corner. The higher the value for the radius, the more rounded the corner will be.

At the time of this writing, the `border-radius` property isn't supported as is; however, the proprietary properties in both Firefox and Safari replicate the effect. The main drawback (other than cross-browser support) is that the names of the border properties are not consistent, as shown in Table 4-1.

Table 4-1. Rounded corner property equivalents

CSS3	Firefox	WebKit
border-**radius**	-moz-border-**radius**	-webkit-border-**radius**
border-**top-left-radius**	-moz-border-**radius-topleft**	-webkit-border-**top-left-radius**
border-**top-right-radius**	-moz-border-**radius-topright**	-webkit-border-**top-right-radius**
border-**bottom-right-radius**	-moz-border-**radius-bottomright**	-webkit-border-**bottom-right-radius**
border-**bottom-left-radius**	-moz-border-**radius-bottomleft**	-webkit-border-**bottom-left-radius**

Specifying corners

Rounded corners are also rendered on individual corners, not just all four corners. To set the rounded effect on only one or a few corners, specify each rounded corner individually in the CSS rule.

For example, the following CSS rule defines that all corners be rounded except for the top-right corner:

```
div#roundbkgd {
  background-image: url(beach.jpg);
  width: 375px;
  height: 500px;
  border: 8px solid #666;

  /* top-left corner */
  border-top-left-radius: 40px;
  -moz-border-radius-topleft: 40px;
  -webkit-border-top-left-radius: 40px;

  /* bottom-right corner */
  border-bottom-right-radius: 40px;
  -moz-border-radius-bottomright: 40px;
  -webkit-border-bottom-right-radius: 40px;

  /* bottom-left corner */
```

```
    border-bottom-left-radius: 40px;
    -moz-border-radius-bottomleft: 40px;
    -webkit-border-bottom-left-radius: 40px;
}
```

Known issues

If the image is inline, or placed within the HTML and not as a background image, the rounded borders are shown behind the image instead of clipping the image, as shown on the left side of Figure 4-2:

```
img {
  width: 375px;
  height: 500px;
  border: 8px solid #666;
  background: #fff;
  display:block;
  border-radius: 40px;
  -moz-border-radius: 40px;
  -webkit-border-radius: 40px;
}
```

To work around this problem, keep the value of the `border-radius` property relatively small (no more than four or five pixels) or set the image within the background of an element (see Recipe 4.5).

Opera is scheduled to support `border-radius` for the next major release after Opera 10.

See Also

The CSS3 specification for `border-radius` at *http://www.w3.org/TR/2005/WD-css3 -background-20050216/#the-border-radius*

4.4 Removing Borders Set on Images by Default in Some Browsers

Problem

You want to remove borders on images that are clickable, as shown in Figure 4-3.

Solution

Set the border for images to `0`:

```
a img {
 border: 0;
}
```
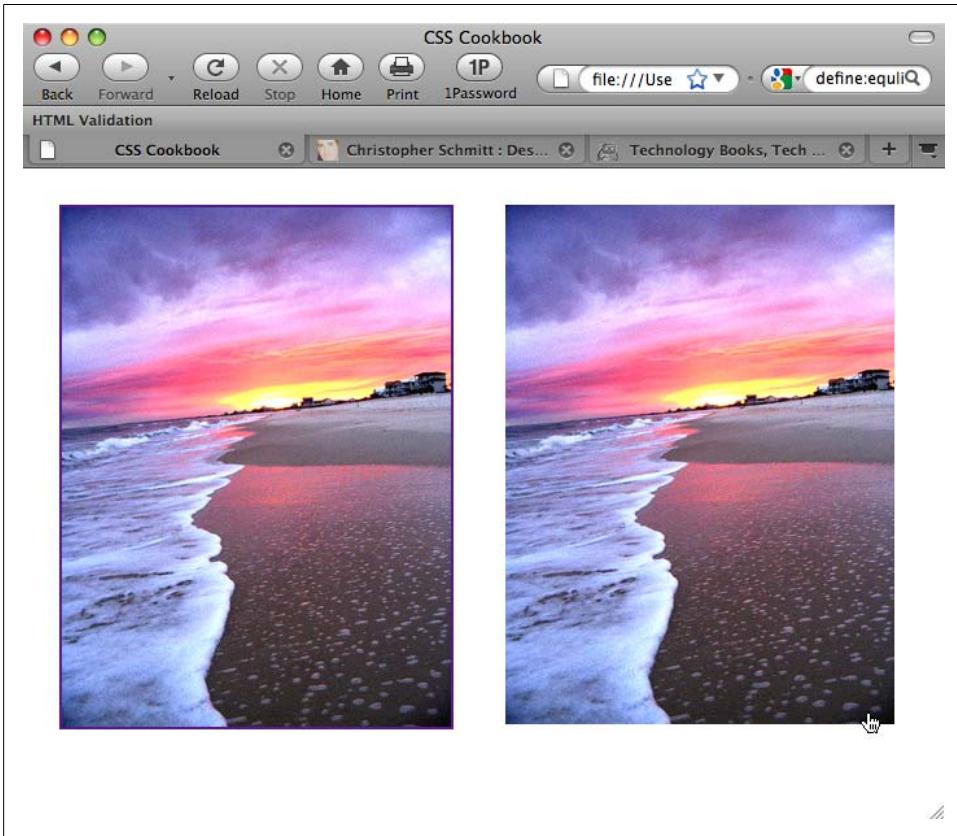

Figure 4-3. An image with a border and one without

Discussion

Before CSS, web developers would set the border of images through the `border` attribute of the `img` element:

```
<a href="http://csscookbook.com">
 <img src="beach.jpg" border="0" alt="beach" />
</a>
```

See Also

Recipe 4.2 for applying a border to an image

4.5 Setting a Background Image

Problem

You want a background image that does not repeat.

Solution

Use the `background-image` and `background-repeat` properties to control the display of an image (see Figure 4-4):

```
body {
 background-image: url(bkgd.jpg);
 background-repeat: no-repeat;
}
```
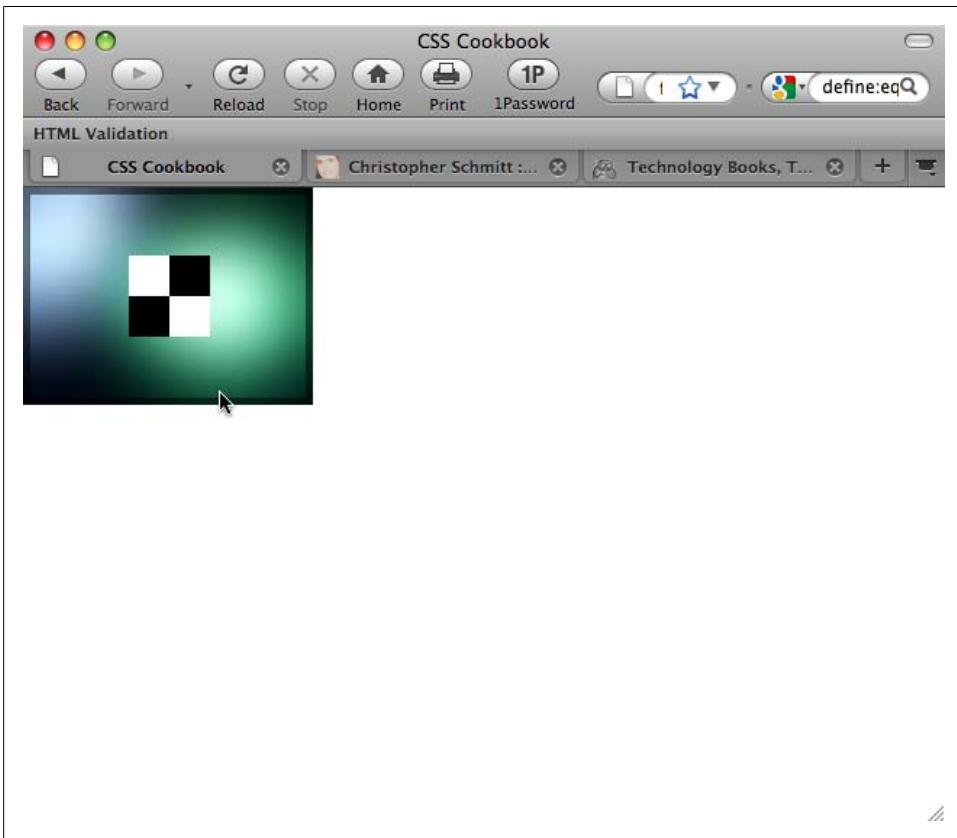

Figure 4-4. The background image displayed once in the upper-left corner

Discussion

You can place text and other inline images over a background image to create a sense of depth on a web page. Also, you can provide a framing device for the web page by tiling a background image along the sides of a web browser.

See Also

Recipe 4.6 for repeating background images in a line either horizontally or vertically

4.6 Creating a Line of Background Images

Problem

You want a series of background images to repeat vertically or horizontally.

Solution

To tile the background images horizontally or along the x-axis, use the following CSS rule (see Figure 4-5):

```
body {
 background-image: url(bkgd.jpg);
 background-repeat: repeat-x;
}
```
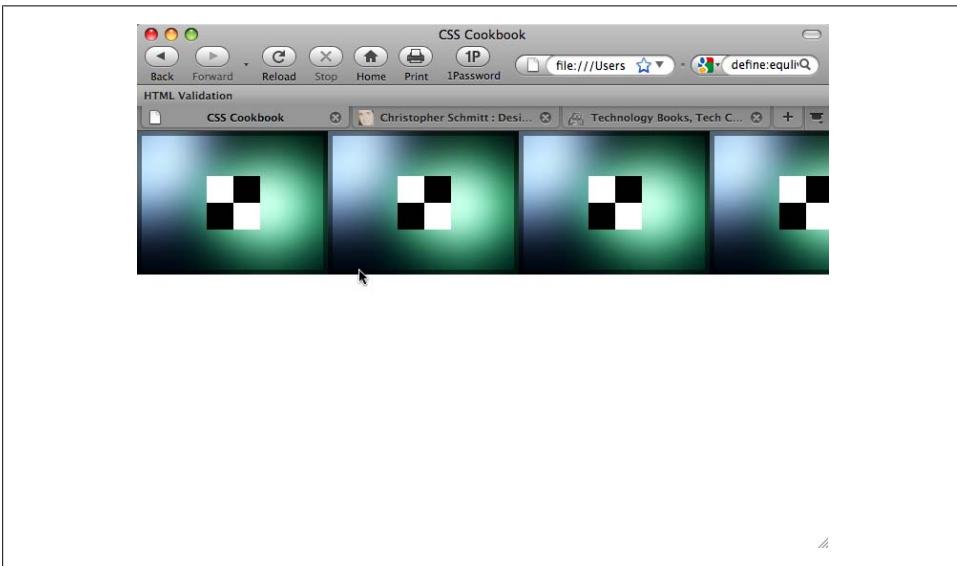

Figure 4-5. The background image tiled horizontally

Discussion

To have the background image repeat along the vertical axis, use the `repeat-y` value for the `background-repeat` property.

See Also

Recipe 4.7 for placing a background image at a specific location in a web page

4.7 Positioning a Background Image

Problem

You want to position a background image in a web page.

Solution

Use the `background-position` property to set the location of the background image. To place an image that starts 75 pixels to the right and 150 pixels below the upper-left corner of the viewport (see Figure 4-6), use the following CSS rule:

```
html {
  height: 100%;
}
body {
  background-image: url(bkgd.jpg);
  background-repeat: no-repeat;
  background-position: 75px 150px;
}
```

Discussion

The `background-position` property contains two values separated by a space. The first value of the pair sets the origin point along the y-axis, and the second value sets the point on the x-axis. If only one value is given, that value is used for the horizontal position and the vertical position is set to 50%.

The Solution used pixel units to determine the placement of the background image; however, you also can use percentages. A value of 50% for `background-position` means the browser places the image in the dead center of the viewport, as shown in Figure 4-7; the values 0% and 100% place the image in the upper-left and lower-right corners, respectively.
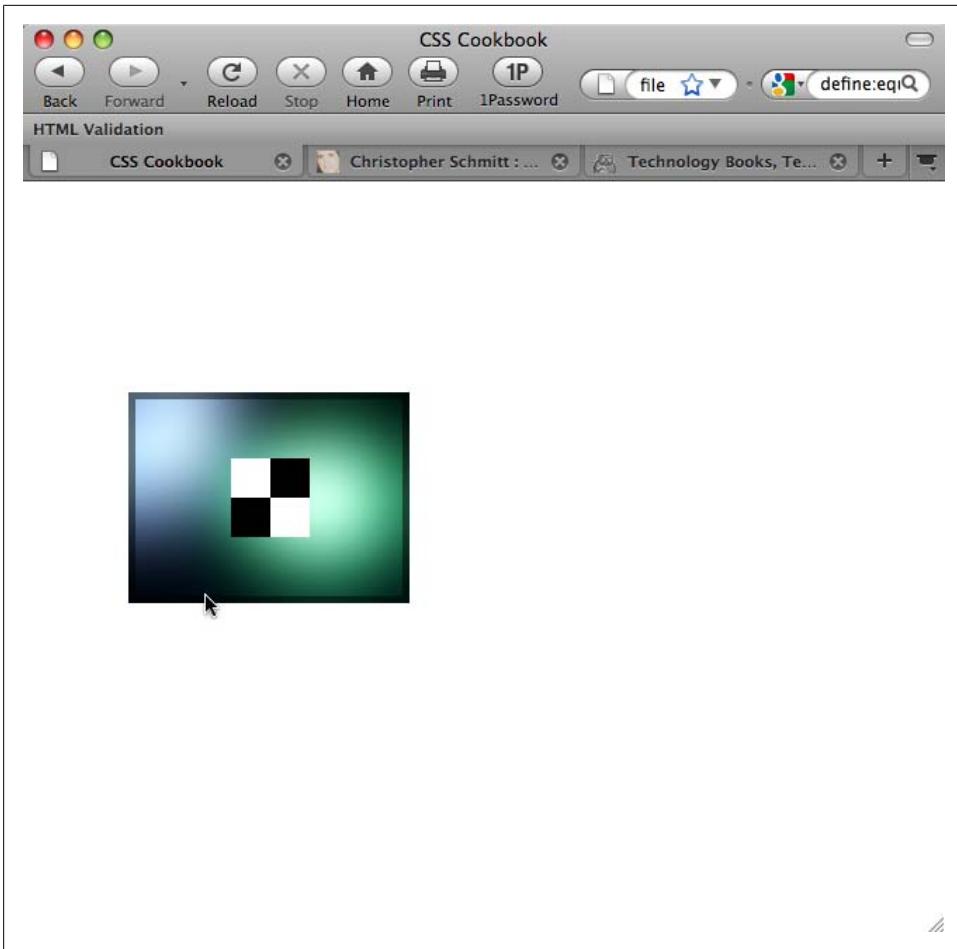
Figure 4-6. The background, placed precisely 75 pixels from the right and 150 pixels from the upper-left corner of the browser's viewport

Along with percentages, you can use the values `top`, `center`, and `bottom` for the y-axis and `left`, `center`, and `right` for the x-axis. Using combinations of these values, you can place the background image at eight points around the edges of the viewport (in the corners and in between), as well as in the middle of the viewport. For example, to re-create the value of `50%` in Figure 4-7, you can use this CSS rule instead:

```
body {
 background-image: url(bkgd.jpg);
 background-repeat: no-repeat;
 background-position: center;
}
```
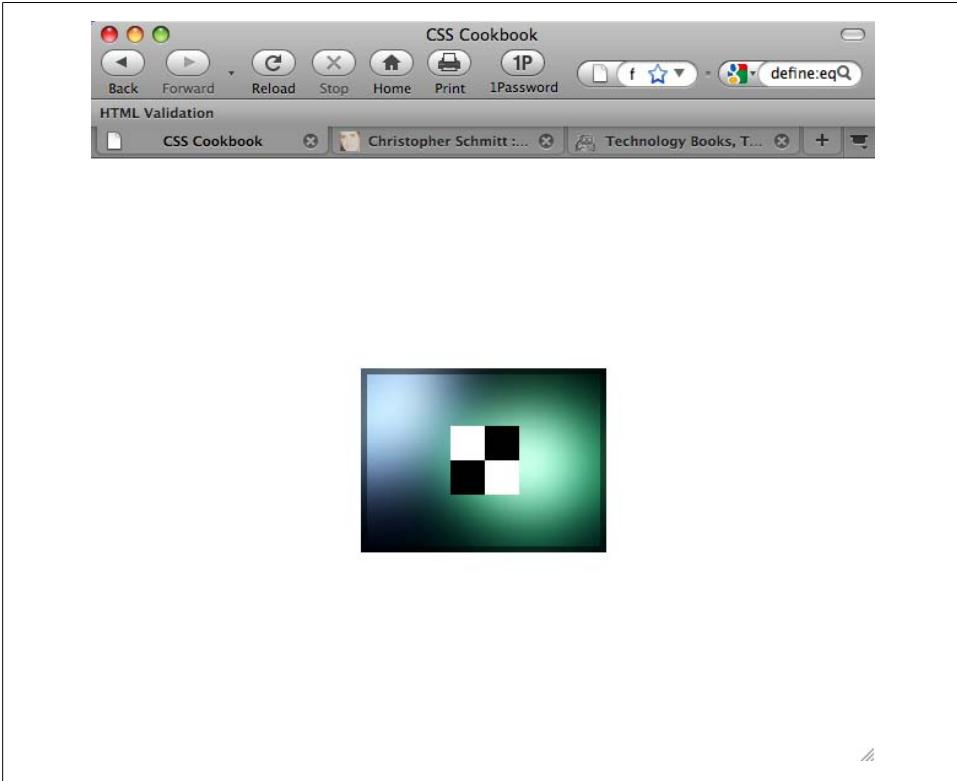
Figure 4-7. The background image centered in the browser window

To place a background image in the lower-right corner, as shown in Figure 4-8, you can use the following CSS rule:

```
body {
 background-image: url(bkgd.jpg);
 background-repeat: no-repeat;
 background-position: bottom right;
}
```

You also can use the `background-position` and `background-repeat` properties for background images that tile but aren't chained to the sides of the viewport.

See Also

Recipe 4.10 for setting an image so that it doesn't scroll; the CSS 2.1 specification for `background-position` at *http://www.w3.org/TR/CSS21/colors.html #propdef-background-position*
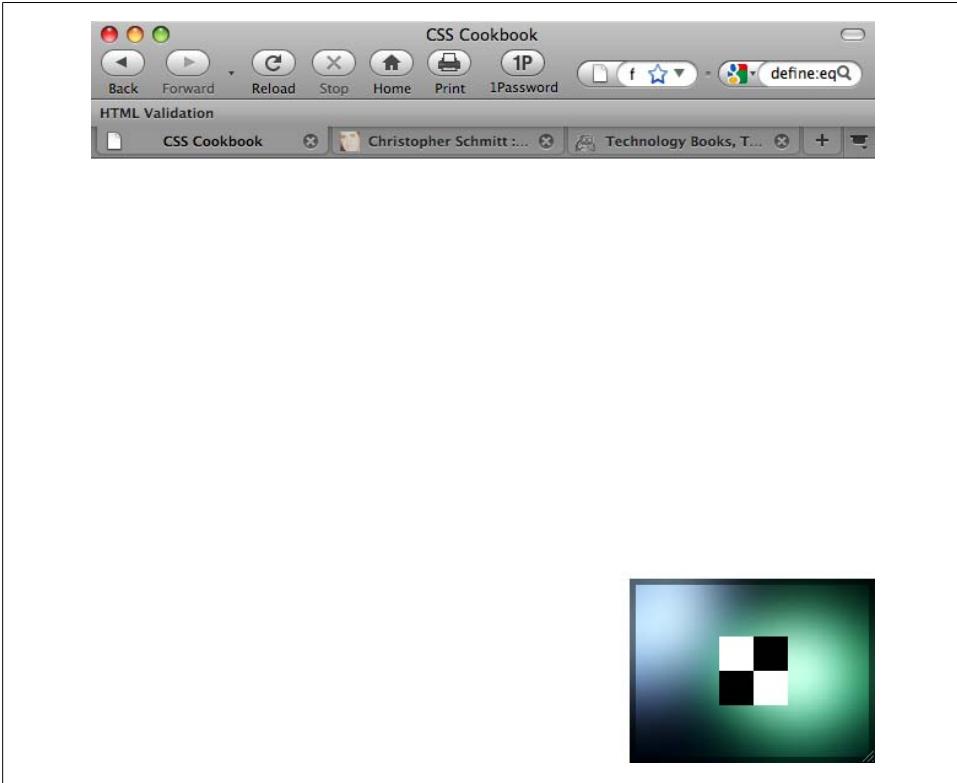
Figure 4-8. The background image placed in the lower-right corner

4.8 Using Multiple Background Images on One HTML Element

Problem

You want to place more than one background image within one HTML element.

Solution

In CSS3, the shorthand `background` property can accept multiple sets of background image information as long as commas separate them, as shown in Figure 4-9:

```
h2 {
 border: 1px solid #666;
 border-radius: 20px;
 -moz-border-radius: 20px;
 -webkit-border-radius: 20px;
 background: white;
 padding-top: 72px;
 text-align: center;
 background: url(mail.gif) top center no-repeat,
```

```
    url(printer.gif) 40% 24px no-repeat,
    url(gift.gif) 60% 24px no-repeat,
    url(content-bkgd.png) 50% 50% repeat-x,
    url(heading-sub-bkgd.png) 3em 3em repeat-x,
    url(plane.gif) center no-repeat;
  font-family: "Gill Sans", Trebuchet, Calibri, sans-serif;
  color: #666;
}
```
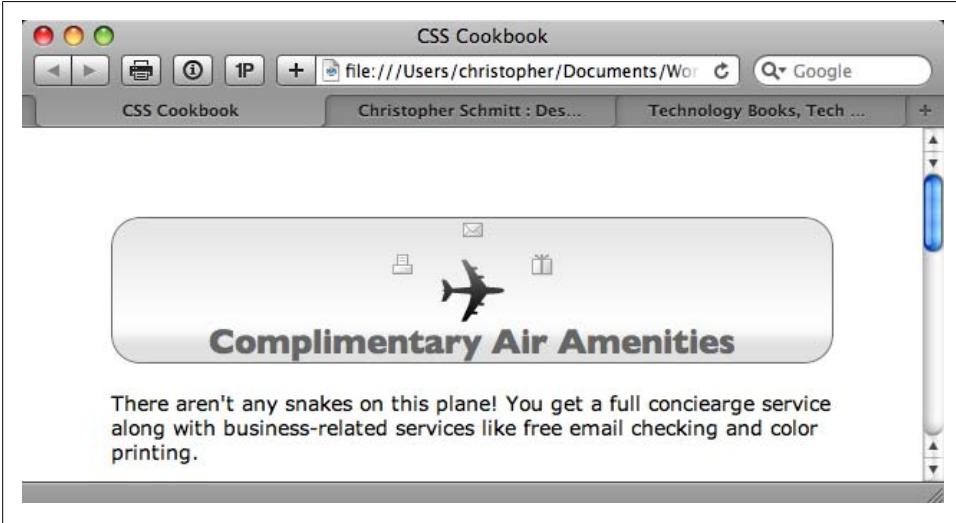

Figure 4-9. Individual icons placed as background images in the heading

> For a discussion of the technique to position images in the background
> of HTML elements, see Recipe 4.7.

Discussion

As of this writing, Safari for Macintosh has implemented the CSS3 specification for
layering multiple background images in one element.

Shorthand properties

Like most shorthand properties, you can split the shorthand code for multiple back-
grounds into separate CSS declaration blocks:

```
h2 {
  border: 1px solid #666;
  border-radius: 20px;
  -moz-border-radius: 20px;
  -webkit-border-radius: 20px;
```

```
      background: white;
      padding-top: 72px;
      text-align: center;
      background-image: url(mail.gif),
       url(printer.gif),
       url(gift.gif),
       url(content-bkgd.png),
       url(heading-sub-bkgd.png),
       url(plane.gif);
      background-position: top center,
       40% 24px,
       60% 24px,
       50% 50%,
       3em 3em,
       center;
      background-repeat: no-repeat,
       no-repeat,
       no-repeat,
       repeat-x,
       repeat-x,
       no-repeat;
      font-family: "Gill Sans", Trebuchet, Calibri, sans-serif;
    }
```

If all the backgrounds in the CSS rule are the same value, you can place one `no-repeat` value in the code and apply it to all the background images:

```
    h2 {
     padding-top: 72px; /* enough padding for the images */
     text-align: center;
     background: url(plane.gif), url(mail.gif), url(printer.gif), url(gift.gif);
     background-position: center, top center, 40% 24px, 60% 24px;
     background-repeat: no-repeat;
    }
```

You can apply this reduction of similar values to all CSS background-related properties, making sure that you want the background images to share the same value.

Not ready for everyday use

For the time being, introducing new elements and applying background images to these new elements is the only way to achieve the technique of multiple images across all modern browsers. For more information and examples of these techniques, see Recipes 4.14 and 4.15.

See Also

The CSS3 specification for layering multiple images at *http://www.w3.org/TR/2005/ WD-css3-background-20050216/#layering*

4.9 Setting Images on a Border

Problem

You want to place an image on a border of an HTML element.

Solution

First, create an image that contains the frame of the image. Wrap the content with
div elements and a unique id attribute value:

```
<div id="section">
 <h2>Images on Borders</h2>
 <p>Epsum factorial non deposit quid pro quo hic escorol. Olypian quarrels et
 gorilla congolium sic ad nauseum. Souvlaki ignitus carborundum
 e pluribus unum..</p>
</div><!-- /#section -->
```

Then use the CSS3 border-image property to place the image along the border width
of the element, as shown in :

```
#section {
 margin-right: 40px;
 color: #000;
 font-family: Verdana, Geneva, Tahoma, sans-serif;
 width: 100px;
 text-align: center;
 border-style: solid;
 border-color: #930;
 border-width: 26px 39px 37px 43px;
 border-image: url(frame.png) 26 39 37 43 stretch stretch;
 -webkit-border-image: url(frame.png) 26 39 37 43 stretch stretch;
-moz-border-image: url(frame.png) 26 39 37 43 stretch round;
 }
```

Discussion

The border-image property is a new CSS3 property that Firefox 3.1 and later and Safari
4 and later support as of this writing.

When the text is resized with the Solution, the border image scales and
contains the text.
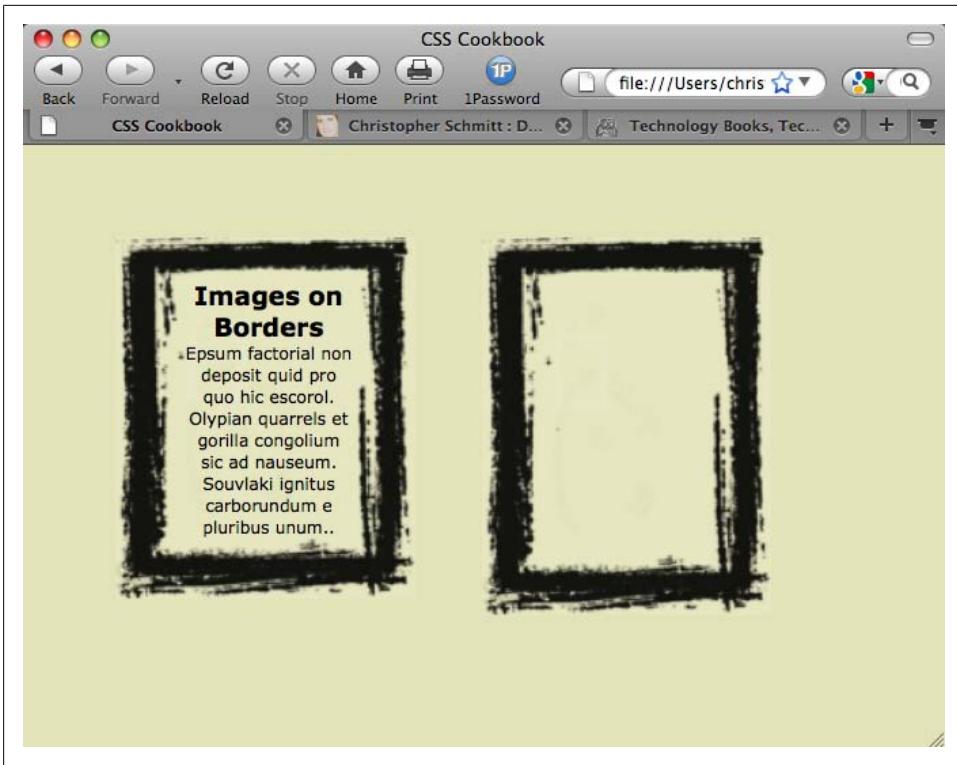
Figure 4-10. One image used to frame content

Not only does the `border-image` property allow you to frame content with one image that can scale, but it also provides a way to create image-rich buttons for web forms, as shown in Figure 4-11.

For example, first use HTML to create a simple button:

```
<form action="/" method="get">
 <button>Submit</button>
</form>
```

Then use the `border-image` property to set a visually interesting button, as shown in Figure 4-11, that is better than the default rendering:

```
button {
 background: none;
 width: 250px;
 padding: 10px 0 10px 0;
 border-style: solid;
 border-color: #666;
 border-width: 0 17px 0 17px;
 border-image: url(bkgd-button.png) 0 17 0 17 stretch stretch;
 -webkit-border-image: url(bkgd-button.png) 0 17 0 17 stretch stretch;
 -moz-border-image: url(bkgd-button.png) 0 17 0 15 stretch stretch;
```

```
  color: white;
  font-family: "Gill Sans", Trebuchet, Calibri, sans-serif;
  font-weight: bold;
  text-transform: uppercase;
  text-shadow: 0px 0px 5px rgba(0,0,0,.8);
}
```
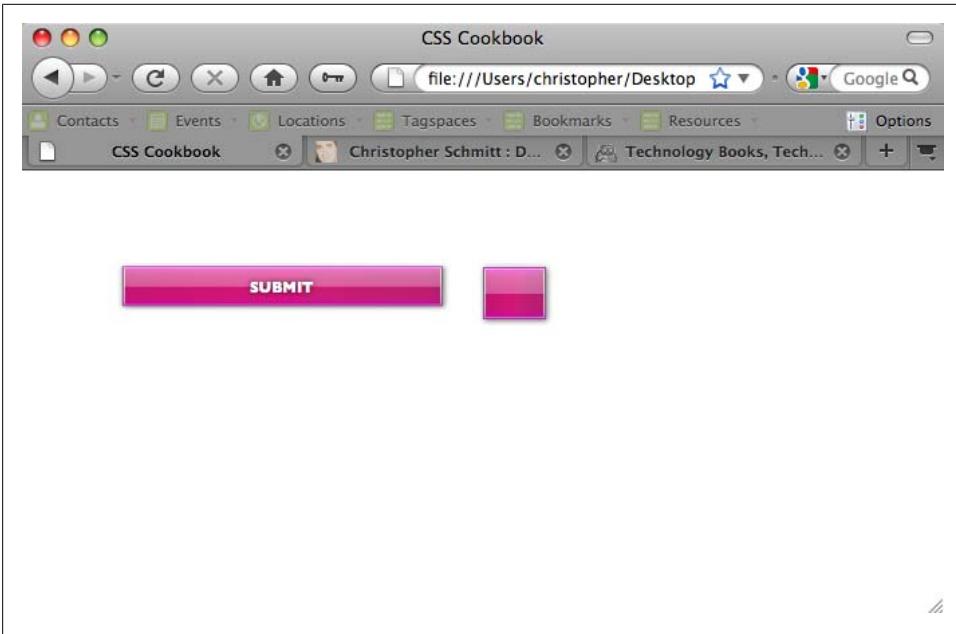

Figure 4-11. One image used to create a button effect

When setting an image on a border, first set the widths of the border:

```
border-width: 0 17px 0 17px;
```

Then bring in the image through the `url()` function with the `background-image` property:

```
border-width: 0 17px 0 17px;
border-image: url(bkgd-button.png);
```

The next four values should match the values of the `border-width` property for the top, right, bottom, and left sides of the HTML element:

```
border-width: 0 17px 0 17px;
border-image: url(bkgd-button.png);
border-image: url(bkgd-button.png) 0 17 0 17;
```

The values of `0` for `border-image` instruct the browser to cover the entire top and bottom borders with the border image. The values of `17` indicate that 17 pixels of the border image on the right and left sides should be used.

Set the next two values to `stretch` so that the background image expands across the distance of the sides to create a seamless fit:

```
border-width: 0 17px 0 17px;
border-image: url(bkgd-button.png);
border-image: url(bkgd-button.png) 0 17 0 17 stretch stretch;
```

Other values besides `stretch` are `repeat` (which tiles the image) and `round` (which also tiles, but makes the tiling of the image fit nicely between the edges).

See Also

The CSS3 specification for `border-image` at *http://www.w3.org/TR/css3-background/#the-border-image*

4.10 Creating a Stationary Background Image

Problem

You want a background image to remain in the browser window, even as the user scrolls down a web page.

Solution

Use the `background-attachment` property set with a `fixed` value, like so:

```
body {
  background-image: url(bkgd.jpg);
  background-repeat: no-repeat;
  background-attachment: fixed;
}
```

Discussion

By using this technique, you are locking down the background image. Therefore, even if a visitor scrolls, the image remains where you placed it originally. Another acceptable value for `background-attachment` is `scroll`, which is the default value. So, even if you don't specify `scroll`, the background image moves up with the rest of the document as the visitor scrolls down.

For example, imagine you want to post on your web page a photo of a recent trip, and you want the photo positioned on the left side of the page and your text on the right. As the reader scrolls down to read more about the trip, the photo from the trip stays in place, as shown in Figure 4-12.
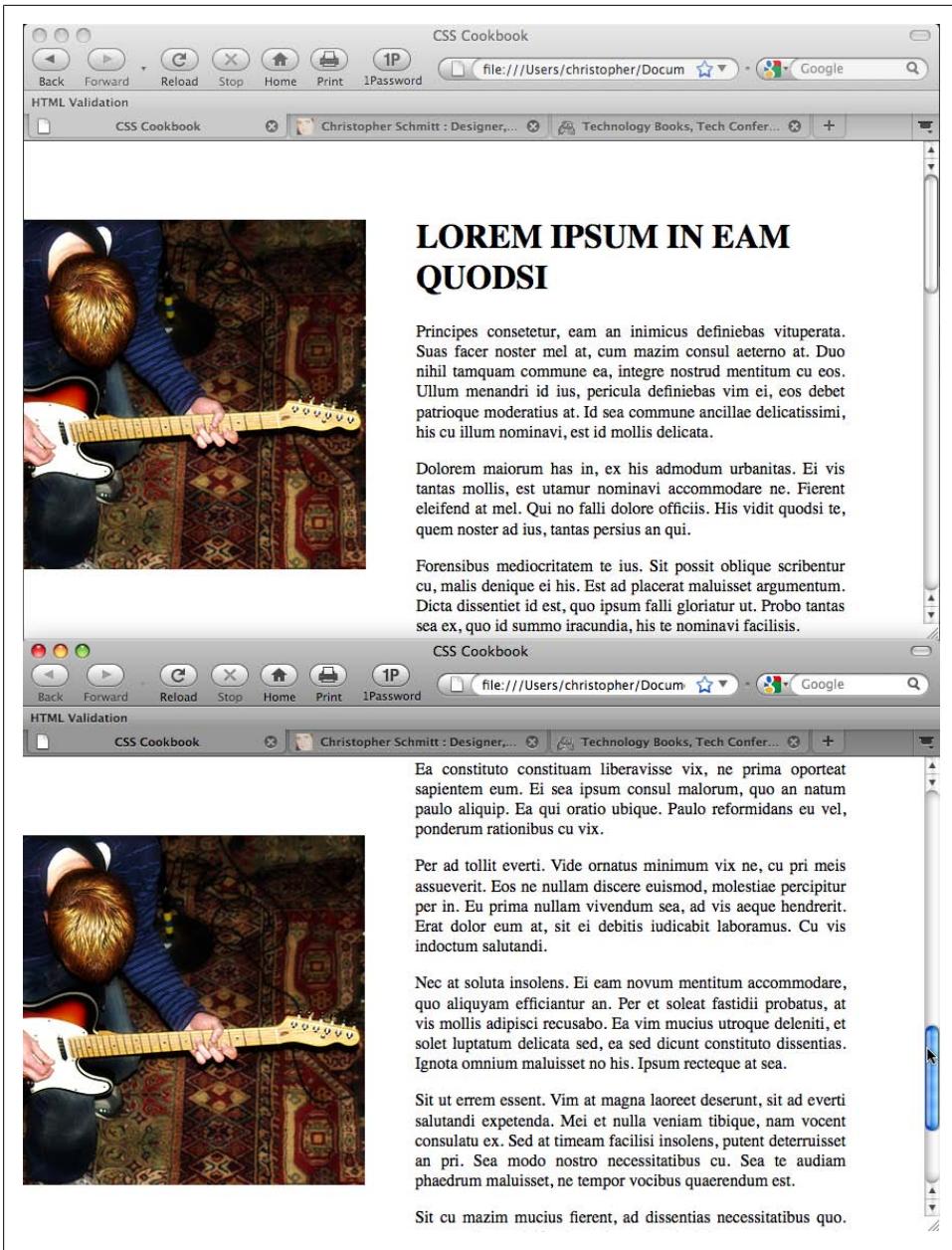
Figure 4-12. The photo staying in place as the visitor scrolls

Here's the code:

```
body {
 background-image: url(bkgd2.jpg);
 background-repeat: no-repeat;
 background-attachment: fixed;
 background-position: -125px 75px;
 margin: 75px 75px 0 375px;
}
h1, h2, h3 {
 padding-top: 0;
 margin-top: 0;
 text-transform: uppercase;
}
p {
 text-align: justify;
}
```

To take this further, you can lock down the image on block-level elements other than body. For example, try the heading elements when designing a review for a movie or concert. The following CSS rule can create an interesting surfing experience:

```
h1, h2, h3 {
 font-size: 200%;
 background-image: url(bkgd2.jpg);
 background-repeat: no-repeat;
 background-attachment: fixed;
 background-position: center;
 padding: 1.5em;
 text-align: center;
 color: white;
}
```

Because of the padding and light color on the headings, there is enough room to see the background image "through" the elements as well as to read the headlines. As the visitor scrolls the web page reading the review, she will see the rest of the image, as shown in Figure 4-13.

See Also

Recipe 4.5 to position a background image; the CSS 2.1 specification for background-attachment at *http://www.w3.org/TR/CSS21/colors.html#propdef-background-attachment*

4.11 Stretching Images As the Browser Resizes

Problem

You want the background image to stretch as the browser resizes.
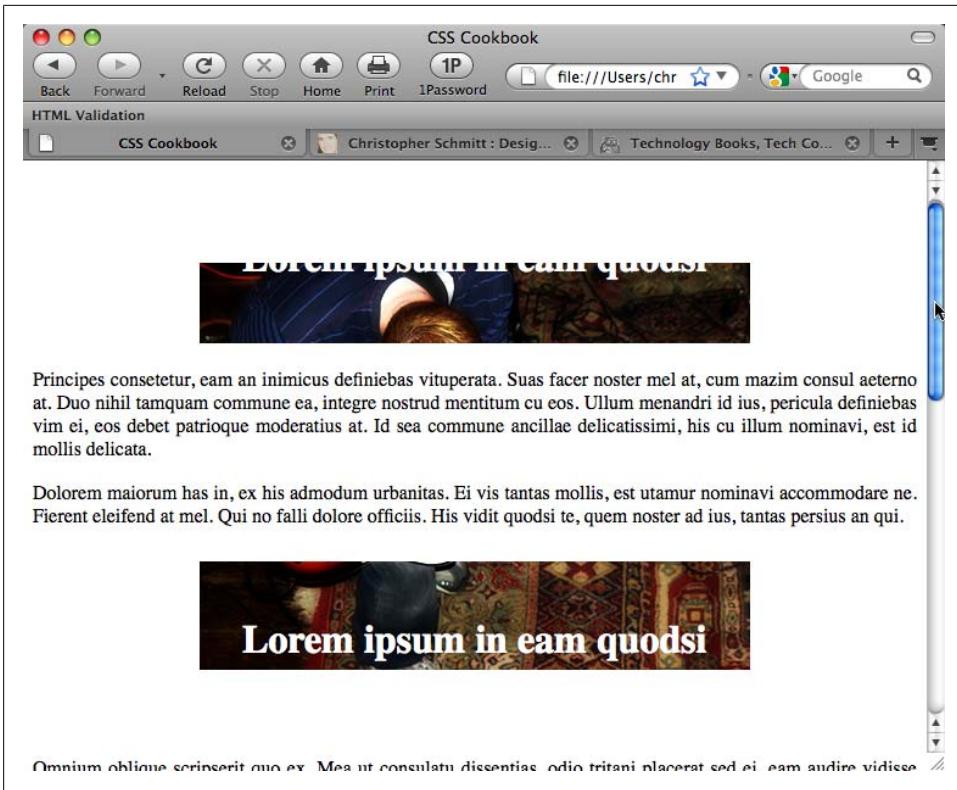
Figure 4-13. The photo coming through the headings instead of the body element

Solution

Use the `background-size` property along with related browser-vendor-specific properties, as shown in Figure 4-14:

```
body {
  background-image: url(button_redstar.gif);
  background-size: 25% auto;
  -o-background-size: 25% auto;
  -webkit-background-size: 25% auto;
  -khtml-background-size: 25% auto;
  background-repeat: repeat-x;
  margin-top: 30%;
}
```

Discussion

When setting the `background-size` property, the browser stretches the image according to its values. The first value sets the width and the second value sets the height.
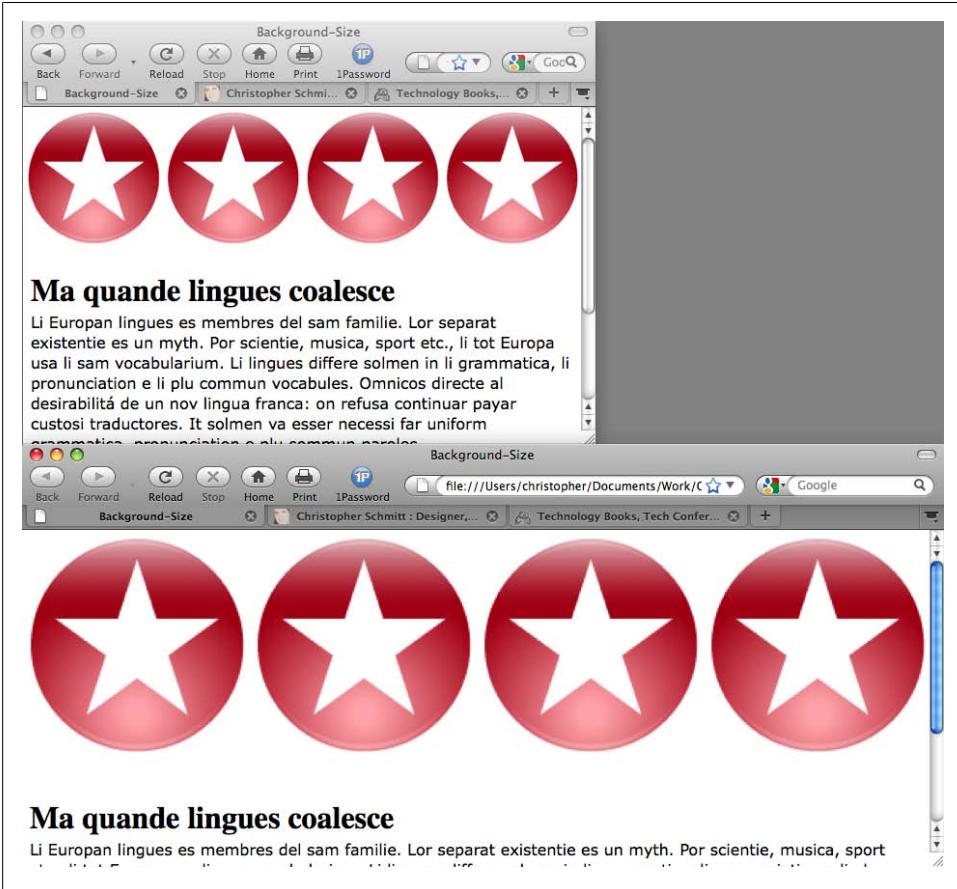
Figure 4-14. Four images placed equally at the top of the viewport, even when resized

In the Solution, setting a value of `25%` instructs the browser to tile out the background image four times along the width of the browser's viewport. Since the `background-repeat` property is set to repeat along the x-axis, only four images are tiling out in the background.

The value of `auto` for `height` means the aspect ratio of the image is preserved.

Firefox 3.6 supports the `background-size` property.

See Also

The CSS3 specification on `background-size` at *http://www.w3.org/TR/css3-background/#the-background-size*

4.12 Stretching an Image Across the Entire Browser Window

Problem

You want to expand an image across the entire browser window.

Solution

For a cross-browser solution, use HTML frames.

First create a *full-bleed.html* file and place an image in the `body` element:

```
<img id="stretch" src="green_car.jpg"  alt="photo of green car" />
```

Use CSS to remove the margins and padding in the body as well as expand the width and height of the image:

```
body {
 margin: 0;
 padding: 0;
}
#stretch {
 position: absolute;
 width: 100%;
 height: 100%;
}
```

Discussion

This Solution works best in most browsers tested by leveraging the ability to remove the image from the normal flow of the document and then resetting the width and height to 100%.

To overlay content on top of the image, use the absolute `position` property and set the new content to a high `z-index` value (or at least higher than the stretched image).

Using background-size

The most ideal and direct method of creating a full-bleed image effect is to use the `background-size` property (see Recipe 4.11) to stretch a background image to the entire width and height of a browser viewport in Safari:

```
body {
 background-image: url(green_car.jpg);
 -webkit-background-size: 100% 100%;
 -o-background-size: 100% 100%;
}
```

However, if the browser is made smaller, the image starts to shrink to maintain its aspect ratio and tiles out copies of the image underneath it.

Using an iframe

Another method is to use an `iframe` HTML element to somehow replicate the HTML frameset:

```
<iframe width="100%" height="100%" src="full-bleed.html" border="0"
noborder="noborder" frameborder="0" padding="0" spacing="0"
scrolling="no"></iframe>
```

However, the stretching of the image within the *full-bleed.html* file does not extend all the way down in some browsers, such as Opera and Safari.

> Another step is to use HTML framesets. However, for accessibility concerns it's best to avoid those if at all possible.

See Also

Recipe 2.23 for more information on positioning elements absolutely

4.13 Making Images Scalable

Problem

You want images to resize as the browser window resizes.

Solution

Define the width of images to percentages, as shown in Figure 4-15:

```
img {
 border: 1px solid #cecece;
 width: 60%;
 float: left;
 margin-right: .7em;
 margin-bottom: .5em;
}
```

Modern browsers will scale the height of the images in relative proportion to the width. So, defining both the width and the height is not necessary.
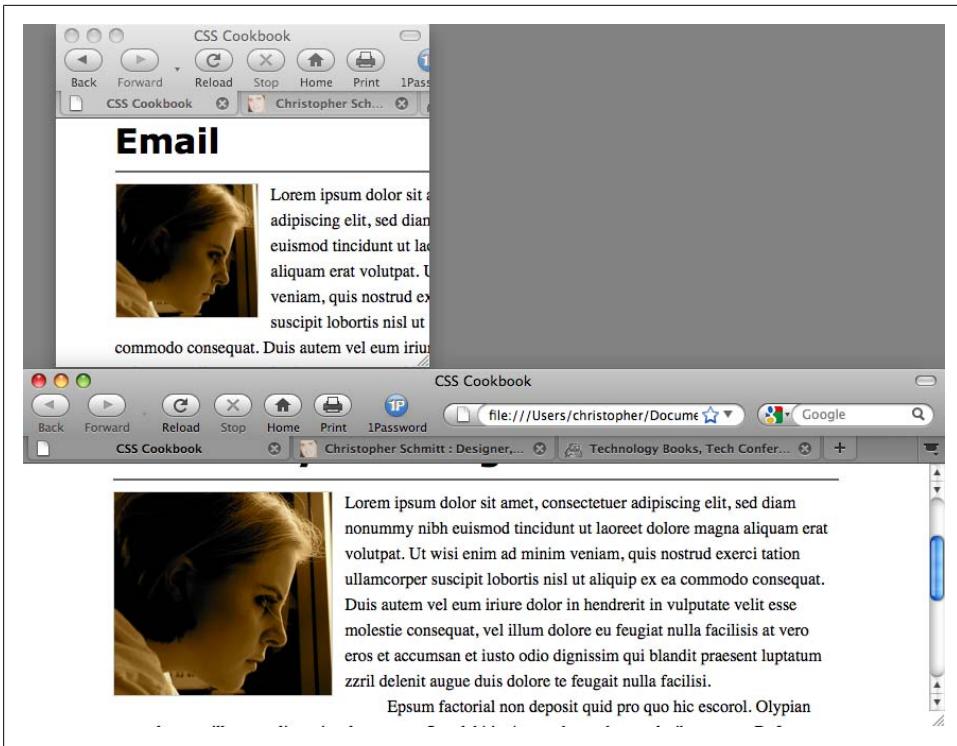
Figure 4-15. The image scaled down, and the image at a larger size since the browser window is larger

Discussion

When building fluid or flexible layouts, you set the HTML text in columns to expand and retract as the browser resizes. However, images with dimensions that are commonly set in pixels retain their size.

To make sure all the page elements are resized in proportion to each other in flexible layouts, developers may set the width and height to percentages.

Keeping images from expanding

When images are set to percentage-based dimensions, browsers might stretch the images beyond the point where the images retain their integrity. For example, compression artifacts that are nearly invisible in a JPEG or GIF image become apparent when they are expanded.

To keep the images from expanding beyond a defined width, use the `max-width` property with length units:

```
img {
 border: 1px solid #cecece;
 width: 60%;
```

```
  max-width: 300px;
  float: left;
  margin-right: .7em;
  margin-bottom: .5em;
}
```

See Also

Chapter 9 for more on flexible layouts

4.14 Setting How a Browser Renders an Image

Problem

You want to set the browser to render images as shown in Figure 4-16.
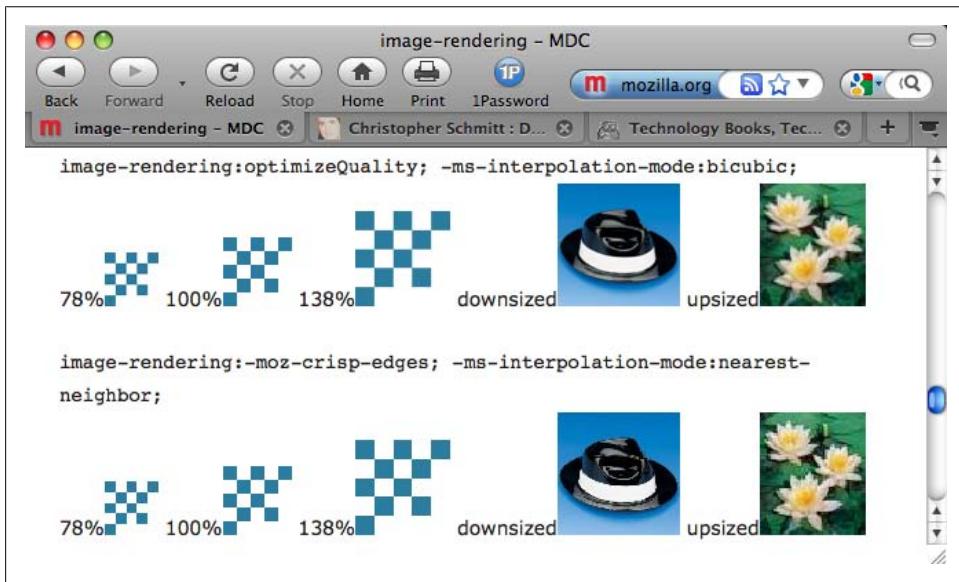
Figure 4-16. Mozilla Development Center demonstrations of image rendering preferences

Solution

Use the `image-rendering` property along with the Microsoft proprietary property, `-ms-interpolation-mode`:

```
#content img[src$=".gif"] {
 image-rendering: -moz-crisp-edges;
 -ms-interpolation-mode: nearest-neighbor;
}
```

Discussion

The `image-rendering` property was originally a Scalable Vector Graphics (SVG) property; however, it has been ported to work on HTML elements. The property instructs the browser on how to render images that are resized.

The CSS rule listed in the Solution displays images with hard edges or high contrast. Use the following CSS rules for smoother rendering:

```
#content img[src$=".gif"] {
 image-rendering: optimizeQuality;
 -ms-interpolation-mode: bicubic;
}
```

> For Firefox browsers, the `image-rendering` property applies to inline and background images as well as HTML5 `video` and `canvas` elements.

Smoothing images for Internet Explorer

Web designer Ethan Marcotte created another solution for resizing images for Internet Explorer that uses JavaScript to use Microsoft's `filter` property (see *http://unstoppablerobotninja.com/entry/fluid-images/*). An easy-to-use jQuery plug-in that extends the solution is available at *http://thinkdrastic.net/journal/wp-content/uploads/2009/05/imgresizer.htm*.

See Also

SVG attribute details for `image-rendering` at *http://www.zvon.org/xxl/svgReference/Output/attr_image-rendering.html*; the MSDN entry for `-ms-interpolation-mode` at *http://msdn.microsoft.com/en-us/library/ms530822(VS.85,loband).aspx*; the Mozilla Developer Center description for `image-rendering` at *https://developer.mozilla.org/en/CSS/image-rendering*

4.15 Rotating Images with CSS

Problem

You want to rotate images.

Solution

First set the `img` element to display as a block-level element:

```
img {
 display: block;
 float: left;
```

```
    margin: 20px;
}
```

Then use a set of proprietary CSS properties for Safari, Firefox, and Internet Explorer browsers to rotate the images 270 degrees, as shown in Figure 4-17:

```
img+img {
  -webkit-transform: rotate(270deg);
  -moz-transform: rotate(270deg);
  filter: progid:DXImageTransform.Microsoft.BasicImage(rotation=3);
}
```
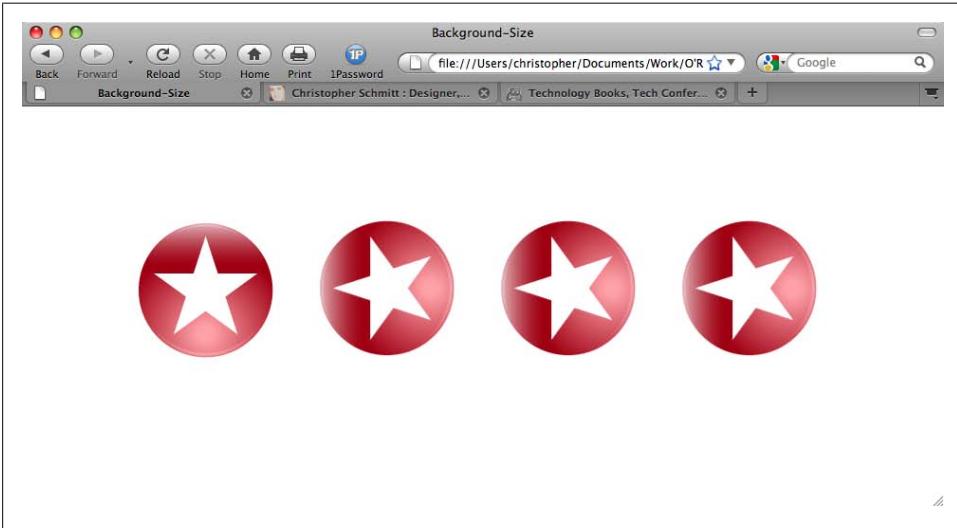

Figure 4-17. The first image rendered by default, but the rest of the images are rotated

Discussion

Web designers rotate block-level elements through the use of proprietary CSS properties, but only in 90-degree increments.

Although the Safari and Firefox proprietary `transform` properties allow for a fine degree of rotating of elements (e.g., 78 degrees), Microsoft's `BasicImage` filter property can rotate in only four stops, as shown in Table 4-2.

Table 4-2. Simple conversion table for cross-browser rotation

Degree rotation	BasicImage filter value
0	0
90	1
180	2
270	3

The Safari and Firefox `transform` properties also allow the benefit of skewing the rendering of block-level elements. However, at the time of this writing, the effect is not available in Internet Explorer 8. For more information on this effect, refer to the "See Also" section of the previous recipe.

See Also

The MSDN article on the `rotation` filter at *http://msdn.microsoft.com/en-us/library/ms532918(VS.85,loband).aspx*; the Mozilla Developer Center article on `-moz-transform` at *https://developer.mozilla.org/en/CSS/-moz-transform*; the Surfin' Safari blog post about `-webkit-transform` at *http://webkit.org/blog/130/css-transforms/*

4.16 Setting Gradients with CSS

Problem

You want to create background gradients with CSS.

Solution

Use proprietary CSS properties to set gradients in the background of elements, as shown in Figure 4-18:

```
div.building {
 border: 1px solid #666;
 float: left;
 width: 300px;
 height: 300px;
 margin: 20px;
 background-image: -webkit-gradient(radial,center center,900,center
 bottom,0,from(#0cf),to(white));
 background-image: -moz-radial-gradient(center,900px,center
 bottom,0,from(#0cf),to(white));
 background-repeat: no-repeat;
}
```

Discussion

As of this writing, CSS-enabled gradients are in Safari 4 and later and Firefox 3.6 and later through the use of their respective vendor-based properties.

Setting gradients in Safari

In Safari, cite `-webkit-gradient();` after the `background` or `background-image` property:

```
background-image: -webkit-gradient();
```
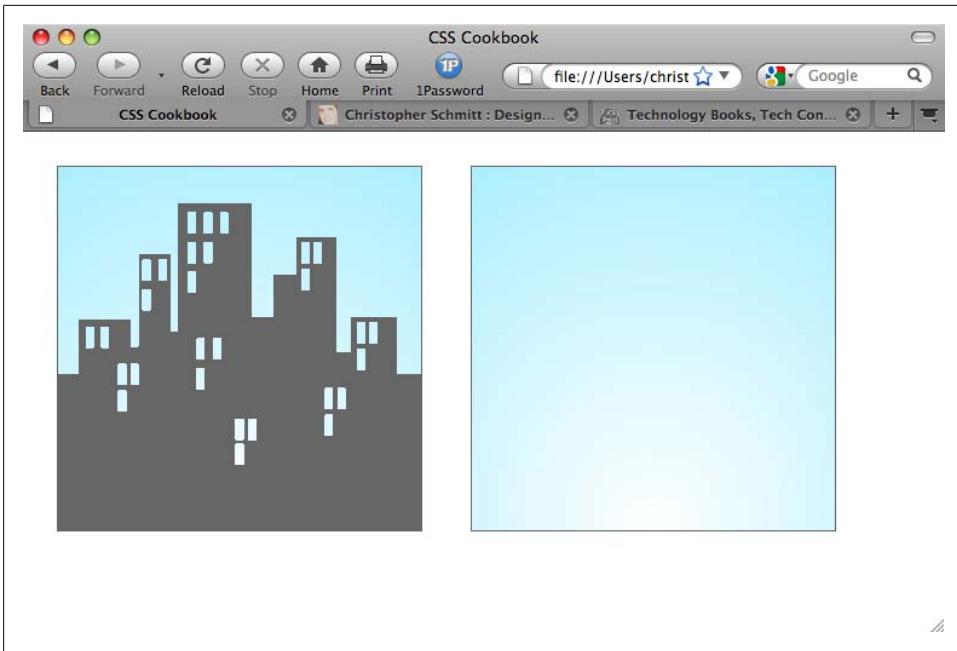
Figure 4-18. Radial gradients set in the background of an element

> When developing a web page design with a CSS-enabled gradient, I recommend using `background-image` over the `background` shorthand property so as not to lock out other background-related properties, such as those that set the `background-color` value of the element.

Next, state which kind of gradient you want to set, *radial* or *linear*:

```
background-image: -webkit-gradient(linear);
```

Then use `background-position` to set the starting point of the gradient along the value of the radius:

```
background-image: -webkit-gradient(radial,center center,900);
```

> Typically, a shorthand value for `background-position` in this instance would be only an instance of the `center` value. However, the Safari browser does not understand that within the confines of the CSS `gradient` property. So, do not use shorthand properties for positioning gradients.

Think of the radius as a stopping point in the radial gradient. When a browser renders a gradient and gets to that specific point cited in the radius, the color gradation stops and the color remains solid.

As of this writing, the radius does not accept unit values, and any numerical value is accepted as meaning pixel units.

After the starting point has been defined, set the ending point:

```
background-image: -webkit-gradient(radial,center center,900,center bottom,0);
```

With the starting and stopping points in place, set the corresponding colors:

```
background-image: -webkit-gradient(radial,center
center,900,center bottom,0,from(#0cf),to(white));
```

You can apply CSS gradients within Safari not only to background images of block-level elements, but also to list bullets, generated content, and border images.

Setting gradients in Firefox

Whereas Safari sets the type of gradient within its own proprietary property, Firefox has properties for both types of gradients: `-moz-radial-gradient()` and `-moz-radial-linear()`.

Unlike the Safari gradient, you can use `background-position` shorthand values and unit values when setting the starting and stopping points of gradients.

Transparency with gradients

Another interesting aspect of Firefox's implementation of CSS gradients concerns transparency. If you don't set a background color on an element, you can set the background color (or colors) for a gradient to be transparent by specifying the color with RGBA:

```
background-image: -moz-linear-gradient(left top, left bottom,
from(rgba(153,51,0,.3)), to(#6b3703), color-stop(0.5, #903000));
```

Since this Solution uses CSS properties that are available in only some of the more recent browsers, a workaround is to create gradients in a digital imaging program and set them through the background of images (see Recipe 4.5).

See Also

The Surfin' Safari blog post "Introducing CSS Gradients" at *http://webkit.org/blog/175/introducing-css-gradients/*; the Mozilla Developer Center article on Firefox gradient properties at *https://developer.mozilla.org/en/CSS/-moz-linear-gradient*

4.17 Creating Transparent PNG Images for IE6 and Later

Problem

You want to create backward-compatible PNG images with transparency for Internet Explorer 6 without JavaScript.

Solution

Use the Adobe Fireworks digital imaging application to make blended PNG8 images.

For an image with transparency, such as a drop shadow, set the image type to PNG8, dither to 100%, and transparency to alpha transparency, as shown in Figure 4-19.
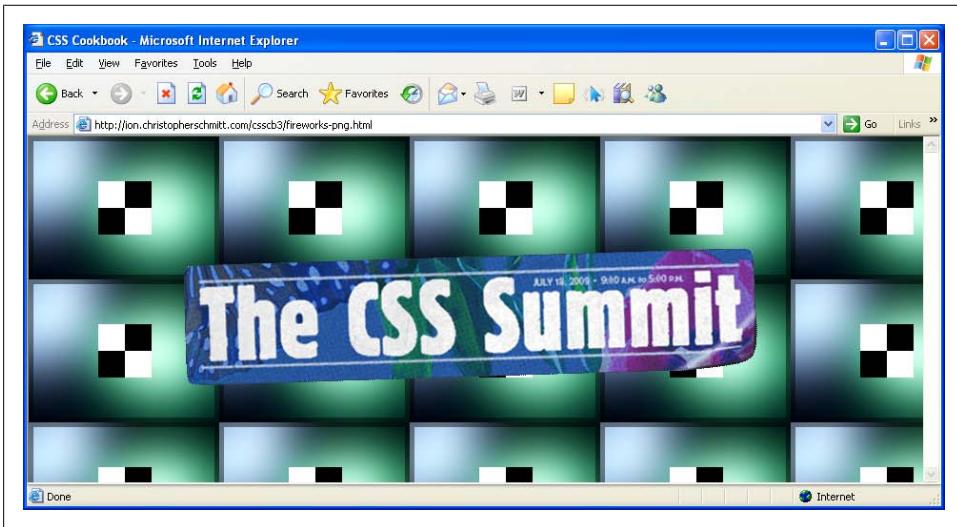
Figure 4-19. Retaining image integrity, even though the alpha transparency of the gradient is clipped

Discussion

The PNG24 file format's alpha transparency doesn't work in IE6, often showing transparency as a white block, as seen in Figure 4-20. Adobe Fireworks exports PNG8 with its semitransparency intact and visible in Internet Explorer 7 and later.

Currently, Adobe Fireworks is the only commercial application that provides this type of blended PNG8 export functionality.

> Other applications that perform this type of special export include pngquant (see *http://www.libpng.org/pub/png/apps/pngquant.html*) and pngnq (see *http://pngnq.sourceforge.net/index.html*).
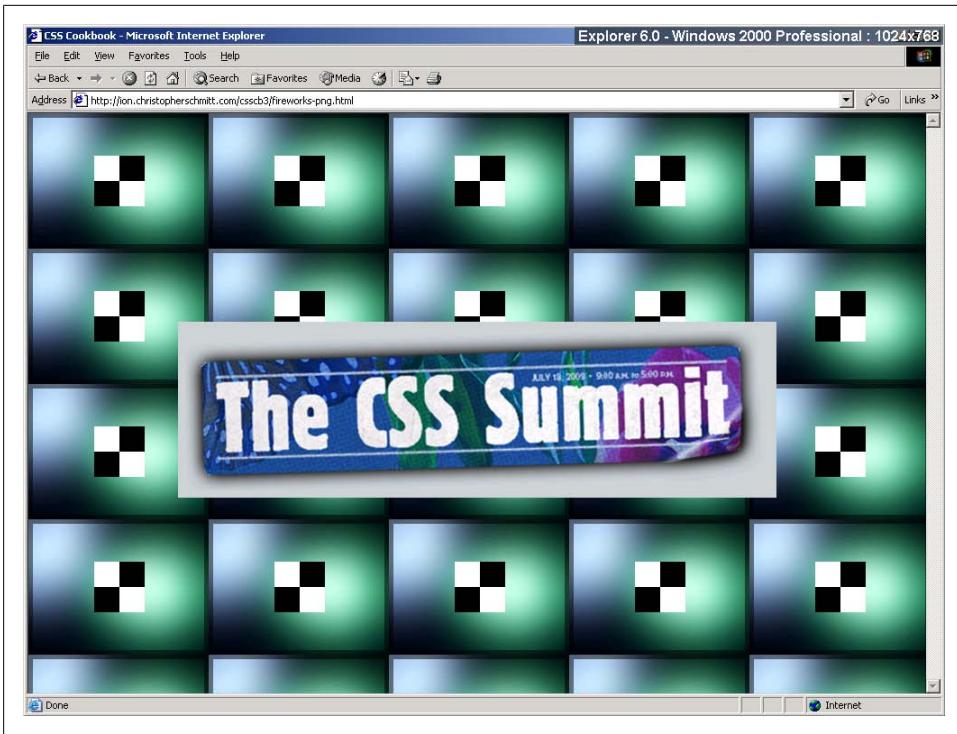
Figure 4-20. The alpha transparency portion of the gradient missing

See Also

The SitePoint article "PNG8—The Clear Winner" at *http://www.sitepoint.com/blogs/2007/09/18/png8-the-clear-winner/*

4.18 Using Transparent PNG Images with JavaScript

Problem

You want to use multiple PNGs with alpha transparency.

Solution

Use Drew McLellan's updated Sleight script for triggering alpha transparency in Internet Explorer versions 5.5 through 6.

Either write the code in a separate JavaScript file or download the code from McLellan's website at *http://allinthehead.com/code/samples/bgsleight.js*:

```
if (navigator.platform == "Win32" &&
 navigator.appName == "Microsoft Internet Explorer" &&
```

```
  window.attachEvent) {
    window.attachEvent("onload", fnLoadPngs);
  }

  function fnLoadPngs() {
   var rslt = navigator.appVersion.match(/MSIE (\d+\.\d+)/, '');
   var itsAllGood = (rslt != null && Number(rslt[1]) >= 5.5);
   for (var i = document.all.length - 1, obj = null;
    (obj = document.all[i]); i--) {
      if (itsAllGood &&
   obj.currentStyle.backgroundImage.match(/\.png/i) != null) {
    this.fnFixPng(obj);
    obj.attachEvent("onpropertychange",
   this.fnPropertyChanged);
    }
   }
  }

  function fnPropertyChanged() {
   if (window.event.propertyName == "style.backgroundImage") {
    var el = window.event.srcElement;
    if (!el.currentStyle.backgroundImage.match(/x\.gif/i)) {
      var bg = el.currentStyle.backgroundImage;
      var src = bg.substring(5,bg.length-2);
      el.filters.item(0).src = src;
      el.style.backgroundImage = "url(x.gif)";
    }
   }
  }

  function fnFixPng(obj) {
   var bg = obj.currentStyle.backgroundImage;
   var src = bg.substring(5,bg.length-2);
   obj.style.filter =
   "progid:DXImageTransform.Microsoft.AlphaImageLoader(src='"
   + src + "', sizingMethod='scale')";
   obj.style.backgroundImage = "url(x.gif)";
  }
```

Attach the JavaScript file to the web page by placing the following code in between the head elements:

```
<script src="/_assets/js/bgsleight.js" type="text/javascript"></script>
```

Be sure to upload the single-pixel transparent GIF (listed as x.gif in the script) to the web server and update the file location reference in the script for your needs.

Discussion

Support for alpha transparency in modern browsers is almost commonplace. Browsers that include native support for PNGs include Chrome, Opera, Safari, and Internet Explorer for Windows 7. However, this list does not include Internet Explorer for Windows 6.

To work around this, Aaron Boodman created a piece of JavaScript that uses Microsoft's proprietary `filter` property to activate Internet Explorer for Windows versions 5.5 through 6 support for inline PNGs with alpha transparency, without interfering with the other browsers that support PNGs natively.

Drew McLellan built off of Boodman's work and modified the JavaScript used in the Solution to make the script work not only for inline images, but also for background images (see *http://allinthehead.com/retro/69/sleight-of-hand*).

> If you use jQuery, a plug-in based on this solution is readily available at *http://jquery.andreaseberhard.de/pngFix/*.

How the script works

As a page is loaded, McLellan's JavaScript is executed. The script goes through the HTML markup looking for `img` elements that point to images with the `png` extension.

Once it finds such `img` code, the script dynamically rewrites the HTML on the fly. The first part of the revision is to replace the PNG image with a single-pixel GIF that is transparent.

Next, the PNG file is set in Internet Explorer's `filter` property to trigger the alpha transparency in that browser. Since this is the only way this can be done, the PNG gets set in the background.

> To deliver proprietary CSS properties to only Internet Explorer, use conditional comments (see Recipe 12.7).

Thus, the PNG is shown in the background behind the transparent GIF.

> PNG24 images, those with full alpha transparency, tend to have rather large file sizes. To help with that issue, an optimizer called Pngcrush (see *http://pmt.sourceforge.net/pngcrush/*) is available that you can execute from the MS-DOS window or Unix command line. PNGThing, a small, easy-to-use Mac application based on this code, is available at *http://mac.softpedia.com/get/Graphics/PNGThing.shtml*.

See Also

The original posting of the Sleight script at *http://www.youngpup.net/2001/sleight*; more information about Microsoft's `filter` property at *http://msdn.microsoft.com/en-us/library/ms532967(VS.85,classic).aspx*

4.19 Overlaying HTML Text on an Image

Problem

You want to position HTML text over an image.

Solution

Set the image within the background and then position and style the HTML text accordingly.

First, wrap the text around a `div` element with an `id` attribute:

```
<div id="frame">
 <div id="banner">
  <h1>White House Confidential <br /><span>
Classified Lawn Care Secrets</span></h1>
 </div><!-- end #banner -->
 <p>...</p>
</div>
```

Insert the image through the `background-image` property, making sure to set the width and height:

```
#banner {
  width: 550px;
  height: 561px;
  overflow: hidden;
  background-image: url(whitehouse.jpg);
  background-position: 0;
  background-repeat: no-repeat;
  position: relative
}
```

Then adjust the type to the desired style, as shown in Figure 4-21:

```
h1 {
  margin: 0;
  padding: 0;
  font-family: Verdana, Arial, sans-serif;
  margin-top: 325px;
  margin-left: 25px;
  position: absolute;
  bottom: 0;
  color: white;
  text-shadow: 0 1px 0 #666;
  text-align: center;
  border-left: 2px solid #666;
  border-right: 2px solid #666;
  border-top: 2px solid #666;

  /* room around text */
  padding-left: 25px;

  /* bring in the translucent background image */
```

```
    background-image: url(white-banner.png);
    background-position: bottom;
    background-repeat: no-repeat;
}
h1 span {
    font-size: .8em;
}
```
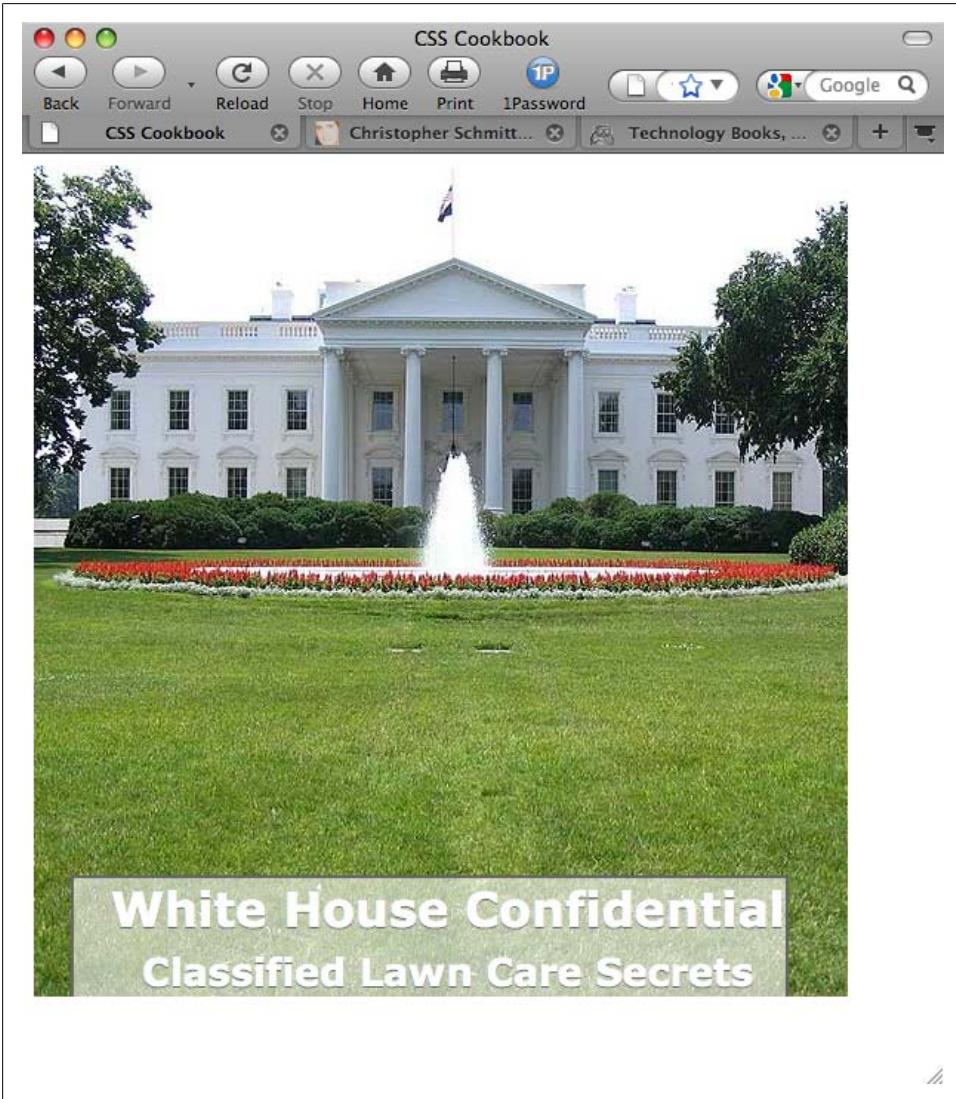

Figure 4-21. The photo coming through the headings instead of the body element

Discussion

Instead of bringing in an image and having it be inline or part of the content of a web page when its purpose is strictly decorative, use the `background-image` property to display the image. This method makes the page more accessible, but still maintains the intended visual.

See Also

Recipe 4.20 for replacing HTML text with an image

4.20 Replacing HTML Text with an Image

Problem

You want to replace HTML text, such as a heading, with an image that contains visually rich imagery or typography.

Solution

Use the Gilder/Levin image replacement technique.

First, introduce a `span` element before the HTML text:

```
<h1>
 <span></span>
 Replacement Text Is Here
</h1>
```

Then set the width and height for the replacement image on the `h1` selector as well as setting the positioning of the element to `relative`:

```
h1 {
 width: 216px;
 height: 72px;
 position: relative:
}
```

Next, set the positioning of the `span` element to `absolute` and adjust the width and height of the `span` inside the `h1` element; the `span` element now overlaps the HTML text. The last step is to bring in the replacement image through the `background` property, as shown in Figure 4-22:

```
h1 span {
 background: url(replacementimage.jpg) no-repeat;
 position: absolute;
 width: 100%;
 height: 100%;
}
```
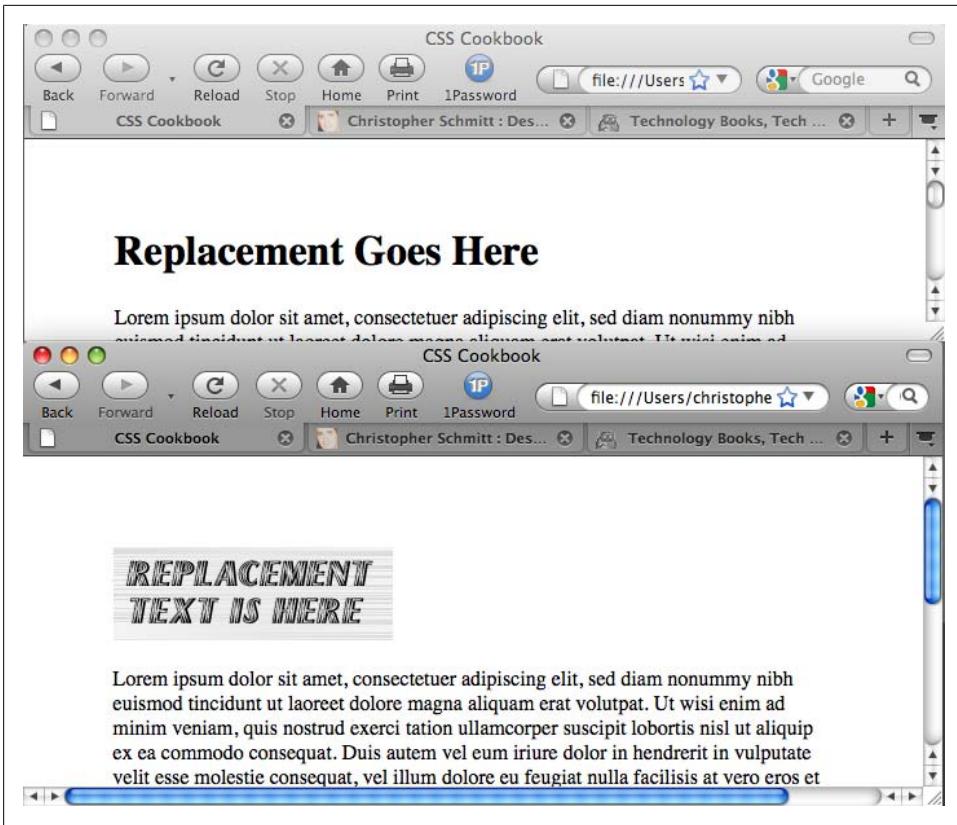
Figure 4-22. The HTML text replaced by a graphic

Discussion

There are several image replacement techniques in web development, and all seem to have their own benefits and drawbacks.

Fahrner Image Replacement method

Todd Fahrner is one of the people credited with the original concept of an image replacement technique.

The markup for the Fahrner Image Replacement (FIR) technique also introduces a nonsemantic span element, except that the span element is wrapped around the content:

```
<h1>
 <span>
  Replacement Text Is Here
 </span>
</h1>
```

Then the CSS rules bring in the replacement image through the selector for the `h1` element while hiding the text:

```
h1 {
  background: url(replacementimage.jpg) no-repeat;
  width: 216px;
  height: 72px;
}
h1 span {
  display: none;
}
```

Problem with the FIR method. Its easy implementation made the FIR technique quite popular in web development. However, screen readers used by people with disabilities would often skip reading the HTML text because the `span` element set the text to be hidden from view. Thus, important text would be lost to those members of a site's audience.

Phark image replacement method

Both the FIR and the Gilder/Levin image replacement methods use a nonsemantic `span` tag to achieve their results. Another image replacement technique, created by Mike Rundle from Phark.net, removes the need for the `span` tag.

First, adjust the HTML by removing the `span` tag:

```
<h1>
 Replacement Text Is Here
</h1>
```

For the CSS rules, use a negative value for the `text-indent` property instead of using the display property to hide the text:

```
h1 {
  text-indent: -9000em;
  background: url(replacementimage.jpg) no-repeat;
  width: 216px;
  height: 72px;
}
```

Problem with the Phark method. Like the other methods, the Phark image replacement method works very well. Its main drawback is that the HTML text does not appear if a site visitor has turned off images from being viewed in his browser.

CSS3 approach to image replacement

The CSS3 specification provides an easy method for image replacement, if browsers were to implement it. For example, to replace text within an `h1` element, all you would require is one declaration block:

```
h1 {
  content: url(logo.gif);
}
```

The specification also makes no limits on what kinds of multimedia can be supported with the `content` property. In theory, a web developer could place a QuickTime movie instead of an animated GIF:

```
h1 {
  content: url(logo.mov);
}
```

At the time of this writing, support for this part of the CSS3 specification is not provided in modern browsers.

See Also

Information on inserting content with CSS3 at *http://www.w3.org/TR/css3-content/#in serting3* and *http://my.opera.com/ODIN/blog/css-3-image-replacement*

4.21 Building a Panoramic Image Presentation

Problem

You want the width of an image to increase or decrease as a user resizes his browser window, as shown in Figure 4-23.

Solution

Place an image element that refers to a panoramic image into the background of a block-level element:

```
<h1>Visit France City!</h1>
 <div><img src="frenchtown.jpg" alt=" " /></div>
<h2>The quaint and charming little destination in France</h2>
```

Position the image element in the upper-right corner of the block-level element and then hide the image by setting the `display` to `none`:

```
div {
  background-image: url(frenchtown.jpg);
  background-repeat: no-repeat;
  background-position: top right;
  height: 300px;
  border: 1px solid black;
  max-width: 714px;
}
div img {
  display: none;
}
```
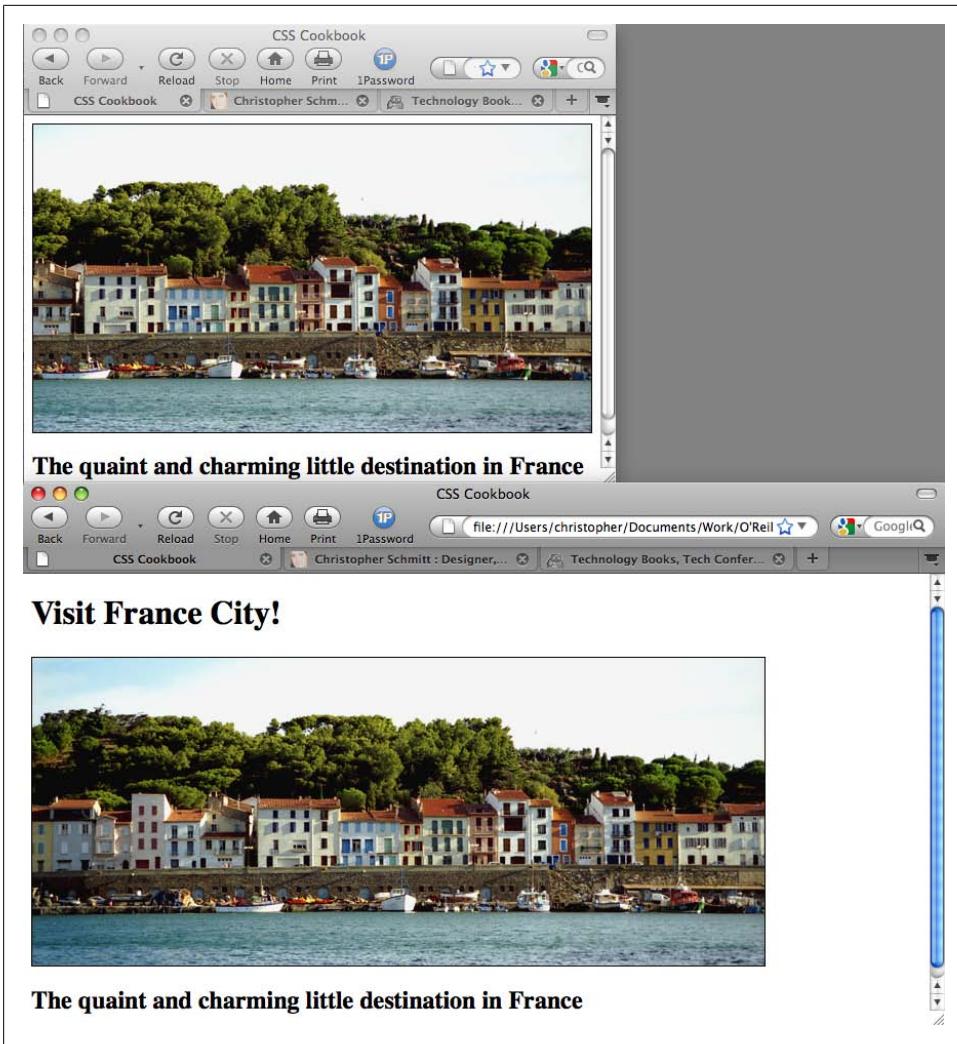
Figure 4-23. Revealing more of the panoramic image as the browser window increases in size

When the image is placed as a background image, it will be resized based on the size of the browser window.

Discussion

To create a panoramic presentation, you need a wide photograph. You then need to position the image element in the upper-right corner of the block-level element so that the image will grow or shrink depending on the size of the browser window. The use of the `max-width` property constrains the width of the `div` element from expanding beyond the width of the image itself.

In this Solution, the same image is used in both the HTML and the CSS. The rationale behind this approach is to make sure the image (or content) displays, even if the user agent rendering the page doesn't understand CSS.

See Also

The CSS 2.1 specification for `max-width` at *http://www.w3.org/TR/CSS21/visudet.html #propdef-max-width*

4.22 Combining Different Image Formats

Problem

You want to combine two different image formats into one presentation. For example, you want to combine GIF and JPEG images into one graphical presentation, as shown in Figure 4-24.
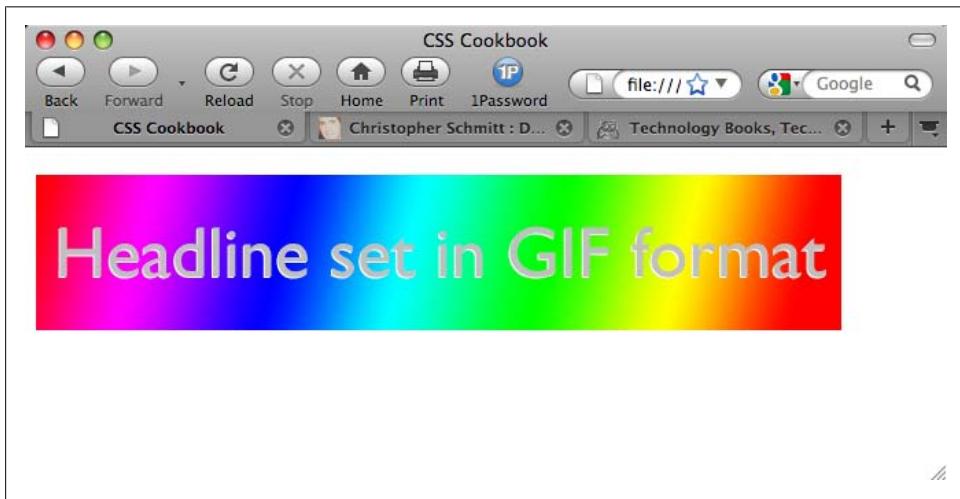
Figure 4-24. Two different image formats combined into one

Solution

Place an image inside a block-level element, such as a `div` or `h2`:

```
<h2><img src="headline_text.gif" alt="Headline image set in
GIF format" /></h2>
```

Using an image-editing program, separate the elements of the image into separate file formats (see Figure 4-25).

Figure 4-25. Two images that will be used to create one image

Name one of the images the same as the image referred to in the `src` attribute for the `img` element. Place the other image in the background of the block-level element to merge both images into one presentation:

```
h2 {
 background-image: url(headline_bkgd.jpg);
 background-repeat: none;
 width: 587px;
 height: 113px;
}
```

Discussion

The two prevailing image formats on the Web are GIF and JPEG (and PNGs making a strong triad). Both compress images in different ways. Typically, images with flat areas of color compress better in the GIF format, whereas JPEG images are better for photos or images that contain fine color gradations.

In the example shown in Figures 4-24 and 4-25, the file size of the two separate images added together is actually less than the file size of the final, combined image. This occurs because part of the image would work against the compression scheme of one file format. If you saved the presentation as one GIF, the photographic portions of the image would create an inflated file size. And if you saved the image as a JPEG, the areas of flat color would inflate the size. By splitting up the images into different formats that leverage their respective compression schemes, you reduce file sizes overall.

Although the method in this Solution uses background properties in CSS, you can accomplish the same effect by positioning block elements that contain inline images. For example, in Figure 4-26 you can see that the line art of the boat was overlaid on the photograph of the two children.
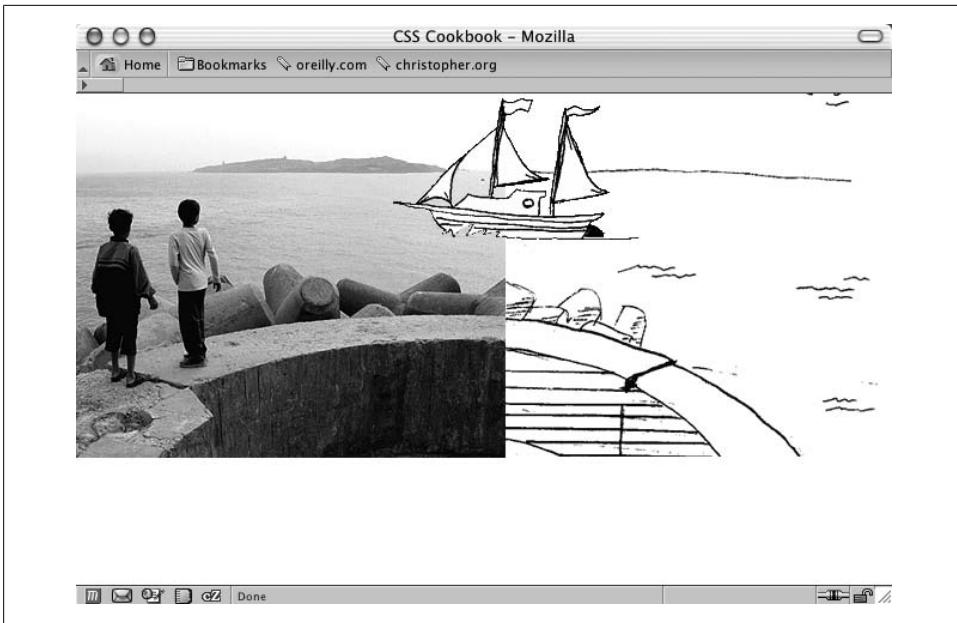
Figure 4-26. Intricate combination of different image formats

To make this method work, wrap the image elements in block-level `div` elements, as shown in the following HTML code:

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
    "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
 <head>
  <title>CSS Cookbook</title>
 </head>
 <body>
  <img src="kids.jpg" width="360" height="304"  alt="kids
playing" />
  <div id="boat"><img src="boat.gif" width="207" height="123"
   alt="boat" /></div>
  <div id="water"><img src="landscape.gif" width="315"
height="323"
   alt="landscape" /></div>
 </body>
</html>
```

Then, through CSS, set the `position` of the elements to `absolute`. By setting the `position` to `absolute`, you take the elements out of the normal flow of the web page, and instead you assign values to the `left`, `top`, and `z-index` properties to determine their new placements:

```
#boat {
 position:absolute;
 width:207px;
```

```
  height:123px;
  z-index:2;
  left: 264px;
  top: 0;
}
#water {
 position:absolute;
 width:315px;
 height:323px;
 z-index:1;
 left: 359px;
 top: -20px;
}
```

The `left` and `top` properties indicate the placement of the images within their nearest positioned ancestor element or the initial containing block. In this case, it's the initial containing block to the `div` elements. Furthermore, the `body` element's margin has a value of `0`, meaning that the origin point is in the upper-left corner of the browser's viewport.

```
body {
 margin: 0;
 }
```

Even though this method works, if the web document is later modified, exact positioning becomes a design liability. For example, adding a simple headline above the images in the HTML results in the anomaly shown in Figure 4-27:

```
<h2>Kids Welcome New Boat!</h2>
 <img src="kids.jpg" width="360" height="304"  alt="kids
playing" />
 <div id="boat"><img src="boat.gif" width="207" height="123"
   alt="boat" /></div>
 <div id="water"><img src="landscape.gif" width="315" height="323"
alt="landscape" /></div>
```
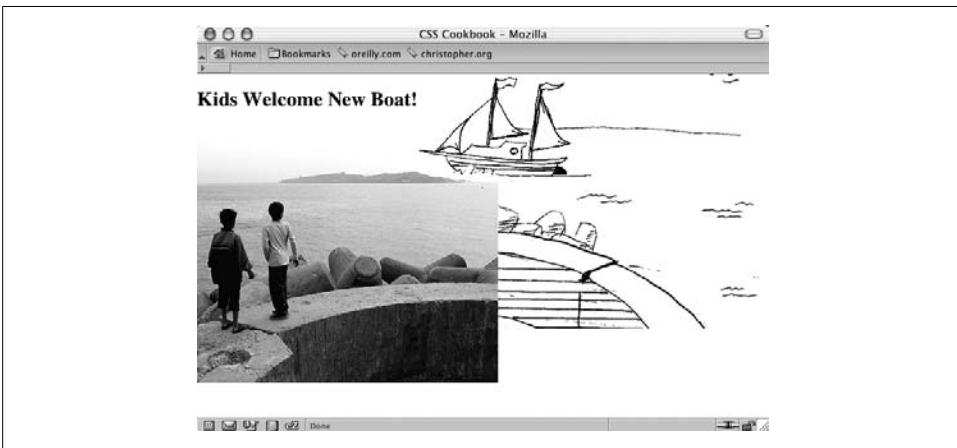

Figure 4-27. Presentation breaks with addition of heading

Because the image of the children has not been positioned with `absolute`, it moves down the flow of the document. The other image stays in place because it has been positioned within the initial containing block and is still in the same place it was before the headline was added.

By using the background-positioning method within block-level elements, you can create a self-contained module. Then, when content is added to and removed from the web page, the presentation remains whole, as seen in Figure 4-28 and shown in the following code:

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
    "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
 <head>
 <title>CSS Cookbook</title>
 <style type="text/css">
 body {
  margin: 5% 10% 0 10%;
 }
 #content {
  background-image: url(landscape.gif);
  background-repeat: no-repeat;
  background-position: bottom right;
  height: 400px;
  width: 674px;
 }
 h2 {
  margin: 0;
  padding: 0;
  background-image: url(kids.jpg);
  background-repeat: no-repeat;
  background-position: bottom left;
  height: 400px;
  width: 600px;
 }
 #boat {
  background-image: url(boat.gif);
  background-repeat: no-repeat;
  display: block;
  width: 207px;
  height: 123px;
  margin-left: 250px;
  margin-top: 75px;
 }
 </style>
 </head>
 <body>
  <div id="content">
   <h2>Kids Welcome New Boat!
    <span id="boat">
    </span>
   </h2>
  </div>
```

```
    </body>
    </html>
```


Figure 4-28. A different approach to combining images

See Also

Recipe 13.2 for creating unexpected incongruity between two elements; Recipe 13.3 for combining unlike elements

4.23 Rounding Corners with Fixed-Width Columns

Problem

You want to create rounded corners on fixed-width columns.

Solution

Create two background images, with one image containing the top corners and the other image containing the bottom corners, as shown in Figure 4-29.

Wrap a `div` element around the content that's within the column:

```
<div id="box">
 <h2>
  I Met a Girl I&#8217;d Like to Know Better
 </h2>
```

```
   <p>Lorem ipsum dolor sit amet, consectetuer adipiscing elit, sed diam
nonummy nibh euismod tincidunt ut laoreet dolore magna aliquam erat volutpat.
Ut wisi enim ad minim veniam.</p>
</div>
```
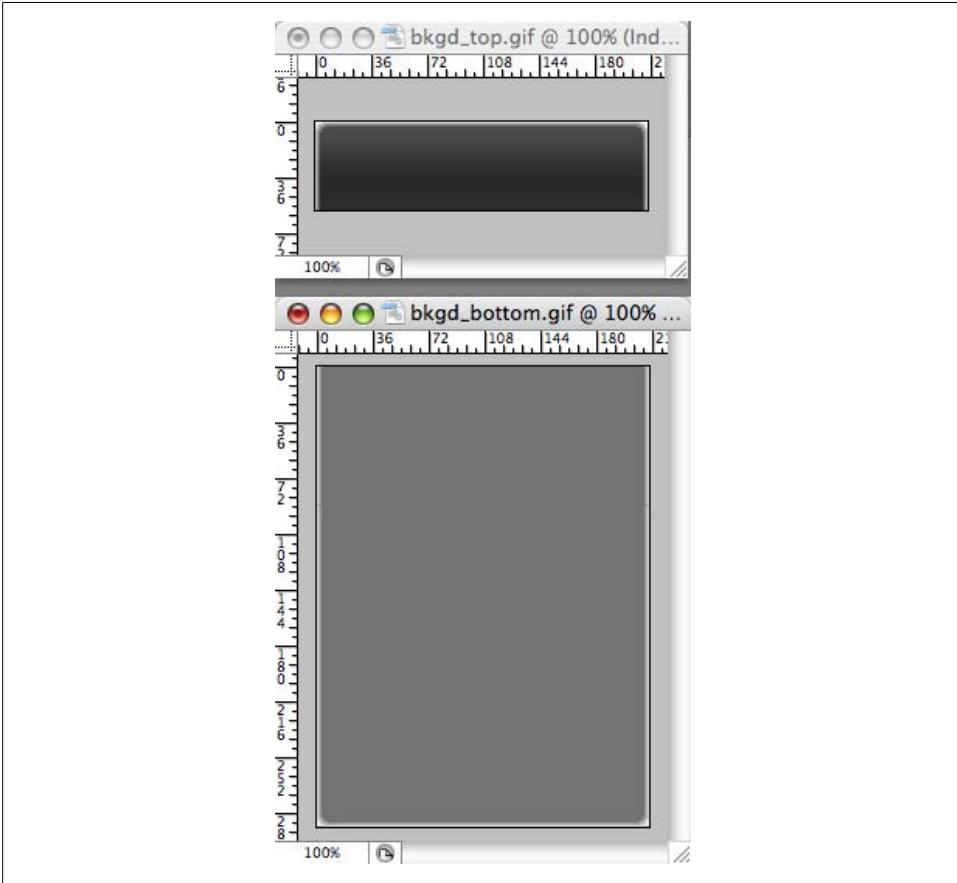

Figure 4-29. One image for the top corners and another for the bottom corners

Place the bottom background image in the `div` element:

```
#box {
 width: 214px;
 background-image: url(bkgd_bottom.gif);
 background-position: bottom;
 background-repeat: no-repeat;
}
```

Then place the top background image in the `h2` element, as shown in Figure 4-30:

```
h2 {
 background-image: url(bkgd_top.gif);
 backgroung-position: left top;
```

```
background-repeat: no-repeat;
padding: 7px 7px 3px 7px;
margin: 0;
border-bottom: 1px solid #999;
font-size: 1.3em;
font-weight: normal;
color: #eee;
}
```
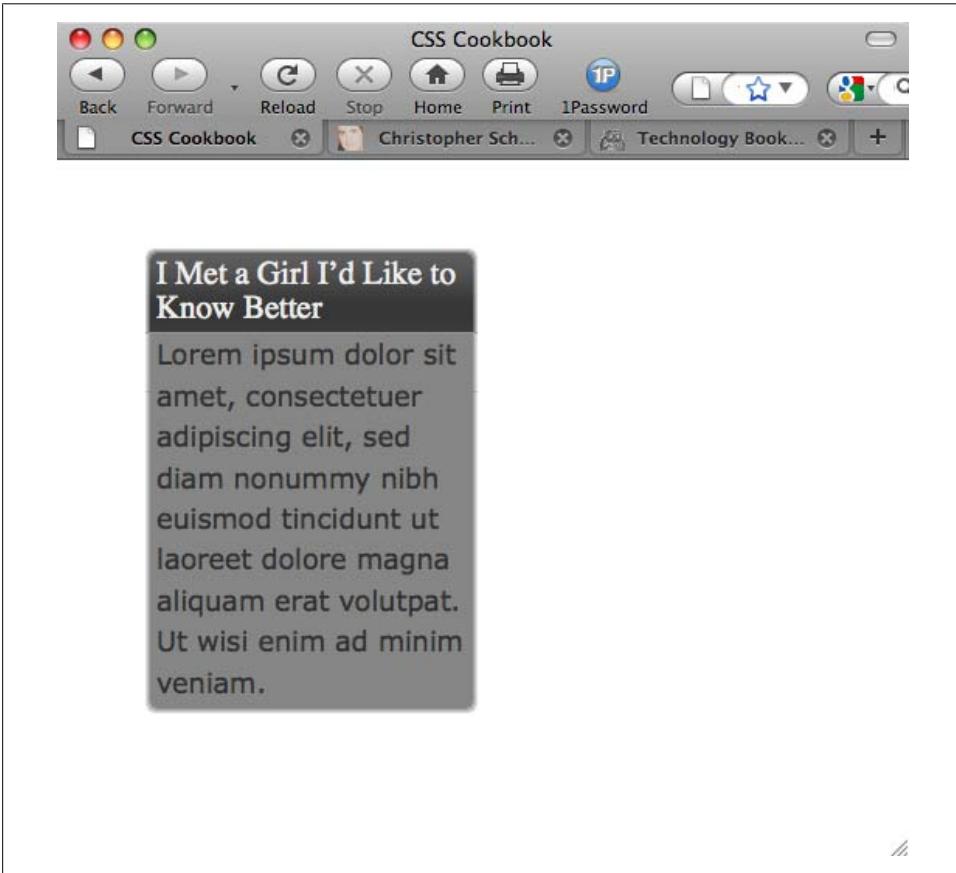

Figure 4-30. A background image placed at the bottom of the column

Discussion

To compensate for different text sizes, make the background images extend for longer than just the space specified in the design. For example, the images used in this Solution are 600 pixels tall; however, it's not unheard of to have graphics that are more than 1,000 pixels tall to ensure that a page's design maintains its integrity with extreme font sizing.

Flexible widths

By fixing the width of the column to a length unit such as pixels, it's possible to place an image containing two corners in one image. With column widths that change when the user resizes the browser, however, the fixed-width solution falls apart.

See Also

Recipes 2.15, 2.16, and 2.17 for rounding corners with flexible widths

4.24 Rounding Corners (Sliding Doors Technique)

Problem

You want to round corners in columns that have flexible widths.

Solution

Use the Sliding Doors technique that was made popular by web designer Douglas Bowman.

Create the design of the rounded corners, as shown in Figure 4-31.
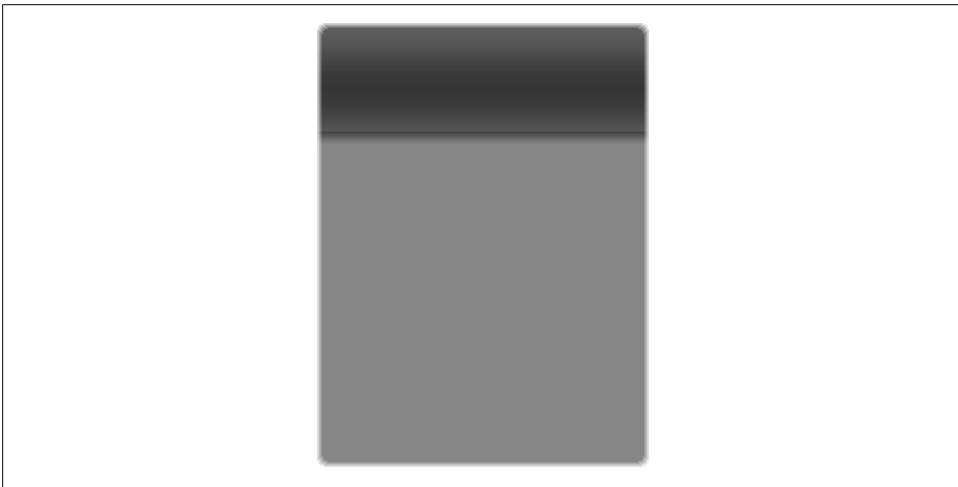
Figure 4-31. The basic design for the column

Then create separate graphics for the four corners, as shown in Figure 4-32.

Wrap the content that is in the column with additional `div` elements:

```
<div id="box">
 <div id="innerhead">
  <h2>
```

```
   I Met a Girl I&#8217;d Like to Know Better
  </h2>
 </div>
 <div id="content">
  <div id="innercontent">
   <p>Lorem ipsum dolor sit amet, consectetuer adipiscing elit, sed
diam nonummy nibh euismod tincidunt ut laoreet dolore magna aliquam
erat volutpat. Ut wisi enim ad minim veniam.</p>
  </div>
 </div>
</div>
```
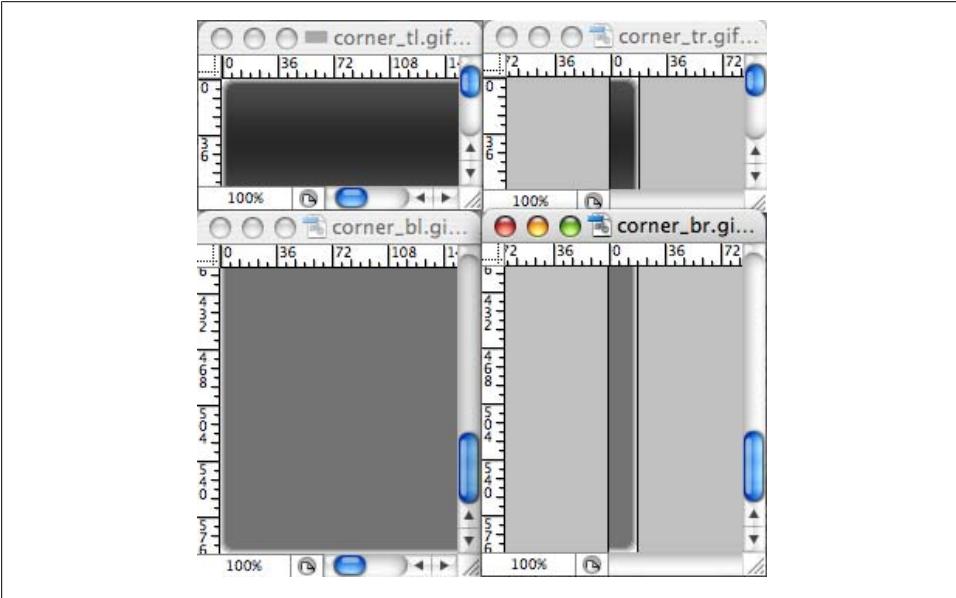

Figure 4-32. The column design split up into four graphics

Then place the background images through CSS, as shown in . The top-left corner goes in the `innerhead` id selector, the top-right corner slides into the preexisting `h2` element, the `content` id selector gets the bottom-left selector, and the `innercontent` id selector houses the bottom-right graphic:

```
#innerhead {
 background-image: url(corner_tl.gif);
 background-position: top left;
background-repeat: no-repeat;
}
h2 {
 background-image: url(corner_tr.gif);
 background-position: top right;
 background-repeat: no-repeat;
 margin: 0;
 padding: 7px;
```

```
  border-bottom: 1px solid #999;
  font-size: 1.3em;
  font-weight: normal;
  color: #eee;
}
#content {
 background-image: url(corner_bl.gif);
 background-position: bottom left;
 background-repeat: no-repeat;
}
#innercontent {
 background-image: url(corner_br.gif);
 background-position: bottom right;
 background-repeat: no-repeat;
}
```


Figure 4-33. Rounded corners appearing on the column

Discussion

The `div` and `h2` elements act as hooks to add background images to all four corners of the column. As the browser resizes, the background images stay in their respective corners, as shown in Figure 4-34.

Figure 4-34. Rounded corners maintained, even though the column expands

To make sure the design integrity is maintained as the column expands, further digital image editing is required. Manipulate one side, either the left or the right, and expand the two graphics both vertically and horizontally. For example, the bottom-right and bottom-left graphics (see Figures 4-35 and 4-36) were expanded for this Solution.
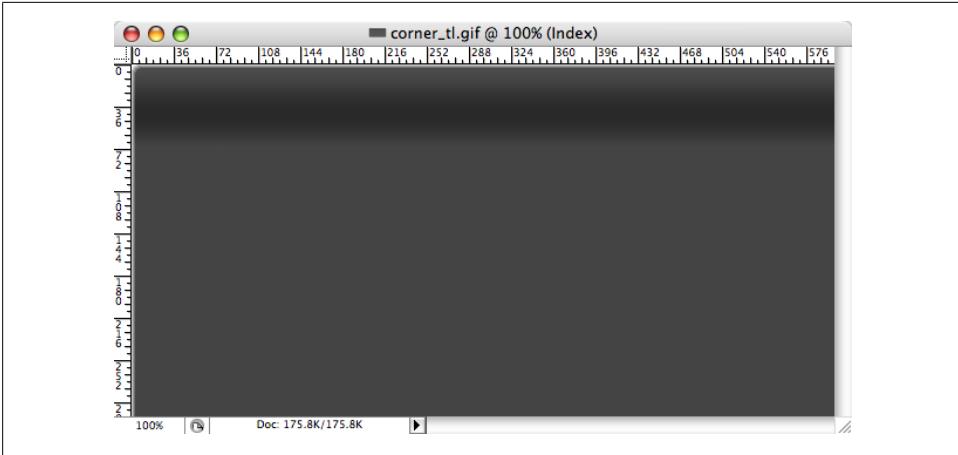
Figure 4-35. The bottom-right graphic, which is 600 pixels wide and more than 250 pixels tall
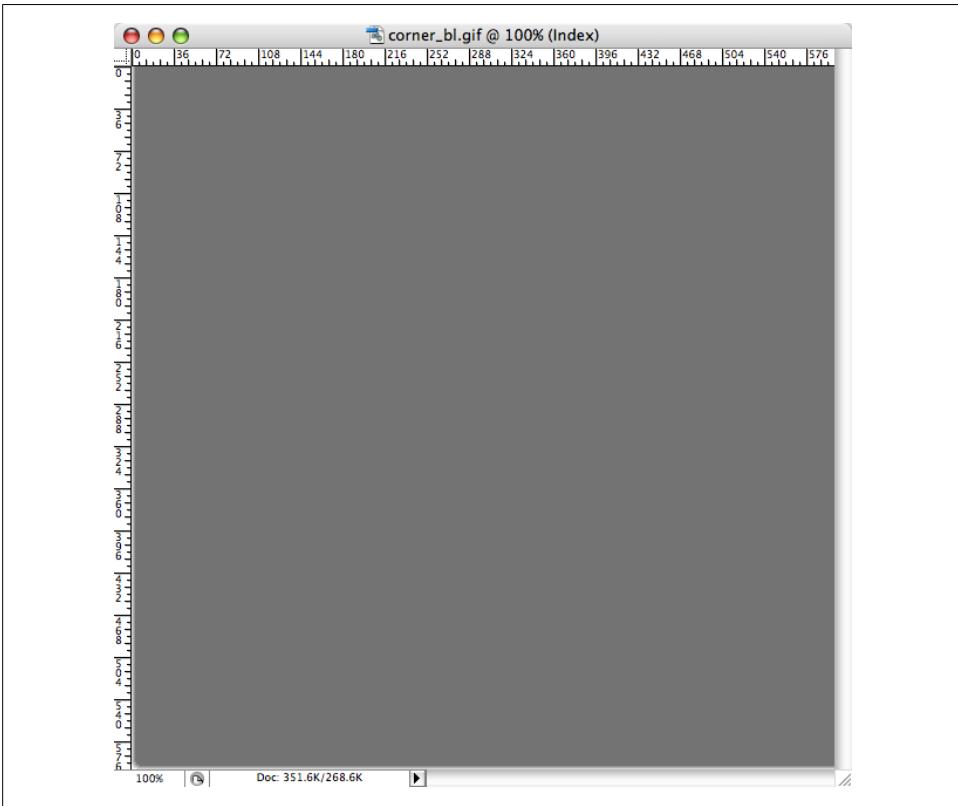
Figure 4-36. The bottom-left graphic, which is 600 pixels wide and 600 pixels tall

See Also

Recipe 2.16 for a simple solution to rounding corners of a column

4.25 Rounding Corners (Mountaintop Technique)

Problem

You want to create one set of graphics for rounded graphics while being able to display many background colors within the column.

Solution

Use the Mountaintop technique that was popularized by web designer Dan Cederholm.

Create a small graphic that will act as the basis for the rounded corners, as shown in Figure 4-37.
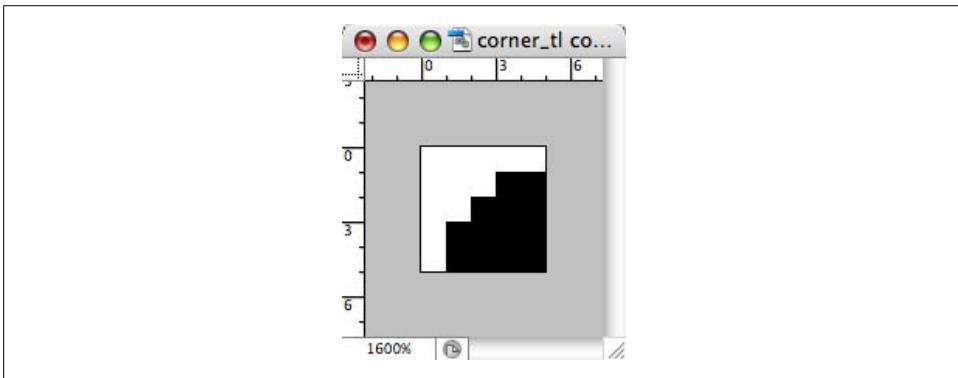
Figure 4-37. The top-left corner graphic

> The black color in Figure 4-37 will be set to transparent when the image is exported as a GIF.

Export the image as a GIF with the filename *corner_tl.gif*.

Then rotate the image 90 degrees (see Figure 4-38) and export it as a GIF image, naming it *corner_tr.gif*. Repeat the last two steps to create the bottom corners, *corner_br.gif* and *corner_bl.gif*.
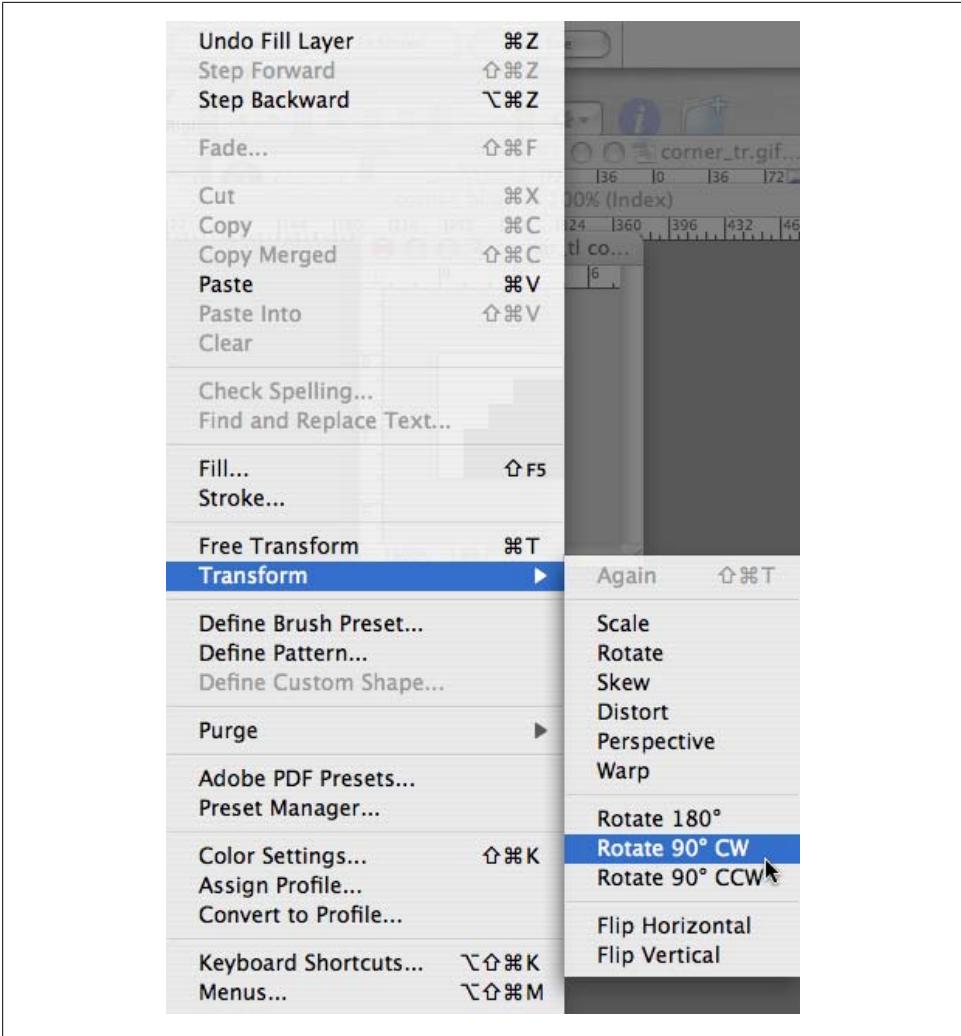
Figure 4-38. Rotating the image 90 degrees

Add additional `div` elements around the column content:

```
<div id="box">
 <div id="head_outer">
  <div id="head_inner">
   <h2>
    I Met a Girl I&#8217;d Like to Know Better
   </h2>
  </div>
 </div>
 <p>Lorem ipsum dolor sit amet, consectetuer adipiscing elit, sed diam
nonummy nibh euismod tincidunt ut laoreet dolore magna aliquam erat
```

```
    volutpat. Ut wisi enim ad minim veniam.</p>
    </div>
```

Then place the four corner graphics within the `id` and `p` selectors, as shown in Figure 4-39:

```
div#box {
 width: 55%;
 background-color: #999999;
 background-image: url(corner_bl.gif);
 background-repeat: no-repeat;
 background-position: bottom left;
}
#head_outer {
 background-image: url(corner_tl.gif);
 background-repeat: no-repeat;
}
#head_inner {
 background-image: url(corner_tr.gif);
 background-repeat: no-repeat;
 background-position: top right;
}
div p {
 margin: 0;
 padding: 7px;
 font-family: Verdana, Arial, Helvetica, sans-serif;
 font-size: 1.1em;
 background-image: url(corner_br.gif);
 background-position: bottom right;
 background-repeat: no-repeat;
 color: #333333;
 font-size: .8em;
 line-height: 1.5;
}
```


Figure 4-39. Mountaintop corner example

Discussion

The beauty of the Mountaintop technique rests in its simplicity. Four small graphics made with low file sizes thanks to the GIF compression algorithm are placed in the background of four block-level elements.

Also, there is no need to expand a couple of images to make sure the design integrity is maintained as the column resizes, as you do with the Solution for Recipe 3.22.

Plus, the Mountaintop technique allows you to quickly change the column's background color without revising the corner graphics, as shown in Figure 4-40. However, you will need to change the corner graphics if the background color of the web page or column's parent element changes.
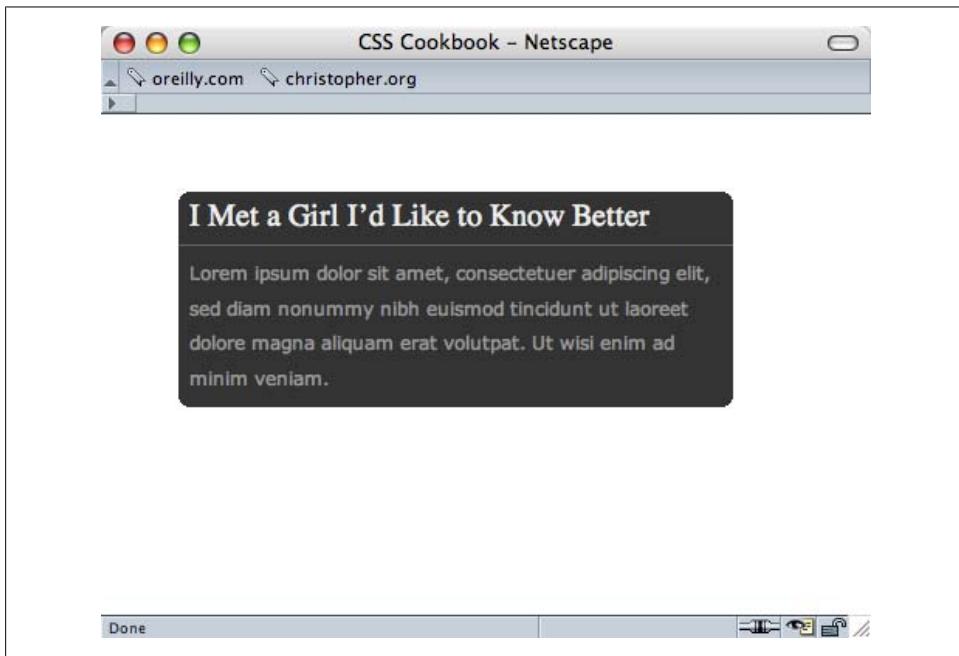
Figure 4-40. Maintaining integrity in the column even though the color has changed and the column has been resized

See Also

Recipe 4.24 for automatically adding corners on columns without custom-made images

4.26 Rounding Corners with JavaScript

Problem

You want to include rounded corners on elements without the hassle of introducing new markup or images manually.

Solution

Use the Nifty Corners Cube code by Alessandro Fulciniti.

First download the components of the Nifty Corners Cube solution, which include one CSS and one JavaScript file, from *http://www.html.it/articoli/niftycube/index.html*.

Upload both the JavaScript and CSS files associated with the Nifty Corners Cube solution. Then link the JavaScript to the web page by using the `src` attribute in the `script` element:

```
<script type="text/javascript" src="/_assets/js/niftycube.js"></script>
```

> You won't link directly to the CSS file, as the JavaScript file does that.

Next, modify the markup that will have rounded corners by giving it a unique value in the `id` attribute:

```
<div id="box">
<h2>Marquee selectus</h2>
<p>...<p>
</div>
```

Then make a separate JavaScript call to the browser indicating which element gets the rounded corners, and define the size of the rounded corners, as shown in Figure 4-41:

```
<script type="text/javascript" src="niftycube.js"></script>
<script type="text/javascript">
 window.onload=function() {
   Nifty("div#box","big");
 }
</script>
```

Discussion

Since it's almost a completely worry-free method for creating rounded corners, the Nifty Corners Cube solution has been called more of a tool than a technique.
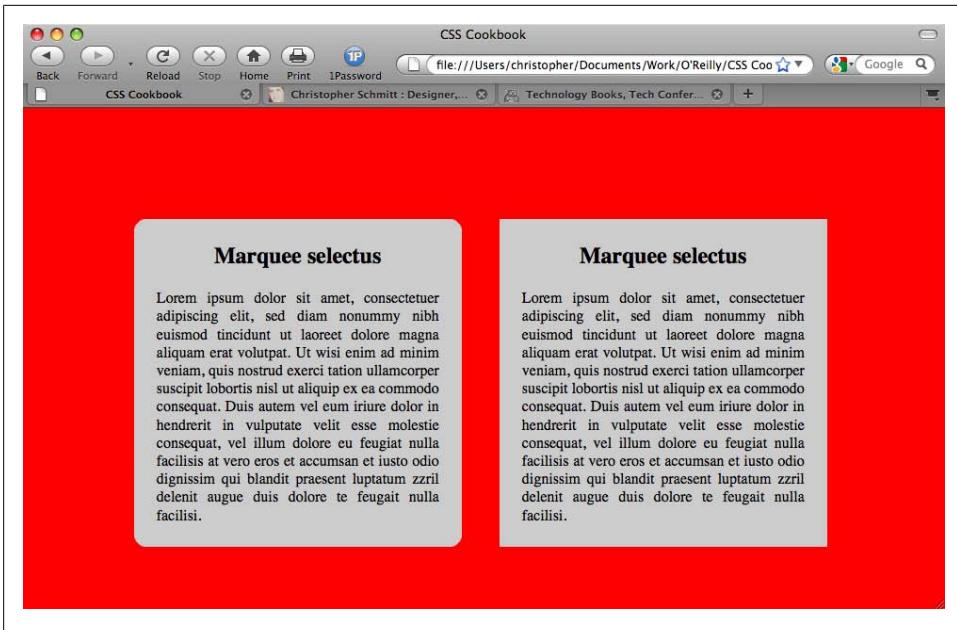
Figure 4-41. The rounded corners (left) and the default rendering (right)

> This Solution is based on JavaScript. If the user does not have JavaScript in his browser or it is turned off, the rounded corners do not appear.

Different colors

Colors are detected automatically. The JavaScript automatically changes the colors to match the background color within the element as well as its parent element (usually the body of the web page). This means a developer only has to worry about setting which element gets the curves and the size.

Different sizes

Four keyword sizes are written into the Nifty Corners Cube JavaScript: `none`, `small`, `normal` (default), and `big`. `small` is equal to the value of 2 pixels, `normal` is 5 pixels, and `big` is 10 pixels.

For example, to adjust the corners so that they are small, the JavaScript call would look like this:

```
<script type="text/javascript">
 window.onload=function() {
  Nifty("div#box","small");
}
</script>
```

Different elements

Nifty Corners Cube accepts numerous selectors, making it easier to dictate which elements should receive rounded corners, as shown in Table 4-3.

Table 4-3. Selectors understood by Nifty Corners Cube JavaScript

Selector	Example
Type	`"div"`
	`"h3"`
id	`"div#box"`
	`"h3#main"`
class	`"div.box"`
	`"h3.box"`
Descendant with id	`"div#box h3"`
	`"h3#main div"`
Descendant with class	`"div.box h3"`
	`"h3.main div"`
Grouping	`"div, h3"`
	`"div, h3.main div, p"`

For example, to apply rounded corners to multiple elements, the JavaScript function might look like the following:

```
<script type="text/javascript">
 window.onload=function() {
   Nifty("div, h3.main div, p","small");
}
</script>
```

Specific corners

The Nifty Corners Cube solution also makes allowances that developers might *not* want to apply rounded edges to all the corners. Table 4-4 lists the keywords that allow developers to single out which corner or corners to round.

Table 4-4. Keywords understood by Nifty Corners Cube JavaScript

Keyword	Meaning
tl	Top-left corner
tr	Top-right corner
bl	Bottom-left corner
br	Bottom-right corner
top	Upper corners

Keyword	Meaning
bottom	Lower corners
left	Left corners
right	Right corners
all (default)	All the corners

For example, to apply rounded corners to the top corners of multiple elements within a web page, the JavaScript function might look like the following:

```
<script type="text/javascript">
 window.onload=function() {
  Nifty("div, h3.main div, p","small top");
}
</script>
```

> Variations of this Solution for the numerous JavaScript frameworks are available today. You can find one such solution for jQuery at *http://www .malsup.com/jquery/corner/*.

See Also

http://www.html.it/articoli/niftycube/index.html for more information about Nifty Corners Cube

4.27 Setting a Shadow on an Element with CSS

Problem

You want to place a box shadow on an element with CSS.

Solution

Use the box-shadow property with proprietary browser vendor CSS properties, as shown in Figure 4-42:

```
#header {
 min-width: 250px;
 text-shadow: 0 -1px 0 rgba(0,0,0,.8);
 box-shadow: 3px 3px 19px rgba(0,0,0,.8);
 -webkit-box-shadow: 3px 3px 19px rgba(0,0,0,.8);
 -moz-box-shadow: 3px 3px 19px rgba(0,0,0,.8);
 background-image: -webkit-gradient(linear, left top, left bottom, from(#930),
 to(#6b3703), color-stop(0.5, #903000));
 background-image: -moz-linear-gradient(left top, left bottom,
 from(rgba(153,51,0,.3)), to(#6b3703), color-stop(0.5, #903000));
 margin: 7px;
```

```
    padding: 14px;
}
```
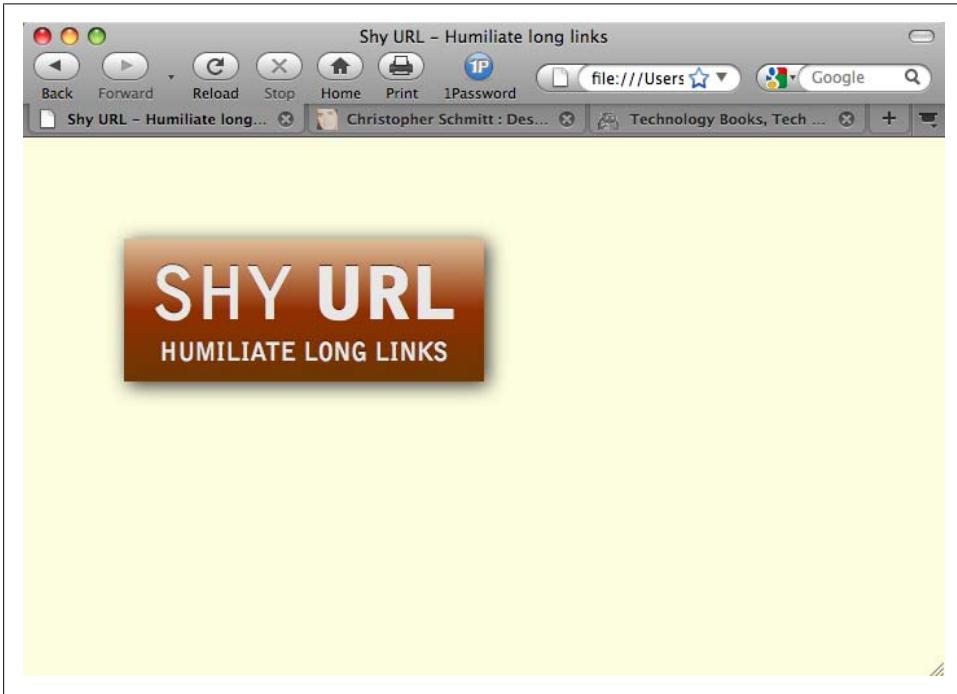

Figure 4-42. An element with a box shadow

Discussion

Box shadows work in a similar way to text shadows (see Recipe 3.32).

The first value represents the distance on the x-axis, and the second value is the value for the y-axis. A positive value means the shadow is placed down and to the right, respectively. Negative values place the shadow up and to the left.

The third value defines the radius or glow of the shadow.

The fourth value sets the color of the shadow. In the Solution, the color is set with RGBA, allowing for opacity. This approach to color (although not supported by all browsers) allows for a smoother transition to the tiling background.

See Also

Recipe 4.28 for a cross-browser method for placing an image; the CSS3 specification for box-shadow at *http://www.w3.org/TR/css3-background/#the-box-shadow*

4.28 Placing a Drop Shadow Behind an Image

Problem

You want to place a drop shadow behind an image, as shown in Figure 4-43.

Figure 4-43. A drop shadow placed behind an image

Solution

Place the image element (as shown in Figure 4-44) inside a `div` element with the `class` attribute set to `imgholder`:

```
<div class="imgholder">
<img src="dadsaranick2.jpg" alt="Photo of Dad, Sara, Nick" />
</div>
```

Set the image alignment of the `div` element to the left so that the text wraps around the image. Next, set the background image of the drop shadow in two background properties. In the first background property, use an image with an alpha transparency such as PNG:

```
div.imgholder {
 float:left;
 background: url(dropshadow.png) no-repeat bottom
  right !important;
```

```
  background: url(dropshadow.gif) no-repeat bottom right;
  margin: 10px 7px 0 10px !important;
  margin: 10px 0 0 5px;
}
```


Figure 4-44. The default rendering of the image

As for the image itself, set the `margin-right` and `margin-bottom` properties to define how much of the drop shadow image shows through. Also set a `border` property as well as `padding` to create a more dramatic effect:

```
div.imgholder img {
  display: block;
  position: relative;
  background-color: #fff;
  border: 1px solid #666;
  margin: -3px 5px 5px -3px;
  padding: 2px;
}
```

Discussion

The first step is to create a drop shadow image in an image-editing program such as Adobe Photoshop. It's best to create a background image of 600 × 600 pixels or larger,

as shown in Figure 4-45. With the image that large, this technique can accommodate almost any image used in a web page.
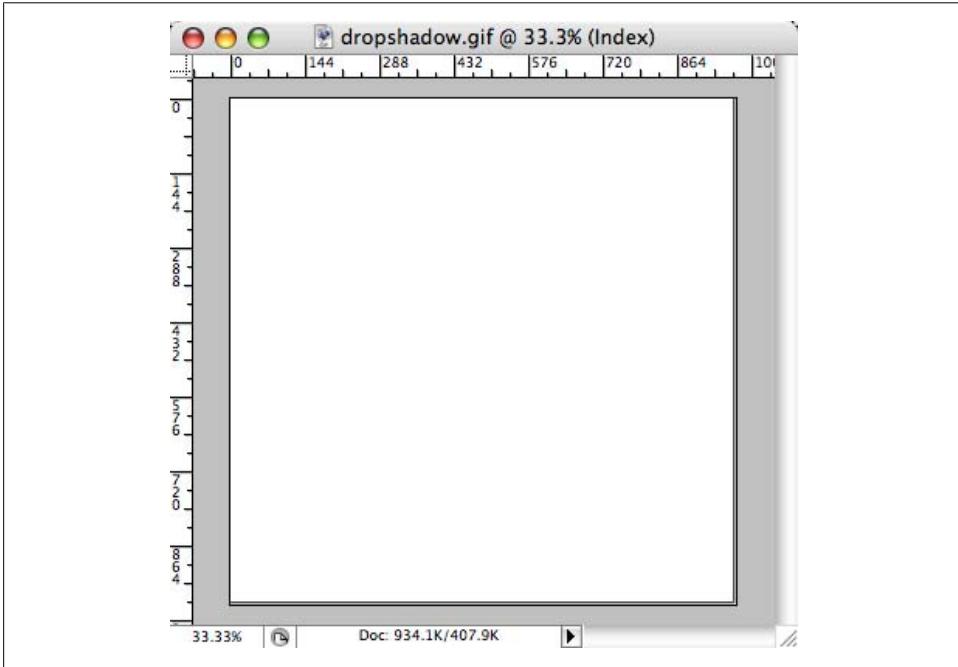
Figure 4-45. The drop shadow on the right and bottom sides

The first image background property uses the `!important` rule to display the PNG file as the drop shadow. By using the PNG file, you can change the background color or image of the web document without affecting the drop shadow. For the other browsers that don't support this rule, such as Internet Explorer for Windows, go to the next background property and use the GIF image as the drop shadow instead.

The `margin-left` and `margin-bottom` properties in the image element control how far away the drop shadow image appears from the image. If your drop shadow distance on the right or left side is more than 5 pixels (as is the one used in this Solution), change the value accordingly.

See Also

Recipe 4.29 for creating smooth drop shadows behind an image

4.29 Placing a Smooth Drop Shadow Behind an Image

Problem

You want to have soft edges for an image's drop shadow.

Solution

Adding another nonsemantic `div` wrapper around another background image allows for the creation of soft edges on drop shadows.

First, create a new image in Adobe Photoshop that will act as a mask to soften the drop shadow image. Using the same dimensions as the drop shadow, delete the entire image content in the file, leaving only a transparent background.

> If you don't have access to Photoshop, try an online version at *https://www.photoshop.com/*, or download a free digital imaging application such as GIMP (see *http://www.gimp.org/*).

Then, using the gradient tool, pick the gradient option that creates a fade from Background Color to Transparent, as shown in Figure 4-46.
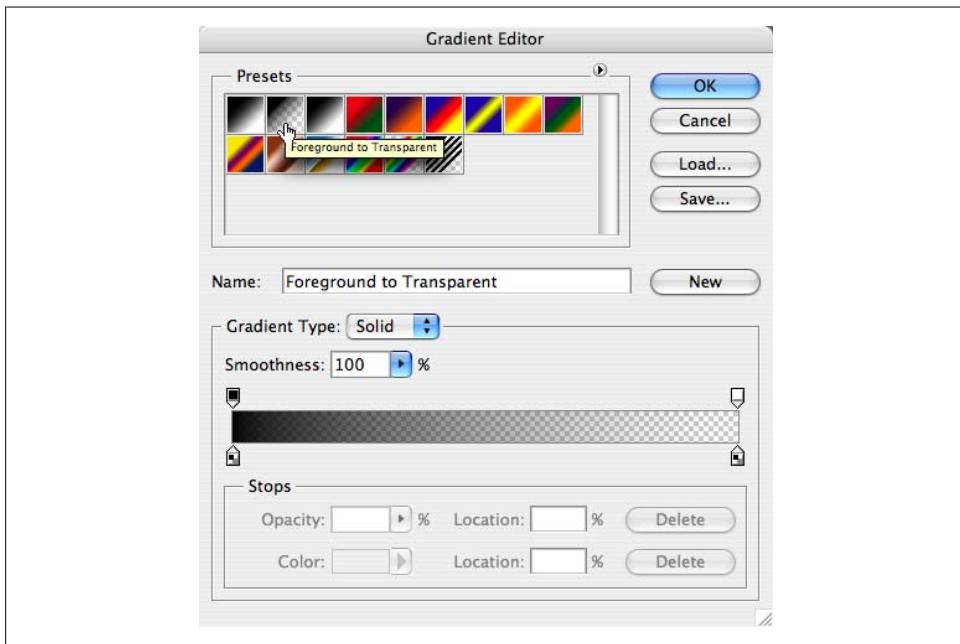
Figure 4-46. Selecting the right gradient fade

Making sure that the background color in the Toolbar matches the background color used in the website, create a 6-pixel fade from the left edge of the canvas toward the right side of the image.

Then repeat the creation of the fade, but this time create the fade from the top of the canvas to the bottom.

Next, save the image as a PNG-24 image with transparency, as shown in Figure 4-47.
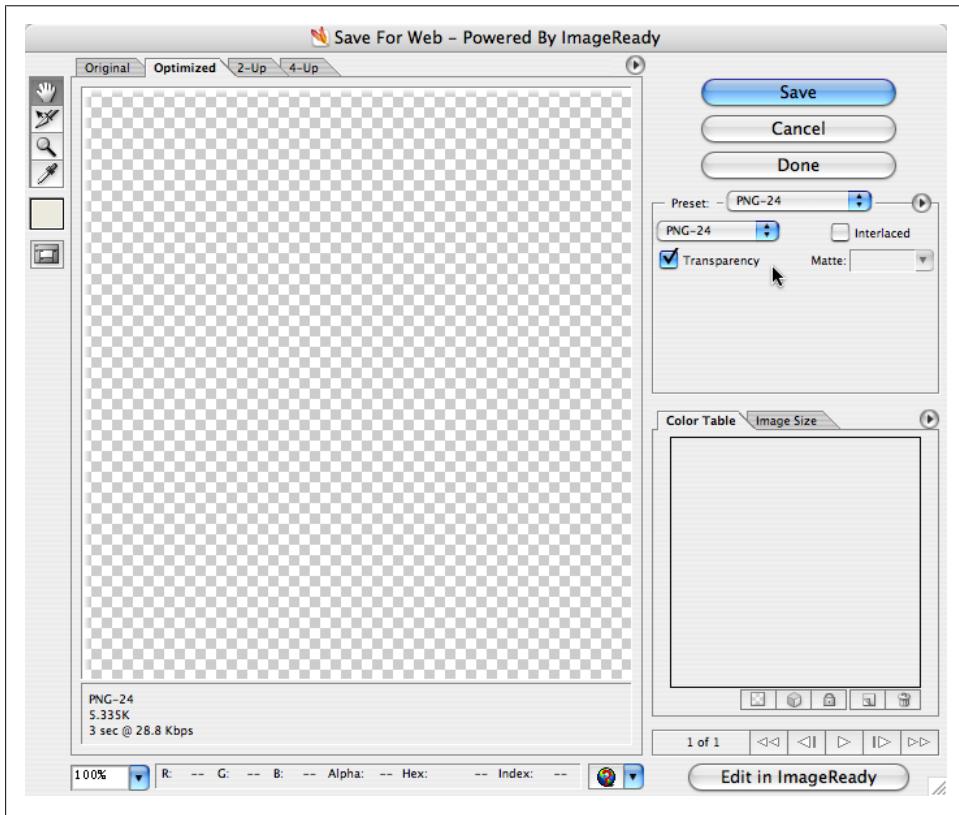
Figure 4-47. Saving the image as a PNG with alpha transparency

With the images set up, adjust the HTML to include a new `div` wrapper:

```
<div class="imgholder">
<div>
  <img src="dadsaranick2.jpg" alt="Photo of Dad, Sara, Nick" />
</div>
</div>
```

Adjusting the first CSS image wrapper, float the image to the left, apply the drop shadow, and set some spacing between the image and the HTML content:

```
div.imgholder {
 float: left;
 background: url(dropshadow.gif) no-repeat bottom right;
 margin: 0 7px 7px 0;
}
```

Next, bring in the mask that will soften the drop shadow background as well as make room to display both the drop shadow and the mask, as shown in Figure 4-48:

```
div.imgholder div {
 background: url(shadowmask.png) no-repeat;
 padding: 0 6px 6px 0;
}
```


Figure 4-48. The smooth edges now on the drop shadows

Finally, add some padding and a border to the image, as shown in Figure 4-49:

```
div.imgholder img {
    display: block;
    position: relative;
    background-color: #fff;
    border: 1px solid #666;
```

```
        padding: 2px;
}
```


Figure 4-49. The image with a drop shadow, styled a bit more

Discussion

The hard part of this Solution is creating a PNG with alpha transparency that works with the drop shadow and matches the background of the website.

Since Internet Explorer for Windows versions 5.5 through 6 do not natively support PNGs with alpha transparency, use the Solution from Recipe 4.18.

See Also

Recipe 4.27 for creating a simple drop shadow on an image

4.30 Making Word Balloons

Problem

You want to create a word-balloon effect, as shown in Figure 4-50.

Figure 4-50. A word balloon

Solution

Mark up the content for a word balloon, and include both the text to appear in the word balloon as well as the name of the person cited as the source (see Figure 4-51):

```
<blockquote>
 <p>
  <span>
   Be bold, baby!
  </span>
 </p>
 <cite>
  Christopher Schmitt
 </cite>
</blockquote>
```
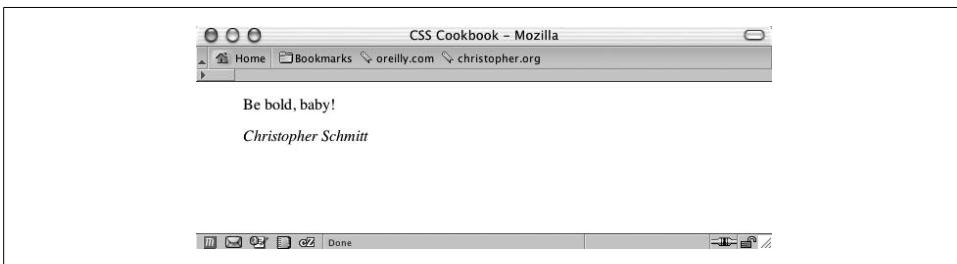

Figure 4-51. Structured content for a word balloon

Form the word balloon using the CSS `border` and `background` properties. Then align the cited text so that it falls underneath the balloon tail image:

```
blockquote {
 width: 250px;
}
blockquote p {
 background: url(balloontip.gif);
 background-repeat: no-repeat;
 background-position: bottom;
 padding-bottom: 28px;
}
blockquote p span {
 display: block;
 padding:  0.25em 0.25em 0.5em 0.5em;
 border: 1pt solid black;
 border-bottom-width: 0;
 font-size: 3em;
 font-family: "Comic Sans MS", Verdana, Helvetica, sans-serif;
 line-height: 0.9em;
}
cite {
 text-align: right;
 display: block;
 width: 250px;
}
```

Discussion

To create a word balloon you need at least one image, which includes a balloon tail and one border of the balloon (see Figure 4-52). The image is available for download at this book's site, *http://csscookbook.com/*. You create the other three sides of the word balloon by setting the border in the `span` tag.
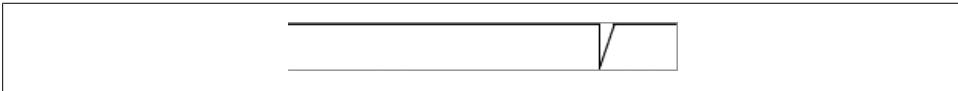
Figure 4-52. The word balloon tail

For a comic book look and feel, be sure to set the font family to Comic Sans MS, a free font from Microsoft, or use a font-embedding technique as shown in Recipe 3.4 for another font:

```
font-family: "Comic Sans MS", Verdana, Helvetica, sans-serif;
```

If you have a computer running the Windows OS, the Comic Sans MS font might be installed on your computer already. Although this is a common font, some users might not have it installed on their systems. If that is the case, the browser will look for the next font, in the order listed in the value, until it finds a font available to render the page.

You can create a more whimsical presentation using the word-balloon technique by adjusting the markup and CSS slightly. First, place a span element with a class attribute set to no around the name in the cite element:

```
<blockquote>
 <p>
  <span>
   Be bold, baby!
  </span>
 </p>
 <cite>
  <span class="no">
   Christopher Schmitt
  </span>
 </cite>
</blockquote>
```

Next, in CSS, add the following rule, which keeps the text from being displayed in the browser:

```
.no {
 display: none;
}
```

Place a photograph in the cite element through the background-position property to finish the effect (see Figure 4-53):

```
cite {
 margin: 0;
 padding: 0;
 background-image: url(baby.jpg);
 background-position: 0 0;
 height: 386px;
 text-align: right;
 display: block;
 width: 250px;
}
```

See Also

Background information about Comic Sans MS at *http://www.microsoft.com/typography/web/fonts/comicsns/default.htm*; propaganda on why not to use Comic Sans MS at *http://www.bancomicsans.com*

Figure 4-53. Word balloon coming from an image

4.31 Hindering People from Stealing Your Images

Problem

You want to make it difficult for people to copy your images from your web page.

Solution

Using a single-pixel transparent GIF as a place marker, wrap a `div` element around the `img` tag:

```
<div class="slide">
 <img src="singlepixel.gif" alt="" />
</div>
```

Then bring the image into the web page by using the `background-image` property, making sure to set the width and height of the image in both the `div` and `img` elements:

```
div.slide {
 width: 500px;
 height: 468px;
 background-image: url(face.jpg);
 background-repeat: no-repeat;
```

```
  }
  .slide img {
   width: 500px;
   height: 468px;
  }
```

Discussion

Having the single-pixel GIF as a placeholder is not necessary for the intended to be displayed; in fact, you could do away with the `img` element altogether and still have the source image be displayed:

```
<div class="slide">
</div>
```

The purpose of the transparent image is to be used as a feint. The users will think they are downloading the image they desire, when in fact they are downloading an innocuous image.

Microsoft's Image Toolbar

In Internet Explorer 6 for Windows, Microsoft started to include a feature called the Image Toolbar.

With this feature, a visitor to your site can easily email, download, or print your image with merely a click of the mouse. To keep the Image Toolbar from appearing on your web pages, add the following meta tags in between the head elements:

```
<meta http-equiv="imagetoolbar" content="no" />
<meta http-equiv="imagetoolbar" content="false" / >
```

It's a bit of a pain for developers to add code to their web page to keep someone else's product from stealing your images, but there is not much a developer can do, as Microsoft produces the most popular browser.

No images are safe

Even with the Solution and Image Toolbar workaround implemented in your web page, no image is safe from being copied from your website to a user's computer.

First, images are automatically stored by the visitor's browser and kept in a temporary folder for quick reloading of web pages. These cached images are routinely deleted after a fixed amount of time or whenever a user clears his browser's cache.

However, the browser often renames these images automatically and most visitors don't even know where the cached files are located on their computer.

The most direct route a user can take is to simply take a screen capture of his desktop with your image displayed on a browser. The user could then import the screenshot into his favorite digital imaging software application and crop the image.

So, these hindering methods might block out some visitors, but they are not solutions that will work all the time.

See Also

More information on the Image Toolbar at *http://www.microsoft.com/windows/ie/ie6/ using/howto/customizing/imgtoolbar.mspx#EXE*; a JavaScript-powered technique to hinder people from stealing images at *http://javascript.internet.com/page-details/no -right-click.html*

4.32 Inserting Reflections on Images Automatically

Problem

You want to place a reflection of a header graphic automatically.

Solution

Download the JavaScript that powers the effect from *http://cow.neondragon.net/stuff/ reflection/*.

After uploading it to the web server, link the JavaScript file into the web page between the head element:

```
<script type="text/javascript" src="scripts/reflection.js">
</script>
```

Insert into the web page the image to which you want to apply the reflection, as shown in Figure 4-54:

```
<img src="christinaleaf.png" alt="christina m. huggins" />
```
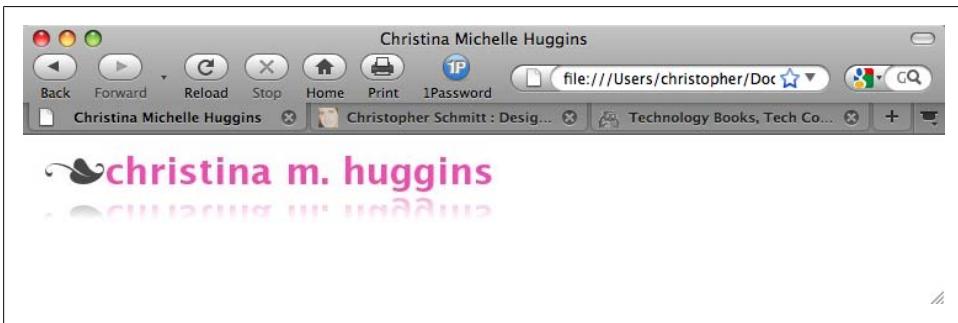

Figure 4-54. A reflection of the header graphic

To activate the reflection as shown in Figure 4-54, insert a `class` attribute with a value of `reflect`:

```
<img src="christinaleaf.png" alt="christina m. huggins" class="reflect" />
```

Discussion

As a page is rendered in the site visitor's browser, the JavaScript reflection goes through the image elements of your web page looking for `class` attributes with the `reflect` value. Then the script uses the source of the image you want to reflect and creates a new image.

If the script finds any image elements that meet those criteria, the script copies the image, flips it, and then applies the default value of 50% for both the opacity and the height to this new reflected image.

Customization features

The reflection script allows you to customize the height and the opacity of the reflection.

To adjust the height of the reflection, add a new value, `rheightXX`, to the image's `class` attribute where *XX* is the percentage of the image's height that should be duplicated in the reflection:

```
<img src="christinaleaf.png" alt="christina m. huggins"
class="reflect rheight99" />
```

As the percentage value increases, the size of the reflection increases. For example, the value of `rheight99` means that 99% of the original image's height will be used in the reflection.

To adjust the opacity of the reflection, add a new value, `ropacityXX`, to the image's `class` attribute where *XX* is the percentage of the transparency of the reflected image:

```
<img src="christinaleaf.png" alt="christina m. huggins"
class="reflect ropacity33" />
```

As the opacity value decreases, the reflection becomes less visible. For example, the value of `ropacity33` means that 33% of the original image's height will be used in the reflection.

You can use both the height and the opacity features at the same time to create subtler effects:

```
<img src="christinaleaf.png" alt="christina m. huggins"
class="reflect rheight99 ropacity33" />
```

Known browser issues

Internet Explorer for Windows 5.5 and later, Firefox 1.5 and later, Chrome, Opera 9 and later, and Safari support the reflection script. Animated images in browsers do not work with the reflection script, except for Internet Explorer for Windows. Also, scaled images appear distorted in Internet Explorer for Windows.

See Also

The blog post announcing the reflection effect at *http://cow.neondragon.net/stuff/reflection/*

4.33 Using Image Sprites

Problem

You want to save on bandwidth by placing all or most of your icons onto one image.

Solution

Place the most often used images into one master image, making sure there is plenty of space around each image.

Create enough space for each icon's own space as well as set the background image and keep it from repeating.

For this example, one icon is placed next to a heading:

```
h2 {
  margin: 0;
  font-family: Verdana, Arial, Helvetica, sans-serif;
  padding: 0 0 0 24px;
  font-weight: normal;
  background-image: url(sprite-source.gif);
  background-repeat: no-repeat;
}
```

Using `id` selectors, bring in each icon to the appropriate heading by using the `background-position` property, as shown in Figure 4-55:

```
h2#warning {
  background-position: -16px -15px;
}
h2#questions {
  background-position: -16px -51px;
}
h2#comment {
  background-position: -16px -87px;
}
h2#document {
  background-position: -16px -123px;
}
h2#print {
  background-position: -16px -159px;
}
h2#search {
  background-position: -16px -195px;
}
```
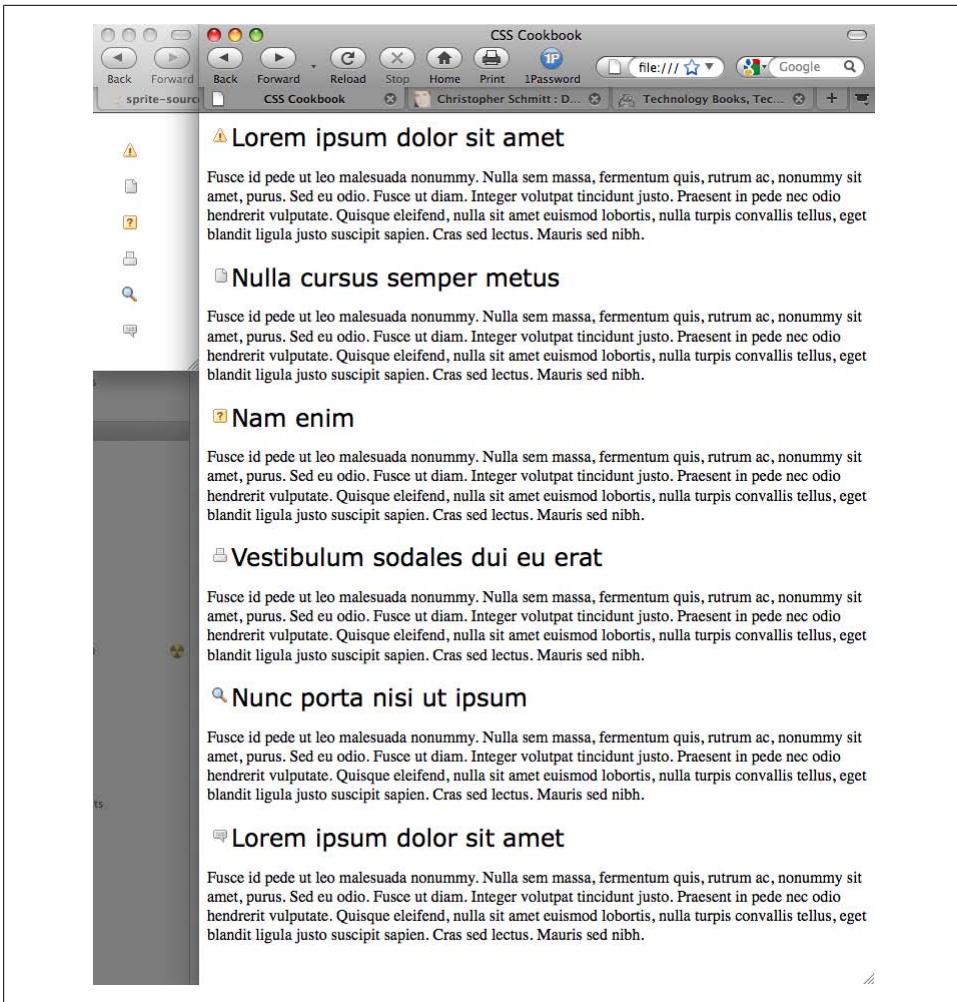
Figure 4-55. The icons displayed from one single image

Discussion

In much the same way developers use the same image over and over again to make use of a browser's ability to cache an image, using sprites helps to push that idea a bit further. By placing the separate graphic elements onto one image, web developers can reduce the number of server calls to their machines from a browser. This solution would be more apt for sites receiving medium to large amounts of traffic.

See Also

The "CSS Sprites" article at *http://www.alistapart.com/articles/sprites*

4.34 Clipping Background Images

Problem

You want to use only a portion of a background image in an HTML element.

Solution

Use the proprietary Firefox CSS value `-mox-image-rect` to isolate part of a background image, as shown in Figure 4-56:

```
div {
 background-image: -moz-image-rect(url(bkgd.jpg), 0, 100, 100%, 0);
 background-repeat: no-repeat;
 width: 200px;
 height: 200px;
 border:1px solid #666;
 float: left;
 margin-right: 20px;
}
div + div {
 background-image: -moz-image-rect(url(bkgd.jpg), 0, 100, 100%, 0);
 background-repeat: repeat;
}
```

Discussion

Similar to background clipping through Firefox, `-moz-image-rect` requires two sets of information. The first is what image is clipped:

```
background-image: -moz-image-rect(url(bkgd.jpg));
```

Next, using pixels or percentages, dictate what portion of the image is visible through a series of four comma-separated values.

These values represent the same sides of an image as the `margin` or `padding` shorthand property—`top`, `right`, `bottom`, and `left`:

```
background-image: -moz-image-rect(url(bkgd.jpg), 0, 100, 100%, 0);
```

> I did not use `px` after the integer, even though the browser understands this to be in pixels. Using `px` is not allowed and keeps the image from displaying. This might change in future releases of the browser, but it is something to look out for when implementing this feature.

The ability to clip background images is only in Firefox nightly builds at the time of this writing. The next major release of the browser after Firefox 3.5 should include support for this feature.
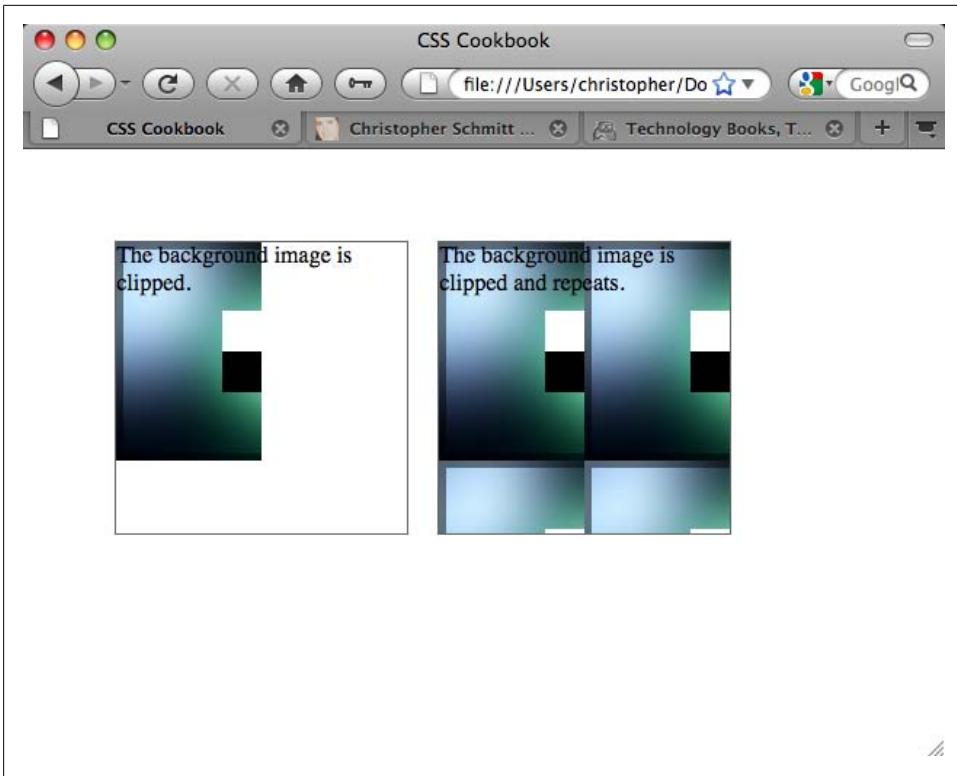
Figure 4-56. Background image clipped

To have a background image show in other browsers that do not support background clipping, include the `background-image` property with standard CSS values in the CSS rule. Other browsers ignore the Firefox background clipping:

```
div {
  background-image: url(bkgd.jpg);
  background-image: -moz-image-rect(url(bkgd.jpg), 0, 1, 100%, 0);
  background-repeat: no-repeat;
  width: 200px;
  height: 200px;
  border:1px solid #666;
  float: left;
  margin-right: 20px;
}
```

See Also

The CSS3.info blog post about Firefox 3.6 support of background clipping at *http://www.css3.info/firefox-3-6-adds-background-clipping/*

4.35 Applying Masks to Images and Borders

Problem

You want to apply a mask to an image and its borders.

Solution

First, create a PNG image with alpha transparency, as shown in Figure 4-57.
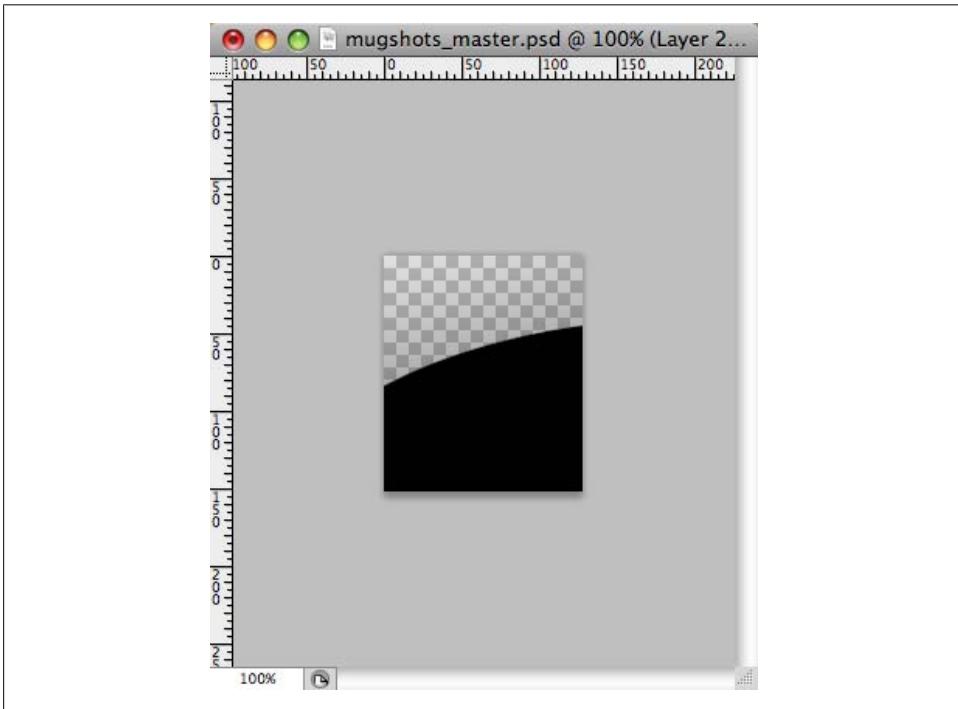
Figure 4-57. An image with alpha transparency created in Photoshop

Then apply the mask through a CSS rule, as shown in Figure 4-58:

```
img {
  display: block;
  float: left;
  margin-right:20px;
  border: 10px solid #ccc;
  padding: 2px;
  background-color: #666;
  -webkit-mask-box-image: url(mask.png);
}
```
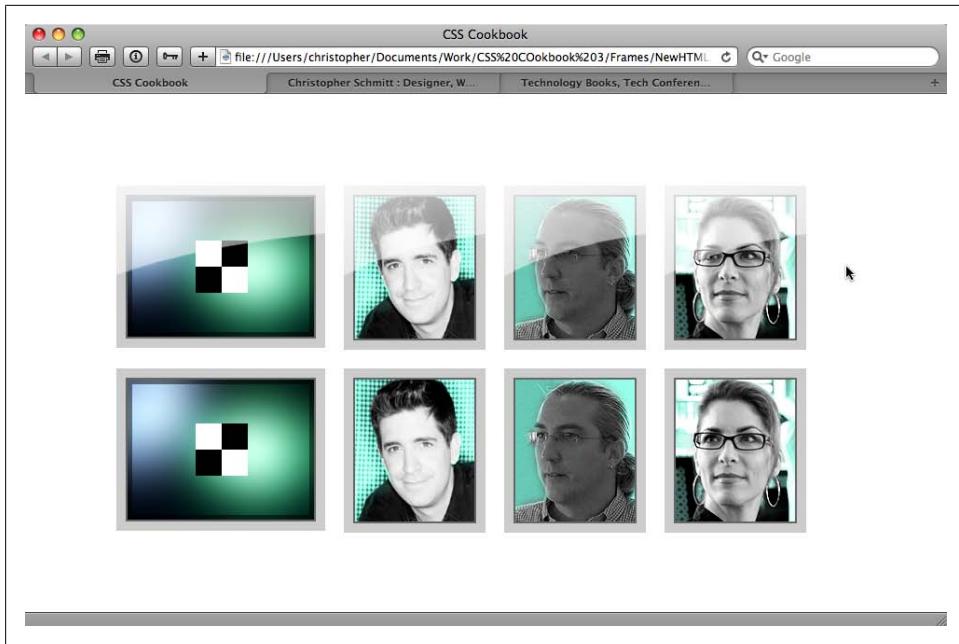
Figure 4-58. The mask applied to the top row of images and borders

Discussion

When creating a mask, every part of the image that is transparent becomes the mask or the part that hides a portion of the background image. This approach is somewhat hard to understand, since typically, alpha transparency is, well, transparent.

In addition to PNG images, you can also use SVG images as masks as well as gradients (see Recipe 4.16), as shown in Figure 4-59:

```
img {
 display: block;
 float: left;
 margin-right:20px;
 border: 10px solid #ccc;
 padding: 2px;
 background-color: #666;
 -webkit-mask-box-image: -webkit-gradient(linear, left bottom, left top,
from(rgba(0,0,0,1)), to(rgba(0,0,0,0)));
}
```

As of this writing, the only browser that supports CSS masking is Safari.

See Also

The Surfin' Safari blog post on CSS masks at *http://webkit.org/blog/181/css-masks/*
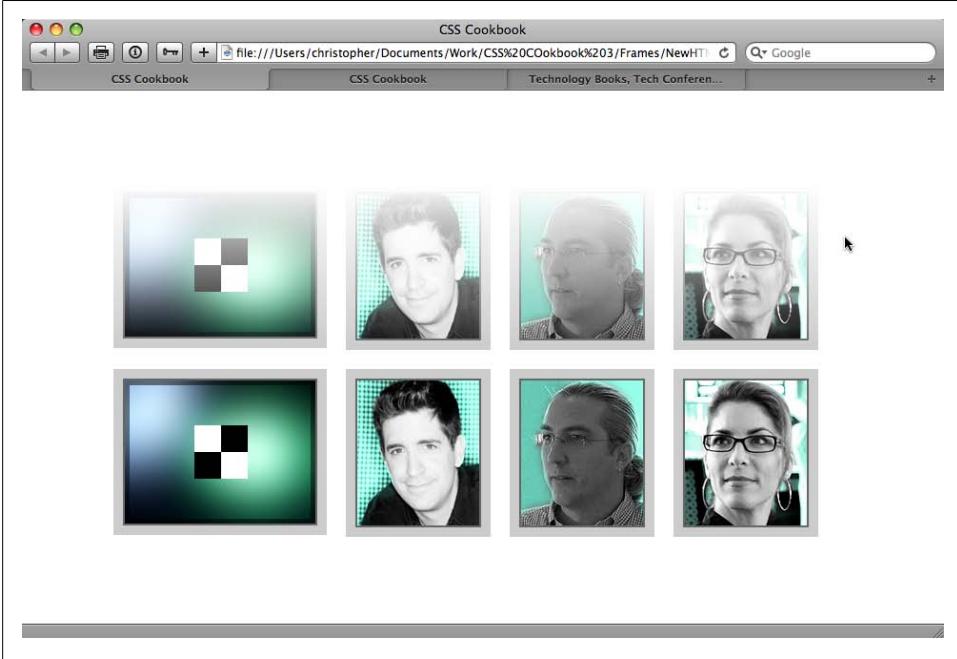
Figure 4-59. Combining gradients and masking

Page Elements

5.0 Introduction

From the most obvious design choices, such as selecting the appropriate typography and imagery, to those that are often overlooked, such as adjusting leading and color schemes, every decision affects how a message in a website is conveyed to visitors.

This chapter covers page elements that help to frame a web page like a frame for a painting. *Page elements* are items that affect the appearance of a web page, but aren't necessarily thought of as comprising a web page's design. For example, the appearance of the scroll bar is a page element.

By manipulating elements such as the margins and borders surrounding the contents of a web page, developers effectively frame the content of the page without actually styling the content.

Such simple changes can affect the page's overall design in a profound way, or they can add a subtle detail that completes the design.

5.1 Eliminating Page Margins

Problem

You want to get rid of the whitespace around the edges of a web page and between the browser chrome and the contents of the page, as shown in Figure 5-1.

Solution

Set the value of the `margin` and `padding` properties for the `html` and `body` elements to `0`:

```
html, body {
  margin: 0;
  padding: 0;
}
```
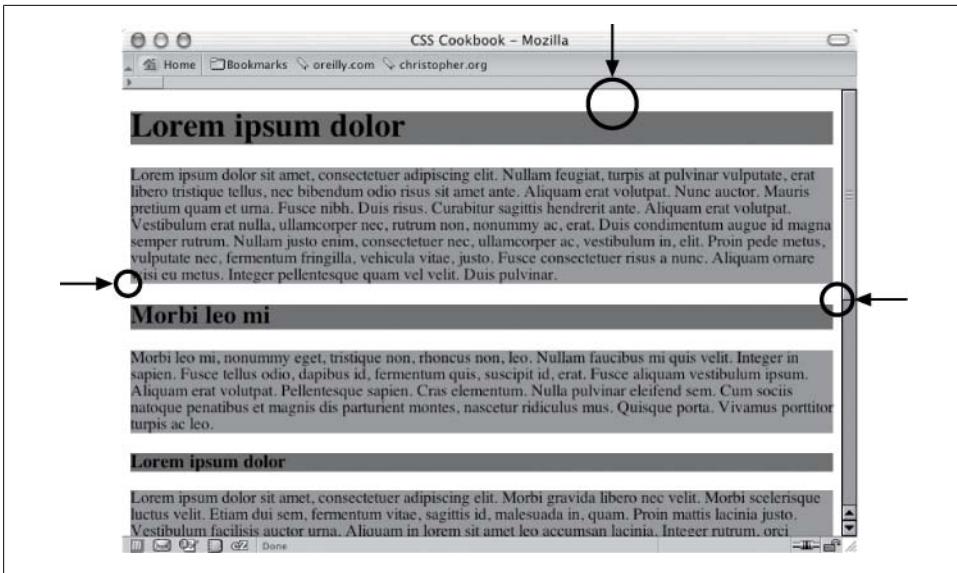
Figure 5-1. Page margins visible as the whitespace around the edges of a web page

Discussion

Setting the `margin` and `padding` properties of the `body` element to `0` helps create a full-bleed effect—in other words, it eliminates the whitespace around a web page (the units are unnecessary when specifying zero).

However, depending on the content of the web page, the `margin` and `padding` properties might not be all you need to change to get a full-bleed effect. Default properties on other elements can have unexpected side effects when attempting to change the page margin.

For example, if `h1` is the `body` element's first child element, some unintended whitespace will appear above the headline and below the top of the browser's viewport. Figure 5-2 shows this undesired effect; the background color of the headings and paragraphs is gray so that you can more clearly see the effect.

To ensure the full-bleed effect in this situation, you should set the margin and padding of the offending element (in this case, `h1`, `h2`, `h3`) to `0` as well as the `body`. This sets all the sides of the element's padding to `0`. If that setup isn't possible (e.g., you need to have a value at the bottom padding or margin), set the `margin-top` and `padding-top` values to `0` to maintain the full-bleed effect:

```
html, body {
 margin: 0;
 padding: 0;
 position: absolute;
 top: 0;
 left: 0;
}
h1, h2, h3 {
 margin-top: 0;
 padding-top: 0;
 background-color: #666;
}
p {
 background-color: #999;
}
```
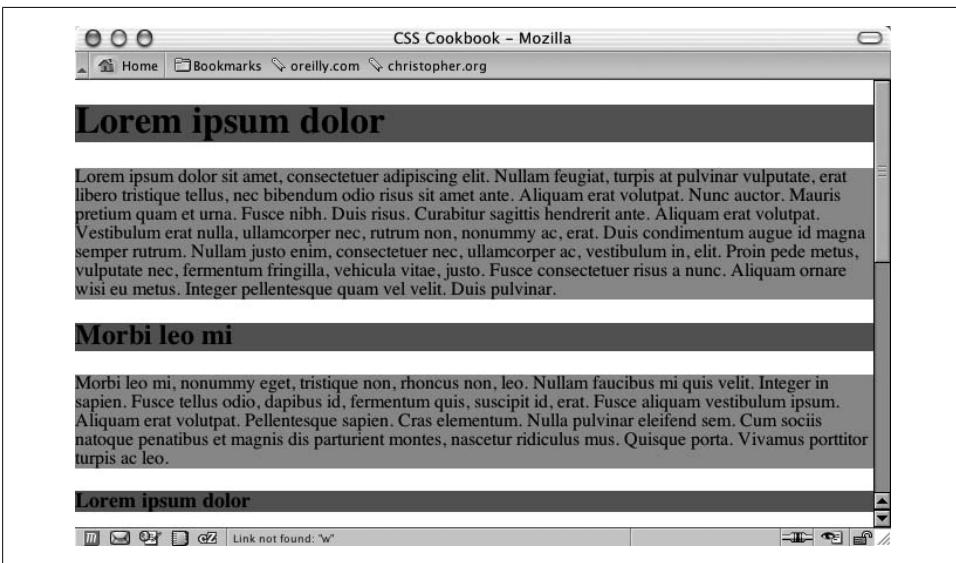

Figure 5-2. Whitespace above the heading and below the top of the browser's viewport

As you can see in Figure 5-3, this accomplishes the full-bleed effect. Notice how the gray background color of the first heading now touches the top of the browser's viewport.

See Also

Recipe 11.1 for writing one-column layouts by setting the margin and padding properties to a value other than 0
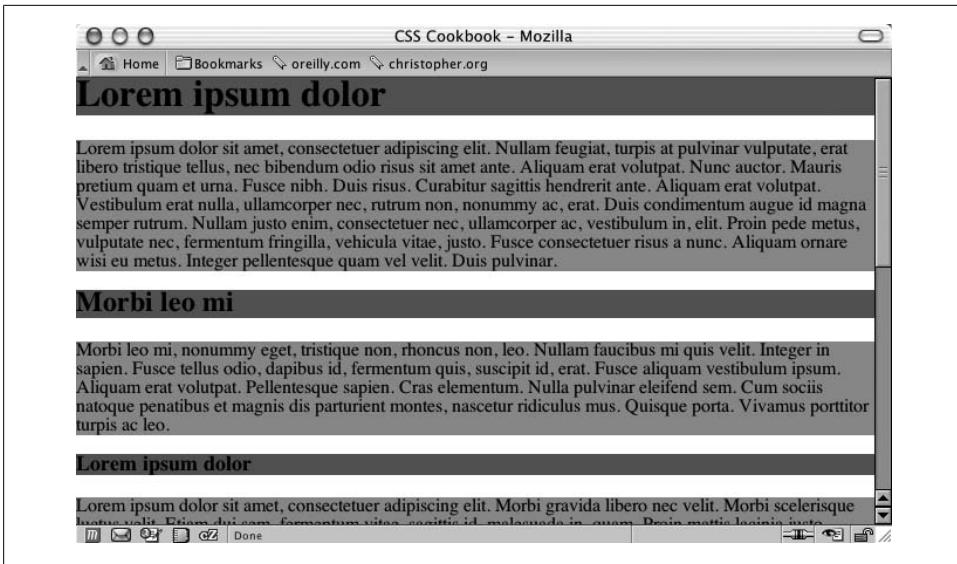
Figure 5-3. Whitespace removed above the heading

5.2 Resetting Browser-Style Defaults for Elements

Problem

You want to keep browsers from setting values for elements within web pages.

Solution

Use a separate stylesheet that sets all or most of the common HTML elements used in a web document to a value of 0 or that eliminates any auto-generated content, such as the stylesheet created by the Yahoo! Developer Network (see *http://developer.yahoo.com/*), called YUI Reset CSS:

```
/*
Copyright (c) 2009, Yahoo! Inc. All rights reserved.
Code licensed under the BSD License:
http://developer.yahoo.net/yui/license.txt
version: 2.7.0
*/
html{color:#000;background:#FFF;}body,div,dl,dt,dd,ul,ol,li,h1,h2,h3,h4,h5,h6,
pre,code,form,fieldset,legend,input,button,textarea,p,blockquote,th,
td{margin:0;padding:0;}table{border-collapse:collapse;border-
spacing:0;}fieldset,img{border:0;}address,caption,cite,code,dfn,em,strong,th,
var,optgroup{font-style:inherit;font-weight:inherit;}del,
ins{text-decoration:none;}li{list-style:none;}caption,th{text-align:left;}
h1,h2,h3,h4,h5,h6{font-size:100%;font-
weight:normal;}q:before,q:after{content:'';}abbr,acronym{border:0;font-
```

```
variant:normal;}sup{vertical-align:baseline;}sub{vertical-
align:baseline;}legend{color:#000;}input,button,textarea,select,optgroup,
option{font-family:inherit;font-size:inherit;font-style:inherit;font-
weight:inherit;}input,button,textarea,select{*font-size:100%;}
```

To apply this reset, apply the code via the `link` element before the rest of the styles in the web document:

```
<link rel="stylesheet" type="text/css"
 href="http://yui.yahooapis.com/2.7.0/build/reset/reset-min.css">
<link rel="stylesheet" type="text/css" href="mywebsitestylesheet.css">
```

Discussion

Browsers take different approaches to how they render elements on a page, from the size of `h1` headings (see Figure 5-4) to whether they should use `margin` or `padding` to indent an ordered list (see Recipe 6.3).

Figure 5-4. Main heading with subtle differences

The goal with a stylesheet that removes the default values of elements (see Figure 5-5) is to help clear some of the hurdles associated with cross-browser web development.
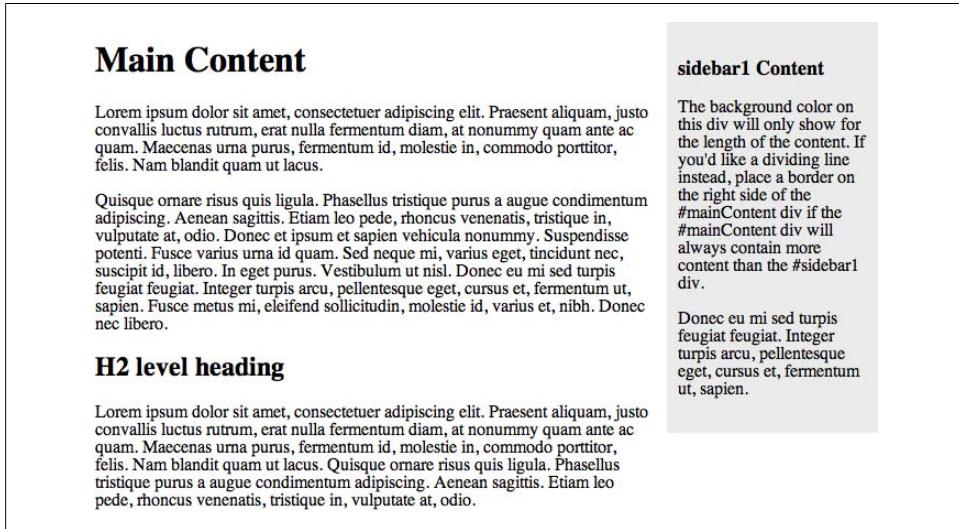
Figure 5-5. Default rendering of HTML text

Since most of the HTML elements are "reset" (as shown in Figure 5-6), the web designer needs to do more work to customize the look of a web page rather than relying on the browser to handle the styling.
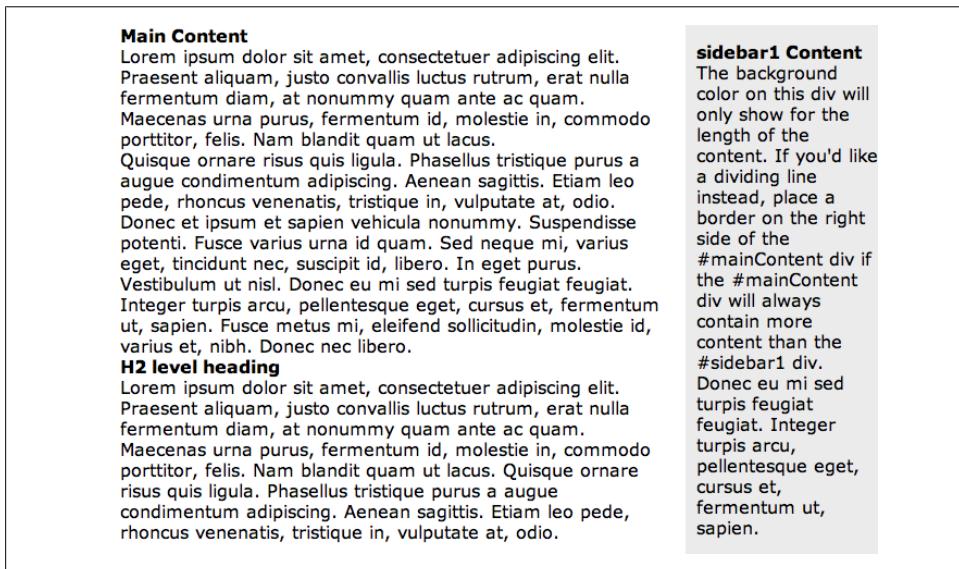
Figure 5-6. CSS reset applied

Extending CSS reset

Eric Meyer, author of *CSS: The Definitive Guide* (O'Reilly), created his own version of CSS reset after reviewing the original Yahoo! files (see *http://meyerweb.com/eric/ thoughts/2007/04/12/reset-styles/*):

```
/* v1.0 | 20080212 */

html, body, div, span, applet, object, iframe,
h1, h2, h3, h4, h5, h6, p, blockquote, pre,
a, abbr, acronym, address, big, cite, code,
del, dfn, em, font, img, ins, kbd, q, s, samp,
small, strike, strong, sub, sup, tt, var,
b, u, i, center,
dl, dt, dd, ol, ul, li,
fieldset, form, label, legend,
table, caption, tbody, tfoot, thead, tr, th, td {
    margin: 0;
    padding: 0;
    border: 0;
    outline: 0;
    font-size: 100%;
    vertical-align: baseline;
    background: transparent;
}
body {
    line-height: 1;
}
ol, ul {
    list-style: none;
}
blockquote, q {
    quotes: none;
}
blockquote:before, blockquote:after,
q:before, q:after {
    content: '';
    content: none;
}

/* remember to define focus styles! */
:focus {
    outline: 0;
}

/* remember to highlight inserts somehow! */
ins {
    text-decoration: none;
}
del {
    text-decoration: line-through;
}

/* tables still need 'cellspacing="0"' in the markup */
table {
```

```
        border-collapse: collapse;
        border-spacing: 0;
    }
```

In Meyer's reset stylesheet, he creates a starting point for the size of the fonts by setting the value for `font-size` to `100%` and `line-height` to `1` (or 100%) in the body general selector.

The background of most of the elements is set to transparent, whereas YUI Reset CSS sets the background value to white (`#fff`) for the `html` element.

The `quote` property is set to `none` for the `blockquote` and `q` elements, which is used for setting the kind of quotation marks for, well, quotations and nested quotations. This is absent from the YUI Reset CSS file.

Also absent from the YUI Reset CSS file are rules for `:focus`, which dictates how an element renders when activated by the user (such as clicking within a form element's input text box).

So, which one should you use? Try both and determine which one is right for you.

A better solution would be to create your *own* CSS reset.

Between the Yahoo! Developer Network and Eric Meyer's reset stylesheet, there are enough differences in their honest approaches to solve the problem that a right solution isn't a matter of picking the latest version or the one with the most compact code. Craft your own CSS reset tool based on how you work and what you need to do the best job you can.

See Also

YUI Reset CSS at *http://developer.yahoo.com/yui/reset/*; Eric Meyer's blog post titled "Reset Reloaded" at *http://meyerweb.com/eric/thoughts/2007/05/01/reset-reloaded/*

5.3 Coloring the Scroll Bar in IE

Problem

You want to adjust the color of the scroll bar on a browser's viewport, or the window on the Internet Explorer browser.

Solution

Use the properties that manipulate scroll bar colors in browsers:

```
body,html {
  scrollbar-face-color: #99ccff;
  scrollbar-shadow-color: #ccccff;
  scrollbar-highlight-color: #ccccff;
  scrollbar-3dlight-color: #99ccff;
  scrollbar-darkshadow-color: #ccccff;
```

```
    scrollbar-track-color: #ccccff;
    scrollbar-arrow-color: #000033;
}
```

> Because these properties aren't part of the W3C recommendations for
> CSS, browser vendors don't have to put in support for these properties.
> This Solution works only on the KDE Konqueror browser and on In-
> ternet Explorer. Other browsers will simply skip over the rules. How-
> ever, these rules won't be validated by services such as *http://jigsaw.w3*
> *.org/css-validator/#validate_by_uri*.

Discussion

Although you might think of a scroll bar as a simple tool, it's actually composed of
several widgets that create a controllable 3D object. Figure 5-7 spotlights the different
properties of a scroll bar. As you can see, to create a truly different color scheme for a
scroll bar, you must alter the value of seven properties.
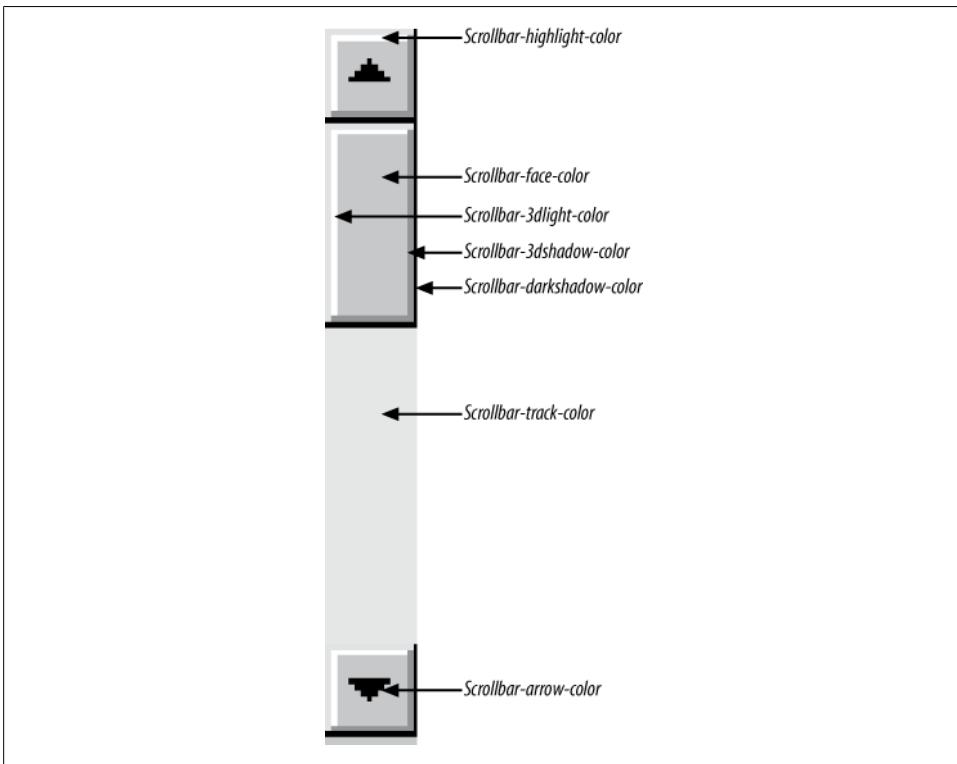
*Figure 5-7. The parts of a scroll bar that can be affected by proprietary CSS for Internet Explorer for
Windows*

In addition to adjusting the scroll bar of the browser viewport, you also can adjust the colors of the scroll bar in the `textarea` for a web form, `framesets`, `iframes`, and generally anything with a scroll bar:

```
.highlight {
 scrollbar-face-color: #99ccff;
 scrollbar-shadow-color: #ccccff;
 scrollbar-highlight-color: #ccccff;
 scrollbar-3dlight-color: #99ccff;
 scrollbar-darkshadow-color: #ccccff;
 scrollbar-track-color: #ccccff;
 scrollbar-arrow-color: #000033;
}
<form>
 <textarea class="highlight"></textarea>
</form>
```

Dealing with quirks mode

When rendering a page that doesn't contain a valid DOCTYPE, versions of IE before IE8 experience what is known as *quirks* mode (see Recipe 1.3) and look for the scroll bar properties in the `body` selector.

When the page contains a valid DOCTYPE in older versions of IE, IE8 is in standards mode and it obeys the `html` selector.

So, just in case the web document's DOCTYPE might change, it's best to ensure that the `body` and `html` selectors are grouped and applied in one CSS rule:

```
html .highlight, body .highlight {
 scrollbar-face-color: #99ccff;
 scrollbar-shadow-color: #ccccff;
 scrollbar-highlight-color: #ccccff;
 scrollbar-3dlight-color: #99ccff;
 scrollbar-darkshadow-color: #ccccff;
 scrollbar-track-color: #ccccff;
 scrollbar-arrow-color: #000033;
}
```

Color scroll bars in IE8

For Internet Explorer 8, Microsoft converted its proprietary CSS properties as vendor extensions with the `-ms-` prefix.

> IE8 allows for masking the browser as Internet Explorer 7, keeping its own unique blend of rendering issues alive. When using the Emulate IE7 mode, the original set of proprietary scroll bar properties are understood by the browser.

So, to ensure cross-browser support within IE versions, make sure you include both sets of properties:

```
.highlight {
scrollbar-face-color: #99ccff;
scrollbar-shadow-color: #ccccff;
scrollbar-highlight-color: #ccccff;
scrollbar-3dlight-color: #99ccff;
scrollbar-darkshadow-color: #ccccff;
scrollbar-track-color: #ccccff;
scrollbar-arrow-color: #000033;
-ms-scrollbar-face-color: #99ccff;
-ms-scrollbar-shadow-color: #ccccff;
-ms-scrollbar-highlight-color: #ccccff;
-ms-scrollbar-3dlight-color: #99ccff;
-ms-scrollbar-darkshadow-color: #ccccff;
-ms-scrollbar-track-color: #ccccff;
-ms-scrollbar-arrow-color: #000033;
}
```

Use conditional comments (see Recipe 12.7) to pinpoint CSS rules to a specific version of IE.

> The Safari browser also has proprietary CSS rules for colorizing a scroll bar. For more information, see *http://webkit.org/blog/363/styling-scroll bars/*.

See Also

Internet Explorer-specific Functionality at *http://msdn.microsoft.com/en-us/library/cc304082(VS.85,loband).aspx#ie_specific*; the "IE Colour scrollbar maker" at *http://www.sean.co.uk/a/webdesign/color_scrollbar_maker_ie.shtm*

5.4 Techniques for Centering Elements on a Web Page

Problem

You want to center parts of a web page, as in Figure 5-8.

Solution

To center text in a block-level element, use the `text-align` property:

```
h1, h2, h3 {
text-align: center;
}
```
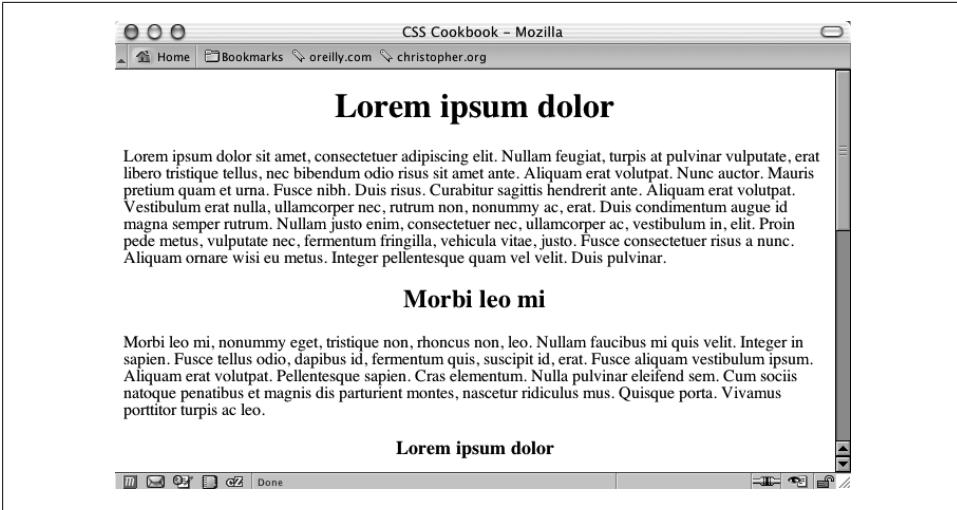
Figure 5-8. The headline text centered

Discussion

By using `text-align`, you can center text inside block-level elements. However, in this example, the heading takes up the entire width of the `body` element, and if you don't apply a background color to the element, you probably won't even notice this is happening. The gray background color in Figure 5-9 shows the actual width of the centered elements.
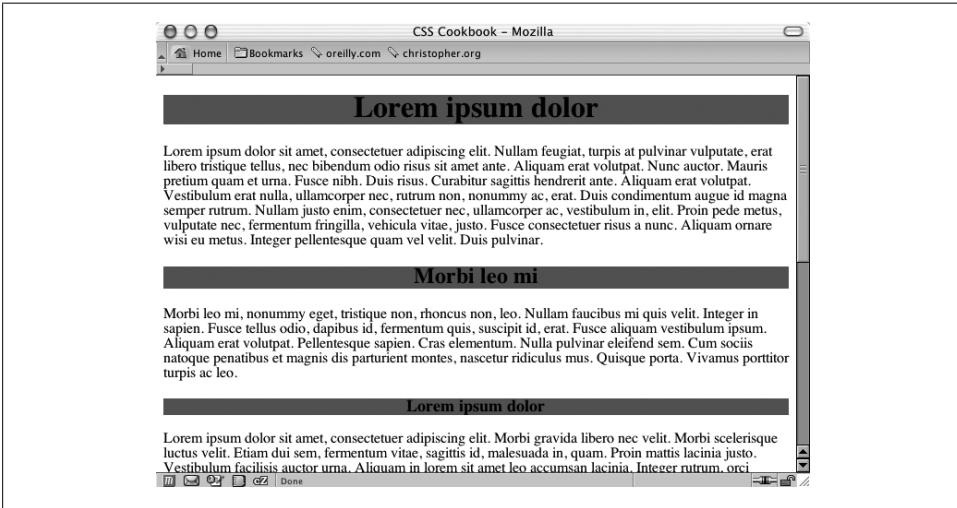
Figure 5-9. The actual width of the elements shown by the gray background color

An alternative approach is to use margins to center text within its container:

```
h1, h2, h3 {
 margin-left: auto;
 margin-right: auto;
 width: 300px;
}
```

When you set the `margin-left` and `margin-right` properties to `auto` (along with a value for the `width` property), you center the element inside its parent element.

Tables

To center a table, set a `class` attribute with a value:

```
<div class="center">
 <table>
  <tr>
   <td>This is the first cell</td>
   <td>This is the second cell</td>
  </tr>
  <tr>
   <td>This is the third cell, it's under the first cell</td>
   <td>This is the fourth cell, it's under the second cell.</td>
  </tr>
 </table>
</div>
```

Then write the following CSS rule:

```
.center {
 width: 50%;
 margin-left: auto;
 margin-right: auto;
}
```

Images

If you want to center an image, wrap a `div` element around the `img` element first. This technique is required because an `img` element, like `em` and `strong`, is inline. It rests in the flow of the web page instead of marking off space like the `p` or `blockquote` block-level elements do. The markup looks like this:

```
<div class="flagicon"><img src="flag.gif" alt="Flag" width="160"
height="60" /></div>
```

And the CSS rule looks like this:

```
.flagicon {
 text-align: center;
}
```

To center elements with fixed widths, such as images, first set the value of the parent's `padding-left` property to `50%`.

Then determine half of the width of the element you are centering and set it as a negative value in the margin-left property. That prevents the element's left side from resting on the 50% line caused by its padding and makes it slide into the middle of the page.

The markup for an image in a web page using this technique looks something like this:

```
<img src="wolf.jpg" width="256" height="192" alt="Photo of wolf.">
```

The CSS rule to produce the result shown in Figure 5-10 looks like this:

```
body {
 padding-left: 50%;
}
img {
 /* equal to the negative of half its width */
 margin-left: -138px;
}
```


Figure 5-10. The image centered without the deprecated center element

Vertical centering

With the element centered horizontally, you can take this technique one step further and center the image (or any other element) vertically as well.

The difference with this method is that it uses the position property to make this work. The markup is the same as that used for the image element in the previous example, but this time the CSS rule is for just one selector (see Figure 5-11):

```
img {
 position: absolute;
 top: 50%;
 left: 50%;
 margin-top: -96px;
 margin-left: -138px;
 height: 192px;
 width: 256px;
}
```

Figure 5-11. The image centered horizontally and vertically on the web page

With absolute positioning (see Recipe 2.23), you take the element out of the normal flow of the document and place it wherever you want.

If you want to center both text and an image (or other images) instead of just one image, enclose all of the content with a `div` element:

```
<div id="centerFrame">
 <p>Epsum factorial non deposit quid pro quo hic escorol. Olypian
quarrels et gorilla congolium sic ad nauseum. Souvlaki ignitus
carborundum e pluribus unum. Defacto lingo est igpay atinlay.</p>
 <img src="wolf.jpg" width="256" height="192" alt="Photo of
wolf." />
</div>
```

Then in the CSS rule, remove the `height` property and adjust the negative value of the top margin to compensate for the additional elements on the page:

```
#centerFrame  {
 position: absolute;
 top: 50%;
 left: 50%;
 /* adjust negative value until content is centered */
 margin-top: -150px;
 margin-left: -138px;
 width: 256px;
}
```

Keep the amount of content that you want centered short. This Solution is going to only roughly center the text and the images because the text will render at different heights on different computers.

If you have numerous images and large amounts of HTML text, users with low resolutions will have to scroll the page to see your centered content.

See Also

Chapter 10 for information on multicolumn layouts, which deal with the position of elements in a web page

5.5 Placing a Page Border

Problem

You want to place a visual frame or border around a web page, as shown in Figure 5-12.

Figure 5-12. A framed web page

Solution

Use the `border` property on the `body` element:

```
body {
 margin: 0;
 padding: 1.5em;
 border: 50px #666 ridge;
}
```

Discussion

The `border` property is a shorthand property, in that it enables you to set the width, color, and style of the border around an element in one step instead of three.

If you didn't use this shorthand property in the preceding Solution, you would have to replace the line that reads `border: 50px #666 ridge;` with the following three lines:

```
border-width: 50px;
border-color: #666;
border-style: ridge;
```

You can create a framing effect with other styles as well, such as dotted, dashed, solid, double, groove, inset, and outset (see Figure 5-13).

Figure 5-13. The available border styles in CSS

Note that the groove style is the inverse of the shades of shadow as seen in the Solution, which uses the `ridge` value.

The only browser incompatibilities to worry about are that in Internet Explorer for Windows the dotted style appears as aliased circles, whereas in Firefox, Opera, and Safari the dotted style appears as blocks.

Borders on images

You also can place a stylized border on images (see Recipe 4.2). Instead of having a default solid line, try experimenting in your designs with grooved or double borders, as shown in Figure 5-14:

```
img.left {
 float: left;
 margin-right: 7px;
 margin-bottom: 3px;
 border: 4px double #666;
}
```
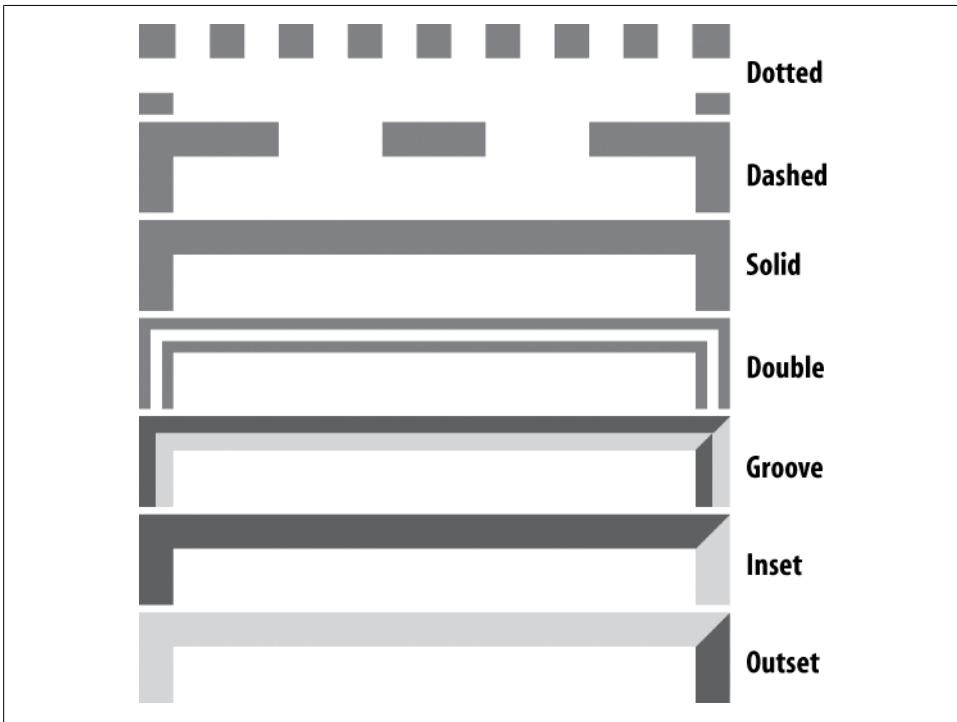

Figure 5-14. A double border around an image

See Also

Recipe 3.21 for creating pull quotes with different border styles

5.6 Placing a Border Around the Browser's Viewport

Problem

You want to place a border around the viewport of the browser.

Solution

First set up a series of eight `div` elements that are placed beneath the content of the web page, but right before the closing `body` element:

```
<div id="top"></div>
<div id="topright"></div>
<div id="right"></div>
<div id="bottomright"></div>
<div id="bottom"></div>
<div id="bottomleft"></div>
<div id="left"></div>
<div id="topleft"></div>
```

Set the corners of the frame to have the same width and height and set the `position` to `fixed`:

```
#topleft, #topright, #bottomleft, #bottomright {
 height: 24px;
 width: 24px;
 position: fixed;
 display: block;
 z-index: 20;
}
```

Set the borders to a fixed position. Also, set the top and bottom sides to a height of 24 pixels and the left and right sides to a width of 24 pixels:

```
#top, #bottom {
 height: 24px;
 position: fixed;
 left: 0;
 right: 0;
 display: block;
 background-color: #ccff00;
 z-index: 30
}
#left, #right {
 width: 24px;
 position: fixed;
 top: 0;
 bottom: 0;
 display: block;
 background-color: #ccff00;
 z-index: 50;
}
```

Then assign each part to its respective corner and side of the viewport:

```
#top {
 top: 0;
}
#bottom {
 bottom: 0;
}
#left {
 left: 0;
}
#right {
 right: 0;
}
#topleft {
 top: 0;
 left: 0;
}
#topright {
 top: 0;
 right: 0;
}
 #bottomleft {
 bottom: 0;
 left: 0;
}
 #bottomright {
 bottom: 0;
 right: 0;
}
```

Discussion

A character of this recipe's approach is that the border expands to the height of the content within the body element. To have a border or framing device that is visible around the entire viewport at all times, no matter the length of content, use fixed positioning (see Recipe 4.10).

Instead of using background colors for the bars, another technique similar to this one is to use PNGs (or even CSS gradients with opacity as in Recipe 4.16) to set a fade effect. As the user scrolls the browser, the text fades out along the edges of the browser's viewport.

See Also

The CSS2 specification for fixed positioning at *http://www.w3.org/TR/CSS2/visuren .html#fixed-positioning*

5.7 Customizing a Horizontal Rule

Problem

You want to change the look of a horizontal rule from the solid line in Figure 5-15 to something more interesting, such as the graphic in Figure 5-16.
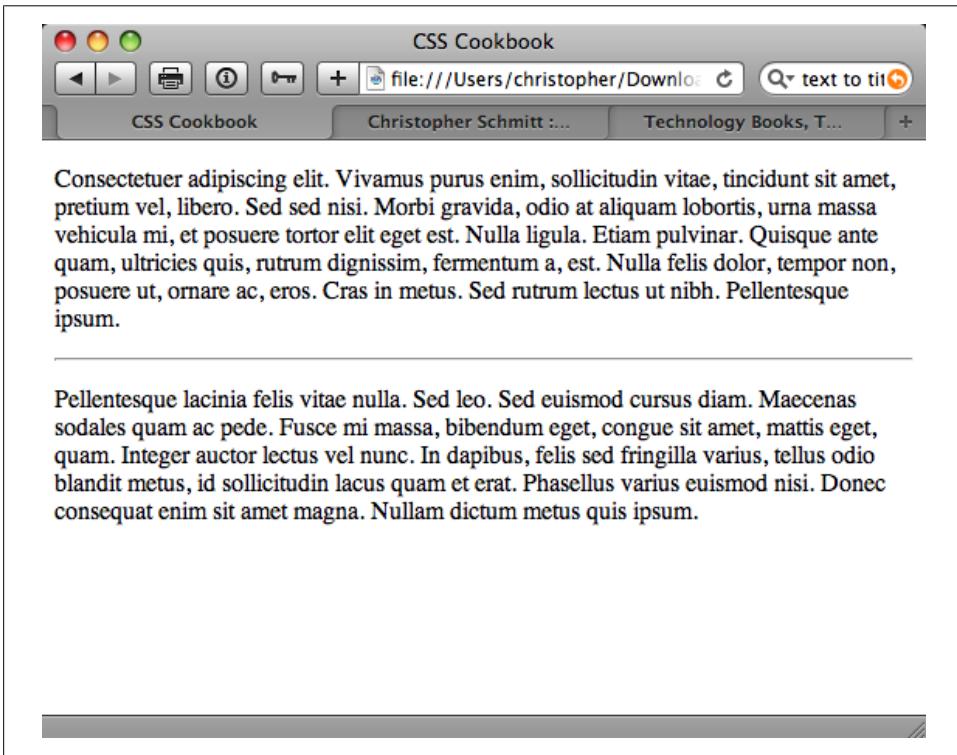
Figure 5-15. The default rendering of a horizontal rule

Solution

Use a mixture of CSS properties on the hr element to obtain the desired effect:

```
<style type="text/css">
hr {
 border: 0;
 height: 43px;
 background-image: url(hr.gif);
 background-position: 50% 0;
 background-repeat: no-repeat;
 margin: .66em 0;
}
</style>
```

```
<!--[if lt IE 8]>
<style type="text/css">
hr {
 display: list-item;
 list-style: url(hr.gif) inside;
 filter: alpha(opacity=0);
 width: 0;
}
</style>
```
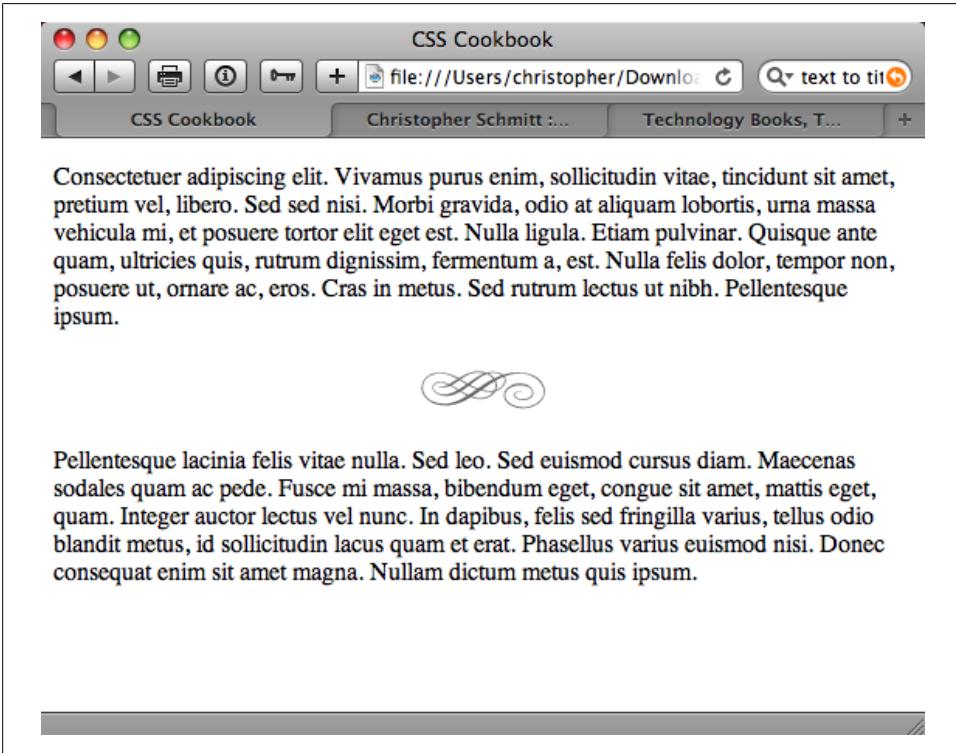

Figure 5-16. A stylized horizontal rule

Discussion

Before HTML 4.0, you could manipulate the presentation of horizontal rules through a set of four attributes: `align`, `width`, `size`, and `noshade`. Since HTML is intended to mark up content and not the look of the content, those values are no longer a part of the HTML specification. (Browser vendors may support the values, but your mileage will vary.) With CSS rules controlling the presentation, you have far greater control over the appearance of horizontal rules.

To create a cross-browser styling of horizontal rules, set the border to zero and then bring in an image through the `background` property. Adjust the margins above and below to taste.

For cross-browser support for older IE browsers, use conditional comments to deliver an alternative method of bringing in the background image:

```
<!--[if lt IE 8]>
<style type="text/css">
hr {
 display: list-item;
 list-style: url(hr.gif) inside;
 filter: alpha(opacity=0);
 width: 0;
}
</style>
<![endif]-->
```

Since older versions of IE cannot insert a background image through the `hr` element, set the `display` property to `list-item`. This allows an image to be brought in through the `list-style` property.

To remove the border of the `hr` element set the opacity to zero using Microsoft's CSS filter.

See Also

The HTML 4.01 specification for `hr` elements at *http://www.w3.org/TR/html401/ present/graphics.html#edef-HR*; an overview of styling an `hr` element at *http://www.so vavsiti.cz/css/hr.html*

5.8 Adding a Lightbox

Problem

You want to overlay images on top of a current web page (as shown in Figure 5-17) without popping a new browser window.

Solution

Download the source code for the lightbox effect from *http://www.huddletogether.com/ projects/lightbox2/#download*.

Along with the Prototype JavaScript Framework Scriptaculous Effects JavaScript libraries, include specialized JavaScript for overlaying images:

```
<title>Mr. McCool's Homepage</title>
<!-- Structure for Lightbox effect -->
<script type="text/javascript" src="prototype.js"></script>
<script type="text/javascript" src="scriptaculous.js?load=effects"></script>
```

```
<!-- Script for Lightbox -->
<script type="text/javascript" src="lightbox.js"></script>
```
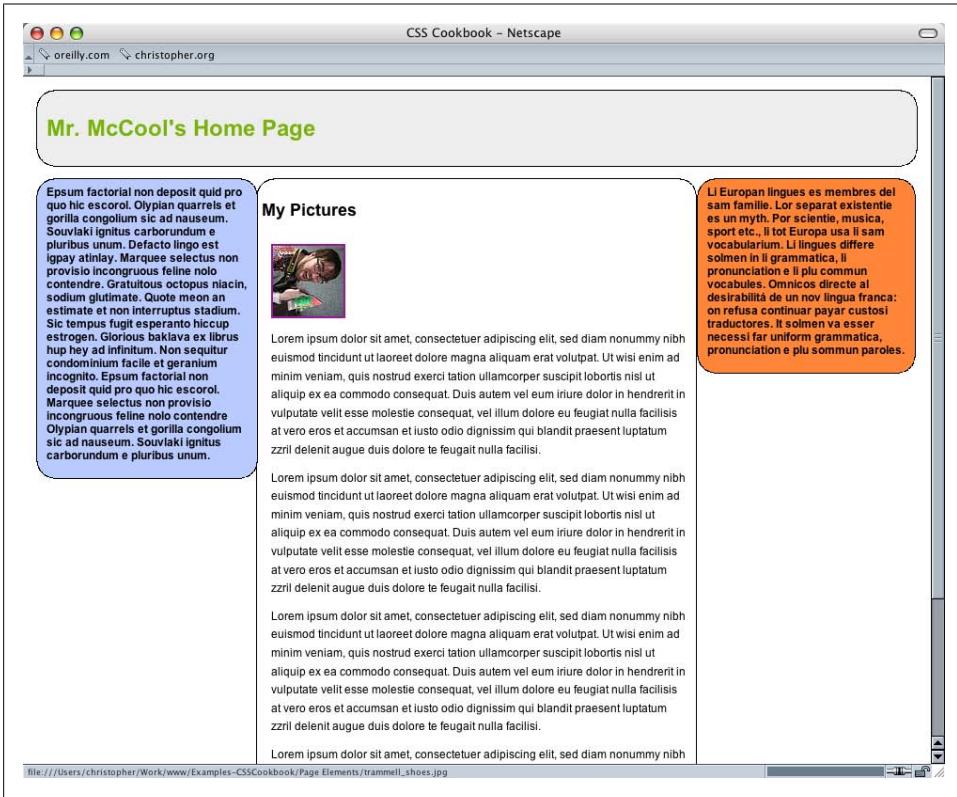

Figure 5-17. The default page

Next, link to the stylesheet that renders the look and feel of the overlay effect:

```
<title>Mr. McCool's Homepage</title>
<script type="text/javascript" src="prototype.js"></script>
<script type="text/javascript" src="scriptaculous.js?load=effects"></script>
<script type="text/javascript" src="lightbox.js"></script>
<link rel="stylesheet" href="lightbox.css" type="text/css" media="screen" />
```

Within the web page content, include a link to an image, making sure to include a rel attribute with a value of lightbox. A common link example would be to wrap a link around a thumbnail image:

```
<a href="trammell_shoes.jpg" rel="lightbox" title="Trammell shows off
his happy shoes."><img src="trammell_shoes_tn.jpg" alt="Mark Trammel
is happy with his shoes." /></a>
```

Clicking on link activates the lightbox effect, as shown in Figure 5-18.
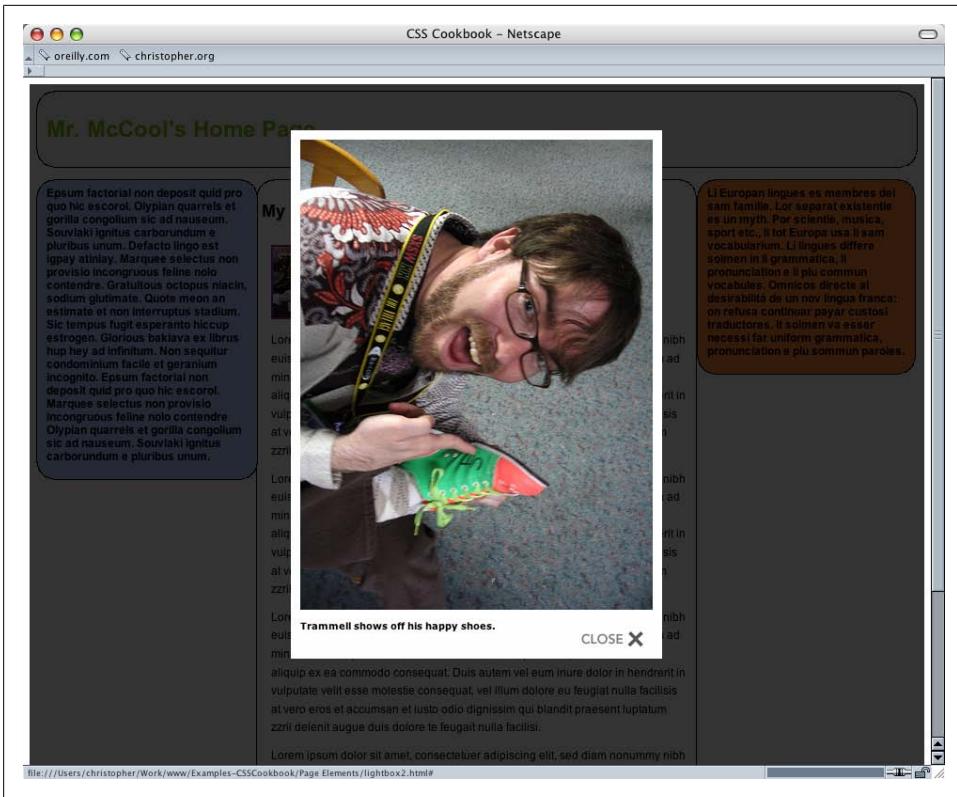
Figure 5-18. The lightbox appearing on top of the page

Discussion

The lightbox effect is built on two core pieces: the Prototype JavaScript Framework and Scriptaculous.

Prototype creates a more object-oriented framework, allowing developers to quickly build web-based applications based on JavaScript. For more information about Prototype, see its official website, *http://www.prototypejs.org/*.

Scriptaculous is a collection of JavaScript libraries. When used in conjunction with Prototype, Scriptaculous allows developers to build dynamic, Asynchronous JavaScript and XML (Ajax) interactions. For more information on Scriptaculous, see *http://script .aculo.us/*.

With the JavaScript foundations in place, web developer Lokesh Dhakar (see *http:// www.lokeshdhakar.com/projects/lightbox2/*) developed a clever image viewer that displays a full-size image without having to leave the web page that displays the thumbnails.

Other JavaScript libraries and gallery plug-ins are also available. For
example, check out jQuery (see Chapter 14) and the galleria image gal-
lery (see *http://code.google.com/p/galleria/*).

Setting up the files

When you download and link the JavaScript files and stylesheet to a web page, make
sure the files are properly linked. For example, if you place the JavaScript and stylesheet
in separate folder locations, make sure the code reflects their locations:

```
<script type="text/javascript" src="/_assets/js/prototype.js"></script>
<script type="text/javascript" src="/_assets/js/scriptaculous.js?load=effects">
</script>
<script type="text/javascript" src="/_assets/js/lightbox.js"></script>
<link rel="stylesheet" href="/_assets/css/lightbox.css"
type="text/css" media="screen" />
```

In the lightbox JavaScript file, also make sure the locations of the images are correct.
If you need to edit the locations of the images, look toward the top of the JavaScript
file for the following lines to modify:

```
var fileLoadingImage = "/_assets/img/loading.gif";
var fileBottomNavCloseImage = "/_assets/img/closelabel.gif";
```

The stylesheet for the lightbox utilizes the background image property three times. Make
sure those images referenced in the properties are also set to the correct locations:

```
#prevLink, #nextLink {
 width: 49%;
 height: 100%;
/* Trick IE into showing hover */
 background: transparent url(/_assets/img/blank.gif) no-repeat;
 display: block;
}
#prevLink:hover, #prevLink:visited:hover {
 background: url(/_assets/img/prevlabel.gif) left 15% no-repeat;
}
#nextLink:hover, #nextLink:visited:hover {
 background: url(/_assets/img/nextlabel.gif) right 15% no-repeat;
}
```

Making a slideshow

In addition to showcasing one image at a time, you can set up the lightbox to display
a slideshow, as shown in Figure 5-19.

To achieve this effect, modify the value of the rel element by using right-angle brackets
after lightbox and inserting a gallery name. In the code example, I used the gallery name
austin because I took the pictures in Austin, Texas:

```
<ul>
 <li><a href="trammell_shoes.jpg" rel="lightbox[austin]"
title="Trammell shows off his happy shoes."><img src="trammell_shoes_tn.jpg"
```

```
alt="Mark Trammel is happy with his shoes." /></a></li>
 <li><a href="molly_andy.jpg" rel="lightbox[austin]" title="Molly and
Andy pose for a shot."><img src="molly_andy_tn.jpg" alt="Molly and Andy
pose for a shot." /></a></li>
 <li><a href="msjen.jpg" rel="lightbox[austin]" title="Ms. Jen at
breakfast."><img src="msjen_tn.jpg" alt="Ms. Jen at breakfast." /></a></li>
 </ul>
```
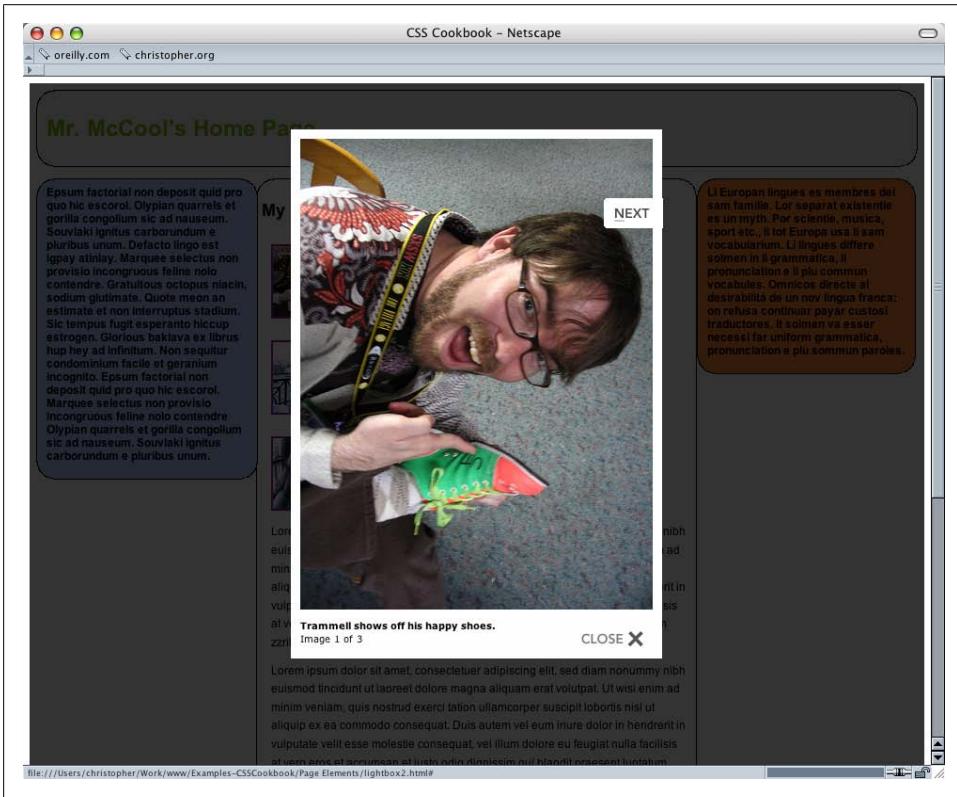

Figure 5-19. The lightbox displaying a slideshow of images

The gallery name needs to be the same for related images to be put into the same slideshow presentation.

Known browser issues

Since the lightbox effect is built on the Prototype Framework, the lightbox effect's support in browsers is based on how many browsers Prototype supports. As of this writing, the following browsers support Prototype:

- Microsoft Internet Explorer for Windows 6 and later
- Firefox 1.0 and later

- Safari 1.2 and later
- Opera 9.25 and later
- Chrome

The lightbox effect degrades gracefully. If a visitor's browser does not support the lightbox effect, the browser will follow the value of the href attribute:

```
<a href="trammell_shoes.jpg" rel="lightbox" title="Trammell shows off
his happy shoes."><img src="trammell_shoes_tn.jpg" alt="Mark Trammel
is happy with his shoes." /></a>
```

In this example, the browser pulls up the file *trammell_shoes.jpg*.

See Also

The article "Learn 3 Excellent JavaScript Libraries at Once" at *http://net.tutsplus.com/ tutorials/javascript-ajax/learn-3-excellent-javascript-libraries-at-once/*

5.9 Changing the Opacity on Elements

Problem

You want to change the opacity or transparency of an element.

Solution

There is an opacity property within CSS that's fairly straightforward to implement (as shown in Figure 5-20):

```
#number4 {
  opacity: .4; /* .4 = 40% transparency */
  filter: alpha(opacity=40);  /* 40 = 40% transparency */
}
```

Discussion

The value of .4 for the opacity property means the element is 40% opaque. A value of 0 means the element is invisible, whereas a value of 1 means there is no transparency.

The proprietary property for Internet Explorer, filter, needs to be set with a value that's equal to the percentage of the transparency. The value of opacity for an alpha filter ranges between 0 and 100. A value of 0 means the element is invisible and a value of 100 means there is no transparency.

> Opacity changes everything contained in the block-level element, whereas setting the opacity with RGBA (see Recipe 5.10) changes the opacity of the element itself.
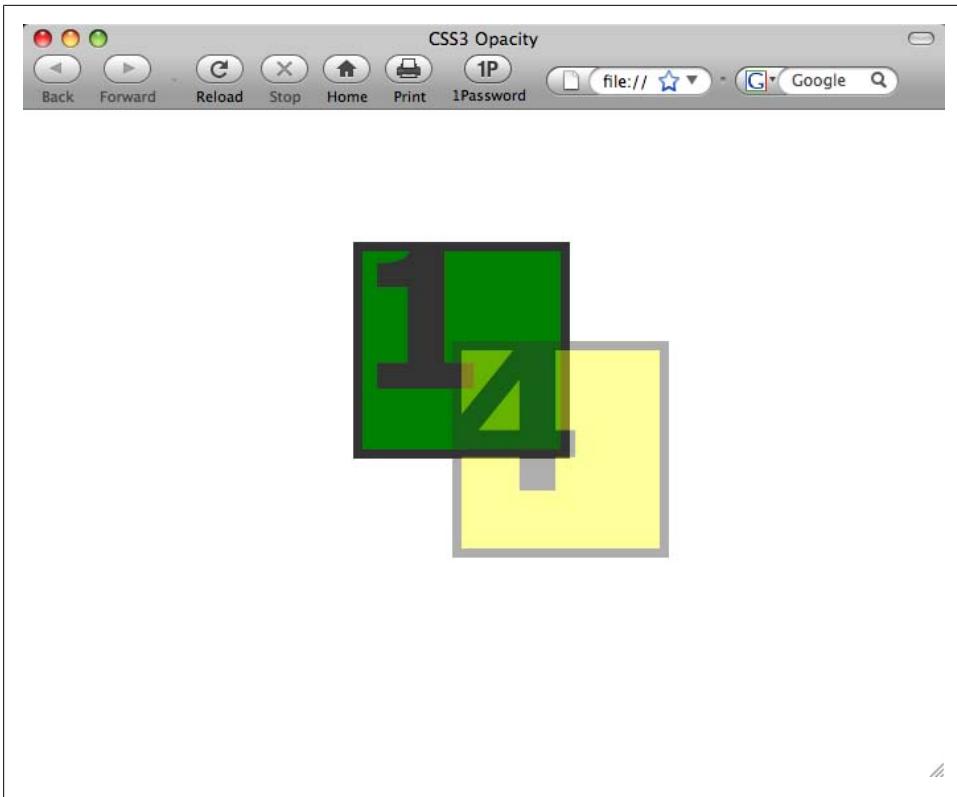
Figure 5-20. Implementing transparency on the number 4 and the box

Browser support

The opacity filter is currently supported in Firefox 1.5 and later, Opera 9 and later, Safari 1.2 and later, and Chrome.

Internet Explorer for Windows 5.5 and later requires the use of its own alpha filter for the effect to be cross-browser compatible.

Since the `filter` property is proprietary, the CSS rule is invalid and the stylesheet it rests in will not validate. A "workaround" is to move IE-specific style rules and apply those rules to only Internet Explorer with conditional comments.

> A drawback to using the opacity filter is that the value is inherited. If a parent element is set to be 10% transparent, the child elements' transparency is also going to be 10%. Watch out for legibility issues within the web page.

See Also

The CSS3 specification for the `opacity` property at *http://www.w3.org/TR/css3-color/#transparency*; Recipe 5.10 for setting the opacity of an element's background color; Recipe 4.14 for setting the browser to render images

5.10 Adjusting the Opacity of Background Colors

Problem

You want to set the opacity of an element's background color.

Solution

Set the transparency of an element's background color using the RGBA value, as shown in Figure 5-21:

```
#number4 {
 background-color: rgba(255, 255, 0, .4);
}
```
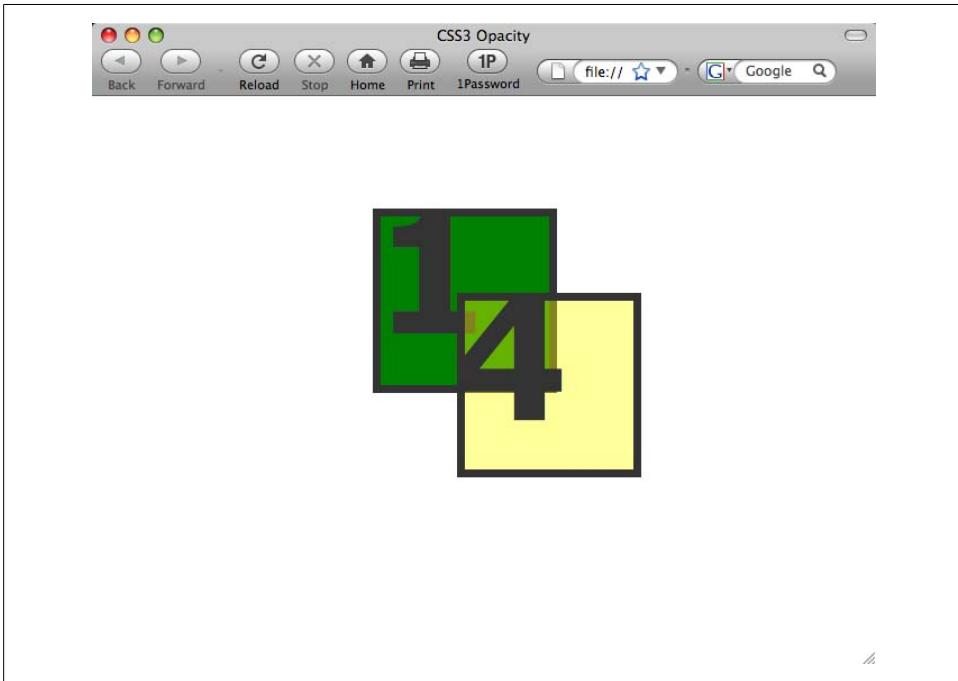

Figure 5-21. A transparent background color

Discussion

Firefox 3 and later, Opera 10 and later, and Safari support RGBA for setting the background color along with a transparent value. When working in cross-browser development, set the `background-color` property first with traditional color coding (RGB, hexadecimal, etc.), and then use another `background-color` property beneath it with a value set in RGBA:

```
#number4 {
  background-color: rgb(255, 255, 0);
  background-color: rgba(255, 255, 0, .4);
}
```

This allows browsers such as Internet Explorer and Firefox 2 to at least render the background color, while Firefox 3, Opera 10 and later, and Safari users see the transparency. Another tactic is to not use color values, but instead use a small, tiled PNG image processed through a digital imaging program such as Adobe Photoshop or Adobe Fireworks set through the `background-image` property. For more information on this technique, see Recipe 4.5.

Supporting Internet Explorer

Through the use of the gradient `filter` property available in Internet Explorer 5.5 and later, it's possible to create transparency on a background color.

The first step is to convert the RGB value of the color to hexadecimal. In this example, `rgb(255,255,0)` converts to #FFFF00.

Next, convert the alpha transparency value to a hexadecimal string (see Table 5-1). In this example, the value is 66.

Table 5-1. Alpha conversion table

Alpha value	Hexadecimal value
0	00
0.1	1A
0.2	33
0.3	4D
0.4	66
0.5	7F
0.6	99
0.7	B3
0.8	CC
0.9	E5
1	FF

Then assemble the hexadecimal value for transparency and color together in one string, starting with the transparency: `#66FFFF00`.

Create a separate CSS rule for the element, setting the color of the background to a value of `transparent`:

```
#number4 {
  background-color: transparent;
}
```

Then, using the `filter` gradient property use the transparency and color hexadecimal string:

```
#number4 {
  background-color: transparent;
  filter:progid:DXImageTransform.Microsoft.gradient(startColorstr=#66FFFF00,
  endColorstr=#66FFFF00);
}
```

Since this is a gradient, you could assign a color change from one value to another. However, you have found a new use for this proprietary filter. With both the starting and ending colors remaining the same along with the transparency value, a cross-browser transparent color is achieved.

Next, add the `zoom` property set to a value of `1` to instruct IE to render the effect or to show that the element "hasLayout" (as shown in Figure 5-22):

```
#number4 {
  background-color: transparent;
  filter:progid:DXImageTransform.Microsoft.gradient(startColorstr=#66FFFF00,
  endColorstr=#66FFFF00);
  zoom: 1;
}
```

> The concept of `hasLayout` is unique to versions of Internet Explorer 7 and earlier. Some elements behave differently depending on whether they have "layout."
>
> To fix these issues, the property is triggered through some CSS selectors, one of them being the `zoom` property. The use of `zoom` to enact `hasLayout` is unique to IE and is promptly ignored by other browsers. For some CSS solutions, you will find `zoom` set to a value of `1` only to get previous versions of IE to render elements so that they have "layout."
>
> For more information on `hasLayout`, see http://msdn.microsoft.com/en-us/library/bb250481(VS.85,loband).aspx.

With this being a CSS rule using a proprietary rule, we can wrap the code with a conditional comment so that only IE browsers process it:

```
<!--[if IE]>
<style type="text/css">
#number4 {
 background-color: transparent;
 filter:progid:DXImageTransform.Microsoft.gradient(startColorstr=#66FFFF00,
 endColorstr=#66FFFF00);
 zoom: 1;
}
</style>
<![endif]-->
```
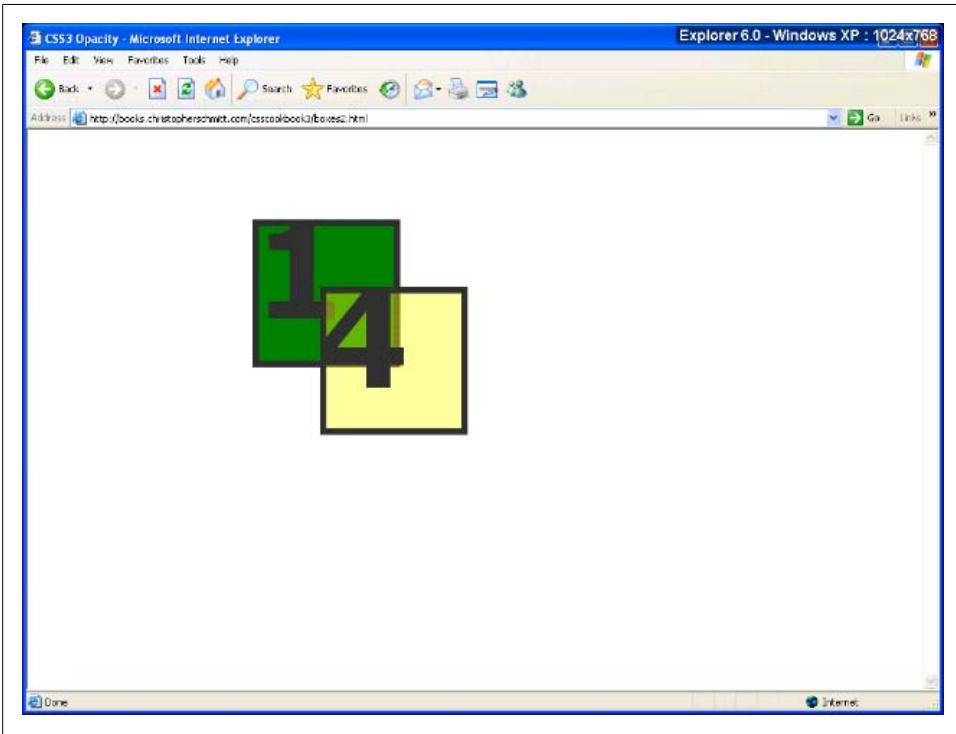

Figure 5-22. Background transparency in IE6

See Also

MSDN's specification on the gradient filter at *http://msdn.microsoft.com/en-us/library/ ms532997(VS.85).aspx*; the online RGB-color-to-hexadecimal-string converter at *http: //www.javascripter.net/faq/rgbtohex.htm*; Recipe 4.14 for setting the browser to render images; Recipe 5.9 for changing the opacity or transparency of an element

Lists

6.0 Introduction

From a wife handing her husband a grocery list as he steps out the door to a music channel presenting its top 100 worst songs of all time, lists help people stay focused and organized.

In web design, it's the same case.

HTML lists facilitate the presentation of organized content to your site's visitors by grouping key elements together. Also, HTML lists are appealing in part because of the way they appear on the page.

List items are typically indented and keyed off by a marker, usually by a filled circle for an unordered list or numbers for an ordered list (see Figure 6-1).

With a few lines of HTML, a web coder can create a bulleted list on a web page without opening an image editor. With CSS, you can create even more visually compelling lists.

With a few simple CSS rules, however, web developers can tailor the presentation of that same list to complement the design of a web page instead of relying on the stodgy browsers' default styling.

This chapter illustrates how to change the numbering of list items, use your own image for a list marker, create a hanging indent that doesn't use a list marker, and more.

6.1 Changing the Format of a List

Problem

You want to change the default list style—for example, to change the bullet or numbering, as shown in Figure 6-2.
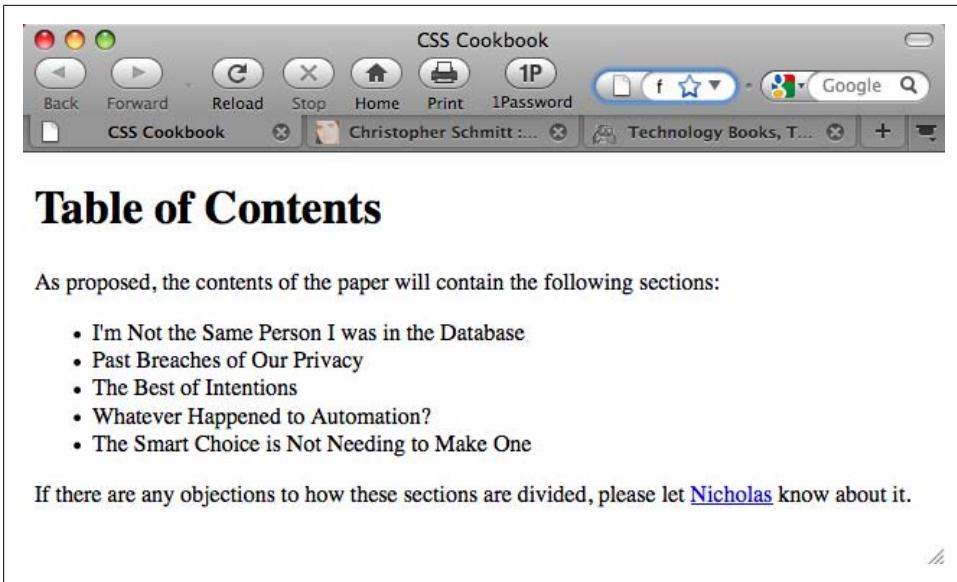
Figure 6-1. The default rendering of a list
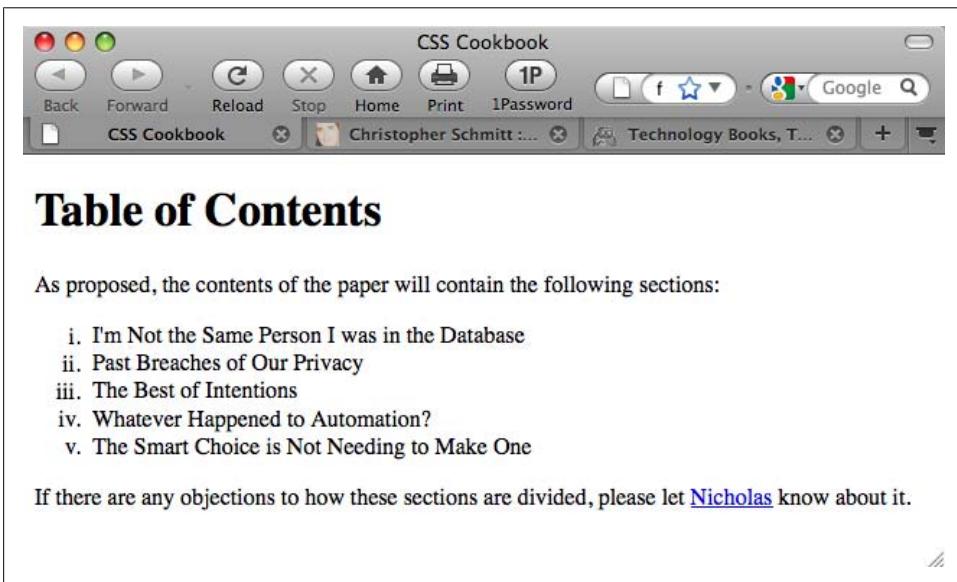
Figure 6-2. The list markers changed to lowercase roman numerals

Solution

Use the `list-style-type` property to change the bullet or type of counter:

```
li {
 list-style-type: lower-roman;
}
```

Discussion

The CSS 2.1 specification offers several styles for numbering a list, as shown in Table 6-1. Browsers typically vary the bullet style from one level of nesting to the next. To stop lists from presenting this traditional system of setting the list marker, change the value of `list-style-type` for each child list.

Table 6-1. Bullet styles

Style/value	Description	Browser support
square	Usually a filled-in square, although the exact representation isn't defined	All major browsers
disc	Usually a filled-in circle, although the exact representation isn't defined	All major browsers
circle	Usually an unfilled circle, although the exact representation isn't defined	All major browsers
decimal	Starts with 1 and continues with 2, 3, 4, etc.	All major browsers
decimal-leading-zero	Starts with 01 and continues with 02, 03, 04, etc.; the number of leading zeros may equal the number of digits used in a list; for example, 0001 might be used for a 5,876–item list	All major browsers
lower-roman	Starts with lowercase roman numerals	All major browsers
upper-roman	Starts with uppercase roman numerals	All major browsers
lower-alpha	Starts with lowercase ASCII letters	All major browsers
upper-alpha	Starts with uppercase ASCII letters	All major browsers
lower-latin	Starts with lowercase ASCII letters	All major browsers
upper-latin	Starts with uppercase ASCII letters	All major browsers
lower-greek	Starts with classical Greek letters, starting with alpha and then beta, gamma, etc.	Safari, Firefox, IE8, Opera
hebrew	Starts counting with traditional Hebrew	Safari, Firefox
hiragana	Starts counting with the Japanese hiragana system	Firefox
katakana	Starts counting with the Japanese traditional katakana system	Firefox, Safari, Chrome
hiragana-iroha	Starts counting with the Japanese hiragana-iroha system	Firefox, Safari, Chrome
none	No marker is displayed	All major browsers

See Also

for using custom images for list markers; Chapter 12, "Lists and Generated Content," in *CSS: The Definitive Guide* by Eric A. Meyer (O'Reilly)

6.2 Changing the Color of a List Bullet

Problem

You want to change the color of a list's bullet without adding a graphic.

Solution

First, insert a set of `span` tags around the text within the list items:

```
<ul>
 <li><span>I'm Not the Same Person I was in the Database</span></li>
 <li><span>Past Breaches of Our Privacy</span></li>
 <li>The Best of Intentions</li>
 <li>Whatever Happened to Automation?</li>
 <li>The Smart Choice is Not Needing to Make One</li>
</ul>
```

Set the color of the list through a general type selector:

```
ul {
 color: #F00;
}
```

Then use a descendant selector to set the color of the text within the list item's `span` element, as shown in Figure 6-3:

```
ul span {
 color: black;
}
```

Discussion

When you set the color for the bullet that is appended to a list item through the `color` property, the text within the list item also inherits color, as seen in the bottom list items in Figure 6-3.

To stop the inheritance, you must insert a `span` element with its own color property.

See Also

for inserting a custom image marker to look into an alternative solution, which avoids extraneous `span` elements to achieve the same effect
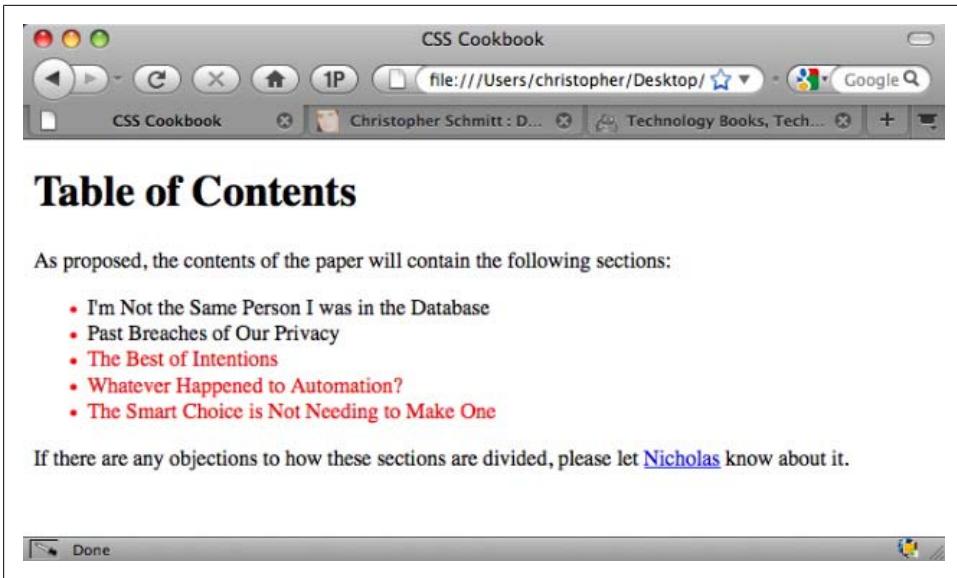
Figure 6-3. Bullets now highlighted, but only the first two list items reset to a different color

6.3 Writing Cross-Browser Indentation in Lists

Problem

Different browsers use different methods to indent lists. You want to specify left margins for your list that will render on all browsers.

Solution

Set both the `margin-left` and `padding-left` properties for the `ul` element:

```
ul {
  margin-left: 40px;
  padding-left: 0px;
}
```

Discussion

Different browsers use different methods to pad or indent a list.

Firefox, Chrome, and Safari browsers indent a list on the *padding*, whereas Internet Explorer and Opera pad a list through the *margin*.

To gain cross-browser effectiveness, you need to set the values for *both* the left margins and the padding for the list. Keep the amount of the indentation in one of the properties. Splitting the amount into two different properties results in inconsistent presentation across the browsers.

Missing markers

If you set the margin and padding to zero while the list is contained by only the `body` element, the browser renders the markers outside the viewport, making them invisible to the user. To work around that problem, set the left margin or left padding of the `ul` to at least 1 em.

See Also

Recipe 6.11 for creating hanging indents; the CSS 2.1 specification for `padding` at *http://www.w3.org/TR/CSS21/box.html#propdef-padding*; the CSS 2.1 specification for `margin` at *http://www.w3.org/TR/CSS21/box.html#propdef-margin*

6.4 Placing Dividers Between List Items

Problem

You want to create list dividers between list items.

Solution

Use the `border` property to create a visual divider:

```
li {
 border-top: 1px solid black;
 padding: .3em 0;
}
```

Then apply a border to the bottom of the `ul` element to create the bottom border, as shown in Figure 6-4:

```
ul {
 margin-left: 40px;
 padding-left: 0px;
 border-bottom: 1px solid black;
 list-style: none;
 width: 36%;
}
```

Discussion

With the box model, padding is inside the border and margin. By setting the length on the padding, you make the border on the bottom of the unordered list encompass both the empty space to the left of the list items and the length of the list.
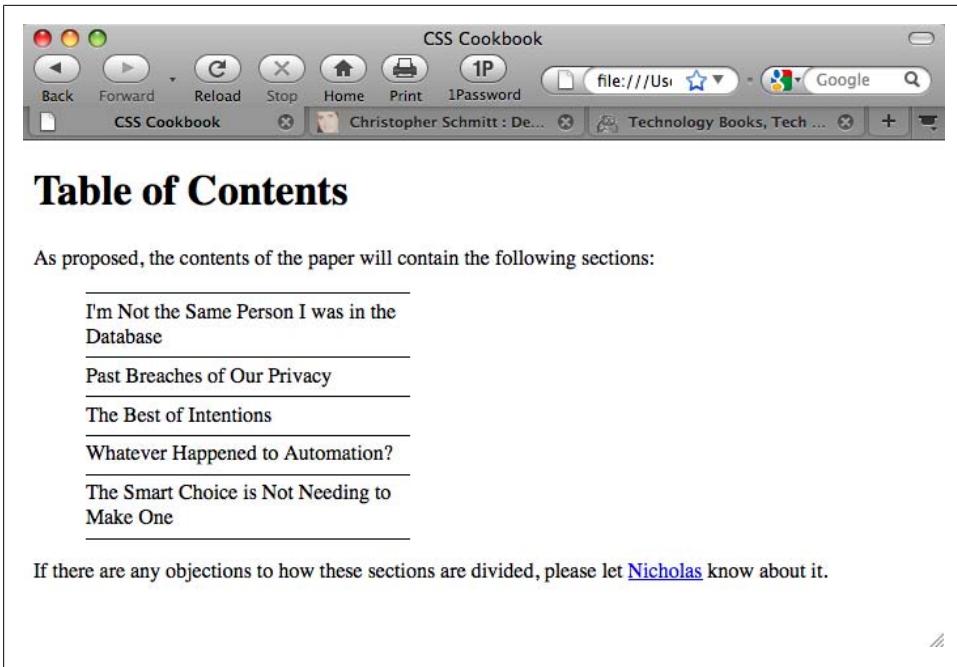
Figure 6-4. Dividers placed between list items

To ensure consistency for the length of the dividers, apply a value only to the `margin-left` property of the unordered list. Otherwise, the length of the border on both the list items and the unordered list will be inconsistent.

For example, if the list items are indented through the `padding-left` property, the bottom border is longer than the border for the individual list items, as shown in Figure 6-5:

```
li {
 border-top: 1px solid black;
 padding: .3em 0;
}
ul {
 margin-left: 0px;
 padding-left: 40px;
 border-bottom: 1px solid black;
 list-style: none;
 width: 36%;
}
```

See Also

Recipe 5.2 for creating cross-browser indents for lists
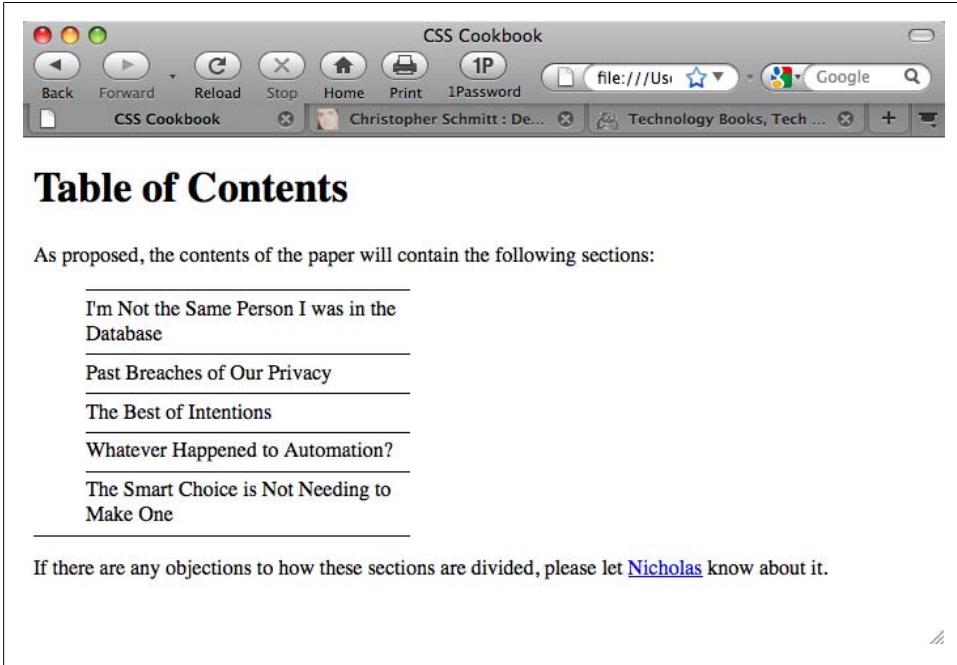
Figure 6-5. The bottom divider, which is longer than the other dividers

6.5 Creating Custom Text Markers for Lists

Problem

You want to use a custom text marker in a list.

Solution

Indent the first line of text and insert the custom text, along with the right-angle quotes acting as pointers, through auto-generated content (see Figure 6-6):

```
ul {
  list-style: none;
  margin: 0;
  padding: 0 0 0 1em;
  text-indent: -1em;
}
li {
  width: 33%;
  padding: 0;
  margin: 0 0 0.25em 0;
}
li:before {
```

```
    content: "\00BB \0020";
  }
```
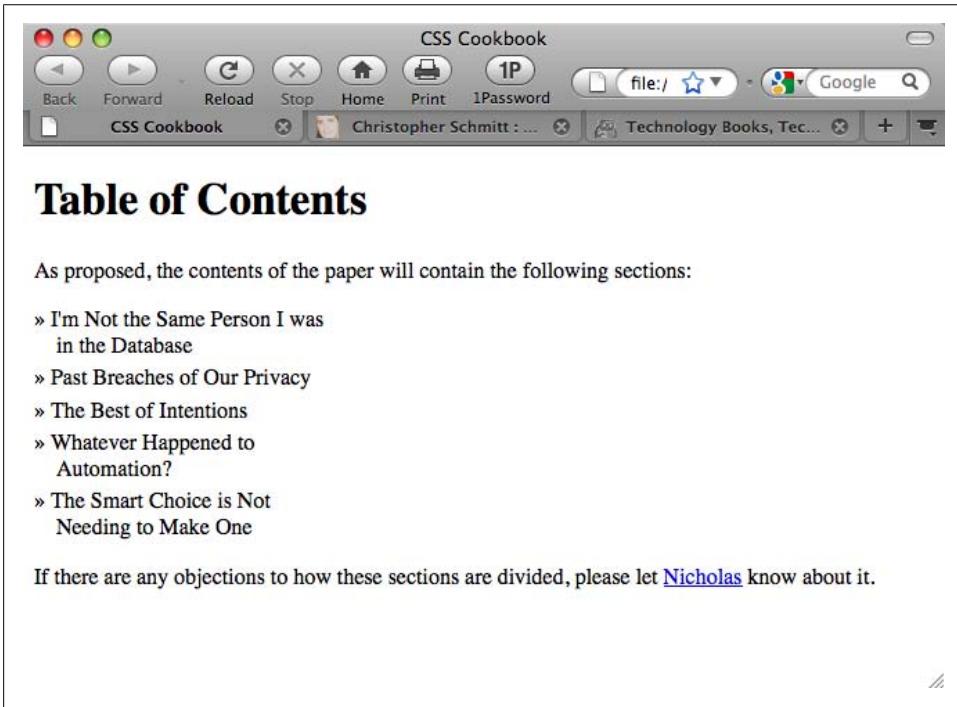

Figure 6-6. Text marker for a list

Discussion

Setting the `list-style` property to a value of `none` turns off the list marker usually associated with a list. Typically, a marker is appended to the left of each list item.

Instead of appending the marker to the list item, the custom text marker will be placed inline with the content of the item. Because the text marker is inside the list item, you need to push the marker out of the list item box. Indenting the first line of the marker with a negative value creates this push.

The negative value for the `text-indent` property moves the first line to the left, whereas a positive value moves the indent to the right:

```
ul {
  list-style: none;
  margin: 0;
  padding: 0 0 0 1em;
  text-indent: -1em;
}
```

The `:before` pseudo-element generates the text marker. You can easily insert the content of simple keyboard characters, like so:

```
li:before {
 content: ">> ";
}
```

However, for embedding special characters, the CSS 2.1 specification calls for Unicode (ISO 10646) values. So, you need to write out the character in its *escaped Unicode hexadecimal equivalent* and not the usual HTML4 entities such as `»` (see *http://www.alanwood.net/demos/ansi.html*).

You escape values in CSS by inserting a backslash before each Unicode hexadecimal value:

```
li:before {
 content: "\00BB \0020";
}
```

At the time of this writing, this solution worked in Firefox, Safari, Chrome, and Opera browsers because they can handle the creation of auto-generated content. Unfortunately, this list omits versions of Internet Explorer for Windows earlier than IE8 as they cannot handle auto-generated content.

To create a cross-browser effect, don't use auto-generated content. Instead, insert the text marker manually before the list item:

```
<ul>
 <li>&#187; I'm not the Same Person I was in the Database</li>
 <li>&#187; Past Breaches of Our Privacy</li>
 <li>&#187; The Best of Intentions</li>
 <li>&#187; Whatever Happened to Automation?</li>
 <li>&#187; The Smart Choice is Not Needing to Make One</li>
</ul>
```

The main drawback with this approach is that you have two markers for every list item (the browser-generated list marker and the manually inserted text marker) if CSS is turned off in the browser and the user sees only the content. Although this isn't a *critical* problem, it adds an unneeded design element to the web page.

See Also

The CSS 2.1 specification for escaping characters at *http://www.w3.org/TR/REC-CSS2/syndata.html#escaped-characters*

6.6 Creating Custom Image Markers for Lists

Problem

You want to use your own graphic for a list marker. For example, Figure 6-7 uses a diamond image.
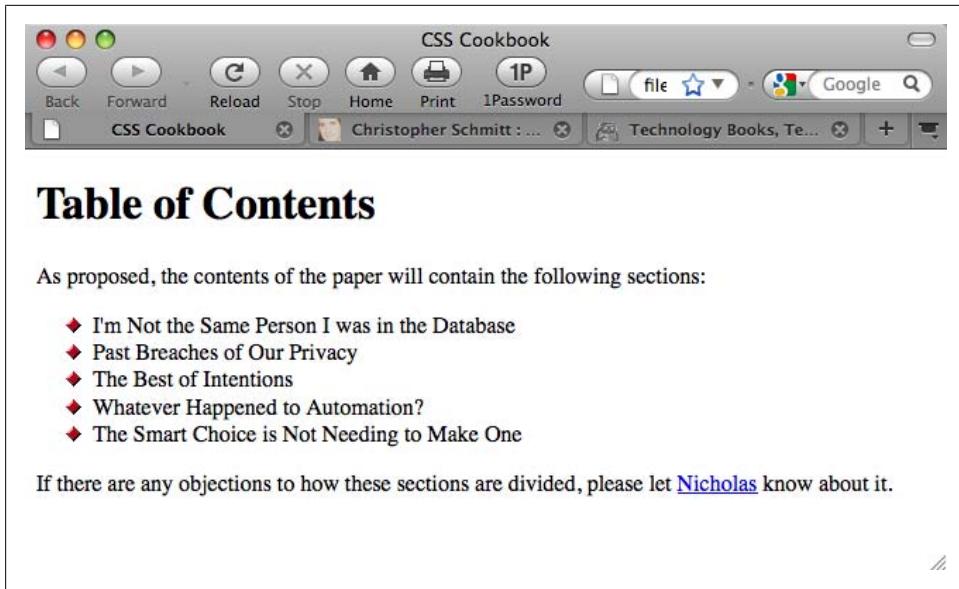
Figure 6-7. Custom-made image markers for a list

Solution

Use the `list-style-image` property to use a graphic for a bullet marker:

```
li {
 list-style-type: disc;
 list-style-image: url(bullet.gif);
}
```

Discussion

Set the location of the image you want to use as a marker as the value of the `list-style-image` property. You can't control the size of the image used as a list marker through CSS, so the image you specify should already be at the size you want to use.

Images that are too large might interfere with the legibility of the list item or the marker might not be displayed entirely in the viewport, as shown in Figure 6-8. When creating custom bullets, make sure they are of the appropriate size to complement the design of your web page.

Stopping inheritance

The value for the image marker is *inherited*, meaning that nested lists pick up the image as the marker, as does the parent.
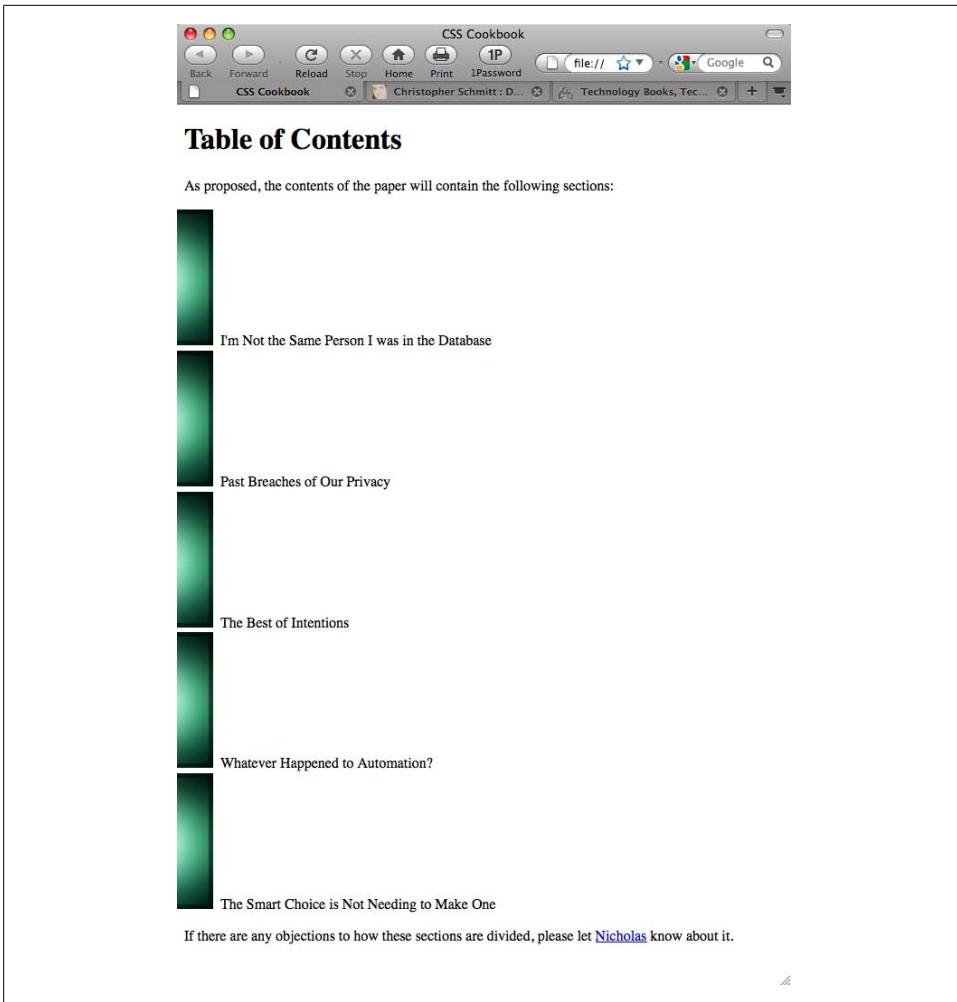
Figure 6-8. A large image used for a marker, which isn't fully displayed

To stop this inheritance, a value of `none` needs to be set for the child lists:

```
ul {
 list-style-type: disc;
 list-style-image: url(bullet.gif);
}
ul ul {list-style-image: none;}
```

Always include the `list-style-type` property to provide a fallback should the image not be usable. In the Solution, the list marker `disc` is used if the image, *bullet.gif*, can't be displayed.

See Also

for creating custom text markers; the CSS 2.1 specification for `list-image-type` at *http://www.w3.org/TR/CSS21/generate.html#propdef-list-style-image*

6.7 Inserting Larger Custom Image Markers for Lists

Problem

You want to use a large custom graphic for a list marker without running into constraints by using the `list-style-image` property.

Solution

First, remove the default list marker through the `list-style` property, as shown in :

```
ul {
  margin-left: 40px;
  padding-left: 0px;
  list-style: none;
}
```
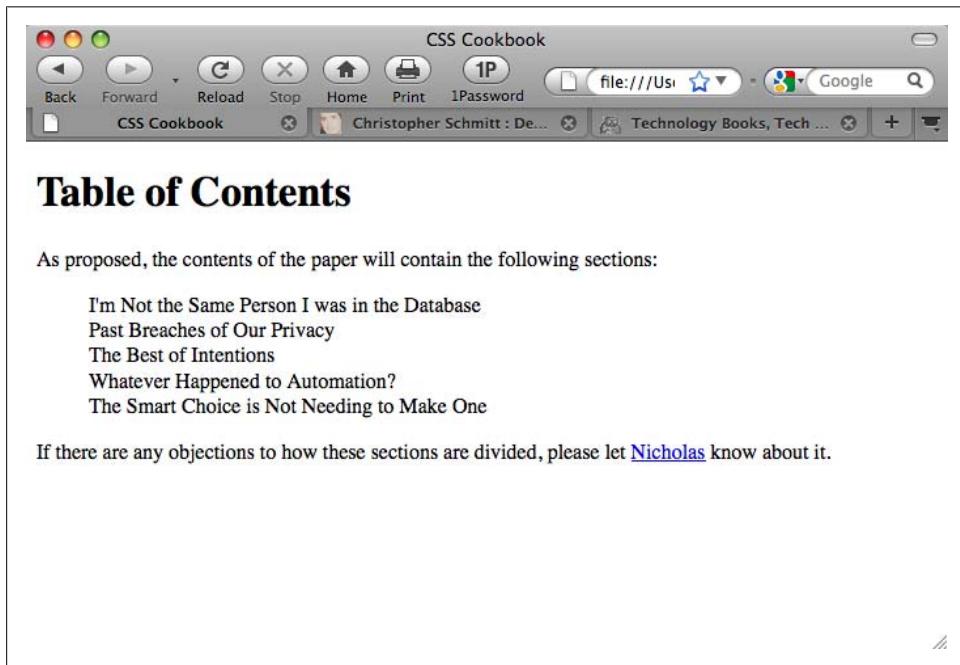

Figure 6-9. Default list markers removed

Apply enough padding on the right side of the list item to allow the new image marker to be placed, as shown in Figure 6-10:

```
ul {
 margin-left: 40px;
 padding-left: 0px;
 list-style: none;
}
li {
 padding: .3em 0 1em 40px;
 font: 1.1em/1.2 Verdana, Arial, Verdana, sans-serif;
}
```
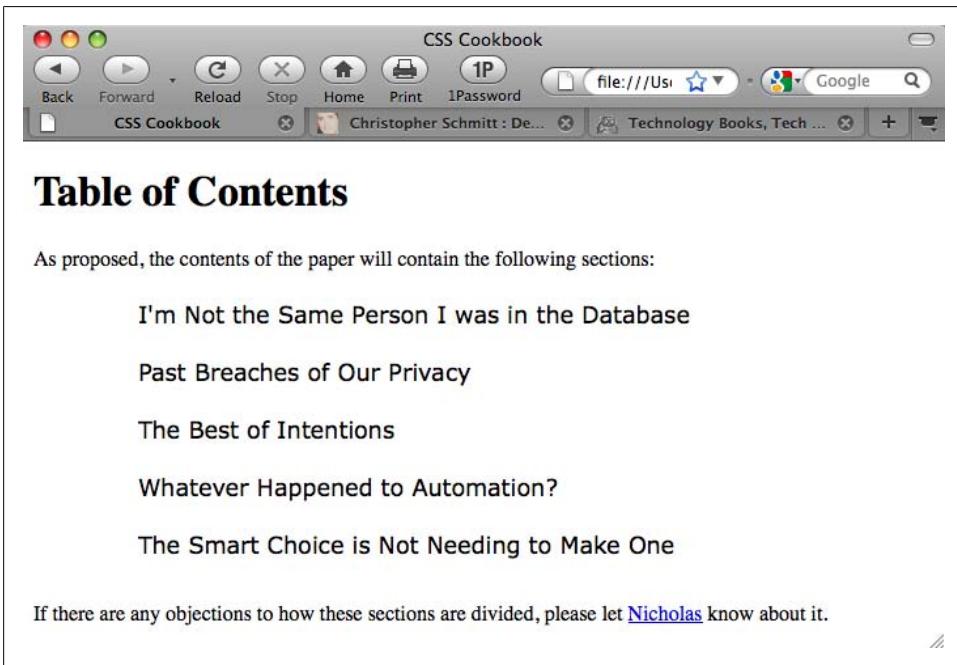

Figure 6-10. Extra padding placed on the left side of the list

Then insert the new custom marker through the `background` property, as shown in Figure 6-11:

```
ul {
 margin-left: 40px;
 padding-left: 0px;
 list-style: none;
}
li {
 padding: .3em 0 1em 40px;
 font: 1.1em/1.2 Verdana, Arial, Verdana, sans-serif;
 background: url(search_32.gif) no-repeat;
}
```
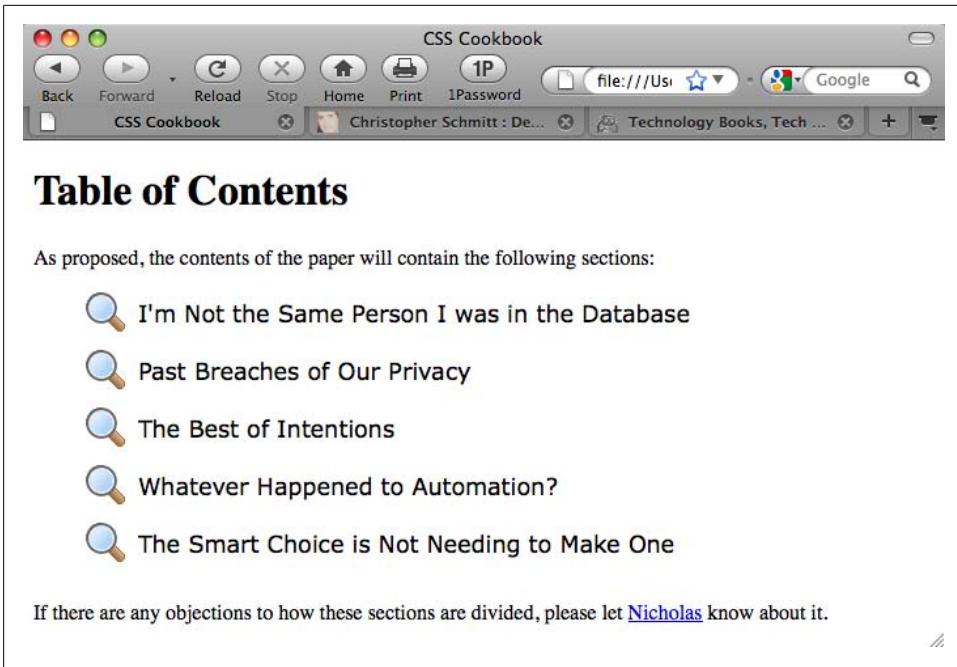
Figure 6-11. A large image used for the markers

Discussion

Using the `background` property to enhance the presentation of a list allows for greater flexibility than using the `list-style-image` property. With this technique, you can use a custom list marker of any size as long as enough padding is set on the left of the list item.

 When using a transparent image, setting the background color might be required.

See Also

Recipe 5.7 for more a complex version of this Solution

6.8 Making a List Presentation Rich with Imagery

Problem

You want to add attention to a list by complementing it with rich imagery.

Solution

Integrate the background images for both the `ul` and `li` elements.

First, create a background image for the unordered list set and an image for the list marker, as shown in Figure 6-12.
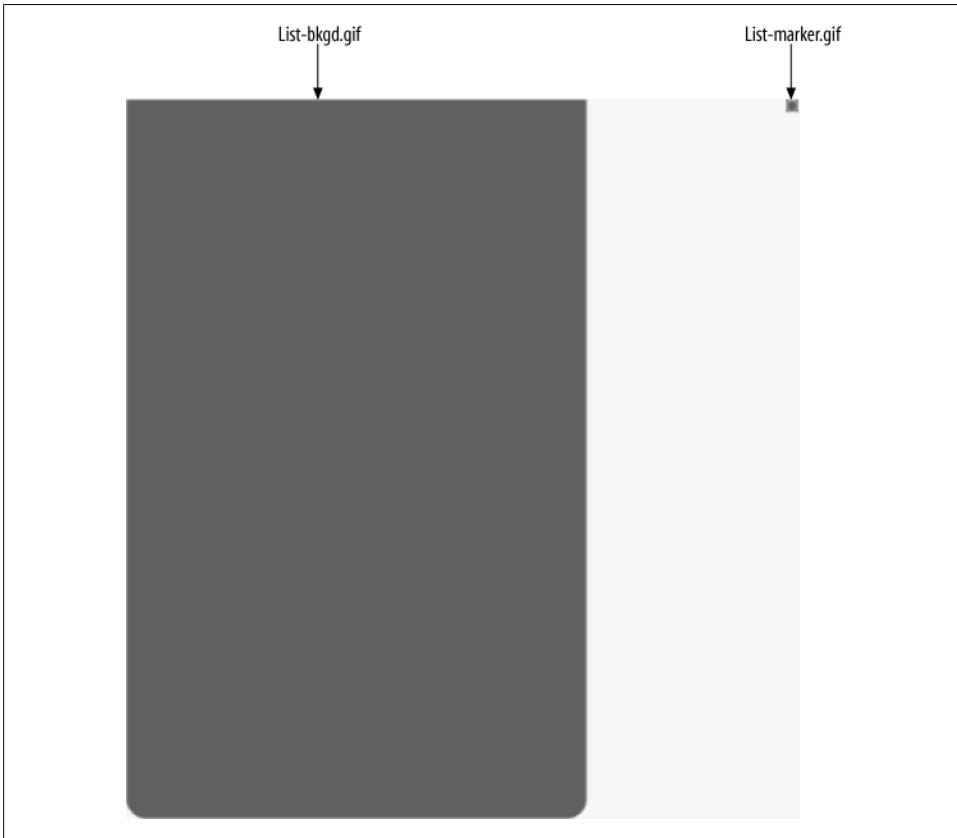
Figure 6-12. The two images used for our custom list presentation

Next, set up the unordered list element to bring in the background image. Also, include the `width` property, setting it to the same width as the background image, as shown in Figure 6-13:

```
ul {
    background: url(list-bkgd.gif) bottom;
    width: 298px;
    list-style: none;
    padding: 0 0 12px;
    margin: 0;
}
```
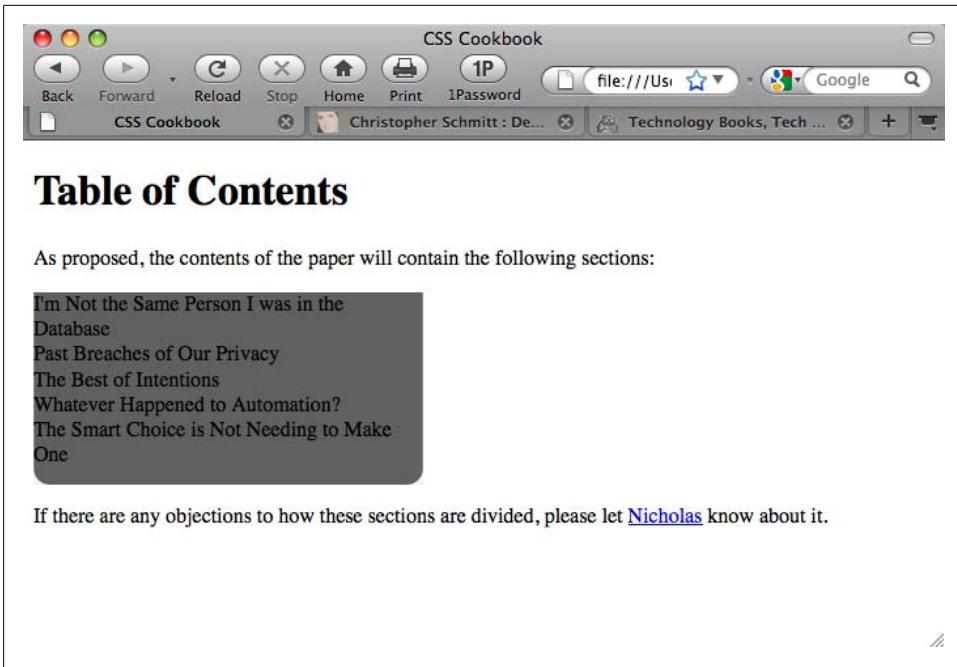
Figure 6-13. Background image for the entire list, set

Next, include the list marker through the list item. Also, place borders at the bottom to act as dividers between the list items, as shown in Figure 6-14:

```
ul {
    background: url(list-bkgd.gif) bottom;
    width: 298px;
    list-style: none;
    padding: 0 0 12px;
    margin: 0;
}
li {
    color: #eee;
    font-family: Verdana, Arial, Verdana, sans-serif;
    padding: 7px 7px 7px 20px;
    border-bottom: 1px solid #888;
    background: url(list-marker.gif) no-repeat 5px .8em;
}
```

Discussion

A number of different techniques come together to achieve this Solution.

The first part of the Solution deals with placing a background image into the ul element. Since the image has a set width and height, make sure to set the width through CSS.
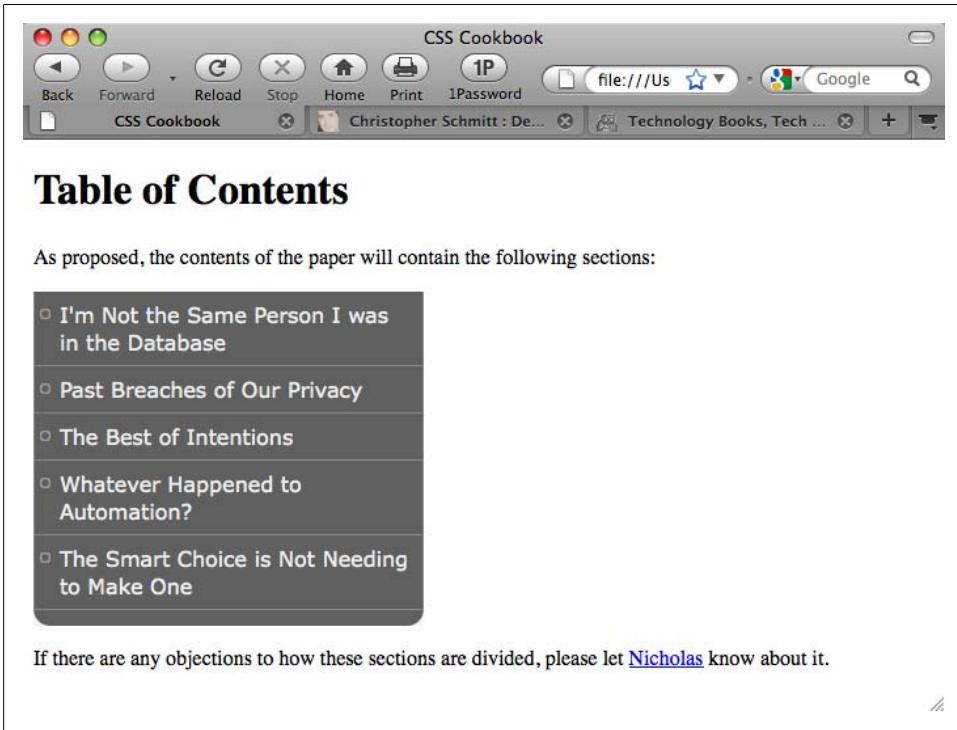
Figure 6-14. Completing the presentation with the styled list items

Regarding the height, many issues keep web developers from setting that property. A user might increase the size of the default text, making the text larger. Also, the style for the list may be used for lists with a high or a low number of items.

To compensate for almost any situation, the background image needs to have a large height.

In this Solution, the background image is set to 465 pixels, which is more than enough space for normal viewing of a handful of items. However, in case someone's browser has set the fonts to a large size, the design solution is still intact, as shown in Figure 6-15.

Since the background image has curved edges on the bottom, a padding of 12 pixels was applied to the bottom so that the list items would not cover it up.

Also, the positioning of the background image was set to `bottom`. This allowed the background image to always display the curves even if the text size expands or the number of list items increases.

Next, the list items involve a couple of techniques. First, dividers are placed between the list items. Unlike in Recipe 5.3, a divider isn't needed on the bottom of the `ul` element.
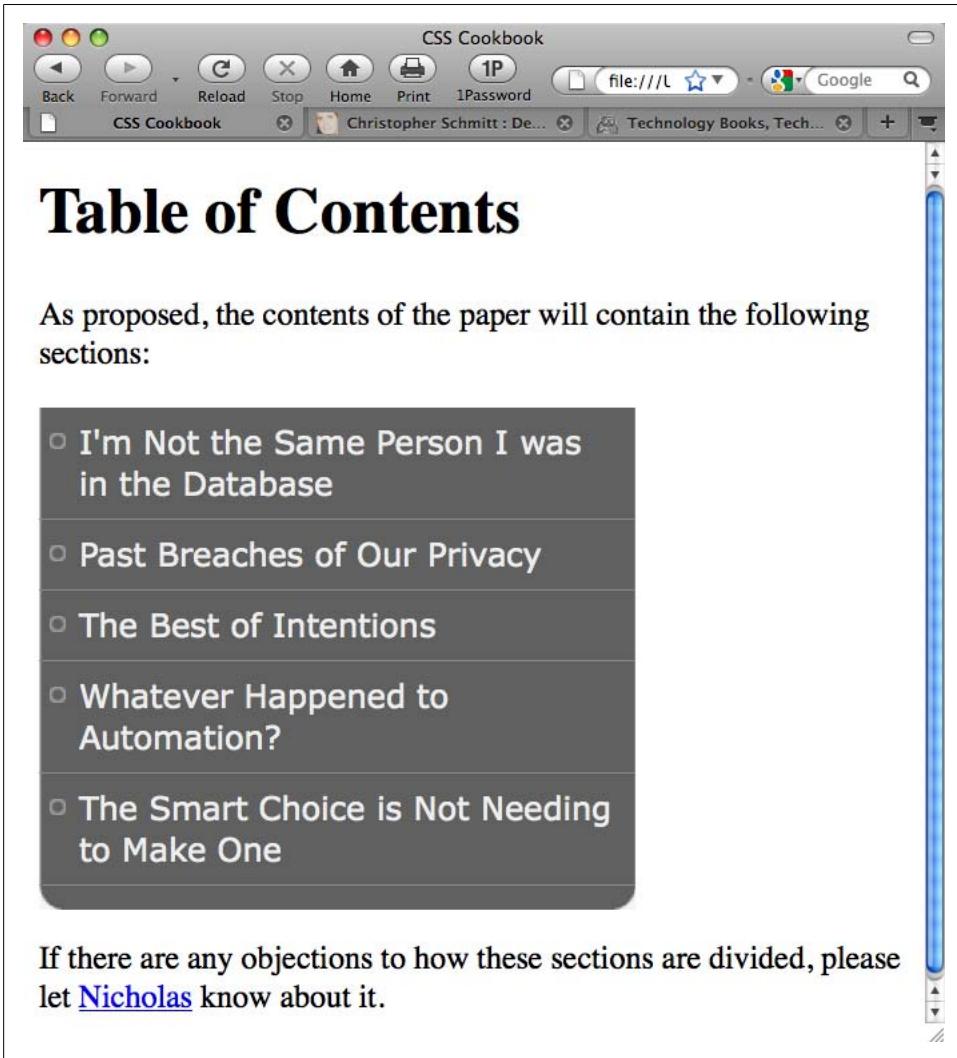
Figure 6-15. A design that remains intact, even as the text size increases

Second, the list markers are inserted using the technique from Recipe 4.5.

See Also

Chapter 7 for ways to translate this text into a working navigation menu

6.9 Creating Inline Lists

Problem

You want list items to be displayed within a paragraph, as in Figure 6-16, in which the boldface, comma-separated list was generated from an HTML `ul` list.
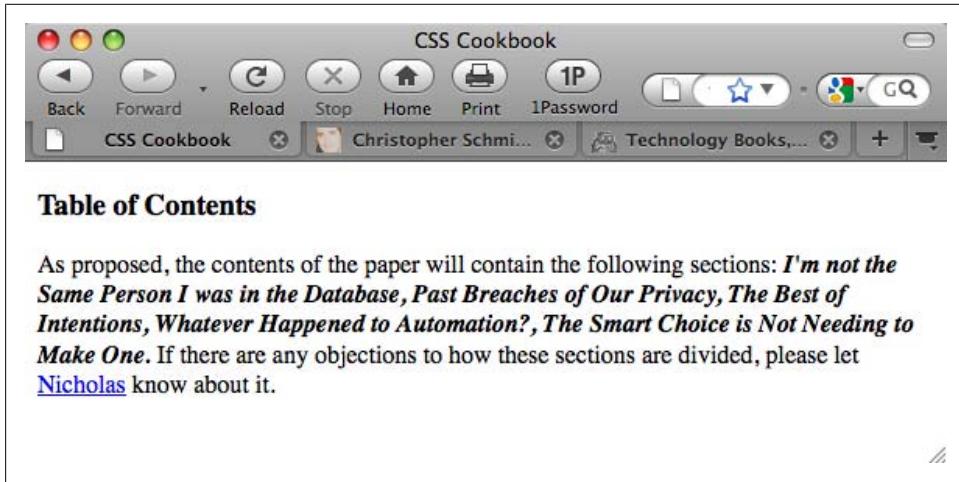
Figure 6-16. The list formatted to appear inside a paragraph

Solution

Set the paragraphs before (and, if needed, after) the list:

```
<h3>
 Table of Contents
</h3>
<p>
 As proposed, the contents of the paper will contain the
following sections:
</p>
<ul>
 <li>I'm not the Same Person I was in the Database</li>
 <li>Past Breaches of Our Privacy</li>
 <li>The Best of Intentions</li>
 <li>Whatever Happened to Automation?</li>
 <li class="last">The Smart Choice is Not Needing to Make One</li>
</ul>
<p>
 If there are any objections to how these sections are divided,
please let <a href="mailto:nick@example.com">Nicholas</a> know about
it.
</p>
```

Through CSS, set the paragraphs to display as inline elements and then use auto-generated content to show the commas between items and the period at the end of the list:

```
ul, li {
  display: inline;
  margin: 0;
  padding: 0;
  font-weight: bold;
  font-style: italic;
}
li:after {
  content: ", ";
}
li.last:after {
  content: ".";
}
p {
  display: inline;
}
```

Discussion

Through this method, you retain the structure of lists and paragraphs, but you stretch the capability of CSS to present the list inside a paragraph. However, you hide the obvious visual appearance of a list in favor of having the contents placed inside a paragraph.

The critical part of this Solution is setting the `display` property to `inline` on the list items and paragraphs. By using the `inline` value, you are placing the elements on the same line instead of separating them with whitespace above and below each element.
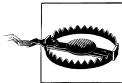
Internet Explorer for Windows 7 and earlier does not support generated content.

See Also

The CSS 2.1 specification for `display` at *http://www.w3.org/TR/CSS21/visuren.html #propdef-display*

6.10 Making Hanging Indents in a List

Problem

You want the first line of a list item to begin farther to the left than the rest of the list, thereby creating a hanging indent, as in Figure 6-17.
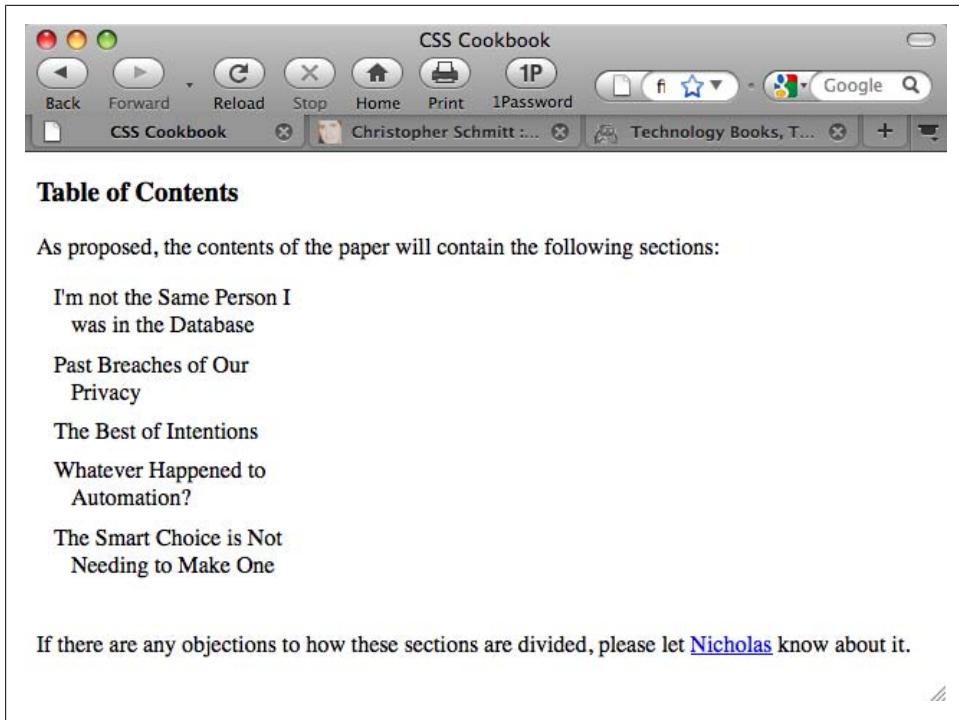
Figure 6-17. Hanging indents on a list

Solution

Use a negative value for the `text-indent` property:

```
ul {
 width: 30%;
 padding: 0 0 0.75em 0;
 margin: 0;
 list-style: none;
}
li {
 text-indent: -0.75em;
 margin: 0.33em 0.5em 0.5em 1.5em;
}
```

Discussion

Although list markers (numeric, image, or text) help to call attention to the actual list, sometimes you might not want to add those kinds of design elements to a list. Instead of relying on markers to carry off the list design, use a hanging indent.

In this Solution, you indent the list by three-quarters of an em unit, creating a visible but subtle hanging indent effect. You can push this design technique from subtle to the foreground by reducing the `text-indent` value further, or by increasing the font size of the text in the list item.

See Also

Recipe 3.24 for setting indents in paragraphs; the CSS 2.1 specification for `text-indent` at *http://www.w3.org/TR/CSS21/text.html#propdef-text-indent*

6.11 Moving the Marker Inside the List

Problem

You want the list marker to be pulled inside the border of the list items, as in Figure 6-18. This creates an effect in which the text wraps around the marker.
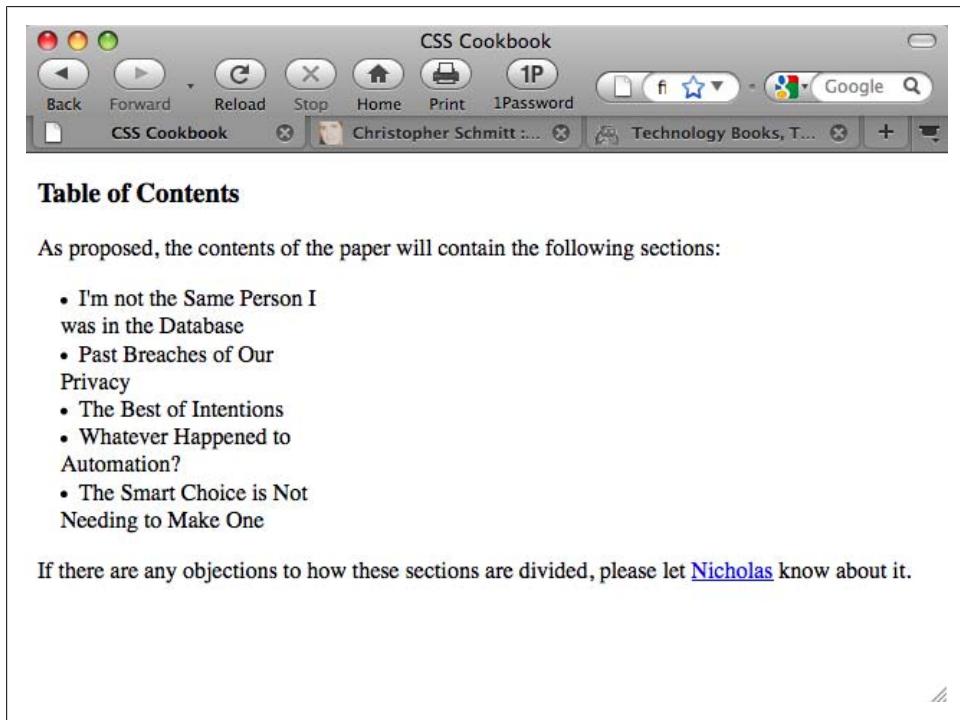
Figure 6-18. Moving the marker inside the list item

Solution

Use the `list-style-position` property and set the value to `inside`:

```
li {
 list-style-position: inside;
 width: 33%;
 padding: 0;
 margin: 0;
}
ul {
 margin: 0;
 padding: 0 0 0 1em;
}
```

Discussion

Normally the list marker stands outside the text and the result is a very distinctive list. Some designs, however, might require the marker to appear as part of the text. A designer might choose to keep the marker inside, for example, to eliminate the need to have enough whitespace on the left side.

Also, replacing the list marker with your own custom marker can visually enhance this recipe. For example, Figure 6-19 shows arrows rather than the default bullet.
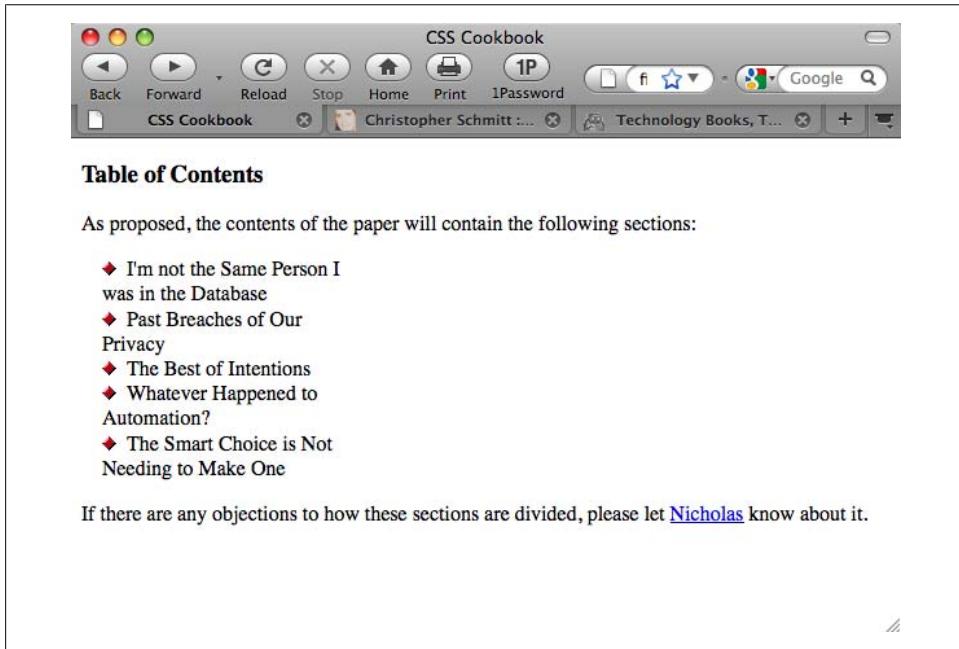
Figure 6-19. Custom marker inside the list item

The CSS 2.1 specification for list-style-position at *http://www.w3.org/TR/CSS21/ generate.html#propdef-list-style-position*

6.12 Styling a Definition List

Problem

You want to line up definitions with terms from a standard definition list.

Solution

Create a valid definition list through HTML, as shown in Figure 6-20:

```
<dl>
<dt>CSS</dt>
<dd>Cascading Style Sheets</dd>
<dt>HTML</dt>
<dd>HyperText Markup Language</dd>
<dd>How To Meet Ladies</dd>
<dd>Epsum factorial non deposit quid pro quo hic escorol. Olypian quarrels et
gorilla congolium sic ad nauseum. Souvlaki ignitus carborundum e pluribus
unum. Defacto lingo est igpay atinlay. Marquee selectus non provisio
incongruous feline nolo contendre.</dd>
</dl>
```
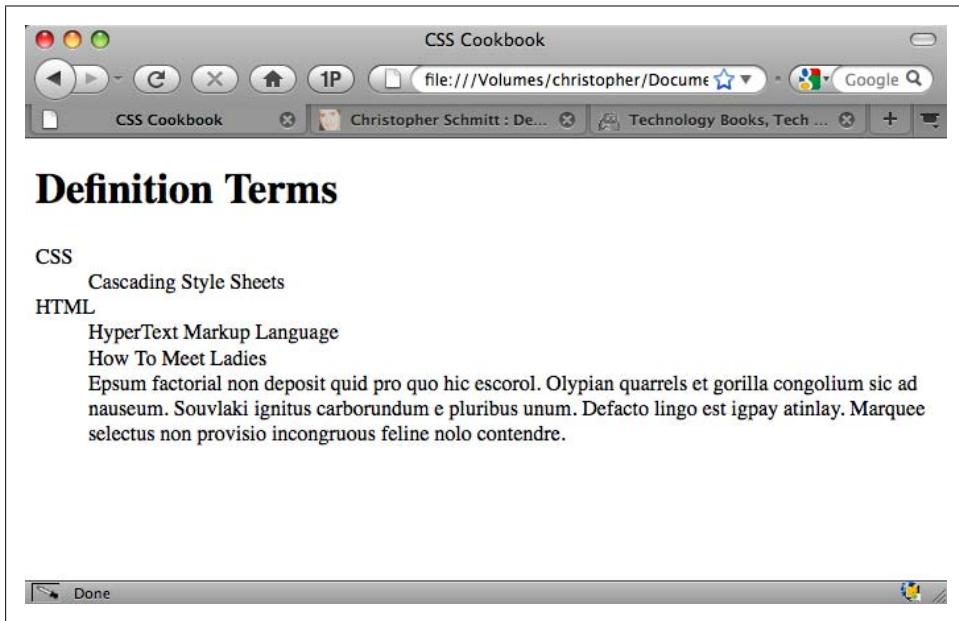

Figure 6-20. Default rendering of the definition list

Then create a margin on the left side of the entire definition list:

```
dl {
 margin-left: 5em;
 }
```

Apply a width to the definition terms to be less than the value given to the left margin, as shown in Figure 6-21:

```
dt {
 width: 4em;
 }
```
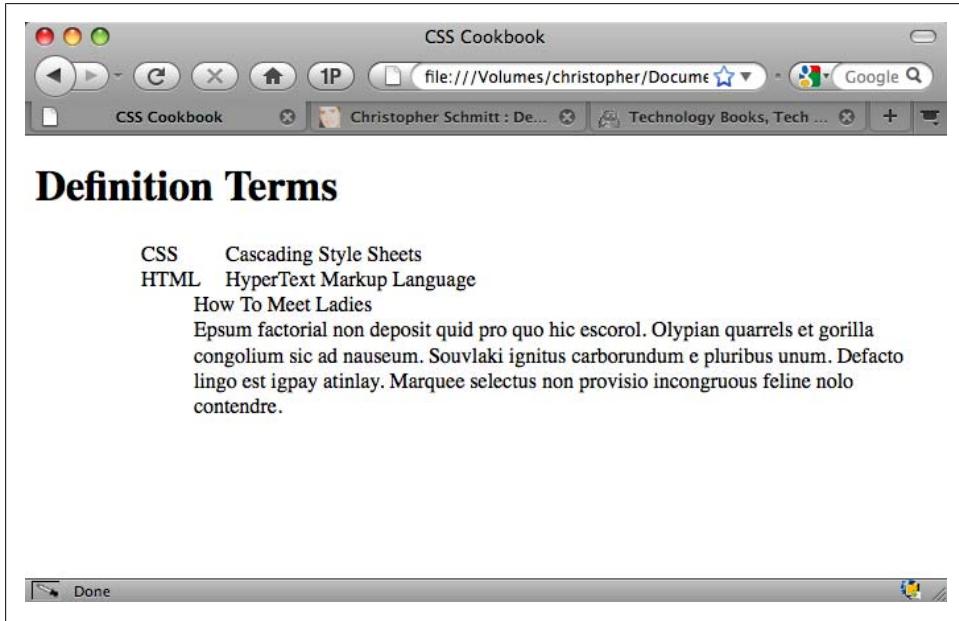

Figure 6-21. The definition term's width set to 4 em

Next, float and clear the definition term while also using a negative value on the left side of the definition term to bring the term to the left of its definition, as shown in Figure 6-22:

```
dt {
 width: 4em;
 float: left;
 clear: left;
 margin:0 0 1em -5em;
 font-weight: bold;
 }
```
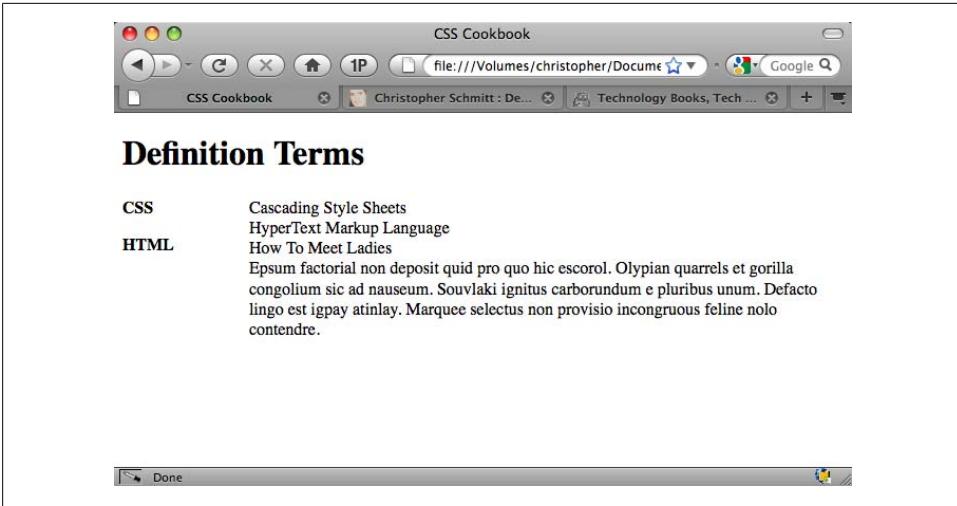
Figure 6-22. Moving the term to the left side of the definitions

For the definitions, set their floats to the left as well and set their widths to be 100%, as shown in Figure 6-23:

```
dd {
 float: left;
 width: 100%;
}
```
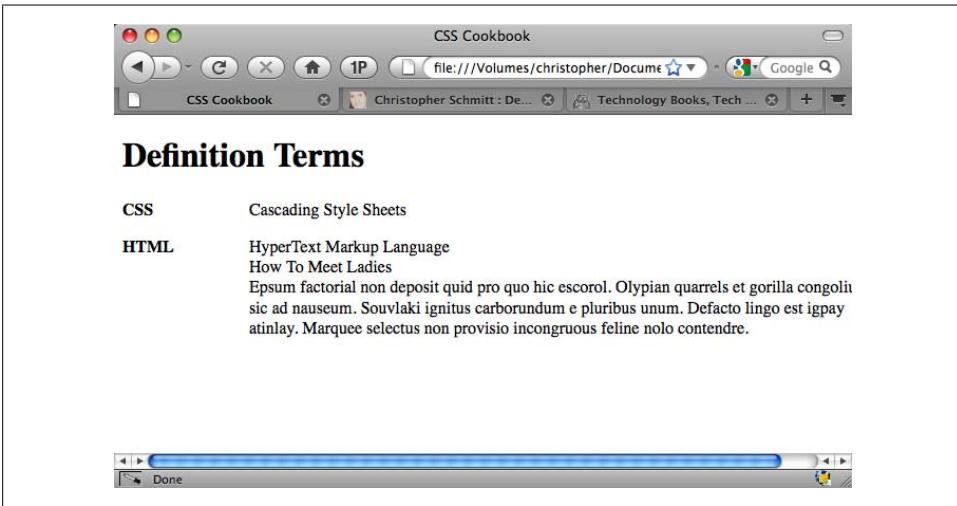

Figure 6-23. Adjusting the definitions' width

Then adjust the margin and padding to reinforce the visual distinction between the definitions, as shown in Figure 6-24:

```
dd {
 float:left;
 width:100%;
 padding: .2em 0 0 0;
 margin: 0 0 1em 0;
}
```
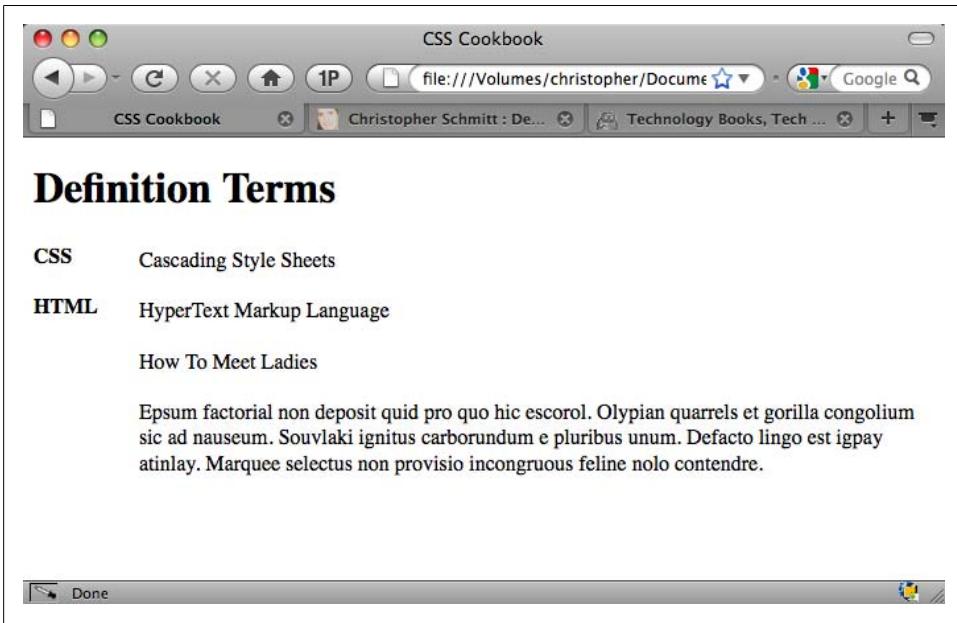

Figure 6-24. Adjusting the padding and margins of the definitions

After that, style elements to taste for better visual rendering, as shown in Figure 6-25:

```
dt {
 width: 4em;
 float: left;
 clear: left;
 margin:0 0 1em -5em;
 font-weight: bold;
 border-top: 1px solid #000;
 padding: .2em 0 0 0;
}
dd {
 float: left;
 width: 100%;
 padding: .2em 0 0 0;
 margin: 0 0 1em 0;
 color: #333;
}
```

```
dt+dd {
  border-top: 1px solid #000;
}
```
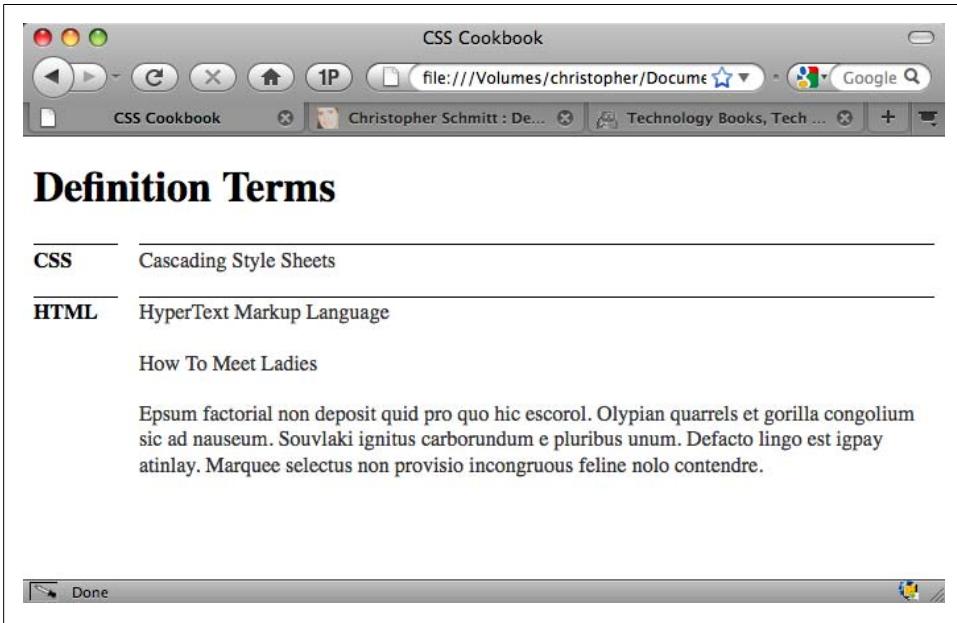

Figure 6-25. Polishing the look of the definition list

Discussion

Placing a term next to its definition is a fairly common solution. By applying a margin to the definition list as a whole on its left side, you can make the terms slide into the open area. After that, using floats (along with judicious use of padding) finalizes the manipulation.

Using generated content

To indicate that there are definitions after a term, use the `:after` pseudo-element on the definition term:

```
dt:after {
  content: ":";
}
```

Since terms may have more than one definition, it's possible to assign numbers to each definition. The CSS specification has a counter-mechanism that is suited for this purpose.

First, use the `counter-reset` property on the `dt` element:

```
dt {
 counter-reset: item;
}
```

The `counter-reset` property either creates or resets a counter. As the `dt` elements are rendered and the CSS is associated with the element, the counter is initiated and then subsequently reset with each rendering of this element in the document.

The next step is to tell the browser to output the number before each definition through the `counters()` function:

```
dd:before {
 content: counters(item, "") ". " ;
}
```

Within the `counters()` function, two parameters are passed: the counter to be used and then a string. The string is used to separate subsections. Examples of separators within a counter include the period within Recipe 1.8 and the hyphen in Recipe 6.11. In this Solution, there aren't any subsections, so the string is empty.

To insert a period after the number and a space, quotation marks are used after the `counters()` function.

With the counter output in place in the document, the next step is to tick the counter each time there is a new definition. This is done through the `counter-increment` property, which accepts the value of the counter name given to the `counter-reset` property:

```
dd:before {
 content: counters(item, "") ". " ;
 counter-increment:item;
}
```

Figure 6-26 shows the final result.

> Generated content is not supported in versions of Internet Explorer for Windows earlier than IE8. All other modern browsers do support generated content.

See Also

Robert O'Rourke's original work on getting the definition list to look like a table at *http://www.sanchothefat.com/dev/layouts/definition-lists-ugly.html*, after being inspired by Bruce Lawson's CSS Challenge at *http://www.brucelawson.co.uk/2009/css-challenge/*
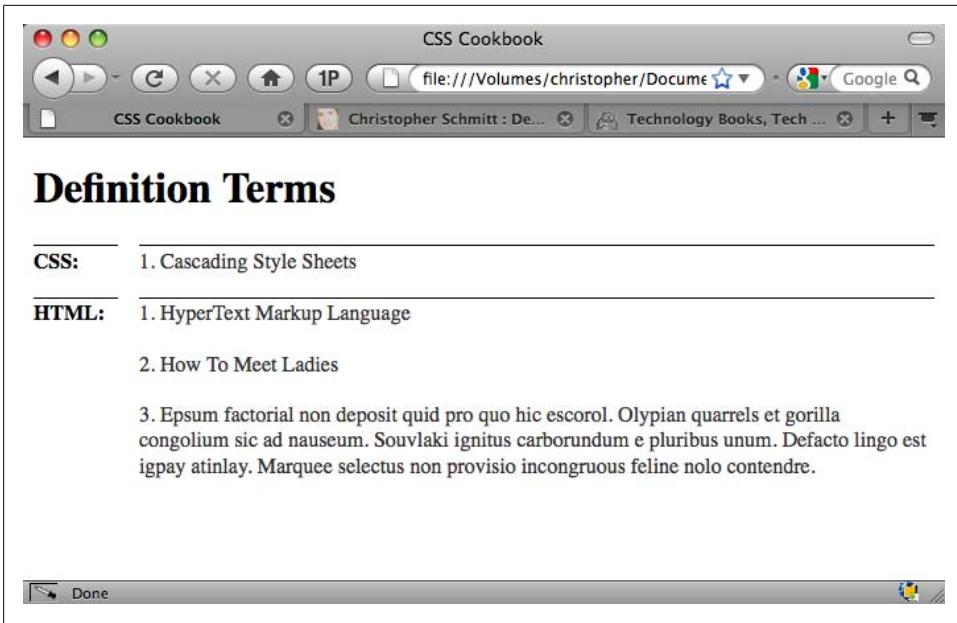
Figure 6-26. Using generated content in the definition list

6.13 Styling a Screenplay with the HTML5 dialog Element

Problem

You want to stylize a screenplay.

Solution

Mark up the content of the screenplay with the HTML5 `dialog` element:

```
<div id="screenplay">

<h3>Cut to</h3>

<p>Int. Kitchen - Continuous</p>

<dialog>
 <dt>Beth</dt>
 <dd> I told you the one about Salma Hayek?</dd>
</dialog>

<p>Beth walks closer to John.</p>

<p>The innocuous baby monitor gets <strong>louder</strong>.</p>

<dialog>
 <dt>Beth</dt>
```

```
<dd>Nursing a hungry baby in some destitute African village?</dd>
<dt>John</dt>
<dd><span class="how">(gasps)</span>No.</dd>
<dt>Beth</dt>
<dd>This actually happened, but the commentator, I forget who, ended the piece with
    "your move, Jolie"</dd>
</dialog>

</div><!-- /#screenplay -->
```

Then apply style rules to adjust the formatting of the content to look like a screenplay:

```
body {
 font-size: 62.5%;
 font-family: "Courier New", Courier, monospace;
 margin: 0 auto;
 width: 612px;
}
#screenplay {
 padding: 0 10.9em;
}
#screenplay h3 + p {
 text-transform: uppercase;
}
#screenplay h3 {
 text-transform: uppercase;
 text-align: right;
 background: white;
}
#screenplay h3:after {
 content: ":";
}
dialog {
 font-size: 1.2em;
}
dt {
 text-transform: uppercase;
 text-align: center;
 margin-top: 1.6em;
}
dd {
 margin-left: 7.2em;
}
span.how {
 display: block;
 text-align: center;
 margin-right: 7.2em;
 padding-right: 5em;
}
#screenplay strong {
 text-transform: uppercase;
}
```

Discussion

The HTML5 specification brings in a new element, `dialog`, specifically for indicating conversation. The format the markup uses is the same as `dt` and `dd` elements, but it replaces the `dl` element with `dialog`.

See Also

The HTML5 specification for `dialog` at *http://www.w3.org/TR/2008/WD-html5 -20080122/#the-dialog*

6.14 Turning a List into a Directory Tree

Problem

You want to re-create a directory tree structure from a list.

Solution

First, set up a series of nested ordered lists to serve as the basis for the directory tree structure:

```
<ul class="itinerary">
 <li>Morning Sessions
  <ul>
   <li>Troubleshooting IE6</li>
   <li>Object Oriented CSS</li>
   <li>Fluid Typography</li>
   <li>Tomorrow's CSS3 Today</li>
  </ul>
 </li>
 <li>Afternoon Sessions
  <ul>
   <li>Web Form Elements</li>
   <li>Flexible Layouts</li>
   <li>Coding Layouts</li>
   <li>Future CSS &amp; Markup</li>
  </ul>
 </li>
</ul>
```

Create three sets of small graphics: a vertical pipe or trunk; a branch; and an end branch graphic, as shown in Figure 6-27.
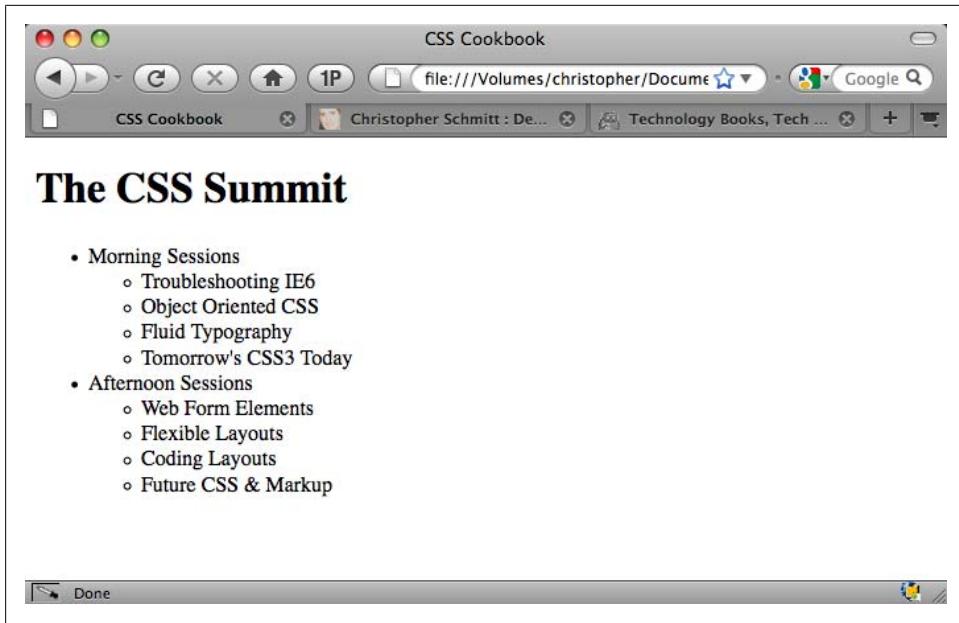
Figure 6-27. Default rendering of the unordered lists

Apply the vertical pipe graphic to the sides of the unordered lists, as shown in Figure 6-28:

```
.itinerary, .itinerary ul {
 list-style-type: none;
 background-image: url(pipe.gif);
 background-repeat: repeat-y;
 margin: 0;
 padding: 0;
}
.itinerary ul {
 margin-left: 12px;
}
```

Apply a branch graphic at each list item:

```
.itinerary li {
    margin: 0;
    padding: 0 12px 0 28px;
    background-image: url(branch.gif);
    background-repeat: no-repeat;
    line-height: 1.5;
}
```
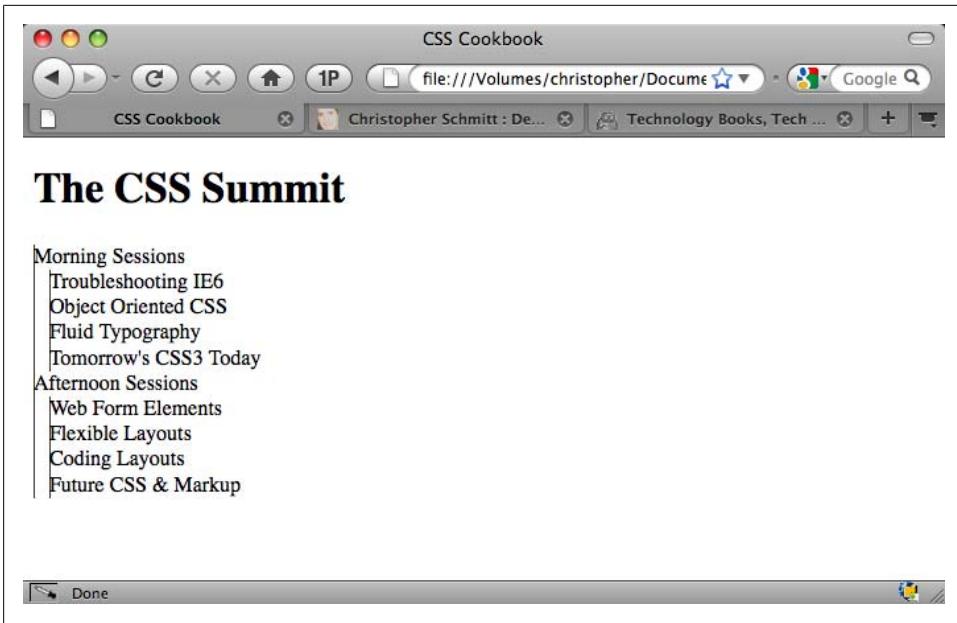
Figure 6-28. The vertical lines set

Then hardcode the last list item in each unordered list with a `class` attribute in the HTML:

```
<ul class="itinerary">
 <li>Morning Sessions
  <ul>
   <li>Troubleshooting IE6</li>
   <li>Object Oriented CSS</li>
   <li>Fluid Typography</li>
   <li class="branchend">Tomorrow's CSS3 Today</li>
  </ul>
 </li>
 <li class="branchend">Afternoon Sessions
  <ul>
   <li>Web Form Elements</li>
   <li>Flexible Layouts</li>
   <li>Coding Layouts</li>
   <li class="branchend">Future CSS &amp; Markup</li>
  </ul>
 </li>
</ul>
```

Now apply a class selector to bring in the end branch graphic, as shown in Figure 6-29:

```
.itinerary  li.branchend {
/* matches background color of */
/* parent element or page */
 background-color: #fff;
 background-image: url(branchend.gif);
}
```
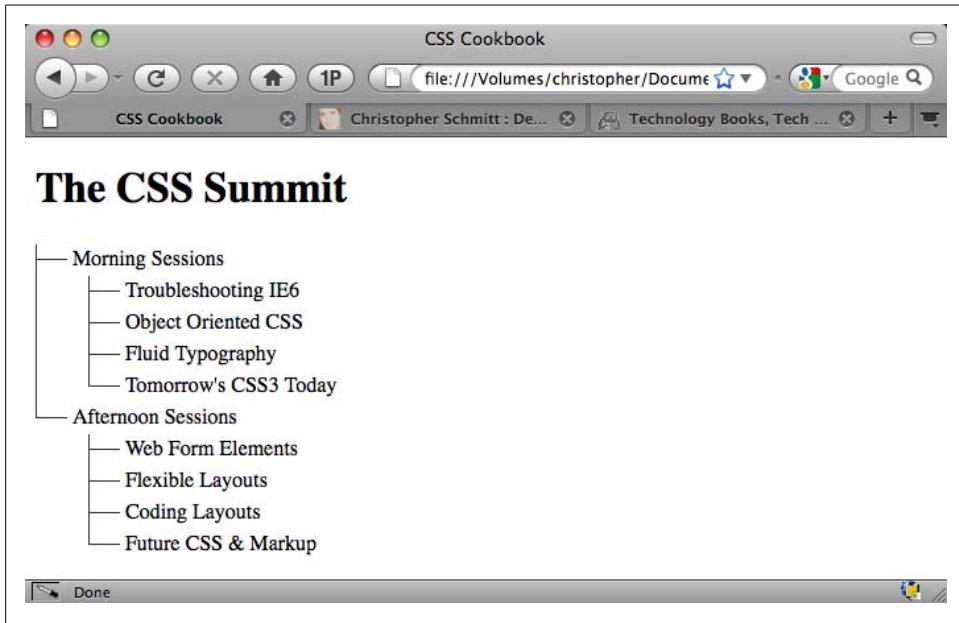

Figure 6-29. Applying the end branches

Discussion

The technique for this Solution builds off Recipe 6.8, which uses icons placed in the background of the list item. This Solution calls for three different small images to be placed at certain areas in the ordered lists to pull off the effect.

Using CSS3

To place the end branch of the directory tree, we had to include a `class` attribute in the markup for the Solution to work.

In CSS3, the `:last-of-type` pseudo-class can replace the need for that `class` attribute:

```
.itinerary  li:last-of-type {
/* matches background color of */
/* parent element or page */

 background-color: #fff;
 background-image: url(branchend.gif);
}
```

At the time of this writing, the `:last-of-type` pseudo-class is supported in Safari 3 and later and Opera9.5 and later.

For a listing of CSS3 selectors, see Appendix D.

See Also

Michal Wojciechowski's "Turning Lists into Trees" at *http://odyniec.net/articles/turn ing-lists-into-trees/*

6.15 Creating a Star Ranking System

Problem

You want to display a star rating system that allows users to visually pick their own ratings.

Solution

The first step is to set up the HTML to include an unordered list with five options, as shown in Figure 6-30:

```
<div class="product" id="prod345781">
<h1>CSS Cookbook</h1>
<p>Submit your review:</p>
 <ul class="rating">
  <li class="one"><a href="#">1 Star</a></li>
  <li class="two"><a href="#">2 Stars</a></li>
  <li class="three"><a href="#">3 Stars</a></li>
  <li class="four"><a href="#">4 Stars</a></li>
  <li class="five"><a href="#">5 Stars</a></li>
 </ul>
</div>
```

Next, create an image containing every combination of star ratings, along with an active hover state, as shown in Figure 6-31. (You may want to make each star a square shape, as it makes coding the CSS a little bit easier.)
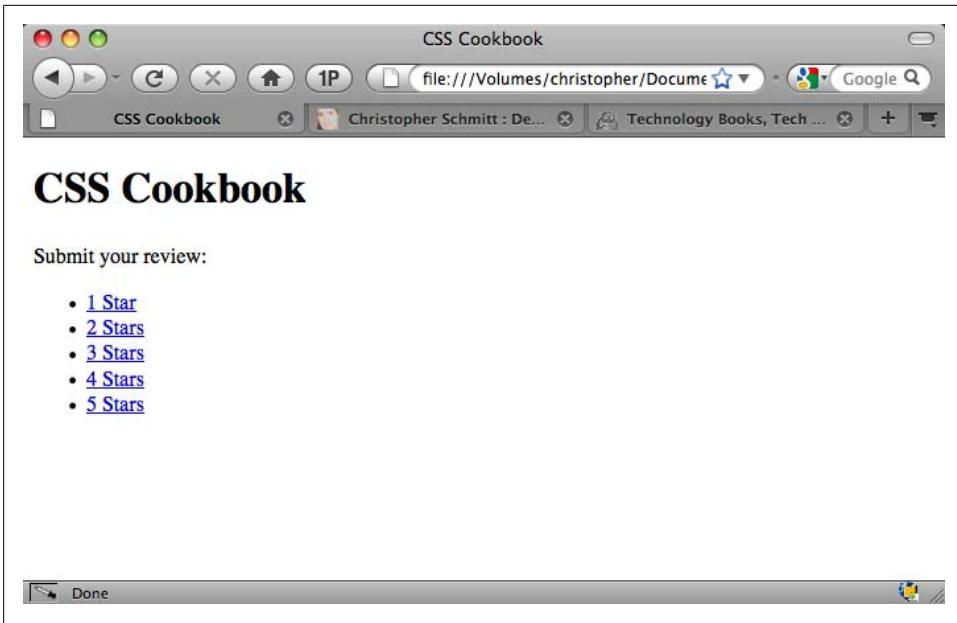
Figure 6-30. Default rendering of the star ranking HTML
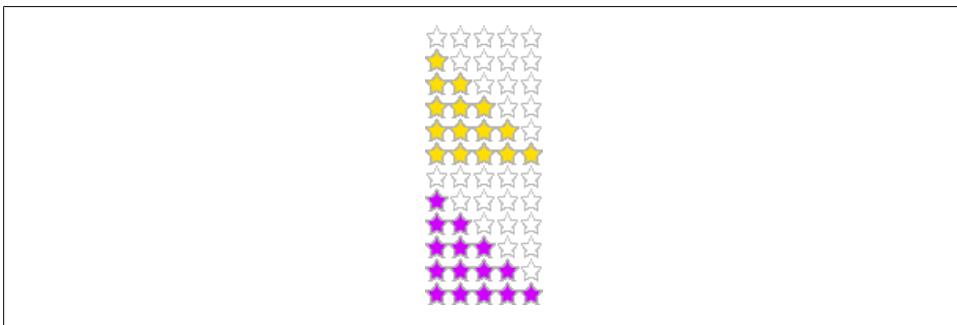
Figure 6-31. Every combination of star rankings in one image

With the star image set, use CSS rules to restrict the width and height of the unordered list and bring in the star matrix:

```
.rating {
margin: 0;
padding: 0;
list-style: none;
clear: both;
width: 75px;
height: 15px;
background-image: url(stars.gif);
background-repeat: no-repeat;
```

```
    position: relative;
  }
```

Next, float each list item (for IE6 support) while removing its text using a negative value with the `text-indent` property:

```
.rating li {
  text-indent: -9999em;
  float: left; /* for IE6 */
}
```

The next step is to absolutely position each list item's link in a row within the 75 px boundary of the unordered list set previously:

```
.rating li a {
  position: absolute;
  top: 0;
  left: 0;
  z-index: 20;
  height: 15px;
  width: 15px;
  display: block;
}
.rating .one a {
  left: 0;
}
.rating .two a {
  left: 15px;
}
.rating .three a {
  left: 30px;
}
.rating .four a {
  left: 45px;
}
.rating .five a {
  left: 60px;
}
```

With the blocks in place, you can apply the default rating to the product through CSS. For example, a 2 out of 5 star review would need a simple `background-position` declaration block, as shown in Figure 6-32:

```
#prod345781 .rating {
  /* background-position: 0 0px;      0 out of 5 */
  /* background-position: 0 -15px;    1 out of 5 */
      background-position: 0 -30px; /* 2 out of 5 */
  /* background-position: 0 -45px;    3 out of 5 */
  /* background-position: 0 -60px;    4 out of 5 */
  /* background-position: 0 -75px;    1 out of 5 */
}
```
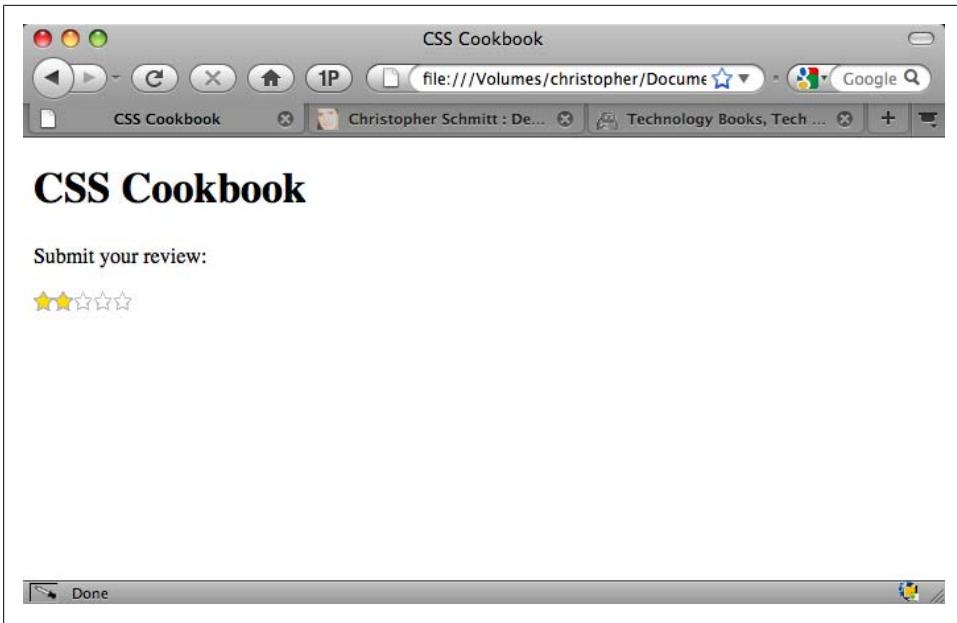
Figure 6-32. A two-star rating

To add the user feedback portion of the star ranking system, set each list item's link to expand to fit the entire 75-pixel width and reinsert the star image as the background image:

```
#prod345781 .rating li a:hover {
  z-index: 10;
  width: 75px;
  height: 15px;
  overflow: hidden;
  left: 0;
  background-image: url(stars.gif);
  background-repeat: no-repeat;
}
```

Then write specific rules that move the background image so that the second set of color stars appears. How far the background image moves upward depends on which star ranking the user is mousing over, as shown in Figure 6-33:

```
#prod345781 .rating .one a:hover {
  background-position: 0 -105px; /* 1 out of 5 */
}
#prod345781 .rating .two a:hover {
  background-position: 0 -120px; /* 2 out of 5 */
}
#prod345781 .rating .three a:hover {
  background-position: 0 -135px; /* 3 out of 5 */
}
#prod345781 .rating .four a:hover {
```

```
  background-position: 0 -150px; /* 4 out of 5 */
  }
#prod345781 .rating .five a:hover {
  background-position: 0 -165px; /* 5 out of 5 */
  }
```
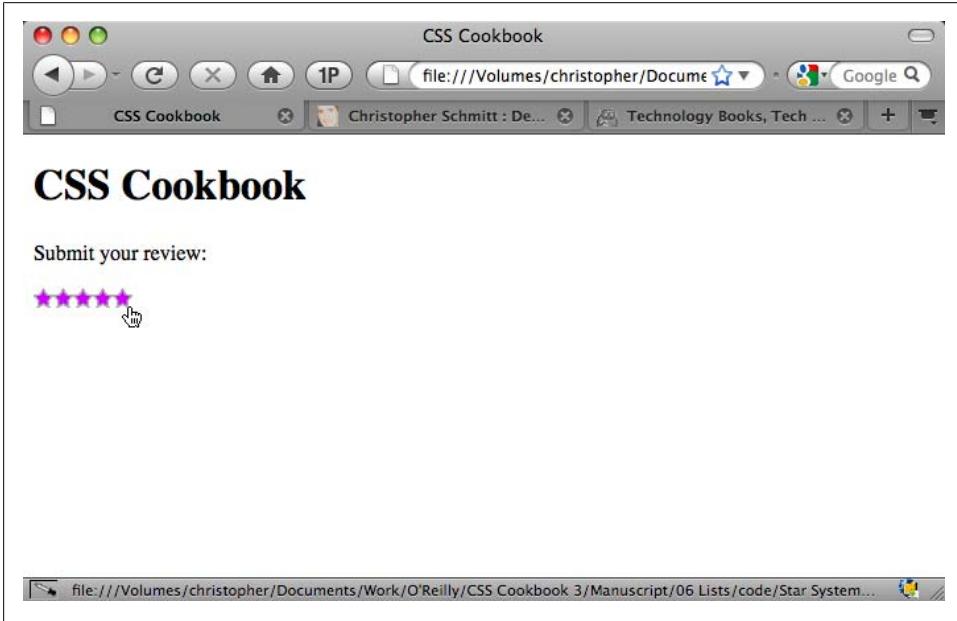

Figure 6-33. Hovering over the stars, which shows the user's personal rating of a product or service

Discussion

This Solution relies on several techniques.

The first crucial technique is the use of a *CSS sprite*, an image with several icons placed within the same image (see Recipe 4.33 for more information). Using the background image of the unordered list itself, the initial star rating is set through background positioning of the star image.

The next technique is to change the positioning technique, called *shackling* (see Recipe 2.25). By absolutely positioning each link within the unordered list side by side, the user is able to click on how many stars to assign the product or service.

Finally, the last technique reuses the CSS sprite image. As the user selects which star rating to give the product or service, the width of the link changes to fill the entire width of all the stars. This allows the background image to come in and appear over the unordered list's own background image.

Even though these two elements are using the same background image, the link is placed higher or above the `li` property's background. Therefore, the link's background image is visible while the `li` property's background is not.

Setting the `background-position` value to a higher negative value moves the different set of star colors into position.

See Also

Paul O'Brien's in-depth article about this star matrix technique at *http://www.search -this.com/2007/05/23/css-the-star-matrix-pre-loaded/*

Links and Navigation

7.0 Introduction

Without links, the point of the Web would be lost.

Links let you follow a trail of information from one web page to another and from one idea to another, regardless of where the site's server is located in the world.

In 1996, web usability expert Jakob Nielsen listed the use of nonstandard link colors as one of the top 10 mistakes in web design (see *http://www.useit.com/alertbox/9605 .html*). However, his advice to use blue for the link color for pages that the user hasn't visited and purple or red to represent previously visited pages came from consistency concerns, not aesthetics.

Thankfully, he has updated his thoughts on link colors for the new millennium (see *http://www.useit.com/alertbox/link-list-color.html*). Links, being an essential part of the World Wide Web, can be both consistent *and* visually pleasing.

This chapter shows how to improve aesthetics by changing link styles. You'll learn everything, from how to remove the underline from links to how to change cursors, create rollovers without the need for JavaScript, create a horizontal tab menu, and much more.

7.1 Easily Generating Text-Based Menus and Submenus

Problem

You want to quickly generate the markup for a navigation list along with premade styles.

Solution

Go to Accessify's menu builder, List-O-Matic (see *http://accessify.com/tools-and-wiz
ards/developer-tools/list-o-matic/*).

Fill out labels for navigation menus, link addresses, and the optional title attributes, as
shown in Figure 7-1.

Figure 7-1. Online web application for generating accessible menus with unordered lists

Next, pick the style of navigation menu, as shown in Figure 7-2.
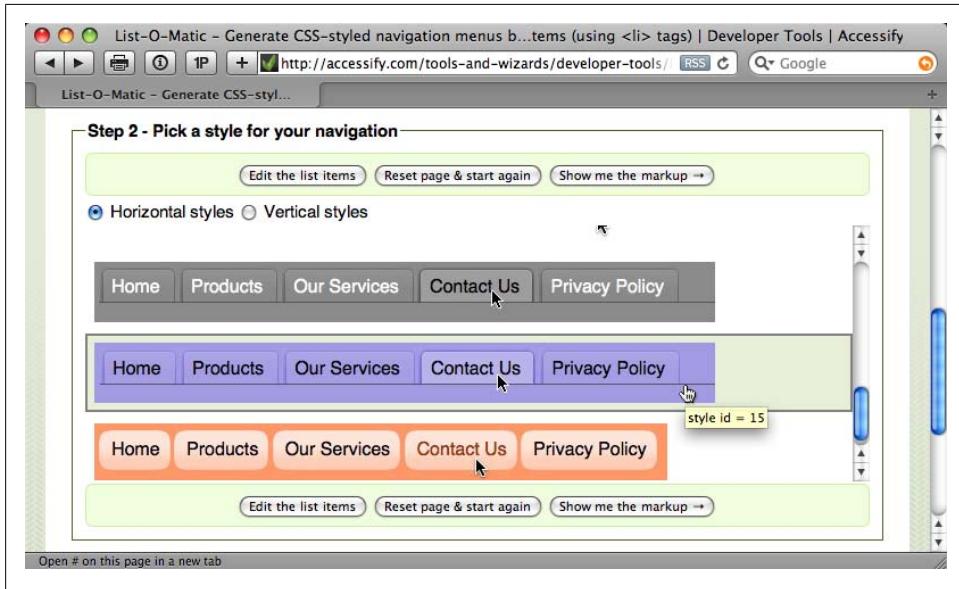
Figure 7-2. Examples of the menu designs

Click "Show me the markup" to get the markup and CSS rules that can be added directly to your web page.

Discussion

Utilizing both unordered lists and links (see Recipe 1.10), Accessify's List-O-Matic handles the heavy lifting of coding and styling a navigation menu. To fit a style within your site, be sure to customize the CSS rules to your site's design.

See Also

A video tutorial on how to use List-O-Matic, with a voiceover by someone with a British accent, at *http://accessify.com/screencasts/list-o-matic/*

7.2 Removing Underlines from Links (and Adding Other Styles)

Problem

You want to remove the default underlining of links, as shown in Figure 7-3.
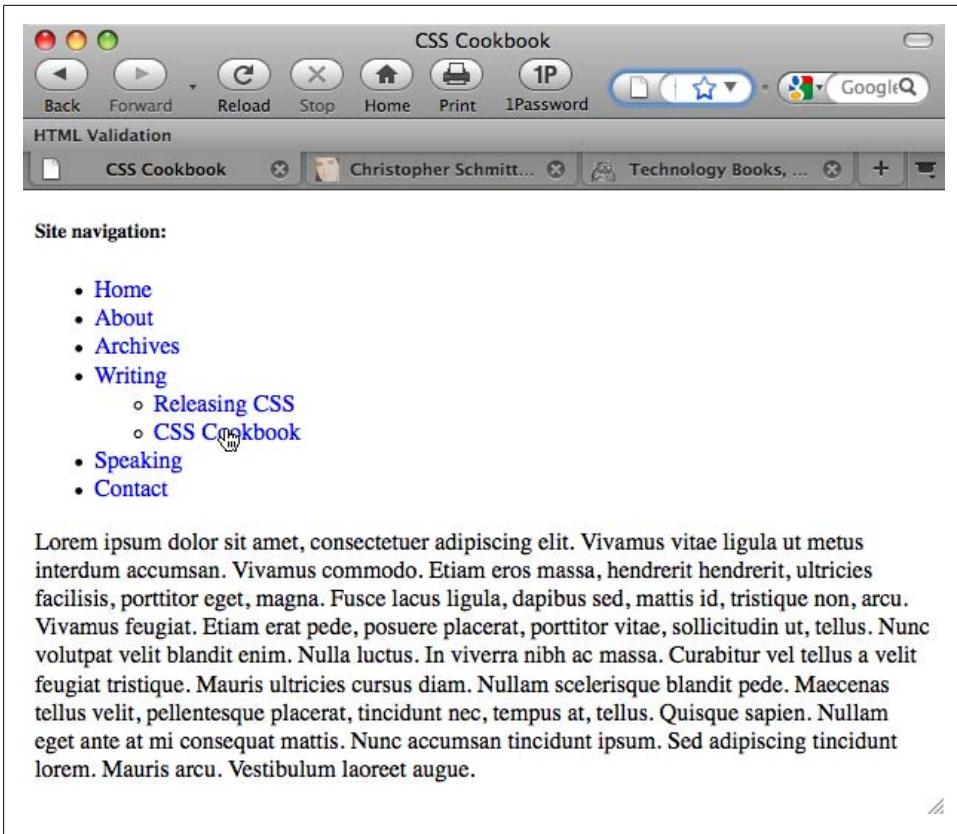
Figure 7-3. Links without underlines

Solution

Use the `text-decoration` property with the pseudo-class selector for unvisited and visited links:

```
a:link, a:visited {
 text-decoration: none;
}
```

Discussion

Use the `:link` and `:visited` pseudo-classes to apply styles to links within a web document. The `:link` pseudo-class applies to links that the user has not visited. The `:visited` pseudo-class corresponds to links that the user has visited.

The `text-decoration` property can take up to five settings, shown in Table 7-1.

Table 7-1. Text-decoration settings

Text-decoration value	Result
underline	A line is placed beneath the text
overline	A line is placed above the text
blink	The text flashes
line-through	A line is placed through the middle of the text
none	No effect is associated with the text

These `text-decoration` properties are often used to enhance the presentation of a web page. Instead of having all the links in a document underlined, designers set `text-decoration` to `none` along with changing the link's background color, text color, or both:

```
a:link, a:visited {
  text-decoration: none;
  background-color: red;
  color: white;
}
```

To complement the design for site visitors who might have color blindness and therefore might not be able to determine a link color from the default color of regular HTML text, designers also set the weight of the font to bold:

```
a:link, a:visited {
  font-weight: bold;
  text-decoration: none;
  color: red;
}
```

The value of `line-through` might be an interesting element you can add to a page design to indicate that a link has already been visited by a user, similar to an item scratched off a to-do list, as shown in Figure 7-4:

```
a:link {
 font-weight: bold;
 text-decoration: none;
 color: red;
}
a:visited {
 font-weight: bold;
 text-decoration: line-through;
 color: black;
}
```

See Also

The CSS 2.1 specification for `text-decoration` at *http://www.w3.org/TR/CSS21/text .html#propdef-text-decoration*; Jakob Nielsen's updated "Design Guidelines for Visualizing Links" at *http://www.useit.com/alertbox/20040510.html*
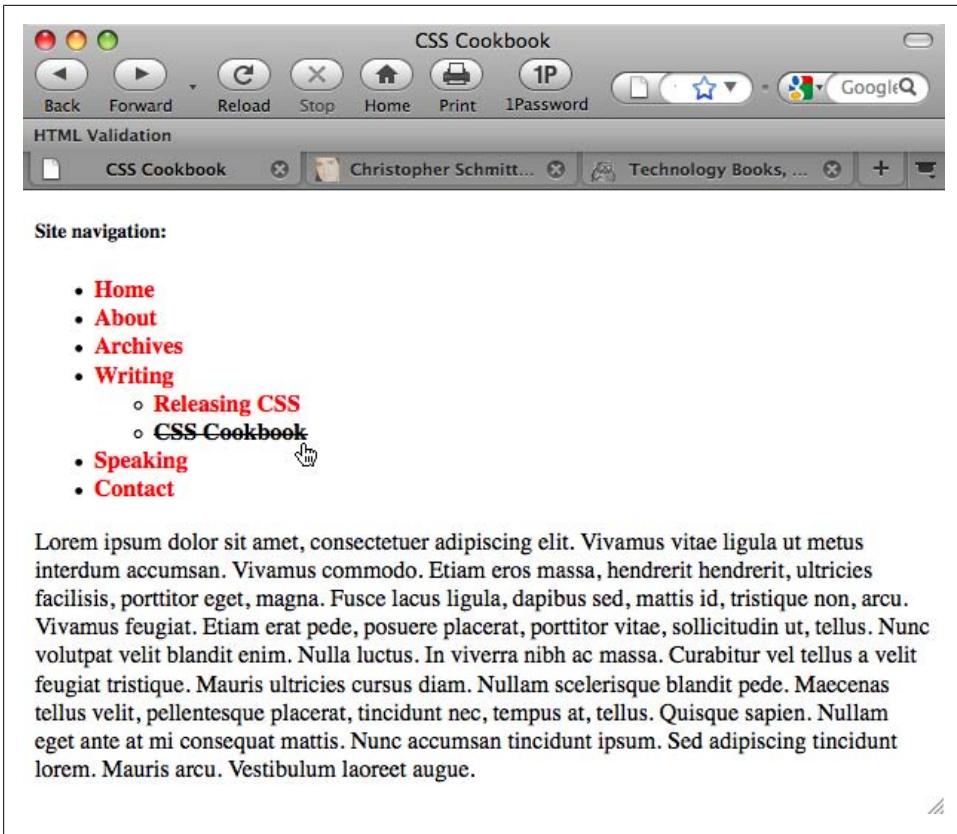
Figure 7-4. Visited link crossed out

7.3 Changing Link Colors

Problem

You want to create different styes for links: one style for navigation and another style for links within the main text.

Solution

Use the `:link`, `:visited`, `:hover`, and `:active` pseudo-classes, in that order:

```
body {
 color: #9ff;
}
a:link {
 color: #3cf;
}
a:visited {
 color: #cecece;
```

```
}
a:hover {
 color: #366;
}
a:active {
 color: #399;
}
```

Discussion

The hyperlink pseudo-classes are equal in terms of priority within the cascade; you avoid this conflict by listing the selectors in the following order: `:link`, `:visited`, `:hover`, and `:active`. The mnemonic device commonly used to remember the order is "LoVe/HAte."

A visited or an unvisited link can enter the hover and active states at the same time. Since hyperlink pseudo-classes have the same ranking, the one listed last is what the user sees, and that's why `:hover` won't work in some cases. When `:hover` appears before `:active` or `:visited`, the `:active` or `:visited` selector hides the hover state based on the cascading rules.

See Also

The CSS 2.1 specification for the dynamic pseudo-classes `:hover`, `:active`, and `:focus` at *http://www.w3.org/TR/CSS21/selector.html#dynamic-pseudo-classes*; Eric Meyer's Q&A on link specificity at *http://www.meyerweb.com/eric/css/link-specificity.html*

7.4 Removing Dotted Lines When Clicking on a Link in Internet Explorer

Problem

You want to remove the dotted lines that appear when you click on links in Internet Explorer.

Solution

Set the `outline` property to `none` for links:

```
a {
 outline: none;
}
```

Discussion

The `outline` property is not a part of the box model like `margin`, `border`, and `padding` are. Even though the border is taken into account when adding up the width of an element, the outline is not.

> Unlike for borders, the sides of an outline do not have specific CSS properties. For example, there is not an `outline-top` property.

The dotted outlines common in Internet Explorer for Windows aid in accessibility, allowing site visitors to know where they clicked or what is clickable on a page. However, there might be a few times when an outline of a link would compromise the visual style of a design.

To provide some feedback for site visitors (even if `outline` is set to `none` or `not`), it's recommended to set the `:focus` pseudo-class when styling links along with setting the rollover effects:

```
a:hover, a:active, a:focus {
 color: #399;
}
```

The use of `:focus` occurs when an element, such as an `input` element, is activated by the user's keyboard or other input.

> Internet Explorer requires a valid DOCTYPE (see Recipe 1.3) for the `outline` property to be applied.

See Also

The CSS2 specification for `outline` at *http://www.w3.org/TR/CSS2/ui.html#dynamic-outlines*

7.5 Changing Link Colors in Different Sections of a Page

Problem

You want to apply different links to the main text and the navigation.

Solution

First, wrap sections of the page with `div` elements and different attribute values:

```
<div id="nav">
 [...]
</div><!-- /#nav -->
<div id="content">
 [...]
</div><!-- /#content -->
```

Then use descendant selectors with `ID` selectors along with the LV/HA method discussed in Recipe 7.3 to isolate different link styles to different areas of a web page:

```
/* navigation link design */
#nav a:link {
 color: blue;
}
#nav a:visited {
 color: purple;
}
/* content link design */
#content a:link {
 color: white;
}
#content a:visited {
 color: yellow;
}
```

Discussion

The use of the `ID` selector to identify sections of a web page opens the door for applying different styles to the same elements. Rely on the same selectors to create links with different styles by section. For more on the `ID` selector, see Recipe 2.2. Applying LV/HA mnemonic order to links also ensures that your links operate as expected.

See Also

W3Schools' tutorial on CSS pseudo-classes at *http://www.w3schools.com/css/css_pseudo_classes.asp*

7.6 Placing Icons at the End of Different Kinds of Links

Problem

You want a way to display icons at the end of an inline link, as shown in Figure 7-5.
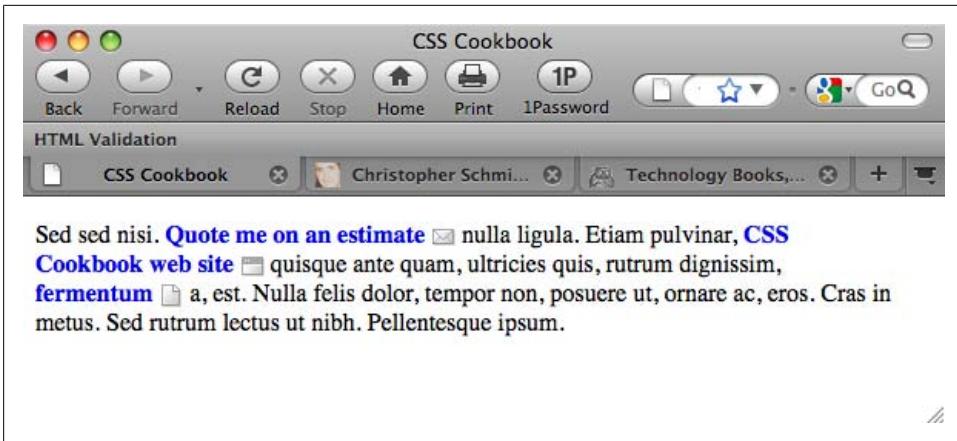
Figure 7-5. Icons placed at the end of links

Solution

Set up links within a document:

```
<p>Sed sed nisi. <a href="mailto:orders@csscookbook.com">Quote me on
an estimate</a> nulla ligula. Etiam pulvinar,
<a href="http://www.csscookbook.com/">CSS Cookbook web site</a> quisque
ante quam, ultricies quis, rutrum dignissim, <a href="sample-chapter.pdf">
fermentum</a> a, est. Nulla felis dolor, tempor non, posuere ut, ornare ac,
eros. Cras in metus. Sed rutrum lectus ut nibh. Pellentesque ipsum.</p>
```

Then use attribute selectors (see Recipe 2.5):

```
a {
 text-decoration: none;
 font-weight: bold;
}
a[href^="mailto:"] {
 padding-right: 20px;
 background: url(mail.gif) no-repeat right;
}
a[href^="http://"] {
 padding-right: 20px;
 background: url(website.gif) no-repeat right;
}
a[href$="pdf"] {
 padding-right: 20px;
 background: url(document.gif) no-repeat right;
}
```

Discussion

Using attribute selectors is another method that doesn't require the additional markup of placing an `img` element in the content.

Recipe 2.5 discusses many ways in which you can use attribute selectors to place icons (or text) in front of or before links (as well as other elements).

Since the Solution uses content generation, it's not suitable for IE7 or earlier browsers. IE8 supports content generation.

See Also

Dave Shea's presentation on adding an icon with a background image in an inline link at *http://www.mezzoblue.com/presentations/2006/sxsw/css/q1.html*; an explanation as to why this fails in IE at *http://www.brunildo.org/test/InlineBlockLayout.html*

7.7 Changing Cursors

Problem

You want to change the cursor when the mouse pointer rolls over a link, as shown in Figure 7-6.
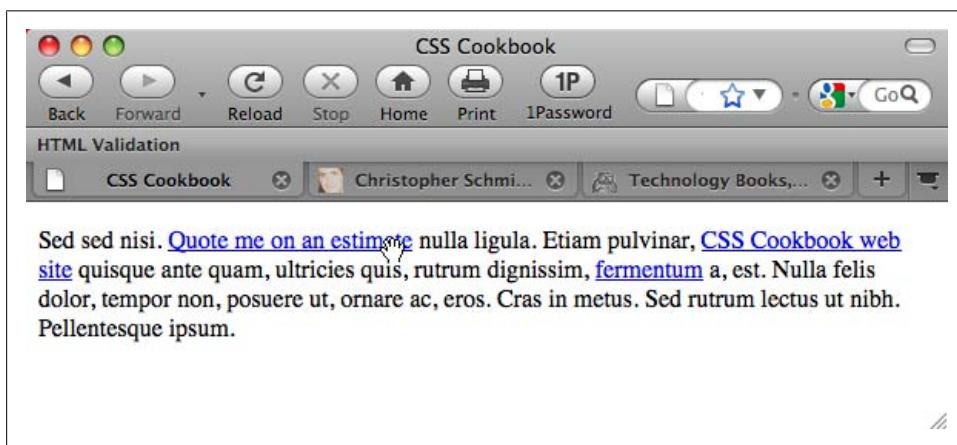
Figure 7-6. The cursor changing to a hand

Solution

Use the `cursor` property to change the cursor:

```
a:link, a:visited {
 cursor: move;
}
```

Discussion

The `cursor` property can take multiple values, as listed in Table 7-2. However, support for these values varies from browser to browser. Opera 7 and later and Internet Explorer

for Windows 5.5 and later both support the `cursor` property. Although Firefox supports most values, the browser doesn't support `uri`. Also, in Firefox, child elements do not inherit the `cursor` property from parent elements.

Table 7-2. Cursor property values

Value	Description	Sample
`auto`	Cursor changes to an image that is determined by the browser	
`crosshair`	Cursor changes to two perpendicular lines intersecting in the middle; this is similar to an enlarged plus sign	$+$
`default`	Platform-dependent cursor that in most browsers is rendered as an arrow; browser vendors or computer operating systems may dictate a different cursor style	
`pointer`	Used to illustrate that the mouse pointer is over a link; sometimes rendered as a hand with an extended index finger; browser vendors or computer operating systems may dictate a different cursor style	
`move`	Illustrates that an element can be moved; sometimes rendered as a crosshair with arrow-heads on the tips or a five-fingered hand	
`e-resize,` `ne-resize,` `nw-resize,` `n-resize,` `se-resize,` `sw-resize,` `s-resize,` `w-resize`	An arrow illustrating the direction in which a side can be moved; for example, `se-resize` indicates a southeast direction	
`text`	Illustrates that text can be selected; sometimes rendered like an I-beam commonly used in word processing programs	
`wait`	Illustrates that the computer is busy; sometimes rendered as an hourglass	
`progress`	Illustrates that the computer is busy, but the user still can interact with the browser	
`help`	Illustrates that information or help is available, often at the destination of the link; sometimes rendered as a question mark or an arrow with a question mark	
`<uri>`	Cursor can be swapped with an externally defined cursor such as an image, Windows cursor file, SVG cursor, etc.	N/A

The code to include a custom cursor is similar to that used to set a background image on an element:

```
a.help:link, a.help:visited{
 cursor: url(bewildered.gif);
}
```

While employing different cursors, most users will find changes to their routine surfing habits to vary from being a whimsical annoyance to being an extreme aggravation, depending on how excessive your implementation is. Therefore, change the cursor a user is accustomed to seeing at your own risk.

See Also

The CSS 2.1 specification for `cursor` at *http://www.w3.org/TR/CSS21/ui.html#propdef-cursor*; examples of the various cursors in action at *http://www.zimmertech.com/tutorials/css/20/changing-cursors-tutorial.php*

7.8 Creating Rollovers Without JavaScript

Problem

You want to create a simple rollover effect without using JavaScript to swap images.

Solution

Use the `:hover` and `:active` pseudo-classes to create the rollover:

```
a:link {
 color: #777;
 text-decoration: none;
}
a:visited {
 color: #333;
 text-decoration: none;
}
a:link:hover, a:visited:hover {
 color: #777;
 background-color: #ccc;
}
a:link:active, a:visited:active {
 color: #ccc;
 background-color: #ccc;
}
```

Discussion

The `:hover` pseudo-class mimics the common JavaScript event `onmouseover`. Instead of executing a function in JavaScript, when a user rolls over a link with `:hover`, a different set of styles is applied to the link.

With the selectors having the same specificity, selectors written out of order may stop one of the other styles from appearing. Avoid this common problem with LV/HA (see Recipe 7.5).

Although `:hover` and `:active` can be applied to any element, they are commonly used on links.

Known issues

In the Solution, the two pseudo-classes make sure the rollover effects occur only on anchor links. Without `:hover` and `:active`, modern browsers could legally apply the rollover effects on any anchor elements, as shown in this code and in Figure 7-7:

```
<h2><a name="europan">Li Europan lingues</a></h2>
```
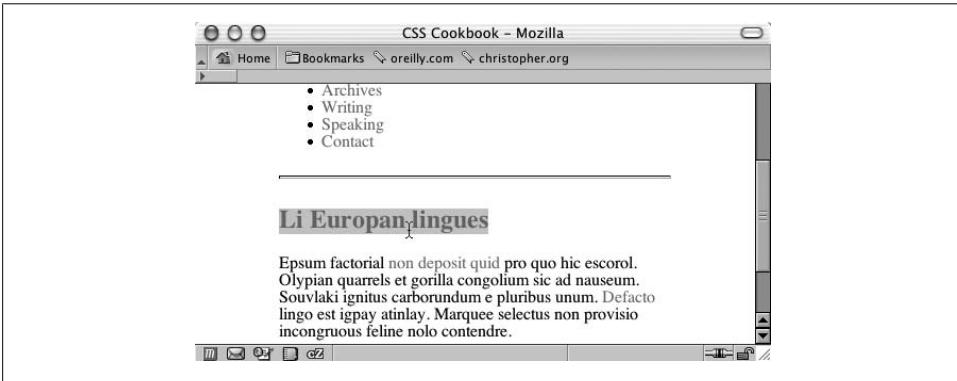

Figure 7-7. An unwanted rollover effect on a heading

However, it's recommended that instead of using `name` attributes within links, use `id` attributes for linking within a document:

```
<h2 id="europan">Li Europan lingues</h2>
```

See Also

The CSS 2.1 specification for `:active` and `:hover` at *http://www.w3.org/TR/CSS21/selector.html#x36*; an explanation about links and specificity at *http://www.meyerweb.com/eric/css/link-specificity.html*

7.9 Animating Rollovers on Links with CSS3 Transitions

Problem

You want to adjust the time a rollover effect takes on a link.

Solution

Use CSS3 `transition` properties to set an animation on the rollover effects, as shown in Figure 7-8:

```
#navsite a {
 -webkit-transition-timing-function: linear;
 -webkit-transition-duration: .66s;
 -webkit-transition-property: background-color;
```

```
}
a:link {
  color: #777;
  text-decoration: none;
}
a:visited {
  color: #333;
  text-decoration: none;
}
a:link:hover, a:visited:hover {
  color: #777;
  background-color: #ccc;
}
a:link:active, a:visited:active {
  color: #ccc;
  background-color: #ccc;
}
```
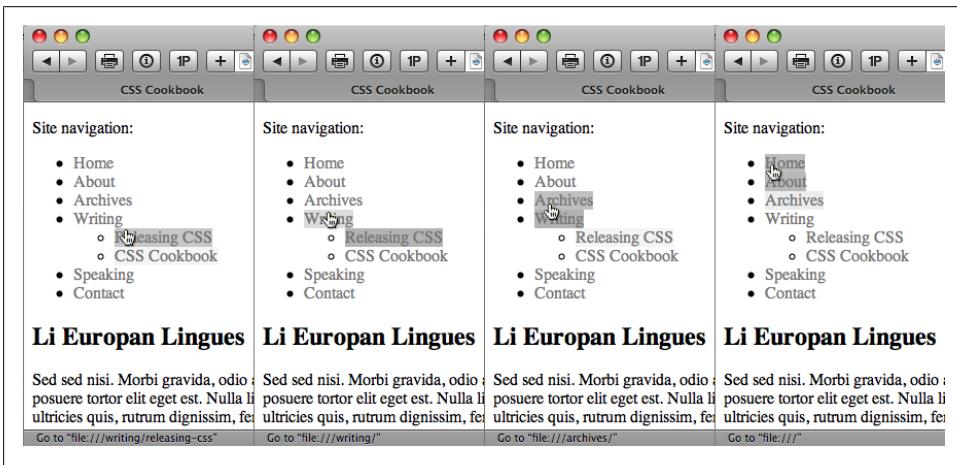

Figure 7-8. The background color fading out as the cursor glides over links

Discussion

The `transition` property is a new addition to the CSS specification introduced by Apple's Safari browser developers, and as of this writing it is supported only by the Safari browser. Use of the `transition` property within Safari requires the browser proprietary extension of `-webkit-` for the effect to work.

Timing function

The `transitioning-timing-function` function states the type of animation the effect is going to take. In the Solution, the value is set to `linear`, which means each frame of the animation length takes the same amount of time.

Other values for `transitioning-timing-function` include `ease`, `ease-in`, `ease-out`, and `cubic-bezier(x1, y1, x2, y2)`.

The first two values of `cubic-bezier` represent the transition on a curve, as shown in Figure 7-9. Values for `y1` and `y2` represent the start and end of the transition and are always equal to the values of 0.0 and 1.0, respectively. The speed with which the transition takes hold is represented by the values `x1` and `x2`. The greater the value for `x1` and `x2`, the slower the transition occurs.
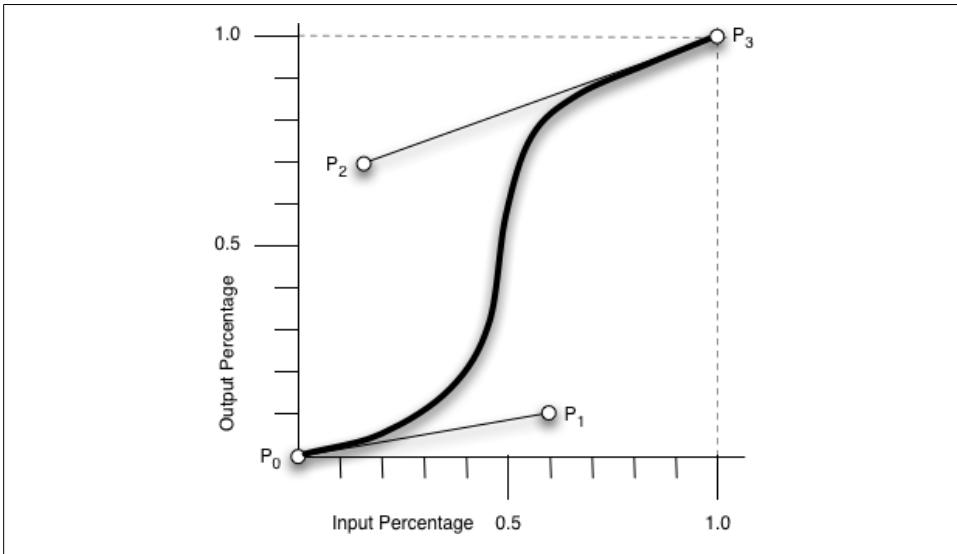
Figure 7-9. A mathematical representation of transitions (source, W3C: http://www.w3.org/TR/css3 -transitions/#transition-timing-function_tag)

A value of `ease-in` starts the transition at a slow speed and then speeds up. This value is equivalent to `cubic-bezier(0.42, 0, 1.0, 1)`.

The `ease-out` value starts the transition at a fast speed and then slows down. This value is equivalent to `cubic-bezier(0.42, 0, 1.0, 1)`.

The `ease` value is equivalent to `cubic-bezier(0.25, 0.1, 0.25, 1.0)`.

Duration and delay

The `transition-duration` property's default value is `0`. Any negative value is treated as though it's zero. Units the value may take include, but are not limited to, `s` for seconds and `ms` for milliseconds.

The `transition-delay` property sets the amount of time before a transition starts.

Transition property

The `transition-property` property defines which CSS visual property the transition is applied to. In this Solution, the transition is applied to the background color.

Shorthand property

You can write the properties of the transition effect in one value for the `transition` property. For example, you can shorten the `transition` properties in the Solution as follows:

```
#navsite a {
 -webkit-transition: background-color .66s linear;
}
```

Extending transitions to other properties

Web designer Faruk Ateş's personal site (see *http://farukat.es/*), as shown in Figure 7-10, uses the `transition` property to change the color of the element, as well as the color, width, box shadow, text shadow, *and* opacity:

```
#web20 li a {
 background: transparent url(icons.png) 100% 0 no-repeat;
 border: none;
 color: #848484;
 display: block;
 font-size: 11px;
 font-weight: normal;
 height: 63px;
 left: -19px;
 line-height: 18px;
 margin: 3px 0;
 padding-left: 19px;
 position: relative;
 text-decoration: none;
 width: 220px;
 text-shadow: rgba(0,0,0, 0) 1px 1px 2px;
 -moz-box-shadow: rgba(0,0,0, 0) 2px 2px 2px;
 -webkit-box-shadow: rgba(0,0,0, 0) 2px 2px 2px;
 -moz-border-radius-topright: 31px;
 -moz-border-radius-bottomright: 31px;
 -webkit-border-top-right-radius: 31px;
 -webkit-border-bottom-right-radius: 31px;
 -webkit-transition: background-color .25s ease,
   color .5s ease,
   width .2s ease-out,
   -webkit-box-shadow .25s ease,
   text-shadow .2s ease,
   opacity .2s ease;
}
```

See Also

The Surfin' Safari blog post introducing transitions at *http://webkit.org/blog/138/css -animation/*; the CSS3 specification for transitions at *http://www.w3.org/TR/css3-tran sitions/#transitions-*
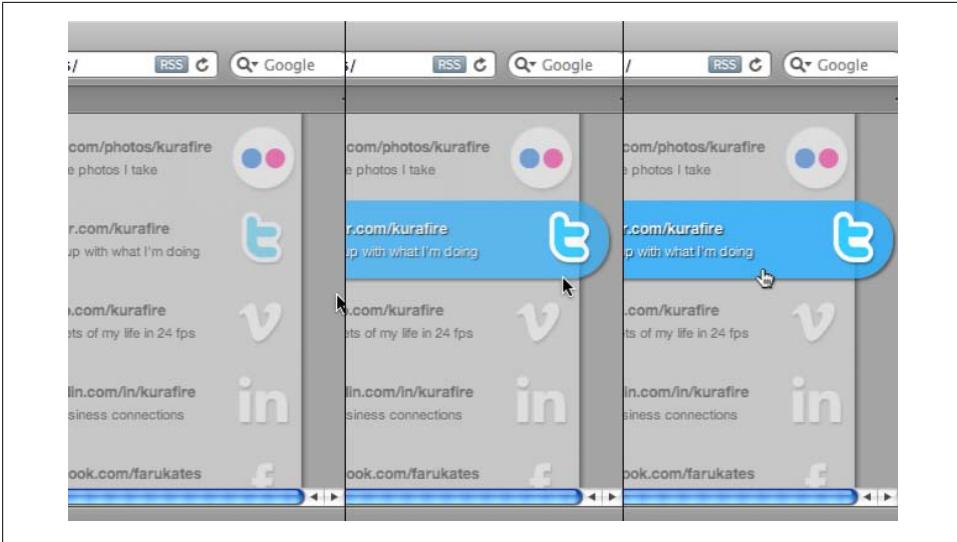
Figure 7-10. A complex CSS-enabled rollover effect

7.10 Creating Text Navigation Menus and Rollovers

Problem

You have a list of links, but you want to build an elegant menu, as shown in Figure 7-11.

Download at WoweBook.com
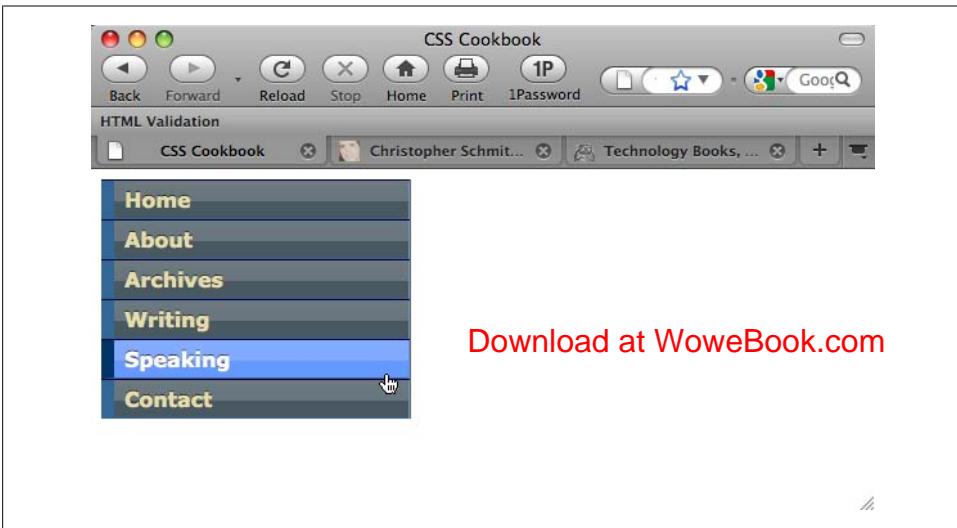
Figure 7-11. A set of stylized links

Solution

First, mark up the list of links in an unordered list so that they wrap around a `div` element with an `id` attribute:

```
<div id="navsite">
 <p>Site navigation:</p>
 <ul>
  <li><a href="/">Home</a></li>
  <li><a href="/about/">About</a></li>
  <li><a href="/archives/">Archives</a></li>
  <li><a href="/writing/">Writing</a></li>
  <li><a href="/speaking/">Speaking</a></li>
  <li><a href="/contact/">Contact</a></li>
 </ul>
</div>
```

Next, use the `border` property on the anchor elements to create the bulk of the design:

```
#navsite p {
 display: none;
}
a {
 text-shadow: 0 -1px 0px rgba(0,0,0,.8);
}
#navsite {
 font-family: Verdana, Helvetica, Arial, sans-serif;
 font-size: 1em;
 font-weight: bold;
 border-right: 1px solid #666;
 padding: 0;
 margin-bottom: 1em;
 color: #333;
 width: 15em;
}
#navsite ul {
 list-style: none;
 margin: 0;
 padding: 0;
}
#navsite ul li {
 margin: 0;
 border-top: 1px solid #003;
}
#navsite ul li a:link, #navsite ul li a:visited {
 display: block;
 padding: 4px 4px 4px 0.5em;
 border-left: 10px solid #369;
 border-right: 1px solid #69c;
 border-bottom: 1px solid #369;
 color: #E5DEAC;
 text-decoration: none;
 background-color: #495961;
 background-image: url(title-glass.png);
 background-position: 50%;
 background-repeat: repeat-x;
```

```
}
html>body #navsite ul li a {
 width: auto;
}
#navsite ul li a:hover {
 border-left: 10px solid #036;
 border-right: 1px solid #69c;
 border-bottom: 1px solid #369;
 background-color: #69f;
 color: #fff;
}
```

Discussion

A menu makes it easier for visitors to navigate your site. To help the user find the navigation menu, stylize the menu links so that they stand out from the regular text. Do this by using the `id` selector when writing the CSS rules. As the Solution shows, successfully creating the menu requires some browser bug workarounds as well as straightforward CSS design implementation.

Setting up the list

In the division marked with the `div`, a line of text labels the set of links as navigational links:

```
<p>Site navigation:</p>
```

If the user's browser doesn't have CSS support, the line of text is visible. To hide the text from CSS-enabled browsers, set `display` to `none`:

```
#navsite p {
 display: none;
}
```

> I prefer to mark the division of the web document and label the navigation. You might prefer or recommend another method. Some web developers choose to forego the use of the `div` element wrapper and `p` element as a label and simply apply the `id` value on the `ul` element. Then they adjust the CSS rules on the unordered list.

The next step is to stylize the `div` element that encapsulates the set of menu links. In this CSS rule, styles are set for the links to inherit properties set on the `div` element. Also, set the values of the `width`, `border-right`, `padding`, and `margin-bottom` properties to keep the menu from bunching up:

```
#navsite {
 font-family: Verdana, Helvetica, Arial, sans-serif;
 font-size: 1em;
 font-weight: bold;
 border-right: 1px solid #666;
 padding: 0;
```

```
      margin-bottom: 1em;
      color: #333;
      width: 15em;
    }
```

The next CSS rule eliminates any potential problems with the indentation of lists (see Recipe 4.2) by setting the `margin` and `padding` to `0` as well as by eliminating any list markers:

```
    #navsite ul {
      list-style: none;
      margin: 0;
      padding: 0;
    }
```

In the following rule you're making sure margins aren't applied to each list item. This CSS rule also places a 1-pixel border at the top of the list item. This design element helps to reinforce the separation of the list items:

```
    #navsite ul li {
      margin: 0;
      border-top: 1px solid #003;
    }
```

Styling the links

The next rule sets the styles for the links. By default, links are inline elements. The links need to be rendered as block-level elements so that the entire part of the "link design" becomes clickable, and not just the text. Setting the `display` property to `block` accomplishes this transformation.

Use the following declarations to stylize the appearance of the borders, text color, text decoration, and width:

```
    #navsite ul li a:link, #navsite ul li a:visited {
      display: block;
      padding: 4px 4px 4px 0.5em;
      border-left: 10px solid #369;
      border-right: 1px solid #69c;
      border-bottom: 1px solid #369;
      color: #E5DEAC;
      text-decoration: none;
      background-color: #495961;
      background-image: url(title-glass.png);
      background-position: 50%;
      background-repeat: repeat-x;
    }
```

The final declaration for the links sets the width at 100%. This rule was set to make sure Internet Explorer for Windows makes the entire area clickable. The drawback with this rule is that it causes problems in older browsers such as IE Mac and Netscape Navigator (which you may have to support). To work around this problem, use the child selector (which IE7 can't process) to reset the width of the link:

```
html>body #navsite ul li a {
 width: auto;
}
```

The last CSS rule states the styles for the rollover effect of the links:

```
#navsite ul li a:hover {
 border-left: 10px solid #036;
 border-right: 1px solid #69c;
 border-bottom: 1px solid #369;
 background-color: #69f;
 color: #fff;
}
```

Unordered lists for navigation

An unordered list is a perfect way to structure a menu of links both in theory and in practical application. On the one hand, a set of links *is* a set of unordered items. And using unordered lists for navigation creates a solid structure for your web document based on both logic and semantically correct markup.

On the other hand, with the links set in an unordered list, it's easier to style the links into a menu presentation than it is to style a series of `div` elements:

```
<div id="navsite">
<p>Site navigation:</p>
<div><a href="/">Home</a></div>
<div><a href="/about/">About</a></div>
<div><a href="/archives/">Archives</a></div>
<div><a href="/writing/">Writing</a></div>
<div><a href="/speaking/">Speaking</a></div>
<div><a href="/contact/">Contact</a></div>
</div>
```

See Also

The article "CSS Design: Taming Lists" by Mark Newhouse at *http://www.alistapart .com/articles/taminglists/*; the article/tutorial "Semantics, HTML, XHTML, and Structure" by Shirley E. Kaiser at *http://brainstormsandraves.com/articles/semantics/structure/*

7.11 Adding Submenus to Vertical Menus

Problem

You want to add an additional menu below the main set of navigation links, as shown in Figure 7-12.

Figure 7-12. A submenu added to the vertical menu

Solution

Expand the main navigation menu to include a nested unordered list within the appropriate list item. In the following example, two publications were placed within the Writing list item:

```
<div id="navsite">
 <p>Site navigation:</p>
 <ul>
  <li><a href="/">Home</a></li>
  <li><a href="/about/">About</a></li>
  <li><a href="/archives/">Archives</a></li>
  <li><a href="/writing/">Writing</a>
   <ul>
    <li><a href="/writing/releasing-css">Releasing CSS</a></li>
    <li><a href="/writing/css-cookbook">CSS Cookbook</a></li>
```

```
    </ul>
   </li>
   <li><a href="/speaking/">Speaking</a></li>
   <li><a href="/contact/">Contact</a></li>
  </ul>
</div>
```

Apply an amount of margin to the left side of the nested unordered list through a descendant selector:

```
#navsite ul ul {
 background-color: white;
 margin-left: 10px;
}
```

Then style the links and rollover effects, as shown earlier in Figure 7-12:

```
#navsite ul ul li a:link, #navsite ul ul li a:visited  {
 border-left: 10px solid #69c;
 border-right: 1px solid #9cF;
 border-bottom: 1px solid #69c;
 background-color: #888;
}
#navsite ul li a:hover, #navsite ul ul li a:hover {
 border-left: 10px solid #036;
 border-right: 1px solid #69c;
 border-bottom: 1px solid #369;
 background-color: #69f;
 color: #fff;
}
```

Discussion

Using descendant selectors (see Recipe 2.2) allows you to pinpoint styles to nested links. By indenting the list on the margin, you make the links appear to be tucked under their parent link.

Limiting styles

The main drawback to using descendant selectors for this Solution is that if there are multiple nested lists (e.g., a list within a list within yet another list), the descendant selectors are also applied to the third nested list. So, the same look and feel is applied even as you apply more nested lists.

One way to solve this problem is to use a child selector to limit the styles from styling multiple nested lists:

```
#navsite ul li > ul li a:link, #navsite ul li > ul li a:visited {
 border-left: 10px solid #69c;
 border-right: 1px solid #9cF;
 border-bottom: 1px solid #69c;
 background-color: #888;
}
#navsite ul li a:hover, #navsite ul li > ul li a:hover {
 border-left: 10px solid #036;
 border-right: 1px solid #69c;
 border-bottom: 1px solid #369;
 background-color: #69f;
 color: #fff;
}
```

See Also

The Listamatic website for numerous examples and code of navigation menus at *http: //css.maxdesign.com.au/listamatic/*

7.12 Building Horizontal Navigation Menus

Problem

You want to create a horizontal navigation menu out of an unordered set of links; Figure 7-13 shows the default.
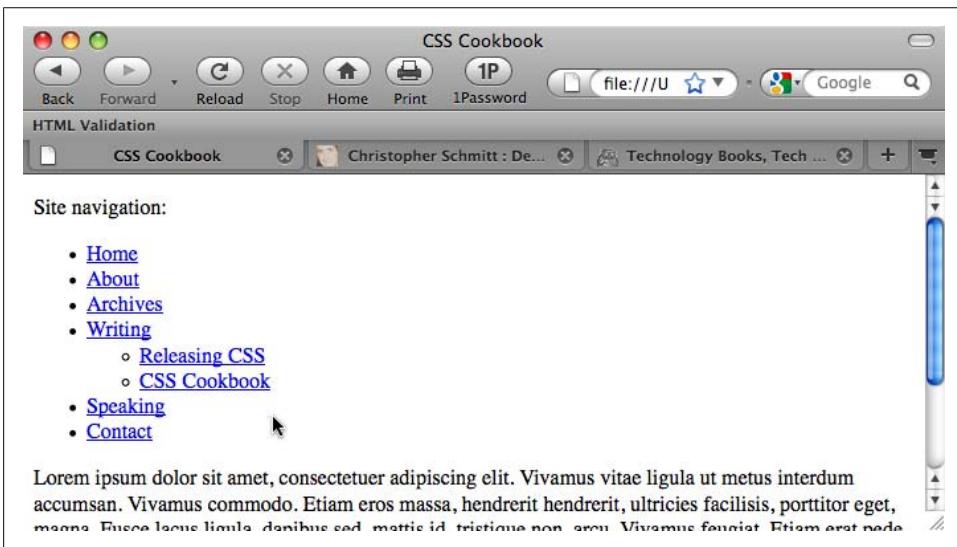
Figure 7-13. The default appearance of the links

Solution

First, create a properly constructed set of unordered links:

```
<div id="navsite" class="clearfix">
<p>Site navigation:</p>
<ul>
 <li><a href="/">Home</a></li>
 <li><a href="/about/">About</a></li>
 <li><a href="/archives/">Archives</a></li>
 <li><a href="/writing/">Writing</a></li>
 <li><a href="/speaking/" id="current">Speaking</a></li>
 <li><a href="/contact/">Contact</a></li>
</ul>
</div>
```

Then set the CSS rules for the navigation structure, making sure to adjust the list items to float, as shown in Figure 7-14:

```
#navsite p {
 display: none;
}
#navsite ul {
     width: 100%;
     float: left;
 padding: 0;
 margin-left: 0;
 border-bottom: 1px solid #778;
 font: bold 12px Verdana, sans-serif;
}
#navsite ul li {
 list-style: none;
 margin: 0;
 float: left;
}
#navsite ul li a {
 padding: 12px 0.5em;
 margin-left: 3px;
 border: 1px solid #778;
 border-bottom: none;
 background-color: #666;
 text-decoration: none;
 background-image: url(title-glass.png);
 background-position: 50%;
 background-repeat: repeat-x;
 display: block;
 width: 7em;
}
#navsite ul li a:link {
 color: white;
}
#navsite ul li a:visited {
 color: #667;
}
#navsite ul li a:link:hover, #navsite ul li a:visited:hover {
 color: #000;
```

```
 background-color: #aae;
 border-color: #227;
}
#navsite ul li a#current {
 background-color: white;
 border-bottom: 1px solid white;
color: #448;
margin-bottom: -1px;
}
#navsite ul li a#current:hover {
 background-image: url(title-glass.png);
 background-position: 50%;
 background-repeat: repeat-x;
}
.clearfix:after {
 content: ".";
 display: block;
 height: 0;
 clear: both;
 visibility: hidden;
}
/* for IE6 */
* html .clearfix {
 height: 1%;
}
/* for IE7 */
*:first-child+html .clearfix {
 min-height: 1px;
}
```
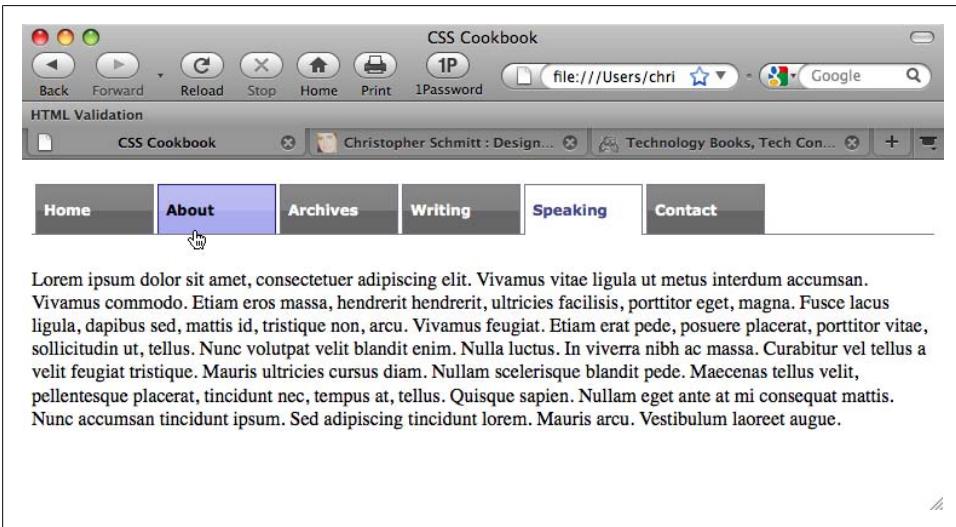

Figure 7-14. The tab-based navigation

Discussion

The first part of the Solution hides the heading. This is done because the visual representation of the tab navigation design is enough to inform users that these are navigation links:

```
#navsite p {
  display: none;
}
```

The next rule defines the padding and margin for the box that is created by the unordered list element, `ul`. The line that stretches across the bottom of the folder tabs is drawn by the `border-bottom` property (see Figure 7-15):

```
#navsite ul {
  width: 100%;
  float: left;
  padding: 0;
  margin-left: 0;
  border-bottom: 1px solid #778;
  font: bold 12px Verdana, sans-serif;
}
```
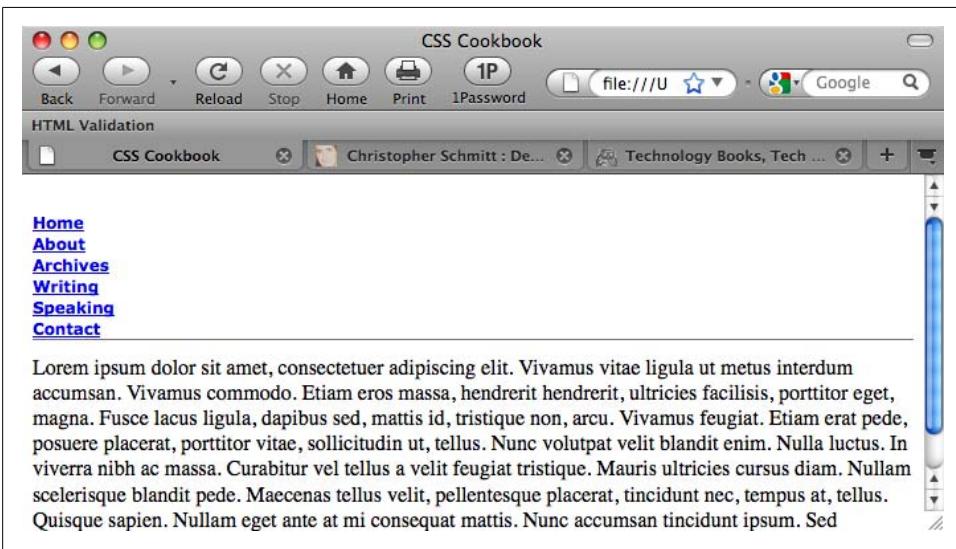

Figure 7-15. The line the navigation tabs rest upon

The declaration that makes this horizontal navigation work with the unordered list is `float` for the list items:

```
#navsite ul li {
  list-style: none;
  margin: 0;
  float: left;
}
```

Another method for building horizontal menus is to use the `inline` property. Although both approaches do obtain a horizontal appearance, by setting a list item to `float`, you still retain the block-level properties of the element, allowing you to set properties such as `margin` and `width` for consistent menu tags. If you don't want consistent tab widths, or if you prefer the spacing between the menu labels to be consistent, the `inline` property is an acceptable alternative with the `padding-left` or `padding-right` property.

Instead of stacking the list items on top of each other by default, the browser now lays out the list items as it would text, images, and other inline elements (see Figure 7-16).
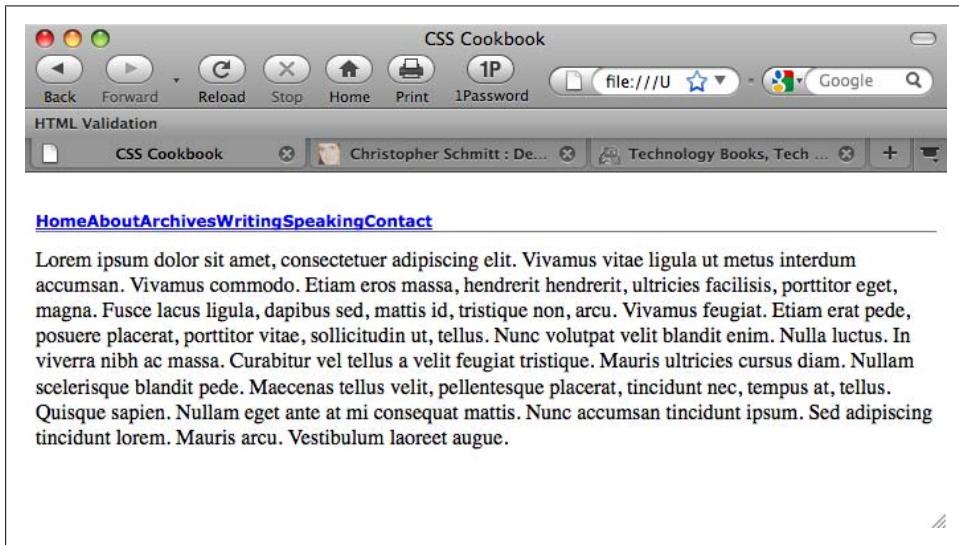
Figure 7-16. The list spread out horizontally

To create the look of the folder tab, use the `border` property in the following CSS rule:

```css
#navsite ul li a {
  padding: 12px 0.5em;
  margin-left: 3px;
  border: 1px solid #778;
  border-bottom: none;
  background-color: #666;
  text-decoration: none;
  background-image: url(title-glass.png);
  background-position: 50%;
  background-repeat: repeat-x;
  display: block;
  width: 7em;
}
```

The first `border` property is a shorthand property that dictates a solid, 1-pixel border around the link. However, immediately following the `border` property is the `border-bottom` property, which tells the browser not to display a border beneath the link.

The value of the `border-bottom` property is displayed over the `border` shorthand property (as shown in Figure 7-17). This overwriting occurs because the `border-bottom` declaration overrides the values in the border declaration because of the order in which they are declared.
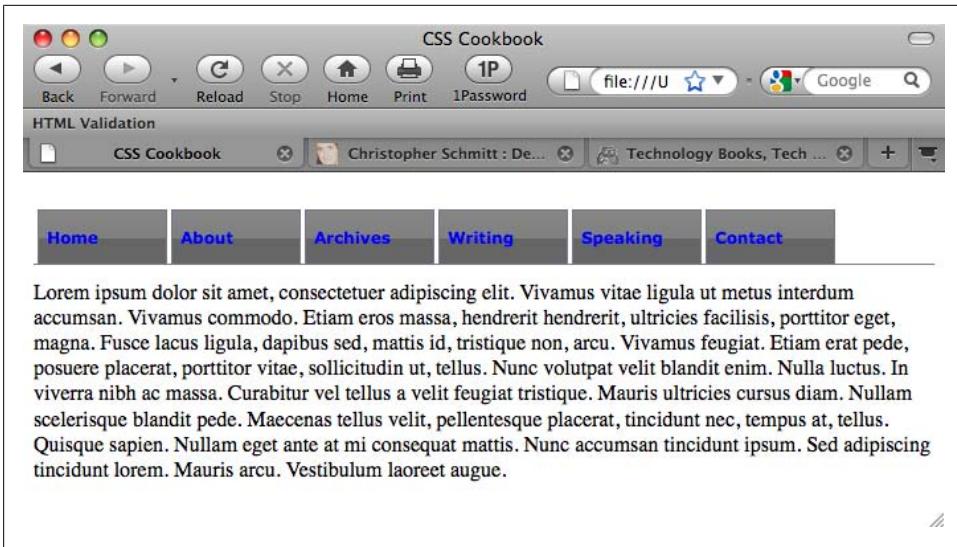
Figure 7-17. Visible tabs

After you've created the look of the border tab, set the color of the text links and rollover states:

```
#navsite ul li a:link {
 color: white;
}
#navsite ul li a:visited {
 color: #667;
}
#navsite ul li a:link:hover, #navsite ul li a:visited:hover {
 color: #000;
 background-color: #aae;
 border-color: #227;
}
```

The final CSS rule defines how the "current" link appears. This style is applied to the link that represents the page being viewed by the user (see Figure 7-18):

```
#navsite ul li a#current {
 background: white;
```

```
  border-bottom: 1px solid white;
}
```
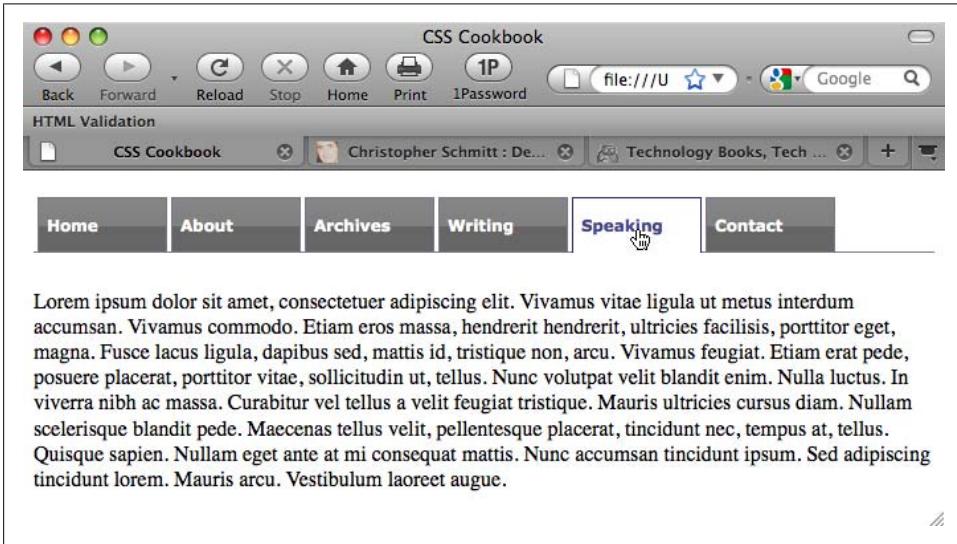

Figure 7-18. The look of the current link

Next, add a self-clearing float (see Recipe 2.22) to the entire navigation menu on the **div** element. This makes sure that any text or content in the rest of the page doesn't wrap around the menu navigation:

```
.clearfix:after {
 content: ".";
 display: block;
 height: 0;
 clear: both;
 visibility: hidden;
}
/* for IE6 */
* html .clearfix {
 height: 1%;
}
/* for IE7 */
*:first-child+html .clearfix {
 min-height: 1px;
}
```

See Also

The original tab menu bar (as well as other navigation styles) at *http://css.maxdesign .com.au/listamatic/horizontal05.htm*

7.13 Building Horizontal Navigation Menus with Drop-Down Menus

Problem

You want to add a drop-down menu to a horizontal navigation menu, as shown in Figure 7-19.
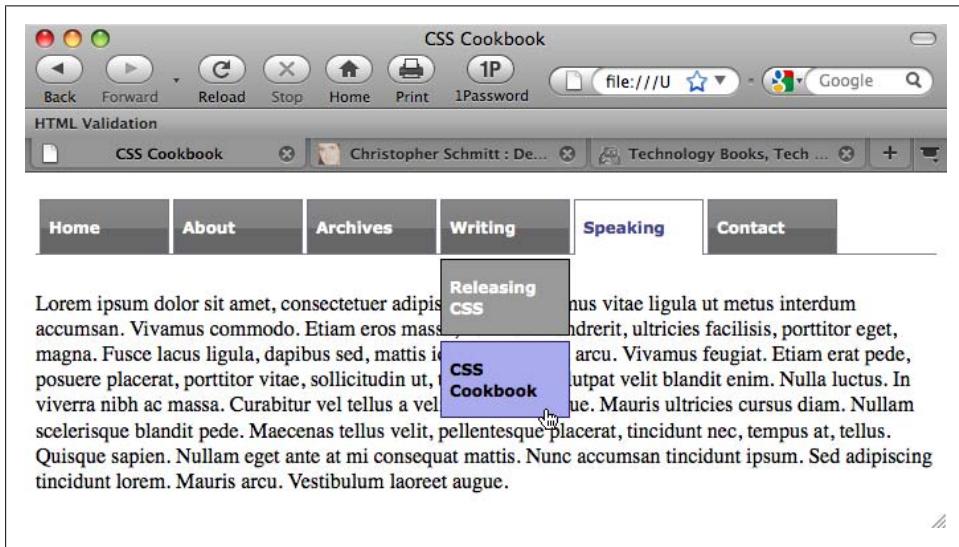
Figure 7-19. The look of the current link

Solution

As in Recipe 7.11, expand the main navigation menu to include a nested unordered list within the appropriate list item:

```
<div id="navsite">
 <p>Site navigation:</p>
 <ul>
  <li><a href="/">Home</a></li>
  <li><a href="/about/">About</a></li>
  <li><a href="/archives/">Archives</a></li>
  <li><a href="/writing/">Writing</a>
   <ul>
    <li><a href="/writing/releasing-css">Releasing CSS</a></li>
    <li><a href="/writing/css-cookbook">CSS Cookbook</a></li>
   </ul>
  </li>
  <li><a href="/speaking/">Speaking</a></li>
  <li><a href="/contact/">Contact</a></li>
```

```
    </ul>
  </div>
```

Then, using descendant selectors to pinpoint styles to the nested unordered list, set a very low negative value for the position of the drop-down menu options:

```
#navsite ul li ul {
 position: absolute;
 width: 7em;
 left: -999em;
 float: none;
 border-bottom: none;
}
```

Use the `:hover` pseudo-class to bring them back:

```
#navsite ul li:hover ul {
 left: auto;
}
#navsite ul li ul li a {
 background: #999;
 border: 1px solid black;
 margin-top: 4px;
}
```

Discussion

This Solution works well in modern browsers. However, for cross-browser support you can use an additional piece of JavaScript and modified CSS, called Son of Suckerfish Dropdowns (see *http://htmldog.com/articles/suckerfish/dropdowns/*).

An easy method for converting Son of Suckerfish Dropdowns to simple jQuery functions is available at *http://nederdev.com/articles/suckerfish-meets-jquery*.

Drop-down menus over Flash

If you have a drop-down menu appearing over Flash in a web document, sometimes the drop-down menu may not appear or may become hidden behind the Flash movie. To work around this behavior, try setting the `wmode` parameter to `transparent` to allow the drop-down menu to appear:

```
<object classid="clsid:D27CDB6E-AE6D-11cf-96B8-444553540000"
codebase="http://download.macromedia.com/" width="190" height="290">
 <param name="movie" value="flash/file.swf">
 <param name="quality" value="high">
 <param name="wmode" value="transparent">
 <embed src="flash/file.swf" width="190" height="290" quality="high"
pluginspage="http://www.macromedia.com/go/getflashplayer"
type="application/x-shockwave-flash" wmode="transparent"></embed>
</object>
```

See Also

The Listamatic website for numerous examples and code of navigation menus at *http://css.maxdesign.com.au/listamatic/*

7.14 Building a Navigation Menu with Access Keys

Problem

You want to create a navigation menu with access keys.

Solution

Create a set of unordered links with `accesskey` within the anchor elements:

```
<div id="navsite">
 <ul>
  <li><a href="/" accesskey="h">Home</a></li>
  <li><a href="/about/" accesskey="b">About</a></li>
  <li><a href="/archives/" accesskey="a">Archives</a></li>
  <li><a href="/writing/" accesskey="w">Writing</a></li>
  <li><a href="/speaking/" accesskey="s">Speaking</a></li>
  <li><a href="/contact/" accesskey="c">Contact</a></li>
 </ul>
</div>
```

Next, add `span` elements around the letters you want to identify as access keys:

```
<div id="navsite">
 <ul>
  <li><a href="/" accesskey="h"><span class="akey">H</span>ome</a></li>
  <li><a href="/about/" accesskey="b">A<span class="akey">b</span>out</a></li>
  <li><a href="/archives/" accesskey="a"><span class="akey">A</span>rchives</a></li>
  <li><a href="/writing/" accesskey="w"><span class="akey">W</span>riting</a></li>
  <li><a href="/speaking/" accesskey="s"><span class="akey">S</span>peaking</a></li>
  <li><a href="/contact/" accesskey="c"><span class="akey">C</span>ontact</a></li>
 </ul>
</div>
```

Then, style the access keys through a class selector, as shown in Figure 7-20:

```
.akey {
 text-decoration: underline;
}
```

Discussion

Access keys allow site visitors to navigate a website easily without the use of a mouse. In the Solution, access keys were assigned to the navigation elements. Once the user presses a key, he will navigate to the page specified in the link.
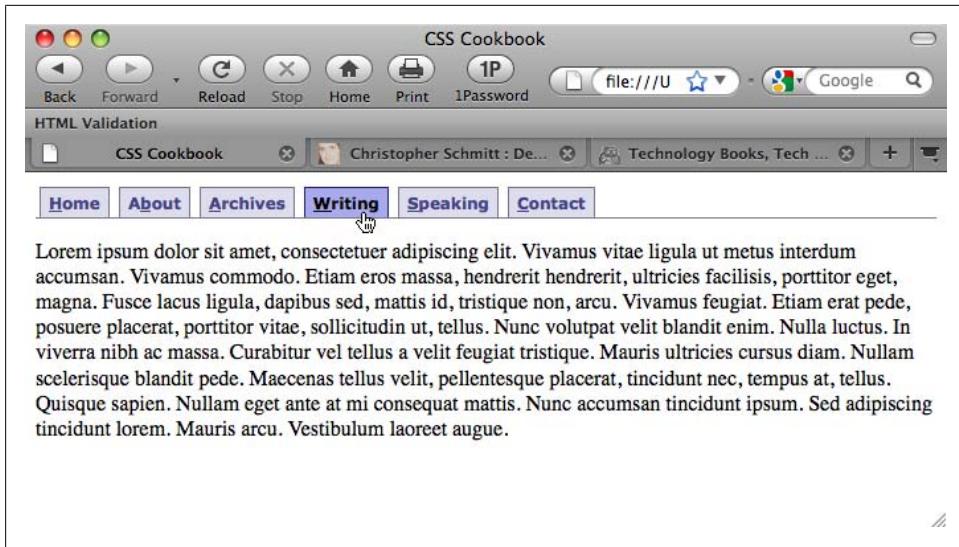
Figure 7-20. The look of the current link

If access keys are used consistently, a site visitor may use the same set of access keys to navigate, to create a cohesive user experience.

Known browser issues

Access keys are supposed to work in IE4 and later, Firefox, Safari, Chrome, and Opera 7 and later.

One of the obstacles of access keys is that there isn't a standard set of keys associated with each link—for example, would using the letter *h* be better for "Home Page" (as done in this example), or would the letter *m* be better to represent "Main Page"?

See Also

The HTML4 specification for access keys at *http://www.w3.org/TR/html4/interact/ forms.html#h-17.11.2*; the article "Accesskeys: Unlocking Hidden Navigation" by Stuart Robertson at *http://alistapart.com/articles/accesskeys/*

7.15 Creating Breadcrumb Navigation

Problem

You want to use a nested list, as shown in Figure 7-21, to create a line of breadcrumb navigation, which is a set of links that lead back to the home page (see Figure 7-22).

Figure 7-21. The default rendering of the nested list

Figure 7-22. The breadcrumb trail

Solution

The first step is to create a properly constructed set of nested, unordered links that represent the page's location in the site:

```
<div id="crumbs">
 <h3>Location:</h3>
 <ul>
 <li><a href="/">Home</a>
  <ul>
   <li><a href="/writing/">Writing</a>
    <ul>
     <li><a href="/writing/books/">Books</a>
      <ul>
       <li><a href="/writing/books/">CSS Cookbook</a></li>
      </ul>
     </li>
    </ul>
   </li>
  </ul>
 </li>
 </ul>
</div>
```

Now set the `display` property of both the `ul` and the `li` of the lists:

```
#crumbs {
 background-color: #eee;
 padding: 4px;
}
#crumbs h3 {
 display: none;
}
#crumbs ul {
 display: inline;
 padding-left: 0;
 margin-left: 0;
}
#crumbs ul li {
 display: inline;
}
#crumbs ul li a:link {
 padding: .2em;
}
```

Within each nested list, place a small background image of an arrow to the left of the link:

```
#crumbs ul ul li{
 background-image: url(arrow_r.gif);
 background-repeat: no-repeat;
 background-position: left;
 padding-left: 20px;
}
```

Discussion

Based on the fairy tale *Hansel and Gretel*, a *breadcrumb trail* is used to help people find their way home. On the Web, the breadcrumb trail illustrates a path to the page the user is viewing (as shown in Figure 7-23).
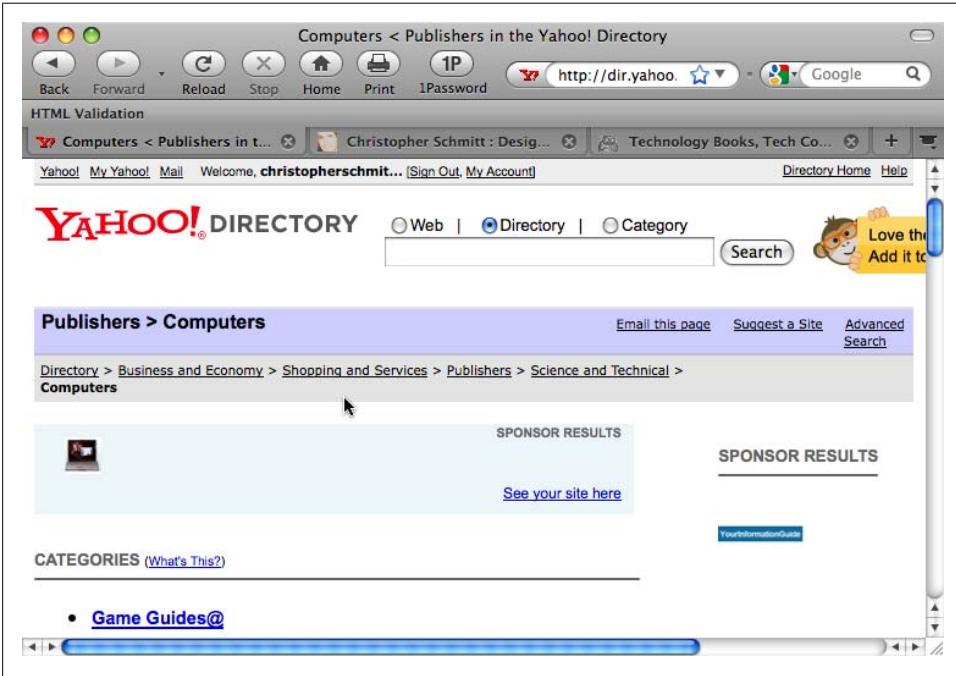
Figure 7-23. An example of a breadcrumb trail

The Solution could drop the `background-image` property if more browsers supported the `:before` pseudo-element. The Solution would then incorporate another CSS rule, like so:

```
#crumbs ul ul li:before {
 content: url(arrow.gif);
}
```

As of this writing, all the major browsers support the `:before` pseudo-element, except for IE7 and earlier versions.

See Also

An annotated version of *Hansel and Gretel* at *http://www.surlalunefairytales.com/han selgretel/index.html*; a research paper on the effectiveness of breadcrumb navigation at *http://psychology.wichita.edu/surl/usabilitynews/52/breadcrumb.htm*

7.16 Creating Image-Based Rollovers

Problem

You want image-based rollovers to replace text links.

Solution

First, wrap the text inside the anchor element in a `span`:

```
<a href="/" id="linkhome">
 <span>Homepage</span>
</a>
```

Next, instead of JavaScript, use the `background-image` property within the `:hover` and `:active` pseudo-class selectors to swap the images (see Figure 7-24):

```
a span {
 display: none;
}
a:link {
 display: block;
 width: 100px;
 height: 50px;
 background-image: url(submit.png);
 background-repeat: no-repeat;
 background-position: top left;
}
a:link:hover {
 display: block;
 width: 100px;
 height: 50px;
 background-image: url(submit-roll.png);
 background-repeat: no-repeat;
 background-position: top left;
}
a:link:active {
 display: block;
 width: 100px;
 height: 50px;
 background-image: url(submit-on.png);
 background-repeat: no-repeat;
 background-position: top left;
}
```

Discussion

Replacing text with an image has five benefits. First, it separates the text from the presentation. The image that contains more elaborately formatted type is part of the presentation, and therefore is controlled by a style, while the content in the markup remains pure text. The second benefit is that an image heading can be modified across
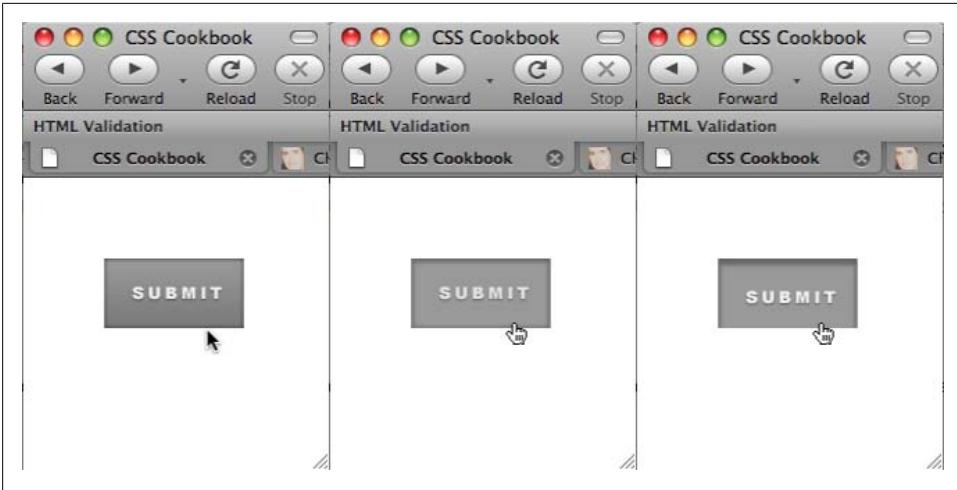
Figure 7-24. The link with default, rollover, and active states

a whole site by one change of the stylesheet. The third benefit is that this method works for alternative styles and stylesheet switching.

With a span element inside an element, it is possible to hide HTML text and let a design element, such as a rollover image, show as a background image. The fourth benefit of this Solution is that if a user doesn't have CSS enabled in his browser, the default HTML text will display instead, sparing the user from having to download unneeded images. The fifth benefit is that the Solution is cleaner and simpler than one that involves JavaScript.

You also can use this technique for page elements that don't require a rollover—for example, inserting an image to replace heading text to ensure that a specific font that isn't commonly found on people's computers is displayed as an image. To do so, first set up the markup (see Figure 7-25):

```
<h2 id="headworld"><span>Hello, World!</span></h2>
```
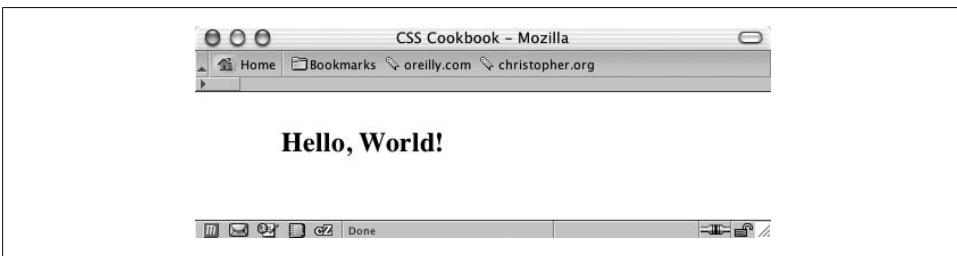

Figure 7-25. Default rendering of the heading

Then set the following CSS rules to insert the image (see Figure 7-26):

```
h2#headworld span {
 display: none;
}
h2#headworld {
 width: 395px;
 height: 95px;
 background-image: url(heading.gif);
 background-repeat: no-repeat;
 background-position: top left;
}
```
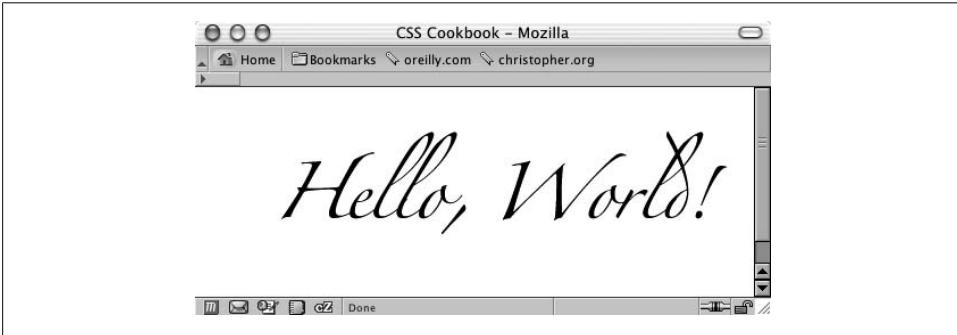

Figure 7-26. Replacing the text with an image

Many people refer to this method as the Fahrner Image Replacement (FIR) method, named after Todd Fahrner (see Recipe 4.20).

A drawback to this Solution concerns *screen readers*, which are programs that make computers accessible to blind or severely vision-impaired people. Certain screen readers won't read elements set to `display: none`. For more information, read the article "Facts and Opinion About Fahrner Image Replacement" at *http://www.alistapart.com/articles/fir/*.

Leahy-Langridge image replacement

An alternative to this solution is the Leahy-Langridge Image Replacement (LIR) method. Developed independently by Seamus Leahy and Stuart Langridge, the LIR method pushes the text out of view. A benefit of using this technique is that an extra `span` element isn't required to hide the text. For example, the HTML for a heading is basic:

```
<h2 id="headworld">Hello, World!</h2>
```

The image for the heading comes through the background because the CSS rule sets the padding to the exact height of the image header. So, the `height` property is set to `0`:

```
h2#headworld {
 /* The width of the image */
 width: 395px;
 /* The height of the image is the first padding value */
```

```
    padding: 95px 0 0 0;
    overflow: hidden;
    background-image: url("heading.gif");
    background-repeat: no-repeat;
    voice-family: "\"}\"";
    voice-family:inherit;
    height /**/: 95px;
    height: 0px !important;
}
```

The last four lines of the CSS rule are needed to work around IE7 and its previous versions' poor box model support (see Recipe 2.10). Therefore, the older versions of IE get a height value of 95 pixels, while the other browsers receive zero pixels.

> Another method is to use conditional comments to deliver specific values for IE browsers. For more information, see Recipe 12.7.

Pixy method

Another method for creating an image-based rollover is performed by the `background-position` property. Known as the Pixy method (also referred to as *CSS sprites* as written in Recipe 4.32), the technique involves attaching all three rollover states into one image and then moving the position of the image with the `background-position` property, as shown in Figure 7-27:

```
a span {
 display: none;
}
a:link, a:visited {
 display: block;
 width: 125px;
 height: 30px;
 background-image: url(btn_omni.gif);
 background-repeat: no-repeat;
 background-position: 0 0;
}
a:link:hover, a:visited:hover {
 display: block;
 width: 125px;
 height: 30px;
 background-image: url(btn_omni.gif);
 background-repeat: no-repeat;
 /* move the image 30 pixels up */
 background-position: 0 -30px;
}
a:link:active, a:visited:active  {
 display: block;
 width: 125px;
 height: 30px;
 background-image: url(btn_omni.gif);
 background-repeat: no-repeat;
```

```
      /* move the image 60 pixels up */
      background-position: 0 -60px;
    }
```
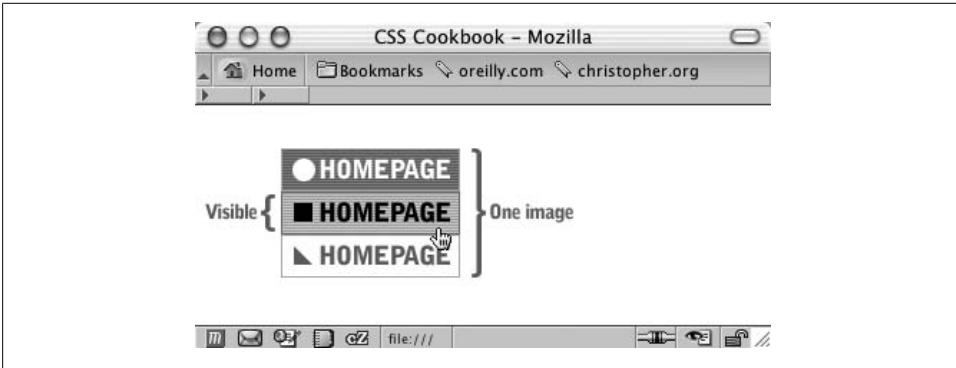

Figure 7-27. Showing a portion of the rollover image

> The drawback of almost all current image replacement techniques is
> that users see nothing if images are turned off, are disabled, or simply
> don't load while the CSS is still supported. It is important to research
> and use the method that's best for your situation. Avoid replacing im-
> ages in important titles. For more information about image replacement
> methods, see Recipe 4.20.

See Also

Recipe 4.20 for replacing HTML text with visually rich imagery or typography; another
demonstration of the LIR technique by Seamus Leahy at *http://www.moronicbajebus*
.com/playground/cssplay/image-replacement/; an explanation on how to create faster
CSS-enabled rollovers without having to preload images at *http://wellstyled.com/css-no*
preload-rollovers.html; a rundown of the FIR technique at *http://www.stopdesign.com/*
also/articles/replace_text/

7.17 Creating Collapsible Menus

Problem

You want to hide a set of links and give the user a way to reveal those links when
needed. For example, rather than two bulleted lists of links, hide one (as shown in
Figure 7-28) and let the user reveal it by clicking on a plus sign (+) icon (as shown in
Figure 7-29).
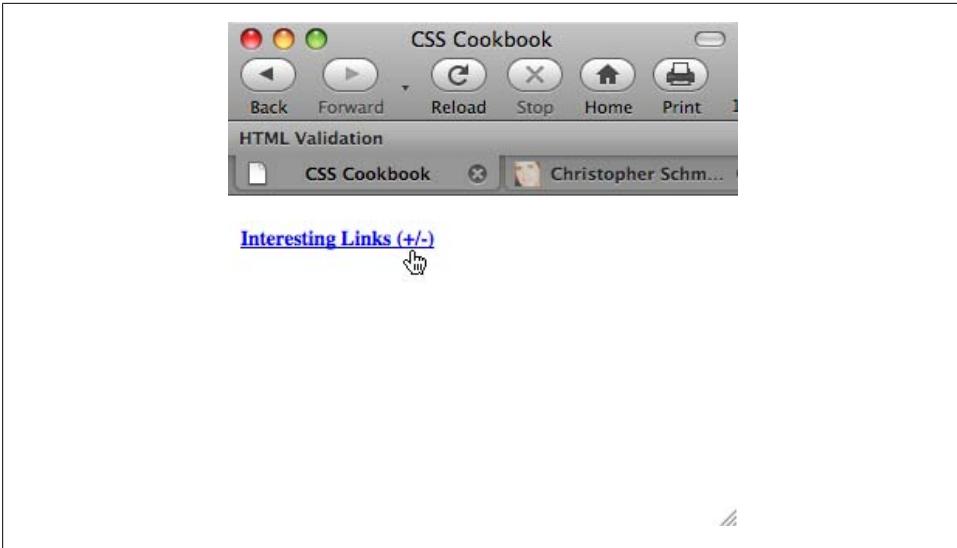
Figure 7-28. Preventing the second set of links from displaying
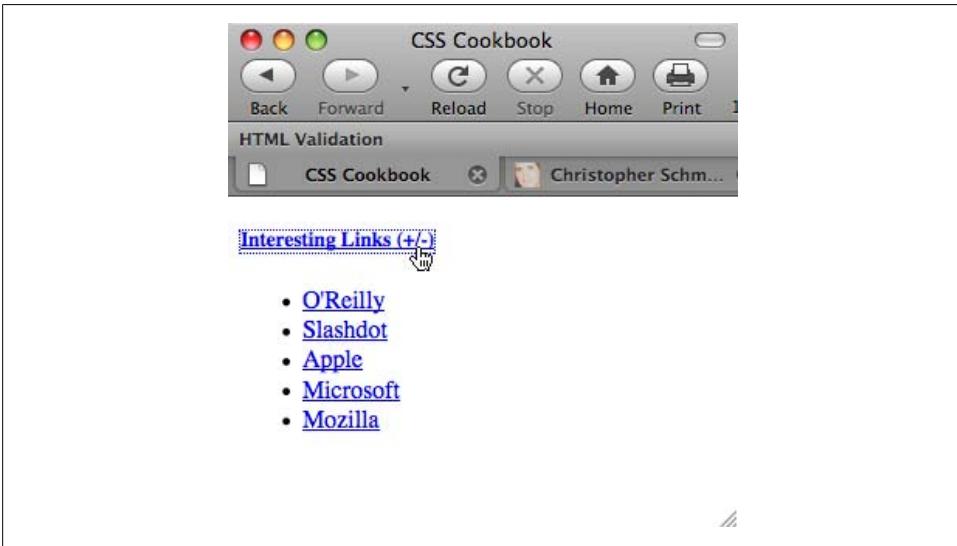
Figure 7-29. The links displayed when the link on the heading is clicked

Solution

First, set up the HTML links to be collapsible with an `id` attribute in the `ul` element:

```
<h5>Interesting Links (+/-)</h5>
<ul id="menulink">
 <li><a href="http://www.oreilly.com/">O'Reilly</a></li>
```

```
<li><a href="http://www.slashdot.org/">Slashdot</a></li>
<li><a href="http://www.apple.com/">Apple</a></li>
<li><a href="http://www.microsoft.com/">Microsoft</a></li>
<li><a href="http://www.mozilla.org/">Mozilla</a></li>
</ul>
```

Then create a CSS rule to prevent the second set of links from displaying when the page is first loaded:

```
#menulink {
  display: none;
}
```

Now add the following JavaScript function that toggles the list of links by swapping the value of `display` from `block` to `none`, or vice versa:

```
function kadabra(zap) {
 if (document.getElementById) {
  var abra = document.getElementById(zap).style;
  if (abra.display == "block") {
   abra.display = "none";
   } else {
   abra.display = "block";
  }
  return false;
  } else {
  return true;
 }
}
```

Insert an anchor element with a JavaScript `onclick` event around the heading. When a user clicks the link, the click triggers the JavaScript function.

```
<h5><a href="#" onclick="return kadabra('menulink');">
Interesting Links (+/-)</a></h5>
```

> JavaScript frames such as jQuery (see Chapter 14) can re-create this Solution without inserting JavaScript-related events into HTML elements (a technique called *unobtrusive JavaScript*).

Discussion

The JavaScript in this function uses `getElementbyId` to toggle the display of the list of menu links. You can scale this technique to show multiple menus or portions of a web document without adding additional lines of JavaScript.

```
<p>Are you sure you want to know the truth? If so,
follow <a href="#" onclick="return kadabra('spoiler'); ">this
link.</a></p>
<p id="spoiler">Darth Vadar was Luke's father!</p>
```

Note that this technique works in Netscape Navigator 6 and later, Opera 7.5 and later, Internet Explorer for Windows 5 and later, and Safari.

See Also

https://developer.mozilla.org/en/DOM/document.getElementById for information on
`getElementById`

7.18 Creating Contextual Menus

Problem

You have a navigation menu, created with Recipe 7.10, and you want to highlight the
current page's location on the menu, as shown in Figure 7-30.
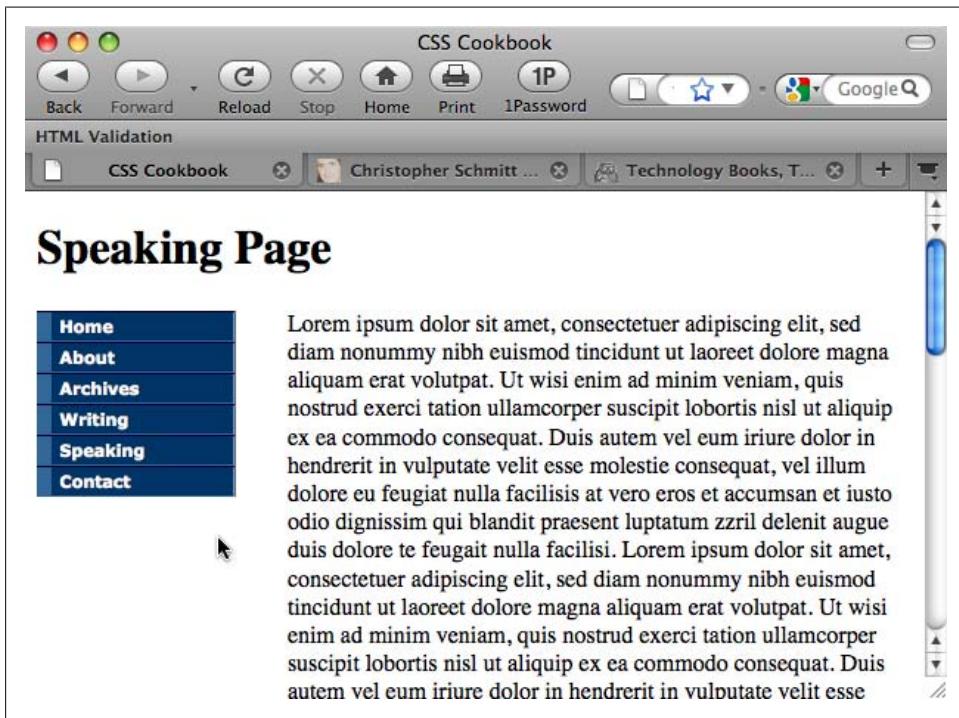
Figure 7-30. The navigation set of links

Solution

Place an `id` attribute in the `body` element of the web document:

```
<body id="pagespk">
```

Also, place `id` attributes in the anchor elements for each link in the menu:

```
<div id="navsite">
  <h5>Site navigation:</h5>
  <ul>
   <li><a href="/" id="linkhom">Home</a></li>
   <li><a href="/about/" id="linkabt">About</a></li>
   <li><a href="/archives/" id="linkarh">Archives</a></li>
   <li><a href="/writing/" id="linkwri">Writing</a></li>
   <li><a href="/speaking/" id="linkspk">Speaking</a></li>
   <li><a href="/contact/" id="linkcnt">Contact</a></li>
  </ul>
</div>
```

With CSS, place two `id` selectors into one descendant selector to finish the menu (see Figure 7-31):

```
#pagespk a#linkspk {
 border-left: 10px solid #f33;
 border-right: 1px solid #f66;
 border-bottom: 1px solid #f33;
 background-color: #fcc;
 color: #333;
 }
```
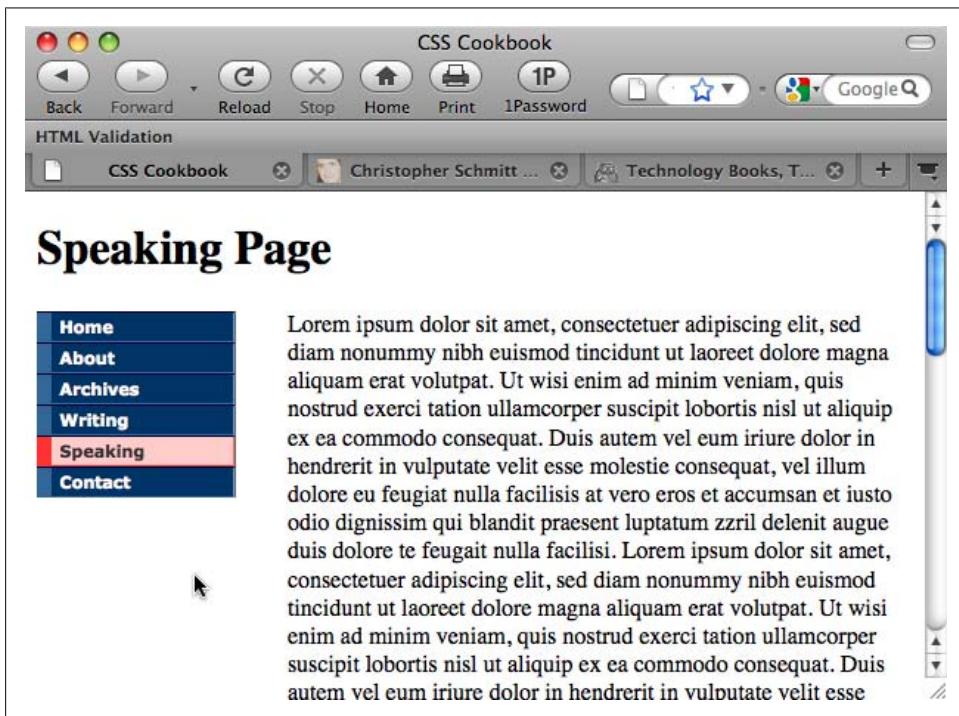

Figure 7-31. The current link, which is different from the rest of the links

Discussion

If you have a small site, you can show a link in a set of navigation links representing the current page by stripping out the anchor link for that page:

```
<div id="navsite">
 <h5>Site navigation:</h5>
 <ul>
  <li><a href="/"Home</a></li>
  <li><a href="/about/">About</a></li>
  <li><a href="/archives/">Archives</a></li>
  <li><a href="/writing/" >Writing</a></li>
  <li>Speaking</li>
  <li><a href="/contact/" >Contact</a></li>
 </ul>
</div>
```

For larger sites that might contain secondary menus, stripping out the link tags on each page increases production and maintenance time. By marking up the links appropriately, you can call the links from a server-side include, and then you can edit the CSS rules that control the style of the navigation links as needed.

To expand the one CSS rule to include all the links in the navigation menu, group the descendant selectors by a comma and at least one space:

```
#pagehom a#linkhom:link,
#pageabt a#linkabt:link,
#pagearh a#linkarh:link,
#pagewri a#linkwri:link,
#pagespk a#linkspk:link,
#pagecnt a#linkcnt:link  {
 border-left: 10px solid #f33;
 border-right: 1px solid #f66;
 border-bottom: 1px solid #f33;
 background-color: #fcc;
 color: #333;
 }
```

In each web document, make sure to put the appropriate `id` attribute in the `body` element. For example, for the home or main page of the site, the `body` element is `<body id="pagehom">`.

See Also

The CSS 2.1 specification on descendant selectors at *http://www.w3.org/TR/CSS21/ selector.html#descendant-selectors*

7.19 Making Tool Tips with the title Attribute

Problem

You want tool tips to appear on a hovered link.

Solution

Use the `title` attribute within the link tag to create a tool tip, as shown in Figure 7-32:

```
<a href="http://www.google.com/" title="Search the Web">...</a>
```
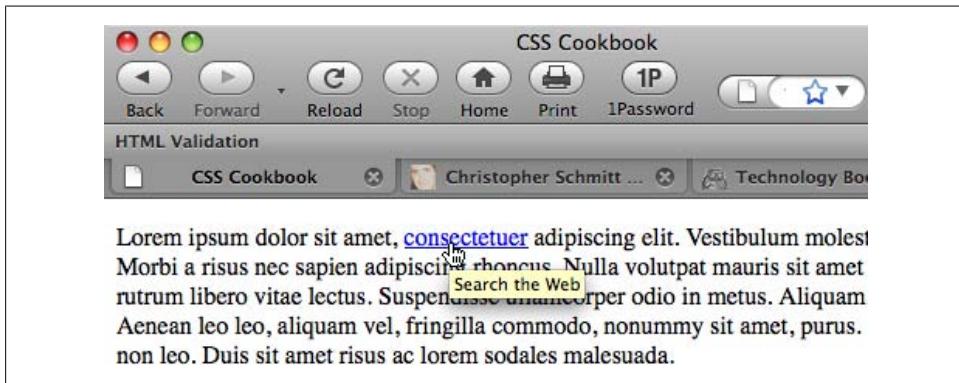

Figure 7-32. The value of the title attribute displayed as a tool tip

Discussion

You can apply a tool tip to almost any element within a web page to enhance accessibility. Try using the tool tip technique on table cells and form input elements.

See Also

The HTML 4.1 specification for the `title` attribute at *http://www.w3.org/TR/html4/struct/global.html#h-7.4.3*

7.20 Designing a Dynamic Tabbed Menu

Problem

You want to build a curved tab navigation menu that works even when text is resized; Figure 7-33 shows the default.

Figure 7-33. Dynamic folder tab navigation

Solution

First, write the markup for the navigation menu:

```
<div id="header">
 <h2>Personal Site dot-com</h2>
 <h5>Site navigation:</h5>
 <ul>
  <li><a href="/">Home</a></li>
  <li><a href="/about/">About</a></li>
  <li><a href="/archives/">Archives</a></li>
  <li><a href="/writing/">Writing</a></li>
  <li id="current"><a href="/speaking/">Speaking</a></li>
  <li><a href="/contact/">Contact</a></li>
 </ul>
</div>
```

Then create two folder tab images: one tab for anchor links and another tab to represent the current page viewed by the user. Split the folder tab image into two images, as shown in Figure 7-34.
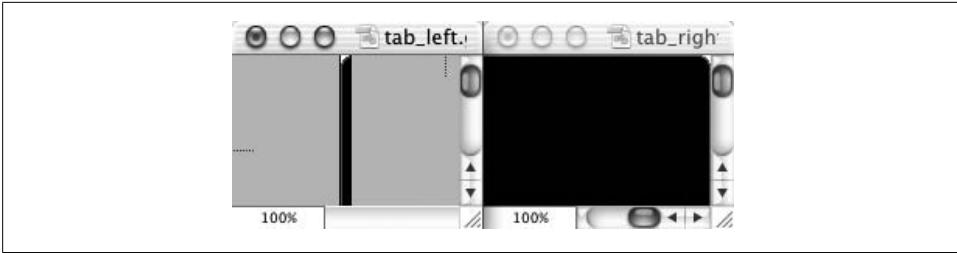
Figure 7-34. The folder tab image split in two; note the curves in the upper corners of the images

Then place the right side of the folder tab in the background of the list item:

```
#header li {
  float:left;
  background-image: url(tab_right.gif);
  background-repeat: no-repeat;
  background-position: right top;
  margin:0;
  padding: 0;
}
```

Place the left side of the folder tab in the background of the anchor element:

```
#header a {
  display: block;
  background-image: url("tab_left.gif");
  background-repeat: no-repeat;
  background-position: left top;
  padding: 5px 15px;
  color: #ccc;
  text-decoration: none;
  font-family: Georgia, Times, "Times New Roman", serif;
}
```

Assign a custom folder tab to represent the current web document being viewed:

```
#header #current {
  background-image:url("tab_right_current.gif");
}
#header #current a {
  background-image:url("tab_left_current.gif");
  color: black;
}
```

Place the image with a line measuring 1 pixel high at the bottom of the grouping.

Discussion

Keeping the text in the navigation links aids in three areas of web development: accessibility, design, and maintenance. For example, users with poor eyesight can adjust the size of the text and tabs without breaking the design.

Because users can resize the text to very large settings, the background images that comprise the folder tabs need to be large as well; otherwise, the folder tabs will break. In this Solution, the folder tab images have a height of 450 pixels.

Web developers prefer this method because it lets them easily maintain the list of links. To change a navigation label or correct a typo, developers can simply edit the HTML text without having to return to a digital imaging program to create folder tab images.

Another benefit of this method is that the folder tabs can be designed in a more aesthetically pleasing way. Recipe 7.12 demonstrates how to create a navigation setup with folder tabs using the border property. This look creates a boxy or squared edge to the folder tabs. With this current recipe, however, web developers can curve the tabs and introduce color blending for improved aesthetics.

See Also

Recipe 3.22, which uses a similar rubber band technique to create pull quotes with images; the article "Sliding Doors of CSS, Part II" at http://www.alistapart.com/articles/slidingdoors2/, which expands on this folder tab navigation concept

7.21 Changing Styles on Anchored Links

Problem

You want to change the style of elements within a web page when a user clicks on a link.

Solution

First, set up the markup with normal anchored links within the document. For this Solution, the anchored links (technically referred to as *fragment identifiers*) are placed within an image map:

```
<img src="target_header.jpg" alt="Header" border="0" usemap="#Map" />
<map name="Map" id="Map">
 <area shape="circle" coords="115,136,72" href="#mark" />
 <area shape="circle" coords="244,145,55" href="#jessica" />
 <area shape="circle" coords="340,88,58" href="#trueman" />
 <area shape="circle" coords="480,287,79" href="#katrina" />
</map>
<div class="bios">
 <dl id="katrina">
  <dt>Katrina</dt>
  <dd>...</dd>
 </dl>
 <dl id="jessica">
  <dt>Jessica</dt>
  <dd>...</dd>
 </dl>
 <dl id="trueman">
  <dt>Trueman</dt>
```

```
  <dd>...</dd>
</dl>
 <dl id="mark">
  <dt>Mark</dt>
  <dd>...</dd>
</dl>
</div><!-- end /#bios -->
```

Then set up CSS rules for the default styles for the web page (as shown in Figure 7-35):

```
.bios dt {
 font-weight: bold;
}
.bios dd {
 margin: 0;
 padding: 0;
}
```
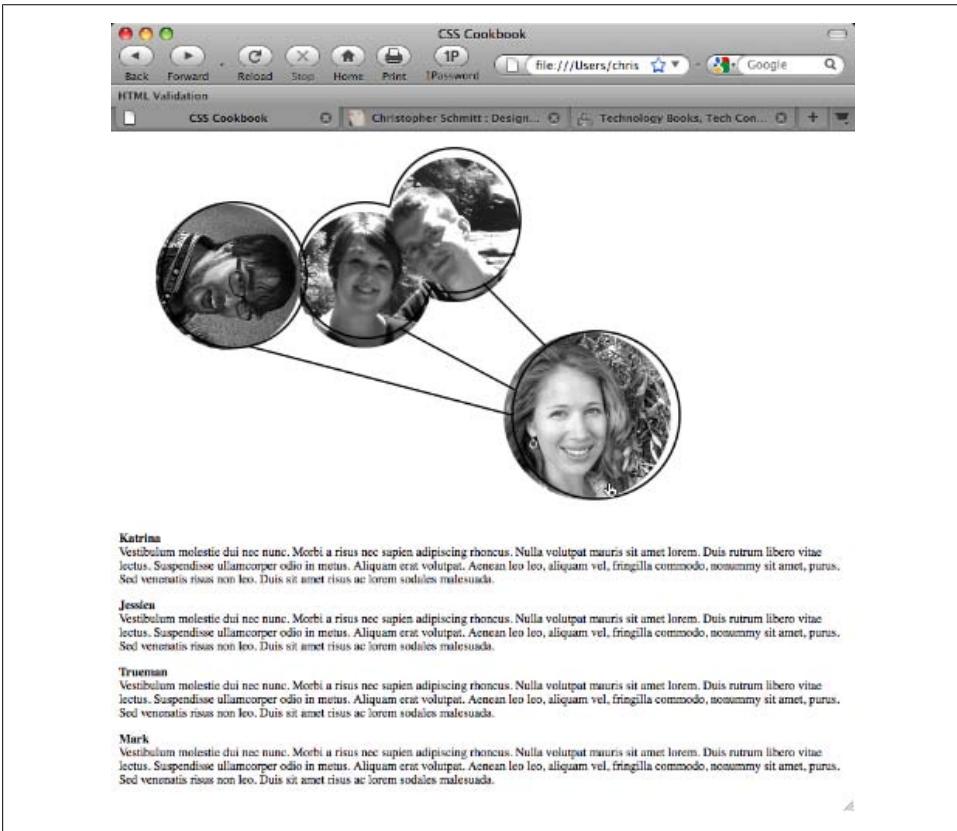

Figure 7-35. The default rendering of the web page

Then use the `:target` pseudo-class to define the look of the elements when the user clicks on the anchored link, as shown in Figure 7-36:

```css
.bios dl:target {
 background-color: #999999;
 border: 1px solid black;
 padding: 1em;
 font-weight: bold;
 line-height: 1.5;
}
.bios dl:target dt {
 font-style: italic;
 color: white;
 font-size: 1.5em;
 background-color: #cccccc;
 margin-right: 20px;
}
.bios dl:target dd {
 margin-right: 20px;
 background-color: #cccccc;
 padding: 0 1em 1em 1em;
}
```
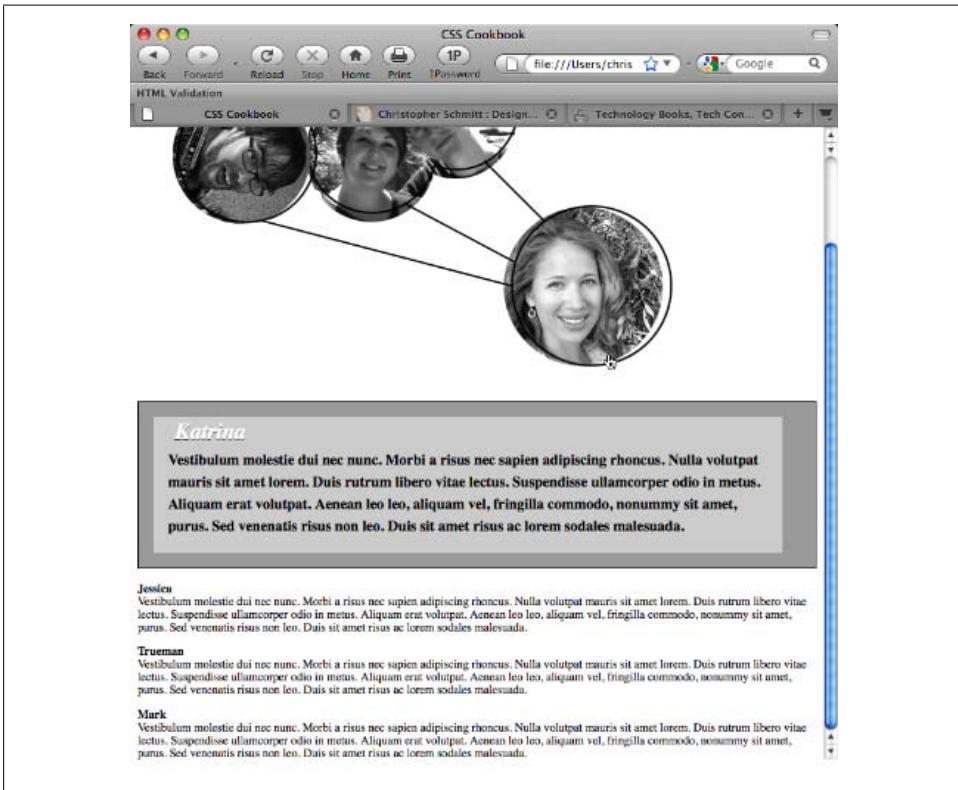

Figure 7-36. The Katrina portion of the page with its changed style

To return the targeted element(s) back to the default style when the user clicks on another anchored link, use the `:not` pseudo-class, as shown in Figure 7-37:

```
.bios dl:not(:target) {
  border: none;
  padding: 0;
  font-size: .8em;
}
```
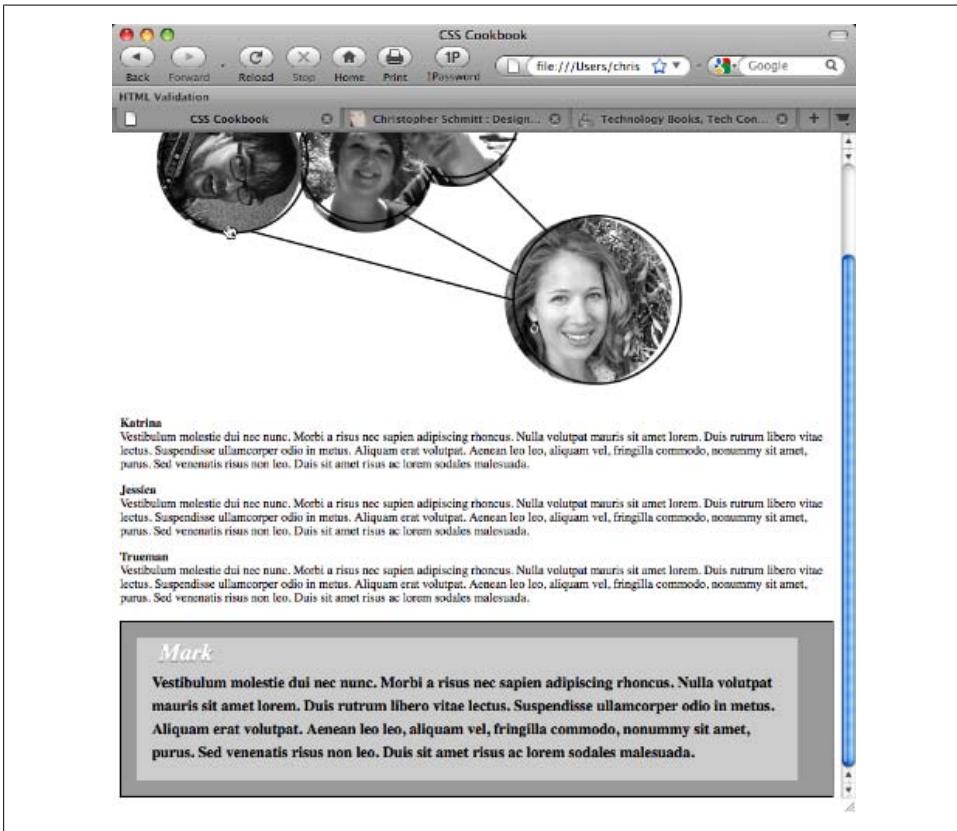

Figure 7-37. The Katrina portion, reverted to its default value when another link was activated

Discussion

The `:target` and `:not` pseudo-classes are a part of the CSS3 specification and thus aren't well known to most web designers. However, the selectors can perform a great deal of heavy lifting.

Pure CSS collapsible menus

By working with these selectors, you could replace the JavaScript-based solution with a few extra CSS rules. First, update the markup to add the anchor link:

```
<h5>
 <a href="#menulink">Interesting Links</a>
</h5>
<ul id="menulink">
 <li><a href="http://www.ora.com/">O'Reilly</a></li>
 <li><a href="http://www.slashdot.org/">Slashdot</a></li>
 <li><a href="http://www.apple.com/">Apple</a></li>
 <li><a href="http://www.microsoft.com/">Microsoft</a></li>
 <li><a href="http://www.mozilla.org/">Mozilla</a></li>
</ul>
```

Then set up the following CSS rules:

```
/* default rendering */
ul#menulink {
 display: none;
}

/* when 'targeted' */
ul:target {
 display: block;
}

/* revert back to default rendering */
ul:not(:target) {
 display: none;
}
```

Known browser issues

Currently, the `:target` and `:not` pseudo-classes are supported in Firefox, Safari, Chrome, Opera, and Internet Explorer 7 for Windows.

See Also

The CSS3 specification for `:target` at *http://www.w3.org/TR/css3-selectors/#target-pseudo*

Forms

8.0 Introduction

Without HTML forms, we wouldn't be able to log in to web-based email accounts, order books with one click, or trade stocks online. Although forms make the Web go around, they are ugly due to the generic way in which browsers display them.

The default rendering of online forms usually includes beveled `input` and `textarea` fields, as well as boring-looking buttons. Such a look and feel might be acceptable if you are making a form for use on a small intranet or on a small website, but it is unacceptable if you want to project a professional image.

Even Google, lauded for its minimalism, has resorted to changing its highly praised search form to use WebKit's proprietary CSS properties to create more realistic form controls.

Fortunately, with a few CSS rules you can create forms that stand out from the pack. This chapter helps you get straight into the techniques for creating higher-quality forms.

You will learn the settings for HTML user `input` elements such as buttons, text areas, and fields. You will also learn how to set up a submit-once-only button to keep site visitors from mistakenly sending several processes to the server. At the end of the chapter are two sample designs: a simple login form without tables and a long registration form with tables.

> Appendix D serves as an excellent resource that complements this chapter. In addition, see *http://webformelements.com/* for a visual compendium detailing the effect of a majority of the visual CSS properties on `form` elements in 10 of today's modern browsers.

8.1 Modifying the Spacing Around a Form

Problem

You want to modify the space around a form.

Solution

Set the margin to zero while adjusting the padding values of the form element, as shown in Figure 8-1:

```
form {
 margin: 0;
 padding: 1em 0;
 border: 1px dotted red; /* set in order to see padding effect */
}
```
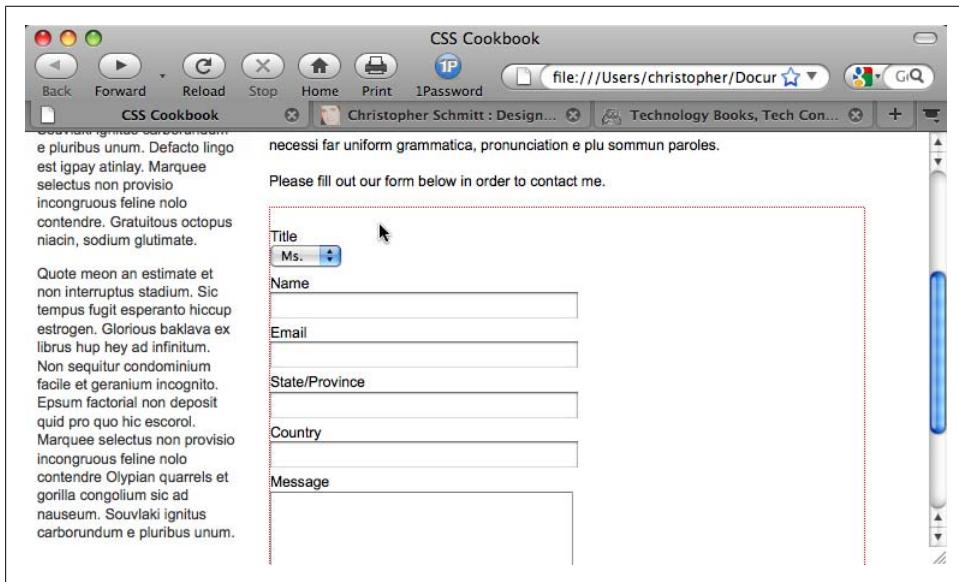

Figure 8-1. Padding applied under the form's border

Discussion

When positioning forms into a web page design, developers find that they will need to modify the space between the form and other page elements in the design. Typically, the most common modification is to adjust the padding at the top and bottom of the form.

See Also

Recipe 6.2 for styling `input` elements

8.2 Removing the Space Around a Form

Problem

You want to remove the space around a small form.

Solution

Set the `form` element to display as an inline element instead of a block-level element:

```
form {
 display: inline;
 padding: 0
}
```

Discussion

By default, the `form` element is a block-level element, which means that it forces a line break above and below itself. To remove that spacing, change the default property to `inline`.

See Also

Recipe 1.5 for a discussion of block-level and inline-level elements

8.3 Setting Styles for Input Elements

Problem

You want to change the appearance of the background color of `input` elements. Such effects can take you from Figure 8-2 to 8-3.

Solution

Use a `class` selector to design the `input` elements of the form:

```
<h2>Simple Quiz</h2>
 <form action="simplequiz.php" method="post">
 <p>
  Are you
   <input type="radio" value="male" name="sex"
class="radioinput" />
  Male or
   <input type="radio" value="female" name="sex"
class="radioinput" />
```

```
   Female?
  </p>
<p>
 What pizza toppings do you like? <input type="checkbox" name=""
value="l" class="checkbxinput"> Pepperoni <input type="checkbox"
name="" value="mushrooms" class="checkbxinput"> Mushrooms <input
type="checkbox" name="" value="pineapple" class="checkbxinput">
 Pineapple
 </p>
 <label for="question1">Who is buried in Grant's tomb?</label>
 <input type="text" name="question1" id="question1"
class="textinput"
value="Type answer here" /><br />
 <label for="question2">In what country is the Great Wall of
China Located?</label>
 <input type="text" name="question2" id="question2"
class="textinput"
value="Type answer here" /><br />
 <label for="password">What is your password?</label>
 <input type="password" name="password" id="password"
class="pwordinput"
value="" /><br />
 <input name="reset" type="reset" id="reset" value="Reset" />
 <input type="submit" name="Submit" value="Submit"
class="buttonSubmit" />
</form>
```
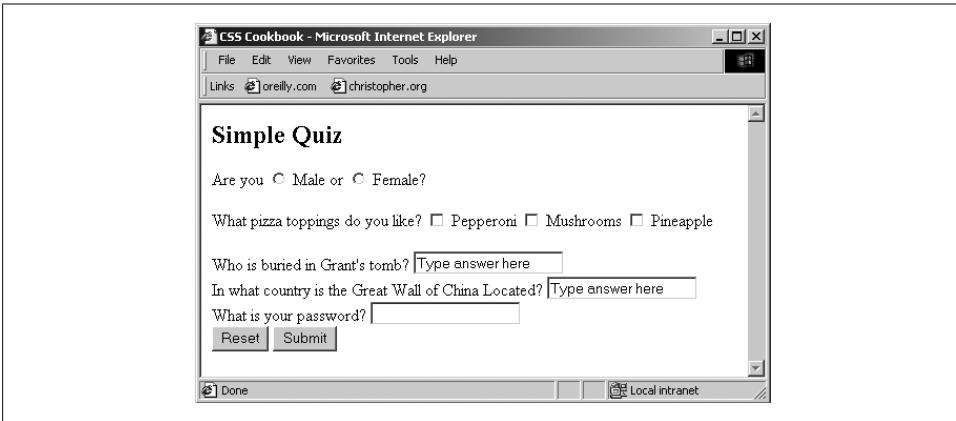

Figure 8-2. The form without styles

Then apply CSS rules to change the presentation of the input elements:

```
.textinput {
margin-bottom: 1.5em;
width: 50%;
color: #666;
background-color: #ccc;
}
.pwordinput {
```

```
  color: white;
  background-color: white;
}
.radioinput {
 color: green;
 background-color: #ccc;
}
.checkbxinput {
 color: green;
 background-color: green;
}
```
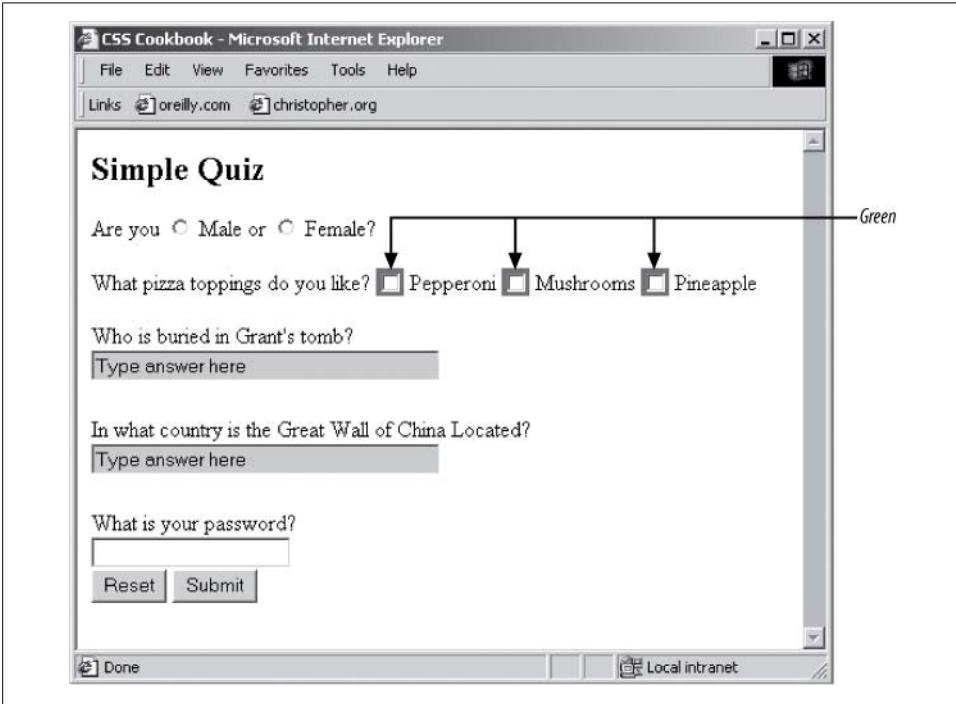

Figure 8-3. Styles applied to the input fields

Discussion

As of this writing, Opera, IE, Safari, Chrome, and Firefox do not support the `color` property for radio buttons or checkboxes (see *http://tr.im/colorradiobutton* and *http://tr.im/colorcheckboxes*).

IE and Opera support background color for radio buttons and checkboxes. However, IE wraps the background color around the window whereas Opera sets the background color within the form controls (see *http://tr.im/bkgdcolorradiobuttons* and *http://tr.im/bkgdcolorcheckboxes*).

Using attribute selectors

Rather than using class selectors as illustrated in the Solution, another way to stylize different kinds of input fields is through attribute selectors.

With attribute selectors, you remove `class` attributes from the HTML and use only the following CSS rules:

```
input[type="text"] {
 margin-bottom: 1.5em;
 width: 50%;
 color: #666;
 background-color: #ccc;
}
input[type="password"] {
 color: white;
 background-color: white;
}
```

The main drawback to this approach is that IE6 and earlier versions do not support attribute selectors. However, IE7 and later versions do support attribute selectors.

Attribute selectors currently work in Firefox, Chrome, Safari, and Opera. If you want to ensure cross-browser support, you need to use class selectors to determine styles for different form controls.

> To review whether a browser styles an input `form` element, and to see how it does it, visit *http://www.flickr.com/photos/teleject/sets/72157615099024461/*.

See Also

The CSS 2.1 specification for dynamic pseudo-classes at *http://www.w3.org/TR/CSS21/selector.html#x33*; the CSS 2.1 specification for attribute selectors at *http://www.w3.org/TR/CSS21/selector.html#attribute-selectors*

8.4 Changing Styles on Form Elements When a User Clicks on Them

Problem

You want to change the background color of a `form` element when a user clicks on it.

Solution

Use the `:focus` pseudo-class to trigger a change in which CSS rules get applied to the form's element, as shown in Figure 8-4:

```
.textinput:focus {
 background-color: yellow;
}
```
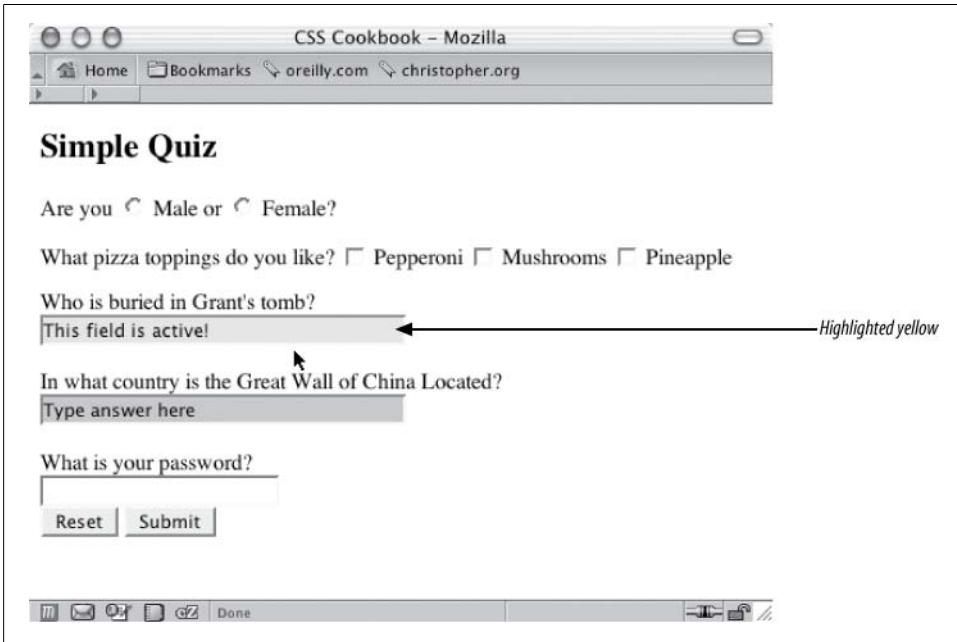

Figure 8-4. Using :focus to light up an input field

Discussion

Using the :focus pseudo-class is an easy way to add a bit of usability or user feedback to a form. Changing styles in this way lets the user know where she is in a form and what she is working on, without confusing her.

See Also

Recipe 2.6 for a discussion about other pseudo-classes

8.5 Applying Different Styles to Different Input Elements in the Same Form

Problem

You want to style multiple input elements in the same form differently.

Solution

Use two or more different class selectors to apply different styles.

First, apply `class` attributes with different values to the `input` elements:

```
<label for="fmname">Name</label>
<input type="text" name="fmname" class="fmname" />
<label for="fmemail">Email</label>
<input type="text" name="fmemail" class="fmemail" />
```

Then set up the styles for each `class` attribute in the `input` elements:

```
.fmname {
 text-align: left;
}
.fmemail {
 text-align: center;
}
```

Discussion

Most browsers support the use of class selectors to apply multiple styles to common elements on a page.

Using attribute selectors

Another method of assigning different styles to common elements is available through browsers that understand the CSS3 specification for attribute selectors, as discussed in Recipe 8.3. For this Solution, the code for using attribute selectors (after removing the `class` attributes and their respective values from the HTML) follows:

```
input[name="fmname"] {
 text-align: left;
}
input[name="fmemail"] {
 text-align: center;
}
```

See Also

Recipe 8.5 for styling different form buttons in the same form

8.6 Setting Styles for textarea Elements

Problem

You want to set styles for `textarea` elements in a web form to change the text's color, size, weight, and other properties, as shown in Figure 8-5.
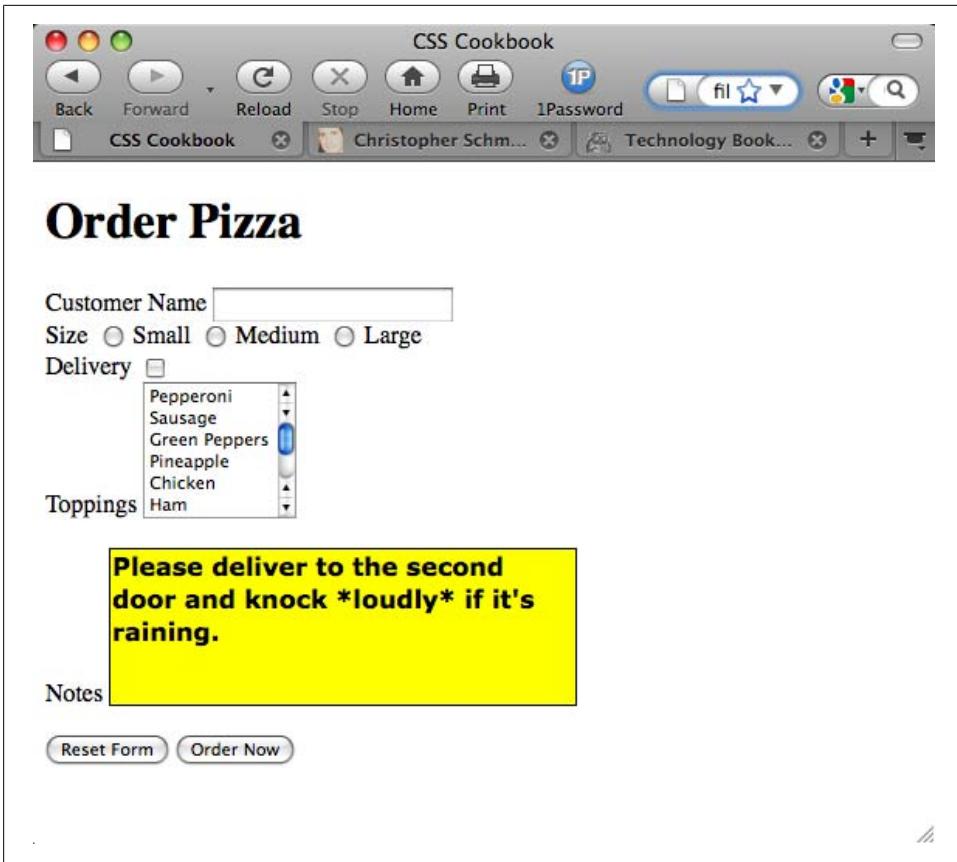
Figure 8-5. A textarea element with styles applied

Solution

Use a type selector to associate styles with `textarea` elements:

```
textarea {
  width: 300px;
  height: 100px;
  background-color: yellow;
  font-size: 1em;
  font-weight: bold;
  font-family: Verdana, Arial, Helvetica, sans-serif;
  border: 1px solid black;
}
```

Discussion

Associating styles to `textarea` elements is fairly straightforward through the use of a type selector:

```
textarea {
 background-color: blue;
}
```

By adding the :focus pseudo-class, you can change the style of the active textarea field:

```
textarea:focus {
 background-color: green;
}
```

So, as a user fills out a form, the textarea field that he is currently filling out will change color.

> To review whether a browser styles a textarea form element, and to see how it does it, visit *http://www.flickr.com/photos/teleject/sets/72157615099816279/*.

See Also

The CSS 2.1 specification for dynamic pseudo-classes at *http://www.w3.org/TR/CSS21/selector.html#x33*; the CSS 2.1 specification for attribute selectors at *http://www.w3.org/TR/CSS21/selector.html#attribute-selectors*

8.7 Setting Styles for select and option Elements

Problem

You want to alter the look of list menus in a form by changing the color and font, as shown in Figure 8-6.

Solution

Use a type selector to associate styles with select elements:

```
select {
 color: white;
 background-color: blue;
 font-size: 0.9em;
}
option {
 padding: 4px;
}
```

Discussion

Unlike input form elements, there is only one type of select element, so associating styles to that element is straightforward and can be done through a type selector. Styling the option element is just as easy.
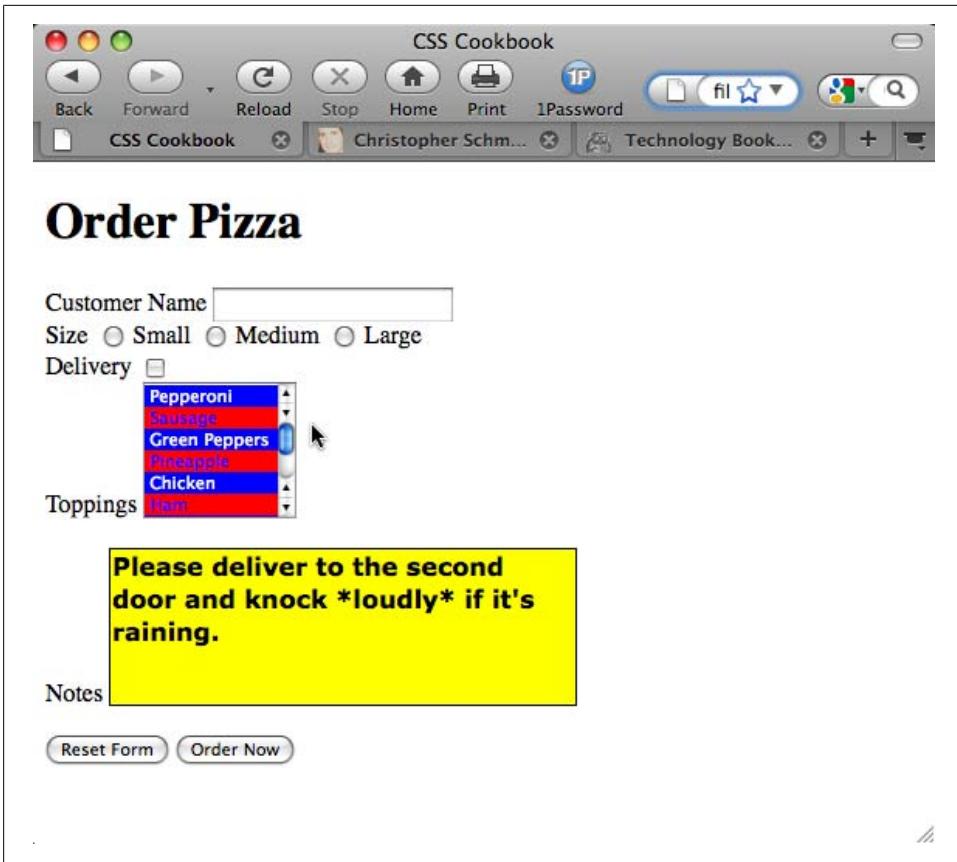
Figure 8-6. The select and option elements with styles applied

To stylize alternating options in a select list, first include the class attribute in the option element:

```
<select name="Topping_ID" size="6" multiple>
 <option value="1">Pepperoni</option>
 <option value="2" class="even">Sausage</option>
 <option value="3">Green Peppers</option>
 <option value="4" class="even">Pineapple</option>
 <option value="5">Chicken</option>
 <option value="6" class="even">Ham</option>
 <option value="7">Olives</option>
 <option value="8" class="even">Onions</option>
 <option value="9">Red Peppers</option>
</select>
```

If you are using XHTML, you need to set the `multiple` attribute as its own value as well:

```
<select name="Topping_ID" size="6" multiple="multiple">
```

Then set up the CSS rules for the two sets of `option` elements, making sure the `option` elements with an even value (as noted by the `even` class selector) look different from the others. For example, `option` elements with an `even` selector have a background color of red, whereas the "regular" `option` elements have a background color of blue (as shown earlier in Figure 8-6):

```
select {
 font-size: 0.9em;
}
option {
 color: white;
 background-color: blue;
}
option.even {
color: blue;
 background-color: red;
}
```

Instead of using a class selector, you can use the `:nth-child(odd)` selector to style alternating `option` elements. However, at the time of this writing some of the popular browsers do not support this selector.

To review whether a browser styles a `select` element, and to see how it does it, visit *http://www.flickr.com/photos/teleject/sets/72157615222275442/* and *http://www.flickr.com/photos/teleject/sets/72157615152109523/*.

See Also

Recipe 8.3 for information on how to change the color and size of `input` element text

8.8 Creating a Macintosh-Styled Search Field

Problem

You want to style a search field for the Safari browser.

Solution

Use proprietary HTML extensions that are available only to the Safari browser.

Place one `input` element in between the `div` element. Then set the value for the `type` attribute to `search`, as shown in Figure 8-7:

```
<form method="get" action="/search.php">
<div>
<label for="q">Search</label>
<input type="search" placeholder="keywords"
autosave="com.domain.search" results="7" name="q" />
</div>
</form>
```
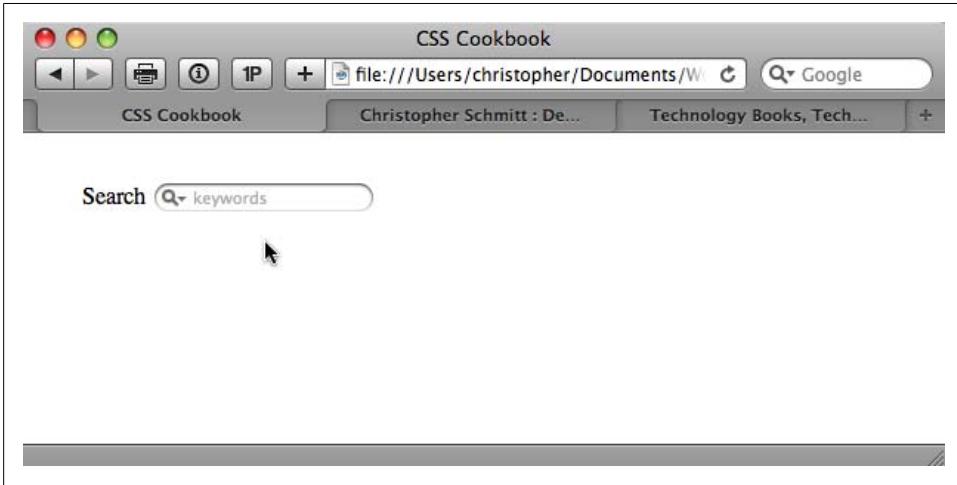

Figure 8-7. The Safari search field

Discussion

The Safari browser developers from Apple created an extension to HTML forms to allow for a more robust user interface. A browser other than Safari will render the input field like a regular text input form field that is still usable as a typical search bar.

Search field attributes

The `placeholder` attribute allows web developers to set the text residing in the search field. This text appears in the same way as text set for the `value` attribute in a text input field, as shown in Figure 8-8:

```
<label for="fmwebsite">Web Site:</label>
<input type="text" name="fmwebsite" value="http://" />
```

The difference between the `placeholder` attribute and the standard `value` attribute is that users have to manually delete the text placed in the form field.
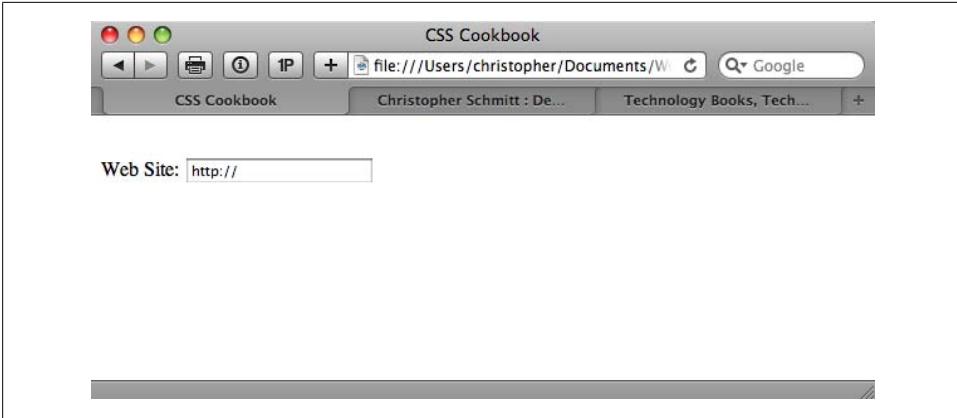
Figure 8-8. The input field in the Safari browser
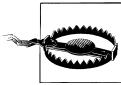
It is not recommended that you place both a `value` and a `placeholder` attribute in the same search field. This technique results in the text for the value overriding the placeholder value for Safari users. Safari users will have to manually delete the text supplied through the `value` attribute, and thus will not get the intended functionality allowed in the `placeholder` attribute.

The `autosave` attribute is a marker that allows past searches to be stored on the user's local machine. The user will be able to click on the magnifying glass icon and see past searches.

The `results` attribute accepts a numerical value. This numerical value represents the number of searches that will be stored on the user's local computer, as shown in Figure 8-9.
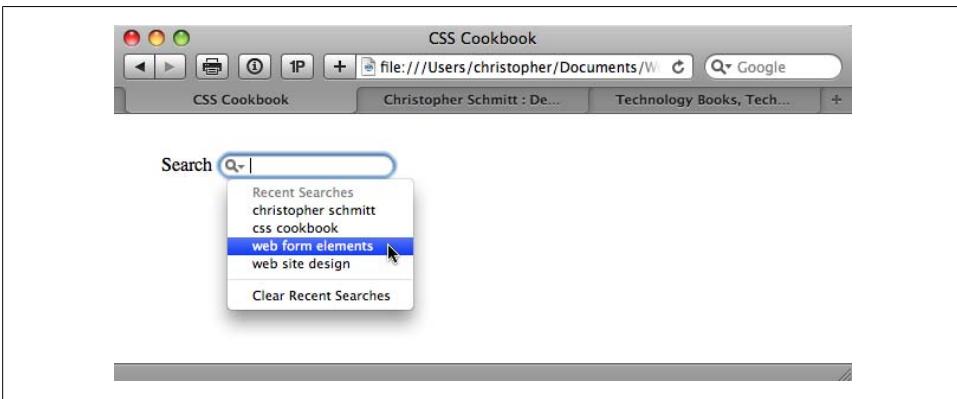
Figure 8-9. Saved searches appearing below the search field

The saved searches can appear to be placed from one site to another. For example, if one site uses the same value for `autosave` as another site, the same saved searches will appear on both sites' search fields. This technique could be used on a set of different domains that are in a common network. The user has access to his search history, thus resulting in a better user experience.

Best practices

Since the search field does not require an input button, you should use the search field in a form that has only one input field. Forms with only one input `form` element will accept the pressing of an Enter or Return key as form submission. The addition of `input` elements means the browser may need a Submit button that must be activated to process the form.

See Also

The Surfin' Safari blog about the search field extension at *http://weblogs.mozillazine.org/hyatt/archives/2004_07.html#005890*; Recipe 6.7 for a technique on how to add icons by padding elements with the `background-image` property

8.9 Styling Form Buttons

Problem

You want to stylize the color, padding, borders, and rollover effects for Submit and Reset buttons on a form. The top of Figure 8-10 shows a form without styles applied to the buttons, and the bottom of Figure 8-10 shows the form with stylized buttons.

Solution

First use a class selector to design the buttons:

```
<form action="simplequiz.php" method="post">
 <label for="question">Who is president of the U.S.?
</label>
 <input type="text" name="question" id="textfield"
 value="Type answer here" /><br />
 <input name="reset" type="reset" value="Reset"
 class="buttonReset" />
 <input type="submit" name="Submit" value="Submit"
class="buttonSubmit" />
</form>
```
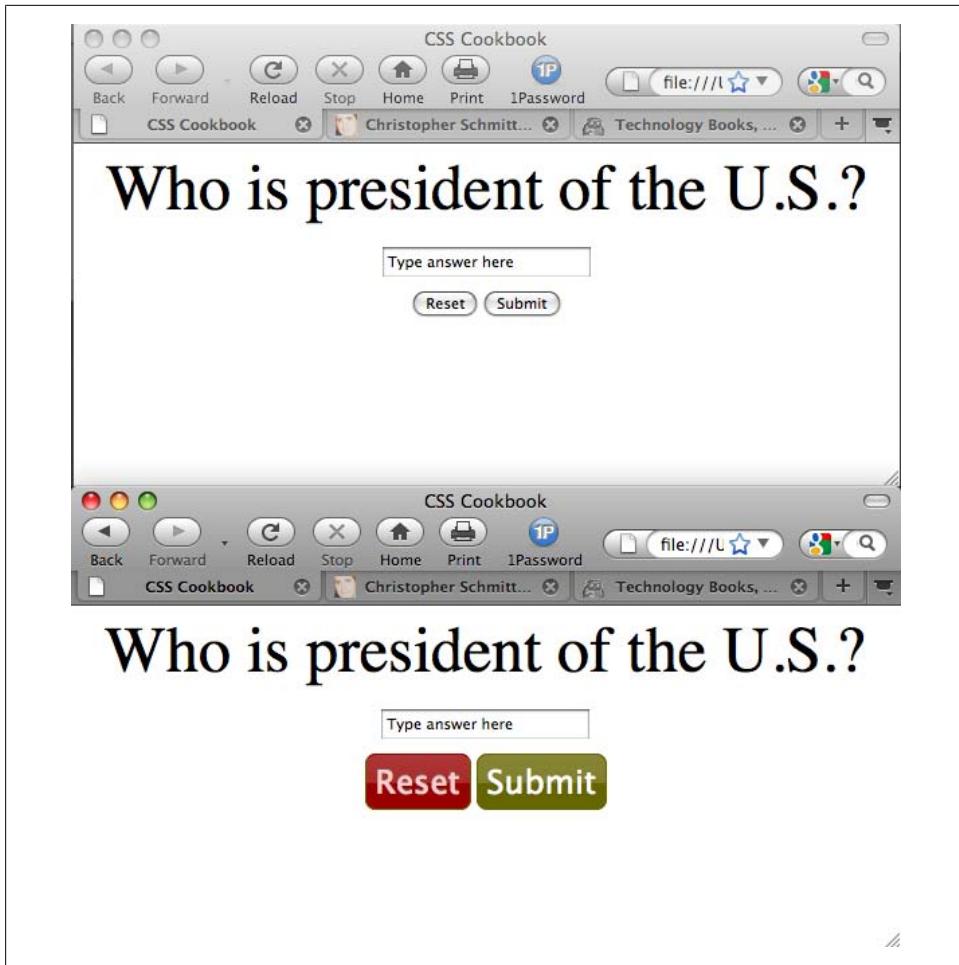
Figure 8-10. The form buttons without styles applied (top) and with styles applied (bottom)

Then use CSS to stylize the buttons:

```
.buttonReset {
color: #fcc;
background-color: #900;
font-size: 1.5em;
border: 1px solid #660;
padding: 4px;
background-image: url(title-glass.png);
background-repeat: repeat-x;
background-position: 50%;
text-shadow: 0 -1px 0 #666;
-moz-border-radius: 8px;
-webkit-border-radius: 8px;
border-top: 1px solid #900;
```

```
 }
 .buttonSubmit {
  color: white;
  background-color: #660;
  font-size: 1.5em;
  border: 1px solid #660;
  padding: 4px;
  background-image: url(title-glass.png);
  background-repeat: repeat-x;
  background-position: 50%;
  -moz-border-radius: 8px;
  -webkit-border-radius: 8px;
  border-top: 1px solid #660;
 }
```

Discussion

You also can stylize buttons using the rollover state. To create rollovers for buttons, use a JavaScript function:

```
<script language="JavaScript" type="text/javascript">
function classChange(styleChange,item) {
 item.className = styleChange;
}
</script>
```

Next, add two additional CSS rules, one for the rollover state for the Reset button and another for the Submit button:

```
.buttonResetRoll {
 color: white;
 background-color: #c00;
 font-size: 1.5em;
 border: 1px solid #660;
 padding: 4px;
}
.buttonSubmitRoll {
 color: white;
 background-color: #cc0;
 font-size: 1.5em;
 border: 1px solid #660;
 padding: 4px;
}
```

After the function is in place and the extra CSS rules are set up, place the events in the button markup so that you can toggle between the off and on states of the form buttons (see Figure 8-11):

```
<form action="simplequiz.php" method="post">
<label for="question">Who is president of the U.S.?</label>
<input type="text" name="question" id="textfield"
value="Type answer here" /><br />
<input name="reset" type="reset" id="reset" value="Reset"
class="buttonReset"
onMouseOver="classChange('buttonResetRoll',this)"
```

```
onMouseOut="classChange('buttonReset',this)" />
 <input type="submit" name="Submit" value="Submit"
class="buttonSubmit"
onMouseOver="classChange('buttonSubmitRoll',this)"
onMouseOut="classChange('buttonSubmit',this)" />
</form>
```


Figure 8-11. A rollover state created through CSS and JavaScript

You will need to use class selectors to set button styles for IE6 users.

Using attribute selectors to write CSS rules for the form buttons doesn't require the extra markup in the HTML element that comes from using class selectors. For example, the attribute selector syntax for the buttons using only CSS would look something like this:

```
input[type="reset"] {
 color: #fcc;
 background-color: #900;
 font-size: 1.5em;
 border: 1px solid #660;
 padding: 4px;
}
input[type="submit"] {
 color: white;
 background-color: #660;
 font-size: 1.5em;
 border: 1px solid #660;
```

```
  padding: 4px;
 }
```

You also can use the `width` property to determine the horizontal size of the button.

> To review whether a browser styles a `select` element, and to see how it does it, visit *http://www.flickr.com/photos/teleject/sets/72157615221157426/*.

See Also

Recipe 6.9 for more tips on mimicking an image button with CSS; the CSS 2.1 specification for attribute selectors at *http://www.w3.org/tr/css21/selector.html#attribute-selectors*

8.10 Creating an Image Submit Button

Problem

You want to create a custom Submit button with an image file, as shown in Figure 8-12.
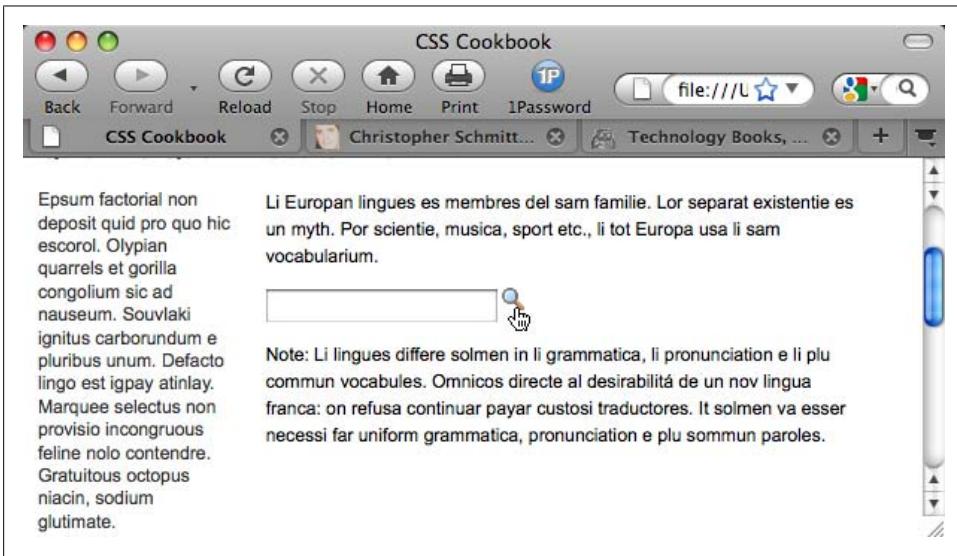
Figure 8-12. A rollover state created through CSS and JavaScript

Solution

Use the `input` element with the `type` attribute set to `image`:

```
<input type="image" name="submit" src="submit.gif" />
```

Discussion

Although inserting an image as a Submit button utilizes HTML, once you have placed the Submit button image in a web page you can modify it through CSS properties such as `border` and `margin`.

See Also

The HTML specification for `input` at *http://www.w3.org/TR/html4/interact/forms.html #h-17.4*

8.11 Setting Up a Submit-Once-Only Button

Problem

You want to keep people from clicking the Submit button more than once.

Solution

First create a class for keeping the button from being displayed:

```
.buttonSubmitHide {
 display: none;
}
```

Then use the following JavaScript programmed to switch styles by class selectors:

```
<script language="JavaScript" type="text/javascript">
function classChange(styleChange,item) {
 item.className = styleChange;
}
</script>
```

Now trigger the function by using an `onsubmit` event to remove the Submit button from the web document:

```
<h2>Order Confirmation</h2>
<form action="login.php" method="post"
 onsubmit="classChange('buttonSubmitHide',submit);
return true;">
 <div align="center">
  <p>Are you sure you want to purchase 12 cans of soda over the
Web?</p>
  <label for="uname">Final Price:</label>
  <input type="text" name="uname" id="uname" value="$7.95" />
<br />
   (includes tax, s+h extra)<br />
```

```
    <input type="submit" name="submit" value="submit"
class="buttonSubmit" />
 </div>
</form>
```

Discussion

The JavaScript function in the Solution triggers a change in which a style is applied to the element. You must use the form's `onsubmit` event to execute the function so that the form's action will still be executed. If the function were triggered with an `onclick` event on the Submit button, some browsers would execute only the class-changing function. Then, because the button is no longer visible, the user would not be able to trigger the form.

See Also

JavaScript & DHTML Cookbook by Danny Goodman (O'Reilly) for more recipes that combine JavaScript and CSS

8.12 Creating a Submit Button That Looks Like HTML Text

Problem

You want to make a Submit button look like plain HTML text.

Solution

Use several CSS formatting properties to make a form's Submit button look like HTML text.

First, insert a `class` attribute and value:

```
<input type="submit" name="submit" value="send &raquo;" class="submit" />
```

Then apply CSS properties to strip away the Submit button borders and background color, as shown at the top of Figure 8-13:

```
.submit {
 border: none;
 background-color: #fff;
 padding: 0;
 margin: 0;
 width: 5em;
}
```

Now add the `:hover` pseudo-class to create the standard rollover effect, as shown at the bottom of Figure 8-13:

```
.submit:hover {
 text-decoration: underline;
}
```
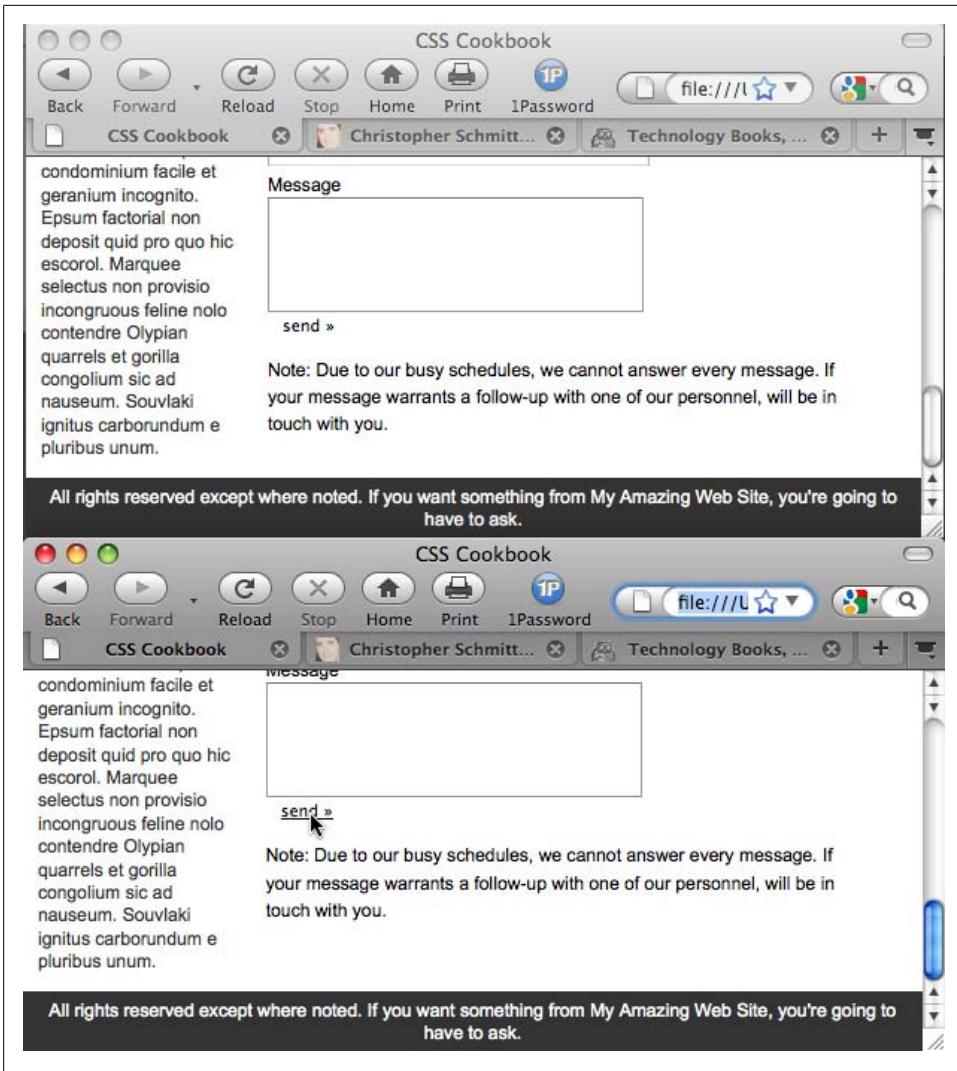
Figure 8-13. A Submit button that looks like HTML text

Discussion

A Submit button that looks like HTML text is perfect for designers who feel that a generic-looking Submit button may not fit in their designs, yet they do not want to use an image for a button.

In addition, bringing in the design element of a Submit button can sometimes be counterproductive in terms of user experience. By stripping down a Submit button so that it appears as text, you may be able to put users' fears about submitting information across the Internet to rest.

Browser support

This recipe works in browsers that allow modifications to Submit buttons. Modern browsers that support this recipe include Chrome, Safari, Firefox, IE8, and Opera.

See Also

Recipe 8.13 for making actual HTML text operate like a Submit button

8.13 Making an HTML Text Link Operate Like a Submit Button

Problem

You want to make an HTML text link execute a form.

Solution

Use JavaScript to trigger the form:

```
<form name="msgform" method="get" action="results.php">
<label for="fmmsg">Message</label>
<textarea name="fmmsg" accesskey="m" id="fmmsg" rows="5" cols="14"></textarea>
<a href="javascript:document.msgform.submit();">Submit</a>
</form>
```

Discussion

Whereas Recipe 8.10 showed how to disguise a Submit button to look like HTML text, this recipe showcases how to make a text link work as a Submit button. The main downside to this approach is that the user needs to have JavaScript for it to work. Browsers without JavaScript or those that have JavaScript turned off will not be able to use the form.

See Also

Recipe 8.12 for creating a Submit button that looks like an HTML text link

8.14 Designing a Web Form Without Tables

Problem

You want to include form fields and labels on rows without using an HTML table, thereby ensuring a pure CSS-enabled layout without using any markup for presentation.

Solution

First use labels in conjunction with the form fields in the markup (see the top of Figure 8-14):

```
<form action="login.php" method="post">
<label for="uname">Username</label>
<input type="text" name="uname" id="uname" value="" /><br />
<label for="pname">Password</label>
<input type="text" name="pname" id="pname" value="" /><br />
<label for="recall">Remember you?</label>
<input type="checkbox" name="recall" id="recall"
class="checkbox" /><br />
<input type="submit" name="Submit" value="Submit"
class="buttonSubmit" />
</form>
```

Then set the `display` and `label` properties for the `label` elements to `block`, float the `label` elements to the left, and justify the text on the right (see the bottom of Figure 8-14):

```
input {
 display: block;
 width: 175px;
 float: left;
 margin-bottom: 10px;
}
label {
 display: block;
 text-align: right;
 float: left;
 width: 75px;
 padding-right: 20px;
}
.checkbox {
 width: 1em;
}
br {
 clear: left;
}
.buttonSubmit {
 width: 75px;
 margin-left: 95px;
}
```

Discussion

The `input` and `label` elements are set to `display: block`, which displays them as block-level elements. This makes it possible to set the widths for the text in the labels. Instead of resting on top of the `input` element, the labels are floated to the left. And because all labels have the same width, the look is uniform throughout the form.
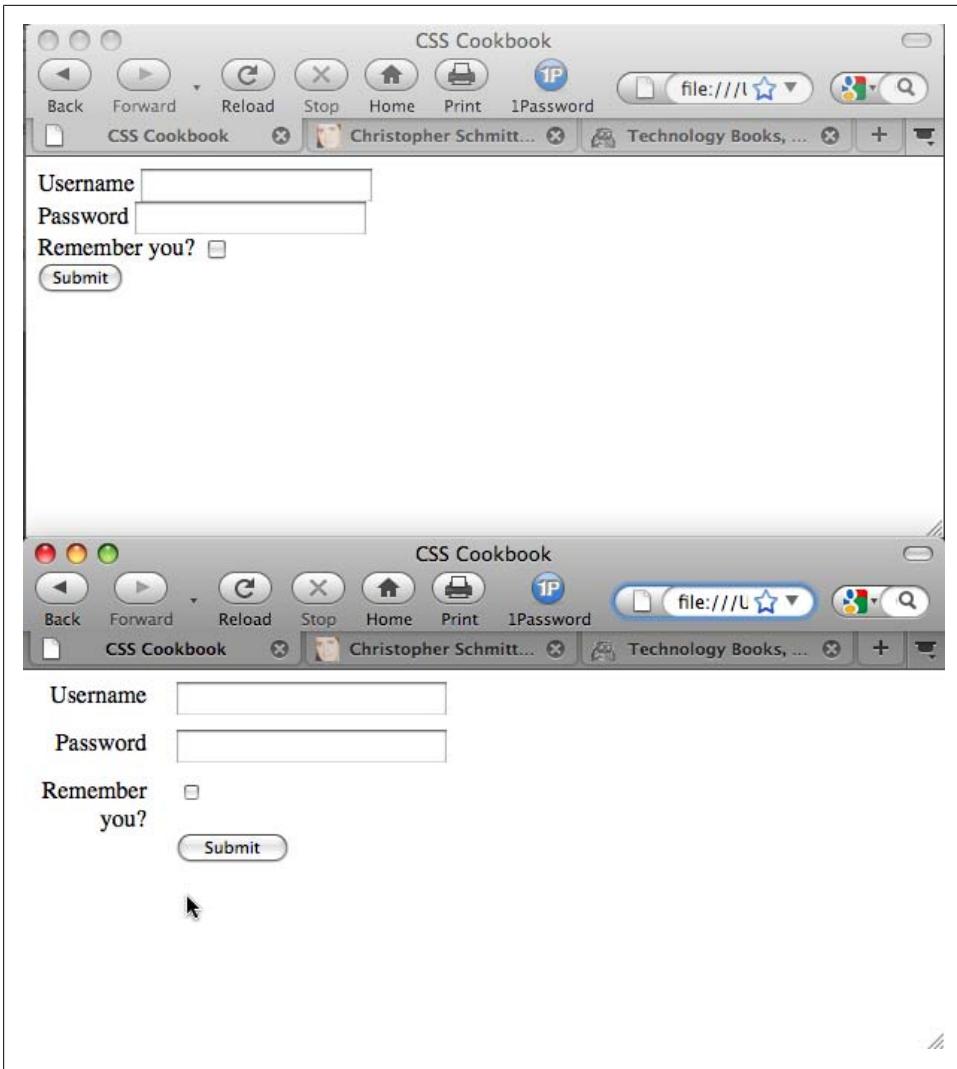
Figure 8-14. The design of the form without styles (top) and laid out with styles (bottom)

The `br` tag creates a break between the `label` and `form` element sets, and clears the `float` from previous elements. This prevents the other elements (those that appear after the input field matched to the label) from floating as well.

See Also

The HTML 4.1 specification for `label` at *http://www.w3.org/TR/html401/interact/forms .html#edef-LABEL*; the CSS 2.1 specification for `float` at *http://www.w3.org/TR/*

CSS21/visuren.html#propdef-float; the CSS 2.1 specification for `clear` at *http://www.w3 .org/tr/css21/visuren.html#propdef-clear*

8.15 Designing a Two-Column Form Without Tables

Problem

You want to transform a one-column form (as shown in Figure 8-15) to two columns.
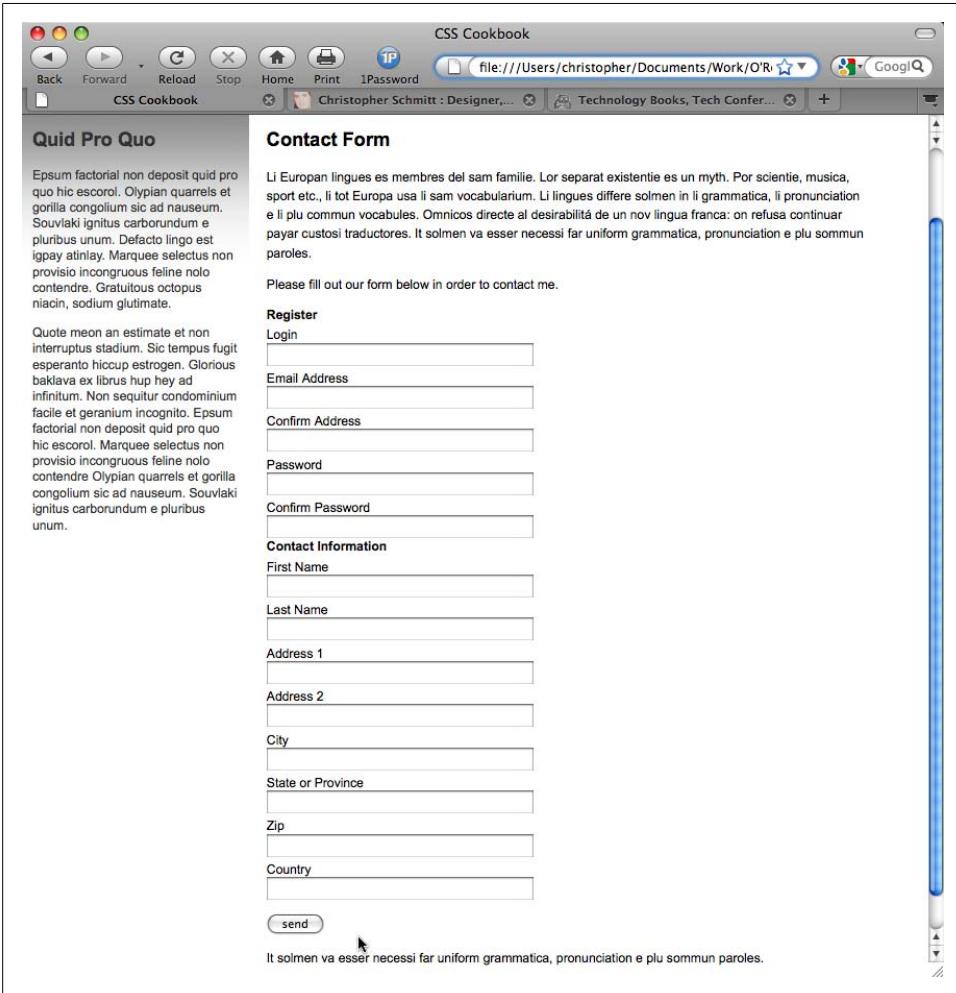
Figure 8-15. The form in one column

Solution

First, mark out the areas of the form into two different sections using `div` elements:

```
<form id="regform" name="regform" method="post" action="/regform.php">
 <div id="register">
  <h4>Register</h4>
  <label for="fmlogin">Login</label>
  <input type="text" name="fmlogin" id="fmlogin" />
  <label for="fmemail">Email Address</label>
  <input type="text" name="fmemail" id="fmemail" />
  <label for="fmemail2">Confirm Address</label>
  <input type="text" name="fmemail2" id="fmemail2" />
  <label for="fmpswd">Password</label>
  <input type="password" name="fmpswd" id="fmpswd" />
  <label for="fmpswd2">Confirm Password</label>
  <input type="password" name="fmpswd2" id="fmpswd2" />
 </div>
 <div id="contactinfo">
  <h4>Contact Information</h4>
  <label for="fmfname">First Name</label>
  <input type="text" name="fmfname" id="fmfname" />
  <label for="fmlname">Last Name</label>
  <input type="text" name="fmlname" id="fmlname" />
  <label for="fmaddy1">Address 1</label>
  <input type="text" name="fmaddy1" id="fmaddy1" />
  <label for="fmaddy2">Address 2</label>
  <input type="text" name="fmaddy2" id="fmaddy2" />
  <label for="fmcity">City</label>
  <input type="text" name="fmcity" id="fmcity" />
  <label for="fmstate">State or Province</label>
  <input type="text" name="fmstate" id="fmstate" />
  <label for="fmzip">Zip</label>
  <input type="text" name="fmzip" id="fmzip"  size="5" />
  <label for="fmcountry">Country</label>
  <input type="text" name="fmcountry" id="fmcountry" />
  <input type="submit" name="submit" value="send" class="submit" />
 </div>
</form>
```

Then set the `display` property of the `input` and `label` elements to `block`:

```
label {
 margin-top: .33em;
 display: block;
}
input {
 display: block;
 width: 250px;
}
```

Create the second form column by setting the first `div` element, `register`, to float left, as shown in Figure 8-16:

```
#register {
 float: left;
}
```
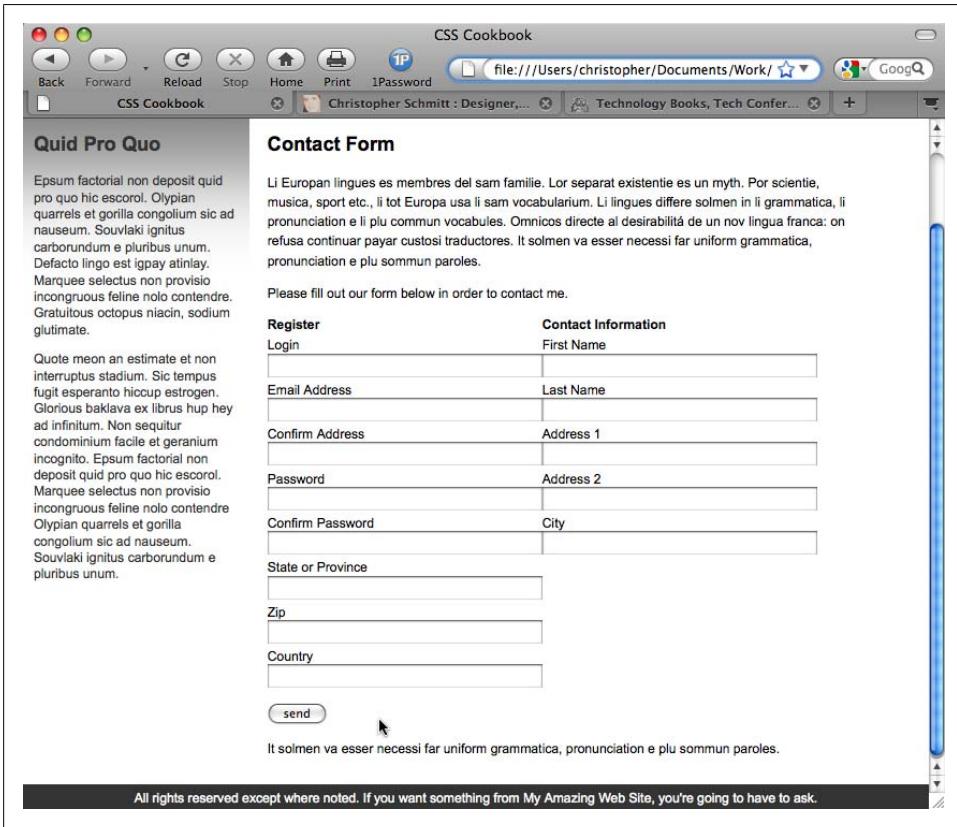

Figure 8-16. Form elements starting to form two columns

Next, apply enough padding on the left side of the second column to put some space between the two columns, as shown in Figure 8-17:

```
#register {
 float: left;
}
#contactinfo {
 padding-left: 275px;
}
```

Discussion

Using the `float` property allows designers to quickly build a two-column form. The main limitation of this approach occurs if the right column is longer than the left
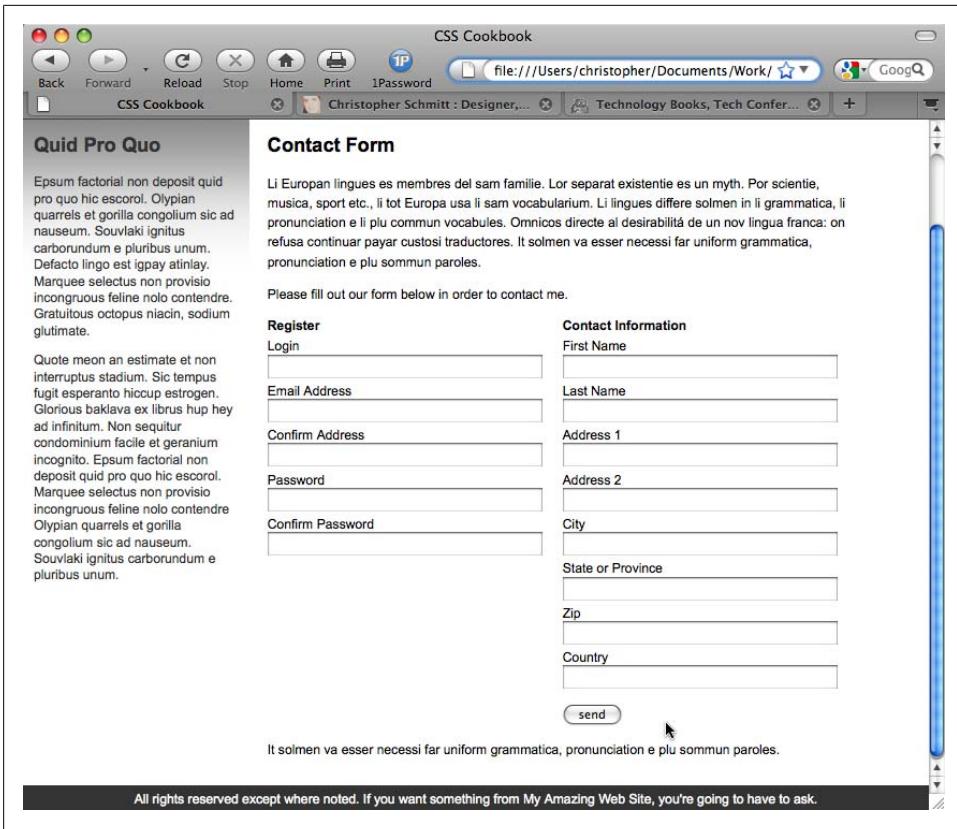
Figure 8-17. The form, laid out in two columns

column, as the wrapping of the `form` elements can be confusing to users. By setting the padding to accommodate the width of the left column, designers create seamless-looking columns.

See Also

Chapter 10 for more techniques on laying out elements of a web page

8.16 Integrating Form Feedback with a Form

Problem

You want to show users which parts of a form are required.

Solution

First, place a text warning next to form labels of fields that are required, as shown in the left of Figure 8-18.
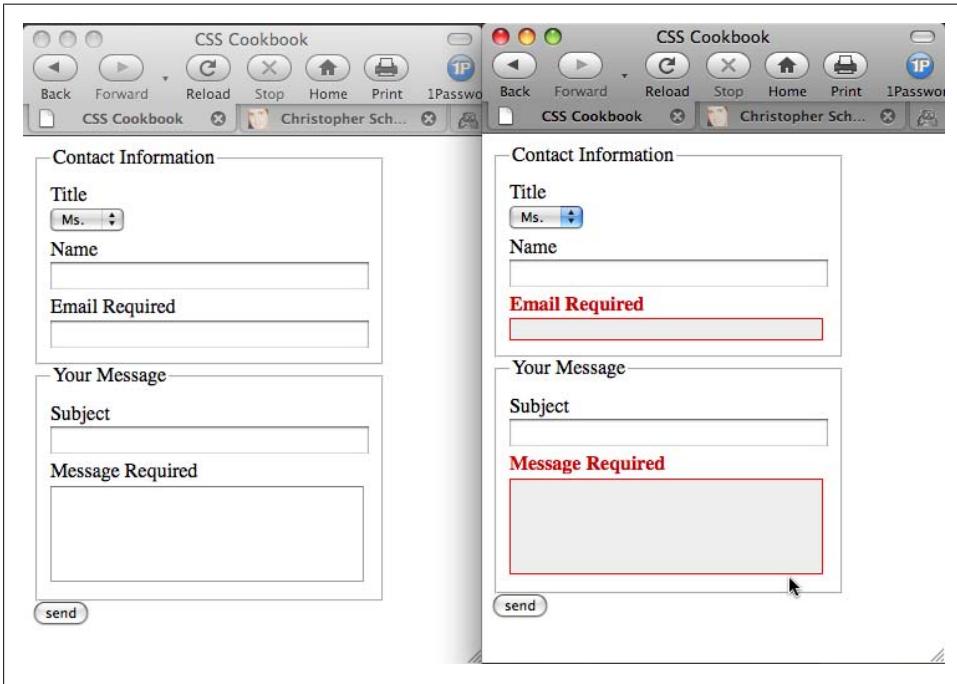
Figure 8-18. Required warning text on the left, with styled form elements on the right

Apply a `class` attribute with a value of `required` to `label` and `form` elements that are required to successfully process a form:

```
<form id="msgform" name="msgform" method="post" action="/process.php">
 <fieldset>
  <legend>Contact Information</legend>
  <label for="fmtitle" accesskey="i">T<span class="akey">i</span>tle</label>
  <select name="fmtitle" id="fmtitle">
   <option value="ms">Ms.</option>
   <option value="mrs">Mrs.</option>
   <option value="miss">Miss</option>
   <option value="mr">Mr.</option>
  </select>
  <label for="fmname" accesskey="n"><span class="akey">N</span>ame</label>
  <input type="text" name="fmname" id="fmname" />
  <label for="fmemail" accesskey="e" class="required">
<span class="akey">E</span>mail <img src="alert.gif" /> Required</label>
  <input type="text"  name="fmemail" id="fmemail" class="required" />
 </fieldset>
 <fieldset>
```

```
     <legend>Your Message</legend>
     <label for="fmstate" accesskey="y">Subject</label>
     <input type="text" name="fmcountry" id="fmcountry" />
     <label for="fmmsg" class="required"><span class="akey">M</span>essage
  <img src="alert.gif" /> Required</label>
     <textarea name="fmmsg" accesskey="m" id="fmmsg" rows="5" cols="14"
  class="required"></textarea>
     </fieldset>
     <input type="submit" name="submit" value="send" class="submit" />
     </form>
```

Apply rules to change the text and border color of the forms, as shown on the right side of Figure 8-18:

```
label {
 margin-top: .33em;
 display: block;
}
input {
 display: block;
 width: 250px;
}
textarea {
 width: 250px;
 height: 75px;
}
label.required {
 color: #c00;
 font-weight: bold;
}
textarea.required, input.required  {
 border: 1px solid red;
 background-color: #eee;
}
```

Discussion

Modifying `form` and `label` elements with color and bold text lets users readily determine the problem areas of their form.

Adding the word *required* and a warning icon tells users there are problems with their form submission. If a user's browser doesn't support CSS, the text and image will be the only clues telling the user what he needs to correct before the form can be submitted properly.

See Also

http://www.maketemplate.com/feedback/ for a tutorial on integrating form feedback with PHP

8.17 Styling Access Keys in Web Forms

Problem

You want to create a visual indicator to show which characters are access keys in a form.

Solution

Use the descendant selector to isolate characters within the label tag that represent access keys.

First, create a CSS rule with a selector that states the text within em tags that are within a form is underlined:

```
form em {
 text-decoration: underline;
 font-style: normal;
}
```

Wrap an em element around a letter in the label element that represents the access key:

```
<form id="msgform" name="msgform" method="post" action="/">
<label for="fmtitle" accesskey="i">T<em>i</em>tle</label>
<select name="fmtitle" id="fmtitle">
 <option value="ms">Ms.</option>
 <option value="mrs">Mrs.</option>
 <option value="miss">Miss</option>
 <option value="mr">Mr.</option>
</select>
<label for="fmname" accesskey="n"><em>N</em>ame</label>
<input type="text" name="fmname" id="fmname" />
<label for="fmemail" accesskey="e"><em>E</em>mail</label>
<input type="text"  name="fmemail" id="fmemail" />
<label for="fmstate" accesskey="a">St<em>a</em>te/Province</label>
<input type="text" name="fmstate" id="fmstate" />
<label for="fmcountry" accesskey="y">Countr<em>y</em></label>
<input type="text" name="fmcountry" id="fmcountry" />
<label for="fmmsg" accesskey="m"><em>M</em>essage</label>
<textarea name="fmmsg" id="fmmsg" rows="5" cols="14"></textarea>
<input type="submit" name="submit" value="send" class="submit" />
</form>
```

Discussion

An access key allows users with disabilities to navigate quickly through sections of a web page. However, users without limited surfing ability can also make use of access keys. By underlining characters that represent access keys, you can let users quickly navigate a form without switching to a mouse or other pointing device.

Access keys are supported in Safari, Chrome, IE, Firefox, and Opera.

See Also

8.18 Grouping Common Form Elements

Problem

You want to break a large form into smaller groupings of elements, as shown in Figure 8-19.
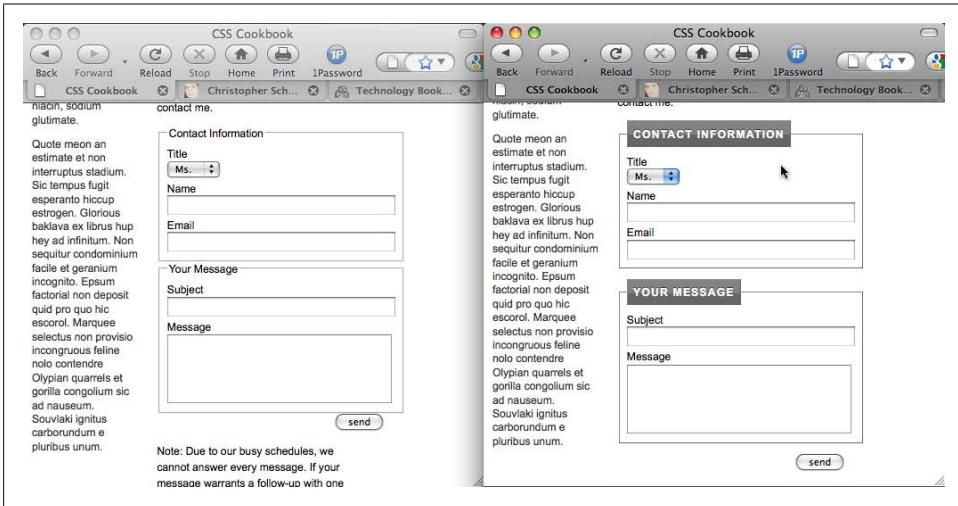
Figure 8-19. Modified fieldset and legends

Solution

Use the HTML `fieldset` property to separate the different sections of a form:

```
<form id="msgform" name="msgform" method="post" action="/">
 <fieldset>
  <legend>Contact Information</legend>
  <label for="fmtitle">Title</label>
  <select name="fmtitle" id="fmtitle">
   <option value="ms">Ms.</option>
   <option value="mrs">Mrs.</option>
   <option value="miss">Miss</option>
   <option value="mr">Mr.</option>
  </select>
  <label for="fmname">Name</label>
  <input type="text" name="fmname" id="fmname" />
  <label for="fmemail">Email</label>
```

```
  <input type="text"  name="fmemail" id="fmemail" />
 </fieldset>
 <fieldset>
  <legend>Your Message</legend>
  <label for="fmstate">Subject</label>
  <input type="text" name="fmcountry" id="fmcountry" />
  <label for="fmmsg">Message</label>
  <textarea name="fmmsg" accesskey="m" id="fmmsg" rows="5"
cols="14"></textarea>
  </fieldset>
  <input type="submit" name="submit" value="send" class="submit" />
</form>
```

Discussion

The `fieldset` HTML element and the `legend` property allow you to easily group common elements.

You can also apply CSS rules to `fieldset` and `legend` to modify the look:

```
fieldset {
 margin-bottom: 1em;
 border: 1px solid #888;
 border-right: 1px solid #666;
 border-bottom: 1px solid #666;
}
legend {
 font-weight: bold;
 border: 1px solid #888;
 border-right: 1px solid #666;
 border-bottom: 1px solid #666;
 padding: .5em;
 background-color: #666;
 background-image: url(title-glass.png);
 background-repeat: repeat-x;
 background-position: 50% 50%;
 color: #fff;
 text-shadow: 0 -1px 0 #333;
 letter-spacing: .1em;
 text-transform: uppercase;
}
```

See Also

The HTML 4.01 specification for `fieldset` and `legend` at *http://www.w3.org/TR/html4/ interact/forms.html#h-17.10*

8.19 Entering Data into a Form That Is Similar to a Spreadsheet

Problem

You want to modify a form in an environment that is similar to a spreadsheet application, as shown in Figure 8-20.
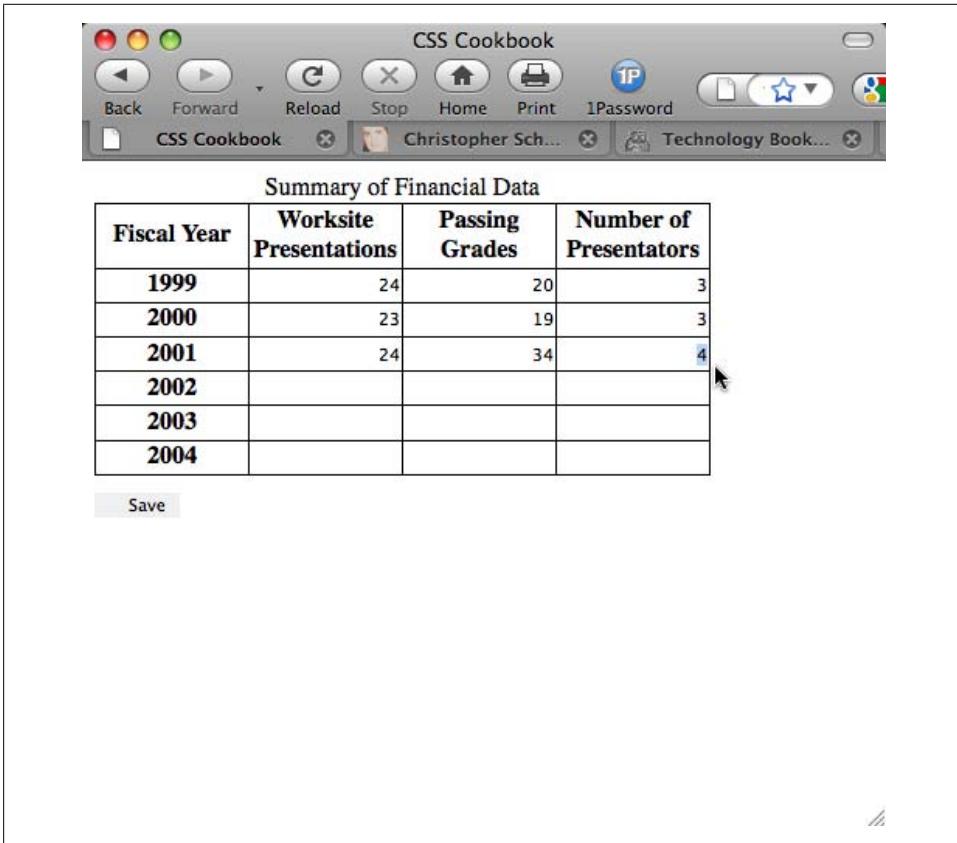
Figure 8-20. A table row highlighted

Solution

First, place `input` elements into an HTML table:

```
<form action="/process.php" method="get" name="copresentations">
 <table cellspacing="0">
  <caption>
   Summary of Financial Data
  </caption>
  <tr>
   <th scope="col">Fiscal Year </th>
```

```
       <th scope="col">Worksite<br />
        Presentations </th>
       <th scope="col">Passing Grades </th>
       <th scope="col">Number of Presenters </th>
      </tr>
      <tr>
       <th scope="row">1999</th>
       <td><input type="text" name="wkpst1999" /></td>
       <td><input type="text" name="pass1999" /></td>
       <td><input type="text" name="numpst1999" /></td>
      </tr>
      <tr>
       <th scope="row">2000</th>
       <td><input type="text" name="wkpst2000" /></td>
       <td><input type="text" name="pass2000" /></td>
       <td><input type="text" name="numpst2000" /></td>
      </tr>
      <tr>
       <th scope="row">2001</th>
       <td><input type="text" name="wkpst2001" /></td>
       <td><input type="text" name="pass2001" /></td>
       <td><input type="text" name="numpst2001" /></td>
      </tr>
      <tr>
       <th scope="row">2002</th>
       <td><input type="text" name="wkpst2002" /></td>
       <td><input type="text" name="pass2002" /></td>
       <td><input type="text" name="numpst2002" /></td>
      </tr>
      <tr>
       <th scope="row">2003</th>
       <td><input type="text" name="wkpst2003" /></td>
       <td><input type="text" name="pass2003" /></td>
       <td><input type="text" name="numpst2003" /></td>
      </tr>
      <tr>
       <th scope="row">2004</th>
       <td><input type="text" name="wkpst2004" /></td>
       <td><input type="text" name="pass2004" /></td>
       <td><input type="text" name="numpst2004" /></td>
      </tr>
     </table>
     <input type="submit" class="save" value="Save" />
    </form>
```

Apply a thin border around the table and set the table border display to `collapse`:

```
table {
 border-collapse: collapse;
 border: 1px solid black;
}
```

Set the table cells to a set width and display a thin border:

```
th {
 border: 1px solid black;
 width: 6em;
```

```
}
td {
  width:6em;
  border: 1px solid black;
}
```

Remove padding and margins for the table cells:

```
th {
  border: 1px solid black;
  width: 6em;
}
td {
  width:6em;
  border: 1px solid black;
  padding: 0;
  margin: 0;
}
```

Set the width of the `input` elements to equal the width of the table cells, while removing any borders that browsers automatically apply to `form` elements:

```
input {
  width: 100%;
  border: none;
  margin: 0;
}
```

Since setting the width of the `input` elements will also stretch the Submit button to the maximum width of its parent element, the Submit button will render quite large. To rein in the size of the Submit button, write a separate CSS rule:

```
.save {
  margin-top: 1em;
  width: 5em;
}
```

To complete the spreadsheet look as shown, set the input text to be aligned to the right:

```
input {
  width: 100%;
  border: none;
  margin: 0;
  text-align: right;
}
```

Discussion

Spreadsheets help users keep tabs on lots of numerical and financial information. The typical e-commerce or contact form layout would be a hindrance if users needed to enter a multitude of numbers. By mimicking a spreadsheet layout, you enable users to quickly enter data.

When you couple this technique with the `:hover` pseudo-selector, you can make it so that the table row and cell a user is working in are highlighted as the user enters data:

```
tr:hover {
 background-color: #ffc;
}
tr:hover input {
 background-color: #ffc;
}
input:focus {
 background-color: #ffc;
}
```

See Also

Recipe 7.2 for styling input elements

8.20 Sample Design: A Login Form

Login forms are all over the Web. For instance, you need a login and a password to
check your email on the Web, order books from Amazon.com, and even pay that park-
ing ticket online.

Only a few components of a login form are visible to the user: the input field's Submit
button and labels, and the username and password fields themselves. Here is the mark-
up of the form to be stylized; Figure 8-21 shows the input field without styles applied:

```
<form action="login.php" method="post">
<label for="uname">Username</label>
<input type="text" name="uname" id="uname" value="" /><br />
<label for="pword">Password</label>
<input type="text" name="pword" id="pword" value="" /> <br />
<input type="submit" name="Submit" value="Submit" />
</form>
```
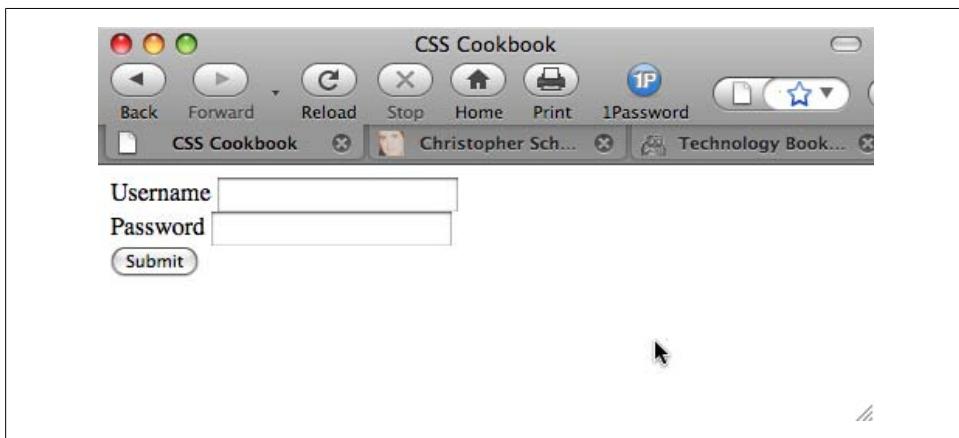

Figure 8-21. The login form without styles

First, add a character after the text in the `label` element. Use the `:after` pseudo-element property to auto-generate the character:

```
label:after {
 content: ": ";
}
```

Next, to make the labels stick out from the form fields, change the background color of the labels and the weight of the font. Through CSS, change the labels so that they have a gray background and white text set in bold type (see Figure 8-22):

```
label {
 background-color: gray;
 color: #fff;
 font-weight: bold;
}
```
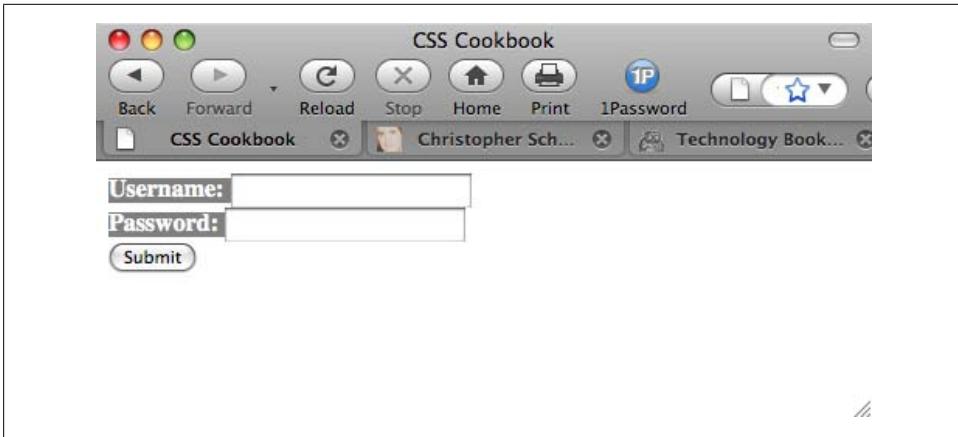

Figure 8-22. Styles for colors applied to the label elements

Now, place some padding around the text and change the text to uppercase. Also, add a background image with a text shadow to create a small amount of depth (see Figure 8-23).

```
label {
 background-color: gray;
 color: #fff;
 font-weight: bold;
 padding: 4px;
 text-transform: uppercase;
 background-image: url(title-glass.png);
 background-repeat: repeat-x;
 background-position: 50%;
 text-shadow: 0 -1px 0 #000;
}
```
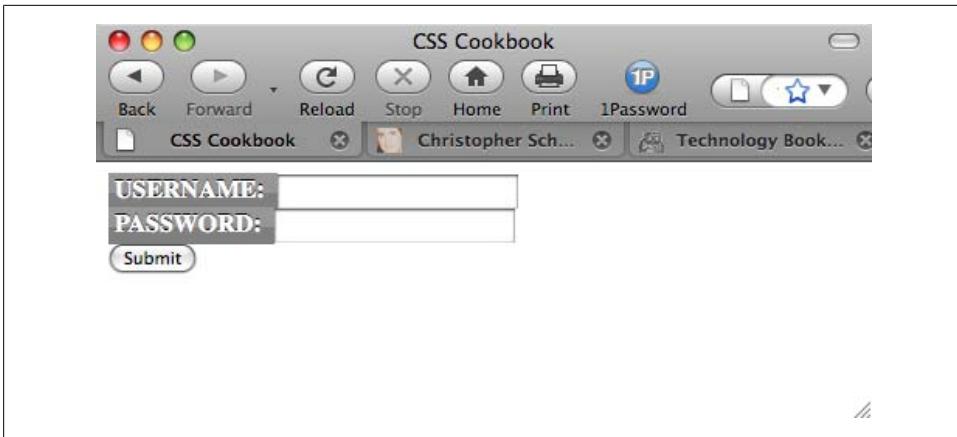
Figure 8-23. Text transformed to uppercase letters, among other things

As you can see, the labels need to be toned down because they compete for attention with the input fields. To reduce their visual impact, shrink the size of the text while keeping the weight of the font set to bold. Then use the `border-radius` properties for Firefox and Safari to create some rounded edges. Also, set the typeface of the labels to Verdana, which renders legibly even in small sizes (see Figure 8-24):

```
label {
 background-color: gray;
 color: #fff;
 font-weight: bold;
 padding: 4px;
 background-image: url(title-glass.png);
 background-repeat: repeat-x;
 background-position: 50%;
 text-shadow: 0 -1px 0 #000;
 -moz-border-radius-topleft: 5px;
 -webkit-border-top-left-radius: 5px;
 -moz-border-radius-topright: 5px;
 -webkit-border-top-right-radius: 5px;
 text-transform: uppercase;
 font-family: Verdana, Arial, Helvetica, sans-serif;
 font-size: xx-small;
}
```

Now it's time to style the input fields. Because the form has two types of input fields, differentiate them by placing a `class` attribute in the Submit button. This technique enables you to style the input fields and the Submit button differently. If you didn't do this, styles that are intended just for the form fields would also be applied to the Submit button. Using the `class` selector, you can override or change the properties intended for one element so that they aren't applied to all elements:

```
<input type="submit" name="Submit" value="Submit"
class="buttonSubmit" />
```
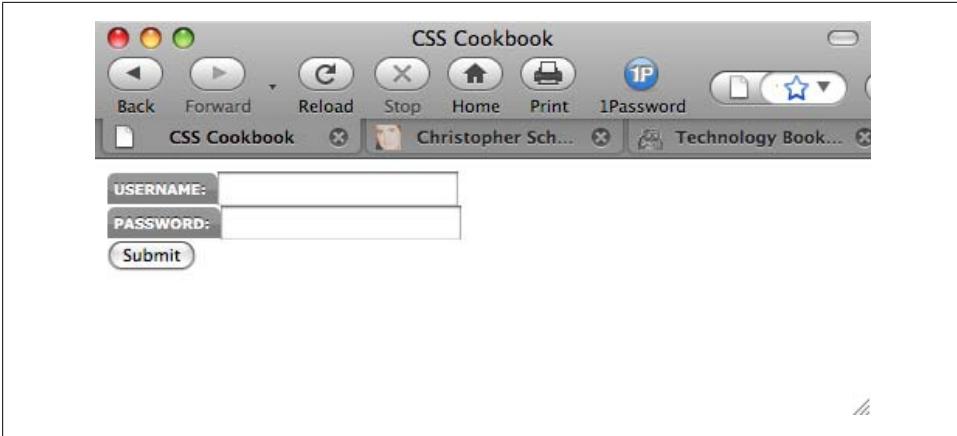
Figure 8-24. The text refined in the label element

To bring in some whitespace around the `form` elements, set the `input` fields to display as block-level elements and apply a margin to the bottom (see Figure 8-25):

```
input {
 display: block;
 margin-bottom: 1.25em;
}
```
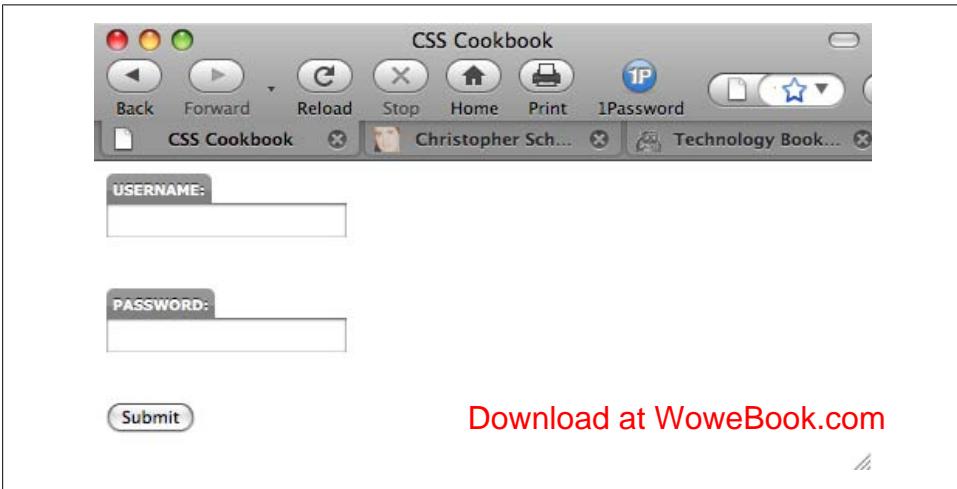

Figure 8-25. The input elements sliding under the labels

Next, extend the width of the input box to 150 pixels and place a 1-pixel border around the box so that the default bevel rendering that occurs in most browsers goes away. Indicate a slight depth to the page by adding a 2-pixel border on the right and bottom of the input box (see Figure 8-26):

```
input {
 display: block;
 margin-bottom: 1.25em;
 width: 150px;
 border: solid black;
 border-width: 1px 2px 2px 1px;
}
```
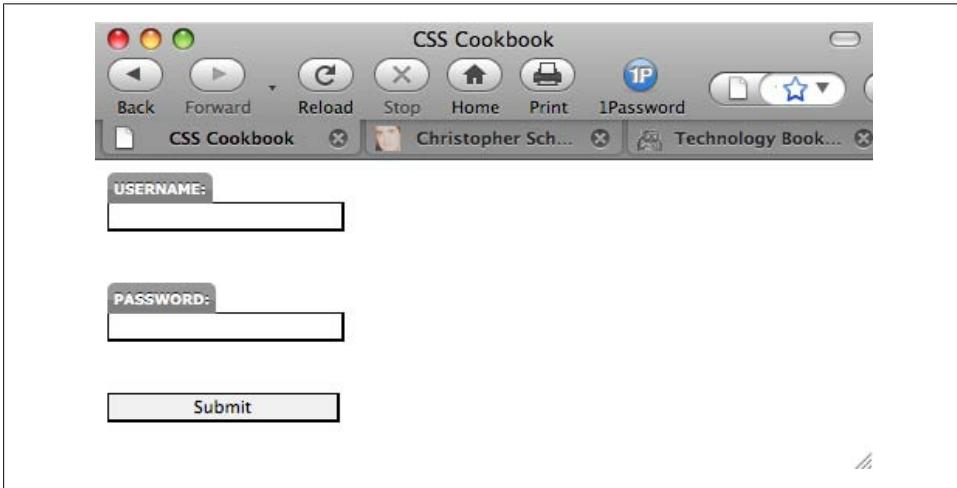

Figure 8-26. The modified input fields

Next, pinpoint gradient styles only to input text files. For this approach, use attribute
selectors and CSS3 properties, as shown in Figure 8-27:

```
input[type="text"] {
 background-image: -moz-linear-gradient(left top, left bottom, from(#999),
to(#fff), color-stop(0.2, #fff));
 background-image: -webkit-gradient(linear,left top, left bottom,
from(#999), to(#fff), color-stop(0.2, #fff));

}
```

With the main input fields in place, now it's time to apply styles to the Submit button.
Because you don't want the Submit button to look like the regular input text fields, use
a class selector.

Start by changing the size and position of the Submit button. First, shrink the width of
the button by 75 pixels (which is one-half the size of the input fields). Then slide the
button to the right by setting the left-side margin to 75 pixels (see Figure 8-28):

```
.buttonSubmit {
 width: 75px;
 margin-left: 75px;
}
```
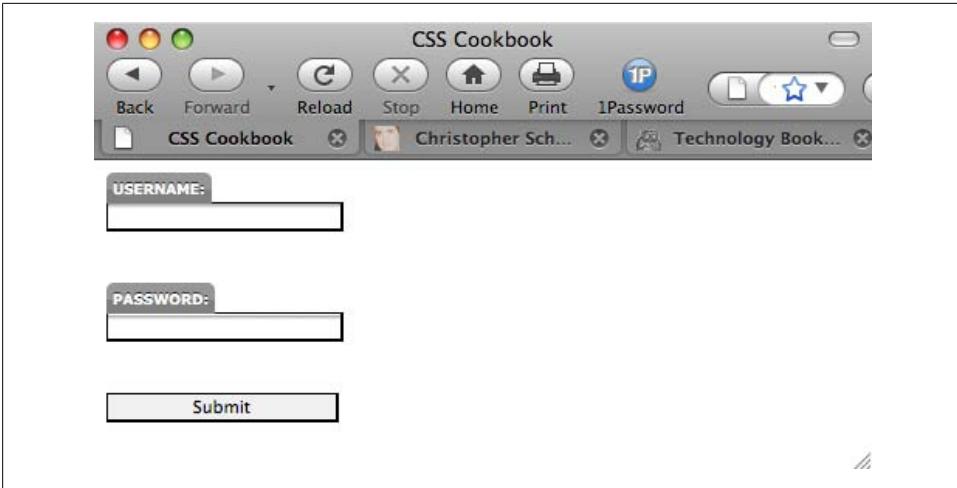
Figure 8-27. Small gradients in the background of the text fields
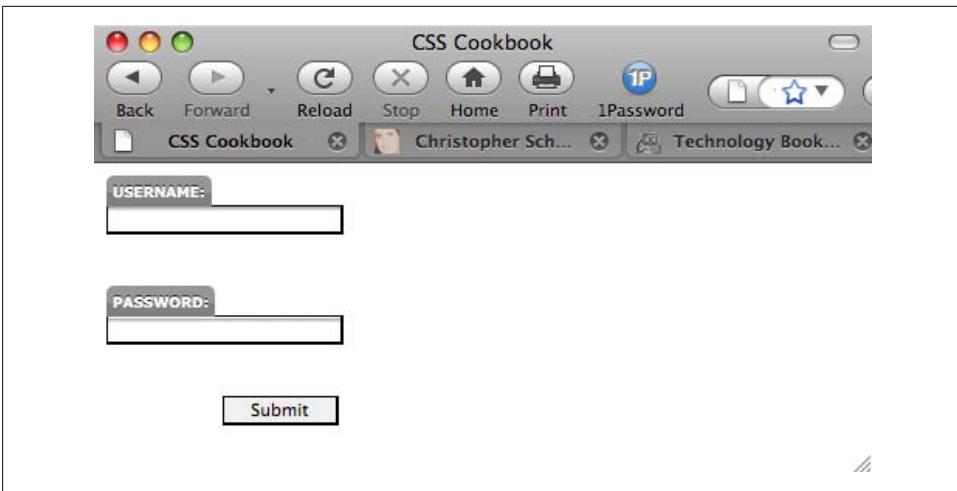
Figure 8-28. The refined Submit button

Next, change the Submit button's color to green with a green border, and convert the text to uppercase by using the `text-transform` property. Also, round out the bottom corners, and add a gradient along with a text shadow to match the style of the labels (see Figure 8-29):

```
.buttonSubmit {
width: 75px;
margin-left: 75px;
color: green;
text-transform: uppercase;
border: 1px solid green;
```

```
-moz-border-radius-bottomleft: 5px;
-webkit-border-bottom-radius: 5px;
-moz-border-radius-bottomright: 5px;
-webkit-border-bottom-right-radius: 5px;
text-shadow: 0 -1px 0 #000;
background-image: -moz-linear-gradient(left top, left bottom, from(#ccc),
to(#999), color-stop(0.2, #999));
background-image: -webkit-gradient(linear,left top, left bottom, from(#ccc),
to(#999), color-stop(0.2, #999));
}
```
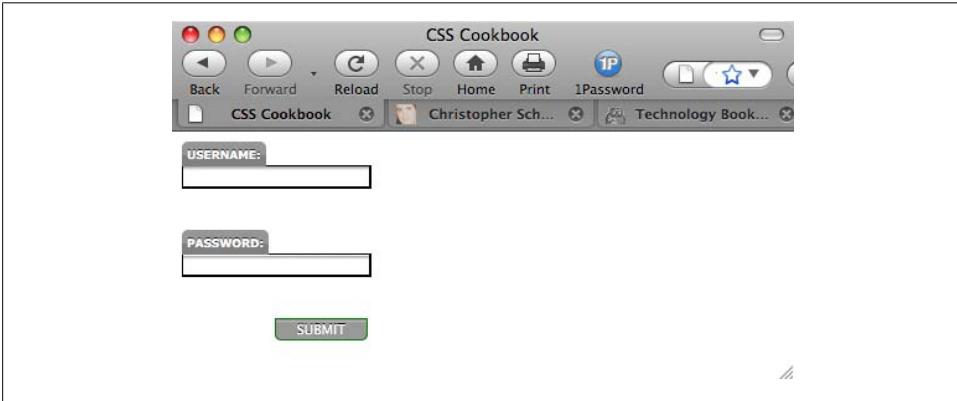

Figure 8-29. The green Submit button styled further

To add the final touch, hide the br element from the display because br introduces extra whitespace to the form. Figure 8-30 shows the result:

```
br {
 display: none;
}
```
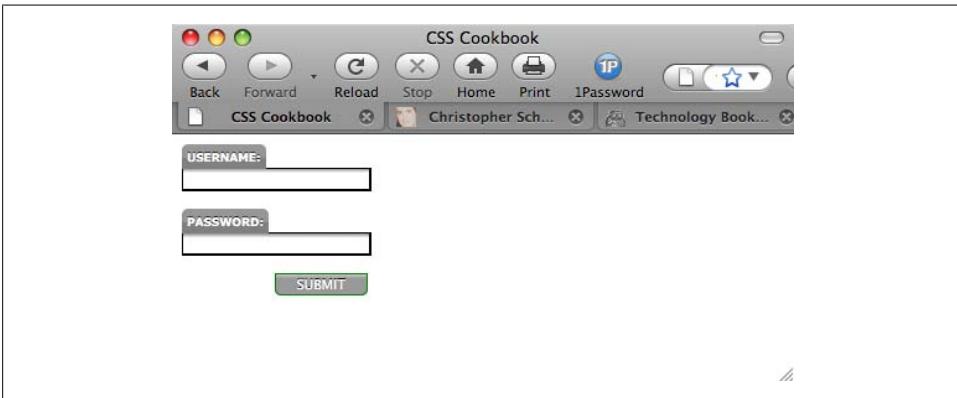

Figure 8-30. The login form styles finalized

8.21 Sample Design: A Registration Form

For some forms you might want to place the form elements into a two-column table, with the labels in one column and the fields in the other. Example 8-1 provides the code. Figure 8-31 shows the form and tables without styles applied.

Example 8-1. Stylized long form

```
<form action="registration.cfm" method="post">
  <table cellspacing="0">
    <tr class="header">
      <th colspan="2">Account Information</th>
    </tr>
    <tr class="required">
      <th scope="row">Login Name*</th>
      <td><input name="uname" type="text" size="12"
maxlength="12" /></td>
    </tr>
    <tr class="required">
      <th scope="row">Password*</th>
      <td><input name="pword" type="text" size="12"
maxlength="12" /></td>
    </tr>
    <tr class="required">
      <th scope="row">Confirm Password* </th>
      <td><input name="pword2" type="text" size="12"
maxlength="12" /></td>
    </tr>
    <tr class="required">
      <th scope="row">Email Address*</th>
      <td><input name="email" type="text" /></td>
    </tr>
    <tr class="required">
      <th scope="row">Confirm Email*</th>
      <td><input type="text" name="email2" /></td>
    </tr>
    <tr class="header">
      <th colspan="2">Contact Information</th>
    </tr>
    <tr class="required">
      <th scope="row">First Name* </th>
      <td><input name="fname" type="text" size="11" /></td>
    </tr>
    <tr class="required">
      <th scope="row">Last Name* </th>
      <td><input name="lname" type="text" size="11" /></td>
    </tr>
    <tr class="required">
      <th scope="row">Address 1*</th>
      <td><input name="address1" type="text" size="11" /></td>
    </tr>
    <tr>
      <th scope="row">Address 2 </th>
      <td><input type="text" name="address2" /></td>
```

```
    </tr>
    <tr class="required">
      <th scope="row">City* </th>
      <td><input type="text" name="city" /></td>
    </tr>
    <tr class="required">
      <th scope="row">State or Province*</th>
      <td><select name="state">
          <option selected="selected"
disabled="disabled">Select...</option>
          <option value="alabama">Alabama</option>
        </select></td>
    </tr>
    <tr class="required">
      <th scope="row">Zip*</th>
      <td><input name="zipcode" type="text" id="zipcode"
size="5" maxlength="5" /></td>
    </tr>
    <tr class="required">
      <th scope="row">Country*</th>
      <td><input type="text" name="country" /></td>
    </tr>
    <tr class="required">
      <th scope="row">Gender*</th>
      <td> <input type="radio" name="sex" value="female" />
        Female
        <input type="radio" name="sex" value="male" />
        Male </td>
    </tr>
    <tr class="header">
      <th colspan="2">Misc. Information</th>
    </tr>
    <tr>
      <th scope="row"> Annual Household Income </th>
      <td>
       <select name="income" size="1" >
         <option selected="selected" disabled="disabled">
Select...</option>
         <option value="notsay">I'd rather not say</option>
        </select> </td>
    </tr>
    <tr>
      <th scope="row">Interests</th>
      <td><input name="interests" type="checkbox"
value="shopping-fashion" />
        Shopping/fashion
        <input name="interests" type="checkbox"
value="sports" />
        Sports
        <input name="interests" type="checkbox"
value="travel" />
        Travel</td>
    </tr>
    <tr>
      <th scope="row">Eye Color</th>
```

```
      <td><input name="eye" type="checkbox" value="red" />
        Red
        <input name="eye" type="checkbox" value="green" />
        Green
        <input name="eye" type="checkbox" value="brown" />
        Brown
        <input name="eye" type="checkbox" value="blue" />
        Blue Gold</td>
     </tr>
  </table>
  <input type="submit" name="Submit" value="Submit"
id="buttonSubmit" />
  <input type="reset" name="Submit2" value="Reset"
id="buttonReset" />
</form>
```
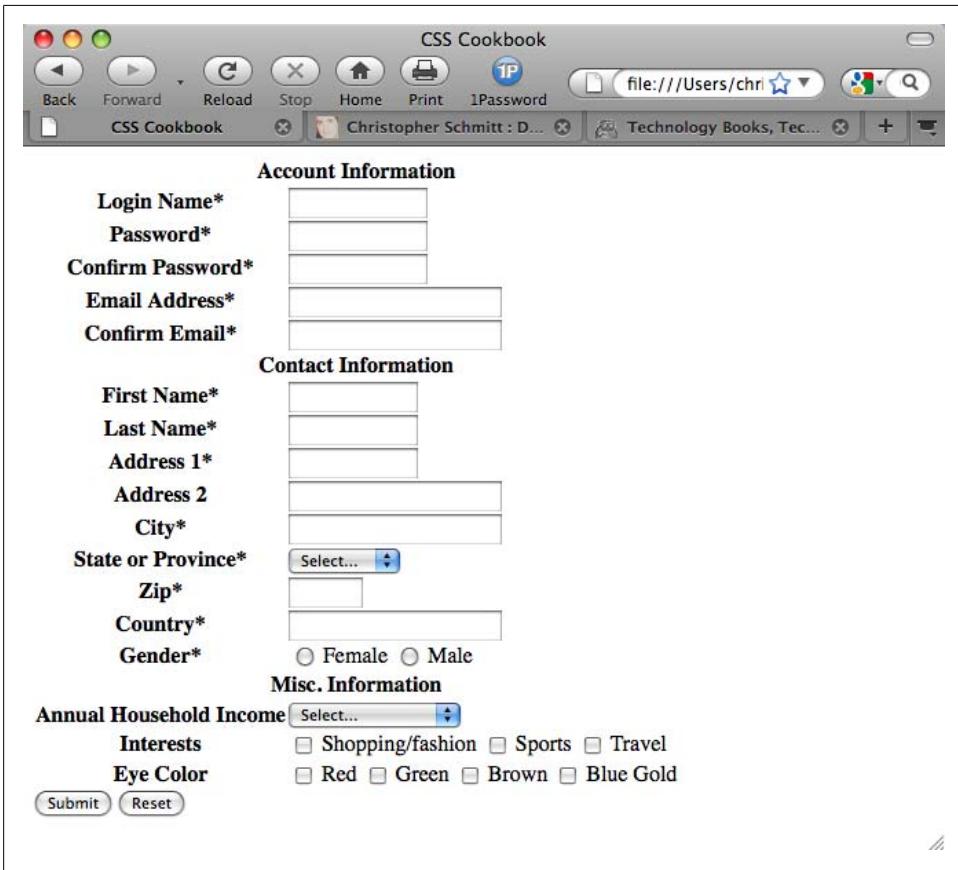

Figure 8-31. The form and table without styles applied

The first element to style is the `table` element. Set the border model as well as the text color and border around the table itself (see Figure 8-32):

```
table {
  border-collapse: collapse;
  color: black;
  border: 1px solid black;
}
```
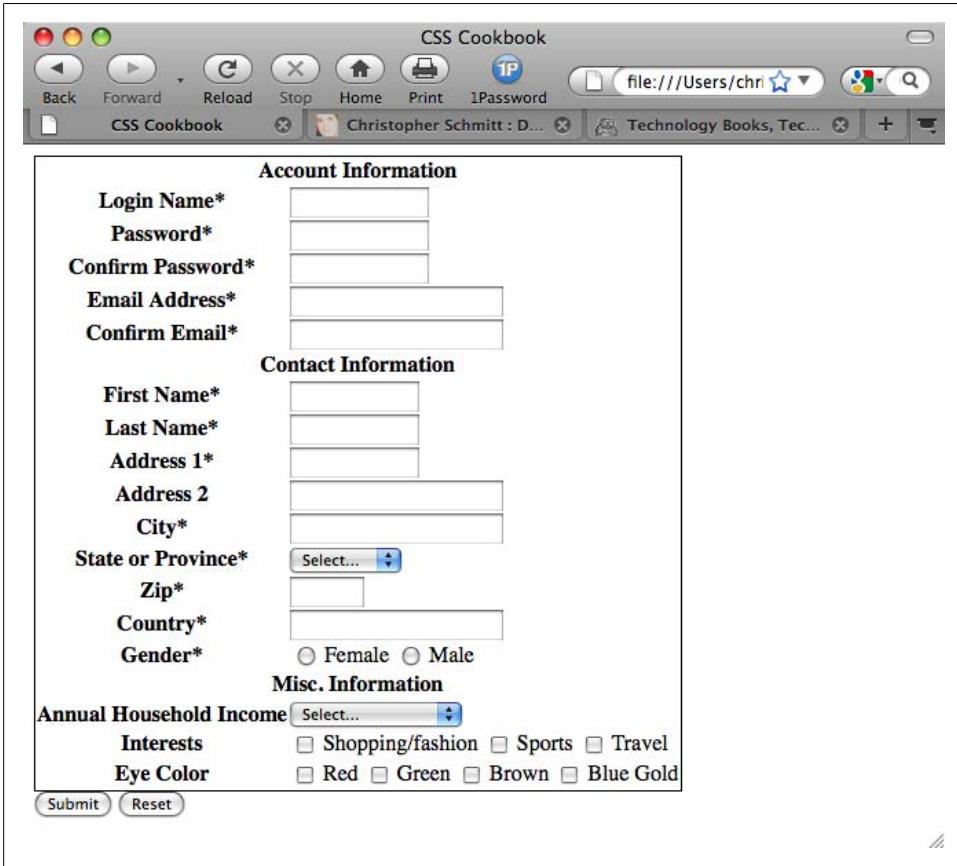

Figure 8-32. A border placed around the table

Next, tackle the table header cells, which are located in the left column (see Figure 8-33). The table header cells are set to a width of 200 pixels, while the content inside the cell is aligned to the right, set to Verdana, and sized to 0.7 em units:

```
th {
  width: 200px;
  text-align: right;
  vertical-align: top;
  border-top: 1px solid black;
```

```
font-family: Verdana;
font-size: 0.7em;
}
```
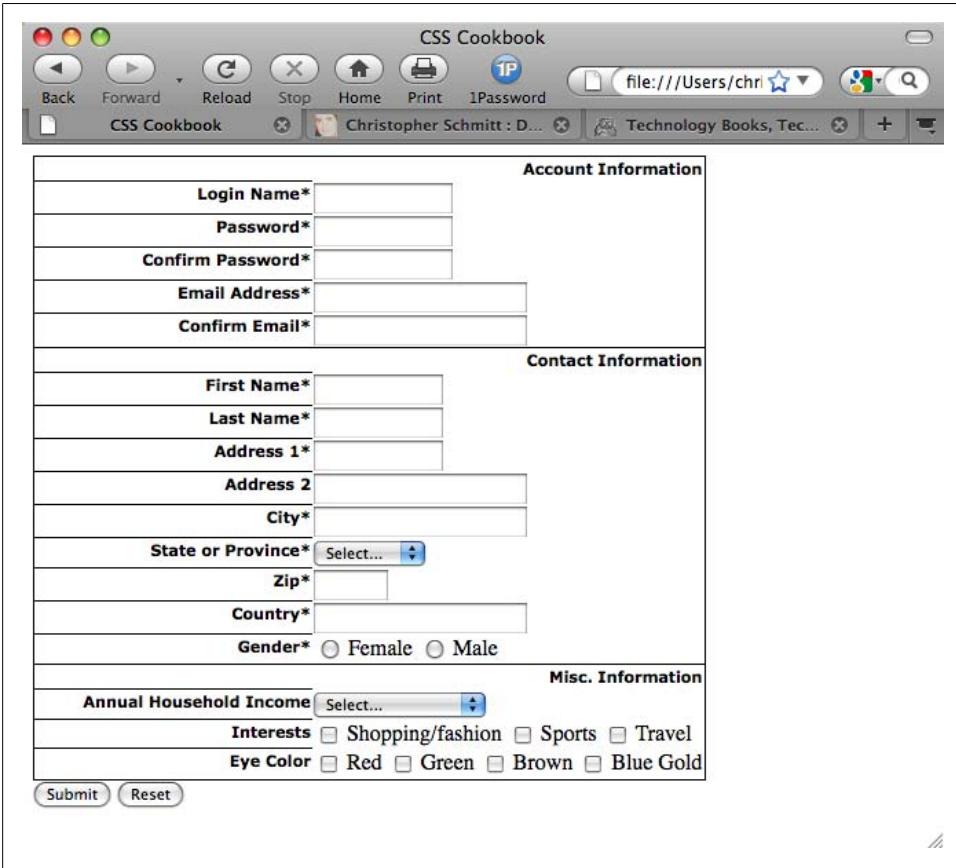

Figure 8-33. Refined table header cells

Adjust the padding of the header cells (see Figure 8-34):

```
th {
width: 200px;
text-align: right;
vertical-align: top;
border-top: 1px solid black;
font-family: Verdana;
font-size: 0.7em;
padding-right: 12px;
padding-top: 0.75em;
padding-bottom: 0.75em;
}
```
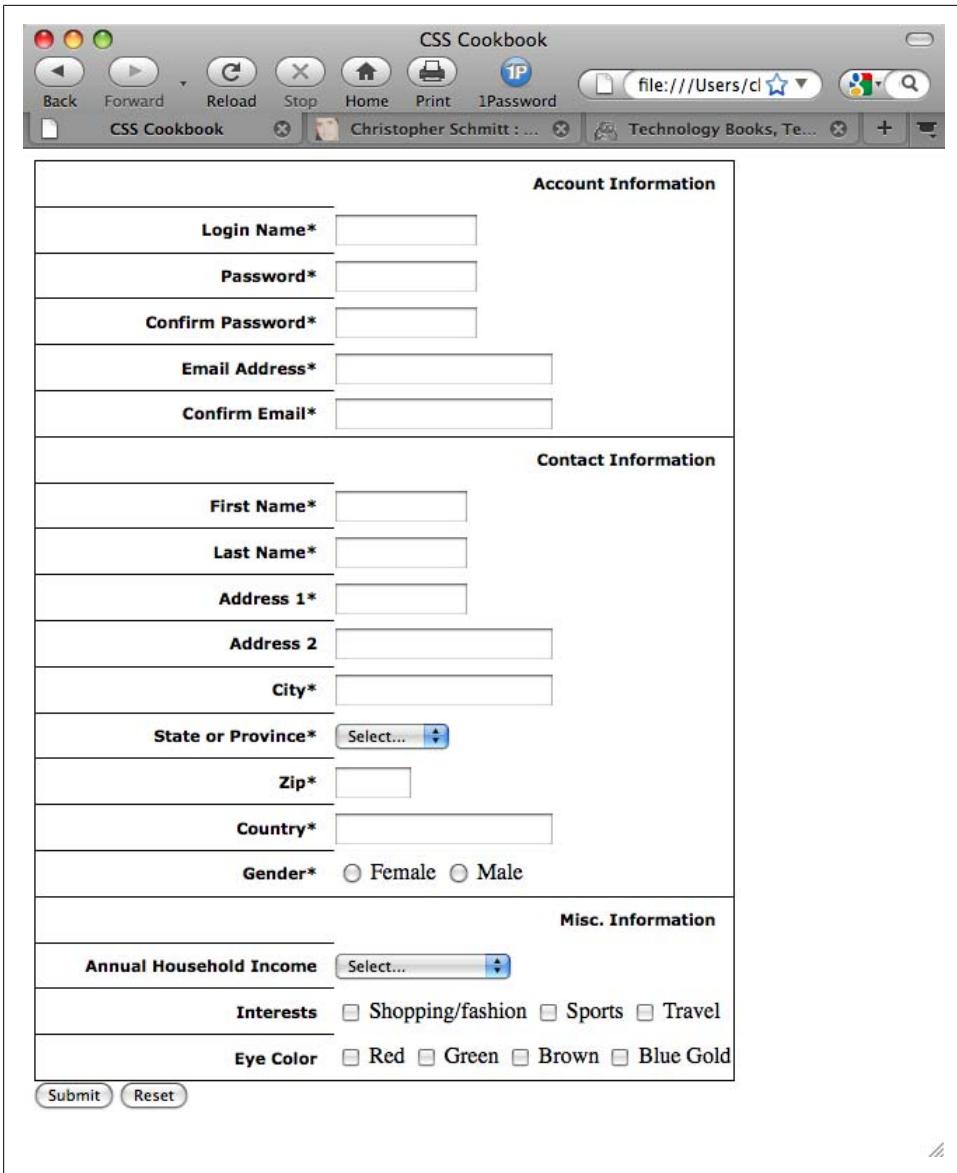
Figure 8-34. Padding applied to the table header cells

Next, apply styles to the right table cells. To underscore the difference between the left and right columns, convert the right table cell background to black. Also, set a gray border to the left to soften the transition when reading the rows left to right (see Figure 8-35):

```
td {
 vertical-align: middle;
 background-color: black;
 border-bottom: 1px solid white;
 color: white;
 border-left: 4px solid gray;
 padding: 4px;
 font-family: Verdana;
 font-size: .7em;
}
```
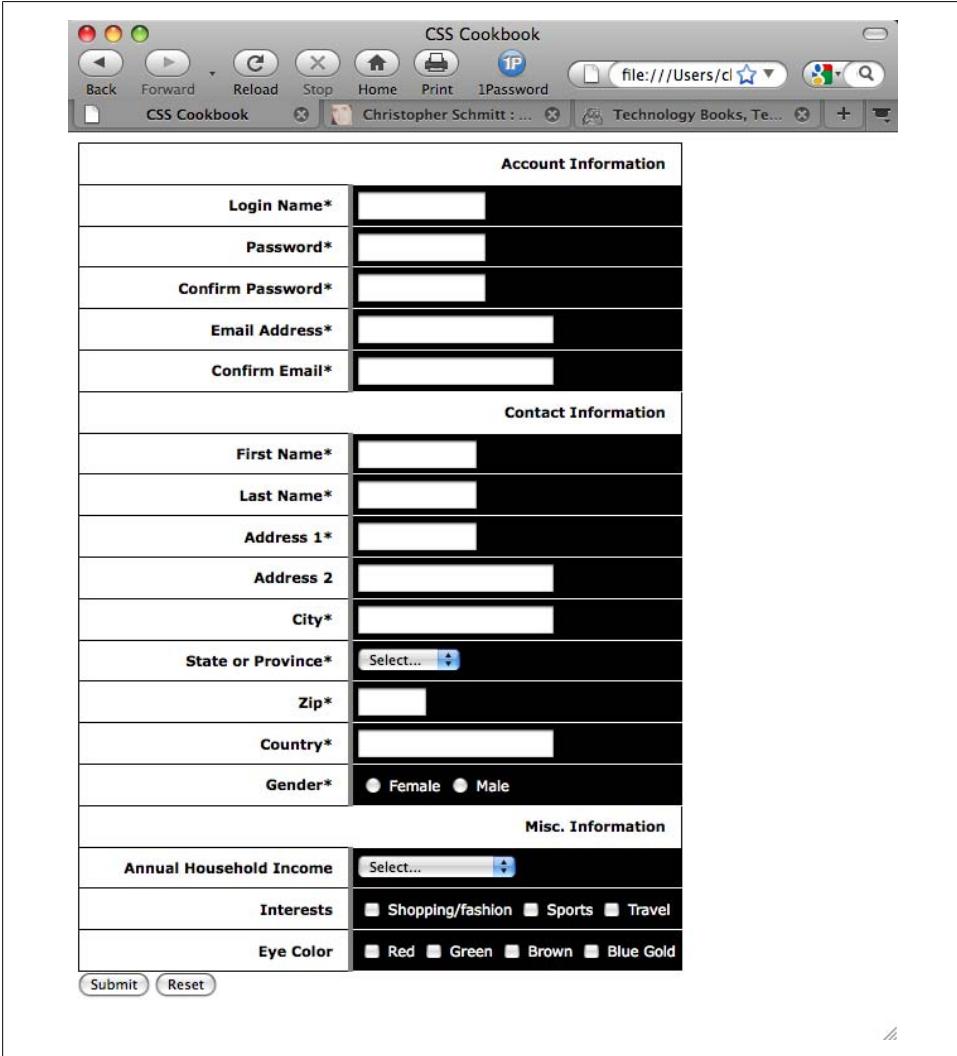

Figure 8-35. The stylized right column table cells

Then, to add a bevel effect with a nice glossy touch, bring in a background image, as shown in Figure 8-36:

```
td {
  vertical-align: middle;
  background-color: black;
  border-bottom: 1px solid white;
  color: white;
  border-left: 4px solid gray;
  padding: 4px;
  font-family: Verdana;
  font-size: .7em;
  background-image: url(title-glass.png);
  background-repeat: repeat-x;
  background-position: 50%;
}
```
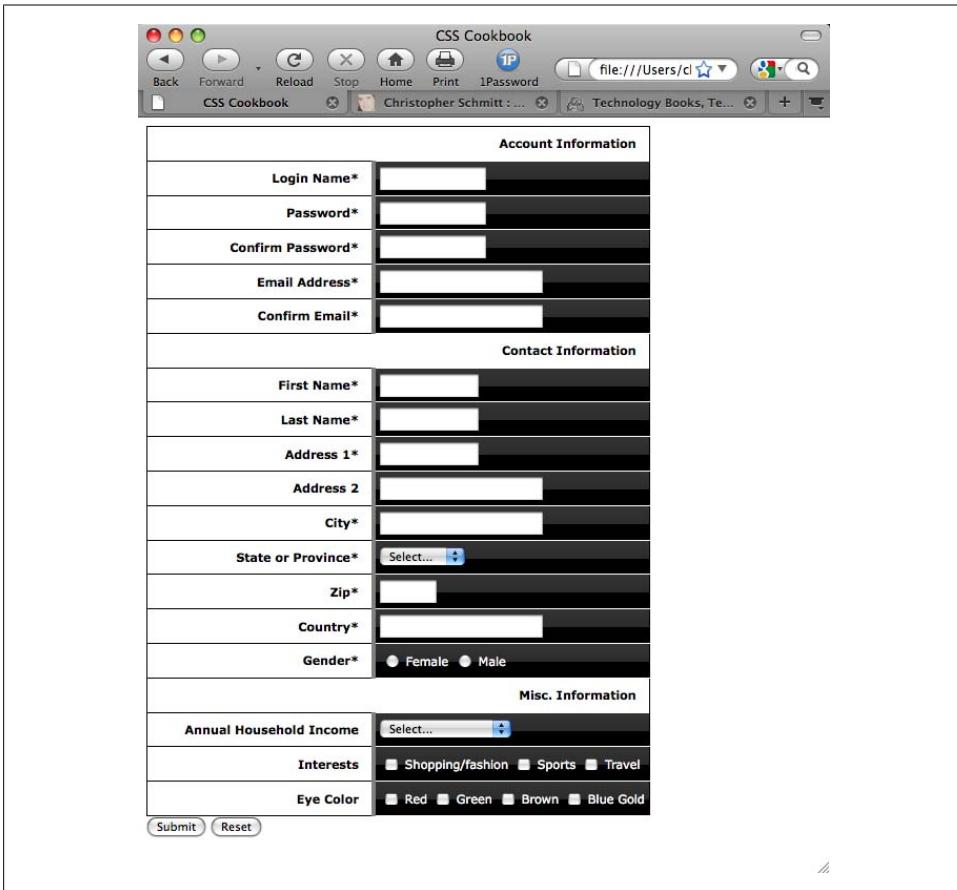

Figure 8-36. Bevel backgrounds

Certain fields are required to execute the registration, so change the color of the text labels for those fields. This change in color will indicate at a glance which fields are required (see Figure 8-37):

```
.required {
 color: red;
}
```
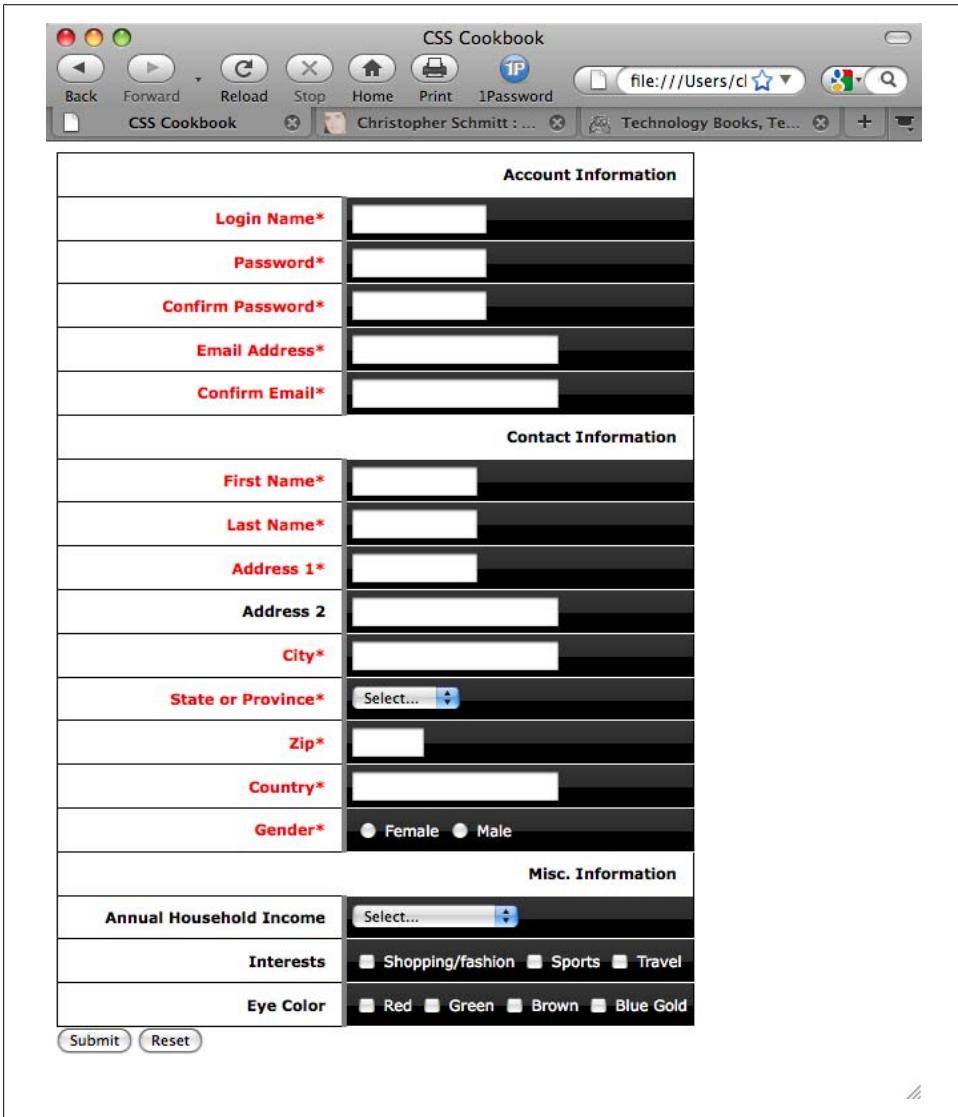

Figure 8-37. The required fields marked with red text

Note that the CSS rule states that the color is red, but for printing purposes the color will come out a shade of gray.

Adjust the form headers that indicate the different sections of the form by making the text uppercase and slightly larger than the other text in the form (see Figure 8-38):

```
.header th {
text-align: left;
text-transform: uppercase;
font-size: .9em;
}
```
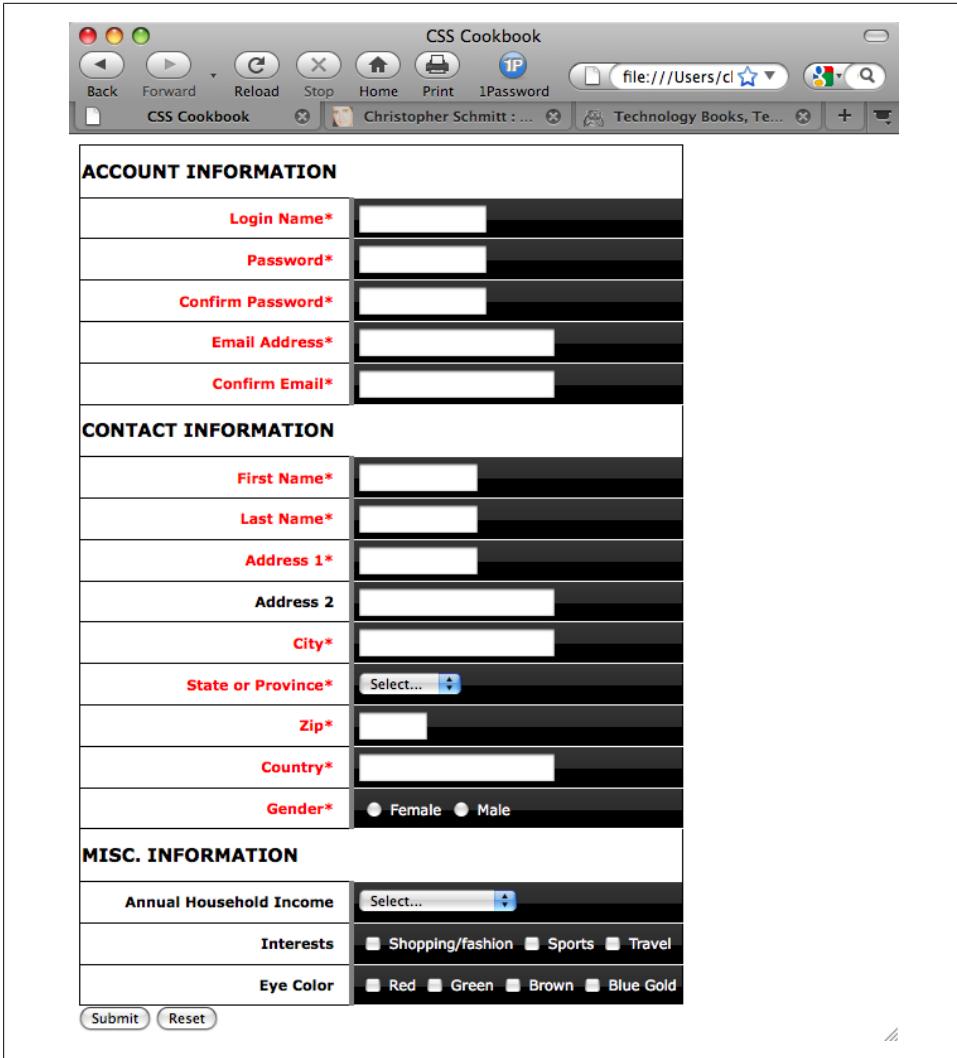

Figure 8-38. The refined form section headers

Slide the form headers so that they rest on top of the second column. To determine where to place the headers, add the size of the left column (200 pixels), the padding of the right column (4 pixels), the width of the border on the left of the right column (4 pixels), and the padding of the right column (12 pixels):

```
.header th {
  text-align: left;
  text-transform: uppercase;
  font-size: .9em;
  padding-left: 220px;
}
```

Then add a touch of visual appeal by applying thicker borders to the top and bottom of the header (see Figure 8-39):

```
.header th {
  text-align: left;
  text-transform: uppercase;
  font-size: .9em;
  padding-left: 220px;
  border-bottom: 2px solid gray;
  border-top: 2px solid black;
}
```
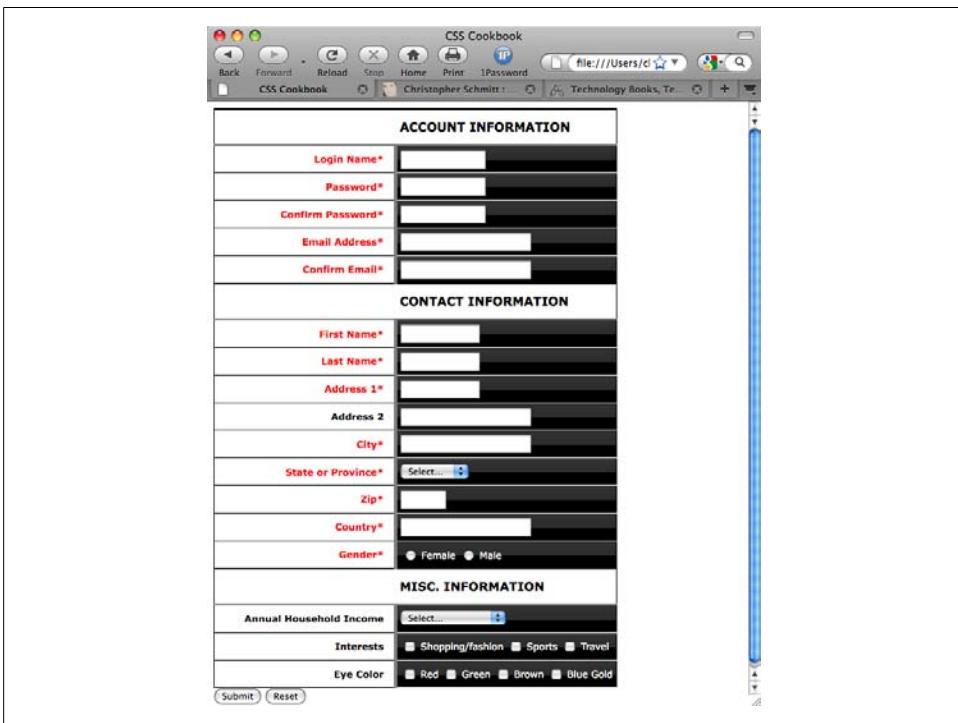

Figure 8-39. Padding added to the section headers

For the finishing touch, move the Submit and Reset buttons so that they fall under the form fields, just like the section headings, by assigning the left side of the margin to be 220 pixels (see Figure 8-40):

```
#buttonSubmit {
  margin-left: 220px;
  margin-top: 4px;
}
```
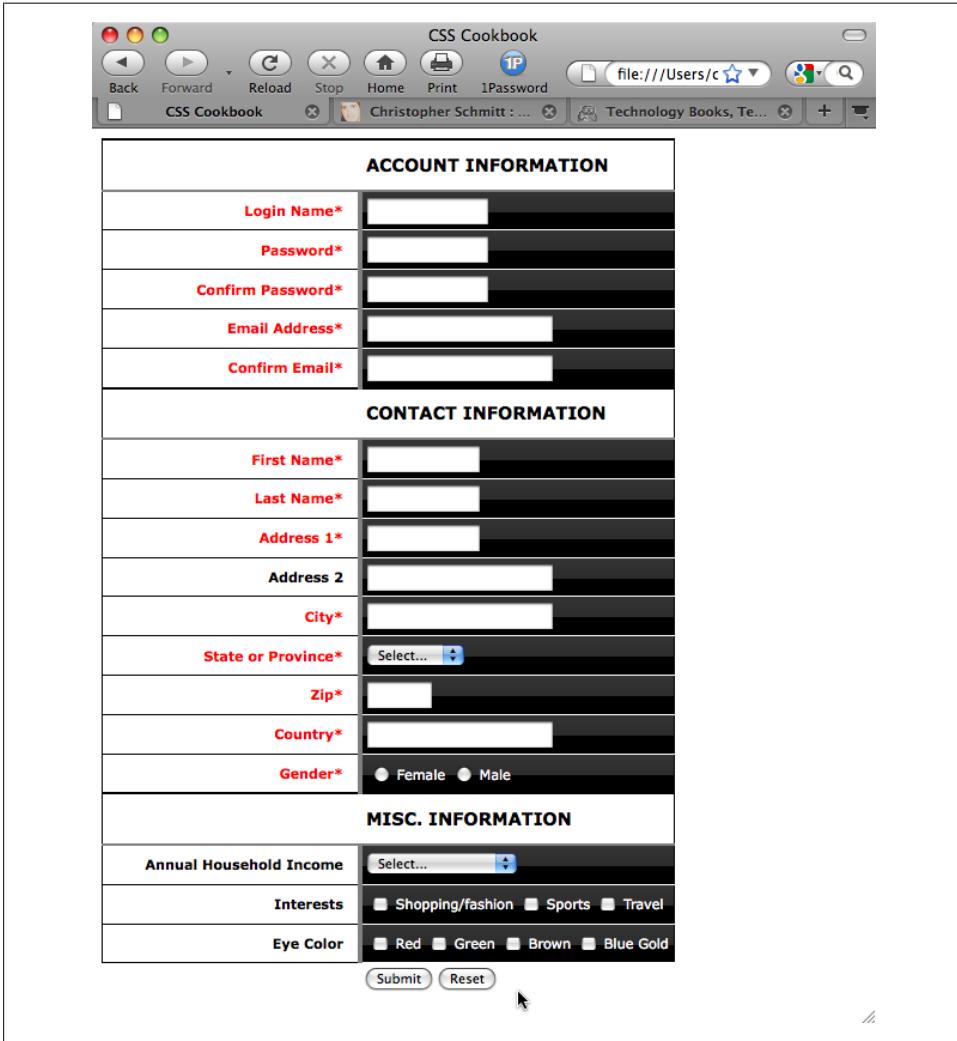

Figure 8-40. The Submit and Reset buttons moved into place

Tables

9.0 Introduction

With CSS, web designers learned they could forego the practice of manipulating HTML tables to hold designs together. Practices such as cutting up an image to place the image "pieces" into separate table cells or nesting tables for web page layouts have now become outmoded. However, the use of tables still has its place. Web developers use HTML tables to present tabular data, such as a calendar or scientific data, and therefore can use CSS to stylize those tables.

This chapter teaches you how to make your tables look better by stylizing table headers, setting borders for a table and its cells, and reducing gaps between images in table cells. The sample design at the end of the chapter takes you through the steps required to stylize a calendar.

9.1 Setting the Borders and Cell Padding for Tables

Problem

You want to set the borders and the amount of space within table cells to create a stronger visual display than the default rendering of a table, as shown in Figure 9-1, for example.
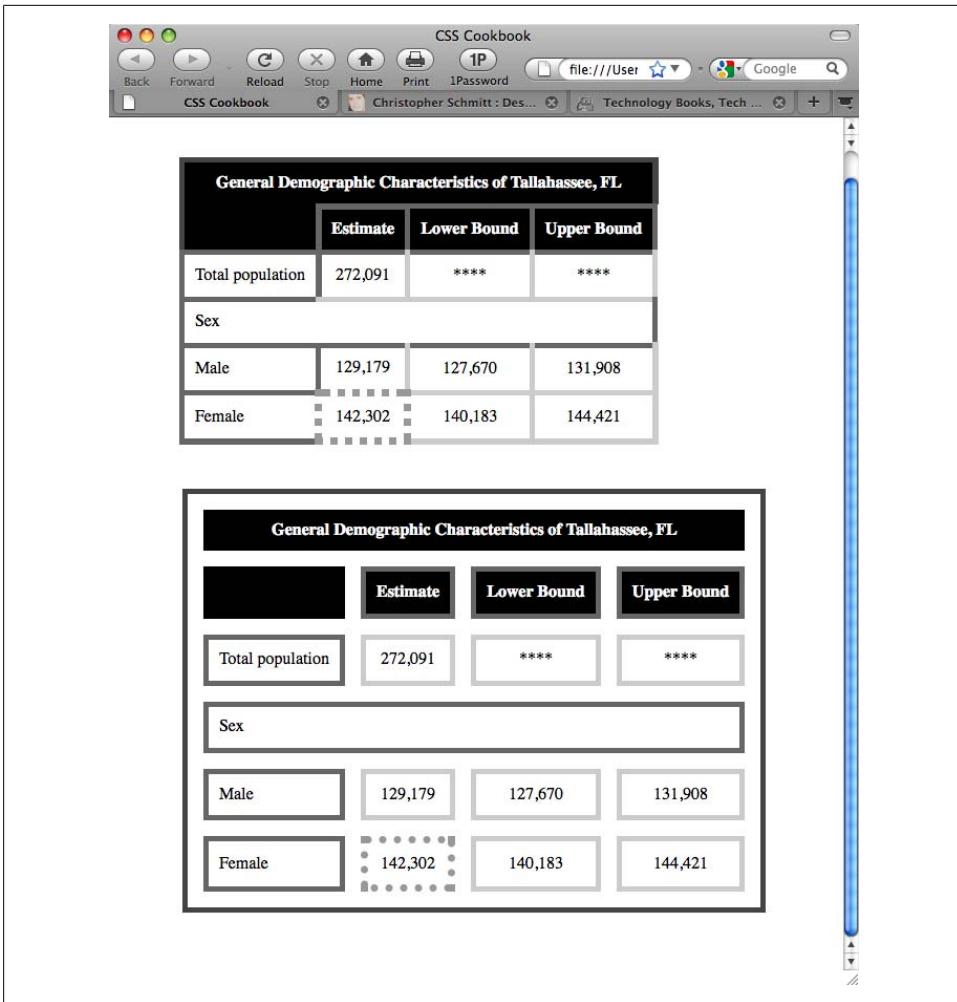
Figure 9-1. Borders and padding applied to the table and table cells

Solution

Use the `padding` property to address the amount of space between the content in the cell and the edges of the cell. Use the `border` property to set the borders on both the table and its cells:

```
table {
 border-collapse: collapse;
 border: 5px solid #444;
}
td {
 padding: 10px;
}
```

```
th {
  padding: 10px;
  color: white;
  background-color: black;
}
td, th+th {
  border: 5px solid #666;
}
td+td {
  border: 5px solid #ccc;
  text-align: center;
}
td#winner {
  border: 7px dotted #999;
}
```

Discussion

There are two border models for HTML tables: *collapse* and *separate*. With the collapse model, table cells share borders and appear to be placed on top of each other, as shown in the first table in Figure 9-1.

The second, bottom table in Figure 9-1 showcases the separate model, in which the table cells are detached with their own distinct borders.

Browser issues

At the time of this writing, the collapse model is more widely implemented by browsers, and thus is used more frequently by designers, than the separate model.

All browsers today default to the collapse model, except for Firefox, which defaults to the separate model. As the CSS standard doesn't specify that behavior, you should explicitly set the collapse model in your stylesheets in case a future browser does not have the same defaults. Set the border model by using the `border-collapse` property set to `collapse`:

```
table {
  border-collapse: collapse;
}
```

Defining border cells

The `table` element's `border` attribute determines borders for the table and its enclosing cells. You can set the CSS `border` property through a separate border thickness for the table and individual cells.

When applying a border to a cell that runs counter to a previous CSS rule, the following four CSS specification rules are followed for conflict resolution:

- If `border-style` is set to `hidden`, all other border styles are concealed.
- If `border-style` is set to `none`, any other border style wins.

- Unless a cell has `border-style` set to `hidden` or `none`, a thicker border overrides the narrower borders. If adjoining cells have the same width, the style of the border will be determined in the following order: `double`, `solid`, `dashed`, `dotted`, `ridge`, `outset`, `groove`, `inset`.
- If adjoining cells have a different color while possessing the same style and width, the border color will be determined in the following order: cell, row, row group, column, column group, and then table.

With the separate border model, every cell contains its own borders and can be styled independently of other cell borders. Within the separate model, the `border-spacing` property is used to set the horizontal and vertical space, respectively, between cells:

```
table#runoffdata {
 border-collapse: separate;
 border-spacing: 4px 4px;
}
```

If the `border-collapse` property is set to `separate`, any styles set for rows, columns, or groups of table cells aren't applied. Also, styles for table cells that don't contain content can be displayed or hidden using the `empty-cells` property with a value of `show` or `hide`, respectively.

Although the separate border model gives more control to web developers, as of this writing separate is supported only in Firefox, Safari, and Chrome, and not in IE. Therefore, most web designers stick to the collapse model.

See Also

The CSS 2.1 specification for border models at *http://www.w3.org/TR/CSS21/tables .html#propdef-border-collapse*; Chapter 11 of *CSS: The Definitive Guide* by Eric A. Meyer (O'Reilly) for more discussion on tables

9.2 Setting the Cell Spacing

Problem

You want to adjust the space between the table border and cell borders.

Solution

Use the `cellspacing` table attribute:

```
<table cellspacing="15">
 <tr>
   <th colspan="2">
    General Demographic Characteristics of Tallahassee, FL
   </th>
 </tr>
 <tr>
```

```
  <th>
  </th>
  <th>
   Estimate
  </th>
 </tr>
 <tr>
  <td>
   Total population
  </td>
  <td>
   272,091
  </td>
 </tr>
</table>
```

Discussion

The CSS 2.1 specification describes a standard mechanism to manipulate the `cellspacing` table attribute through the use of the `border-spacing` property when the `border-collapse` value is set to `separate`:

```
border-collapse: separate;
border-spacing: 15px;
```

However, implementation of this part of the specification isn't available in Internet Explorer 6 for Windows.

Using the `cellspacing` HTML attribute is the recommended solution that works in modern browsers and is backward compatible.

See Also

Recipe 9.1 for setting table borders and cell padding; the CSS 2.1 specification for `border-collapse` at *http://www.w3.org/TR/CSS21/tables.html#propdef-border-collapse*; the CSS 2.1 specification for `border-spacing` at *http://www.w3.org/TR/CSS21/tables.html#propdef-border-spacing*; Recipe 8.2; Recipe 8.8

9.3 Setting the Style for Captions

Problem

You want to set the style for the description of a table.

Solution

Use the `caption` element selector to stylize the caption:

```
table caption {
 font: 1.5em Georgia, "Times New Roman", Times, serif;
```

```
  padding: 1em;
  }
```

Discussion

Captions are used to describe the contents within a table and should be placed after the opening `table` element and before another `table` element such as `thead` or `tr`:

```
<table id="shoppingcartTable" summary="List of products
in your shopping cart.">
 <caption>Shopping Cart Listing - <strong>Subtotal: $45.16</strong>;
changed quantities? <input type="submit"  value="Update price(s)" /></caption>
 ...
</table>
```

Browsers may vary in how to render the `caption` element. However, `caption` will always be displayed by any browser, can be styled via CSS, and is the most accessible method of displaying a table caption.

See Also

The HTML 4.01 specification for `caption` at *http://www.w3.org/TR/html4/struct/tables .html#h-11.2.2*

9.4 Setting the Styles Within Table Cells

Problem

You want to stylize links within a table cell to make them appear visually different from the rest of the page.

Solution

Use a descendant selector (sometimes referred to as a *contextual selector*) to manipulate the styles for content in a table cell:

```
td a {
  display: block;
  background-color: #333;
  color: white;
  text-decoration: none;
  padding: 4px;
}
```

Discussion

By using the type and descendant selectors—the `td a` in the CSS rule—to apply the styles, you reduce the amount of markup needed to perfect your designs and you reduce the document's file size. The style affects only the `a` elements within the table cells (marked by `td` tags).

If you need more control over the design of the content within a table cell, use a class selector:

```
<td class="navText">
 <a href="/">Home</a>
</td>
```

You then can apply the CSS rules to the cell's content through a combination of class and descendant selectors:

```
td.navText a {
 font-size: x-small;
}
```

If you want to stylize content within a table cell that contains more content or markup than just a link, use a `div` element to wrap around the content and then use a class selector through that very same `div` element.

In the following example, an unordered list is enclosed within a `div` element set with a `class` attribute:

```
<td>
 <div class="tblcontent">
 <p>To-do list on your day off.</p>
 <ul>
   <li><a href="http://www.imdb.com/title/tt0120737">Watch <cite>Fellowship of
the Rings</cite>, Extended Version</a></li>
   <li><a href="http://www.imdb.com/title/tt0167261/">Watch
<cite>Two Towers</cite>, Extended Version</a></li>
   <li><a href="http://www.imdb.com/title/tt0167260/">Watch <cite>Return of the
King</cite>, Extended Version</a></li>
  <li>Start or join local Elvish society.</li>
 </ul>
</div>
</td>
```

The CSS rules to stylize the content within the table cell could look like this:

```
.tblcontent p {
 margin: 0;
 padding: 0;
 font-weight: bold;
}
.tblcontent ul {
 margin: 0;
 padding: 0;
}
.tblcontent li {
 margin: 0;
 padding: 0;
 line-height: 1.5;
}
.tblcontent li a {
 padding-left: 15px;
 background-image: url(bullet.gif);
```

```
    background-repeat: no-repeat;
}
```

See Also

The CSS 2.1 specification regarding type selectors at *http://www.w3.org/TR/CSS21/se lector.html#type-selectors*; *http://www.w3.org/TR/CSS21/selector.html#descendant-se lectors* for information about descendant selectors

9.5 Setting the Styles for Table Header Elements

Problem

You want to differentiate the style of the table headers from the content in regular table cells; Figure 9-2 shows a table with traditional table headers, and Figure 9-3 shows a stylized version of the same table.

Solution

Use the th element selector to stylize the table header:

```
th {
 text-align: left;
 padding: 1em 1.5em 1em 0.5em;
 color: white;
 border-right: 1px solid rgba(0, 204, 255, .8);
 border-bottom: 1px solid rgba(0, 204, 255, .8);;
 text-shadow: 0 1px 0 rgba(0, 0, 0, .8);
 background: blue url(title-glass.png) repeat-x 50%;
 font: .9em Arial, Helvetica, Verdana, sans-serif;
}
```
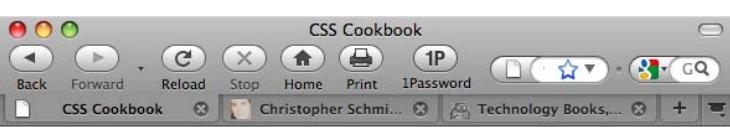

Figure 9-2. The table as it appears before styles are applied to the table headers
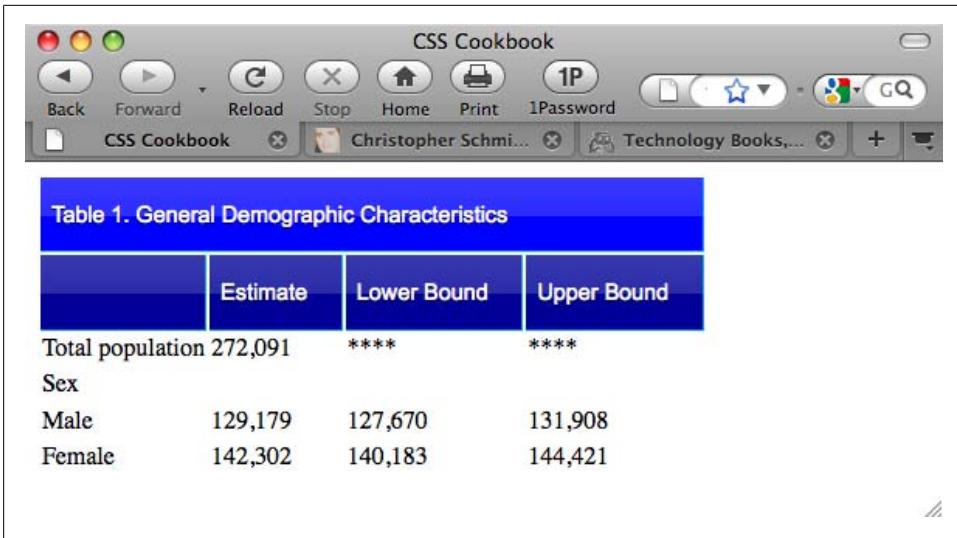
Figure 9-3. Styles applied to the table headers

For tables with multiple rows of `th` elements that require different styles, use a class selector to differentiate the rows:

```
.secondrow th {
/* Use a lighter shade of blue in the background */
 background-color: #009;
 }
```

Put the appropriate rows into that class:

```
<tr>
 <th colspan="4">
 Table 1. General Demographic Characteristics
 </th>
</tr>
<tr class="secondrow">
 <th>

 </th>
 <th>
  Estimate
 </th>
 <th>
  Lower Bound
 </th>
 <th>
  Upper Bound
 </th>
</tr>
```

Discussion

The th element characterizes the contents of the cell as header information. When setting the styles for the element, use styles that make the cell stand out from the other content in the table cells. It's a header for a table that should act like a heading would for an article.

You can generate contrasting styles by simply adjusting any of the following properties: `font-family`, `background-color`, `font-size`, `font-weight`, and `text-align`. (See Recipe 3.1 for information on specifying fonts and Recipe 3.6 for information on setting font measurements and sizes.) Regardless of what you adjust, chances are you will be improving the look of the table headers.

> The nonbreaking space characters placed in the table headers are used so that the browser treats the heading as one word, and therefore doesn't force a break in the heading.

See Also

The CSS 2.1 specification for type selectors at *http://www.w3.org/TR/CSS21/selector .html#type-selectors*

9.6 Removing Gaps from Images Placed in Table Cells

Problem

You want to get rid of space in a table cell that contains only an image. In other words, you want to go from Figure 9-4 to 9-5.

Solution

Set the image to be displayed as a block-level element:

```
td img {
 display: block;
}
```

Discussion

The browser puts the image on the baseline used for text content since it is being placed as an inline element. Therefore, set the element as a block-level element to force the browser to render the image differently. This baseline isn't at the bottom of the cell, because some letters (e.g., *g*, *p*, *q*, and *y*) have descenders that hang below that baseline (see Figure 9-6).
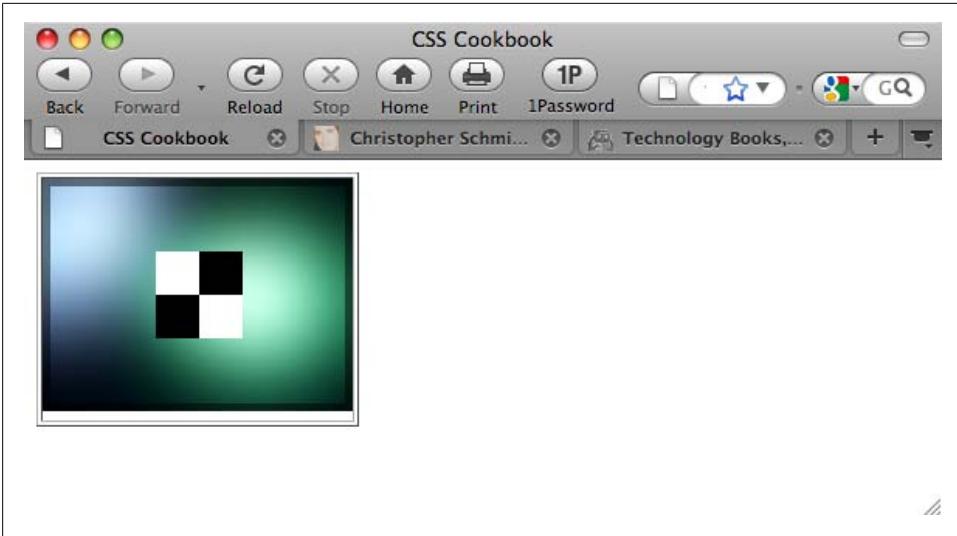
Figure 9-4. A gap appearing below an image in a table cell
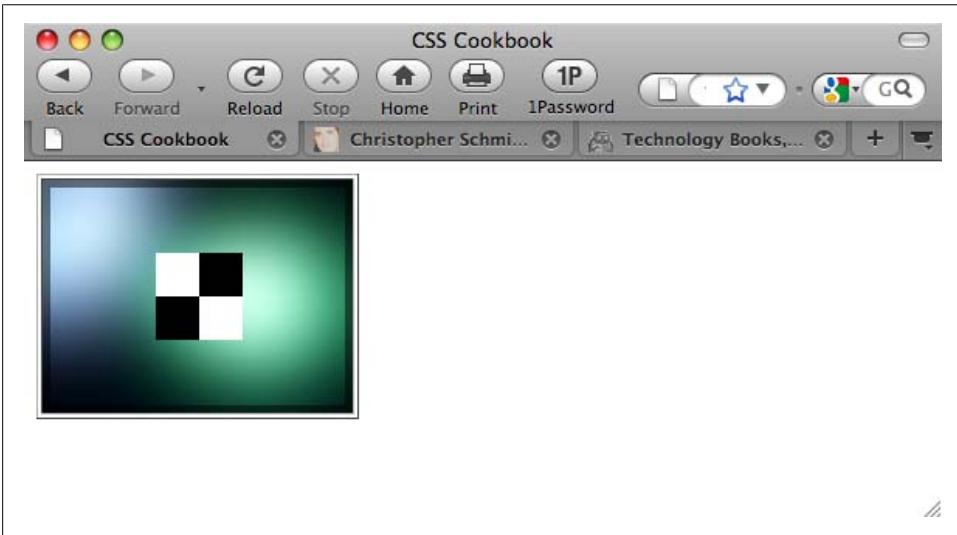
Figure 9-5. Displaying an image in a table cell as a block-level element

Because the baseline is a percentage of the total font size, you can't simply remove the descender space. By instructing the browser to handle the image differently, you can avoid automatic creation of the descender whitespace altogether. Thus, set the `display` property for the image to `block`, as shown in the Solution.
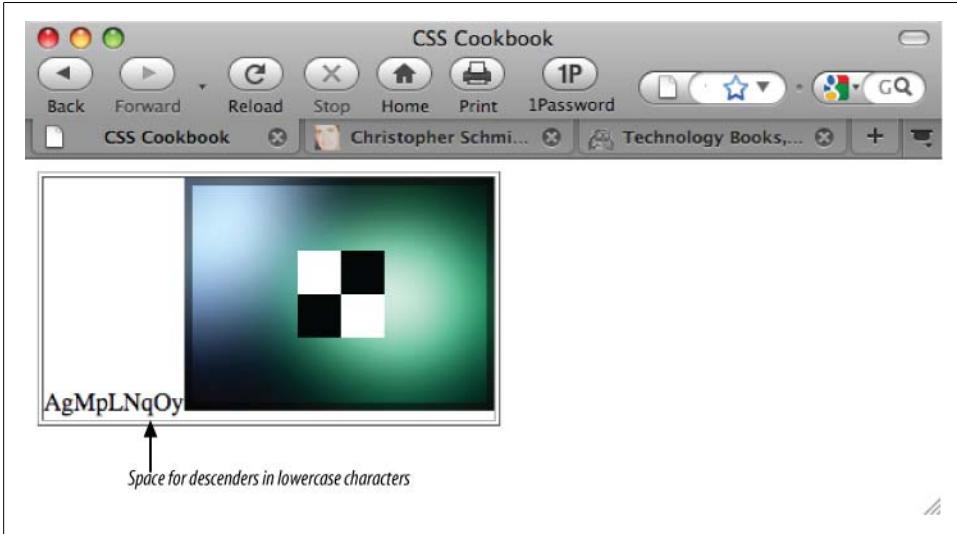
Figure 9-6. The descenders of the lowercase letters g, p, q, and y, which highlight the whitespace below the image

Vertical alignment

Sometimes, setting the image's `display` to `block` isn't the best solution for removing whitespace around an image in a table cell. In such cases, you can remove the space by setting the image's `vertical-align` property to `bottom` as long as the image is taller than the line box.

See Also

The CSS 2.1 specification for `display` at *http://www.w3.org/TR/CSS21/visuren.html #propdef-display*; *https://developer.mozilla.org/en/Mozilla's_DOCTYPE_sniffing* for information on quirks mode and almost standards mode

9.7 Eliminating Gaps Between Table Cells

Problem

You want to remove gaps from one table cell to another.

Solution

Set the table to use the collapse border model:

```
#shoppingcartTable {
 border-collapse: collapse;
 width: 100%;
```

```
  border: 1px solid #666;
 }
 #shoppingcartTable th {
  background: #888 url(th_bkgd.jpg) repeat-x;
  font: italic 1.5em Georgia, "Times New Roman", Times, serif;
  padding: .5em 0 .5em 7px;
  text-align: left;
  border-top: 1px solid #666;
  border-bottom: 1px solid #666;
  text-shadow: #ccc -2px 2px -2px;
 }
```

Discussion

By setting the `border-collapse` property to `collapse`, you make the browser remove the spacing between the table cells. Therefore, when you apply a border to the table cells, the result is a clean, uninterrupted line across the table row or column.

See Also

Recipe 9.1 for more discussion about the border collapse model

9.8 Creating Alternating Background Colors in Table Rows

Problem

You want to have table rows with alternating background colors, as shown in Figure 9-7.

Solution

Create a class selector specifically designed for odd-numbered table rows:

```
 tr {
  background-color: #eee;
 }
 tr.odd {
  background-color: #ccc;
 }
```

Then append every other table row with a `class` attribute with `odd` set as its value, as shown in Figure 9-8:

```
 <tr>
  <td class="dltprod">
   <p>Item added on March 22, 2006.</p>
   <a href="" title="Delete this product"><img src="x.gif" alt="delete"
 class="dltitem" /></a>
  </td>
  <td class="prodcell">
   <img class="prod" alt="product image" src="u2-dismantle.jpg" />
```
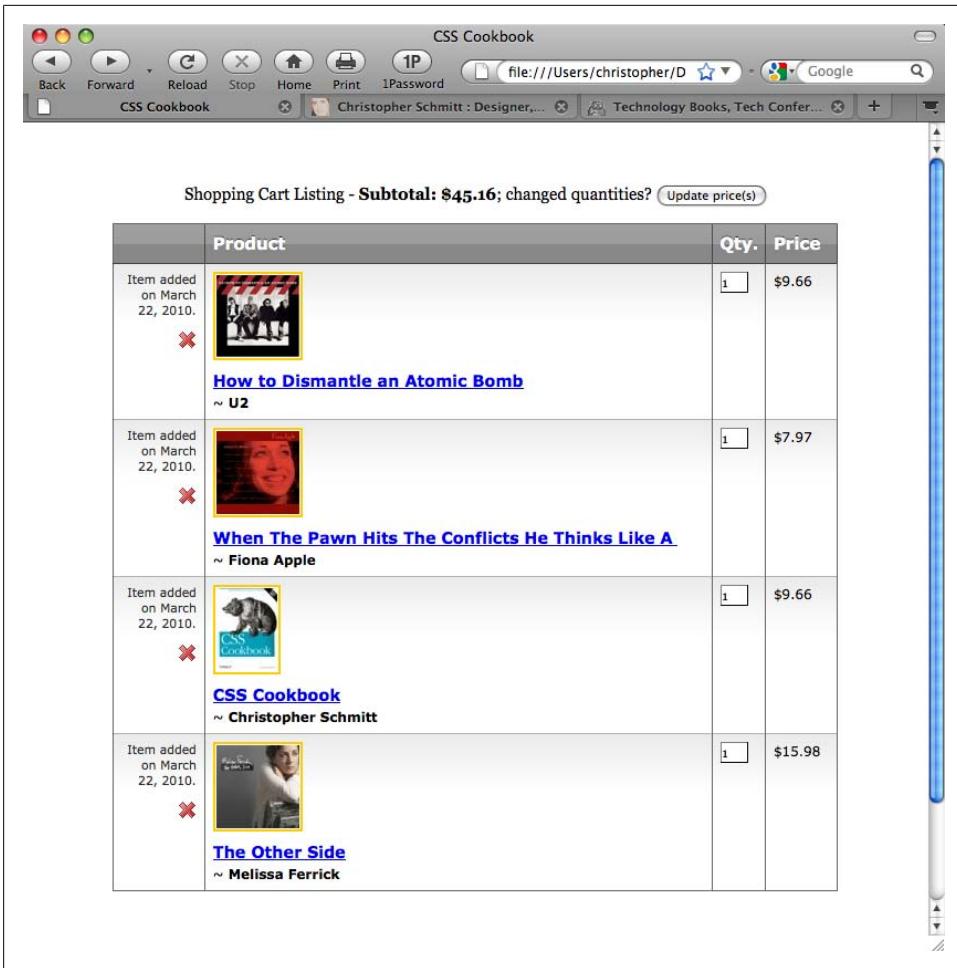
Figure 9-7. A table without any color in the background cells

```
    <div class="prodtitle"><a href="/product.php?id=B0006399FS">How
to Dismantle an Atomic Bomb</a></div>
     ~ <strong>U2</strong>
   </td>
   <td><input type="text" value="1" name="qty" size="2" /></td>
   <td>$9.66</td>
 </tr>
 <tr class="odd">
  <td class="dltprod">
   <p>Item added on March 22, 2006.</p>
   <a href="" title="Delete this product"><img src="x.gif" alt="delete"
class="dltitem" /></a>
  </td>
  <td class="prodcell">
   <img class="prod" alt="product image" src="apple-whenthepawn.jpg" />
```

```
   <div class="prodtitle"><a href="/product.php?id=B00002MZ4W">When The Pawn
Hits...</a></div>
    ~ <strong>Fiona Apple</strong>
  </td>
  <td><input type="text" value="1" name="qty" size="2" /></td>
  <td>$7.97</td>
</tr>
```
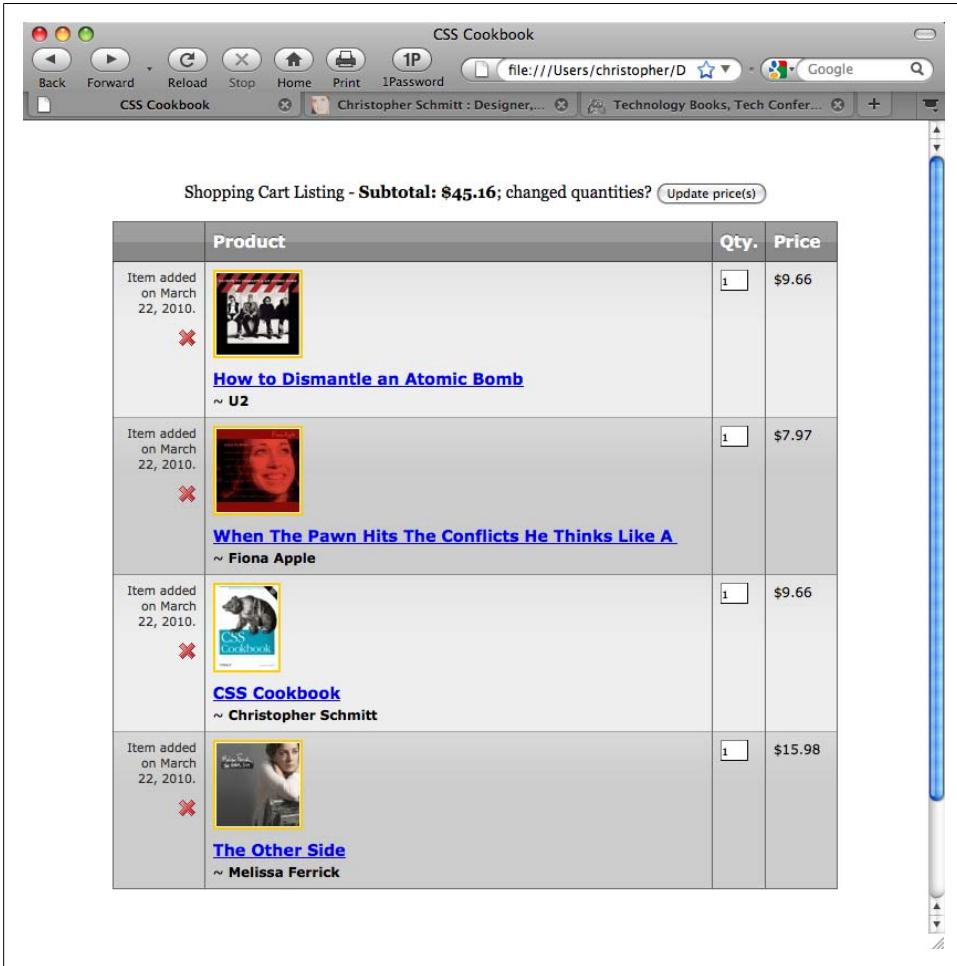

Figure 9-8. Alternating colors in the table rows

Discussion

This solution of marking up every other tr element—although laborious for long tables if handcoded—ensures cross-browser compatibility.

A second solution helps eliminate the need for extra markup within HTML tables. Using the CSS3 `:nth-child` selector, the solution is straightforward:

```
tr {
 background-color: #eee;
}
tr:nth-child(odd) {
 background-color: #ccc;
}
```

However, support for `:nth-child` is limited to Safari 3 and later, Firefox 3.5 and later, and Opera 9.5 and later.

Alternative solutions

Other solutions go beyond just CSS. One solution is to use JavaScript that interacts with the Document Object Model (DOM) and automatically applies the styles to every other table row. You can find such a solution in Recipe 14.6. The downside to this solution is that it will fail if the user has disabled JavaScript in her browser.

Another programming solution is to use a server-side programming language such as PHP or ColdFusion to write a simple script that automates the generation of the table. (This technique is also beneficial if a backend database is being used to create and maintain the tabular data.)

See Also

The CSS3 specification for the `:nth-child` pseudo-class selector at *http://www.w3.org/ TR/css3-selectors/#nth-child-pseudo*

9.9 Adding a Highlighting Effect on a Table Row

Problem

You want to highlight a whole row within a table when the cursor moves over a table cell within that table row.

Solution

Use the `:hover` pseudo-class on the `tr` element, as shown in Figure 9-9:

```
tr:hover {
 background: yellow;
}
```
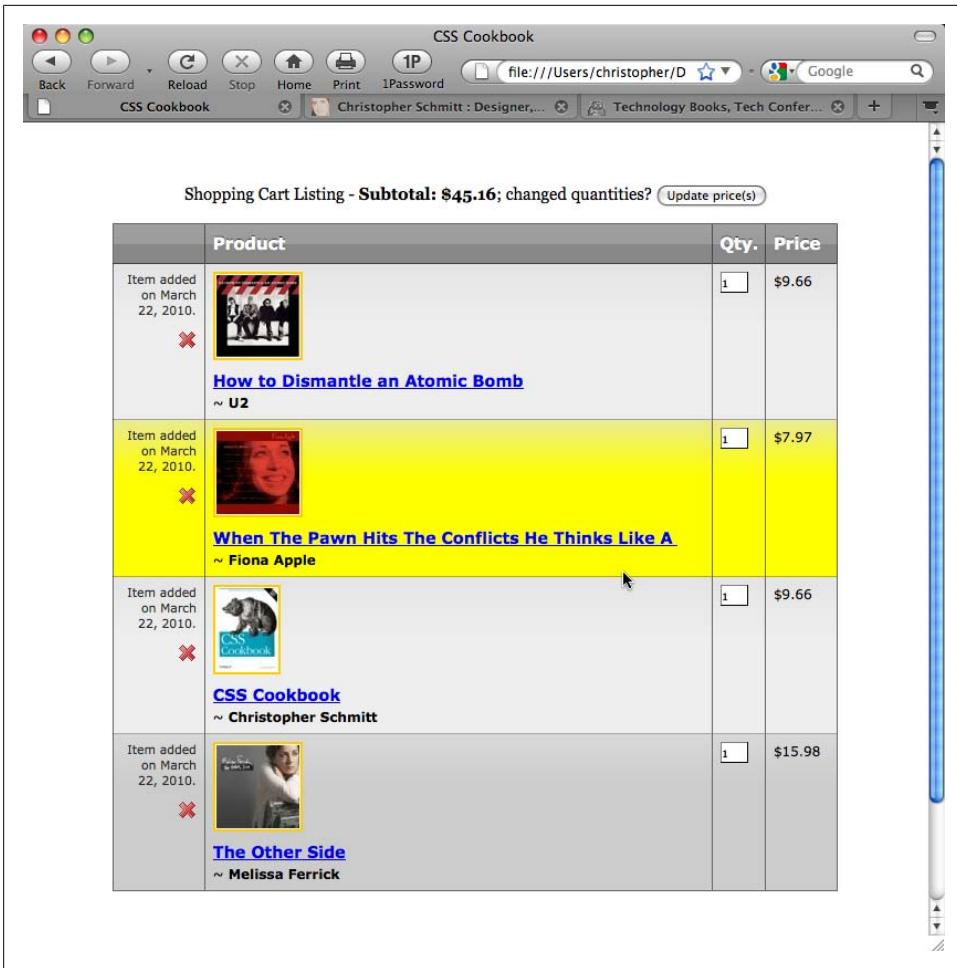
Figure 9-9. A table row highlighted as the cursor moves across the table

Discussion

The :hover pseudo-class is commonly seen on links to create rollover effects. However, the CSS specification doesn't limit its use to just links. You can also apply it to other elements, such as p and div.

Modern browsers support this solution. However, IE7 and earlier versions do not create a hover effect on an element other than a link.

See Also

The CSS 2.1 specification for dynamic pseudo-classes at *http://www.w3.org/TR/CSS21/ selector.html#dynamic-pseudo-classes*

9.10 Sample Design: An Elegant Calendar

Great for organization, calendars enable us to schedule lunches, remember birthdays, and plan honeymoons. As designers, we can think of all those months, dates, and appointments as tabular data.

If you display your calendar as a generic HTML table, chances are the table looks rather plain, and if it contains numerous events it probably looks somewhat convoluted as well. In this design, we will use CSS to create a calendar that is more legible than what you could create using vanilla HTML.

First, take a look at Figure 9-10, which shows the markup for the calendar without styles.

Figure 9-10. The calendar without styles

Next, look at the markup itself to see how it's set up. As you learned in Recipe 9.2, you need to set the `cellspacing` attribute in the `table` element:

```
<table cellspacing="0">
```

Now, set the first three rows of the table headers (`th`) containing the year, month, and days in their own rows within their own table headers:

```
<tr>
  <th colspan="7" id="year">
    <a href="year.html?previous">&lt;</a> 2010 <a
href="year.html?next">&gt;</a>
```

```
     </th>
    </tr>
    <tr>
     <th colspan="7" id="month">
      <a href="month.html?previous">&lt;</a> October <a
href="month.html?next">&gt;</a>
     </th>
    </tr>
    <tr id="days">
     <th>Sunday</th>
     <th>Monday</th>
     <th>Tuesday</th>
     <th>Wednesday</th>
     <th>Thursday</th>
     <th>Friday </th>
     <th>Saturday</th>
    </tr>
```

The first date is October 1, which in this calendar falls on a Sunday. To signify that Sundays and Saturdays are days of the weekend, use a `class` selector in the `td` element.

In each date of the month there is a link on the date itself (which would, in theory, take the user to a detailed listing of the day) as well as a link to add more events to the day. Wrap these two links in a `div` element so that when new events are added there is a clear division between the two sections in the table cell:

```
    <tr>
    <td class="weekend">
     <div>
      <a href="1.html" class="date">1</a>
      <a href="add.html" class="addevent">+</a>
     </div>
    </td>
```

The next date, October 2, has an event listed. The event is marked up as a link and is placed below the `div` containing the date and the `addevent` links (because October 2 is a weekday, the `weekend` class isn't applied to the `td` element):

```
    <td>
     <div>
      <a href="2.html" class="date">2</a>
      <a href="add.html" class="addevent">+</a>
     </div>
     <a href="16.html?id=1" class="event">Regular City
    Commission meeting agenda</a>
    </td>
```

The rest of the markup follows a similar structure:

```
    <td>
     <div>
      <a href="3.html" class="date">3</a>
      <a href="add.html" class="addevent">+</a>
     </div>
    </td>
    <td>
```

```
      <div>
        <a href="4.html" class="date">4</a>
        <a href="add.html" class="addevent">+</a>
      </div>
    </td>
    <td>
     <div>
      <a href="5.html" class="date">5</a>
      <a href="add.html" class="addevent">+</a>
     </div>
     <a href="5.html?id=1" class="event">Dad's birthday</a>
    </td>
    <td>
      <div>
       <a href="6.html" class="date">6</a>
       <a href="add.html" class="addevent">+</a>
      </div>
    </td>
    <td class="weekend">
      <div>
       <a href="7.html" class="date">7</a>
       <a href="add.html" class="addevent">+</a>
      </div>
      <a href="7.html?id=1" class="event">FSU at UM</a>
    </td>
  </tr>

  [...]

  <tr>
   <td class="weekend">
     <div>
      <a href="29.html" class="date">29</a>
      <a href="add.html" class="addevent">+</a>
     </div>
     <div class="event">Buy candy</div>
   </td>
   <td>
     <div>
      <a href="30.html" class="date">30</a>
      <a href="add.html" class="addevent">+</a>
     </div>
     <a href="16.html?id=1" class="event">Regular City
Commission meeting agenda</a>
   </td>
   <td>
     <div>
      <a href="31.html" class="date">31</a>
      <a href="add.html" class="addevent">+</a>
     </div>
     <a href="31.html?id=1" class="event">Halloween</a>
     <a href="31.html?id=2" class="event">Flu shot</a>
   </td>
   <td>
    <div class="emptydate"> </div>
```

```
    </td>
    <td>
     <div class="emptydate"> </div>
    </td>
    <td>
     <div class="emptydate"> </div>
    </td>
    <td class="weekend">
     <div class="emptydate"> </div>
    </td>
   </tr>
  </table>
```

With the calendar marked up, you can begin setting up the styles. First, set the `font-size` to `62.5%` as discussed in Recipe 3.6.

Next, apply the styles to the table and links. The `width` of the table is set to `100%` and the border model (see Recipe 9.1) is set to `collapse`, the common model web designers are used to and most browsers get right in their CSS implementations; the underline decoration is turned off (see Figure 9-11):

```
body {
 font-size: 62.5%;
}
table {
 width: 100%;
 border-collapse: collapse;
}
td a:link, td a:visited {
 text-decoration: none;
 font-family: "Gill Sans", Calibri, Trebuchet, sans-serif;
}
```
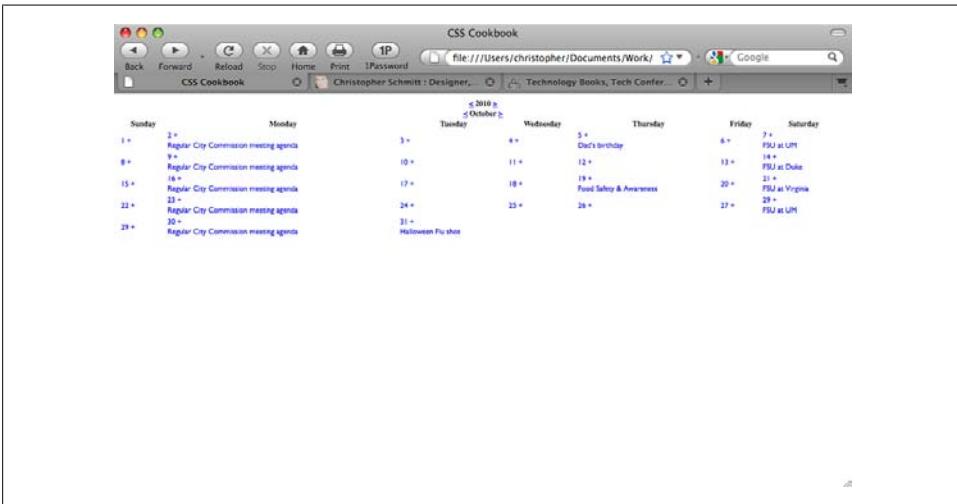

Figure 9-11. Underline decoration of the links removed

Next, set up the styles for the first three rows of the table. The rows are marked with ID selectors because you want the styles to show up only once in the document. Stylize these rows in a straightforward manner using the monospace font stack (see Recipe 3.2) for the heading font and then decreasing the font sizes, with the month sized the largest (see Figure 9-12):

```
#year {
 font-family: Consolas, "Lucida Console", Monaco, monospace;
 font-size: 3em;
 padding: 0;
 margin: 0;
}
#month {
 font-family: Consolas, "Lucida Console", Monaco, monospace;
 font-size: 2em;
 padding: 0;
 margin: 0;
}
#days {
 background: black url(title-glass.png) repeat-x 50%;
 color: white;
 font-family: "Gill Sans", Calibri, Trebuchet, sans-serif;
 width: 75px;
 text-shadow: 0px -1px 0px rgba(0, 0, 0, .8);
 border: 1px solid rgba(0,0,0,.5);
 border-top: none;
}
#days th {
 padding: 4px;
}
```
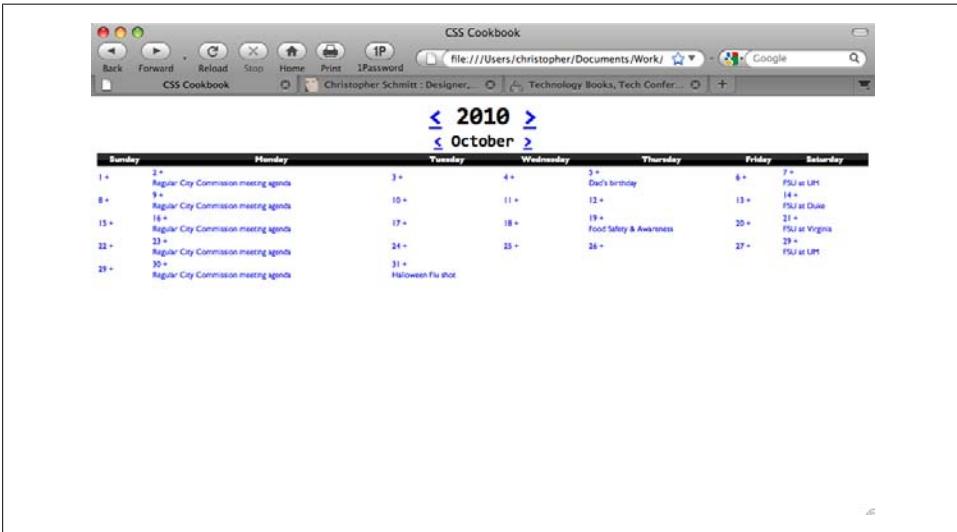

Figure 9-12. Styling the first three rows

Now it's time to stylize the dates and add event links in each cell. To reproduce the box date effect seen in most calendars, place a border to the right and bottom of the text and float the content to the left.

You want the add event links to be close to the dates. Floating the link to the right means the link will be positioned next to the date of the following day. By floating the add event link to the left, you are telling the user that the plus sign (+) means "add an event for that particular day" (see Figure 9-13):

```
.date {
  border-right: 1px solid black;
  border-bottom: 1px solid black;
  font-family: Consolas, "Lucida Console", Monaco, monospace;
  text-decoration: none;
  float: left;
  width: 1.5em;
  height: 1.5em;
  background-color: white;
  text-align: center;
}
.addevent {
  display: block;
  float: left;
  width: 1em;
  height: 1em;
  text-align: center;
  background-color: #666;
  color: white;
  font-weight: bold;
  text-decoration: none;
}
```


Figure 9-13. Styles introduced to the date and add event links

Now it's time to look at how the event listings can be stylized. Because the previous links are floated, you need to create a visible break and move the events below the date.

Setting the `clear` property to `both` achieves this visual break. The `clear` property is used to indicate which sides of an element should not be positioned next to a floated element.

In this case, you don't want the left side to run next to the date and add event links. However, just in case the design changes in the future and the dates are positioned on the opposite side, use a value of `both` instead of `left`.

Next, change the `display` of the link to `block` and place padding on the bottom (see Figure 9-14). You're making these changes to prevent multiple events in a table cell from running into each other. Also, the padding acts as a nice visual buffer, allowing the eye to easily discern between two events:

```
.event {
 clear: both;
 padding-left: 1em;
 padding-bottom: .75em;
 display: block;
}
```
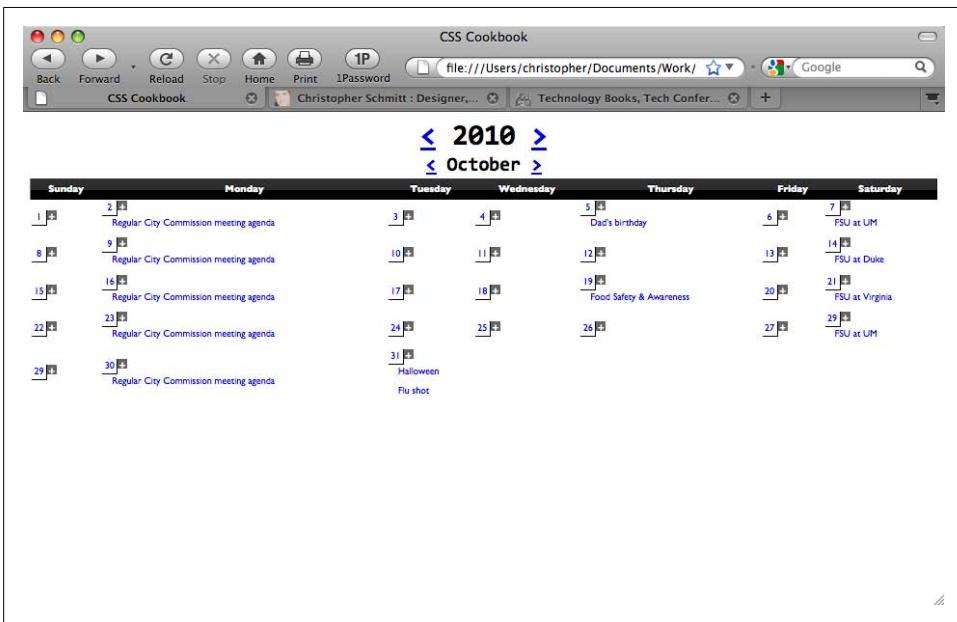

Figure 9-14. Event links treated like block-level elements

To each table cell, apply a `width` of `14%`. You're using 14% because 7 (representing the seven sections of the calendar, or days of the week) goes into 100 (representing 100% of the viewport) approximately 14 times. Also, place a white border on all sides of the cell and position all the content to the top with the `vertical-align` property (see Figure 9-15).

```
td {
  width: 14%;
  background-color: #ccc;
  border: 1px solid white;
  vertical-align: top;
  font-size: 1.2em;
  padding: 1px;
  background: url(content-bkgd.png) repeat-x;
  border: 1px solid rgba(0,0,0,.5);
  border-top: none;
}
```
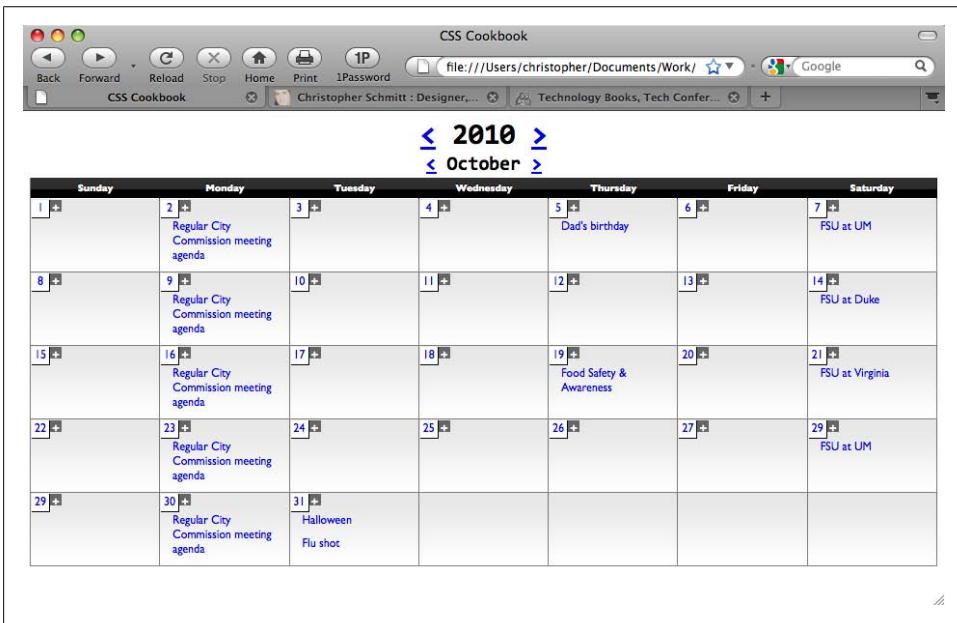

Figure 9-15. The content in each cell moved to the top

Make the background color of the weekend dates darker than that used for the weekday dates (see Figure 9-16):

```
.weekend {
  background-color: #999;
}
```
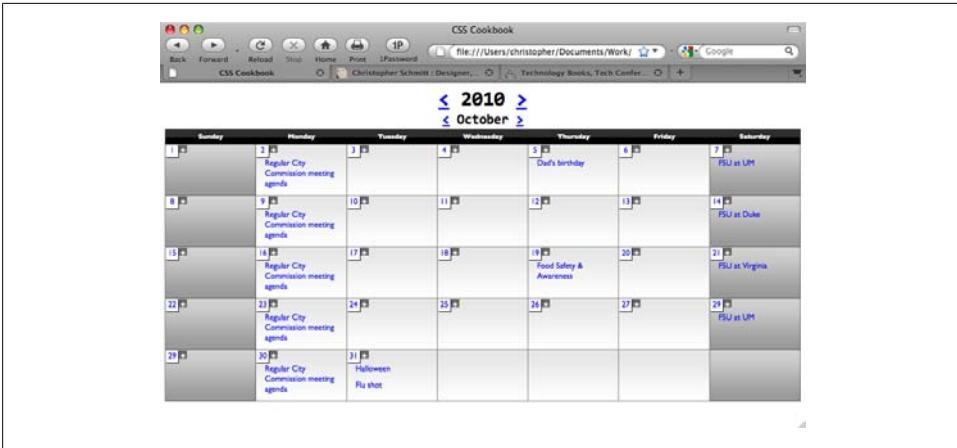
Figure 9-16. The weekend days marked with a darker gray background color

Slightly gray-out the look of the remaining days in the calendar (see Figure 9-17):

```
.emptydate {
 border-right: 1px solid #666;
 border-bottom: 1px solid #666;
 font-family: monospace;
 text-decoration: none;
 float: left;
 width: 1.5em;
 height: 1.5em;
 background-color: #ccc;
 text-align: center;
}
```


Figure 9-17. Empty dates for the next month stylized

For the current day (in this example the current day is the 27th), place a 2-pixel black border around the box:

```
#today {
  border: 2px solid black;
}
```

And with that, the calendar is complete, as shown in Figure 9-18.
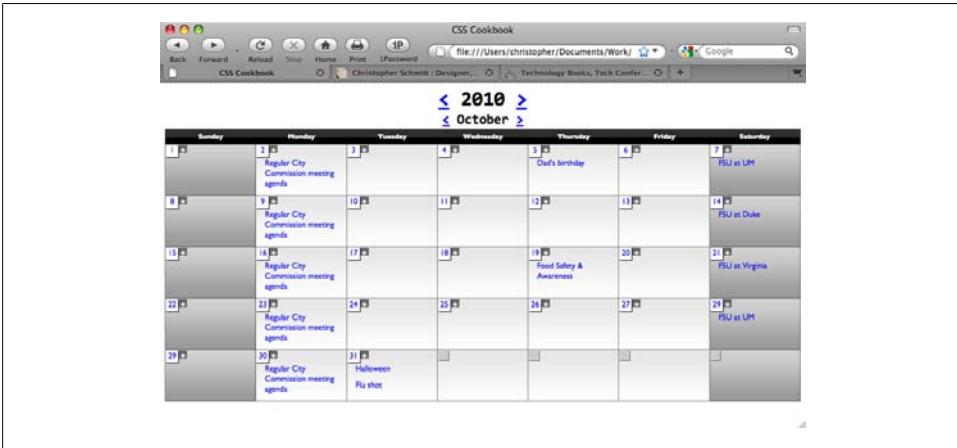
Figure 9-18. The current date in the calendar with a darker border

Designing Web Pages for Printing

10.0 Introduction

To create a printer-friendly version of a web page, web developers traditionally would either manually convert the web page content to a separate stripped-down page design or use a script to dynamically generate a separate page design.

With CSS, however, you can automatically apply a new stylesheet to web documents as they are printed, thereby eliminating the time and server resources needed to create a printer-friendly page.

Support for print-media CSS is fairly commonplace these days. All of the major modern browsers support this aspect of the technology, including Firefox, Internet Explorer for Windows, Safari, Chrome, and Opera.

This chapter teaches the basics of how to tell the browser which stylesheet to use when sending a document to print. It also discusses how to switch graphics from web to print CSS, as well as a series of techniques for developing a document for printing.

> Because this book focuses on the practical, cross-browser nature of CSS, the recipes in this chapter are geared toward styling the contents of the page rather than dealing with the theory of CSS printing properties. For more information on CSS printing properties, see *CSS: The Definitive Guide* by Eric A. Meyer (O'Reilly).

10.1 Applying a Stylesheet for Printing to a Web Page

Problem

You want to create a printer-friendly page without having to create a separate HTML file.

Solution

First, create a separate stylesheet containing the CSS rules for printing. For this example, the stylesheet with print-only CSS rules is named *print.css*.

Then, associate the stylesheet and set the `media` property to `print`:

```
<link rel="stylesheet" type="text/css" href="adv.css"
media="screen" />
<link rel="stylesheet" type="text/css" href="print.css"
media="print" />
```

Discussion

To create a print stylesheet, comment out the screen stylesheet and then create a separate, secondary stylesheet. In this second stylesheet, build the rules to dictate how you want the page to look when printed. After you have completed the stylesheet, associate the stylesheet with a `link` element, as mentioned in the Solution.

Media types

Stylesheets can dictate the presentation of documents to a wide range of media. By default, the value for the `media` attribute is `all`. Without the attribute in the `link` element, the user agent will apply the CSS rules in the stylesheet to all media.

Although the most common attribute you probably have encountered is `screen`, which is used mainly for displaying documents on color monitors, the CSS 2.1 specification actually defines a total of 10 media types, as shown in Table 10-1.

Table 10-1. Media types for stylesheets

Media type	Description
all	Suitable for all devices
braille	Intended for Braille tactile feedback devices
embossed	Intended for paged Braille printers
handheld	Intended for handheld devices (typically small-screen, limited-bandwidth devices)
print	Intended for paged material and for documents viewed on-screen in print preview mode
projection	Intended for projected presentations—for example, projectors
screen	Intended primarily for color computer screens
speech	Intended for speech synthesizers
tty	Intended for media using a fixed-pitch character grid (such as teletypes, terminals, or portable devices with limited display capabilities)
tv	Intended for television-type devices (with low-resolution, limited-scrollable color screens and available sound)

When defining the styles for your web page, you can use one stylesheet for all forms of media:

```
<link rel="stylesheet" type="text/css" href="uber.css"
media="all" />
```

Or you can use one stylesheet for several, but not all, forms of media.

For instance, to use one stylesheet for both projection and print media, separate the media values with a comma:

```
<link rel="stylesheet" type="text/css" href="print.css"
media="print,projection" />
```

In the preceding code, the *print.css* stylesheet is used for projection and print media when rendering the web document. (It's probably not the ideal solution, as a design for print probably won't be appropriate for projection.)

Using @import when assigning media types

You can use other methods besides `link` to assign media types. One method is `@import`, as shown in the following line, which specifies the stylesheet for print and projection media:

```
@import URI(print.css) print,projection;
```

You need to place the `@import` rule within a `style` element or within an external style-sheet. However, since Internet Explorer doesn't render print stylesheets through the `@import` rule, it's best to avoid its use.

Using @media when assigning media types

Another method you can use to associate and dictate stylesheets and media types is `@media`, which enables you to write blocks of CSS rules that can be set for different media, all in *one* stylesheet:

```
<style type="text/css">
@media print {
 body {
   font-size: 10pt;
   background-color: white;
   color: black;
 }
}
@media screen {
 body {
   font-size: medium;
   background-color: black;
   color: white;
 }
}
</style>
```

See Also

"Media types" in Section 7 of the CSS 2.1 Working Draft at *http://www.w3.org/TR/CSS21/media.html*

10.2 Replacing a Color Logo for a Black-and-White Logo When Printing Web Pages

Problem

You want to swap a color logo for a logo that is more suitable for printing, without inserting two logo images into the HTML or creating a separate printer-friendly web page.

Solution

Code the HTML for the web document to include a black-and-white logo, as shown in Figure 10-1:

```
<div id="header">
 <h1><a href="/"><img src="e4h_logo-print.gif" /></a></h1>
</div><!-- /#header -->
```
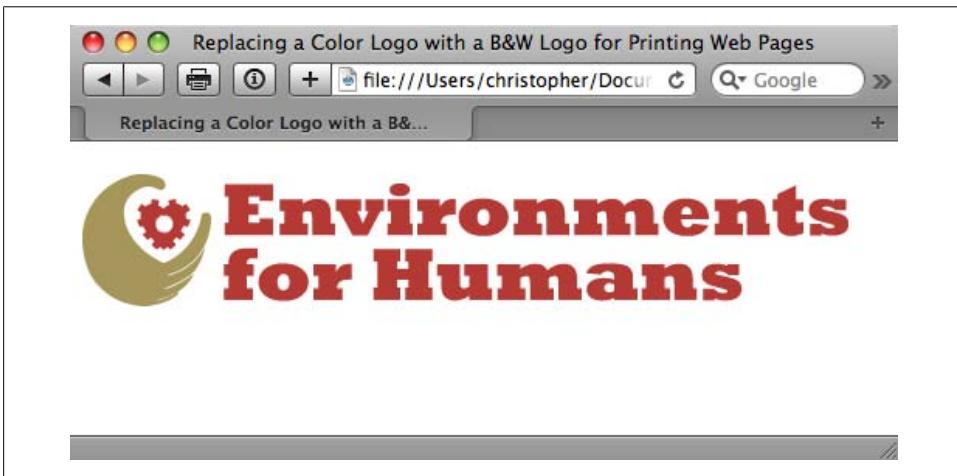

Figure 10-1. The color logo brought in through the background-image property

Next, keep the black-and-white logo from being displayed in the browser:

```
<style type="text/css" rel="stylesheet" media="screen">
 #header h1 img {
  display: none;
 }
</style>
```

Then bring in the color logo through the background of the h1 element, as shown in Figure 10-1:

```
<style type="text/css" rel="stylesheet" media="screen">
 #header h1 img {
  display: none;
 }
 #header h1 a {
  display: block;
  background-image: url(e4h_logo.gif);
  background-repeat: no-repeat;
  width: 494px;
  height: 85px;
 }
</style>
```

Since this stylesheet is reserved for screen media, the browser ignores the screen CSS rules and displays the black-and-white logo, as shown in Figure 10-2.
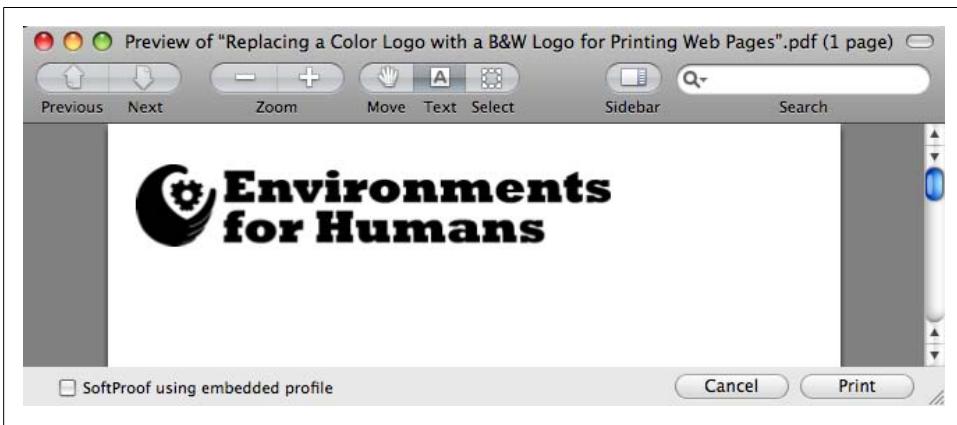
Figure 10-2. The black-and-white logo printed out

Discussion

This technique uses a basic image replacement method, as shown in Recipe 4.17. Instead of removing HTML text, the printer-friendly image is replaced for an image that is more suitable for full-color display.

This swapping of images works by setting specific rules based on the media type being used. When you set the media type in the initial CSS code snippet to screen, the browser ignores the CSS rules that hid the black-and-white image as it starts to process the document for printing.

If you don't distinguish the CSS rules for your main stylesheet with a media type, the browser assumes the value is all. Any additional print-only CSS rules are then mixed with your other CSS rules, which might cause unwanted results when printing the web document, as the cascade tries to determine the look of the page when it's printed. So, when setting up a print-only stylesheet, make sure you set your styles to the correct media type.

See Also

The "CSS Logo Replacement" blog post at *http://www.ibloomstudios.com/articles/css _logo_replacement/*

10.3 Making a Web Form Print-Ready

Problem

You need to have a form that users can fill out online, or that they can print and then fill out offline, as shown in Figure 10-3.

Figure 10-3. An online form

Solution

First, create a print media stylesheet and a class selector that transforms the `form` elements so that they display black text and feature a 1-pixel border on the bottom.

For example, consider the following HTML code for an `input` text element:

```
<label for="fname">First Name</label>
<input class="fillout" name="fname" type="text" id="fname" />
```

To style the `form` element requires the following CSS rule:

```
<style type="text/css" media="print ">
.fillout {
 color: black;
 border-width: 0;
 border: 1px solid #000;
 width: 300pt;
}
</style>
```

For drop-down menus, hide the `select` element altogether and add some additional markup to help produce the bottom border:

```
<label for="bitem">Breakfast Item</label>
<select name="bitem" size="1">
 <option selected="selected">Select</option>
 <option>Milk</option>
 <option>Eggs</option>
 <option>Orange Juice</option>
 <option>Newspaper</option>
 </select><span class="postselect">  </span>
```

Then, in the CSS rules, convert the inline `span` element to a `block` element. This enables you to set the width of the `span` element and places the border at the bottom to equal that of the `input` elements in the preceding CSS rule:

```
<style type="text/css" media="print">
select {
 display: none;
}
.postselect {
 display: block;
 width: 300pt;
 height: 1em;
 border: none;
 border-bottom: 1px solid #000;
}
</style>
```

For elements such as a Submit button, which can't be used on the printed page, set the `display` property to `none`. You can see the finished product in Figure 10-4.
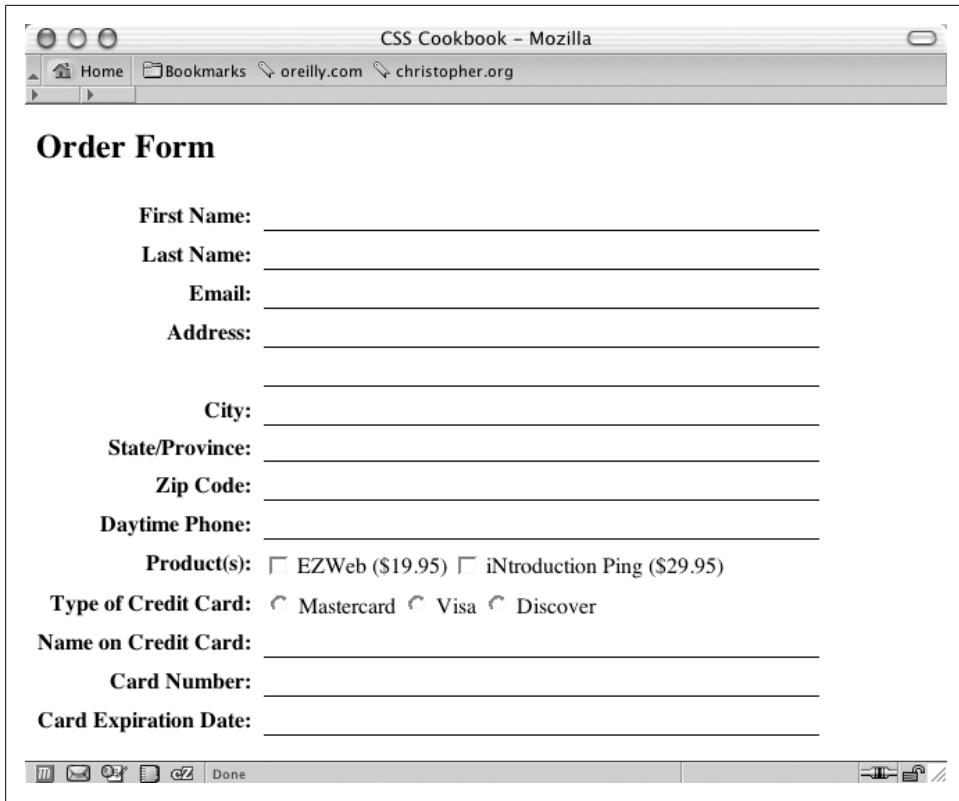
Figure 10-4. The same form primed for printing

Discussion

Lines created in the print stylesheet on an order form tell users they can fill out the form fields. By using the `border` property, you can easily create these lines in a browser, making web forms useful both online and offline.

For `select` elements, the workaround is somewhat of a hack that involves interfering with the ideal semantic markup; it still works and is valid HTML. Place a `span` element after the `select` element:

```
<select name="bitem" size="1">
 <option selected="selected">Select</option>
 <option>Milk</option>
 <option>Eggs</option>
 <option>Orange Juice</option>
 <option>Newspaper</option>
</select>
<span class="postselect">  </span>
```

Then set the `select` element to disappear:

```
select {
 display: none;
}
```

Next, set the `span` element to display as a block to enable the `width` and `height` properties. With those `width` and `height` properties set, you can place the bottom border to match the rest of the `form` elements:

```
.postselect {
 display: block;
 width: 300pt;
 height: 1em;
 border: none;
 border-bottom: 1px solid #000;
}
```

Using attribute selectors to differentiate form elements

Attribute selectors from the CSS specification make it easier to style forms for print. When you use attribute selectors, it's easier to distinguish which `form` elements should be stylized than it is when you insert `class` attributes and their respective values in the markup.

In the following code, the first CSS rule applies only to `input` elements for text, whereas the second rule hides the Submit button and the Select drop-down box:

```
input[type="text"] {
 color: black;
 border-width: 0;
 border: 1px solid #000;
}
input[type="submit"], select {
 display: none;
}
```

> The good news is that most modern browsers now support attribute selectors; however, Internet Explorer 6 does not.

Adding user friendliness

Since the form is now being printed, site visitors cannot use the Submit button to transmit their information. Be sure to provide the next steps users should follow after they have completed the printed form.

For example, if you want users to mail the form, add a mailing address to the page on which the form is printed, as shown in the following code:

```
<div id="print">
 <p>Please mail the form to the following address:</p>
```

```
 <address class="adr">
  <span class="org">
   <span class="organization-name">The White House</span>
  </span><br />
  <span class="street-address work postal">1600 Pennsylvania Avenue NW
</span><br />
  <span class="locality">Washington, DC</span>
  <span class="postal-code">20500</span><br />
  <span class="country-name">USA</span>
 </address>
</div>
```

Notice that the instructions are wrapped with a `div` element where the `class` attribute's value is set to `print`. In the stylesheet for screen delivery, set the `display` property for this specific class to `none`:

```
<style type="text/css" media="screen">
 .print {
  display: none;
 }
</style>
```

With a separate stylesheet for print delivery, allow the instructions to be printed by setting the `display` property to `block`:

```
<style type="text/css" media="print">
 .print {
  display: block;
 }
</style>
```

See Also

Attribute selector documentation in the W3C specification at *http://www.w3.org/TR/CSS21/selector.html#attribute-selectors*; the HTML 4.01 specification for the `label` tag at *http://www.w3.org/TR/html401/interact/forms.html#edef-LABEL*

10.4 Displaying URIs After Links

Problem

You need to display URIs of links in an article when a web page is printed.

Solution

Instruct the browser to print the URIs of links in a paragraph by using the `:after` pseudo-element:

```
p a:after {
 content: " <" attr(href) "> " ;
}
```

Discussion

Selector constructs such as `:after` are known as pseudo-elements. The browser inter-prets the selector as though additional elements were used to mark up the web document.

For example, by using the following CSS, you can make the first letter of a paragraph 2 em units in size:

```
p:first-letter {
 font-size: 2em;
}
```

You use the `:after` selector (or the `:before` selector) to insert generated content after (or before) an element. In this recipe, the value of the `href` attribute, which contains the URI information, is placed after every anchor element in a `p` element.

To have brackets appear around the URI, place quote marks around the brackets. To add a buffer of space between the anchor element and the next inline content, put one space in front of the left bracket and one after the right bracket, and then insert the URI using the `attr(x)` function. CSS finds in the element whatever attribute is replaced for *x*, returning its value as a string.

Another example of the power of this pseudo-element involves returning the value of abbreviations and acronyms in a buzzword-laden document:

```
<p>The W3C makes wonderful things like CSS!</p>
```

To accomplish this, first put the expanded form of the word or phrase in the `title` attribute for `abbr` or `acronym`:

```
<p>The <acronym title="World Wide Web Consortium">W3C</acronym>
makes wonderful things like <abbr title="Cascading Style
Sheets">CSS</abbr>!</p>
```

Then, in the CSS rules, tell the browser to return the value for the `title` attribute:

```
abbr:after, acronym:after {
 content: " (" attr(title) ") ";
}
```

Placing the domain name before absolute links

With absolute links, only the forward slash and any other folder and filename data will appear once the page is printed. To work around this dilemma, the CSS3 specification offers a solution through a substring selector:

```
p a:after {
 content: " <" attr(href) "> " ;
}
p a[href^="/"]:after {
 content: " <http://www.csscookbook.com" attr(href) "> " ;
}
```

The caret (^) signifies that the selector picks every link that starts with the forward slash, which signifies an absolute link.

Known browser issues

Currently, generating content through pseudo-elements works only in Firefox, Chrome, and Safari browsers. Microsoft introduced support for `:after` and `:before` pseudo-elements in IE8.

See Also

Recipe 3.6 for more on setting type in a web document; the CSS 2.1 specification about generated content at *http://www.w3.org/tr/rec-css2/generate.html#content*

10.5 Inserting Special Characters Before Links

Problem

You want to insert special characters, such as », before a link in a print stylesheet.

Solution

Making sure your stylesheet is set to print media, use the `:after` or `:before` pseudo-element to include the URI after a link in the web document:

```
p a:after {
 content: attr(href) ;
}
```

Next, place the hexadecimal equivalent of the special character before the link:

```
p a:after {
 text-decoration: underline;
 content: " \00BB "  attr(href);
}
```

When the page is printed, the text after a link might look like this:

```
» http://www.csscookbook.com/
```

Discussion

Make sure to use the backward slash to escape the hexadecimal value so that the browser does not display the hexadecimal value as generic text. In this case, if the hexadecimal value for right-double-angle quote marks were not escaped, the text "00BB" would be displayed instead:

```
00BB http://www.csscookbook.com/
```

Due to the nature of CSS syntax, it is not possible to use HTML numbers or names to identify special characters with the `content` property. The characters need to be escaped by a backward slash and their hexadecimal value.

Special characters through the CSS `content` property can also be used outside the printed page. Try it within your screen media presentation of your web design. Make sure you include the CSS declaration in a stylesheet with `media` set to `all` or `screen` to view the output.

Known browser issues

Currently, generating content through pseudo-elements works only in Firefox, Mozilla, Chrome, and Safari browsers. Generated content works in Internet Explorer for Windows 8.

See Also

A list of special characters and their hexadecimal equivalents at *http://www.ascii.cl/ htmlcodes.htm*; the CSS 2.1 specification for escaped characters at *http://www.w3.org/ TR/CSS21/syndata.html#escaped-characters*

10.6 Setting Page Breaks for a Printed Document

Problem

You want to place page breaks when printing within a long web document, as shown in Figure 10-5.
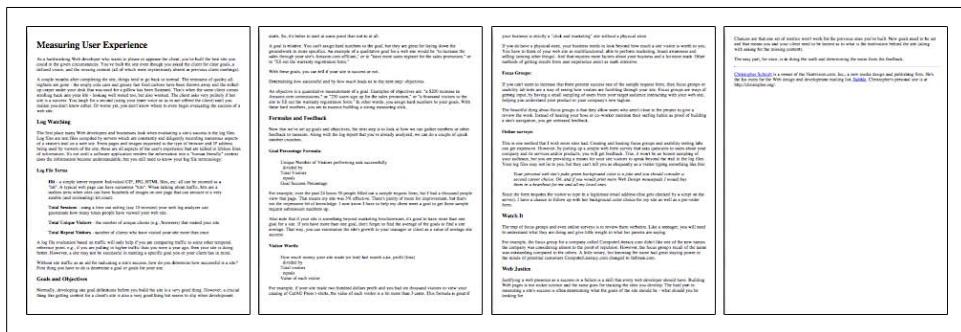
Figure 10-5. The default rendering of a page when printed

Solution

Use the `page-break-before` property to signify that a document should skip to the next page when printed, as shown in Figure 10-6:

```
h3 ~ h3 {
  page-break-before: always;
}
```
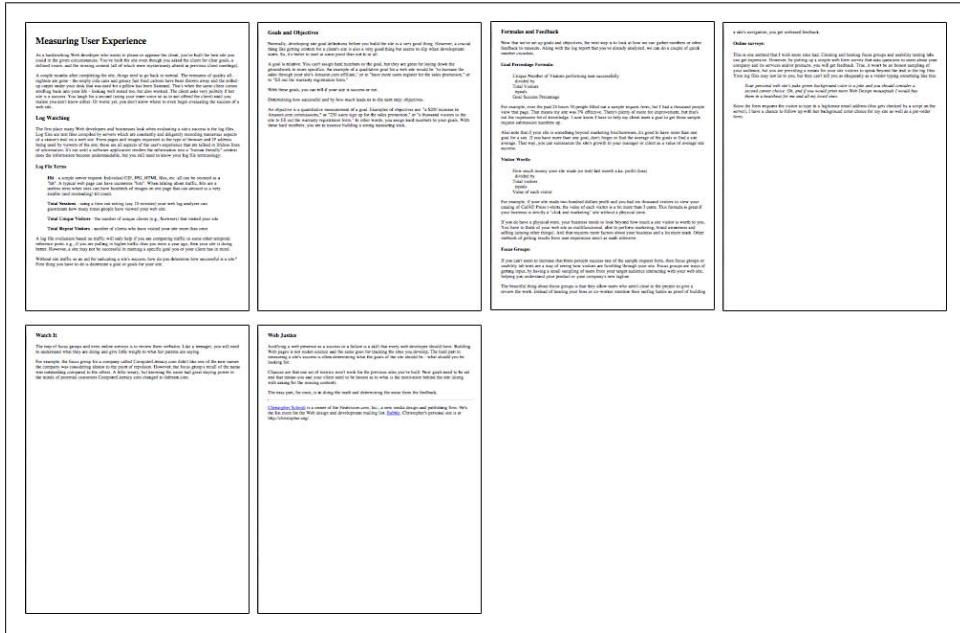

Figure 10-6. Page breaks introduced into the printed document

Discussion

By using semantic markup in your document, it is very straightforward to place page breaks within a web document.

The Solution code uses a CSS3 *general sibling combinatory selector*. The rule states that every time an `h3` element is preceded by another `h3` element, there should be a forced page break. To paraphrase that meaning, basically every `h3` element will be at the top of a printed page starting with the second `h3` element in the row.

Using class selectors

The Solution works because it uses a structured document with semantic heading tags and a browser that understands the CSS3 selector. However, when dealing with a document that does not use semantic markup, pinpointing the page breaks within the HTML is still fairly easy to do.

First, create a class selector containing the `page-break-before` property:

```
.pageBreak {
 page-break-before: always;
}
```

Then, embed the rule whenever you want a page break before the content:

```
<table cellspacing="0" class="pageBreak">
...
</table>
```

See Also

The "Page Breaks" specification in the CSS3 Working Draft at *http://kent.w3.org/TR/css3-page/#page-breaks*

10.7 Sample Design: A Printer-Friendly Page with CSS

In this sample design, you will transform an existing web document (as shown in Figure 10-7) to make it more suitable for print.

Although CSS has changed the way we design for the Web, it also has allowed developers to change the way they provide printer-friendly versions of their documents. Instead of having to create separate pages or write scripts, you can use CSS to create a printer-friendly document as soon as the user clicks the Print button. The HTML for the page isn't in the book because the miracle of CSS lets us change the presentation without having to change the HTML.

When creating a stylesheet for print, you actually use a web browser. This enables you to quickly see how the CSS rules affect the display of the document (just like for media delivery), but it's also easier on the environment and you save money by not wasting ink in the printer. So, comment out the stylesheet used for the screen to create new CSS rules:

```
<!-- Hide screen media CSS while working on print CSS -->
<!-- link href="adv.css" type="text/css" rel="stylesheet"
media="screen" -->
<style type="text/css">
/* Print CSS rules go here */
</style>
```

Setting the Page for Black-and-White Printing

Apply the first CSS rule to the `body` element. In this rule, set the background color to white and set the type to black:

```
body {
 background-color: white;
 color: black;
}
```

Figure 10-7. Web page stylized for screen delivery

Next, set the typeface for the page to a serif font. Reading text online in sans serif is easier on the eyes, but in print media the serif font is still the choice for reading passages of text. For a later fallback choice, you might want to go with the Times typeface for print documents, since it's installed on most (if not all) computers and it's a workhorse of a font. In case your users don't have Times installed, supply alternatives as well:

```
body {
 background-color: white;
 color: black;
 font-family: Times, "Times New Roman", Garamond, serif;
}
```

Now you want to get rid of navigation-related links and other page elements you don't want to see in the final printout. This includes the main navigation bar below the main header, as well as internal anchors in the page itself. If you have a page with ad banners, it might be a good idea to hide those, too (see Figure 10-8):

```
#navigation, hr, body>div>a, #blipvert {
 display: none;
}
```
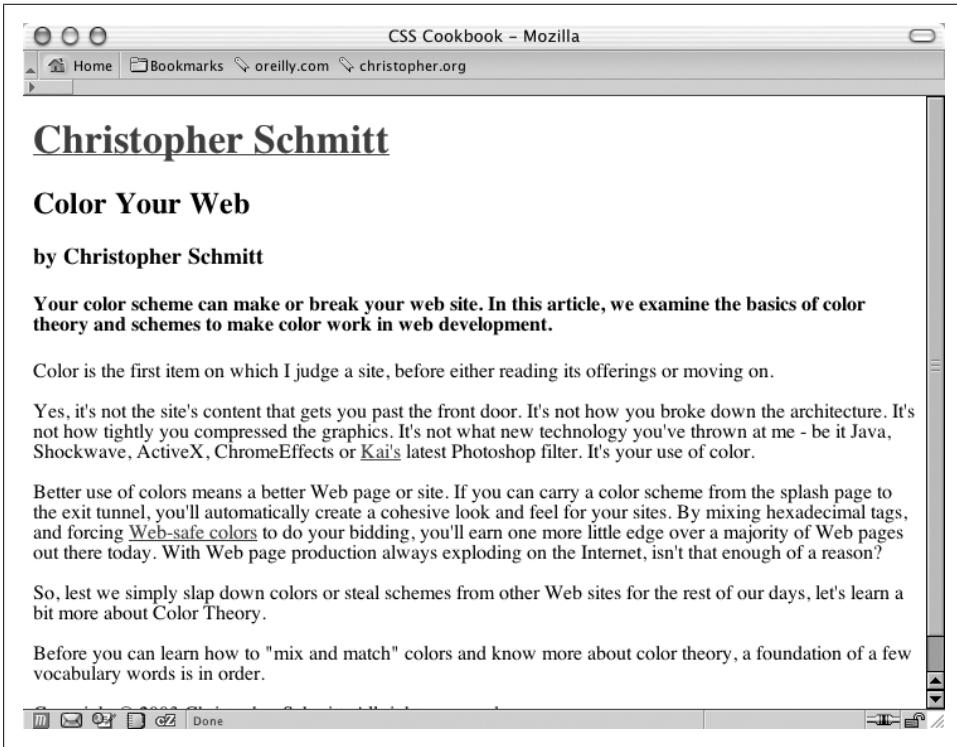

Figure 10-8. Hiding the navigation bar and other elements

Designing the Main Heading

Because you are dealing with black and gray type on a white page, you have few options when it comes to designing how the main heading for the page should look. However, using what you have at your disposal, it's nonetheless easy to create a masthead that calls attention to itself.

First, set the background to black and the text to white:

```
#header h1 {
 color: white;
 background-color: black;
}
```

Because you want people to actually read the header, you want the text to be white to create enough contrast. In this instance, the main header also acts as a homing device— it is a link to the home page. Therefore, the color of the heading is dictated by the style rules set for the links. To remedy this situation, add a separate rule:

```
#header h1 {
 background-color: black;
}
#header h1 a {
 color: white;
}
```

Now that the text is visible, stylize it a bit so that it stands out. Your goal is to center the text, increase the size of the text, and make all the letters uppercase:

```
#header h1 {
 background-color: black;
 font-size: 24pt;
 text-align: center;
 text-transform: uppercase;
}
```

Although this looks good, you can improve it by changing the typeface to sans serif (so that it sticks out from the rest of the text in the document) and by adding some padding around the top and bottom of the heading (see Figure 10-9):

```
#header h1 {
 background-color: black;
 font-size: 24pt;
 text-align: center;
 font-family: Helvetica, Verdana, Arial, sans-serif;
 padding: 7pt;
 text-transform: uppercase;
}
```

Styling the Article Header and Byline

For the article title and byline, create a more dramatic look by zeroing out the margins and padding of both the h2 and h3 elements:

```
#content h2 {
 padding: 0;
 margin: 0;
}
#content h3 {
 padding: 0;
 margin: 0 ;
}
```
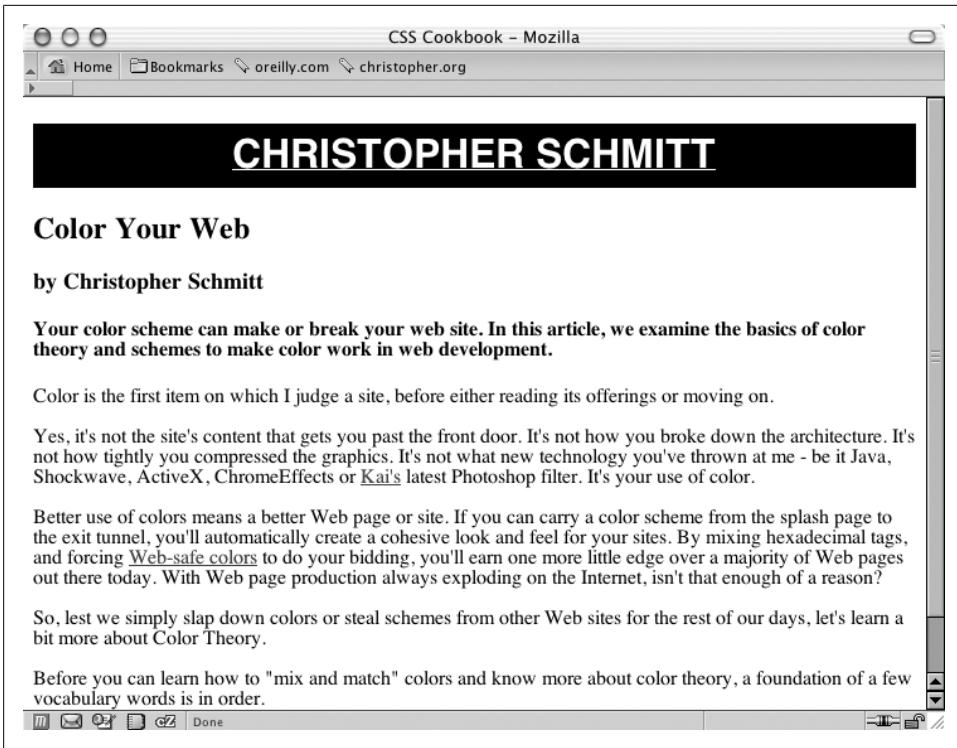
Figure 10-9. Stylizing the main header

Then increase the font size for the article title and create a thin hairline rule below it. Next, align the byline to the right and set the type style to italic (see Figure 10-10):

```
#content h2 {
 padding: 0;
 margin: 0;
 font-size: 20pt;
 border-bottom: 1px solid black;
}
#content h3 {
 padding: 0;
 margin: 0;
 text-align: right;
 font-style: italic;
}
```
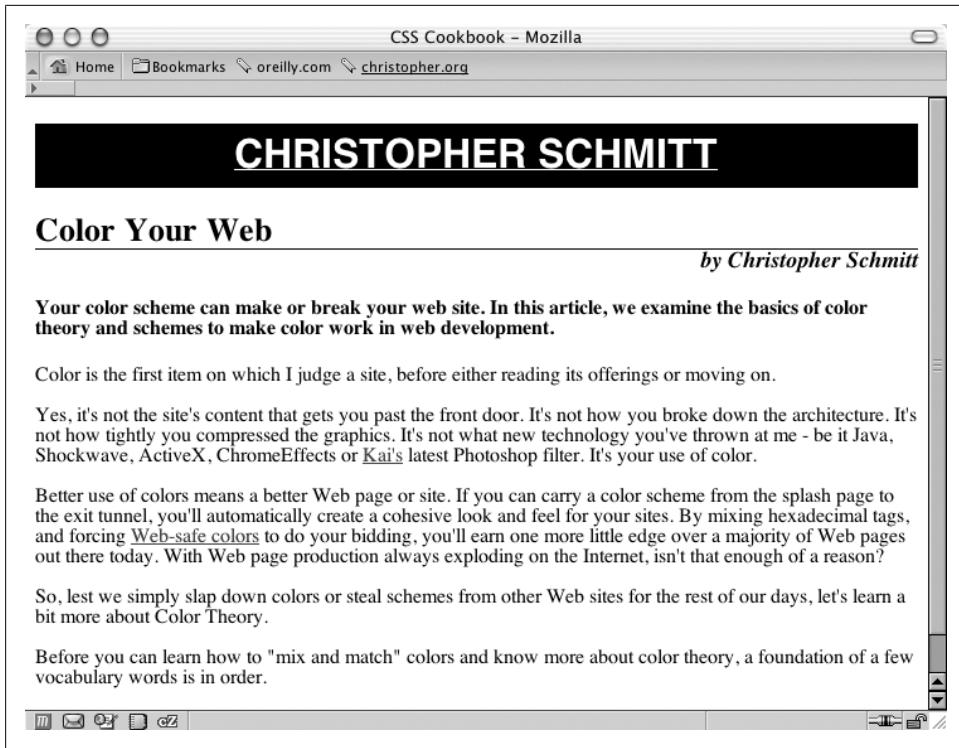
Figure 10-10. Designing the article header and byline

Gaining Attention Through the Teaser

Next up is the content in the h4 element. Because this content serves as a teaser for the article, it should be visually distinctive from the article text. To accomplish that, set the background to about 30% black, change the typeface to sans serif, and put in some padding (see Figure 10-11):

```
#content h4 {
  font-family: Helvetica, Verdana, Arial, sans-serif;
  border-top: 3pt solid black;
  background-color: #BEBEBE; /* ~30% black */
  padding: 12pt;
  margin: 0;
}
```

As for the content of the article, leave the text pretty much as it is except for two points of interest: leading, covered here, and links, covered in the next section.

Remember that in the body element, the font for the entire page is set with the serif typeface, and through inheritance that typeface style is picked up in the paragraph elements as well. However, you may want to space out the lines, or increase the leading, of the text in the paragraph. To do this, change the line-height property:

```
#content p {
  line-height: 18pt;
}
```
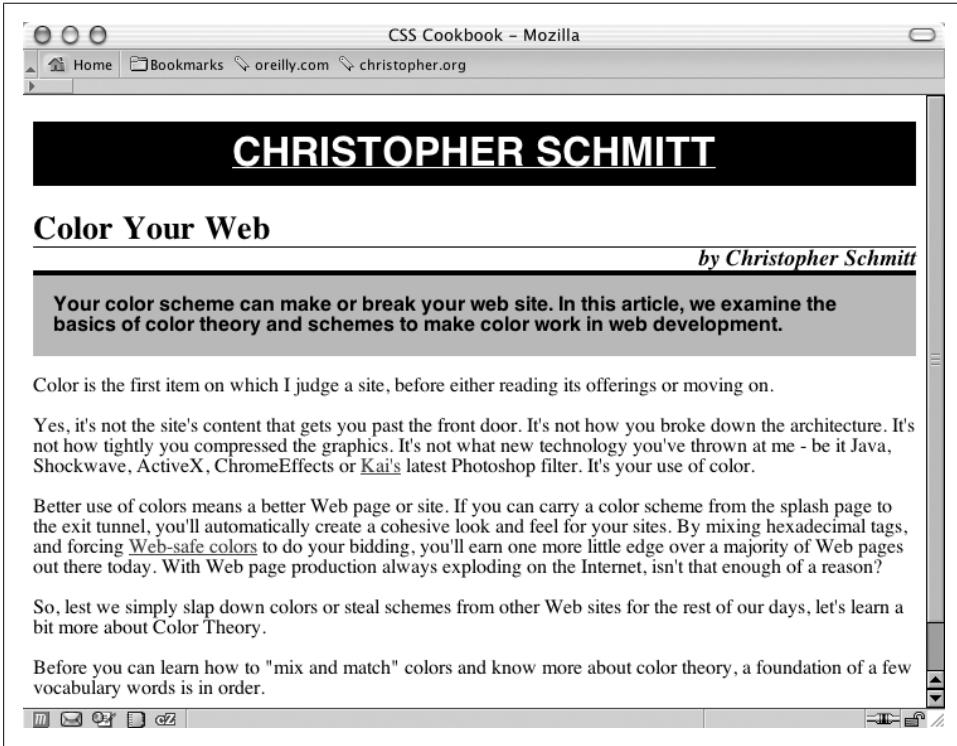

Figure 10-11. Setting up the article teaser

Displaying URIs After Links

Any links in the article become useless when printed. To make them beneficial to the reader when the page is printed, make sure all URIs from the links are displayed. To do that, set up a CSS rule to display the URIs after every link in the content division of the document. Also, for visual effect, remove the default underline of the links, make sure the `font-weight` is bold, and set the color to gray (see Figure 10-12):

```
#content a:after {
  content: " <" attr(href) "> ";
  font-family: courier, monospace;
  font-weight: normal;
}
a {
  text-decoration: none;
  font-weight: bold;
  color: #626466;
}
```
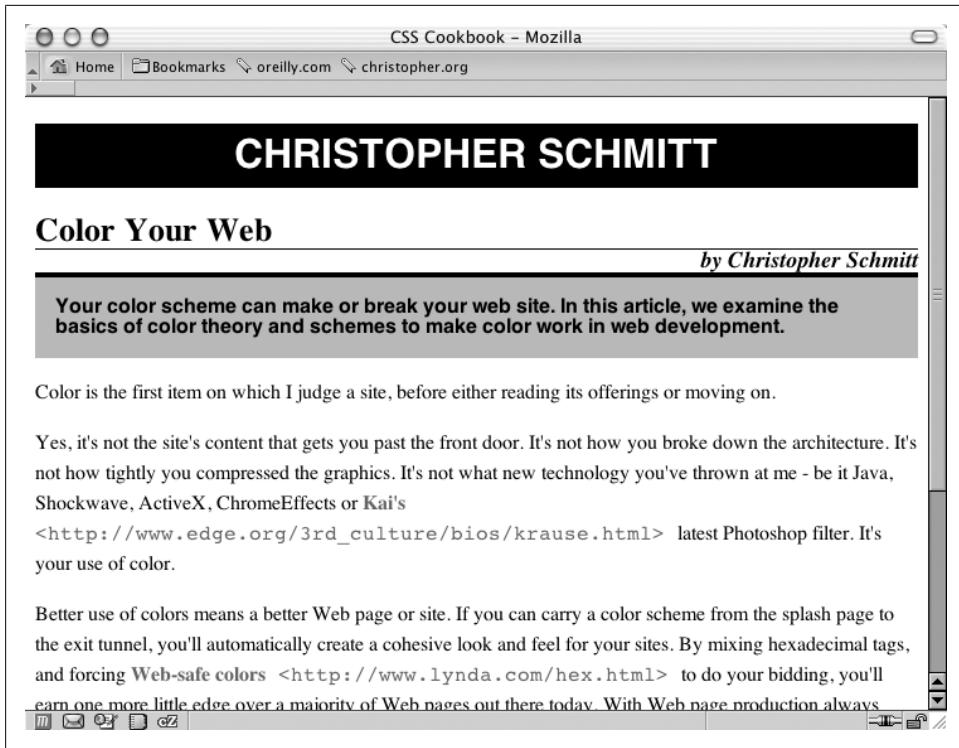
Figure 10-12. Adjusting the links and leading in the content

Finishing with the Footer

At this point you're ready to work your way down the page to the footer that contains the copyright notice. Because the main header is in a sans serif typeface, balance the page by centering the copyright notice, create a line rule through the `border-top` property, and set the typeface to sans serif as well, as shown in Figure 10-13:

```
#footer {
  border-top: 1px solid #000;
  text-align: center;
  font-family: Helvetica, Verdana, Arial, sans-serif;
}
```

With the print CSS finished, copy the CSS rules and put them into an external stylesheet called *print.css*. Then, comment out the CSS for screen media and associate the print CSS through the `link` element:

```
<link href="adv.css" type="text/css" rel="stylesheet"
media="screen" />
<link href="print.css" type="text/css" rel="stylesheet"
media="print" />
```
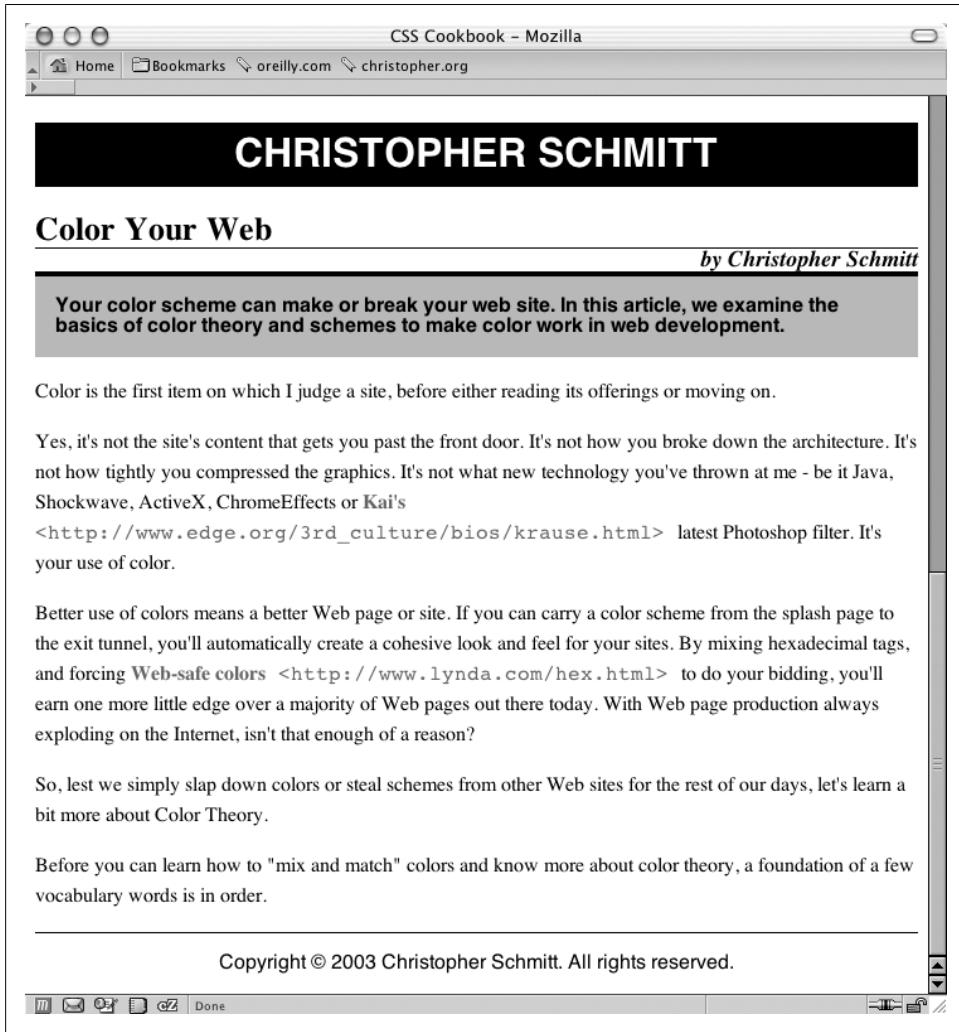
Figure 10-13. The styled footer

This concludes the demonstration of how you can create a printer-friendly stylesheet. Assuming that the site visitors have a browser that can render print media stylesheets, when the site visitors print out the page, they will automatically get the proper layout sent to their printer.

Page Layouts

11.0 Introduction

One of the last frontiers in CSS-enabled design was creation of the page layout.

For a long time, web developers used HTML tables to create their layouts, often nesting tables to create multicolumn, multilevel layouts.

However, this new approach of letting CSS do the heavy lifting brings many advantages. Meaningful content that was once trapped under so many nested tables and images is now placed within meaningful heading and paragraph tags, allowing for improved search engine ranking.

Also, file sizes and maintenance headaches are noticeably diminished. As a result, launching a complete redesign of a website is a snap, when it used to take hours and sometimes days with HTML tables.

This chapter discusses the many ways in which you can create column layouts—including simple one-column layouts, four-column layouts, and everything in between.

11.1 Building a One-Column Layout

Problem

You want to build a layout that consists of one main column, as shown in Figure 11-1.

Solution

Apply a percentage value to the left and right margins of the web document's `body` element:

```
body {
 margin-left: 15%;
 margin-right: 15%;
}
```
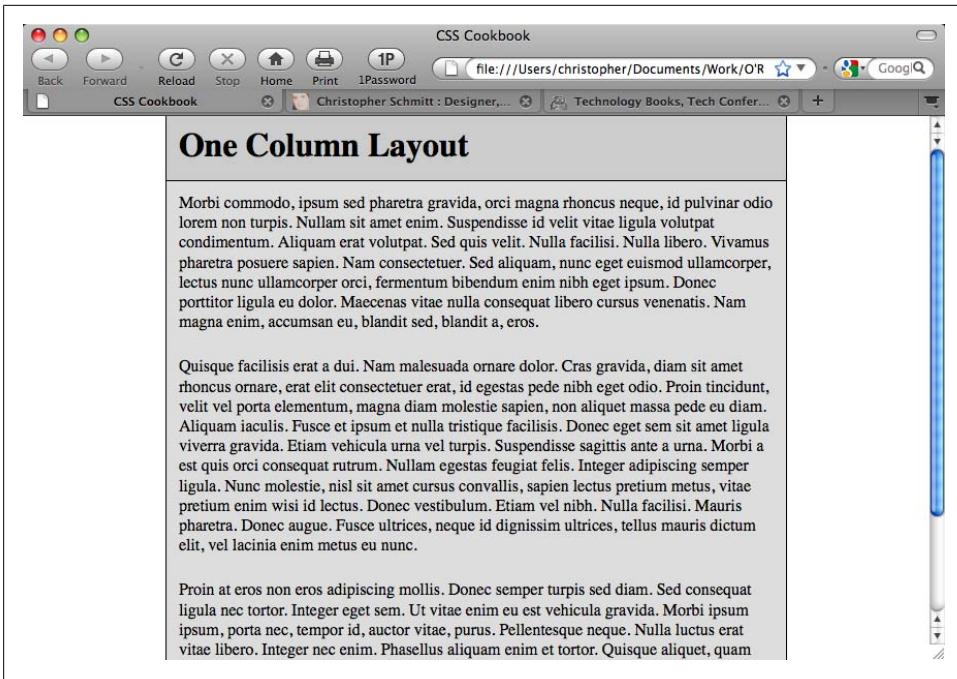
Figure 11-1. One-column page reinforced by an increased margin

Discussion

When you apply a percentage value to the left and right margins of the body, the column width becomes flexible. This allows the content to stretch to the width of the user's browser.

Creating a fixed-width column

This technique aligns the column to the left side of the user's browser. If you want to center a column with a fixed width, wrap a `div` element around the entire contents of the web document with a specific, unique `id` attribute such as `easel`:

```
<div id="easel">
 [...]
</div>
```

Set the `width` property for an `id` selector:

```
#easel {
 width: 600px;
}
```

Then use the `margin` shorthand property to assign a value of `auto` for the left and right sides of the `div` element:

```
#easel {
 width: 600px;
 margin: 0 auto;
}
```

See Also

Recipe 5.4 for centering elements in a web document; Recipe 7.12 for more on horizontal tab navigation

11.2 Building a Two-Column Layout

Problem

You want to create a two-column layout with columns that resize to the width of the browser, as shown in Figure 11-2.
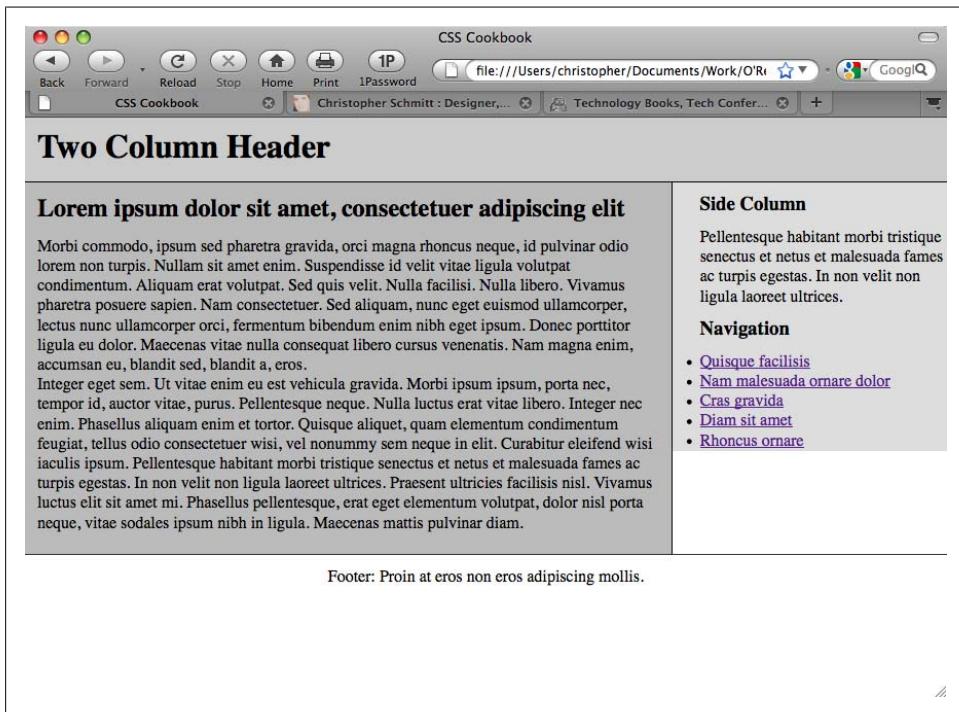
Figure 11-2. Two-column layout achieved through CSS

Solution

First, mark up the content with `div` elements using the `id` attributes that contain appropriate values. Semantic values are preferred, such as `mainContent` or `sidebar`, instead of values that represent their placement on the page.

For *demonstration purposes only*, the values of the `id` attributes are used to show where the content is displayed to help you better understand how the content is going to be displayed in the code:

```
<div id="columnLeft">
 [...]
</div>
<div id="columnRight">
 [...]
</div>
<div id="footer">
 [...]
</div>
```

Then, in CSS, use the `float` property to move the contents of the left column to the left, and set a width that is two-thirds the web document's width:

```
#columnLeft {
 float: left;
 width: 67%;
 background: #fff;
 margin-top: 0;
 margin-right: 1.67em;
 border-right: 1px solid black;
 padding-top: 0;
 padding-right: 1em;
 padding-bottom: 20px;
 }
```

The right column wraps around the contents of the left column. On the right column, set the top of the margin and padding to `0`, allowing the column and the first element in it to become level with the left column:

```
#columnRight {
 padding-left: 2em;
 margin-top: 0;
 padding-top: 0;
 }
h1 {
 margin-top: 0;
 padding-top: 0;
 }
```

To display the footer at the bottom of the web document, set the `clear` property to `both`:

```
#footer {
 clear: both;
 padding-bottom: 1em;
 border-top: 1px solid #333;
```

```
    text-align: center;
    }
```

Discussion

The `float` property (see Recipe 2.21) is similar to the `align` attribute that is used in HTML to allow text and other elements to flow around an image:

```
<img src="this.jpg" width="250" height="150" hspace="7" vspace="7"
alt="example" align="right" />
```

Once the image has been set to align to either the right or the left, the content around the image flows to the opposite side of the image's alignment.

For example, an image aligned to the right forces content to flow around the image on the left side. With CSS, floats provide a similar function, except they offer more exacting control over the presentation by using borders, margins, padding, and other properties.

To make sure the content that comprises the footer is placed at the bottom of the columns, set the `clear` property to a value of `both`. When you set the value to `both`, the browser understands that the content of the footer isn't flowing around the floated left column and positions it below (or past) any floated elements.

The only caveat to this technique for creating a two-column layout is that the content in the left column needs to be longer than the content in the right column. Because the content in the left column appears first in the document, the content in the right column wraps around the left column. Too much content in the column that doesn't float results in the anomaly shown in Figure 11-3.

A method for fixing this problem is to set off the left margin or padding on the right column element so that the column width is at least maintained after the content flows below the float:

```
#mainColumn {
 width: 400px;
 /* Enough padding to compensate for the left column */
 padding-left: 200px;
}
#navigation {
 float: left;
 width: 175px;
}
```

> As an alternative, you can set the `overflow` property to a value of `hidden` (as discussed in Recipe 2.22) instead:
>
> ```
> #mainColumn {
>  overflow: hidden;
> }
> ```
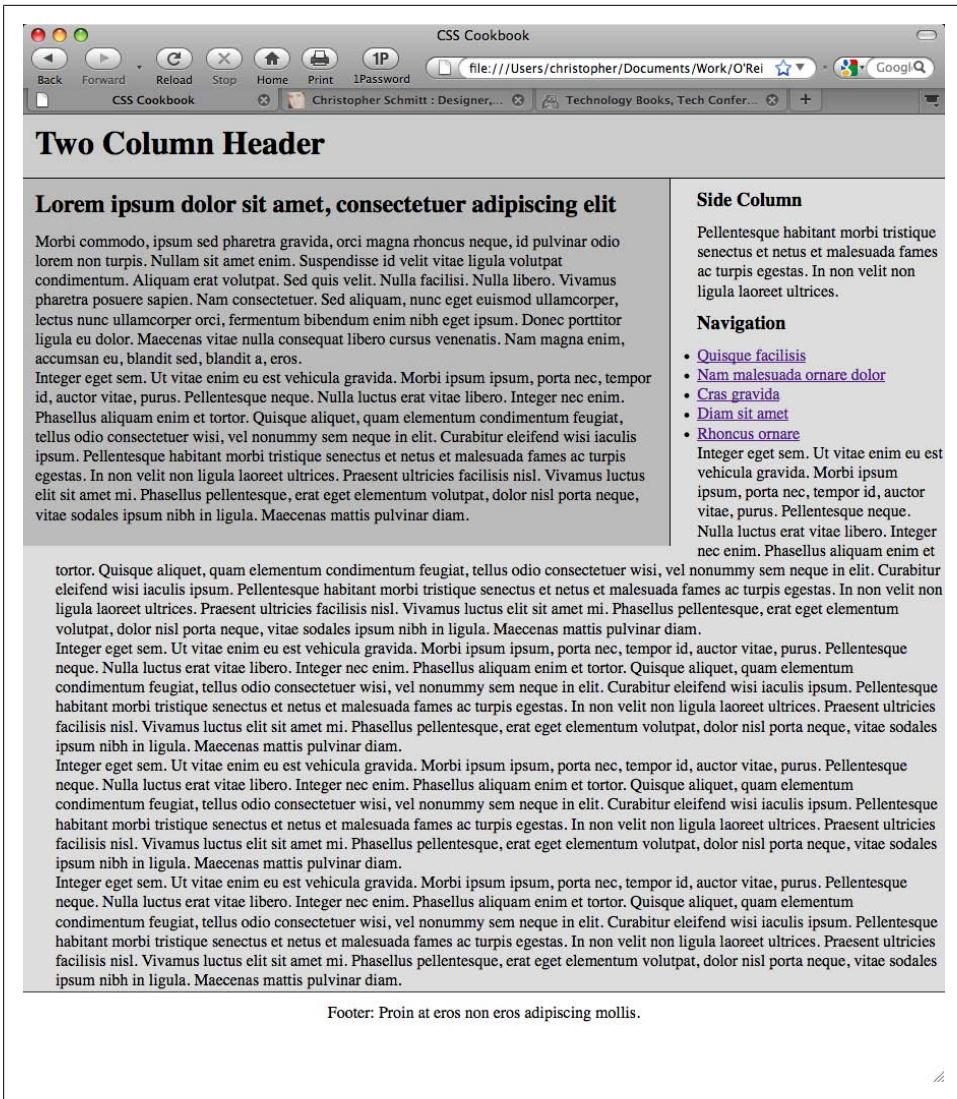
Figure 11-3. Unwanted wrapping of text under the left column

Flipping the layout

If you want to have the columns reversed, switch the order of the columns by using the following markup:

```
<div id="columnRight">
 [...]
</div>
<div id="columnLeft">
 [...]
```

```
</div>
<div id="footer">
 [...]
</div>
```

If I had named the attributes semantically instead of making sure to show where the content in the code is going to be displayed in the browser, I could have simply modified the CSS by floating the elements in opposite directions.

Then apply the following CSS to the columns:

```
#columnRight {
 float: right;
 width: 67%;
 padding-bottom: 20px;
 padding-top: 0;
}
#columnLeft {
 width: 29%;
 padding-right: 1em;
 border-right: 1px solid black;
 padding-top: 0;
}
```

See Also

Recipe 11.3 for information on building a two-column layout with fixed widths; Jeffrey Zeldman's "From Table Hacks to CSS Layout: A Web Designer's Journal" for a background on this Solution at *http://www.alistapart.com/articles/journey/*

11.3 Building a Two-Column Layout with Fixed-Width Columns

Problem

You want to create a two-column layout with fixed-width columns.

Solution

First, mark up the content with `div` elements using the `id` attributes that contain appropriate values representing their placement on the page:

```
<div id="header">
 [...]
</div>
<div id="columnLeft">
 [...]
</div>
<div id="columnRight">
 [...]
```

```
        </div>
        <div id="footer">
        [...]
        </div>
```

Using the `float` property set the width of the left column to a length unit rather than to percentages. Also, set the width of the entire document to a length unit (see Figure 11-4):

```
body {
  margin: 0;
  padding: 0;
  font-family: Georgia, Times, "Times New Roman", serif;
  color: black;
  width: 600px;
  border-right: 1px solid black;
}
#header {
  background-color: #666;
  border-bottom: 1px solid #333;
}
#columnLeft {
  float: left;
  width: 160px;
  margin-left: 10px;
  padding-top: 1em;
}
#columnRight {
  padding-top: 1em;
  margin: 0 2em 0 200px;
}
#footer {
  clear: both;
  background-color: #ccc;
  padding-bottom: 1em;
  border-top: 1px solid #333;
  padding-left: 200px;
}
```

Discussion

By default, block-level elements stretch to the width of their containers. If the browser window is small, the block-level elements shrink—in other words, text inside the contents wraps within the confines of the contents' shrinking walls.

However, when you use *pixel* units rather than percentages, the width of the columns becomes fixed. Even as a browser window shrinks or expands, the column widths remain fixed.

To keep the width of the left column fixed while enabling the main column to stretch, simply remove the `width` property assigned to the `body` element.
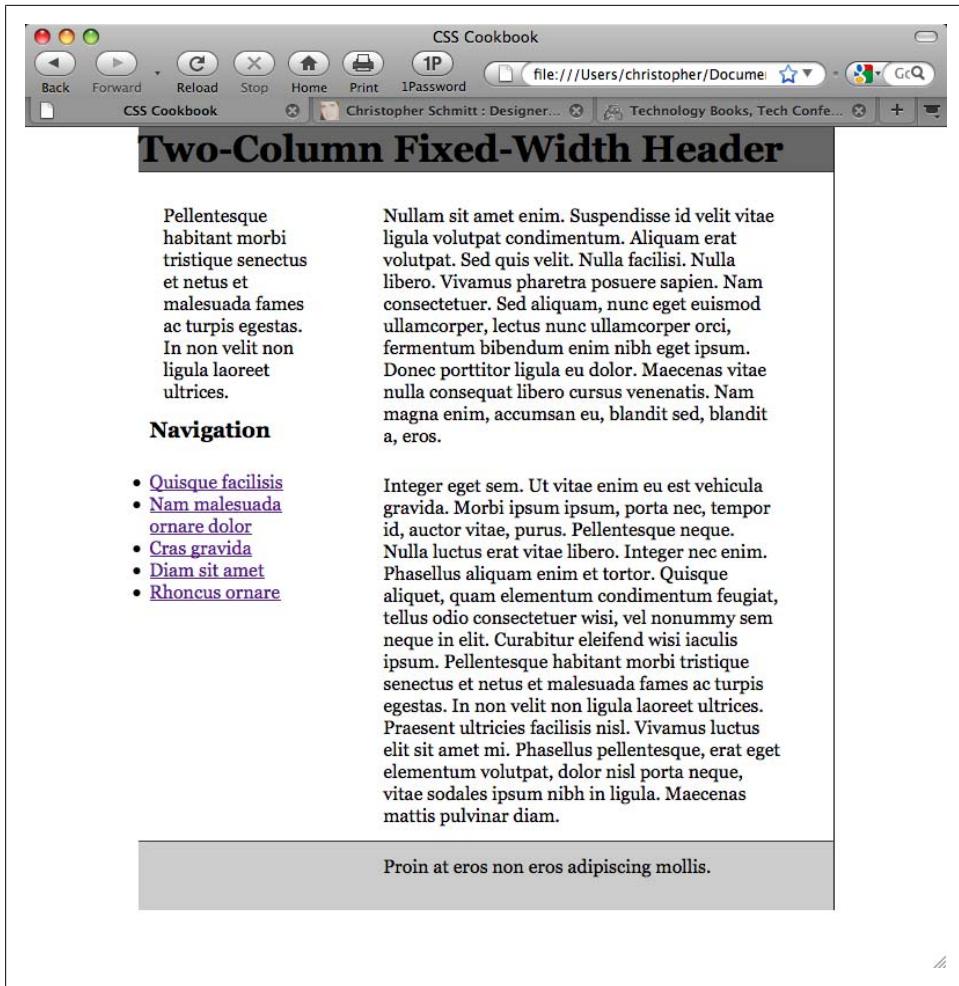
Figure 11-4. The two-column layout enabled by CSS

Flipping the layout

If you want to have the columns reversed, re-order the content with the following markup:

```
<div id="header">
 [...]
</div>
<div id="columnRight">
 [...]
</div>
<div id="columnLeft">
 [...]
</div>
<div id="footer">
```

```
 [...]
</div>
```

Then use the following updated CSS rules:

```
#columnLeft {
 width: 340px;
 margin-left: 10px;
 margin-top: 1em;
}
#columnRight {
 float: right;
 width: 200px;
}
#footer {
 clear: both;
 background-color: #ccc;
 padding-bottom: 1em;
 border-top: 1px solid #333;
 padding-left: 10px;
}
```

See Also

Recipe 11.2 for creating a two-column layout with flexible-width columns

11.4 Creating a Flexible Multicolumn Layout with Floats

Problem

You want to create a three-column layout with columns that resize to the width of the browser, as shown in Figure 11-5.

Solution

First, mark up the content with `div` elements using the `id` attributes that contain appropriate values representing their placement on the page:

```
<div id="header">
 [...]
</div>
<div id="columnLeft">
 [...]
</div>
<div id="columnMain">
 [...]
</div>
<div id="columnRight">
 [...]
</div>
<div id="footer">
 [...]
</div>
```
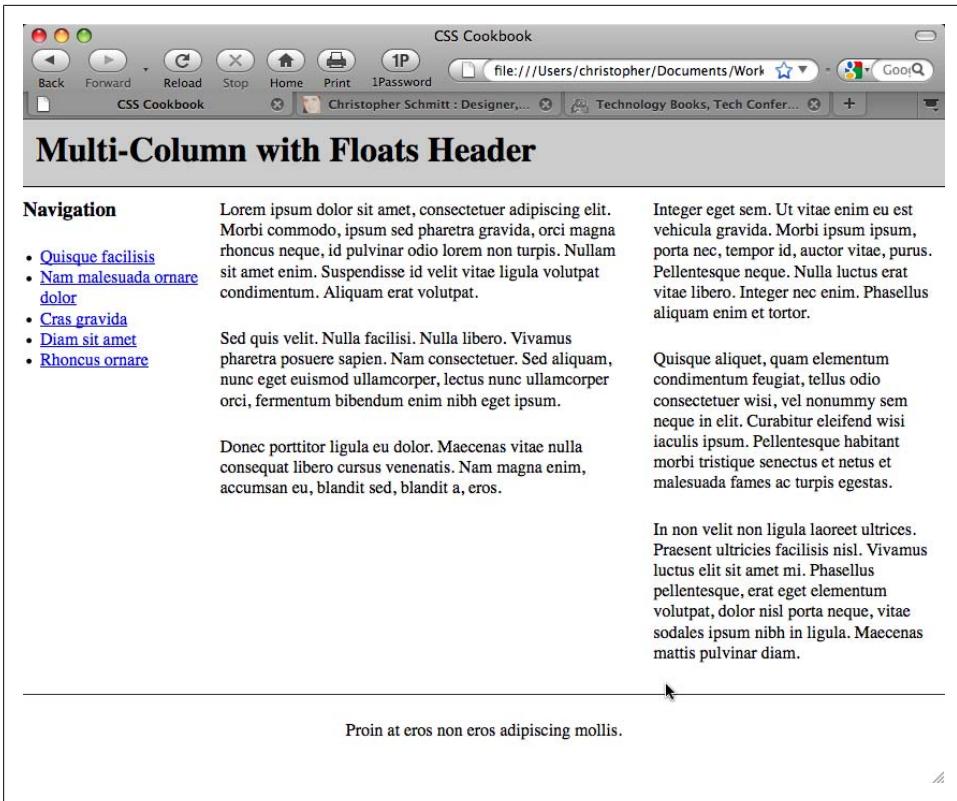
Figure 11-5. Three-column layout achieved through CSS

Next, set each column to float to the left, making sure that the width is a percentage.
All three values of the columns should equal 100% (see Figure 11-6):

```
#columnRight {
 width: 33%;
 float: left;
 background: white;
 padding-bottom: 1em;
}
#columnLeft {
 width: 20%;
 float:left;
 background: white;
 padding-bottom: 1em;
 text-align: justify;
}
#columnMain {
 width:47%;
 float:left;
 background: white;
 padding-bottom: 1em;
```

```
  }
  #footer {
   clear: both;
   padding-bottom: 1em;
   border-top: 1px solid #333;
   text-align: center;
  }
```
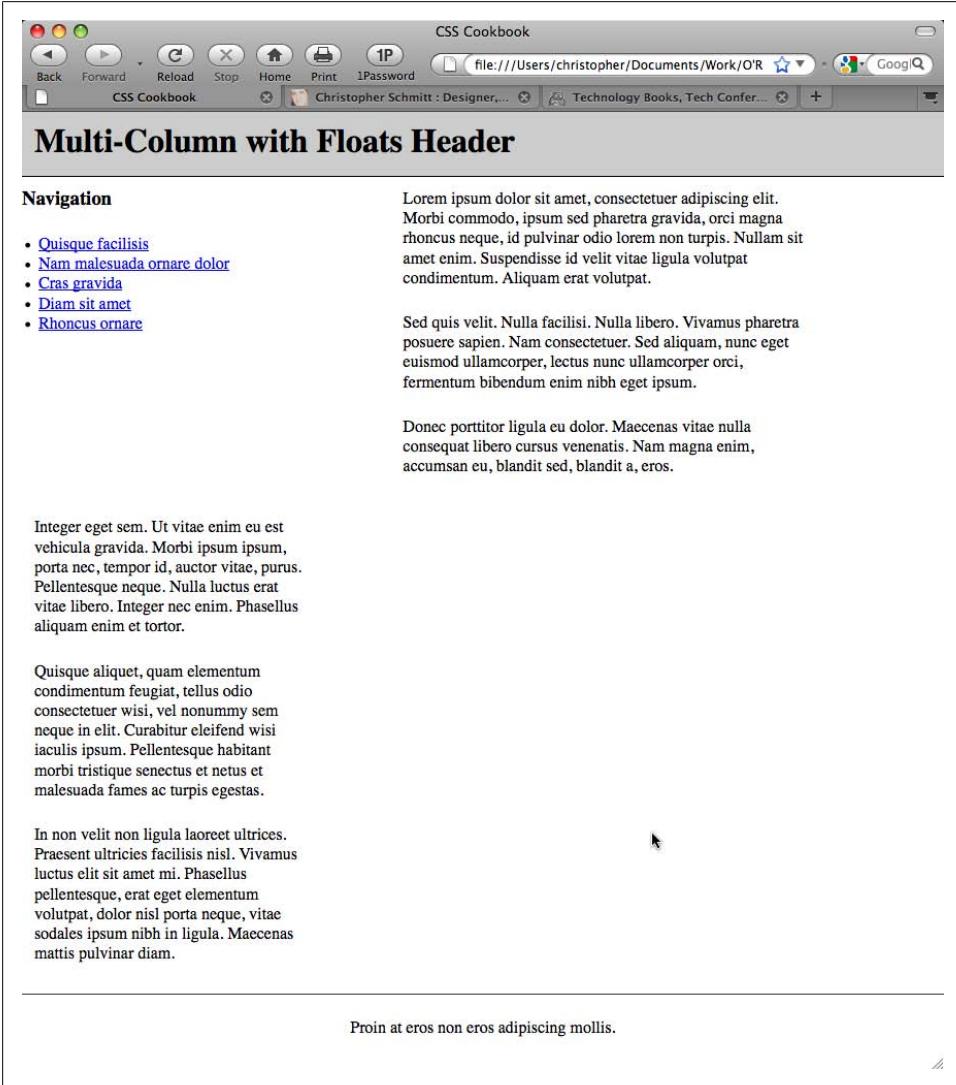

Figure 11-6. An increased width for the main column, forcing the right column to wrap underneath

Discussion

This technique works because all columns are set to float to the left and their widths aren't larger than 100%. Setting the floats to the right can flip the columns, but the result is the same.

Be sure to apply margins and padding to the elements within the columns (unless you account for their widths when sizing the columns). If you don't, the columns will expand beyond 100%, forcing one or more columns to wrap underneath each other.

See Also

Recipe 11.5 for creating a three-column layout with fixed-width columns; *http://www .realworldstyle.com/nn4_3col_header.html* for information on creating a three-column layout with one flexible-width column and two fixed-width columns

11.5 Creating a Fixed-Width Multicolumn Layout with Floats

Problem

You want to create a three-column layout with fixed-width columns.

Solution

First, mark up the content with `div` elements using the `id` attributes that contain appropriate values representing their placement on the page:

```
<div id="header">
 [...]
</div>
<div id="columnMain">
 [...]
</div>
<div id="columnLeft">
 [...]
</div>
<div id="columnRight">
 [...]
</div>
<div id="footer">
 [...]
</div>
```

Next, wrap the `div` elements that compose the main and left columns in another `div` element and set the value of the `id` attribute to `enclose`. Also, wrap another `div` element around the entire set of `div` elements, setting the value to `frame`:

```
<div id="frame">
 <div id="header">
  [...]
 </div>
```

```
<div id="enclose">
 <div id="columnMain">
  [...]
 </div>
 <div id="columnLeft">
  [...]
 </div>
</div>
<div id="columnRight">
 [...]
</div>
<div id="footer">
 [...]
</div>
<div>
```

Set the width of the page using an `id` selector for the "frame" `div` element:

```
#frame {
 margin-left: 20px;
 width: 710px;
}
```

Next, set the "column" `div` elements as well as the `div` element with an `id` value of enclose to `float` (see Figure 11-7):

```
#columnMain {
 float: right;
 width: 380px;
}
#columnLeft {
 float: left;
 width: 150px;
}
#columnRight {
 float: right;
 width: 120px;
}
#enclose {
 float:left;
 width:560px;
}
#footer {
 clear: both;
 padding-top: 1em;
 text-align: center;
}
```

Discussion

Because the width of the columns is set in pixels, the columns are fixed. To display the columns, you need an extra `div` element wrapped around the main and left columns. With this extra `div` element, which contains an `id` attribute value of enclose, the main
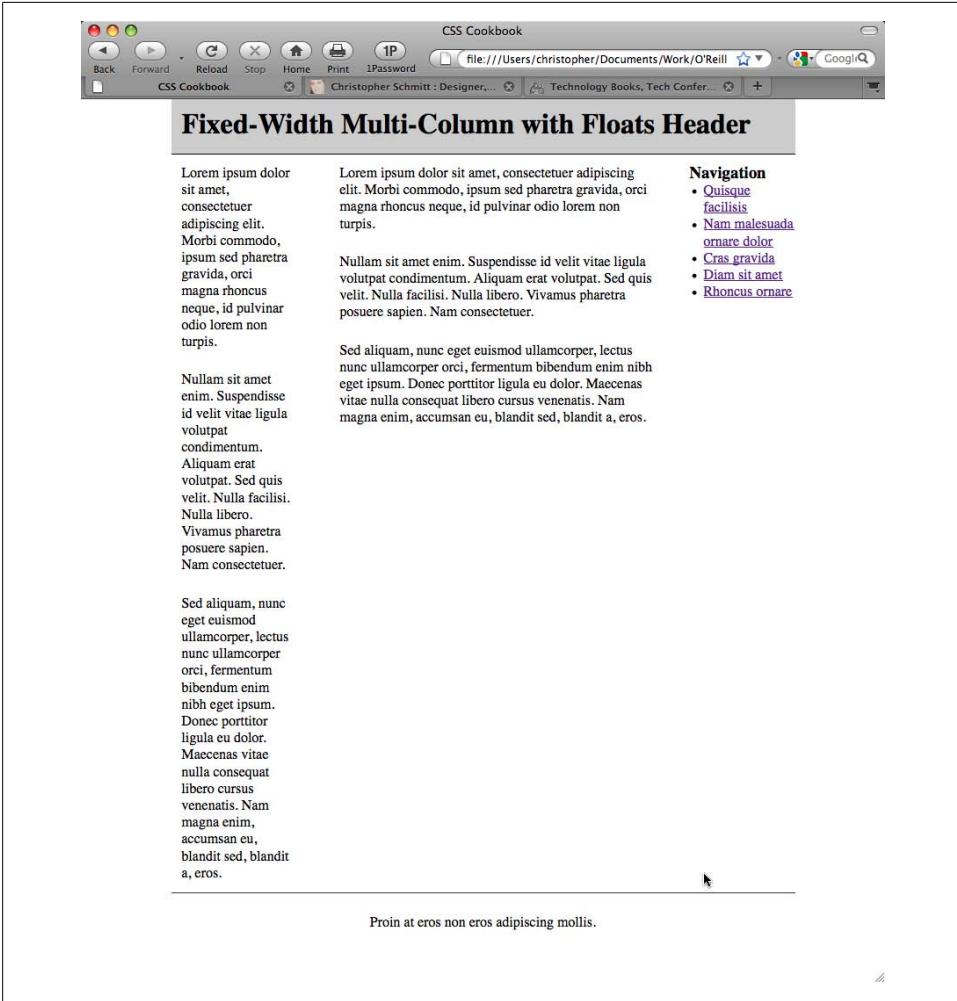
Figure 11-7. Three-column layout with fixed-width columns

and left columns as a whole are set to float to the left. And inside the "enclose" `div`, the main column is aligned to the right while the left column is aligned to the left.

See Also

for creating a three-column layout with flexible columns

11.6 Creating a Flexible Multicolumn Layout with Positioning

Problem

You want to create a four-column layout with columns that resize to the width of the browser, as shown in Figure 11-8.
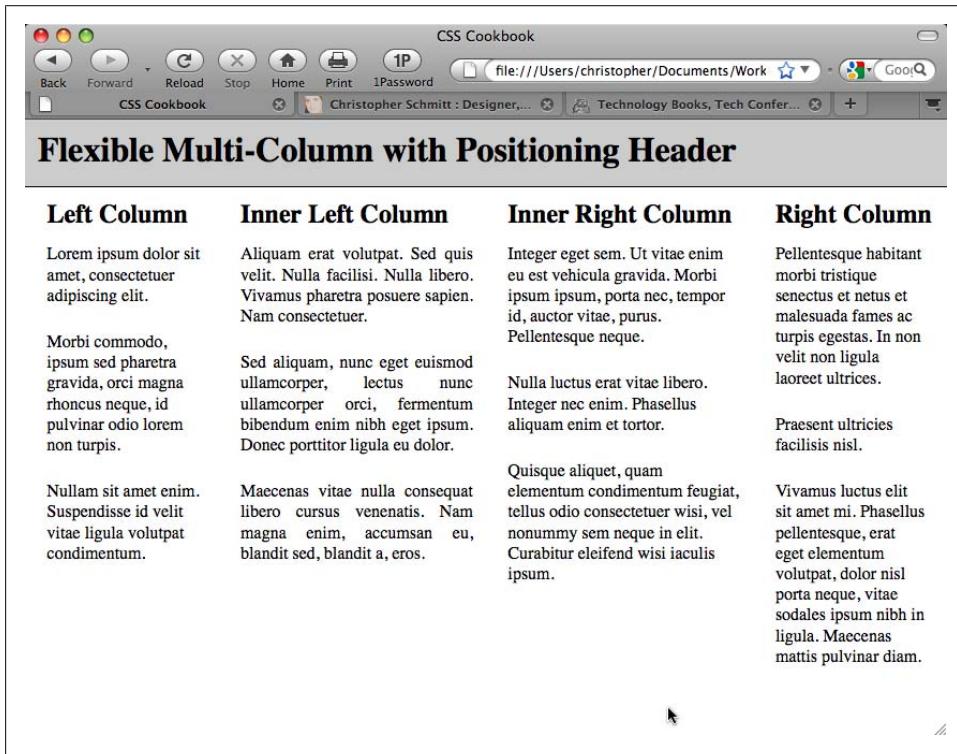
Figure 11-8. Four-column layout with percentage-based widths

Solution

First, mark up the content with `div` elements using the `id` attributes that contain appropriate values representing their placement on the page:

```
<div id="header">
 [...]
</div>
<div id="columnLeft">
 [...]
</div>
<div id="columnInnerLeft">
 [...]
</div>
```

```
[...]
<div id="columnInnerRight">
  [...]
</div>
[...]
<div id="columnRight">
  [...]
</div>
```

Next, use the `position` property in each column, setting the value to `absolute` while setting the placement of the columns with the `left` and `top` properties:

```
#columnLeft {
 position: absolute;
 left:1%;
 width:20%;
 top: 4em;
 background:#fff;
}
#columnInnerLeft {
 position: absolute;
 left: 22%;
 width: 28%;
 top: 4em;
 background: #fff;
 text-align: justify;
 border-width: 0;
}
#columnInnerRight {
 position: absolute;
 left: 51%;
 width: 28%;
 top: 4em;
 background: #fff;
}
#columnRight {
 position: absolute;
 left: 80%;
 width: 19%;
 top: 4em;
 background: #fff;
}
```

Discussion

By setting the `position` property to `absolute`, you take the element completely out of the flow of the document. When an element is set to `float`, other elements in a page can flow around the "floated" element. When an element is set to `absolute`, that element is treated like a ghost by the rest of the elements in the page that are in the normal flow.

The default rendering of an element when positioned absolutely is to the upper-left corner of its closest positioned ancestor or the initial containing block. (In other words, to position a child element set to `absolute` within the parent element, first apply a `position` property and value to its parent element.) If other elements are on the page, this creates an overlap of the content, as shown in Figure 11-9.
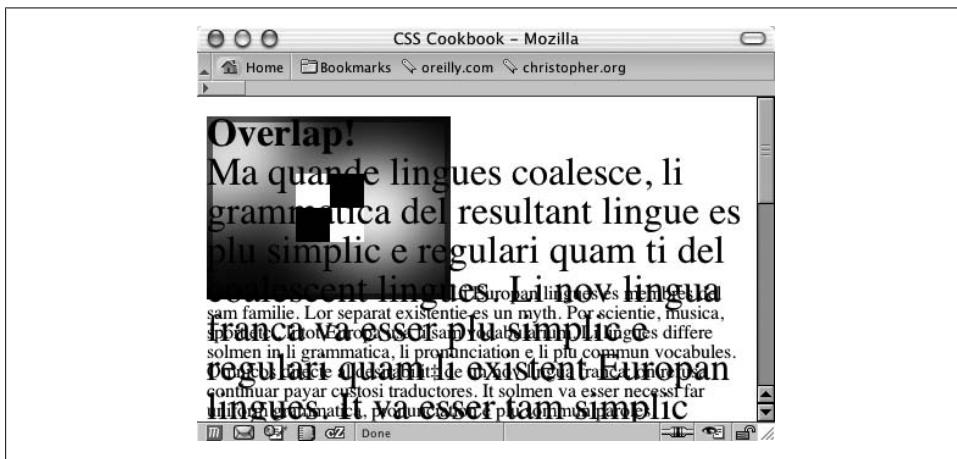
Figure 11-9. Text overlapping an image and other text in a web document

To avoid this problem, use four additional CSS properties that allow the element to be moved into any location: `top`, `left`, `bottom`, and `right`. Be sure to set these *offset* properties of the columns to percentage-based values to maintain flexible widths as a user's browser resizes.

Also use percentages as the values for the `left` property to mark the distance away from the left side of a browser's viewport. However, use em units as the values for the `top` property to compensate for the height of the heading. If you want to use an image for the heading, change the values for `top` to pixels, making sure there is enough room for the graphic header.

Known issues

Although this technique grants freedom in the placement of elements, there are drawbacks to using `absolute` to position elements. For example, if text is resized in the browser, the text might overlap other elements or text, hindering the legibility of the web page.

In addition, although this technique makes it easy to place columns next to each other, placing a footer section at the bottom of the columns is hard to do unless you know exactly where the columns end at the bottom of the page.

See Also

The CSS 2.1 specification on `position` at *http://www.w3.org/TR/CSS21/visuren.html#propdef-position*; the CSS 2.1 specification on positioning elements set to `absolute` at *http://www.w3.org/TR/CSS21/visuren.html#position-props*; more about containing blocks at *http://www.w3.org/TR/2003/WD-CSS21-20030915/visudet.html#containing-block-details*

11.7 Creating a Fixed-Width Multicolumn Layout with Positioning

Problem

You want to create a four-column layout with fixed-width columns.

Solution

First, mark up the content with `div` elements using the `id` attributes that contain appropriate values representing their placement on the page:

```
<div id="header">
 [...]
</div>
<div id="columnLeft">
 [...]
</div>
<div id="columnInnerLeft">
 [...]
</div>
 [...]
<div id="columnInnerRight">
  [...]
</div>
 [...]
<div id="columnRight">
 [...]
</div>
```

Next, use the `position` property in each column, setting the value to `absolute` while setting the placement of the columns with the `left` and `top` properties, making sure to use pixels for the units:

```
#columnLeft {
 position: absolute;
 left:5px;
 width:190px;
 top: 44px;
 background:#fff;
}
#columnInnerLeft {
```

```
  position: absolute;
  left: 205px;
  width: 190px;
  top: 44px;
  background: #fff;
  text-align: justify;
  border-width: 0;
}
#columnInnerRight {
  position: absolute;
  left: 405px;
  width: 190px;
  top: 44px;
  background: #fff;
}
#columnRight {
  position: absolute;
  left: 605px;
  width: 190px;
  top: 44px;
  background: #fff;
}
```

Discussion

Setting the width of the columns as well as the `left` and `top` properties to length units creates the fixed-width columns. This Solution is just as easy with two or three columns. Remember that anything more than four or five columns might be impractical.

> Although it is possible to use absolute positioning, it's far better to use floats as a layout option, as floats allow for more flexible web designs that keep your page from breaking.

See Also

Recipe 11.3 for creating a fixed-width two-column layout; Recipe 11.5 for creating a fixed-width multicolumn layout with floats

11.8 Using Floats to Display Columns in Any Order

Problem

You want to develop a system to display content in columns in any order.

Solution

Given the following markup:

```
<div id="container-outer">
 <div id="container">
  <div id="content" class="column">
   <div class="wrap">
    [...]
   </div>
  </div><!-- /END #content -->

  <div id="navigation" class="column">
   <div class="wrap">
    [...]
   </div>
  </div><!-- /END #navigation -->

  <div id="related-info" class="column">
   <div class="wrap">
    [...]
   </div>
  </div><!-- /END #related-info -->
 </div><!-- /END #container -->
</div><!-- /END #container-outer -->
```

apply the following CSS rules:

```
.column {
 float: left;
}

#content {
 margin-left: 20%;
 width: 60%;
}

#navigation {
 margin-left: -80%;
 width: 20%;
}

#related-info {
 width: 19%;
}

/* IEx patches \*/
* html .column {
 display: inline;
}

* html #navigation li {
 height: 1%;
}
/**/
```

This will yield the basic page layout shown in Figure 11-10, with two narrow, flexible-width sidebars bounding an equally flexible center column.
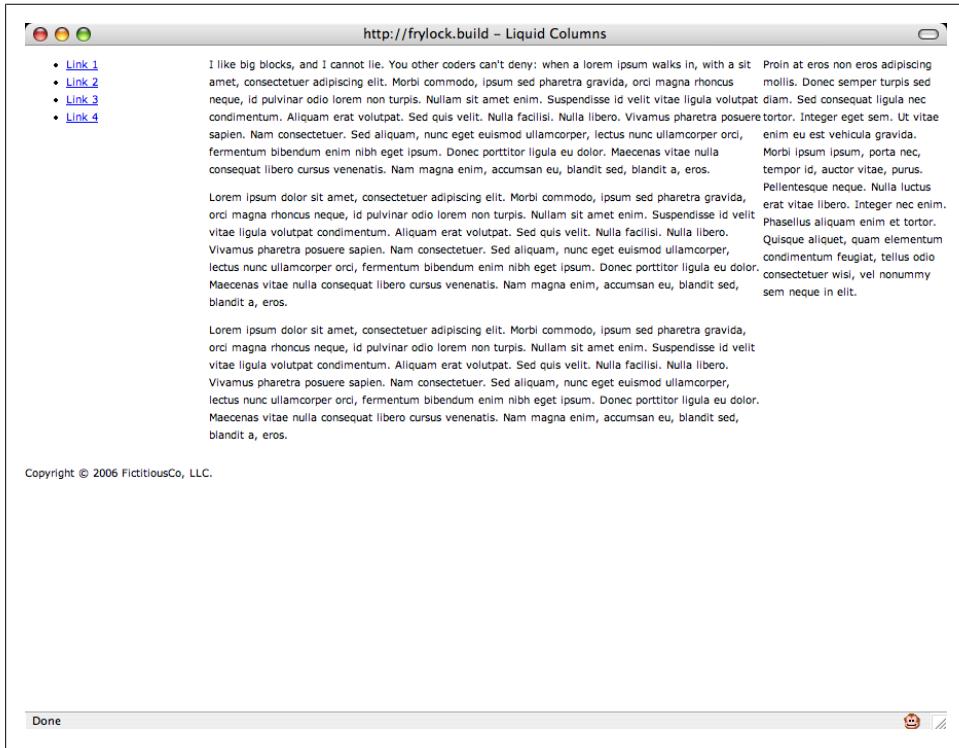
Figure 11-10. Basic formatting of page layout

From this rather bland foundation, you can layer additional CSS on top. Adding the following code to your CSS will yield a design similar to Figure 11-11:

```
body {
 font: normal 62.5%/1.7 Verdana, Geneva, Helvetica, Arial, sans-serif;
 margin: 0;
 padding: 0;
}
#container:after {
 clear: both;
 content: ".";
 display: block;
 height: 0;
 visibility: hidden;
}
#container {
 display: inline-block;
}
#container-outer {
```

```
  background: url("bg-left.gif") repeat-y 20% 0;
}
#container {
  background: url("bg-right.gif") repeat-y 80% 0;
}
.column .wrap {
  padding: 20px;
}
#content .wrap {
  padding: 20px 30px;
}
#content p {
  margin-top: 0;
}
#content p:first-child {
  font: normal 1.4em/1.6 Georgia, Times, "Times New Roman", serif;
}
#content p:first-child:first-line {
  text-transform: uppercase;
}
#navigation ul, #navigation ul li {
  list-style: none;
  margin: 0;
  padding: 0;
}
#navigation ul li {
  margin-bottom: .4em;
}
#navigation li a {
  background: #36C;
  color: #FFF;
  border-left: 7px solid #09F;
  display: block;
  padding: .4em .4em .4em 20px;
  text-decoration: none;
}
#navigation li a:hover {
  border-left: none;
  border-right: 7px solid #09F;
  padding-left: 27px;
}
#related-info {
  color: #555;
  font-style: italic;
}
#copyright {
  border: 1px solid #B2B2B2;
  border-width: 1px 0;
  clear: both;
  padding: 10px 20px;
  text-align: center;
}
#copyright p {
  margin: 0;
}
```
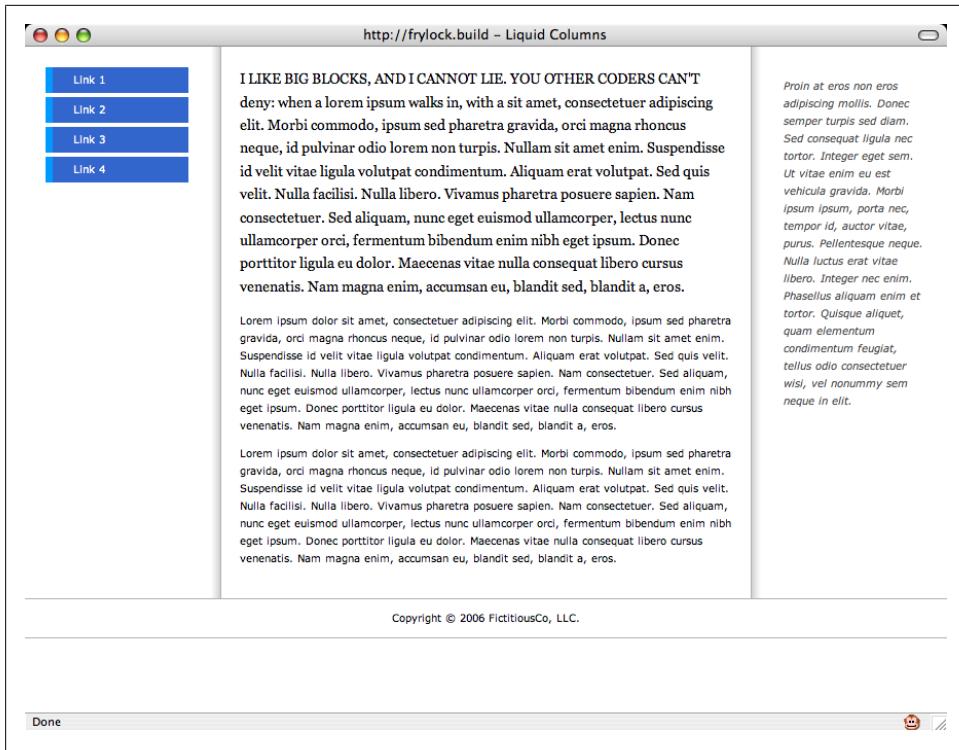
Figure 11-11. Fleshed-out design of multicolumn layout

Discussion

The authors of the CSS specification never intended floats to be used for page-level layout control: rather, they were a means to control the flow of content around an object, much as `align="left"` or `align="right"` would cause text to wrap around an `img` element. But despite the specification's original spirit, floats do offer a powerful and flexible alternative to traditional, table-based layout techniques.

Alex Robinson, a designer, published an influential article on creating the "Any Order Columns" in CSS (*http://www.positioniseverything.net/articles/onetruelayout/*). Robinson's technique allows developers to create multicolumn layouts easily by using floats to display each column in any order, regardless of the order in which those blocks appear in the markup.

The markup

To work with this technique, you first need to establish columns in your markup, like so:

```
<div id="container">
 <div id="content" class="column">
```

```
  [...]
  </div><!-- /END #content -->

  <div id="navigation" class="column">
  [...]
  </div><!-- /END #navigation -->

  <div id="related-info" class="column">
  [...]
  </div><!-- /END #related-info -->
</div><!-- /END #container -->

<div id="copyright">
  <p>Copyright notice goes here.</p>
</div>
```

Inside each `div`, place any markup you would like. Figure 11-12 shows what the un-styled document looks like, with a few paragraphs and an unordered list thrown in for good measure.
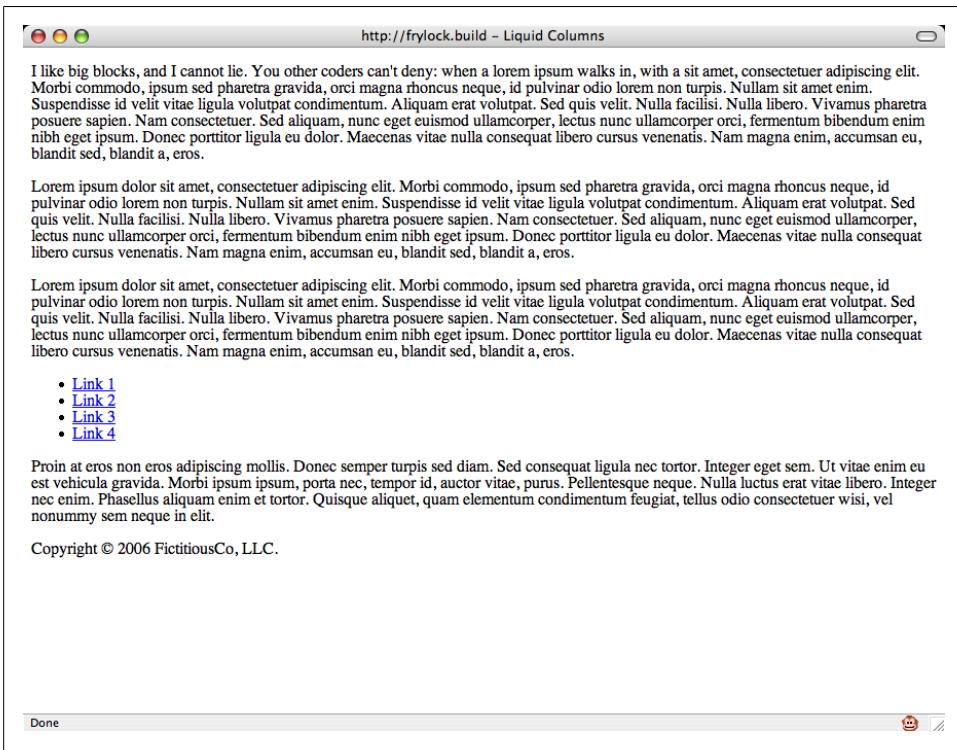
Figure 11-12. Unstyled page layout

From the demonstration so far, you set up a `div` element for each of your three columns and assigned each an `id` that describes the kind of content that will be placed inside. In this Solution, the values for `id` are `content`, `navigation`, and `related-info`.

It would have been just as easy to use `center`, `left`, and `right`, but that wouldn't have been especially forward thinking: what happens when you change your site's CSS file, and the new design requires the "left" `div` to appear on the righthand side of the page?

Defining the columns

With this simple markup structure in place, apply a generic float rule to all three "column" `div`s:

```
.column {
 float: left;
}
```

As shown in Figure 11-13, the layout does not look drastically different. The copyright text is a bit out of alignment, but the bulk of the page appears as it did before, with each "column" `div` stacking horizontally. Once you assign dimensions to these blocks, however, things will rapidly change.
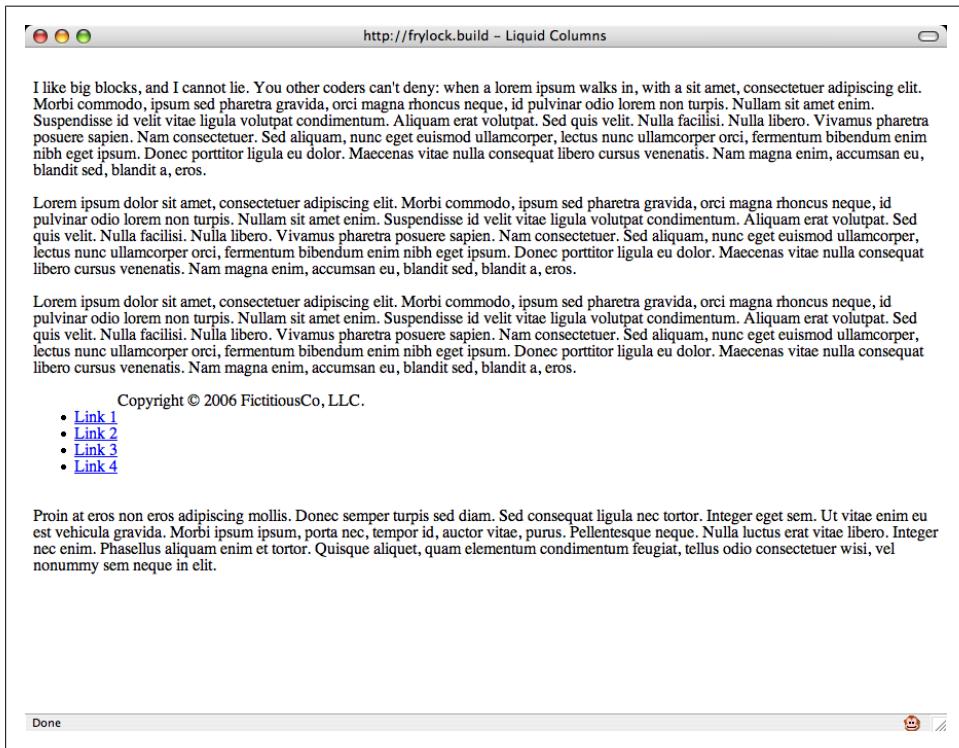
Figure 11-13. The moved copyright notice

First, start with the content block. To set the block to be 60% of the window width and the width of the lefthand sidebar to be 20% of the screen, create the following rule:

```
#content {
  margin-left: 20%;
  width: 60%;
}
```

Figure 11-14 shows that the layout is looking a bit odd, but is starting to take shape.
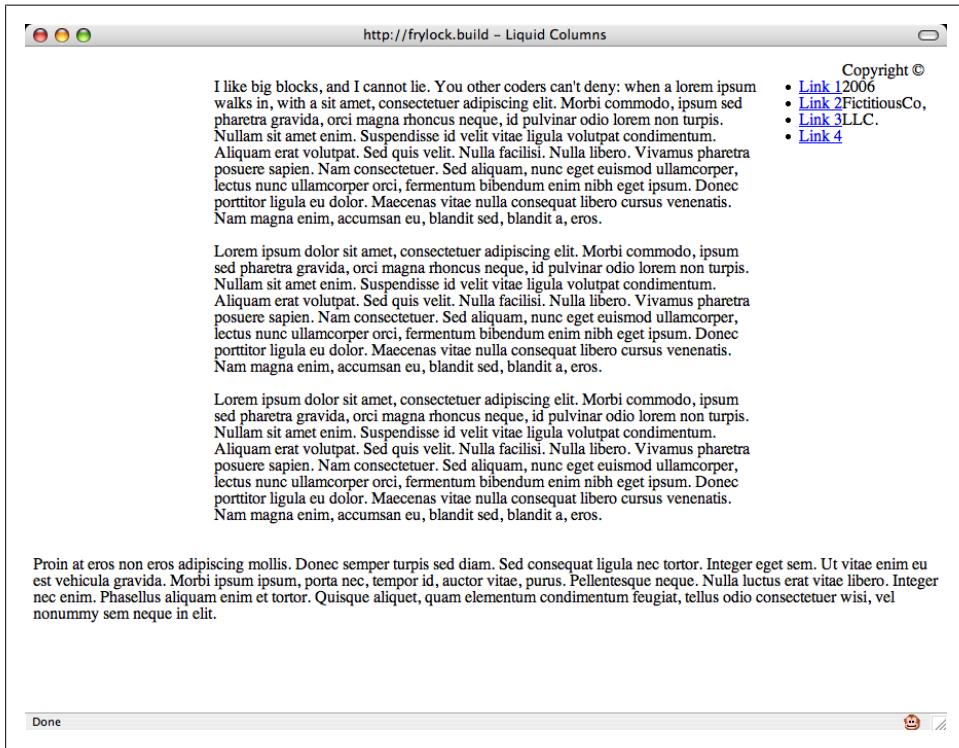
Figure 11-14. Applying styles to the content portion of the layout

By setting a lefthand margin equal to the width of the lefthand sidebar, you've essentially "reserved" some space for it. The next step is to use negative margins to "pull" the navigation `div` across the content `div` to the lefthand side of the page:

```
#navigation {
  margin-left: -80%;
  width: 20%;
}
```

The `margin-left` value applied is a sum of the width of the center column (60%) and its lefthand margin (20%). This pulls the navigation column over to its proper place, as shown in Figure 11-15.
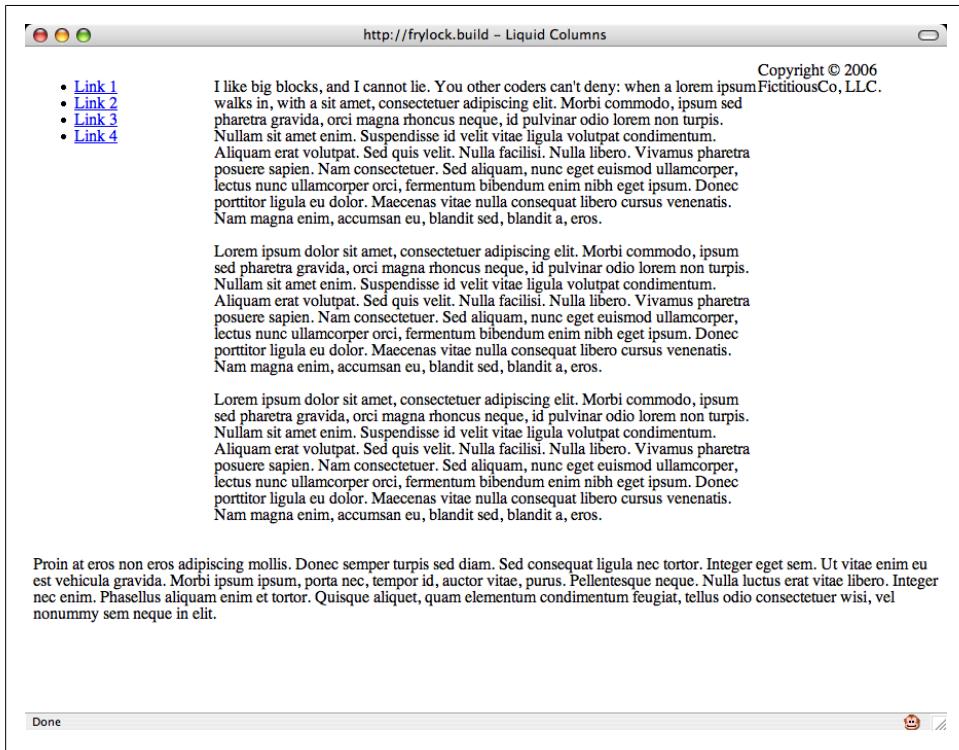
Figure 11-15. The navigation, moved to the left column

Now, simply by setting a width on the `related-info` block, the three-column layout is complete, as shown in Figure 11-16:

```
#related-info {
 width: 20%;
}
```

This looks excellent, though the "copyright" `div` is still a bit off. But it's easy to fix that with the `clear` property, as shown in Figure 11-17:

```
#copyright {
 clear: both;
}
```
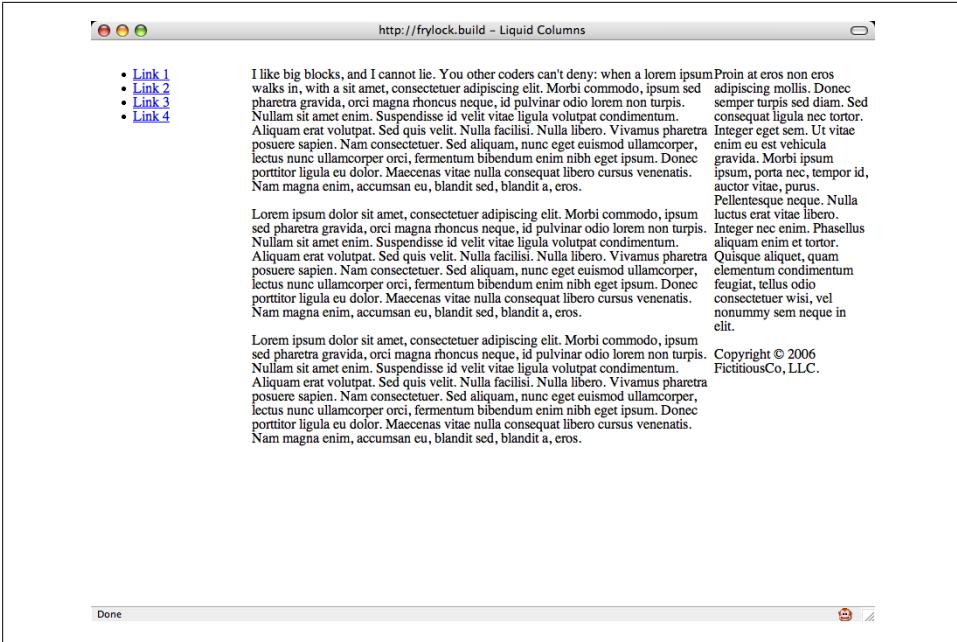
Figure 11-16. Moving the right column content into place
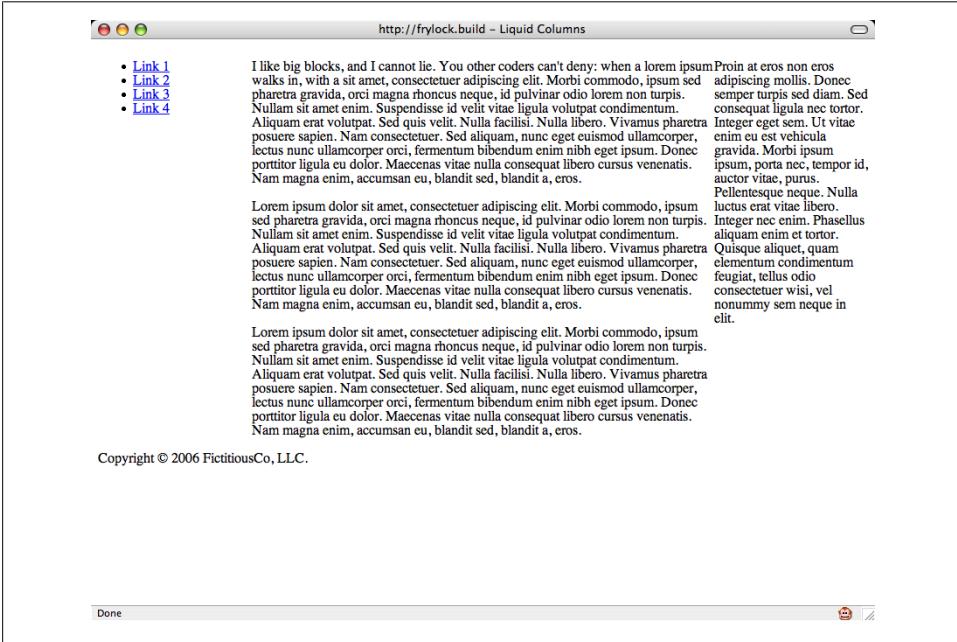
Figure 11-17. Placing the copyright notice at the bottom of the page

Although the layout might look as though the columns are nearly complete, Figure 11-18 shows that IE needs a little extra attention.
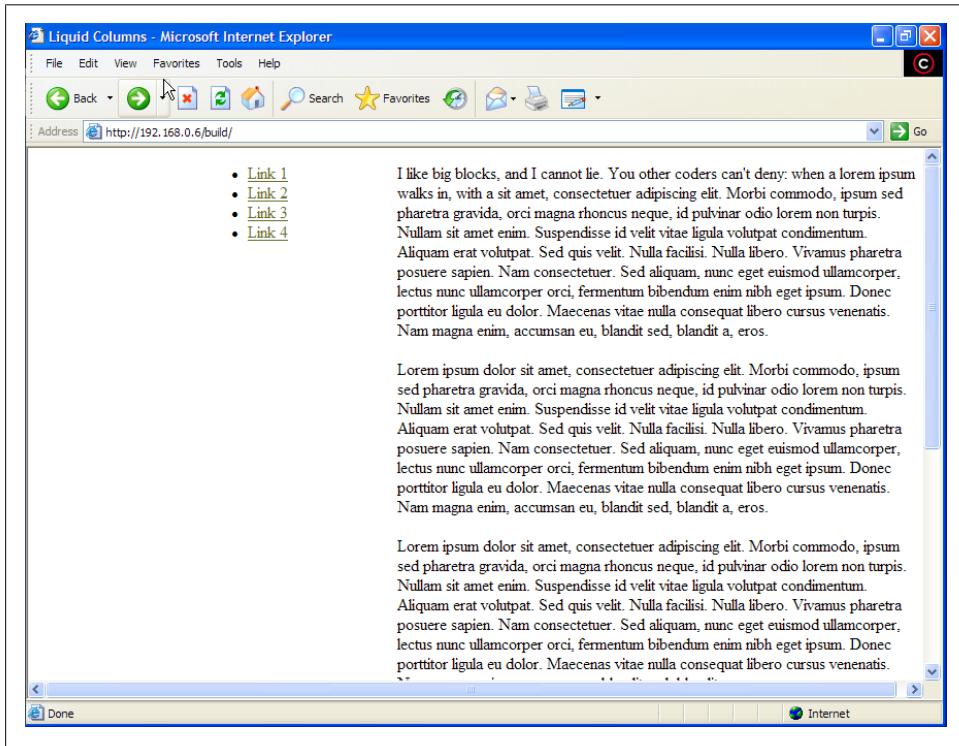
Figure 11-18. Problems with the layout viewed in Internet Explorer for Windows

This is the result of a documented IE bug known as the "Doubled Float-Margin Bug" (*http://positioniseverything.net/explorer/doubled-margin.html*): essentially, when a margin is applied to a floated box in the same direction as the float, that margin is doubled in size.

Since the lefthand margin is applied to a left-floated element, IE takes that 20% margin and doubles it to 40%.

Thankfully, the fix is simple. When you apply `display: inline` to the problematic element, Internet Explorer behaves again. To do this, add the following lines to your CSS:

```
/* IEx patches \*/
* html .column {
 display: inline;
}
/**/
```

The oddly formatted comments and `*` `html` prefix ensure that earlier versions of IE can see this code. And as Figure 11-19 shows, IE is behaving properly.
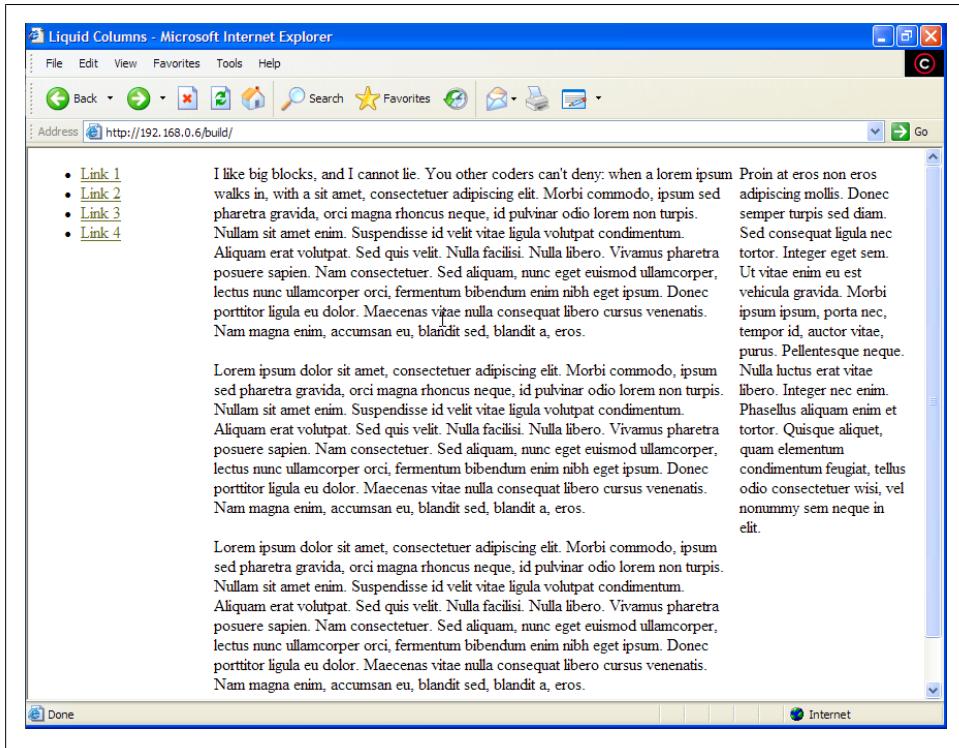
Figure 11-19. The fix applied, and the layout working in Internet Explorer for Windows

The result is a flexible, three-column layout template. But where else can you take this?

Creating whitespace

The space between the columns is called a *gutter*. To customize this layout by increasing the size of the gutters, you can apply some margins around the columns. There are a number of ways to achieve this effect, but start by adding an additional `div` to each column:

```
<div id="container">
 <div id="content" class="column">
  <div class="wrap">
   [...]
  </div>
 </div><!-- /end #content -->

 <div id="navigation" class="column">
  <div class="wrap">
   [...]
```

```
    </div>
  </div><!-- /end #navigation -->

  <div id="related-info" class="column">
    <div class="wrap">
     [...]
    </div>
  </div><!-- /end #related-info -->
</div><!-- /end #container -->
```

With the "wrap" `div`s in place (which we will get back to later in the Discussion), apply padding to them with CSS to create more breathing room, as shown in Figure 11-20:

```
.column .wrap {
 padding: 20px;
}

#content .wrap {
 padding: 20px 30px;
}
```
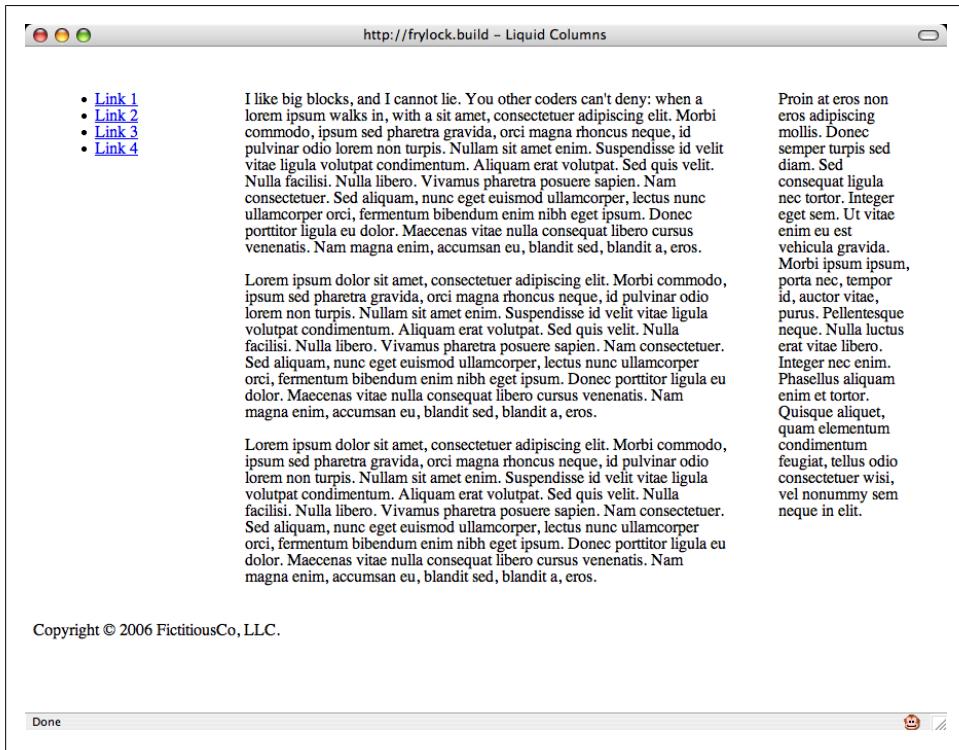

Figure 11-20. Increasing the size of the gutters

Adjusting the order of columns

As you may have noticed by now, the "Any Order Columns" method is grounded in the intelligent use of margins: positive margins are used to reserve space, and negative margins are used to "pull" columns out of their natural position.

Simplify the CSS for a moment, and remove all of the column margins:

```
#content {
 width: 60%;
}
#navigation {
 width: 20%;
}
#related-info {
 width: 19%;
}
```

As a result, the layout now looks like Figure 11-21, with each column appearing in its natural position in the float order.
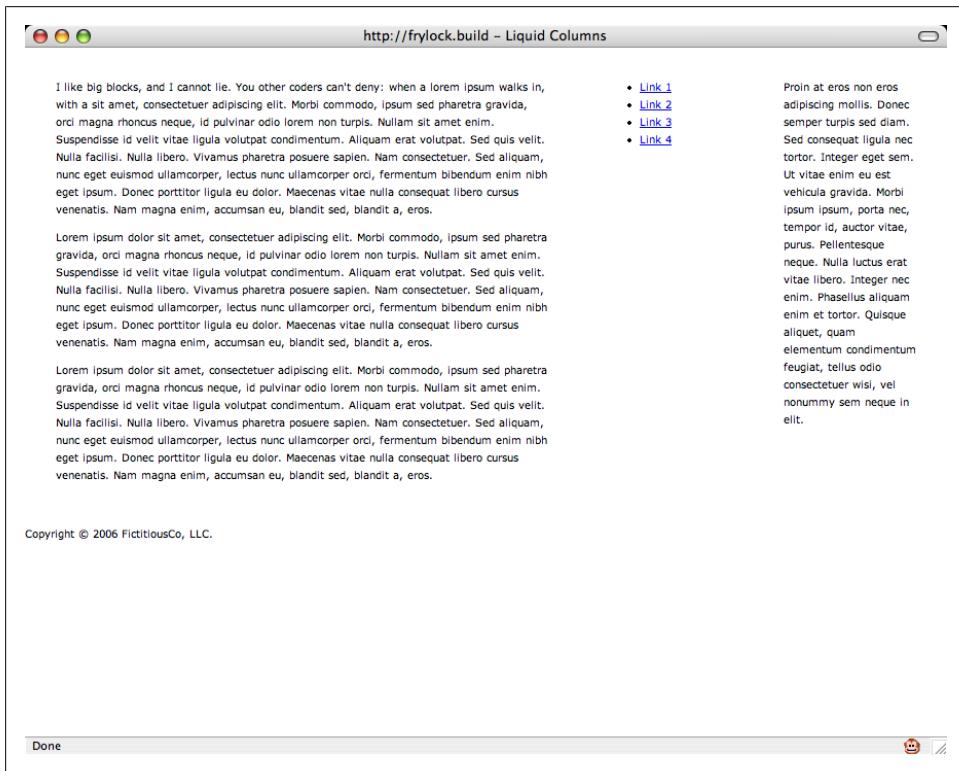
Figure 11-21. Moving the navigation between the columns

By adding a lefthand margin to the "navigation" `div`, and then using a negative lefthand margin to move the "related-info" `div`, you can essentially reverse the order of the second two columns. With the following CSS, you're left with a layout similar to Figure 11-22:

```
#content {
  width: 60%;
}
#navigation {
  margin-left: 20%;
  width: 20%;
}
#related-info {
  margin-left: -39%;
  width: 19%;
}
```
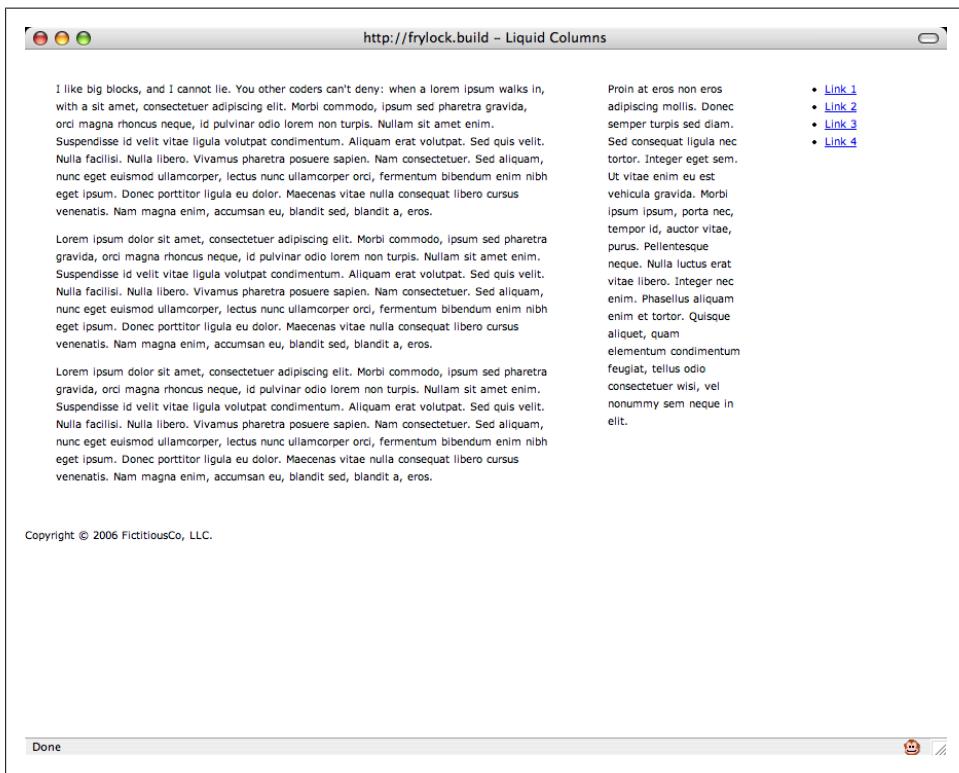

Figure 11-22. Reversing the order of the columns

To complete the demonstration, place the content column on the righthand side of the page, as shown in Figure 11-23, by applying the following code:

```
#content {
  margin-left: 40%;
```

```
  width: 60%;
}
#navigation {
 margin-left: -100%;
 width: 20%;
}
#related-info {
 margin-left: -80%;
 width: 19%;
}
```
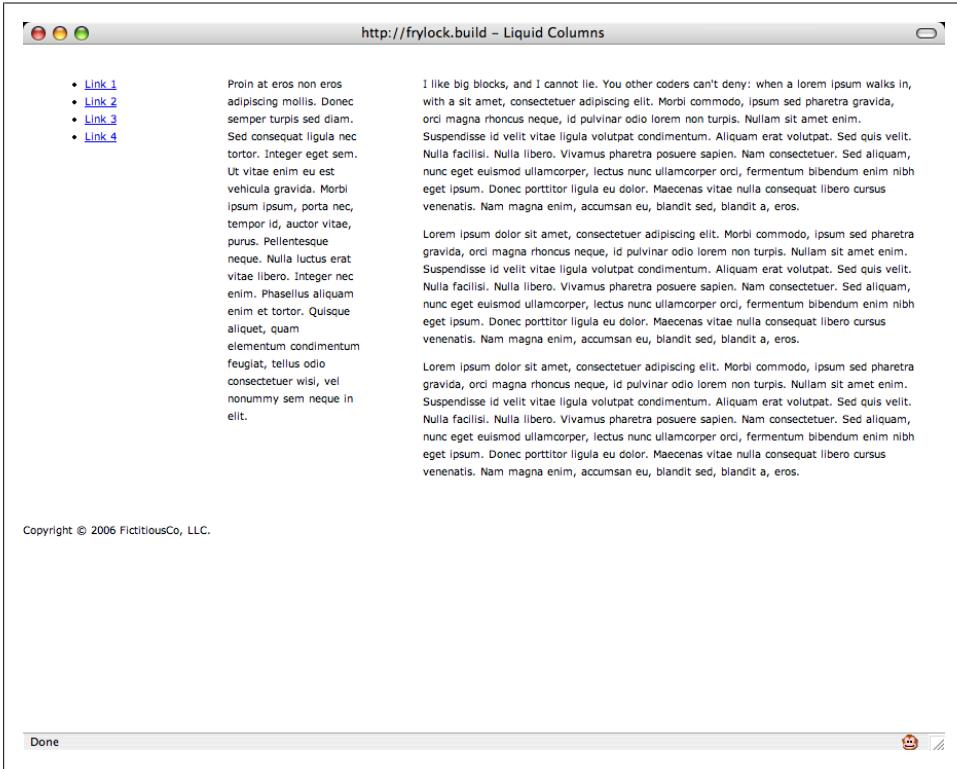

Figure 11-23. Content column moved to the righthand side of the page

As with the first layout, you applied a margin to the content column to "reserve" some whitespace on the lefthand side of the page. Then, you used negative lefthand margins to pull the navigation and "related information" divs into the proper location.

Faking columns

Now we'll return to the first layout, as shown in , and see how to make the columns feel a bit more polished. The first step: background images.
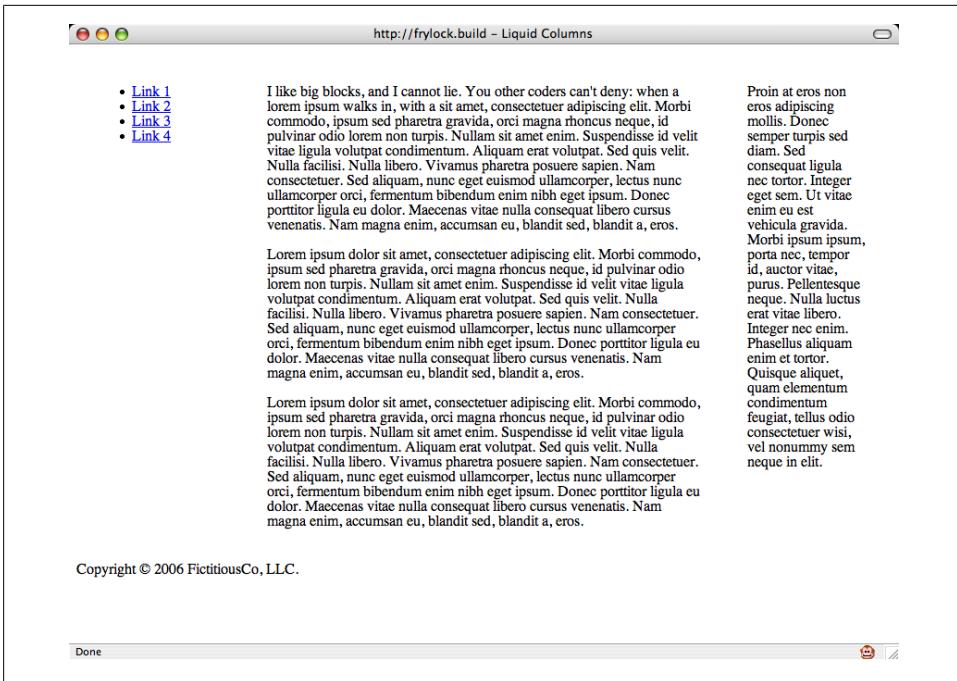
Figure 11-24. Initial layout awaiting column graphics

"Faux columns" is a technique developed by web designer Dan Cederholm (*http://alistapart.com/articles/fauxcolumns/*) that utilizes a horizontally repeating background image.

By using one tiled image, Cederholm's method works incredibly well in a fixed-width design; however, the technique's versatility means that it needs only slight modification to work in a fully flexible layout.

First, you need two images, one for each side of the content column. Figure 11-25 shows the lefthand graphic, and Figure 11-26 shows the righthand graphic.

Figure 11-25. Graphic for lefthand column

Figure 11-26. Graphic for righthand column

Next, wrap the container block in an extra `div`:

```
<div id="container-outer">
 <div id="container">
  [Rest of template goes here]
 </div>
</div>
```

And finally, add the following rules to your stylesheet:

```
#container:after {
 clear: both;
 content: ".";
 display: block;
 height: 0;
 visibility: hidden;
}
#container {
 display: inline-block;
}
/*\*/
#container {
 display: block;
}
/**/
/*\*//*/
#container {
 display: inline-block;
}
/**/
#container-outer {
 background: url("bg-left.gif") repeat-y 20% 0;
}
#container {
 background: url("bg-right.gif") repeat-y 80% 0;
}
```

> Also, you could use the `overflow` property set to `hidden`, as discussed in Recipe 2.22.

With this code in place, the columns appear as full-length columns, as shown in Figure 11-27. From here, feel free to add any typographic styles you'd like; the ones supplied in the Solution will do nicely, and will yield the finished design shown in Figure 11-28.
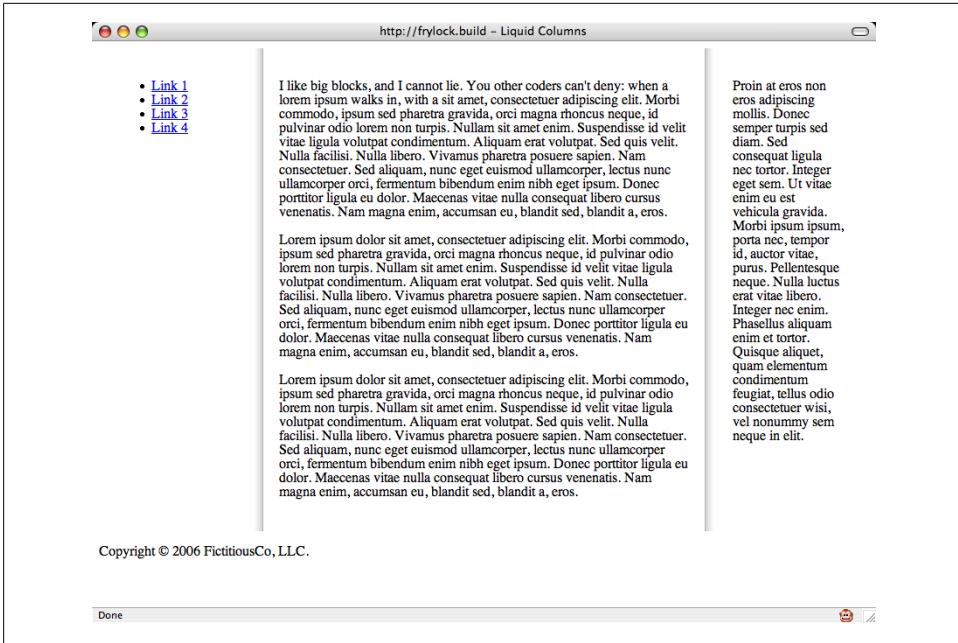
Figure 11-27. Column graphics applied to layout
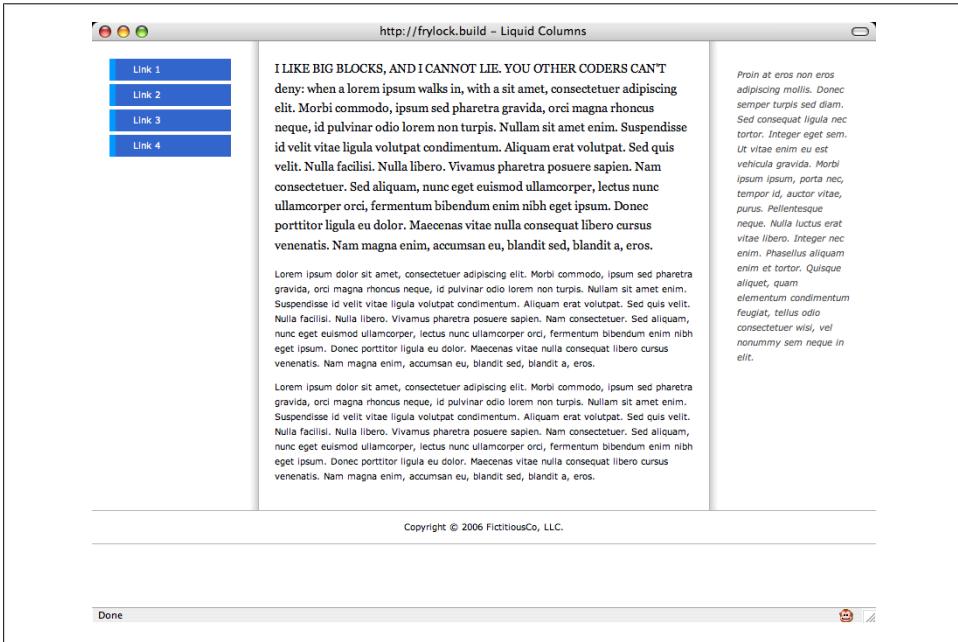
Figure 11-28. Finalized page layout

An alternative solution

The float model for laying out pages is powerful, but floats can have a rather steep learning curve. As a result, many designers find absolute positioning to be an attractive alternative, enabling them to precisely position the different components of their design with *x* and *y* coordinates.

Unfortunately, positioned elements are taken "out of the document flow," which effectively collapses their containing element. As a result, "positioned" designs lack the powerful float concept of clearing, which enables the different parts of a design to be "context aware": that is, a "footer" `div` (such as the copyright block in the Solution) can be cleared of the floated blocks above it, but not of any positioned elements on the page.

Shaun Inman, a talented web designer/developer, has written a lean JavaScript function to fix this problem (*http://shauninman.com/plete/2006/05/clearance-position-inline-absolute.php*). When inserted into your web pages, Inman's script will automatically "clear" elements of any other positioned elements on the page, as shown in Figure 11-29.
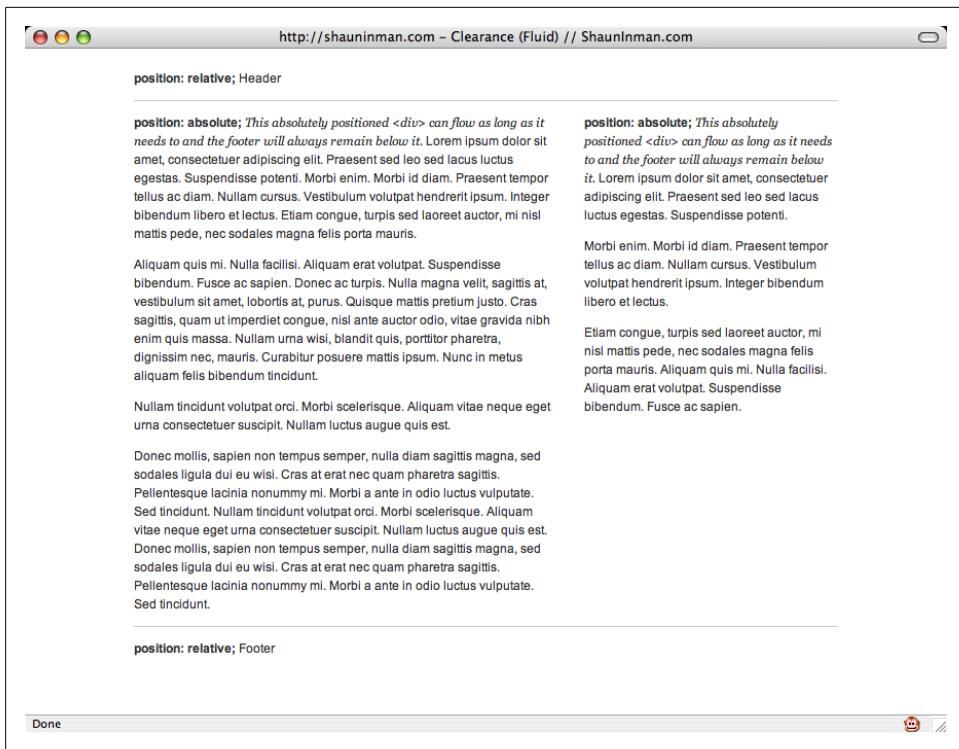
Figure 11-29. Using absolute positioning for a layout

The only potential drawback to this method is that it does rely on JavaScript being active in the user's browser. But if the content is accessible if you disable JavaScript in your target browsers during testing, all should be well.

See Also

Recipe 11.9 for designing an asymmetric layout with absolute positioning

11.9 Designing an Asymmetric Layout

Problem

You want to create a flexible asymmetric or organic layout, as shown in Figure 11-30.
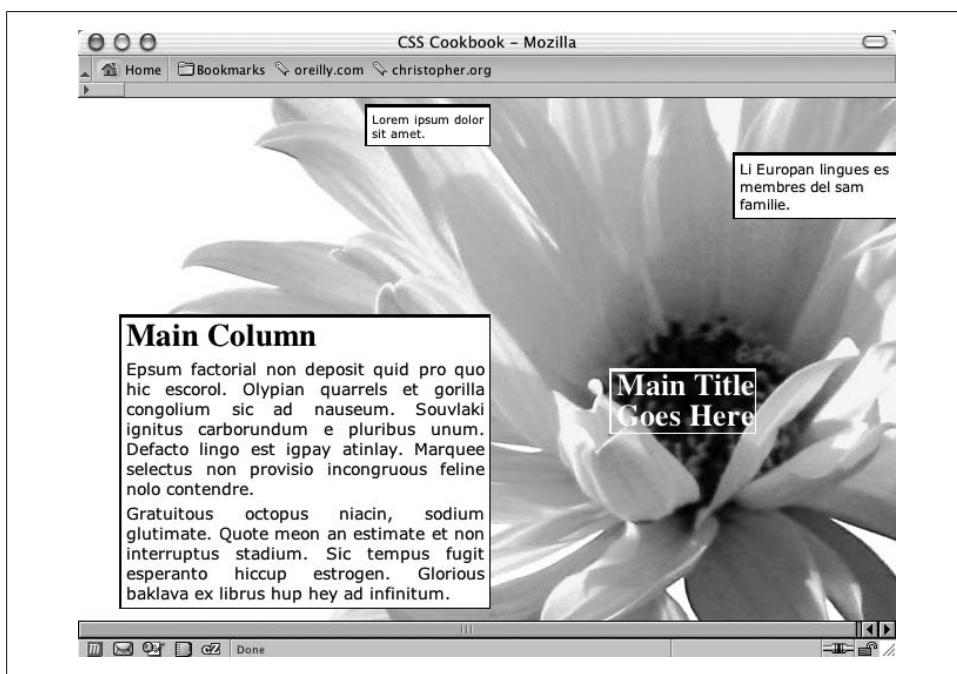
Figure 11-30. Asymmetric placement of the content

Solution

First, mark up the content with `div` elements using the `id` attributes that contain appropriate values representing their placement on the page:

```
<div id="header">
 [...]
</div>
<div id="columnSmall">
```

```
  [...]
</div>
<div id="columnMain">
  [...]
</div>
<div id="columnMedium">
  [...]
</div>
```

Next, use the `position` property in each column, setting the value to `absolute` while setting the placement of the columns with the `left` and `top` properties using percentages. Also, use percentage values for positioning a background image:

```
body {
 margin:5px 0 0 5px;
 background-image: url(flower5.jpg);
 background-position: 50% 35%;
 background-repeat: no-repeat;
 }
#header {
 position: absolute;
 left: 65%;
 top: 50%;
 width: 125px;
 font-size: small;
}
#columnSmall {
 position: absolute;
 left: 35%;
 width: 15%;
 top: 1%;
 background: #fff;
font-size: small;
}
#columnMain {
 position: absolute;
 left: 5%;
 width: 45%;
 top: 40%;
 background: #fff;
 text-align: justify;
 border-width: 0;
 font-size: large;
}
#columnMedium {
 position: absolute;
 left: 80%;
 width: 20%;
 top: 10%;
 background: #fff;
}
```

Discussion

Although websites seem to use traditional column layouts, CSS enables web developers to come up with new ways to present their documents. Through the `position`, `top`, and `left` properties, you can break up the content into chunks, stylize them separately, and place them in unique arrangements.

The background image moves with the content if the browser window is resized, because you used a percentage value to set the position of the background image.

Instead of changing the values for the `position`, `top`, and `left` properties by hand, you can more easily place `div` elements with a WYSIWYG application such as Adobe Dreamweaver.

If you want to create an asymmetric or organic layout with fixed-width columns instead of making this layout resizable, use length units to dictate the exact position of both the content and the background image:

```
body {
  margin:5px 0 0 5px;
  background-image: url(flower5.jpg);
  background-position: -400px -200px;
  background-repeat: no-repeat;
}
#header {
  position: absolute;
  left: 500px;
  top: 200px;
  width: 125px;
  font-size: small;
}
#columnLeft {
  position: absolute;
  left: 200px;
  width: 125px;
  top: 10px;
  background:#fff;
  font-size: small;
}
#columnInnerLeft {
  position: absolute;
  left: 50px;
  width: 375px;
  top: 175px;
  background: #fff;
  text-align: justify;
  border-width: 0;
  font-size: large;
}
#columnInnerRight {
  position: absolute;
  left: 600px;
  width: 150px;
  top: 50px;
```

```
  background: #fff;
}
```

See Also

Recipes 4.5 and 4.6 for setting background images on a web page; *http://www.dream weaver.com* for more information on Adobe Dreamweaver

11.10 Designing Resolution-Independent Layouts

Problem

You want to build a web page that adapts to the resolution of the browser.

Solution

Use web designer Cameron Adams's Resolution JavaScript at *http://www.themaninblue .com/writing/perspective/2006/01/19/*, as shown in Figure 11-31.
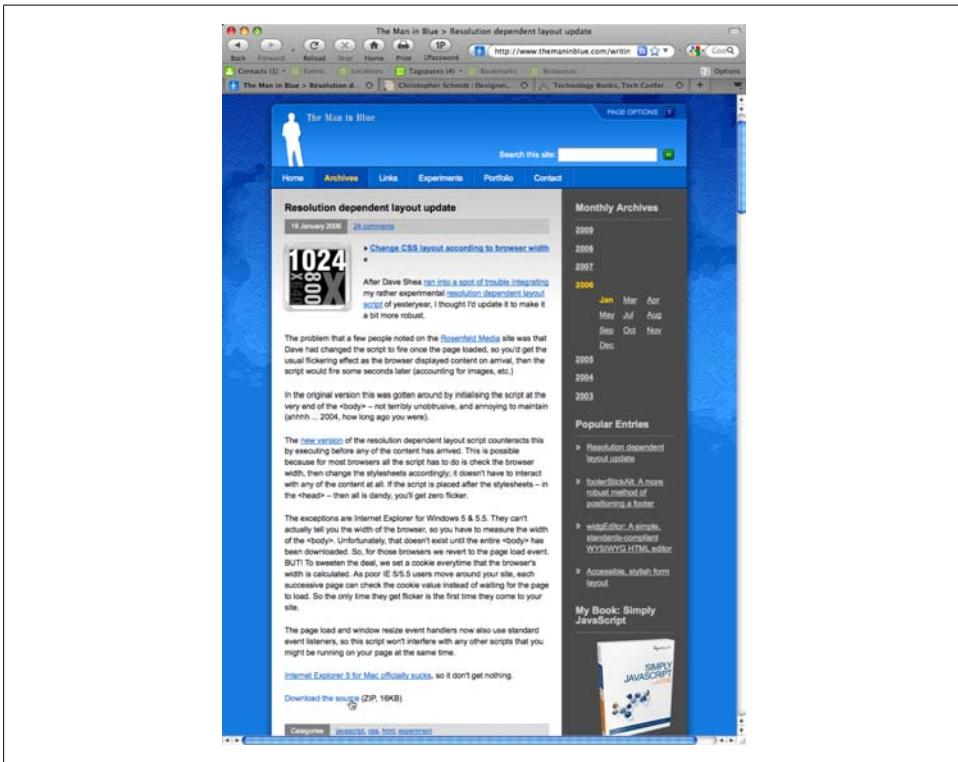
Figure 11-31. Cameron Adams's page about his script

The resolution solution relies on two stylesheets: *main.css* acts as a foundation with basic formatting of the page, and *features_1024.css* offers styling, as shown in Figure 11-32.
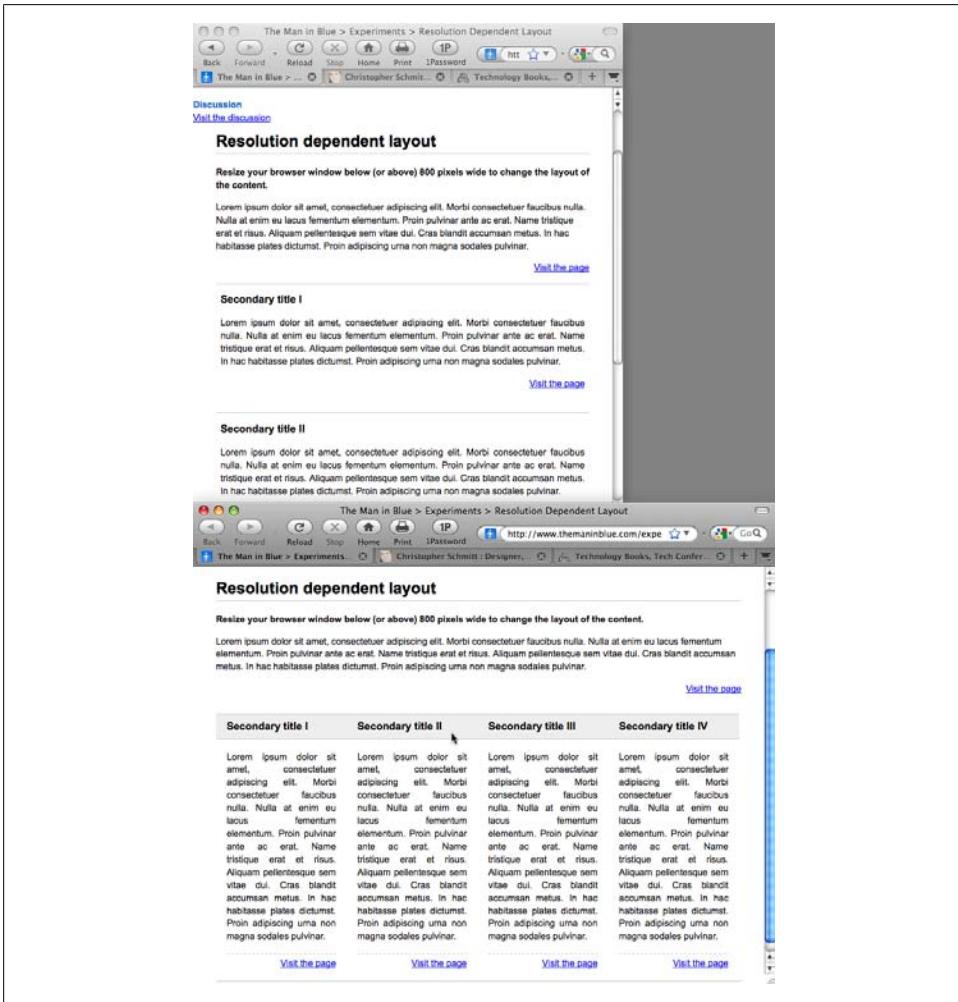
Figure 11-32. Different looks for the same content based on the size of the browser window

Discussion

When the browser is less than 1,024 pixels wide, only the foundation stylesheet is applied. When the browser resolution increases beyond 1,024 pixels, the advanced features are applied.

The Resolution script, although requiring some basic JavaScript knowledge, can be extended to include different stops in resolution support. For example, you can add additional styles at browser widths of 800, 1,024, 1,440, or even 1,660 pixels.

Resolution independence without JavaScript

A part of the CSS3 specification, media queries, adds a feature that allows you to adjust which CSS rules are applied based on the dimensions of the browser's viewport.

By appending additional conditions in the `media` attribute of the `link` attribute, you can deliver stylesheets based on the width of the viewport, as shown in Figure 11-33:

```
<link media="screen and (max-width: 300px)" rel="stylesheet" href="ssr.css"
type="text/css" />
<link media="screen and (min-width: 300px) and (max-width: 750px)"
rel="stylesheet" href="msr.css" type="text/css" />
<link media="screen and (min-width: 750px)" rel="stylesheet" href="lsr.css"
type="text/css" />
```
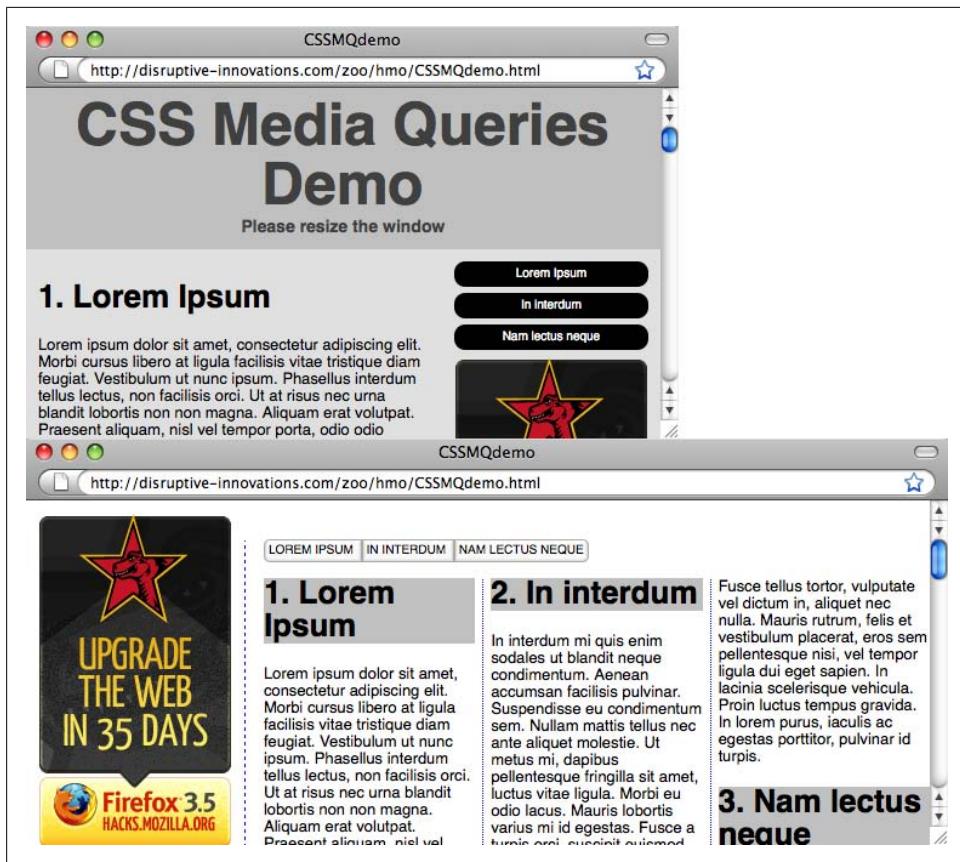

Figure 11-33. Different styles applied without the use of JavaScript

At the time of this writing, support for media queries is available in Firefox 3.5 and Opera 10.

See Also

The CSS3 specification for media queries at *http://www.w3.org/TR/css3-mediaqueries/*; Mozilla Developer Center information on media queries at *https://developer.mozilla.org/En/CSS/Media_queries*; Recipe 14.3 for delivering different stylesheets for mobile devices

Hacks, Workarounds, and Troubleshooting

12.0 Introduction

When designing for the Web, developers historically have used hacks and workarounds due to browser limitations.

The mid-1990s saw a proliferation of such workarounds, among them single-pixel GIFs, `font` tags, and nested tables, to name just a few. Although the CSS2 specification became a recommendation back in May 1998, only recently have browser vendors fully implemented the standard in their products. This gap in time of browsers without CSS support to browsers with full or near-perfect CSS implementation means a handful of the browsers that most people use has poor CSS support.

To overcome the bugs in these popular browsers that have this poor CSS support, web developers have once again resorted to using hacks and workarounds to successfully achieve web page designs.

Even though problems might be solved by using newer versions of browsers, web developers might need to use hacks or workarounds to deliver the appropriate presentation to their audience, for many reasons.

Unlike web developers, most people don't automatically upgrade their browsers each time a new one is available. They tend to stick with the browser that's on their computer because it works fine, and will get a new browser only when they purchase a new computer.

Also, IT departments in many companies lock down the systems and prevent individuals from upgrading software applications on their own.

For web developers struggling to polish their designs, this chapter covers techniques for dealing with browsers that have spotty CSS support. Included in this chapter are methods ranging from troubleshooting CSS rules to testing multiple browsers on one machine.

12.1 Overriding Inline Styles

Problem

You want to change the inline styles from an external stylesheet:

```
<div style="font-size: 100px;">
 Epsum factorial non deposit quid pro quo hic escorol.
</div>
```

Solution

In an external stylesheet, use an attribute selector (see Recipe 2.5) to pinpoint which area of the page has the inline styles:

```
div[style] {
 font-size: 1.2em;
}
```

Then use the !important declaration (see Recipe 2.14) to give more weight to the CSS rule:

```
div[style] {
 font-size: 1.2em !important;
}
```

Discussion

The general rule of thumb regarding the many ways to associate CSS rules to a web page (see Recipe 2.11) is "the rule closest to the HTML wins."

Oftentimes that is true. However, what is happening is that the specificity (see Recipe 2.15) and origin (see Recipe 2.12) are jockeying for control over which rules win out over other conflicting rules. With those systems in place, it's possible to override inline styles.

See Also

Chapter 2 for more information on basic CSS styling

12.2 Diagnosing CSS Bugs and Browser Issues

Problem

You want to troubleshoot an issue with either your code or a browser's rendering of CSS.

Solution

Follow these steps to isolate issues with CSS-enabled designs:

1. Validate the HTML by going to *http://validator.w3.org/* and checking the markup.
2. Validate the CSS by going to *http://jigsaw.w3.org/css-validator/* and checking the CSS.
3. Streamline the values of properties. Add a new CSS rule *at the end* of the stylesheet(s), using the universal selector and set properties for all elements:

    ```
    * {
     margin: 0;
     padding: 0;
    }
    ```
4. Border every block-level element:

    ```
    * {
     margin: 0;
     padding: 0;
     border: 1px solid red;
    }
    ```
5. Try different values for properties.
6. Comment out CSS rules and/or properties that are causing the problem. Uncomment CSS properties one by one until the problem recurs. For information on how to add comments within CSS, see Recipe 1.9.
7. Research similar problems through Google and *http://www.positioniseverything .net*, a well-documented collection of CSS bugs.

Discussion

It's common to make typos when we write. The same is true with coding.

Based on personal experience, 90% of the time issues with a CSS-enabled design come from typos in the CSS syntax. The other problems result from fine-tuning trivial bits of an element or CSS property value.

If after going through the steps in the Solution you find that you're still having trouble, do a search through Google or a site devoted to CSS bugs to determine whether anyone else has written and/or discovered a similar problem.

> Instead of setting the margin and padding to zero on some elements, use a CSS Reset stylesheet to start a project.

See Also

CSS: The Definitive Guide by Eric A. Meyer (O'Reilly) to learn more about the CSS specification

12.3 Using Bookmarklets to Troubleshoot CSS

Problem

You want to use third-party websites to help with troubleshooting web pages.

Solution

As discussed in Recipes 1.15 and 2.27, browser bookmarks with JavaScript allow you to easily check within a validator pages that are currently loaded in the browser, as shown in Figure 12-1.
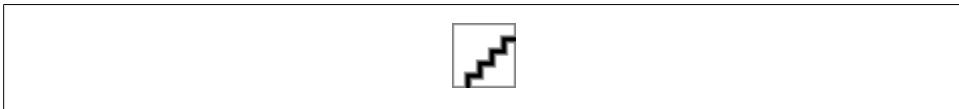
Figure 12-1. Bookmarklet for validating CSS

Here is the code for the HTML bookmarklet:

```
javascript:void(document.location='http://validator.w3.org/check?charset=
%28detect+automatically%29&doctype=Inline&ss=1&group=0&verbose=1&uri='+escape
(document.location))When visiting another site, clicking on the bookmarklet
takes the page currently loaded in the browser and runs it through the
CSS validator.
```

Here is the code for the CSS validator bookmarklet:

```
javascript:void(document.location='http://jigsaw.w3.org/css-
validator/validator?profile=css21&usermedium=all&warning=1&lang=en&uri=
'+escape(document.location))
```

Discussion

In addition to checking the validation of HTML and CSS, bookmarklets are available for checking other aspects of a website.

For example, the W3C provides a service that tests link rot:

```
javascript:void(document.location=
'http://validator.w3.org/checklink?url='+escape(document.location))
```

Link rot is a description of what happens when a link on your site points to another resource on the Web that has disappeared, and returns a File Not Found error.

See Also

Tantek Çelik's bookmarklets at *http://tantek.com/favelets/*

12.4 Using Browser Extensions to Troubleshoot CSS

Problem

You want to utilize the browser to help you find problems with HTML and CSS.

Solution

Use the free Firebug browser extension for Firefox (see *http://getfirebug.com/*) to debug HTML, CSS, and JavaScript within a web page, as shown in Figure 12-2.

Discussion

Instead of saving and reloading an HTML file within a browser, the Firebug extension allows you to inspect and edit the HTML, CSS, and JavaScript of the web page currently in the viewport.

The following developer toolbars built by browser vendors are also available:

- Internet Explorer Developer Toolbar (*http://www.microsoft.com/downloadS/details.aspx?familyid=E59C3964-672D-4511-BB3E-2D5E1DB91038&displaylang=en*)

- Opera Dragonfly (while in the Opera browser, select Tools→Advanced→Developer Tools)

- Safari's Inspect Element (while in the Safari browser, right-click or Ctrl-click on any element within a web page and select Inspect Element)

Figure 12-2. Firebug, which helps you to test and troubleshoot web pages

Microsoft has a web page devoted to IE8 Developer Tools at *http://msdn .microsoft.com/en-us/library/dd565628(VS.85).aspx*.

See Also

The Firebug Working Group mailing list at *http://groups.google.com/group/firebug -working-group*

12.5 Patching Up Internet Explorer 6

Problem

You want to fix some common problems with IE6.

Solution

To fix float-clearing overflow issues in IE6, apply the following CSS declaration to the appropriate selector:

```
height: 1%;
```

To fix margins on floats in IE6 where the margin values are greater than zero, apply the following CSS declaration to the appropriate selector:

```
display: inline;
```

To fix layering issues caused by relative positioning in IE6, apply the following CSS declaration on the appropriate selector:

```
position: static;
```

To add a layout to an element that doesn't have a layout in IE6, apply the following CSS declaration on the appropriate selector:

```
zoom: 1;
```

Discussion

According to a recent survey by Digg.com of its users (see *http://blog.digg.com/?p= 878*), most people who use IE6 feel no need to upgrade to a newer browser or they are locked into using the browser due to technology requirements at work. So, applying unstyled content to IE6 users might be a disservice at least, but prompting them to upgrade when they can't is horrible at most.

However, that does not mean the user experience for IE6 users has to look exactly like the user experience for those who are surfing with IE8 and later, Firefox 3.5 and later, or Safari 4 and later.

Instead, use the following criteria to determine what level of support to give IE6 users:

- Determine what kind of support you are willing to give IE6 users (or browsers that represent a minority in your site logfiles).
- Decide what will make you stop supporting IE6.

- Isolate a coding strategy on how to provide IE6 support (e.g., separate web page or CSS hacks for IE6).
- Let your IE6 users know the steps you are taking. You want to keep them as visitors after the change.

See Also

http://positioniseverything.net/explorer.html for more information on Internet Explorer bugs and workarounds

12.6 Patching Up Internet Explorer 6 with JavaScript

Problem

You want to use JavaScript to make IE6 for Windows render pages better.

Solution

Use programmer Dean Edwards's IE7 script to patch up IE6.

Place the following code in the `head` element of your web page:

```
<!--[if lt IE 7]>
<script src="http://ie7-js.googlecode.com/svn/version/2.0(beta3)/IE7.js"
type="text/javascript"></script>
<![endif]-->
```

Discussion

Before Microsoft released IE7, Dean Edwards worked on a script to fix the CSS short-comings within the IE6 browser. The IE7 script fixes support for selectors, box model issues, overflow issues, PNG alpha transparency issues, and more.

By using conditional comments (see Recipe 12.4), you ensure that only the browsers that need the fix get it.

In the Solution, JavaScript code is delivered through Google's servers. If other websites use this same solution, the code is cached on the IE6 user's machine, making the user's browser experience faster.

Using jQuery

Another JavaScript solution to fix CSS implementation of IE6 is to use jQuery to apply CSS rules on a case-by-case basis. See Recipe 14.4 for more information.

See Also

Natalie Downe and Jeremy Keith's formula for determining whether it's worthwhile to use Dean Edwards's script to support IE6 users at *http://24ways.org/2008/the-ie6-equation*

12.7 Using Conditional Comments to Deliver Styles to Different Versions of Internet Explorer

Problem

You want to deliver specific code to different versions of Internet Explorer for Windows.

Solution

Use Microsoft's Internet Explorer conditional comments:

```
<!--[if IE]>
<p>You are seeing this sentence because you are using an Internet Explorer
browser.</p>
<![endif]-->
```

To deliver code to different versions of Internet Explorer for Windows, use the browser version number:

```
<!--[if IE 5]>
<p>You are seeing this sentence because you are using Internet Explorer 5</p>
<![endif]-->
<!--[if IE 5.0]>
<p>You are seeing this sentence because you are using Internet Explorer 5.0</p>
<![endif]-->
<!--[if IE 5.5]>
<p>You are seeing this sentence because you are using Internet Explorer 5.5</p>
<![endif]-->
<!--[if IE 6]>
<p>You are seeing this sentence because you are using Internet Explorer 6</p>
<![endif]-->
```

To deliver code to Internet Explorer 5 for Windows and later versions, use this code:

```
<!--[if gte IE 5]>
<p>You are seeing this sentence because you are using Internet Explorer 5 and
up</p>
<![endif]-->
```

To deliver code to Internet Explorer 5.5 for Windows and earlier versions, use this code:

```
<!--[if lte IE 5.5]>
<p>You are seeing this sentence because you are using Internet Explorer lower
or equal to 5.5</p>
<![endif]-->
```

To deliver code to Internet Explorer for Windows earlier than Internet Explorer 6, use this code:

```
<!--[if lt IE 6]>
<p>You are seeing this sentence because you are using Internet Explorer lower
than 6</p>
<![endif]-->
```

Discussion

Microsoft developed its own proprietary comment system to deliver specific HTML code to different versions of its browser, Internet Explorer for Windows.

You can use this code only when you place *HTML* between the conditional statements. However, this still means that you can specifically target CSS rules through conditional comments.

For example, to deliver a stylesheet targeted for Internet Explorer 5.x, place a link tag to a stylesheet between two conditional comments:

```
<link rel="stylesheet" type="text/css" media="screen, presentation"
href="/_assets/css/screen/screen.css" />
<link rel="stylesheet" type="text/css" media="aural"
href="/_assets/css/aural.css" />
<!--[if lt IE 6]>
<link rel="stylesheet" type="text/css" media="screen, presentation"
href="/_assets/css/screen/ie.css" />
<![endif]-->
```

You can also place embedded styles in between comment conditionals:

```
<!--[if lt IE 6]>
<style type="text/css">
 h1 {
   font-family: Verdana, Arial, Helvetica, sans-serif;
   font-size: 36px;
 }
</style>
<![endif]-->
```

In addition, you can use comment conditionals in conjunction with an intelligent hacking system. See Recipe 12.9 for more information about setting up a hacking system.

> There is a difference between IE 5 and IE 5.0 when using conditional comments.

Things to keep in mind

To isolate code for just Internet Explorer 5.0, use `IE 5.0`. To deliver code to all Internet Explorer versions within the major release of 5, use `IE 5`.

The markers `lt` and `gt` mean "less than" and "greater than," respectively, whereas `lte` and `gte` mean "less than or equal to" and "greater than or equal to," respectively.

See Also

The MSDN article on conditional comments at *http://msdn.microsoft.com/en-us/library/ms537512(VS.85).aspx*

12.8 Using CSS Filters to Deliver CSS Rules to Almost Any Browser

Problem

You want to associate CSS rules to browsers other than Internet Explorer to fix rendering issues.

Solution

Use CSS filters to let CSS rules apply when viewed by a specific browser.

After determining which browser or browsers to apply CSS "fixes," reference the CSS filters reference chart at *http://centricle.com/ref/css/filters/*, as shown in Figure 12-3.

Click on the CSS selector to determine how to apply the filter to your stylesheets and to read about the filter.

Discussion

Filters are hacks that exploit holes in the browser's support for CSS. Depending on the type of filter, hacks can pinpoint one or two browsers, or a whole range of browsers.

> Do not confuse CSS filters with the `filter` property, which is a proprietary CSS property developed by Microsoft.

See Also

The Wikipedia article on CSS filters at *http://en.wikipedia.org/wiki/CSS_filter*

Figure 12-3. CSS filter matrix

12.9 Setting Up an Intelligent CSS Delivery System for Modern Browsers

Problem

You want to develop a system to separate correct CSS rules from those used for hacks or workarounds.

Solution

Link a stylesheet to a web page:

```
<!--[if gte IE 7]><!-->
<link rel="stylesheet" type="text/css" media="screen, projection"
 href="screen.css" />
<!--<![endif]-->
```

Within the *screen.css* stylesheet, import three separate stylesheets: the CSS Reset stylesheet, the main stylesheet for modern browsers, and a stylesheet containing hacks and workarounds for older IE browsers:

```
/* Import style sheets, hiding from IE/Mac \*/
@import url("reset.css");
@import url("csscookbook.css");
@import url("ie.css");
/* End of import hide */
```

Discussion

Keeping stylesheets separated based on their browser support has a couple of benefits. First, it keeps the base stylesheet clean of any hacks and workarounds.

Second, keeping hacks and workarounds specific to each browser in their own file means you can easily delete the CSS rules if you want to stop supporting that particular browser.

Also, the Solution uses poor CSS commenting implementation in Internet Explorer for Macintosh to keep that browser from seeing the stylesheets. Although Microsoft no longer supports the browser, check your site logfiles to see whether you get any traffic to warrant keeping advanced CSS rules from IE Mac users. (Most likely you can delete the comment hack, but it's better to be safe than sorry.)

For a list of CSS filters to target specific browsers, see Recipe 12.7.

Server-side solution

The technique discussed in the Solution uses CSS-based hacks to deliver stylesheets. Another approach is to use a server-side solution. Mark Pilgrim, a web developer, devised a solution based on `mod_rewrite` in the Apache server.

By detecting the browser's HTTP user agent, each browser gets its own stylesheet in addition to the base stylesheet. For more information about this technique, see *http://diveintomark.org/archives/2003/01/16/the_one_ive_never_tried*.

See Also

Molly Holzschlag's article on hack management at *http://www.peachpit.com/articles/article.aspx?p=170511*

12.10 Testing a Site Design on More Than One Platform with Only One Computer

Problem

You want to test your website on more than one browser, but you have access to only one computer.

Solution

Running emulators on a computer can help you avoid the cost of owning multiple workstations.

The following emulators are available.

For PC users:

Knoppix
> Knoppix is a Linux operating system that resides on a bootable CD-ROM, meaning no installation is required. For more information, see *http://www.knoppix.org/*.

VMware Workstation
> VMware Workstation allows you to set several virtual operating systems to run on top of the Microsoft operating system. Although you need to install software, you don't have to restart the computer every time you want to test a website. For more information, see *http://www.vmware.com/products/ws/*.

Debian Linux
> You can install Debian Linux on a separate partition on the computer, allowing you to boot into either Linux or Windows. For more information, see *http://www.aboutdebian.com/dualboot.htm*.

For Macintosh users:

Parallels Desktop
> Parallels Desktop allows you to set up virtual operating systems including Windows XP Professional, Windows XP Home, and Windows 2000 Professional, and thus different versions of Internet Explorer. For more information, see *http://www.microsoft.com/mac/products/virtualpc/virtualpc.aspx*.

Boot Camp
> Boot Camp, an Apple application, allows you to install the Windows operating system on a partition of Intel Macs. For more information, see *http://www.apple.com/support/bootcamp/*.

For Linux Workstation users:

Wine

> The open source Wine software is an implementation of the Windows API that runs on top of X and Linux. For more information, see *http://www.winehq.com/*.

Discussion

To achieve cross-platform, cross-browser designs with CSS, you must check and test websites in as many sites as possible. To do that on a budget, you need to install more than one operating system on your computer.

Once you've installed more than one operating system on your computer, install a browser on the new system. You can do so quickly by visiting the browser archive at *http://browsers.evolt.org/*.

Using a remote screen-capture service

BrowserCam is a web-based screen-capture service. Fill out a form supplying a link to a web page, and specify which browsers and operating system configurations you want to see. Then the service will take screen captures of those systems for you to check. For more information, see *http://www.browsercam.com/*.

Adobe BrowserLab renders web pages within a Flash-based environment. At the time of this writing, BrowserLab shows page renderings in Firefox 2 and later, IE6, IE7, and Safari 3 for Macintosh. Although the service is free, you must have an Adobe ID account to use it. See *https://browserlab.adobe.com/*.

BrowserShots (see *http://browsershots.org/*) allows for free, but slow, processing of screenshots in multiple browsers. A nominal fee is required for faster processing.

See Also

http://www.thesitewizard.com/webdesign/multiplebrowsers.shtml for more information on setting up more than one browser on one computer

12.11 Testing a Website with a Text Browser

Problem

You want to install more than one version of Internet Explorer for Windows on a machine.

Solution

Use an online web tool such as Lynx Viewer (see *http://www.delorie.com/web/lynxview .html*) that emulates a text browser.

Place a file named *delorie.htm* on the root folder of your website (the file can be blank). The file tells the web application that it is acceptable for the browser to view and render the site through the online service.

Afterward, enter your site's URL and check the results online, as shown in Figure 12-4.
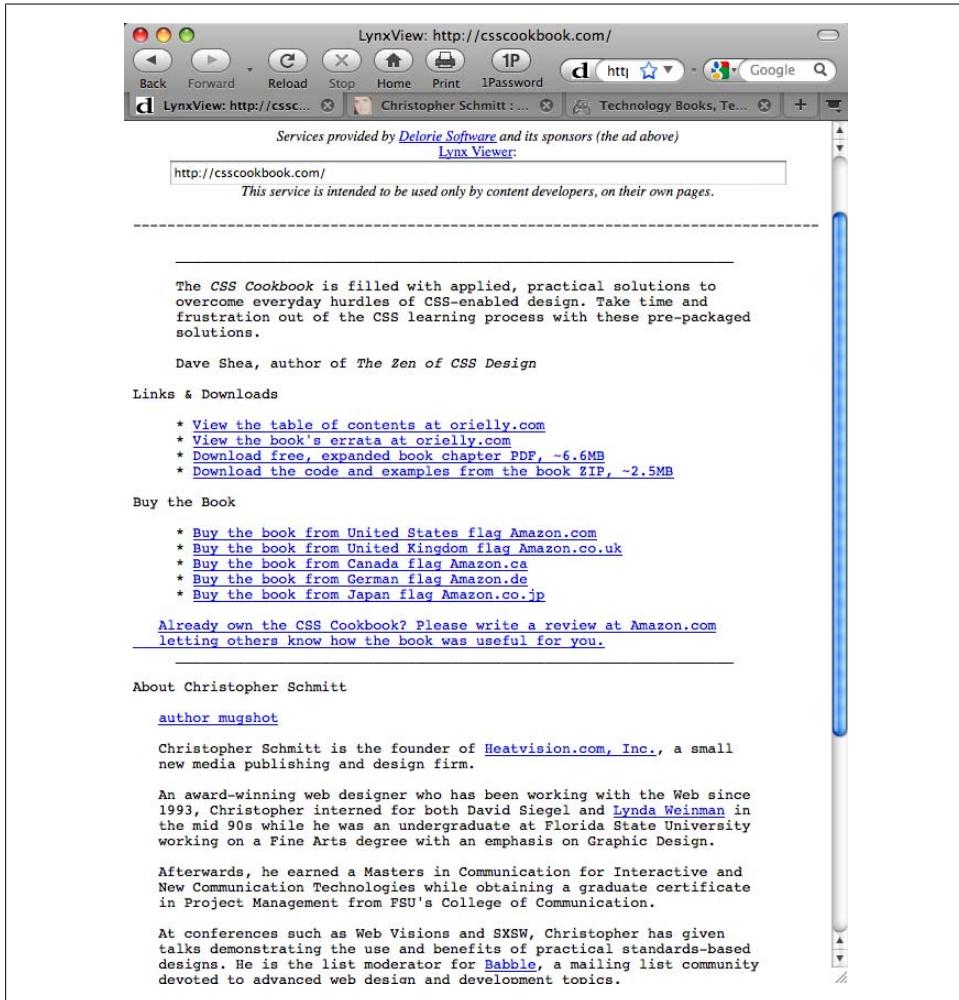
Figure 12-4. A view of a website through Lynx Viewer

Discussion

Lynx is the archetypal text browser. Instead of running a web-based emulator, you can download Lynx and install it on your personal computer. For more information, see *http://lynx.isc.org/*.

See Also

http://people.cc.ku.edu/~grobe/early-lynx.html for a brief history of Lynx

Designing with CSS

13.0 Introduction

Although web builders often spend a lot of time working around browser bugs and reading about the latest tricks from the gurus, it is worth remembering that first and foremost we're designers and CSS is simply a way to turn design ideas into reality.

CSS is the perfect technology for grabbing website visitors' attention. With CSS, instead of hacking HTML tables and slicing images to create eye-catching designs, you can go further with valid markup and still save on file size by ditching excess HTML and images. In short, you can do what any professional web designer should: create maximum impact with minimal resources.

At a basic level, a developer can learn all there is to know about CSS syntax and the technical limitations of the technology. But let's never forget that code merely implements the design. At its heart CSS is a *visual* language, and with that comes the need to understand, at least in some small way, how to use design principles with CSS.

With that in mind, this chapter explains how to design with CSS. Specifically, this chapter describes several methods for capturing attention through CSS-enabled techniques, including how to lead the eye with contrast, use excessively large text, create word balloons out of quotations, and use different image formats to create cohesive presentations.

13.1 Enlarging Text Excessively

Problem

You want to draw attention to a web page by enlarging some of the text, as shown in Figure 13-1.

Figure 13-1. An example of excess type size

Solution

Increase the size of the heading so that it is out of proportion with the rest of the text. First use this HTML:

```
<h1>Hi.</h1>
```

Then use this CSS code:

```
h1 {
  font-size: 17em;
```

```
    margin: 0;
    padding: 0;
    text-align: center;
    font-family: Arial, Verdana, Helvetica, sans-serif;
}
```

Discussion

Obviously, any element that's larger than the other elements in the same environment stands out. This approach makes a page look more dynamic in its presentation, unlike a page layout where all the elements are the same size.

So, when you want to call attention to an area of a web page, one way is to try using an excessive type size.

In this example, the size of the font in "Hi." has been set to 17 em. In the `font-size` property, an em unit is equal to whatever is the font size of the container.

So, 17 em units are equal to 17 times the default font size. There is no theoretical limit to how large you can size text, but in practice, different browsers do max out at some point. Not everyone will have a monitor that's large enough to see type that is 1 mile (or 63,360 inches) tall:

```
h3 {
  font-size: 63360in;
}
```

See Also

Recipe 1.2 for specifying font measurements and sizes; "The Elements of Text and Message Design and Their Impact on Message Legibility: A Literature Review," from the Journal of Design Communication at *http://scholar.lib.vt.edu/ejournals/JDC/Spring -2002/bix.html*; the CSS2 specification for lengths (including em units) at *http://www .w3.org/TR/REC-CSS2/syndata.html#length-units*

13.2 Creating Unexpected Incongruity

Problem

You need to grab the reader's attention by using two elements that don't seem to fit together.

Solution

Place one element visually inside the other. In the web page shown in Figure 13-2, which covers Earth's close call with an asteroid, an image of Earth from space was placed over an image of a game of pool.
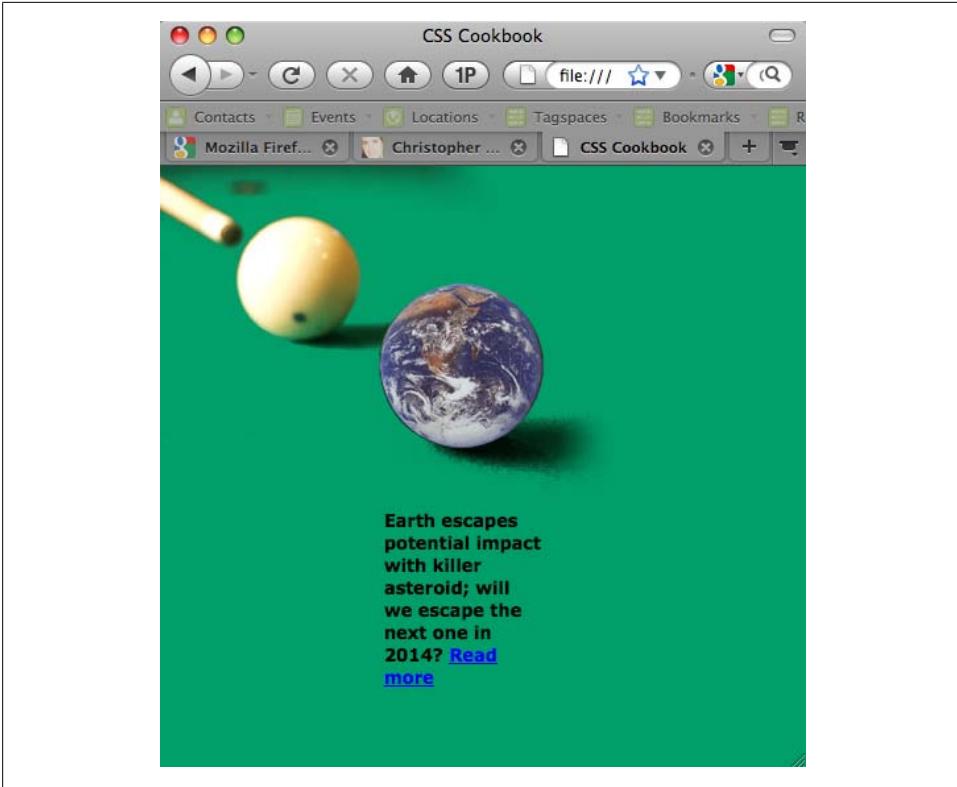
Figure 13-2. An image of Earth placed over an image depicting a game of pool

The HTML for this page is simple:

```
<h2><span class="no">Earth News</span></h2>
<p>Earth escapes potential impact with killer asteroid;
will we escape the next one in 2014? <a href="more.html">Read
more</a></p>
```

For the CSS, place the photo depicting the game of pool into the body element and position it in the upper-left corner. Then use the image replacement technique discussed in Recipe 3.10 to place the photo of Earth for the h2 element:

```
<style type="text/css">
body {
 background-color: #009E69;
 margin: 0;
 background-image: url(billiard.jpg);
 background-repeat: no-repeat;
}
h2 {
 background-image: url(earth.gif);
 position:absolute;
 width:126px;
```

```
   height:126px;
   z-index:1;
   left: 166px;
   top: 69px;
}
.no {
 display: none;
}
p {
 width: 120px;
 margin: 260px 100px 0 170px;
 font-family: Verdana, sans-serif;
 font-size: small;
 font-weight: bold;
}
</style>
```

Discussion

A great way to grab attention is to show something that is unexpected. Cleverly combining two different elements into one image can force viewers to pay attention to the image (as seen in Figure 13-3), or it can simply underscore the purpose of the content.

Figure 13-3. Photos of a child and man, combined

This recipe used two images—one of a pool cue and cue ball, and the other of Earth. The former image was placed as the background image for the body element. The image of Earth was placed in the background of the h2 element and was moved by setting the position to absolute. Then it was composited over the pool image.

See Also

Recipe 4.22 for combining different image formats; Recipe 13.3 for combining unlike elements

13.3 Combining Unlike Elements to Create Contrast

Problem

You want to create contrast on a web page by integrating two different elements, such as serif and sans serif typefaces, as shown in Figure 13-4.
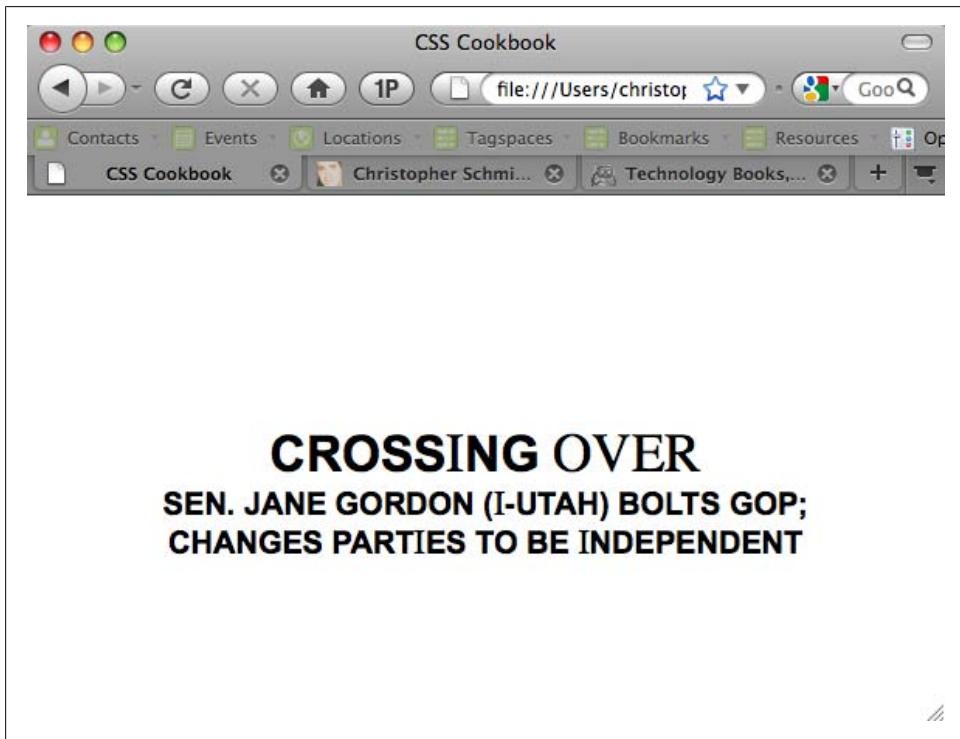
Figure 13-4. Type elements juxtaposed in the same headline

Solution

Use different typefaces in the same headline. First adjust the markup to allow for changes in the font properties:

```
<h2>Cross<span>i</span>ng <span>Over</span></h2>
<h4>Sen. Jane Gordon (<span>I</span>-Utah) bolts GOP;
<br />changes part<span>i</span>es to be
<span>I</span>ndependent</h4>
```

Then manipulate the CSS for the span element to create a mixture of typefaces:

```
body {
 margin: 25% 10% 0 10%;
}
h2 {
 font-size: 2em;
 font-weight: bold;
 font-family: Arial, Verdana, Helvetica, sans-serif;
 text-transform: uppercase;
 text-align: center;
 padding: 0;
 margin: 0;
}
h2 span {
 font-family: Times, "Times New Roman", Georgia, serif;
 font-size: 1.1em;
 font-weight: normal;
 }
h4 {
 margin: 0;
 padding: 0;
 font-size: 1.25em;
 font-weight: bold;
 font-family: Arial, Verdana, Helvetica, sans-serif;
 text-transform: uppercase;
 text-align: center;
}
h4 span {
 font-family: Times, "Times New Roman", Georgia, serif;
 font-size: 1.1em;
 font-weight: normal;
 }
```

Discussion

Combining unlike elements creates a visual contrast. In this example, different characteristics of the serif and sans serif typefaces in the headline created the contrast. However, you can create contrast through imagery as well. For instance, in this example, you could have integrated Democratic and Republican political party symbols and placed them side by side. Or you could have gone for a more symbolic contrast by placing photos of two different types of parties side by side: one depicting a large social

gathering at a club and the other showing a girl blowing a noisemaker over a cupcake with a lit candle on top.

See Also

Recipe 4.22 for combining different image formats

13.4 Leading the Eye with Contrast

Problem

You want to create a sense of depth or motion through text. On a page containing four paragraphs that are almost identical, it's hard to know which paragraph to look at first. If you change the font size across columns in a particular direction (e.g., decrease the size right to left) you lead the reader's eye (see Figure 13-5).
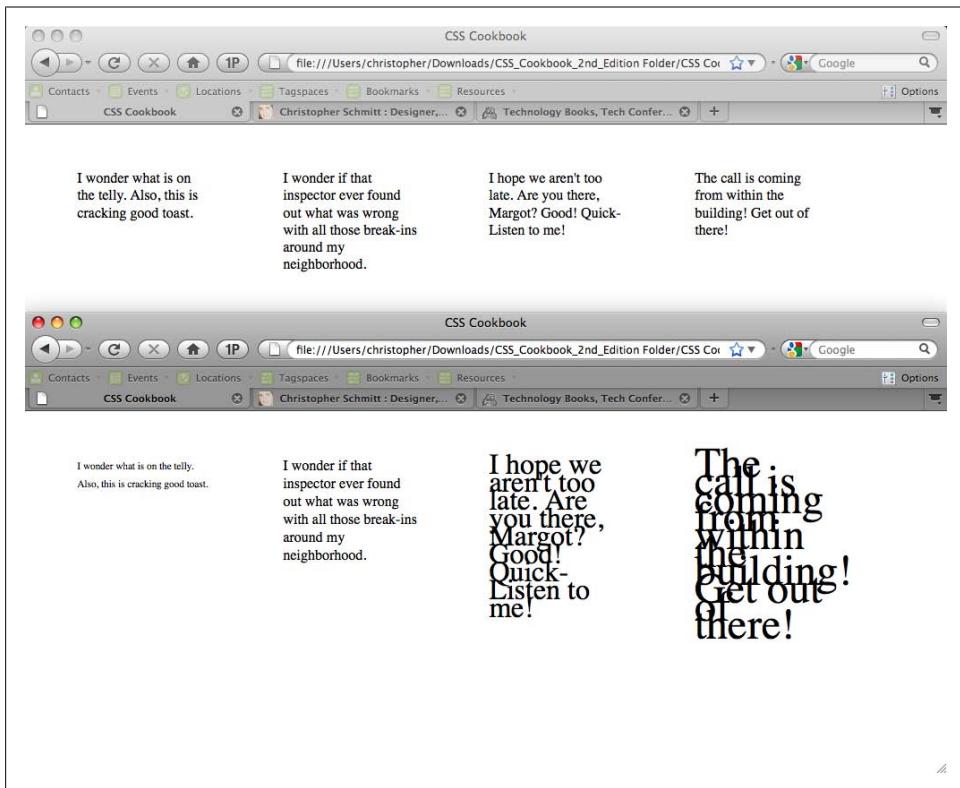
Figure 13-5. Four paragraphs that are almost identical, then changed with increasing contrast

Solution

To lead the reader's eye, change the type size by adding a CSS rule such as this:

```
/* Text size */
#layer4 {
 font-size: .7em;
 line-height: 20px;
}
#layer3 {
 font-size: 1em;
 line-height: 20px;
}
#layer2 {
 font-size: 2em;
 line-height: 10px;
}
#layer1 {
 font-size: 3em;
 line-height: 10px;
}
```

Discussion

Contrast occurs when there is an obvious difference between two elements. If there isn't any contrast on a page, the reader doesn't know what is important on the page. By manipulating an element's visual value, you can create contrast between two like elements. Some of those visual values include the following:

- Size
- Color
- Shape
- Position on a page
- Direction
- Density

Properly marked content has an inherent style because the browser uses its own stylesheet to render the content when another stylesheet isn't present. Headings, such as the h1 element, are stylized in a large, bold font and are separated from the paragraphs. This different font provides the contrast to help readers make sense of the document.

Without the cues that can be provided through a stylesheet, the reader's eye wanders throughout a document.

See Also

Lighthouse International's website, *http://www.lighthouse.org/color_contrast.htm*, for creating more effective contrast

13.5 Checking for Enough Color Contrast

Problem

You want to make sure there is enough contrast between two colors.

Solution

Use the Luminosity Colour Contrast Analyser from JuicyStudio.com at *http://juicystudio.com/services/luminositycontrastratio.php*.

Enter two color values into the validator and click the Calculate Luminosity Contrast Ratio button.

Along with a color sample of the two colors, you receive a summary noting whether you pass luminosity contrast level 2, level 3, or not at all. The example in Figure 13-6 shows that the color combination has passed both levels 2 and 3.

Discussion

The W3C's Web Content Accessibility Guidelines state that to make text legible, designers need to ensure that the content in the foreground can be perceived against the background.

When the color for text is close to the same hue as the background color, the text becomes illegible. For the text to be legible, the colors need to have greater contrast by being farther apart from each other in the spectrum, or the text needs to be significantly darker or lighter than the background.

Levels of luminosity

For colors to pass the second level of luminosity, the luminosity contrast ratio needs to be at least 5:1. That means one color needs to be at least five times as dark or as light as the other color.

For colors to pass the third level, the luminosity contrast ratio must be at least 10:1.

See Also

JuicyStudio.com's explanation of the Suggested Luminosity Contrast Ratio Algorithm at *http://juicystudio.com/article/luminositycontrastratioalgorithm.php*
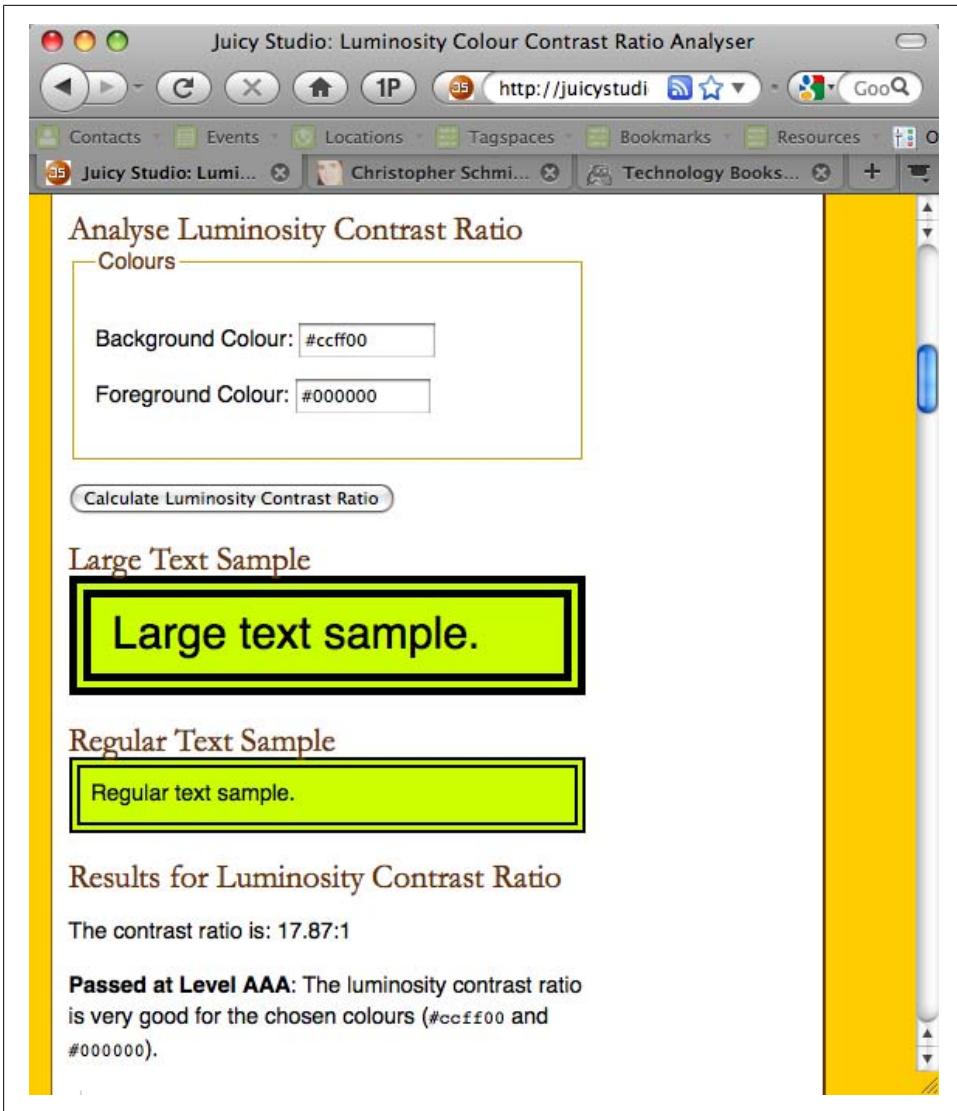
Figure 13-6. The results of the luminosity test

13.6 Emphasizing a Quotation with Smart Quotes

Problem

You want to add emphasis to a quotation by using large and bold quotation marks, as shown in Figure 13-7.
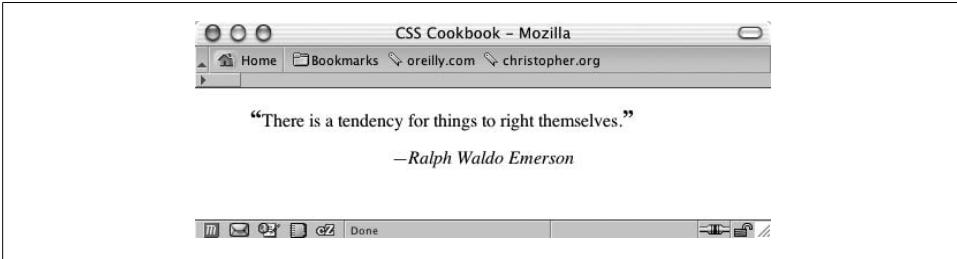
Figure 13-7. A stylized quotation

Solution

First, code the markup for the quotation (see Figure 13-8):

```
<blockquote>
 <p>There is a tendency for things to right themselves.</p>
 <cite>Ralph Waldo Emerson</cite>
</blockquote>
```
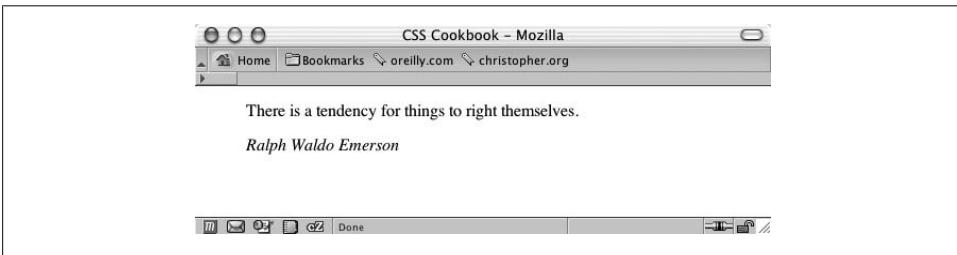

Figure 13-8. Quotation as it would normally appear

Then apply CSS rules to stylize the quote:

```
blockquote {
 padding: 0;
 margin: 0;
 text-align: center;
}
p {
 font-size: 1em;
 padding-bottom: 3em;
 text-transform: lowercase;
 font-family: Georgia, Times, "Times New Roman", serif;
 margin: 0;
 padding: 0;
}
cite {
 display: block;
 text-align: center;
}
```

Finally, use the `:before` and `:after` pseudo-elements to stylize the punctuation in the quotation as well as to place an em dash—a horizontal dash equal to the default size of the font—before the name of the cited source:

```
blockquote p:before {
  content: "\201C";
  font-size: 1.2em;
  font-weight: bold;
  font-family: Georgia, Times, "Times New Roman", serif;
}
blockquote p:after {
  content: "\201D";
  font-size: 1.2em;
  font-weight: bold;
  font-family: Georgia, Times, "Times New Roman", serif;
}
cite:before {
  content: "\2014 ";
}
cite {
  display: block;
  text-align: center;
}
```

Discussion

Pseudo-elements are selector constructs that browsers use first to select portions and then to stylize a web page that can't be marked up through standard HTML. For instance, you can use pseudo-elements to stylize the first line of a paragraph or, in the case of this recipe, to place generated content before and after an actual element.

In this Solution, you inserted smart quotes around the actual quotation. For the left double quotes, you used this declaration:

```
content: "\201C ";
```

Any text that you want displayed after an element needs to be marked off with double quotes. Because you are using double quotes to mark what should be displayed, you can't put another set of double quotes inside the first set. To put quotes around the quotation, you need to use the hexadecimal value for a quotation mark, which is 201C.

Because anything between the quotation marks automatically is generated as is, you need to escape the hexadecimal number that tells the browser to render the quotation marks by placing a forward slash in front of the double quotes.

The `content` property in the CSS 2.1 specification contains values for easily inserting quotation marks. For example, to re-create the left double quotes, use the following declaration:

```
content: open-quote;
```

However, note that open-quote keyword value specification is implemented only in Mozilla and Opera. Also, note that the `:before` and `:after` pseudo-elements don't work in Internet Explorer 5 and later for Windows and Internet Explorer for Macintosh.

See Also

The CSS2 specification for quotations for generated content at *http://www.w3.org/TR/ REC-CSS2/generate.html#quotes*

13.7 Setting a Moving Background Scene When a User Resizes the Window

Problem

You want to have overlaying background images move as a user resizes the browser window, as shown in Figure 13-9.
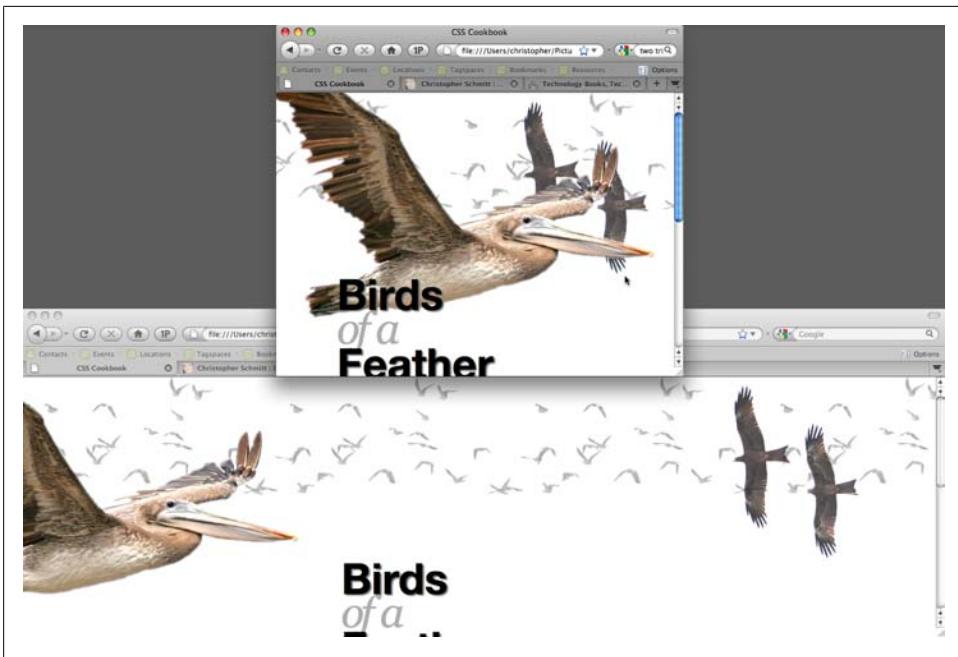
Figure 13-9. Images moving location based on the resizing of the browser window

Solution

Set a background image in the `body` element, with a negative percentage value:

```
body {
  font-size: 62.5%;
  background-color: #fff;
  background-image: url(birds-flock.png);
  background-repeat: repeat-x;
  background-position: -80% 0;
}
```

Then wrap the content on the web page with two additional `div` elements:

```
<body>
  <div id="birds2">
    <div id="birds1">
      <div id="filler">
      [...]
      </div>
    </div>
  </div>
</body>
```

Set background images within the respective `div` elements with different background position values:

```
#birds2 {
  background-image: url(birds-two.png);
  background-position: 90% 0;
  background-color: transparent;
  background-repeat: no-repeat;
  width: 100%;
}
#birds1 {
  background-image: url(birds-one.png);
  background-position: -20% 0;
  background-color: transparent;
  background-repeat: no-repeat;
  width: 100%;
}
```

Discussion

By utilizing percentage-based lengths set to different values, you give the user a sense that he is looking at one object with more than one viewpoint as the backgrounds move in different directions. This effect is termed *parallax*, which is based on the Greek word *parallaxis*, meaning "alteration."

For this effect to work, the background images require alpha-transparent PNG images. The subtle opacity changes as the background images move across each other, helping to sell the visual effect.

Using JavaScript, you can trigger the effect by moving your mouse. For more information and to download the code, see *http://webdev.stephband.info/parallax.html*.

See Also

A more detailed description of the effect at *http://carsonified.com/blog/design/how-to-recreate-silverbacks-parallax-effect/*

13.8 Adding Animation to Elements on a Page

Problem

You want to animate an element within a web page, as shown with the cloud image in Figure 13-10.

Figure 13-10. Clouds moving in the background

Solution

Using Safari's proprietary animation properties, set an element to move back and forth.

First add an HTML element after the opening **body** element:

```
<body>
 <div id="clouds1"></div>
 [...]
</body>
```

Insert a background image along with width, height, and absolute positioning to style the element:

```
#clouds1 {
 width: 627px;
 height: 410px;
 position: absolute;
 right: -300px;
```

```
  top: 0;
  background-image: url(clouds.png);
}
```

Next, set the keyframes for the animation, giving them the names of clouds:

```
@-webkit-keyframes "clouds" {
 from {
  right: 0px;
 }
 to {
  right: 100px;
 }
}
```

Start the animation by associating the keyframes to the "clouds1" `div` element along with instructions on how long the animation should last, how often it repeats, and the direction of the movement:

```
#clouds1 {
  width: 627px;
  height: 410px;
  position: absolute;
  right: -300px;
  top: 0;
  background-image: url(clouds.png);
  -webkit-transform: translate(300px, 0px);
  -webkit-animation-name: "clouds";
  -webkit-animation-duration: 10s;
  -webkit-animation-iteration-count: 10;
  -webkit-animation-direction: alternate;
}
```

Discussion

Although CSS-enabled animation is supported only in Safari as of this writing, the W3C is currently working on the specification. Other browser vendors may opt to develop similar proprietary extensions of their browsers until the animation specification is finalized.

Animation keyframes

The first step when setting animations is to define what's called a `keyframes` rule:

```
@-webkit-keyframes "clouds" {
 from {
  right: 0px;
 }
 to {
  right: 100px;
 }
}
```

The keyframes set the starting and ending points of an animation, but also allow for more refined control of how the animation is displayed. Instead of using the `from` and

to keyframe selectors to state the starting and stopping points, you can use percentage-based values:

```
@-webkit-keyframes "clouds" {
 0% {
  right: 0px;
 }
 33% {
  right: 10px;
 }
 68% {
  right: 90px;
 }
 100% {
  right: 100px;
 }
}
```

The animations can also take more than one property, to create diagonal animations:

```
@-webkit-keyframes "clouds" {
 0% {
  right: 0px;
  top: 0px;
 }
 33% {
  right: 10px;
  top: 10px;
 }
 68% {
  right: 90px;
  top: 90px;
 }
 100% {
  right: 100px;
  top: 100px;
 }
}
```

> To create interesting animation effects, try using the `opacity` property for keyframe selectors as well.

Animation properties

Within the element that is animated, the `animation-transform` property is used to associate which keyframe rule is used:

```
@-webkit-keyframes "clouds" {
 from {
  right: 0px;
 }
 to {
```

```
   right: 100px;
  }
}
#clouds1 {
 width: 627px;
 height: 410px;
 position: absolute;
 right: -300px;
 top: 0;
 background-image: url(clouds.png);
 -webkit-transform: translate(300px, 0px);
 -webkit-animation-name: "clouds";
}
```

The `animation-duration` property sets how long the animation effect is to transpire:

```
#clouds1 {
 width: 627px;
 height: 410px;
 position: absolute;
 right: -300px;
 top: 0;
 background-image: url(clouds.png);
 -webkit-transform: translate(300px, 0px);
 -webkit-animation-name: "clouds";
 -webkit-animation-duration: 10s;
}
```

A negative value for `animation-duration` is treated like a zero.

To limit to 10 the number of times the animation cycles, set the value of the `animation-iteration-count` to `10`:

```
#clouds1 {
 width: 627px;
 height: 410px;
 position: absolute;
 right: -300px;
 top: 0;
 background-image: url(clouds.png);
 -webkit-transform: translate(300px, 0px);
 -webkit-animation-name: "clouds";
 -webkit-animation-duration: 10s;
 -webkit-animation-iteration-count: 10;
}
```

To have the animation loop constantly, set the value of `animation-iteration-count` to `infinite`:

```
 -webkit-animation-iteration-count: infinite;
```

When an animation reaches the end of the keyframe, but is set to repeat another cycle, you can set the animation to go into reverse by setting the `animation-direction` property to `alternate`:

```
#clouds1 {
 width: 627px;
 height: 410px;
 position: absolute;
 right: -300px;
 top: 0;
 background-image: url(clouds.png);
 -webkit-transform: translate(300px, 0px);
 -webkit-animation-name: "clouds";
 -webkit-animation-duration: 10s;
 -webkit-animation-iteration-count: 10;
 -webkit-animation-direction: alternate;
}
```

The default value for `animation-direction` is `normal`, which means the animation loops. Once an element has reached the final point, it disappears and reappears at the starting point to begin again.

See Also

The CSS3 specification for animations at *http://www.w3.org/TR/css3-animations/*

13.9 Creating a Fireworks Display As a User Scrolls

Problem

You want to display fireworks in the background of a page with changing colors as a user scrolls the browser window, as shown in Figure 13-11.
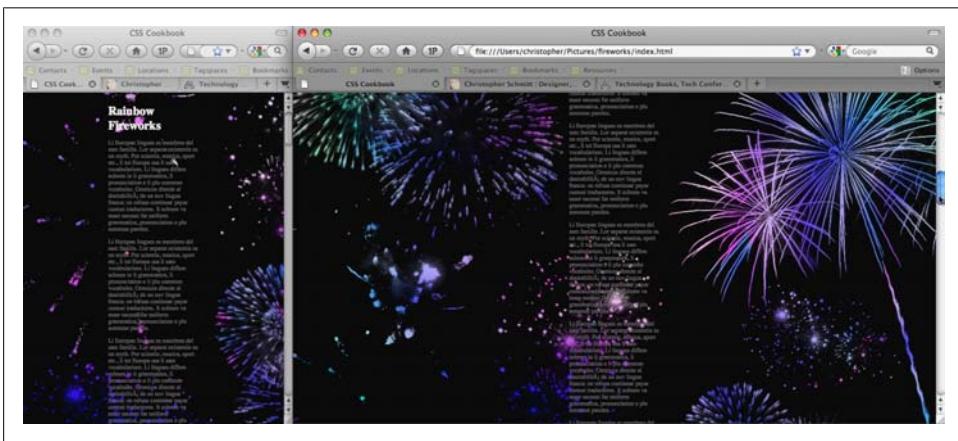
Figure 13-11. The fireworks changing color as the web page is scrolled

Solution

First, set a rainbow-colored background image in the `body` element, making sure to also set the `background-attachment` property to `fixed`:

```
body {
  font-size: 62.5%;
  background-color: #fff;
  background-image: url(bkgd.jpg);
  background-position: center;
  background-color: white;
  background-attachment: fixed;
}
```

Next, wrap the content on the web page with a `div` element:

```
<body>
 <div id="easel">
  [...]
 </div>
</body>
```

Then place an image consisting of reversed silhouette images of fireworks:

```
#easel {
  background-image: url(fireworks.png);
  width: 100%;
  height: 100%;
}
```

Discussion

As in Recipe 13.7, this Solution relies on layering images on top of each other. By constraining the colorful background image through an elaborate keyhole, you make the rainbow colors appear to compose the fireworks.

As a user scrolls the browser, the rainbow background graphic stays in place, but the fireworks image is tiled and appears to scroll. This user behavior completes the effect of a simple rainbow animation.

In addition to scrolling animation, if the user resizes the browser window, the colors of the fireworks also change. This effect is due to centering the rainbow image in the `body` element. As the browser resizes, the browser repositions the background image to be centered.

See Also

Recipe 13.7 for setting a moving background scene when a user resizes the window

13.10 Customizing the View Source Stylesheet for Firefox

Problem

You want to modify the design of the code that appears when viewing the source of web pages in Firefox.

Solution

If you're using a Macintosh, you can follow these steps:

1. Find the Firefox application file, *Firefox.app*, in the Applications folder.
2. Ctrl-click the application to pull up a dialog box and select Show Package Contents.
3. Go to Folder Contents→MacOS→res and open the *viewsource.css* file. Once opened, edit the file to your liking.

If you're a Windows user, you can locate the stylesheet at this directory: *C:\Program Files\Mozilla Firefox\res\viewsource.css*.

Discussion

To use your own images, make sure the image is placed in the same folder as the *viewsource.css* file.

Setting an external editor

Instead of customizing the look of the view-source output with CSS, you can use an external, third-party application in place of Firefox.

Type `about:config` in the location box in the browser. Then search for *view_source.editor.external*. Ctrl-click or right-click the option to change `false` to `true`.

Then Ctrl-click or right-click `view_source.editor.path` and set the value to the path of the code editor.

For PC versions of Firefox using the Notepad++ application:

```
view_source.editor.path: C:\Program Files\Notepad++\notepad++.exe
```

For Mac versions of Firefox using the TextMate application:

```
view_source.editor.path: /Applications/TextEdit.app/Contents/MacOS/TextMate
```

See Also

Recipe 2.1 for information on how authors can create their own stylesheets

13.11 Designing with Grids (CSS Frameworks)

Problem

You want to design a web page using a premade grid system with CSS.

Solution

Before building with HTML or CSS code, first design with a Blueprint Illustrator or Photoshop template by downloading the file(s) at *http://urlgreyhot.com/personal/we blog/photoshop_and_illustrator_templates_blueprint_css_framework*.

Then, using the templates as a basis, design the mockup of the site.

> The default column system for Blueprint CSS is 24 columns that are 30 pixels wide with a 10-pixel margin or gutter.

Download the Blueprint CSS files from *http://www.blueprintcss.org* and include those files in your website development files. Then associate the CSS files in the `head` element:

```
<link rel="stylesheet" href="css/blueprint/screen.css" type="text/css"
media="screen, projection">
<link rel="stylesheet" href="css/blueprint/print.css" type="text/css"
media="print">
<!--[if lt IE 8]>
<link rel="stylesheet" href="css/blueprint/ie.css" type="text/css"
media="screen, projection">
<![endif]-->
```

You should make any additional CSS rules for customizing the page layout in a separate stylesheet, and associate them through a `link` element so as to override the CSS rules (see Recipe 2.15) from the framework:

```
<link rel="stylesheet" href="css/blueprint/screen.css" type="text/css"
media="screen, projection">
<link rel="stylesheet" href="css/blueprint/print.css" type="text/css"
media="print">
<!--[if lt IE 8]>
<link rel="stylesheet" href="css/blueprint/ie.css" type="text/css"
media="screen, projection">
<![endif]-->
<link rel="stylesheet" href="css/example.css" type="text/css"
media="screen, projection">
```

Within the `body` element, wrap the content with a `div` element that has an `id` attribute with a value of `container`:

```
<div class="container">
 ...
</div>
```

To create columns, use the `class` attribute along with a value of `span` and append the number of columns that content should cross:

```
<div class="container">
 <div class="span-24">
  Header
 </div>
 <div class="span-5">
  Side column
 </div>
 <div class="span-14">
  Main column
 </div>
 <div class="span-5">
  Side column
 </div>
</div>
```

Since the margins or gutters are applied to the right side of the columns, the last column needs to be stripped of the margin value; otherwise, the width of the page expands past the 960-pixel width.

You can remove this right-side margin through a class selector, which you can apply with other class attribute values:

```
<div class="container">
 <div class="span-24 last">
  Header
 </div>
 <div class="span-5">
  Side column
 </div>
 <div class="span-14">
  Main column
 </div>
 <div class="span-5 last">
  Side column
 </div>
</div>
```

To nest multiple columns within another column, use the same technique:

```
<div class="container">
 <div class="span-24 last">
  Header
 </div>
 <div class="span-5">
  Side column
 </div>
 <div class="span-14">
  <div class="span-5">
   Sidebar
  </div>
  <div class="span-9 last">
   Main column
  </div>
```

```
  <div class="span-14 last">
   Footer for main column
  </div>
 </div>
 <div class="span-5 last">
  Side column
 </div>
</div>
```

Discussion

Grid layouts have been used in print design for centuries and are often thought of as an earmark of good design.

Using prepared files for frameworks is a quick way to build cross-browser layouts with a grid layout. Typically, CSS frameworks include CSS Reset (see Recipe 5.2), a simple print stylesheet (see Chapter 11), basic type formatting (see Chapter 3), and, most importantly, a method for easily creating a column structure.

Detractors to frameworks cite their bloated source code, inflexible grid structure, and slower downtimes with an increased number of HTTP requests to download files as reasons to not use systems such as Blueprint.

> To change the width, number of columns, or size of the gutters within Blueprint CSS, use the online application at *http://kematzy.com/blue print-generator/*.

See Also

Blueprint CSS tutorials at *http://wiki.github.com/joshuaclayton/blueprint-css/tutorials*

13.12 Sample Design: A Cohesive Web Design

For this sample design, you will convert a web page design into an HTML and CSS design by using several of the solutions discussed in this book.

Setting the Page

The first step is to mark up the document with `div` elements:

```
<div id="header" class="">
 <h1><a href="/" title="Home Page">Kirkland Composition &amp;
Copyediting</a></h1>
</div><!-- /#header -->

<hr />

<div id="nav" class="">
 <div id="nav-site">
```

```
 <p class="hide">Content Navigation:</p>
 <ul>
  <li id="linkservices"><a href="/services/">Services</a></li>
  <li id="linkrates"><a href="/rates/">Rates</a></li>
  <li id="linkclients"><a href="/clients/">Clients</a></li>
  <li id="linksamples"><a href="/samples">Samples</a></li>
 </ul>
</div>
<div id="nav-info">
 <p class="hide">Information Navigation:</p>
 <ul>
  <li><a href="/about/">About</a></li>
  <li><a href="/guarentee/">Guarantee</a></li>
  <li><a href="/contact/">Contact</a></li>
 </ul>
</div><!-- /#nav-info -->
</div><!-- /#nav -->

<hr />

<div id="content" class="">
 <div id="article">
  [...]
 </div><!-- /#article -->
<div id="aside" class="">
  [...]
 </div><!-- /#aside -->
</div><!-- /#content -->

<hr />

<div id="footer" class="">
 [...]
</div>
```

> I used HTML5 elements as `id` attribute values where possible. As HTML5 gains wider adoption in browsers, it will be easier to convert completely to the HTML5 DOCTYPE.

The first step is to use a CSS reset (see Recipe 5.2) to remove out-of-the-box settings by the browser, as shown in Figure 13-12:

```
html, body, div, span, applet, object, iframe, h1, h2, h3, h4, h5, h6, p,
blockquote, a, abbr, acronym, big, font, img, small, center, dl, dt, dd, ol,
ul, li, fieldset, form, label, legend, table, tbody, tfoot, thead, tr, th, td {
 margin: 0;
 padding: 0;
 border: 0;
 outline: 0;
 font-size: 100%;
 vertical-align: baseline;
 background: transparent;
}
```

```
body {
 line-height:1;
}
ol, ul {
 list-style:none;
}
blockquote, q {
 quotes:none;
}
:focus {
 outline:0;
}
ins {
 text-decoration:none;
}
del {
 text-decoration:line-through;
}
table {
 border-collapse:collapse; border-spacing:0;
}
```
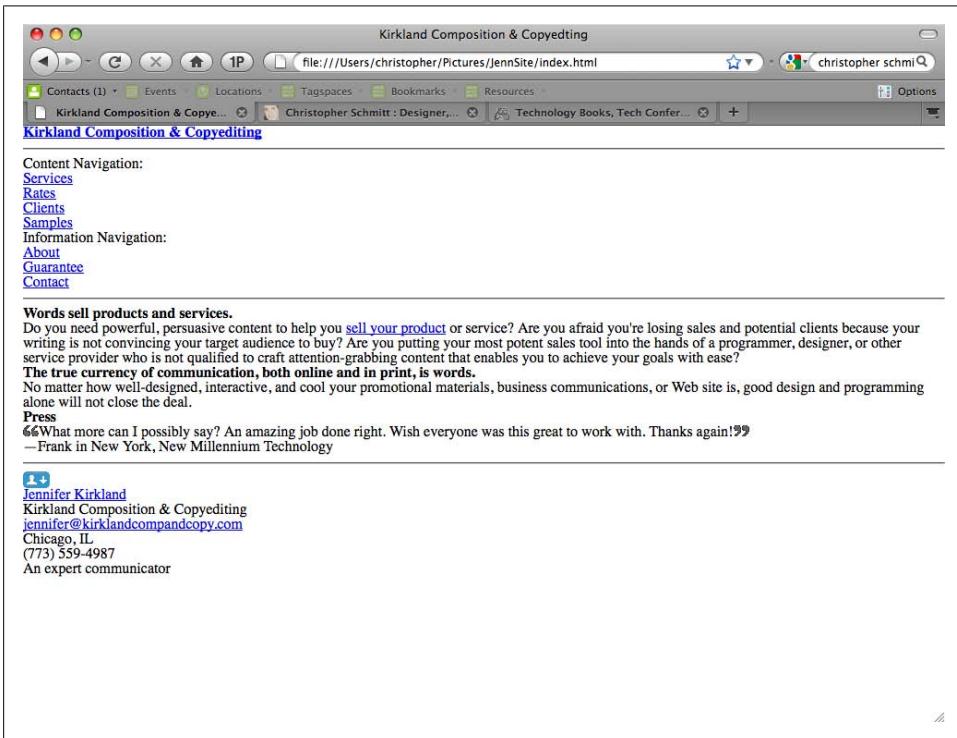

Figure 13-12. The unstyled content of the web page

Set up the elements for the body element, including background colors and images:

```
body {
 background-color: #000;
 color: rgba(255,255,255,.8);
 font-size: 62.5%;
 font-family: Times, "Times New Roman", Georgia, serif;
}
```

Constricting the Content

The next step is to create a reusable class selector for constricting the width of the content for the different sections of the web document, but letting the background images for the header and footer divisions extend beyond the width of the content:

```
.eight5x11 {
 width: 805px;
 margin: 0 auto;
}
```

Then position these constrictors in between the major divisions of the page:

```
<div id="header" class="">
 <div class="eight5x11">
  <h1><a href="/" title="Home Page">Kirkland Composition &amp;
Copyediting</a></h1>
 </div><!-- /.eight5x11 -->
</div><!-- /#header -->

<hr />

<div id="nav" class="">
 <div class="eight5x11">
  <div id="nav-site">

   <p class="hide">Content Navigation:</p>
   <ul>
    <li id="linkservices"><a href="/services/">Services</a></li>
    <li id="linkrates"><a href="/rates/">Rates</a></li>
    <li id="linkclients"><a href="/clients/">Clients</a></li>
    <li id="linksamples"><a href="/samples">Samples</a></li>
   </ul>
  </div>
  <div id="nav-info">
   <p class="hide">Information Navigation:</p>
   <ul>
    <li><a href="/about/">About</a></li>
    <li><a href="/guarantee/">Guarantee</a></li>
    <li><a href="/contact/">Contact</a></li>
   </ul>
  </div><!-- /#nav-info -->
 </div><!-- /.eight5x11 -->
</div><!-- /#nav -->

<hr />
```

```
<div id="content" class="">
 <div class="eight5x11">
  <div id="article">
    [...]
  </div><!-- /#article -->
  <div id="aside" class="">
   [...]
  </div><!-- /#aside -->
 </div><!-- /.eight5x11 -->
</div><!-- /#content -->

<hr />

<div id="footer" class="">
 <div class="eight5x11">
  [...]
 </div><!-- /.eight5x11 -->
</div><!-- /#footer -->
```

Working on the first section in the document, the main header, apply the background image:

```
/* Logo */
#header {
 background-image: url(img/bkgd-header.gif);
 background-repeat: no-repeat;
 background-position: top center;
 margin-top: 30px;
}
```

Since the constrictor class is applied in between the main heading `div` element and the content, you can apply an image that is larger than the width of the content without causing a horizontal scroll bar to appear if the user makes the window shorter.

Bringing in the Logo

You can bring in the logo image through the link of the main title. Setting the value to display as a block (see Recipe 7.10) allows you to set the width and height of the `a` element to let the logo in through the background.

Then, using relative positioning, position the logo to the upper-lefthand corner a bit, as shown in Figure 13-13:

```
/* Logo */
#header {
 background-image: url(img/bkgd-header.gif);
 background-repeat: no-repeat;
 background-position: top center;
 margin-top: 30px;
}
#header h1 a {
 background-image: url(img/logo.png);
 background-repeat: no-repeat;
 background-position: top center;
 display: block;
```

```
    width: 456px;
    height: 270px;
    text-indent: -9999em;
    position: relative;
    left: -78px;
    top: -30px;
    border: none;
}
```
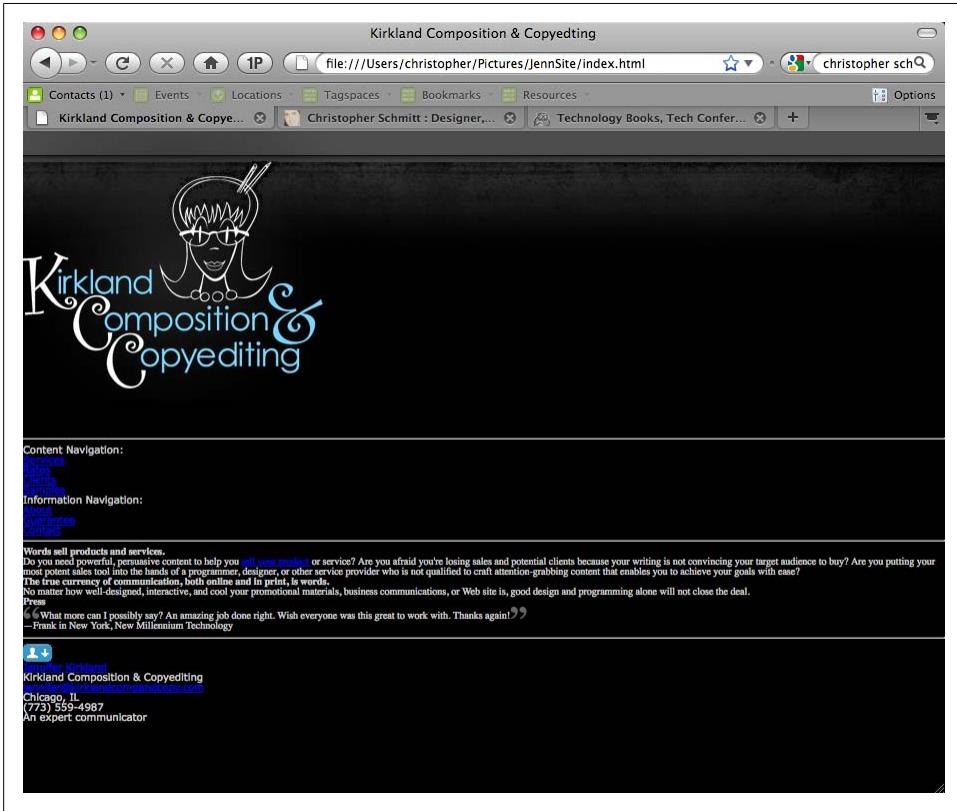

Figure 13-13. The logo making an appearance

Next, remove the hr elements and underlining of the text links:

```
/* Common Elements */
.hide, hr {
  display: none;
}
a {
  text-decoration: none;
}
```

The Navigation Menu

Position the main navigation menu into place:

```
#nav-site {
 width: 433px;
 margin-left: 380px;
 position: relative;
 top: -95px;
}
#nav-site ul, #nav-site ul li {
 display: inline;
}
```

> Another way to position the logo and the navigation menu is to float both elements to the left and then use relative positioning to move them to the desired location on the page.

After the unordered list is in position, it's time to create a horizontal menu (see Recipe 7.12). For this approach, again set the links to display as block-level elements, hide the text, and make sure the background images don't repeat:

```
#nav-site ul li a {
 display: block;
 float: left;
 text-indent: -9999em;
 background-repeat: no-repeat;
}
```

With the main building blocks of the navigation set up, create a CSS sprite image for the navigation menu that includes normal, rollover, and on states, as shown in Figure 13-14.

Figure 13-14. Navigation CSS sprite image

Then, set up each link with its own:

```
#nav-site ul li#linkservices a {
 width: 118px;
 height: 39px;
 background-image: url(img/nav.gif);
}
```

```
#nav-site ul li#linkrates a {
 width: 88px;
 height: 39px;
 background-image: url(img/nav.gif);
 background-position: -118px 0;
}
#nav-site ul li#linkclients a {
 width: 107px;
 height: 39px;
 background-image: url(img/nav.gif);
 background-position: -206px 0;
}
#nav-site ul li#linksamples a {
 width: 120px;
 height: 39px;
 background-image: url(img/nav.gif);
 background-position: -313px 0;
}
```

For each rollover effect, move the background position to show the rollover state:

```
#nav-site ul li#linkservices a:link:hover,
#nav-site ul li#linkservices a:visited:hover {
 background-position: 0 -39px;
}
#nav-site ul li#linkrates a:link:hover,
#nav-site ul li#linkrates a:visited:hover {
 background-position: -118px -39px;
}
#nav-site ul li#linkclients a:link:hover,
#nav-site ul li#linkclients a:visited:hover  {
 background-position: -206px -39px;
}
#nav-site ul li#linksamples a:link:hover,
#nav-site ul li#linksamples a:visited:hover {
 background-position: -313px -39px;
}
```

With the links in place, set up a contextual menu that shows which page is currently being viewed, as shown in Figure 13-15:

```
#pageservices #nav-site ul li#linkservices a {
 background-position: 0 -78px;
}
#pagerates #nav-site ul li#linkrates a {
 background-position: -118px -78px;
}
#pageclients #nav-site ul li#linkclients a {
 background-position: -206px -78px;
}
#pagesamples #nav-site ul li#linksamples a {
 background-position: -313px -78px;
}
```
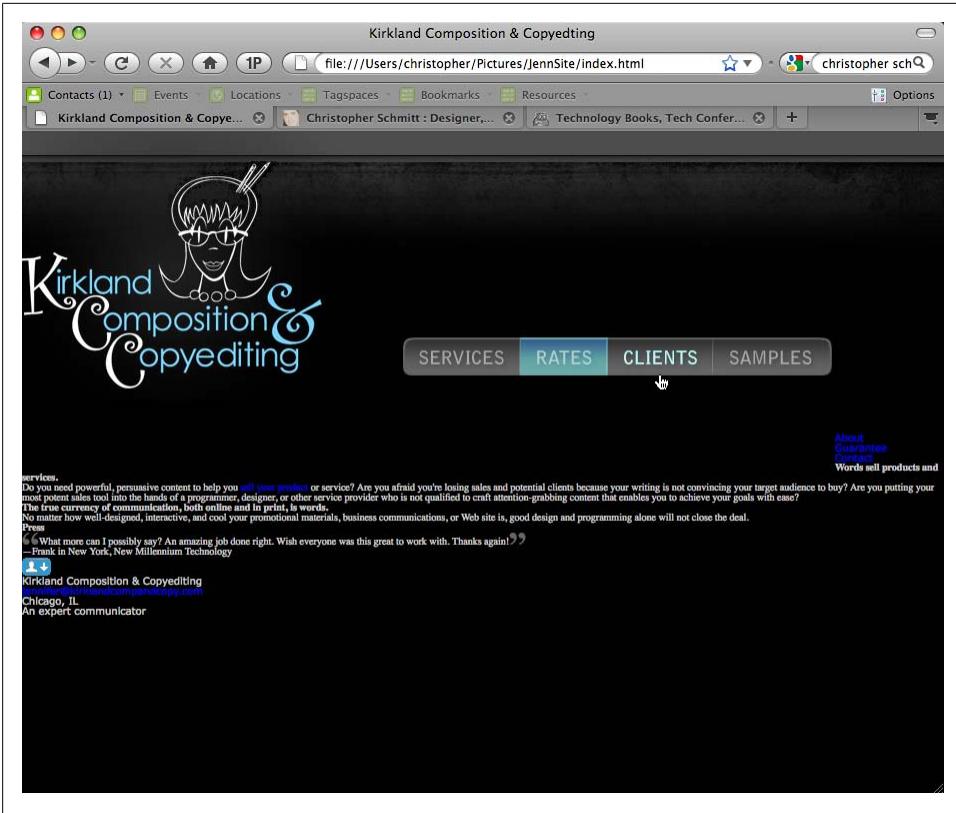
Figure 13-15. The finalized navigation menu

> For the contextual menu to work, an `id` attribute was added to the `body` element with a value of `rates`:
>
> ```
> <body id="pagerates">
> [...]
> </body>
> ```

The Secondary Link Menu

With the main navigation in place, move the text links at the top of the page through absolute positioning and set the individual list items to create a horizontal row:

```
/* Top Navigation Links */
#nav-info {
 position: absolute;
 width: 100%;
 top: 0;
 left: 0;
 font-size: 1.2em;
 height: 30px;
```

```
  }
#nav-info ul {
  width: 805px;
  margin: 0 auto;
  text-align: right;
}
#nav-info ul li {
  display: inline;
  padding-left: 1.2em;
  line-height: 1.9;
  text-transform: lowercase;
}
```

Styling the Content Column

With the header and navigation elements set, you can move onto the content area of the page. First, clear both sides of the float to keep the navigation menu from causing content to wrap around it, and apply a color to the text:

```
/* Content */
#content {
  clear: both;
  color: rgba(255,255,255,.8);
}
```

Now float the main column of text to the right while setting off enough whitespace at the bottom of the column. This whitespace acts as a buffer between the main content area and the footer:

```
/* Content */
#content {
  clear: both;
  color: rgba(255,255,255,.8);
}
#content #article {
  float: right;
  width: 585px;
  margin-bottom: 70px;
}
```

With the main column in position, adjust the line heights on the headings and paragraphs so that they all share the same leading:

```
/* Content */
#content {
  clear: both;
  color: rgba(255,255,255,.8);
}
#content #article {
  float: right;
  width: 585px;
  margin-bottom: 70px;
}
#content #article h2 {
  font-size: 2.4em;
```

```
 font-weight: normal;
 line-height: .9998;
}
#content #article h3 {
 font-size: 1.6em;
 font-weight: bold;
 line-height: 1.4975;
 margin-top: 1.6em;
}
#content #article p {
 font-size: 1.4em;
 line-height: 1.714;
 font-family: Verdana, Geneva, Tahoma, sans-serif;
}
```

The first word of the main heading has a capital *W* followed by a lowercase *o*. There is a little too much spacing, or kerning, between the letters. To adjust the spacing of the letters (see Recipe 3.33), use a `:first-letter` pseudo-element (see Recipe 3.13) to reduce the spacing on just the first letter (as shown in Figure 13-16):

```
/* Content */
#content {
 clear: both;
 color: rgba(255,255,255,.8);
}
#content #article {
 float: right;
 width: 585px;
 margin-bottom: 70px;
}
#content #article h2 {
 font-size: 2.4em;
 font-weight: normal;
 line-height: .9998;
}
#content #article h3 {
 font-size: 1.6em;
 font-weight: bold;
 line-height: 1.4975;
 margin-top: 1.6em;
}
#content #article p {
 font-size: 1.4em;
 line-height: 1.714;
 font-family: Verdana, Geneva, Tahoma, sans-serif;
}
#content #article h2:first-letter {
 letter-spacing: -.1em;
}
```
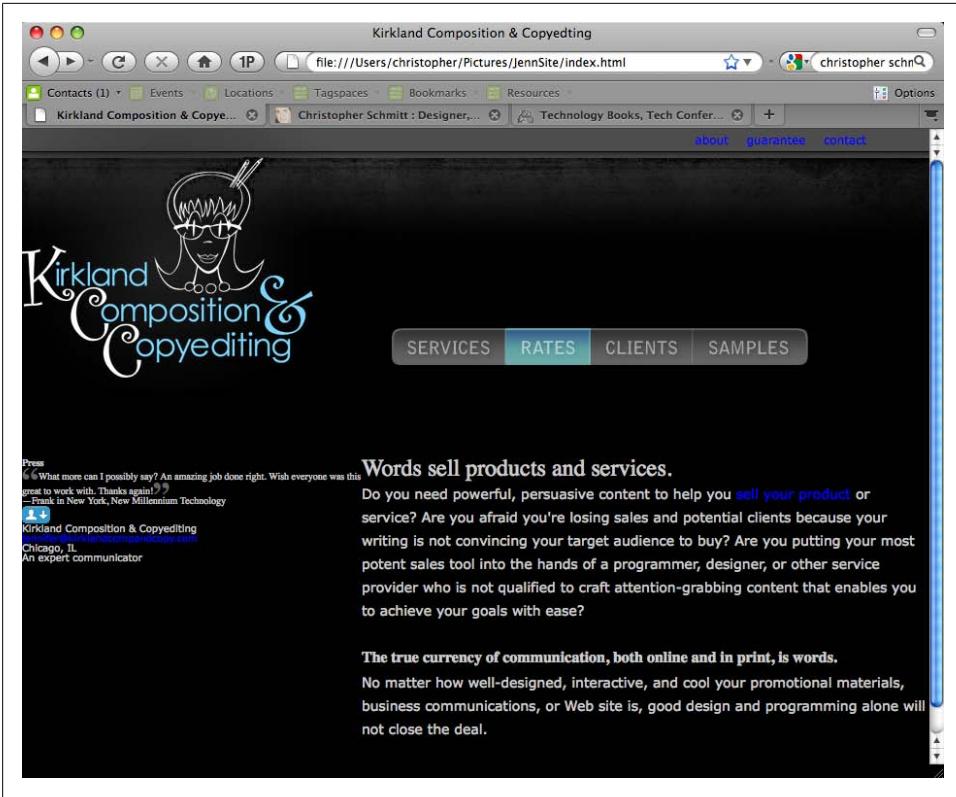
Figure 13-16. Main content column, styled

Working on the Smaller Column

With the main content column styled, it's time to work on the left column. Using the margin you set up, in essence, the width of the side column:

```
#content #aside {
 margin-right: 650px;
 color: #999;
}
```

> Instead of using margins that extend the width of the content column, you can set up a two-column layout by floating the side column to the right and then applying padding to the right side to set some space or a gutter between the two columns.

Then adjust the styles of the text, while keeping the leading for the same as the main content column. By sharing the same leading, both columns of text use the same baseline:

```
#content #aside {
 margin-right: 650px;
 color: #999;
}
#content #aside h4 {
 font-size: 1.4em;
 text-transform: uppercase;
 font-style: italic;
 letter-spacing: .3em;
 line-height: 2.3996;
 margin-bottom: -.7em;
}
#content #aside p {
 font-size: 1.4em;
 line-height: 1.714;
}
```

Using an adjacent sibling selector (see Recipe 2.4), adjust the second paragraph to be in italics:

```
#content #aside {
 margin-right: 650px;
 color: #999;
}
#content #aside h4 {
 font-size: 1.4em;
 text-transform: uppercase;
 font-style: italic;
 letter-spacing: .3em;
 line-height: 2.3996;
 margin-bottom: -.7em;
}
#content #aside p {
 font-size: 1.4em;
 line-height: 1.714;
}
#content #aside p+p {
 font-size: 1.4em;
 font-style: italic;
 line-height: 1.714;
}
```

With the content text styled, apply a set of links just for this portion of the document, as shown in Figure 13-17:

```
#content a:link, #content a:visited {
 color: rgba(51,153,204,.8);
 border-bottom: 1px solid #3399cc;
 border-bottom: 1px solid rgba(51,153,204,.8);
}
#content a:link:hover, #content a:visited:hover {
 color: rgba(51,153,204,.8);
 border-bottom: none;
}
```
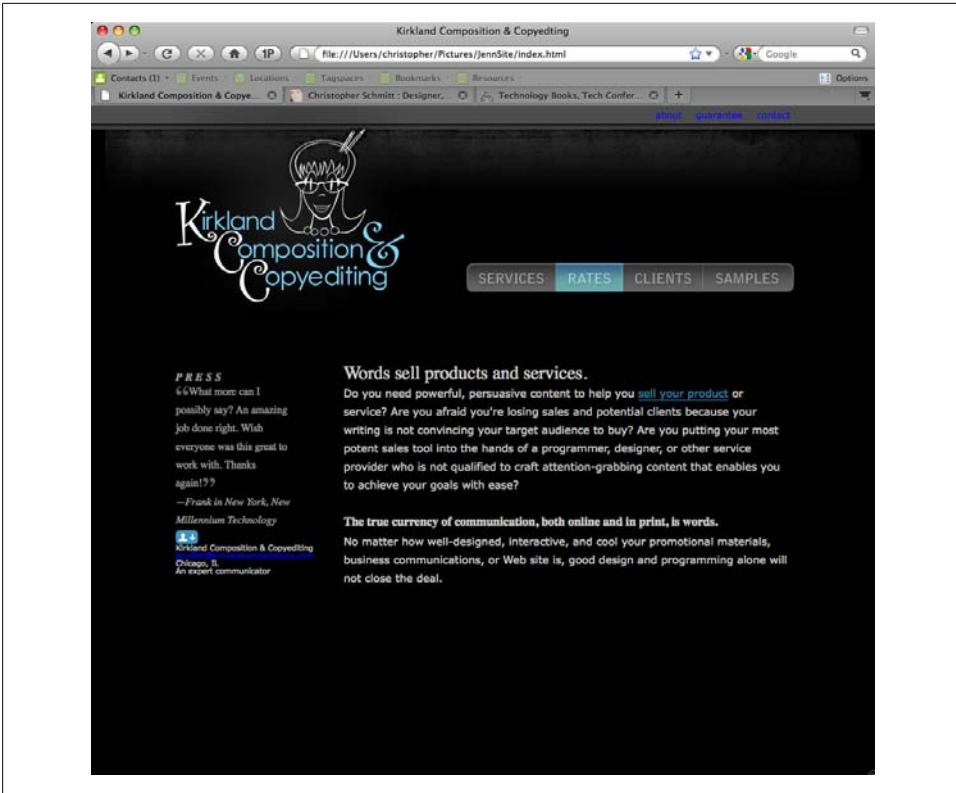
Figure 13-17. Styling the links

Finishing with the Footer

The final area of the web page is the footer. Place the background image for the bottom of the page and push off any content within the footer 25 pixels from the top to let the image breathe a bit:

```
#footer {
 background-image: url(img/bkgd-footer.gif);
 background-repeat: no-repeat;
 background-position: top center;
 font-size: 1.4em;
 line-height: 1.4;
 clear: both;
 padding-top: 25px;
 color: #666;
}
```

Next, style the microformat hCard to allow the download icon to appear to the left side of the page:

```
#footer .vcard {
 padding-left: 40px;
```

```
}
#footer .org {
 color: #999;
}
#footer img {
 float: left;
}
```

Place the tagline for the company on the right side of the page by using absolute positioning, as shown in Figure 13-18:

```
#footer .tagline {
 position: absolute;
 right: 0;
 top: 0;
 font-family: Times, "Times New Roman", Georgia, serif;
 letter-spacing: .4em;
 font-style: italic;
 font-weight: bold;
}
```
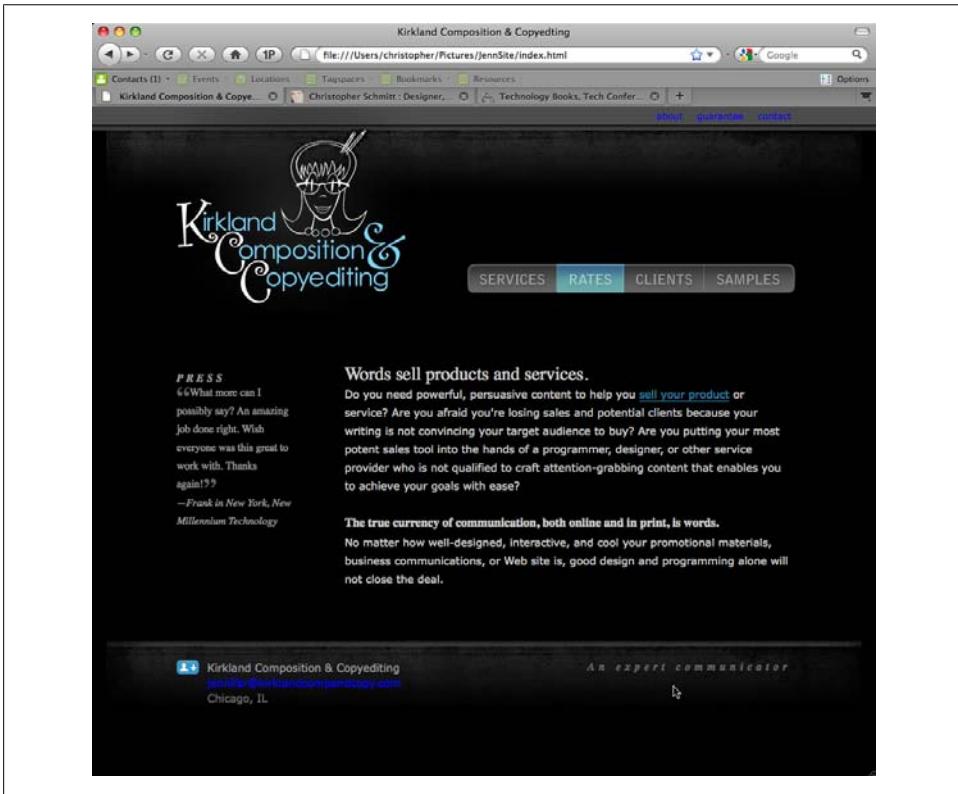

Figure 13-18. The footer's tagline, positioned

The Final Touches

For the final touch, set the links for other areas of the page to be white, as shown in Figure 13-19:

```
a:link, a:visited {
 color: rgba(255,255,255,.8);
 border-bottom: 1px solid #fff;
 border-bottom: 1px solid rgba(255,255,255,.8);
}
a:link:hover, a:visited:hover {
 color: rgba(255,255,255,.8);
 border-bottom: none;
}
```
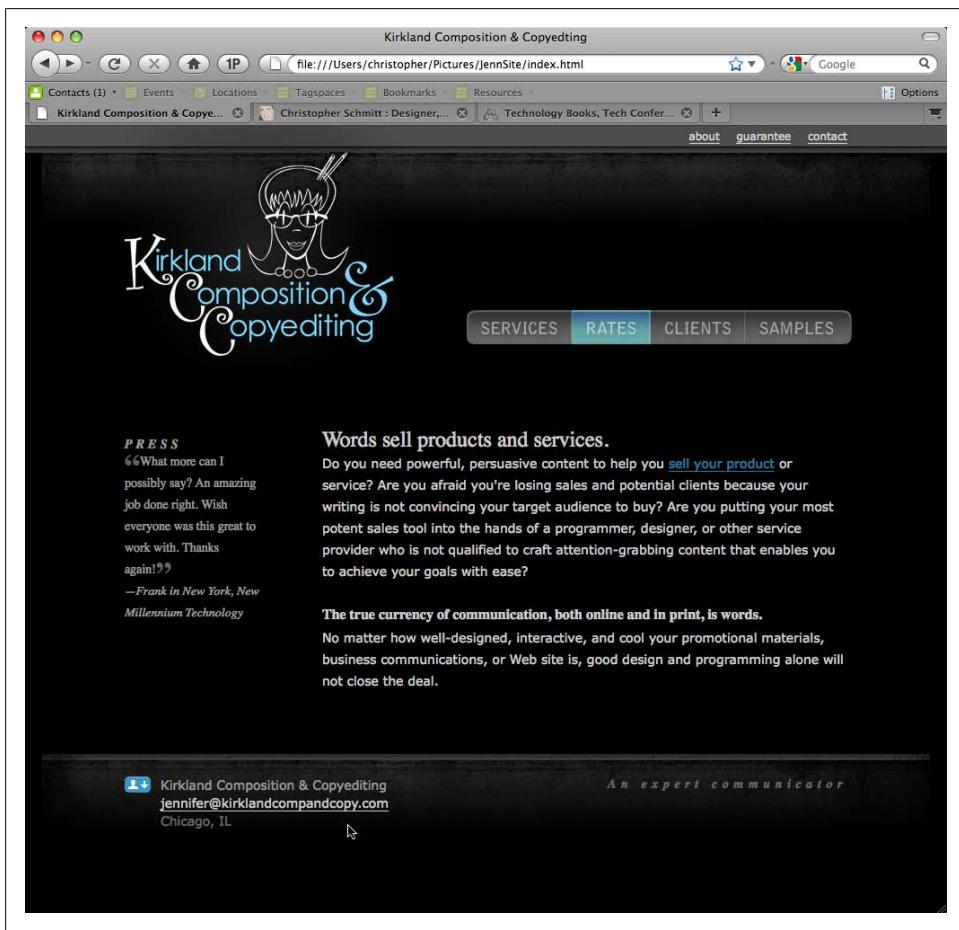

Figure 13-19. The rest of the links, styled

And with that last rule, the web page is styled for modern browsers, but what about earlier versions of IE? The page design does break in those browsers. To deliver CSS patches for those browsers, use conditional comments (see Recipe 12.7):

```
<!--[if lt IE 8]>
<style type="text/css">
#content {
 color: #fff;
}
a:link, a:visited {
 color: #fff;
 border-bottom: 1px solid #fff;
}
a:link:hover, a:visited:hover {
 color: #fff;
 border-bottom: none;
}
#content a:link, #content a:visited {
  color: #3399cc;
  border-bottom: 1px solid #3399cc;
}
#content a:link:hover, #content a:visited:hover {
  color: #3399cc;
  border-bottom: none;
}
</style>
<![endif]-->
```

The next steps would be to create a print stylesheet (see Recipe 10.1) as well as making a printable version of the logo (see Recipe 10.2). I leave those steps as homework for you.

13.13 Sample Design: The U.S. Flag

The goal of this exercise is to create a U.S. flag through CSS and semantic markup that is not only visually appealing, but also useful so that clicking on any of the stars will take you to a different state's main government page and clicking on any of the stripes will take you to the government sites of the original 13 colonies.

The Basic Markup

First, lay the groundwork for the flag by constructing markup. Include a title for the page, as well as the 50 states represented in the U.S. flag.

Wrap the title in a header tag and a list for the 50 states:

```
<h1>
 <a href="http://www.usa.gov/">United States of America</a>
</h1>
<ol>
 <li><a href="http://www.alabama.gov/">
  <strong>State of Alabama</strong>
```

```
  <i></i>
</a></li>
<li><a href="http://www.state.ak.us/">
 <strong>State of Alaska</strong><i></i>
</a></li>
[...]
</ol>
```

The heading and each individual state has a link to its respective government site. Also included are extra sets of italics tags within each link, which you pick up as a selector for styling later.

You need to distinguish the 13 colonies from the rest of the states, as these comprise the 13 stripes of the flag. Do this by emphasizing them, wrapping them in the `em` tag:

```
<li><a href="http://www.ct.gov/">
 <em><strong>State of Connecticut</strong></em>
 <i></i>
</a>
</li>
```

So far, the results don't look very much like a flag, as shown in Figure 13-20. That's where the CSS comes in. Now that you have the basic markup down, begin styling your page.
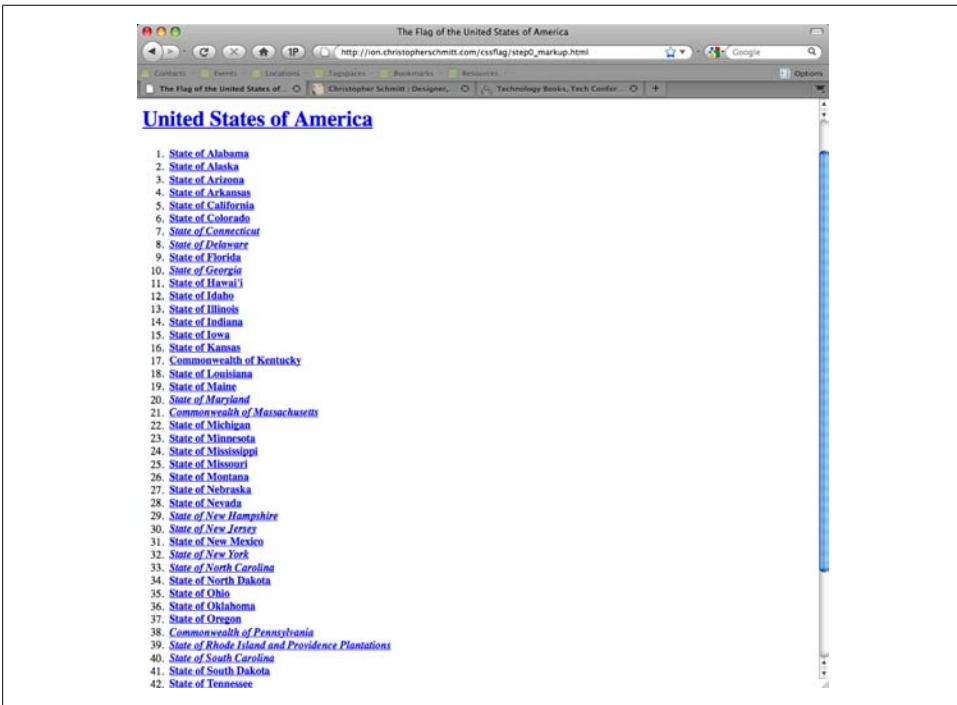
Figure 13-20. The default rendering of the content

Creating the Easel

First, you want to create the "easel" upon which you'll draw your flag. You'll do this by first structuring your HTML further. Wrap all of the page's content within a `div` tag, and assign it an `id` attribute with a value of `easel`:

```
<div id="easel">
 <h1>
  <a href="http://www.usa.gov/">United States of America</a>
 </h1>
 <ol>
  <li><a href="http://www.alabama.gov/">
   <strong>State of Alabama</strong><i></i>
  </a></li>
  [...]
 </ol>
</div>
```

Next, you need to style this `div` to create a canvas. How do you do this? Take a look at the CSS:

```
body {
 margin: 10px 0 0 0;
 padding: 0;
}
#easel {
 width: 955px;
 margin: 0 auto;
 position: relative;
}
```

First you want to set the width of the canvas, and then you want to center it.

To center the canvas for the flag on the page, as shown in Figure 13-21, set the margins on either side of the "easel" `div` to `auto`. The browser automatically splits the margin in half and assigns its value to the left and right.

Now, clear the text from the easel (i.e., create a blank canvas on which to "paint" the flag) by adding the following CSS rules:

```
#easel ol {
 list-style: none;
 margin: 0;
 padding: 0;
}
#easel strong {
  display: none;
}
```

In the preceding code, first you got rid of the preceding bullets as well as the default margins and padding. Then you hid all of the text within the `strong` tag, which includes all of the list items, as shown in Figure 13-22.

Figure 13-21. Centering the content
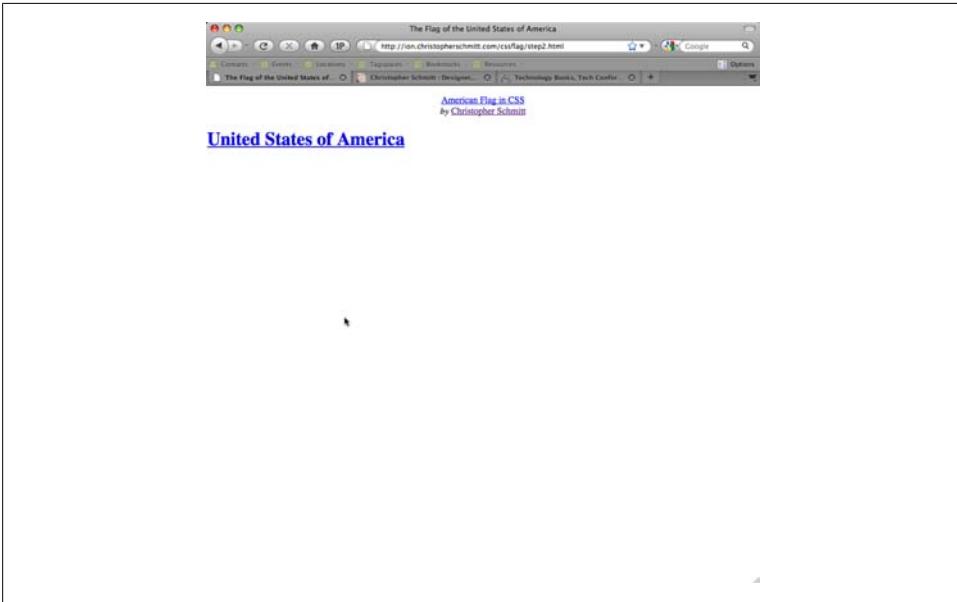
Figure 13-22. Hiding the content

Creating Stripes

Now it's time to paint the flag. You'll start with the stripes.

Remember that the 13 colonies, which are represented by the stripes on the flag, are distinguished from the other states through the `em` tag.

So, you can create the stripes by styling this selector:

```
#easel em {
  width: 955px;
  height: 50px;
  display: block;
  background: #BF0A30;
  position: absolute;
  top: 0;
  left: 0;
  z-index: 1;
}
```

At this stage, you have set the width and height of the flag, as shown in Figure 13-23. And after setting a background color, you turned the selector into a block-level element so that you can position it on the page. You are absolutely positioning your stripes; this means you are offsetting stripes against the edges of the parent element.

Alternatively, relative positioning places the stripes relative to the element's default position on the page. So, here you are placing it in the upper-lefthand corner of the parent element, the "easel" `div`: you place it zero pixels from the top and zero pixels from the left of the edges of the containing element.

The `z-index` property governs the "third dimension" of the elements on the web page. That is, it specifies the elements' stacking order. The higher an element's `z-index` number, the higher it is in the stacking order.

For example, an element with a `z-index` value of 5 is placed on top of an element with a `z-index` value of 1. You want the stripes to be placed beneath the blue field of stars, so you give that a low `z-index` value of 1.

Thus far, it looks like you have only one stripe, as shown in Figure 13-23. Actually, all 13 stripes are on the page, but they are all placed in the exact same position—and they are all red. To distinguish the 13 stripes, you have to position and color each stripe individually.

To pick out each specific stripe for styling, first note what attributes individuate each state in the markup:

```
<li><a href="http://www.ct.gov/">
 <em>State of Connecticut</em>
 <i></i>
</a>
</li>
```
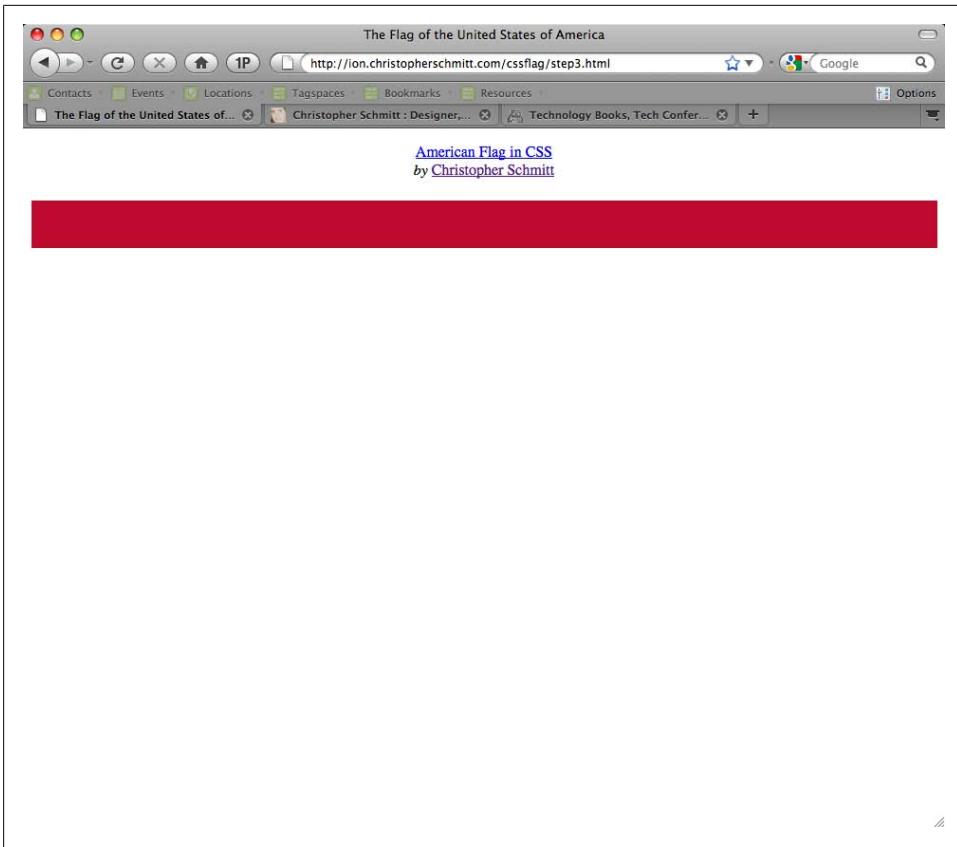
Figure 13-23. The first red stripe

Each state is marked uniquely with a different link address. You can use the `href` attributes to create selectors for each stripe (see Recipe 2.5):

```
#easel a[href="http://delaware.gov/"] em {
  background: white;
 top: 50px;
 left: 0;
}
#easel a[href="http://www.georgia.gov/"] em {
 top: 100px;
 left: 0;
}
[...]
```

Since you originally set the background color for the `em` selector as red, you need to change this color for all the white stripes.

Then, you just need to move each stripe from the top of the page into its designated spot. Each stripe is 50 pixels high, so the second stripe, in this case the one for Delaware, needs to be placed 50 pixels down. The next stripe is placed 100 pixels down, and so on.

Once you have finished coloring and positioning all 13 stripes as needed, your flag should look like Figure 13-24.
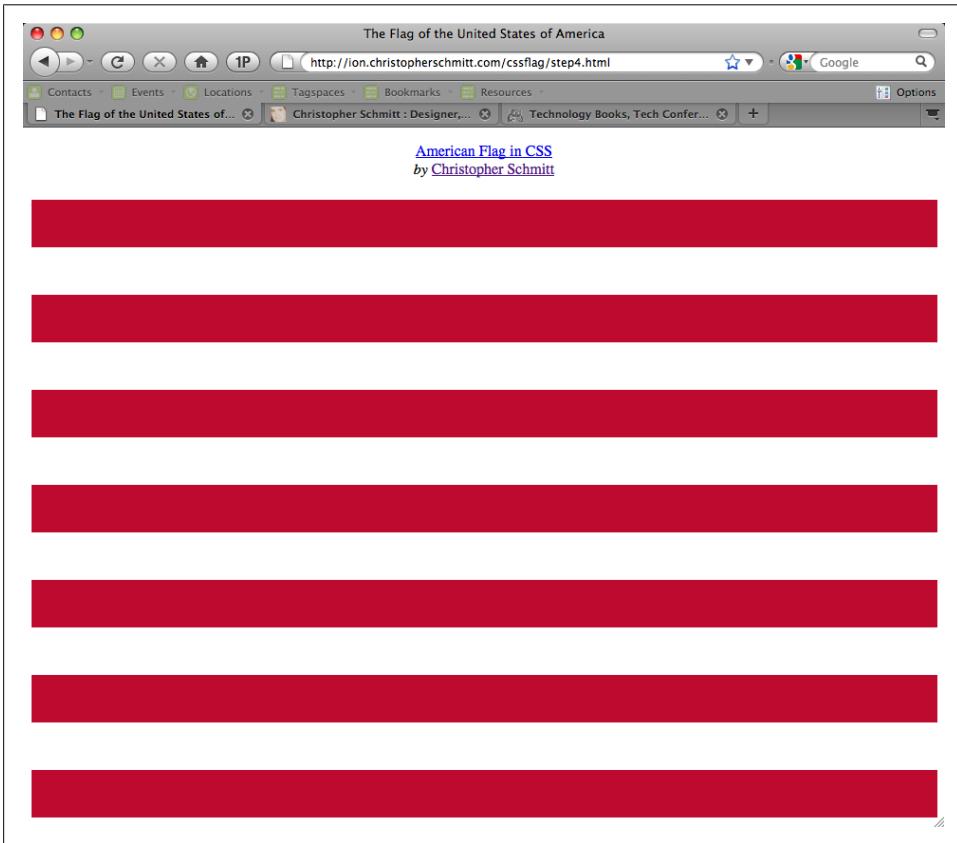
Figure 13-24. All 13 stripes

Creating Stars

Now that your stripes are in place, it's time to move on to the stars. First, you need to create the blue field upon which the stars are placed. To do this, you're going to transform the page header. To turn the header into a blue box, you need to hide the header text, then size, position, and color the anchor selector within the header element:

```
#easel h1 a {
  position: absolute;
  width: 215px;
  height: 175px;
  background: #002868;
  text-indent: -9999em;
  margin: 0;
  padding: 0;
```

```
    z-index: 20;
  }
```

Absolutely positioning the element without any offset properties places the element in the upper-lefthand corner of the "easel" `div`, which is the containing element.

After setting the width, height, and background color, you have effectively "hidden" the header text by indenting it far off the page.

Finally, by setting the `z-index` value to 20, you've stacked it on top of the stripes, as shown in Figure 13-25.
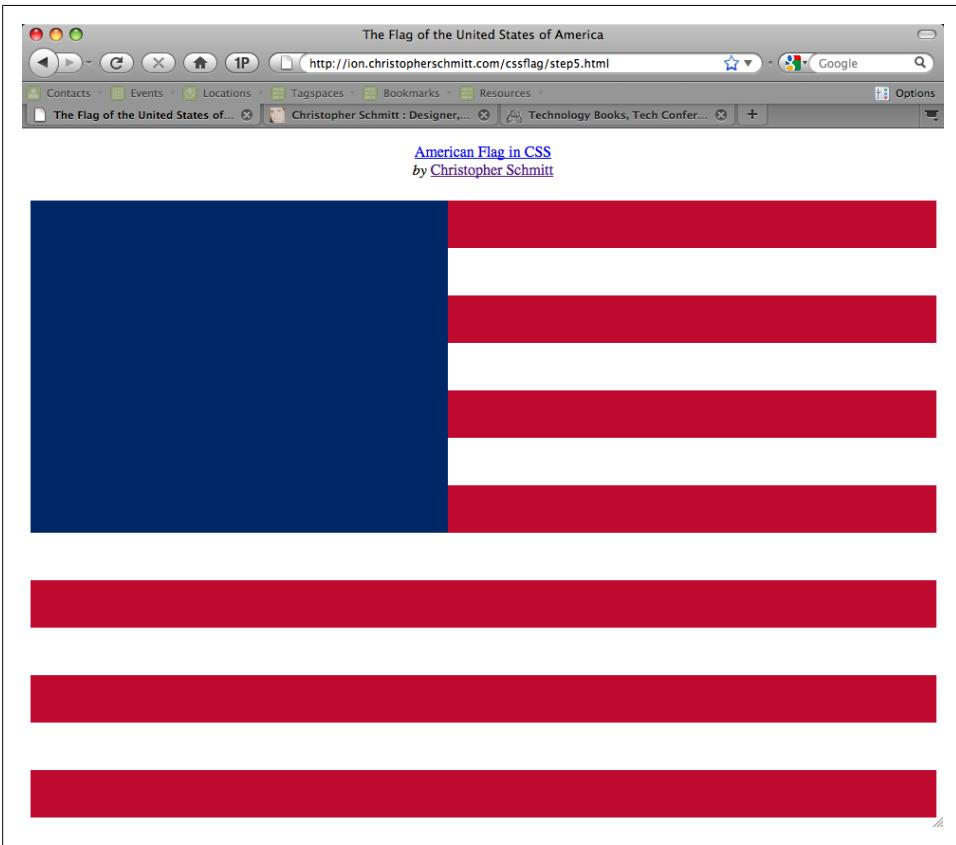
Figure 13-25. The blue union

To create stars as shown in Figure 13-26, employ a technique similar to the method for creating the stripes. Select each state in your CSS through their unique `href` attribute, and then style accordingly.
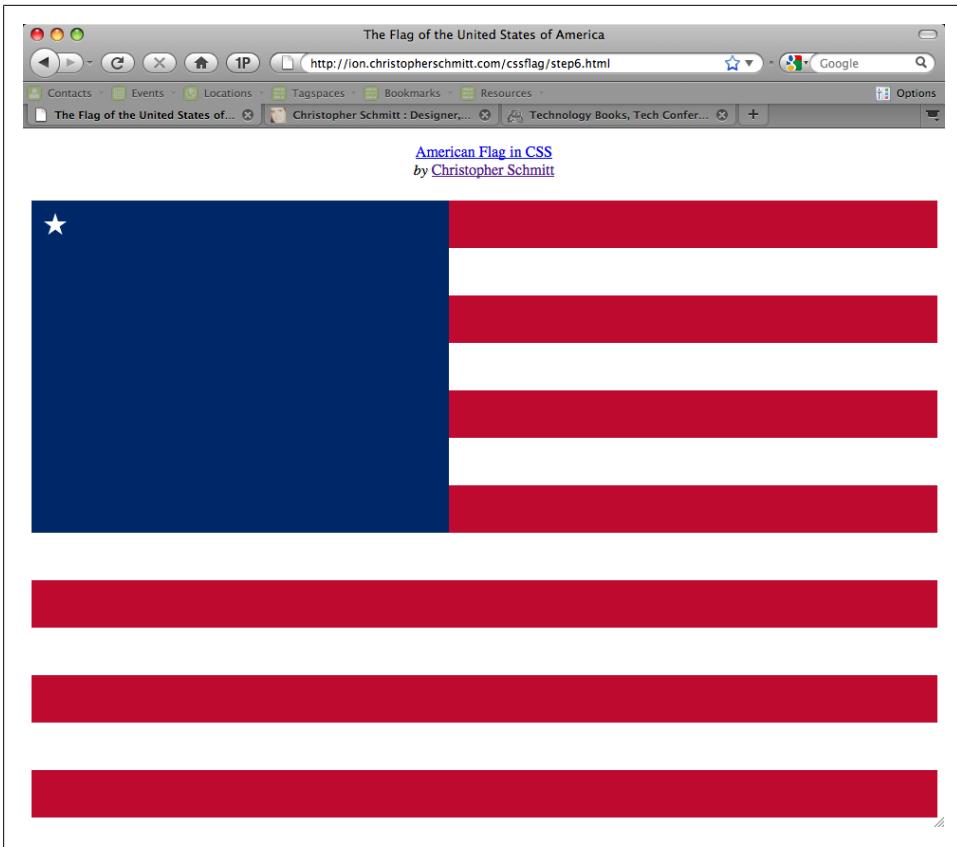
Figure 13-26. The first state

You don't want to style over the same element that makes up your 13 stripes, so you have to use a different selector than the anchor. Use the `italic` tag as your selector for each state, as shown in Figure 13-27:

```
#easel ol li a[href="http://www.alabama.gov/"] i {
 background-image: url(stars.gif);
 display: block;
 position: absolute;
 top: 13px;
 left: 13px;
 z-index: 50;
 width: 24px;
 height: 23px;
}
#easel ol li a[href="http://www.state.ak.us/"] i {
 background-image: url(stars.gif);
 display: block;
 position: absolute;
 top: 13px;
```

```
    left: 90px;
    z-index: 50;
    width: 24px;
    height: 23px;
}
[...]
```
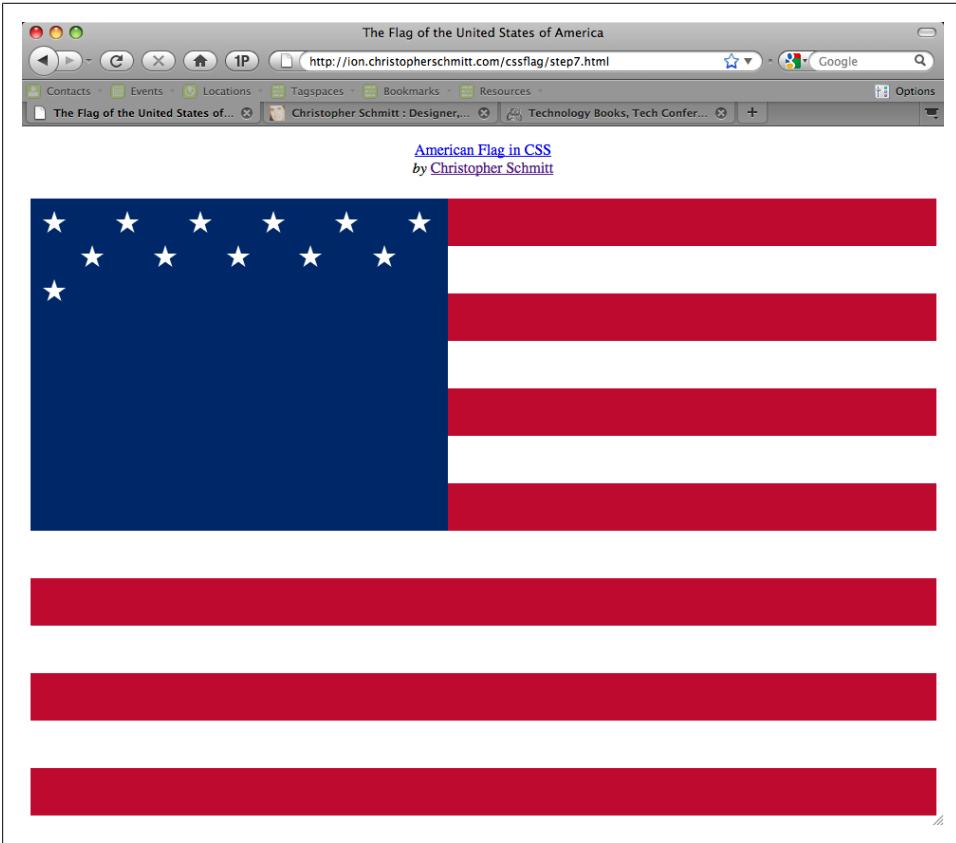

Figure 13-27. The stars starting to flow

The star itself is a small GIF image, as shown in Figure 13-28.

Figure 13-28. The star image

You point to this image in the `background-image` property. Then, you position this star by moving the element from the top-lefthand corner of the containing `div`. Each state's star is positioned differently, according to its place on the blue field. You also set its `z-index` value to 50, to place it on top of both the stripes and the blue field it is set against.

Once you've finished the tedious task of applying CSS to all 50 states, you have a completed U.S. flag, as shown in Figure 13-29.
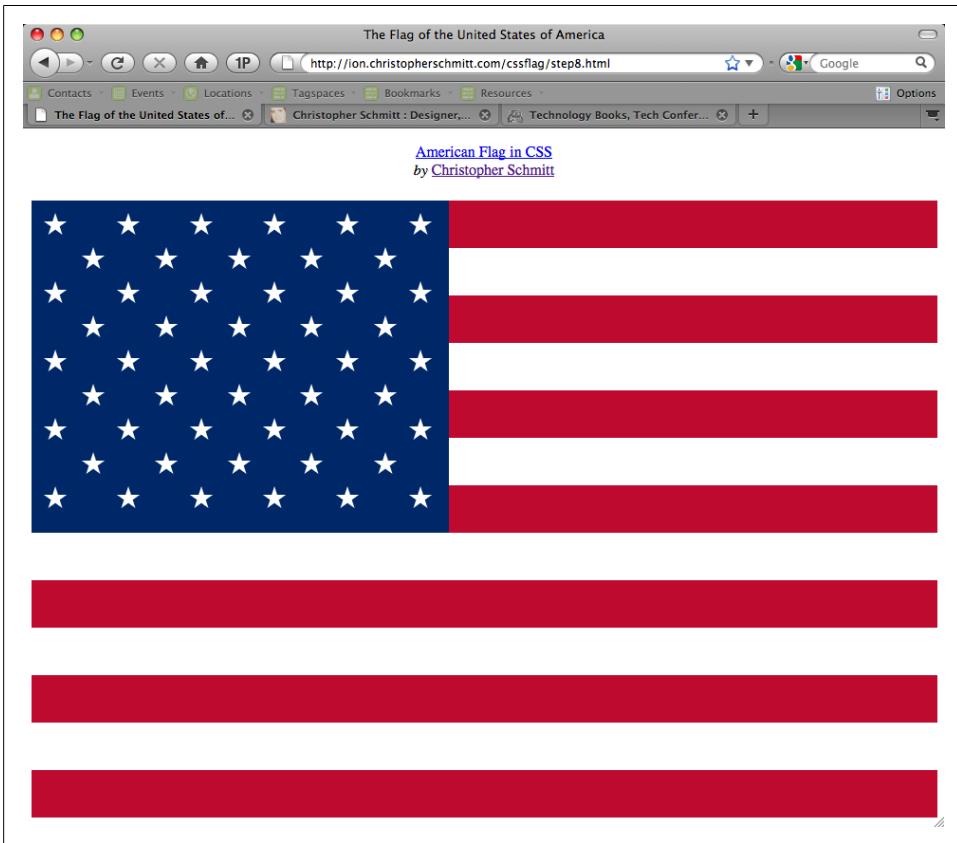
Figure 13-29. A basic U.S. flag

Adding Texture

Why don't you add some additional visual interest to your creation by adding some subtle texturing effects? You can do this easily by strategically placing some transparent PNGs as background images to your elements.

PNGs support alpha transparency, which allows portions of an image to be partially transparent. (See Recipe 4.17 for how to include PNGs in a browser.) This allows for some very interesting effects, such as creating the illusion of a "textured" surface.

Place a transparent image such as the one shown in Figure 13-30 behind the elements comprising your flag. First, place the image behind your red stripes:

```
#easel em {
  width: 955px;
  height: 50px;
  display: block;
  background: #BF0A30;
  position: absolute;
  top: 0;
  left: 0;
  z-index: 1;
  background-image: url(flag_bkgd.png);
  background-position: 50%;
  background-repeat: no-repeat;
  background-attachment: fixed;
}
```
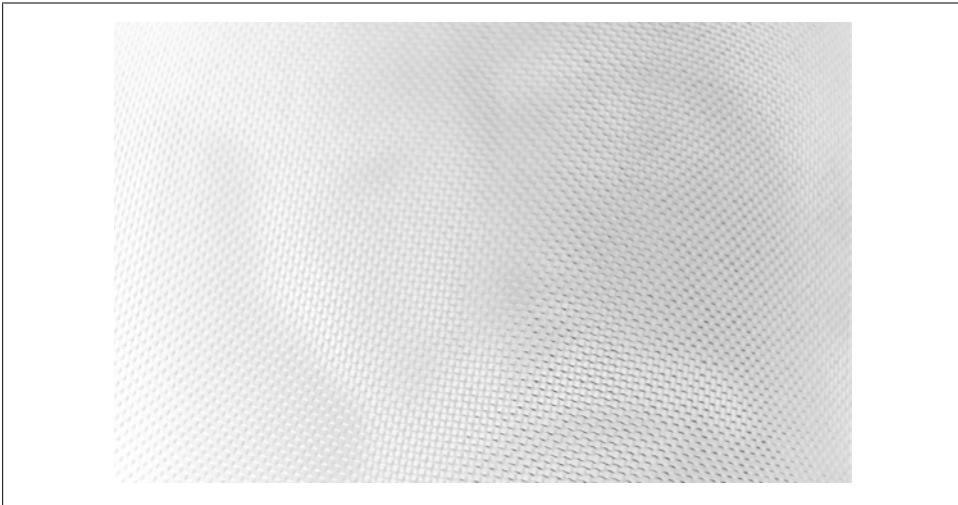

Figure 13-30. Background texture

You are placing the image halfway from the left and top edges of the element, the stripe. The `background-repeat` property specifies that the image is not tiled; that is, the image is not repeated vertically or horizontally.

Do the same for the white stripes:

```
#easel a[href="http://delaware.gov/"] em {
  background: white;
  top: 50px;
  left: 0;
  background-image: url(flag_bkgd.png);
  background-position: 50%;
  background-repeat: no-repeat;
  background-attachment: fixed;
}
```

And lastly, repeat the process for the blue field:

```
#easel h1 a {
 position: absolute;
 width: 215px;
 height: 175px;
 background: #002868;
 text-indent: -9999em;
 margin: 0;
 padding: 0;
 z-index: 20;
 background-image: url(flag_bkgd.png);
 background-position: 50%;
 background-repeat: no-repeat;
 background-attachment: fixed;
}
```

This final bit of CSS results in a U.S. flag, as shown in Figure 13-31, that works in IE7 and later, Firefox 2 and later, Opera 9.5, and Safari.
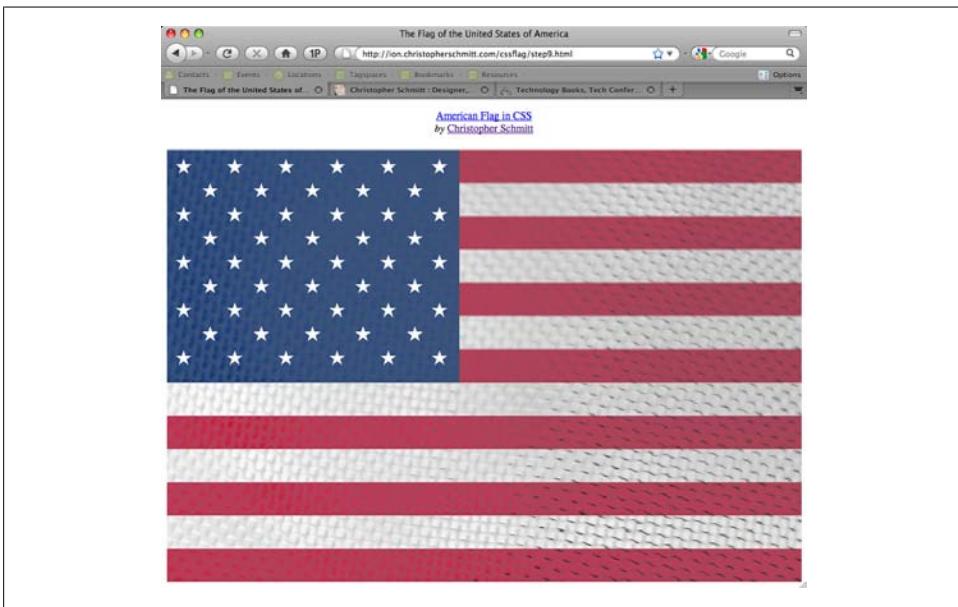
Figure 13-31. The final version of the flag

> Feel free to try your own version of the U.S. flag or a different country's flag. It's a great exercise in utilizing and honing your skills in many aspects of CSS.

Interacting with JavaScript

14.0 Introduction

JavaScript is part of a three-legged stool when it comes to building a successful website.

Web developer Steven Champeon compared the relationship of web technologies to the building blocks of the English language: JavaScript is the verb, while HTML is the noun and CSS is the adjective.

With HTML as the base describing the content through semantic markup, you can efficiently apply stylesheets to render a good-looking site.

JavaScript allows action to take place on the web page. By "action," I do not mean just moving things around the page, like a bad Flash animation. JavaScript enables you to manipulate or change a page that's already loaded within the document.

In this chapter, we will look at how to change the stylesheet in a web page based on the time of day or the screen resolution. Also, we'll look at how to use a popular JavaScript framework to make backward-compatible websites and add a bit of flair to rollovers.

14.1 Determining Whether JavaScript Is Available Within a Browser

Problem

You want to determine whether a browser can handle JavaScript.

Solution

Use the `noscript` element, placed within the `body` element, to let users know that JavaScript is not available in their browser, as shown in Figure 14-1:

```
<body>
 <noscript>
  <p>JavaScript is not enabled. Some features of this website
  will not work.</p>
 </noscript>
</body>
```
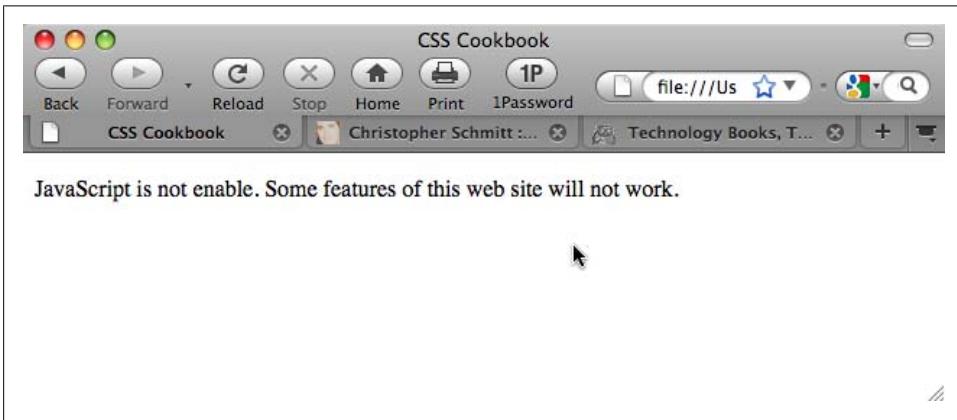

Figure 14-1. A notice about JavaScript not being turned on

Discussion

The `noscript` element is a method for providing different, but accessible, content to a browser that does not have JavaScript installed.

Although JavaScript enjoys support in all desktop browsers in which it's turned on by default, some users choose to turn it off (it's rare, but it happens), or they use another device that cannot render JavaScript.

> If you find that a web page does not validate with a `noscript` element (e.g., within a `p` element), move the element outside its parent element until the document validates.

See Also

The HTML4 specification for `noscript` at *http://www.w3.org/TR/html4/interact/scripts .html#h-18.3.1*

14.2 Applying a Different Stylesheet Based on the Time of Day

Problem

You want to apply a different stylesheet as the day progresses.

Solution

Pull in the time on the user's clock and deliver the appropriate stylesheet:

```
<script type="text/javascript">
 function setTimedStylesheet() {
  var theTime = new Date().getHours();
   if (8 <= theTime&&theTime < 20) {
    document.write("<link rel='stylesheet' href='daytime.css'
type='text/css'>");
   }
   if (20 <= theTime&&theTime < 8) {
    document.write("<link rel='stylesheet' href='nighttime.css'
type='text/css'>");
   }
 }
setTimedStylesheet();
</script>
```

Make sure you include the `noscript` element that includes a link to the default stylesheet in case the browser does not have JavaScript:

```
<script type="text/javascript">
 function setTimedStylesheet() {
  var theTime = new Date().getHours();
   if (8 <= theTime&&theTime < 20) {
    document.write("<link rel='stylesheet' href='daytime.css'
type='text/css'>");
   }
   if (20 <= theTime&&theTime < 8) {
    document.write("<link rel='stylesheet' href='nighttime.css'
type='text/css'>");
   }
 }
setTimedStylesheet();
</script>
<noscript>
 <link href="default.css" rel="stylesheet" type="text/css">
</noscript>
```

Discussion

Creating a customized look and feel based on the time of day isn't a far-fetched notion. Radio and television stations in the United States divide their broadcasts based on the time of day—for example, Daytime Emmy Awards, drive-time radio shows, prime time television shows, and so on.

The main problem with using this method is that you are assuming the clocks on people's computers are actually accurate.

Another solution is to get the time of day from your server through a middleware programming language such as PHP and pass it on as a variable to the JavaScript.

See Also

The `Date` object reference at *http://www.w3schools.com/jsref/jsref_obj_date.asp*

14.3 Redirecting to a Mobile Site Based on the Browser's Screen Width

Problem

You want to apply a change to the stylesheet based on the browser's screen width.

Solution

Detect the screen width of the current browser and redirect the browser to a more mobile-friendly version of the site's design:

```
<script type= "text/javascript">
if (screen.width <= 481) {
  document.location = "http://mobi.example.com/"
 }
</script>
```

Discussion

The base resolution relies on the JavaScript being able to detect the browser width (based in pixels). If the browser width is less than or equal to 481 pixels, it's assumed that the browser is a mobile device.

 Not all mobile devices have JavaScript.

Higher-resolution design

You can also flip the script to detect whether a user's browser has a larger-than-average browser window open:

```
<script type= "text/javascript">
if (screen.width <= 481) {
 document.location = "http://mobi.example.com/"
} else if (screen.width >= 1280) {
```

```
  document.location = "http://high-def.example.com/";
}
</script>
```

See Also

More robust JavaScript for delivering a resolution-independent layout at *http://www
.themaninblue.com/writing/perspective/2004/09/21/*

14.4 Adding a JavaScript Framework to a Web Page

Problem

You want to add a JavaScript framework to a web page, as shown in Figure 14-2.
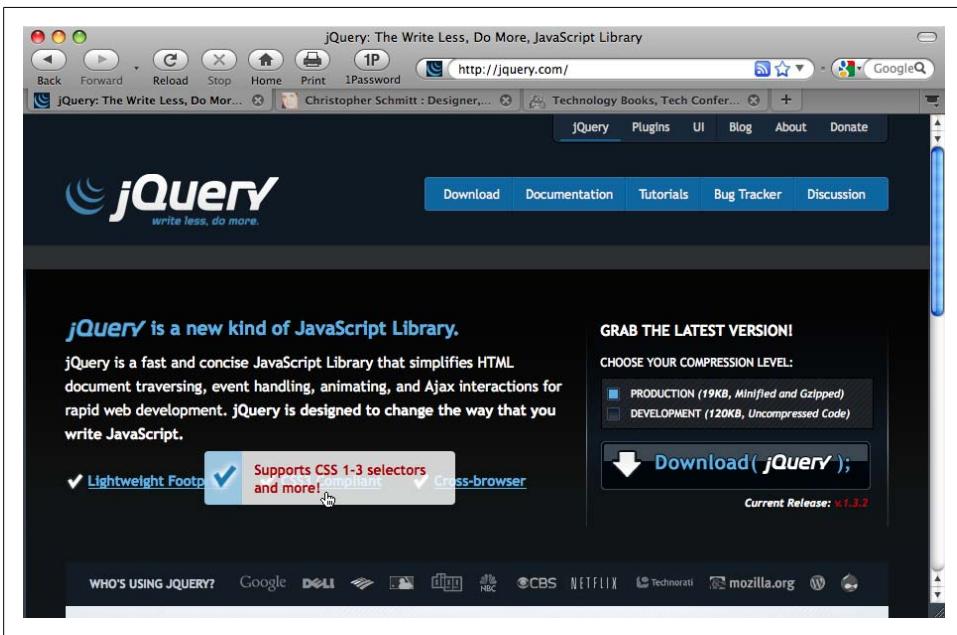
Figure 14-2. The jQuery framework home page

Solution

Use Google's hosting feature to associate the jQuery framework (see Figure 14-3, shown
later) to a web page:

```
<script type="text/javascript"
src="http://ajax.googleapis.com/ajax/libs/jquery/1.3.2/jquery.js"></script>
```

Then, below the citing of the jQuery framework, add a custom script:

```
<script type="text/javascript"
src="http://ajax.googleapis.com/ajax/libs/jquery/1.3.2/jquery.js"></script>
<script type="text/javascript">
 // Your code goes here...
 $(document).ready(function(){
  window.alert("Hello, world! You have JavaScript!")
 });
</script>
```

Discussion

By using Google to host the jQuery framework code, you reap three immediate benefits.

The first benefit is that of caching. If other websites utilize Google's services and link to jQuery, the code is cached within the browser. When a site visitor goes to another site, the page renders faster since the jQuery is already cached. Even with the minified and gzipped version weighing 19 KB, this translates to savings for your users.

A second benefit deals with how many connections a browser can make to a web server. To not overwhelm a server, a browser limits the number of connections made to a web server as it downloads the HTML, imagery, scripts, and so on. Offloading the jQuery framework to another server makes the page render faster.

The third benefit is that Google's servers are likely to be faster at delivering files such as the jQuery framework to the site visitor's machine, unless your server was possibly a stone's throw from your site visitors.

> The alert statement is included as a simple demonstration of where custom JavaScript is placed. If a simple alert statement were all that was needed, the script would be a quick line of code bypassing the need for a JavaScript framework:
>
> ```
> <script type="text/javascript">
>  window.alert("Hello, world! You have JavaScript!")
> </script>
> ```

See Also

The list of jQuery and other Ajax libraries hosted by Google at *http://code.google.com/apis/ajaxlibs/documentation/*

14.5 Using CSS3 Selectors in IE6 and IE7

Problem

You want to use CSS3 selectors in earlier versions of Internet Explorer.

Solution

First, include CSS3 syntax within the stylesheet for CSS3-capable browsers:

```
#content {
 border: 4px solid black;
}
#content p {
 margin: 1em 0;
}
/* removes the bottom margin from the last paragraph */
#content p:last-child {
 margin: 1em 0 0 0;
}
```

Then use jQuery's ability to reference portions of a document through standardized CSS3 syntax:

```
<script type="text/javascript"
src="http://ajax.googleapis.com/ajax/libs/jquery/1.3.2/jquery.js"></script>
<script type="text/javascript">
 // Your code goes here...
 $(document).ready(function(){
  $("#content p:last-child").css("margin","1em 0 0 0");
 });
</script>
```

Discussion

One of the benefits of using a JavaScript framework such as jQuery is its usage of CSS selectors. Instead of styles being applied to a page, selectors associate functions and behaviors to parts of the page.

To use a CSS selector, first use what's called a jQuery object:

```
$(css-selector);
```

Then set a CSS selector within the jQuery object:

```
$("p+p");
```

Next, add the CSS declaration:

```
$("p+p").css("font-weight","normal");
```

> jQuery might not understand some CSS shorthand properties. If jQuery is not affecting the look of the page as intended through a CSS shorthand property, use the CSS properties that make up the shorthand properties instead. For example, use `border-right` instead of simply `border`.

Hiding extraneous JavaScript from modern browsers

Although jQuery is a solution for fixing older versions of Internet Explorer, modern browsers already support the CSS rule. So, reapplying the same effect in a web page is a little overkill.

To reduce browser load, use conditional comments (see Recipe 12.7) to let previous versions of Internet Explorer see the JavaScript:

```
<!--[if lt IE 8]>
<script type="text/javascript">
 // Your code goes here...
 $(document).ready(function(){
  $("#content p:last-child").css("margin","1em 0 0 0");
 });
</script>
<![endif]-->
```

Dean Edwards's IE7 script

Dean Edwards's IE7 script (see *http://code.google.com/p/ie7-js/*) attempts to fix a lot of the issues with previous versions of IE through JavaScript.

By attaching some JavaScript to your page, you can fix a good number of bugs that afflict these browsers. However, the IE7 fix is specific to only the issues with these browsers, and the file size is not trivial. With a file size of 71.1 KB, you have to weigh whether using a large file to fix older browsers for your visitors is worthwhile.

Also, the script is in beta, and its last update occurred in February 2008. Although Dean Edwards's script does a great job of fixing a lot of issues with IE6, some issues might crop up if you push the edges of trying to get IE6 to behave like a modern browser.

A number of the current-day JavaScript frameworks owe much to Dean Edwards's code.

See Also

The css function page at *http://docs.jquery.com/API/1.3/CSS*

14.6 Zebra-Striping an HTML Table with JavaScript

Problem

You want to apply background colors to every other HTML table row without manually adding class selectors.

Solution

Use jQuery's ability to add and remove class attributes to HTML elements.

First, create CSS rules for alternating colors:

```
table.striping tr.diff td {
 background-color: #cbc1be;
}
```

Then write code for jQuery in which alternating rows are selected:

```
<script type="text/javascript"
src="http://ajax.googleapis.com/ajax/libs/jquery/1.3.2/jquery.js"></script>
<script type="text/javascript">
 $(document).ready(function(){
  $(".striping tr:even").addClass("diff");
 });
</script>
```

With the table row selected, add a `class` attribute with a value of `diff` to each row to apply the alternating background colors, as shown in Figure 14-3:

```
<script type="text/javascript"
src="http://ajax.googleapis.com/ajax/libs/jquery/1.3.2/jquery.js"></script>
<script type="text/javascript">
 $(document).ready(function(){
  $(".striping tr:even").addClass("diff");
 });
</script>
```


Figure 14-3. Alternating striping of table rows

Discussion

Unlike Recipe 14.3, where the Solution relied on hardcoding the CSS rule in the Java-Script, the CSS rule here is prewritten. Then the JavaScript goes through the work of automatically applying the `class` attribute to every other row.

Using a pure CSS solution

The CSS-only solution for this recipe is to use `:nth-child` (see Recipe 9.8):

```
tr:nth-child(even) td {
  background-color: #cbc1be;
}
```

You can use conditional comments to hide the JavaScript (as shown in the Discussion in Recipe 12.3) in tandem with the jQuery solution.

See Also

The `addClass` jQuery attribute page at *http://docs.jquery.com/Addclass*

14.7 Highlighting a Table Row with Mouseovers

Problem

You want to provide a method for highlighting a table row, even in Internet Explorer 6.

Solution

Create a CSS rule with a class selector for the background color of the highlighted table row:

```
table.striping tr.over td {
  background-color: #fbc057;
}
```

Then use the jQuery object to pinpoint where the class selector should be applied:

```
$(".striping tr");
```

Instruct jQuery to activate only when the user hovers her mouse over a link:

```
$(".striping tr").mouseover();
```

Next, start a function:

```
$(".striping tr").mouseover(function() {
});
```

Let the jQuery know that the function applies to what is currently selected (which are the table rows):

```
$(".striping tr").mouseover(function() {
 $(this);
});
```

Use the `addClass()` function to insert the `over` class attribute into the table row:

```
$(".striping tr").mouseover(function() {
    $(this).addClass("over");
});
```

Now when a user rolls her mouse cursor over the table rows, the rows become highlighted. However, the Solution so far only inserts the class attribute and does not remove it when the user rolls her mouse cursor off the table row, as shown in Figure 14-4.

Figure 14-4. Table rows changing background color

Use the `removeClass()` function to take away the class attribute, as shown in Figure 14-5:

```
$(".striping tr").mouseover(function() {
  $(this).addClass("over");
});
$(".striping tr").mouseout(function() {
  $(this).removeClass("over");
});
```


Figure 14-5. Table row colors reverting when mouse cursor moves off the table row

Discussion

This Solution introduces two helpful events for creating interesting effects within a web page: `mouseover()` and `mouseout()`. Both are popular regular JavaScript functions that have been used to achieve image-based rollovers before the popularity of CSS-only image-based rollovers.

Chaining functions

With the jQuery events tied to the same elements of a web page, the table rows, it's possible to reduce the code through a process called *chaining*. This technique removes the duplicate jQuery object like so:

```
$(".striping tr").mouseover(function() {
 $(this).addClass("over");
}).mouseout(function() {
 $(this).removeClass("over");
});
```

See Also

The `removeClass` jQuery page at *http://docs.jquery.com/Attributes/removeClass*

14.8 Adding Effects to Simple Image Rollovers

Problem

You want to add fades to rollovers within a web page.

Solution

Set up a jQuery object with a `mouseover` function (as shown in Figure 14-6):

```
$("#site-nav a").mouseover(function () {
});
```

Then use the `fadeTo()` function to set the opacity to 50%:

```
$("#site-nav a").mouseover(function() {
 $(this).fadeTo("slow", .50);
});
```

CERTIFICATIONS   DOES IT REALLY SMELL THAT BAD?   ABOUT US   CONTACT
Salted and Buttered Versions   The One Who Smelled It Has Not Always Dealt It   We Are For Real   Pop on Over

Figure 14-6. Rolling over the images to create a fade effect

Now add an additional `mouseout` function when the user rolls off the navigation to return the opacity to 100%, as shown in Figure 14-7:

```
$("#site-nav a").mouseover(function () {
 $(this).fadeTo("slow", .50);
}).mouseout(function () {
$ (this).fadeTo("slow", 1);
});
```

Figure 14-7. The image fading back to 100% opacity once the user moves the cursor off the image

Discussion

The `fadeTo()` effect accepts one of three predefined speed keywords: `slow`, `normal`, or `fast`. In place of one of those keywords you can use a number representing milliseconds:

```
$("#site-nav a").mouseover(function() {
$(this).fadeTo(2000, .50);
});
```

Fading of elements on a web page is just one of many built-in effects of jQuery. Other effects include custom animations, sliding an element, showing, and hiding. For a complete list, see *http://docs.jquery.com/Effects*.

See Also

The `fadeTo()` jQuery page at *http://docs.jquery.com/Effects/fadeTo*

14.9 Making a Row of Elements with a Variable Amount of Content the Same Height

Problem

You want a row of elements to be the same height as the tallest element within a row.

Solution

First initialize a variable at zero:

```
<script type="text/javascript"
src="http://ajax.googleapis.com/ajax/libs/jquery/1.3.2/jquery.js"></script>
<script type="text/javascript">
 $(document).ready(function(){
  var topHeight = 0;
 });
</script>
```

Instruct jQuery to cycle through each element specified in the jQuery object. In this case, jQuery cycles through every `p` element within the parent `div` element with an `id` value of `content`:

```
<script type="text/javascript"
src="http://ajax.googleapis.com/ajax/libs/jquery/1.3.2/jquery.js"></script>
<script type="text/javascript">
 $(document).ready(function(){
```

```
var topHeight = 0;
$("#content p").each();
});
</script>
```

As jQuery cycles through the p elements, it determines whether the value of the height is larger than the preset value. Since the initial value of the topHeight variable was zero, it's a given that the if statement is going to be executed:

```
<script type="text/javascript"
src="http://ajax.googleapis.com/ajax/libs/jquery/1.3.2/jquery.js"></script>
<script type="text/javascript">
 $(document).ready(function(){
  var topHeight = 0;
  $("#content p").each(function(){
   if ($(this).height() > topHeight) {
   }
  });
 });
</script>
```

Since the topHeight value is changing, capture the value of the variable with this latest, largest height value:

```
<script type="text/javascript"
src="http://ajax.googleapis.com/ajax/libs/jquery/1.3.2/jquery.js"></script>
<script type="text/javascript">
 $(document).ready(function(){
  var topHeight = 0;
  $("#content p").each(function(){
   if ($(this).height() > topHeight) {
    topHeight = $(this).height();
   }
  });
 });
</script>
```

As jQuery completes the cycle through the p elements and has determined the tallest height of the elements, assign this value to the other elements in the row, as shown in Figure 14-8:

```
<script type="text/javascript"
src="http://ajax.googleapis.com/ajax/libs/jquery/1.3.2/jquery.js"></script>
<script type="text/javascript">
 $(document).ready(function(){
  var topHeight = 0;
  $("#content p").each(function(){
   if ($(this).height() > topHeight) { topHeight = $(this).height(); }
  });
  $("#content p").height(topHeight);
 });
</script>
```
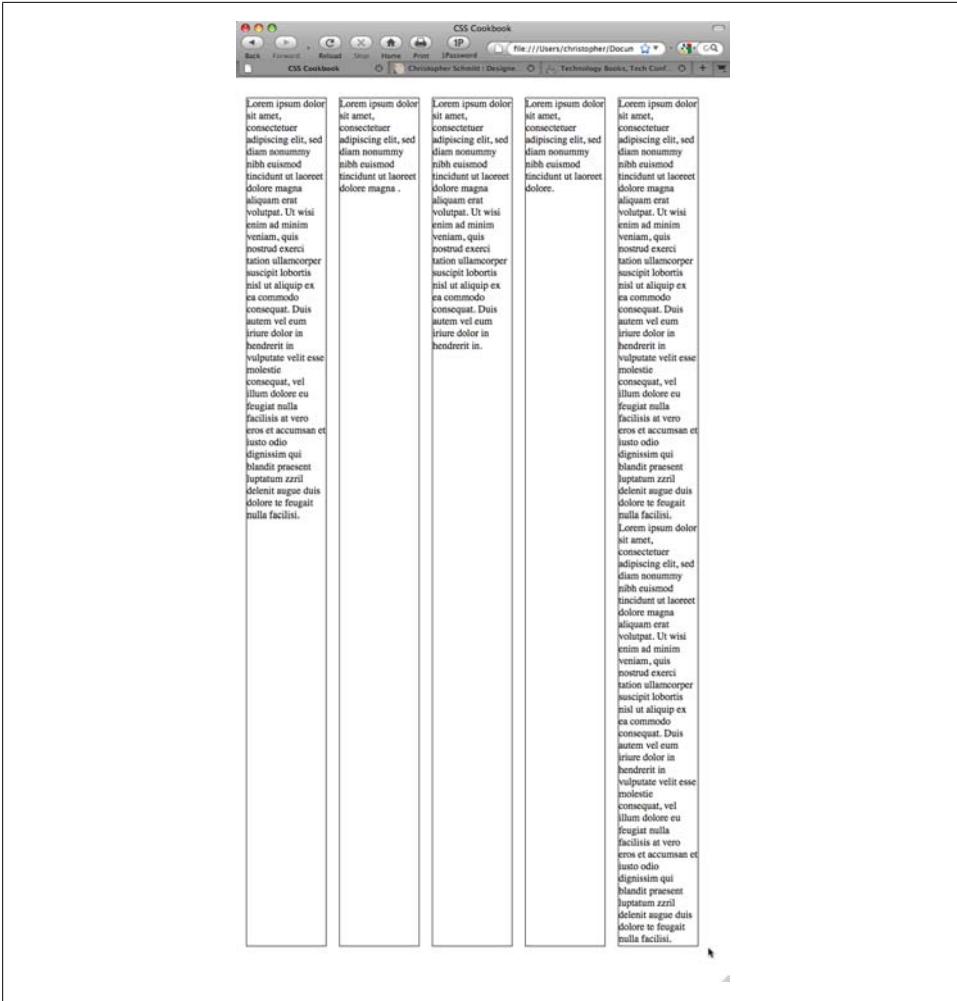
Figure 14-8. Equal heights of all elements

Discussion

This is a rather straightforward, but useful, recipe. Use the `each()` function to cycle through a series of elements in a page to determine their height. Once the value is found and cast within a variable, that value can then be applied to all those elements.

See Also

The jQuery `each()` function page at *http://docs.jquery.com/Utilities/jQuery.each*

14.10 Setting a Link to Open a New Window

Problem

You want to pop open a new window when clicking on a link.

Solution

First, use the `rel` attribute and set a value of `external`:

```
<a href="http://csscookbook.com/" rel="external">Click here</a> to check it out!
```

Set up the jQuery object to target all links with the `rel` attribute and a value of `external` in the web page through an attribute selector:

```
<script type="text/javascript"
src="http://ajax.googleapis.com/ajax/libs/jquery/1.3.2/jquery.js"></script>
<script type="text/javascript">
 $(document).ready(function(){
  $('a[rel="external"]');
 });
</script>
```

Apply the `click()` function:

```
<script type="text/javascript"
src="http://ajax.googleapis.com/ajax/libs/jquery/1.3.2/jquery.js"></script>
<script type="text/javascript">
 $(document).ready(function(){
  $('a[rel="external"]').click();
 });
</script>
```

Insert a function that opens a new window with the link address already written in the `a` element:

```
<script type="text/javascript"
src="http://ajax.googleapis.com/ajax/libs/jquery/1.3.2/jquery.js"></script>
<script type="text/javascript">
 $(document).ready(function(){
  $('a[rel="external"]').click(function() {
   window.open($(this).attr('href'));););
 });
 });
</script>
```

With this setup, the browser will load the link in both the new window and the parent window. To keep that from happening, instruct the browser to not follow the link in the parent browser window:

```
<script type="text/javascript"
src="http://ajax.googleapis.com/ajax/libs/jquery/1.3.2/jquery.js"></script>
<script type="text/javascript">
 $(document).ready(function(){
  $('a[rel="external"]').click( function() {
```

```
  window.open($(this).attr('href'));
  return false;
 });
});
</script>
```

Discussion

Within a Strict DOCTYPE, the use of the target attribute is not allowed and invalidates the markup. Using JavaScript gets around this predicament.

> It's best to avoid popping a new window if at all possible. Don't rely on your users having a desktop browser to view your content.

See Also

The jQuery `click()` function page at *http://docs.jquery.com/Events/click*

14.11 Making an Entire div Element Clickable

Problem

You want to make a block-level element clickable.

Solution

Set a `class` attribute with a value of `link` within a `div` element:

```
<div class="link" id="blipvert">
 <h2>Amazing Sale</h2>
 <p><a href="http://csscookbook.com/">Click here</a> to check it out!</p>
</div>
```

Use the jQuery object to pick out each `div` element with a class selector value of `link`:

```
<script type="text/javascript"
src="http://ajax.googleapis.com/ajax/libs/jquery/1.3.2/jquery.js"></script>
<script type="text/javascript">
 $(document).ready(function(){
  $("div.link").click(function() {
  });
 });
</script>
```

Use the `find()` function to find the link within the `div` element and use its link as the destination when the `div` element is clicked:

```
<script type="text/javascript"
src="http://ajax.googleapis.com/ajax/libs/jquery/1.3.2/jquery.js"></script>
```

```
<script type="text/javascript">
 $(document).ready(function(){
  $("div.link").click(function() {
   window.location=$(this).find("a").attr("href");
   return false;
  });
 });
</script>
```

Discussion

Although not supported in all modern browsers, HTML5 allows for block-level elements to be clickable within a link:

```
<div id="blipvert">
<a href="http://csscookbook.com/">
 <h2>Amazing Sale</h2>
 <p>Click here to check it out!</p>
</a>
</div>
```

See Also

The jQuery `find()` function page at *http://docs.jquery.com/Traversing/find*

14.12 Supporting Transparent PNGs in IE6 with JavaScript

Problem

You want to use alpha-transparent PNGs with Internet Explorer 6.

Solution

Use a plug-in specifically designed for PNG support for older versions of Internet Explorer 6.

First, download the *jquery.pngFix.js* file from *http://jquery.andreaseberhard.de/pngFix/*, as shown in Figure 14-9.

Include the *jquery.pngFix.js* file below the jQuery framework:

```
<script type="text/javascript"
src="http://ajax.googleapis.com/ajax/libs/jquery/1.3.2/jquery.js"></script>
<script type="text/javascript" src="/_assets/js/jquery.pngFix.js"></script>
```

Then activate the plug-in:

```
<script type="text/javascript"
src="http://ajax.googleapis.com/ajax/libs/jquery/1.3.2/jquery.js"></script>
<script type="text/javascript" src="/_assets/js/jquery.pngFix.js"></script>
 <script type="text/javascript">
  $(document).ready(function(){
   $(document).pngFix();
```

```
    });
</script>
```
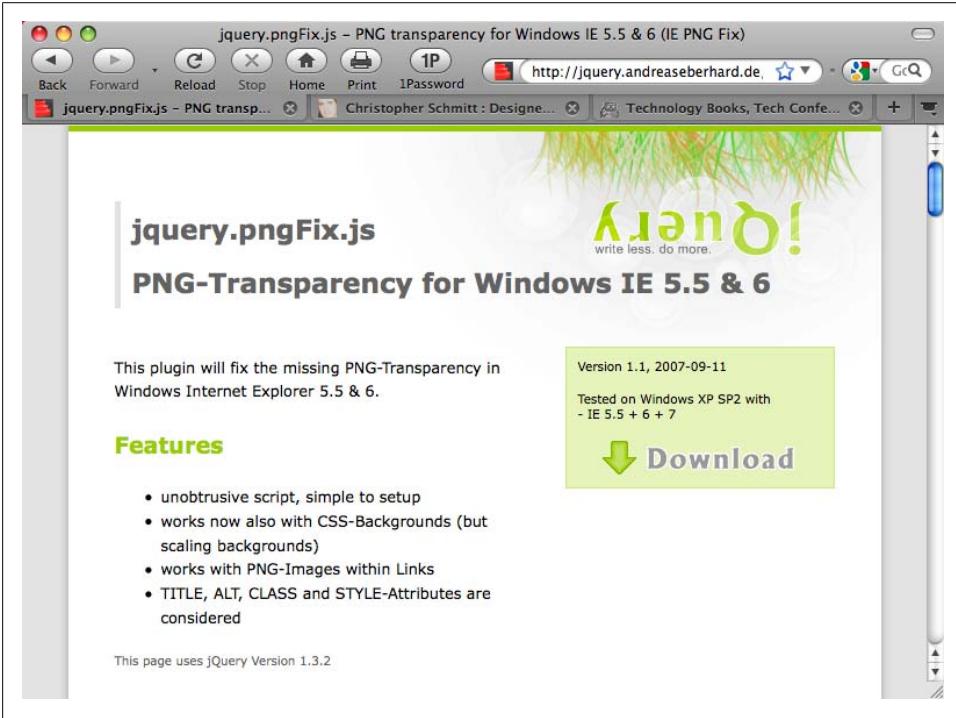

Figure 14-9. Home page for the jQuery plug-in

Discussion

Since this Solution deals with older versions of Internet Explorer, conditional comments can isolate the files from modern browsers that natively support alpha-transparent PNGs:

```
<script type="text/javascript"
src="http://ajax.googleapis.com/ajax/libs/jquery/1.3.2/jquery.js"></script>
<!--[if lt IE 7]>
<script type="text/javascript" src="/_assets/js/jquery.pngFix.js"></script>
<script type="text/javascript">
    $(document).ready(function(){
        $(document).pngFix();
    });
</script>
<![endif]-->
```

The trouble with PNGs and IE6

Although the JavaScript solution requires a lot of handcoding throughout the document, web designers should be aware of a couple of issues concerning the way in which Internet Explorer handles alpha-transparent PNGs.

First, the solution makes use of Microsoft's proprietary `filter` property, which can handle alpha-transparent PNGs. Although this allows alpha transparency into the web designer's toolkit, having inline PNG images is not possible with IE as the images can be used on only the *background* of elements.

Although the image is placed into the background of the element, the image is stretched to fit the dimensions of that element. This is the second problem, as it runs contrary to the common experience web designers might expect: that the images retain their own dimensions and simply tile out.

So, when using PNG images for IE6, make sure the dimensions of the PNG image match exactly the dimensions of the element; otherwise, unwanted stretching of the image might occur.

Finding additional jQuery plug-ins

One of the benefits of using jQuery is the wide developer base for the framework. If you can think of a problem, chances are an industrious JavaScript coder has devised a plug-in to solve it. Simply perform a Google search for your problem along with the keyword *jQuery* and you might be happily surprised.

> An additional resource to this book is *jQuery Cookbook* by Cody Lindley (O'Reilly).

See Also

Recipe 4.17 for creating PNG8 with alpha transparency that works in IE6

14.13 Delivering HTML5 and CSS3 to Browsers That Can Handle Them

Problem

You want to provide styles that can take advantage of CSS3 properties and provide alternatives to browsers that cannot.

Solution

Download the Modernizr JavaScript library at *http://www.modernizr.com*.

Include a reference to the Modernizr library within the `head` element:

```
<script src="modernizr-0.9.min.js"></script>
```

Then use class selectors to have specific CSS3 properties applied to browsers that can render them:

```
h1 {
 background-color: #333;
 color: #fff;
}
.rgba h1 {
 background-color: rgba(0,0,0, .8);
}
```

For HTML5, use a similar approach. First, mark up HTML5 elements such as an `audio` element:

```
<audio>
 <source src="example.ogg" />
 <source src="example.mp3" />
</audio>
```

Then apply CSS rules to hide the `audio` element for browsers that do not support it:

```
.no-audio audio {
 display: none;
}
```

Discussion

Although development of new features and abilities within browsers is welcomed, there are some hassles. As browsers start to implement HTML5 and CSS3 standards at a rapid pace, dealing with uneven support of CSS3 and HTML5 across modern browsers becomes an issue.

Web designers Faruk Ateş and Paul Irish created this simple JavaScript library that allows for basic cross-browser development.

As of this writing, Modernizr checks for the following HTML5 and CSS3 features:

- `opacity:`
- CSS animations
- CSS columns
- CSS gradients
- CSS reflections
- CSS 2D transforms
- CSS 3D transforms

- CSS transitions
- Geolocation API
- `@font-face`
- Canvas
- Canvas text
- HTML5 audio
- HTML5 video
- `rgba()`
- `hsla()`
- `border-image:`
- `border-radius:`
- `box-shadow:`
- Multiple backgrounds

See Also

The Modernizr documentation at *http://www.modernizr.com/docs/*

Resources

When working with CSS, keep these two tips in mind: simplify and verify.

After you've crafted your CSS rules, simplify them by using only the selectors and properties you believe you need; any extras could cause some confusion down the road. Then verify the HTML, XHTML, and CSS with the help of validators as you code. Those two steps solve most problems developers encounter when working with CSS.

However, if you still run into trouble, this appendix contains some of the top references, discussion groups, and tools on the Internet to help in your CSS development.

General HTML and CSS Instruction

"A Roadmap to Standards" (*http://www.mezzoblue.com/archives/2004/04/30/a_road map_to/index.php*)
   This essay by David Shea is a good introduction and pep talk for web designers who want to learn about web standards–based development.

"CSS from the Ground Up" (*http://www.wpdfd.com/editorial/basics/index.html*)
   Web developers new to CSS will benefit from this well-paced tutorial available from Web Page Design for Designers.

"Basics of CSS Positioning" (*http://www.communitymx.com/content/article.cfm?cid= 3B56F&print=true*)
   For more information about positioning with CSS, try this tutorial by Community MX.

Floatutorial (*http://css.maxdesign.com.au/floatutorial/index.htm*)
   Learn about floating elements with CSS in various practice coding examples provided by Max Design.

Selectutorial (*http://css.maxdesign.com.au/selectutorial/index.htm*)
   Gain a better understanding of CSS selectors with this tutorial, also by Max Design, which also demonstrates how to use selectors to construct a three-column layout.

Design Resources

A List Apart's "CSS Topics" (*http://www.alistapart.com/topics/code/css/*)
> At this website, most of the articles published on the topic of CSS come in from web designers sharing their thoughts and breakthroughs with CSS-enabled design.

Layout Reservoir (*http://www.bluerobot.com/web/layouts/*)
> This small but valuable resource from BlueRobot.com covers two- and three-column layouts.

CSS/Edge (*http://www.meyerweb.com/eric/css/edge/*)
> Eric A. Meyer's workshop displays some of his more advanced CSS experiments.

CSS Zen Garden (*http://www.csszengarden.com/*)
> CSS Zen Garden showcases how web developers from all over the world restyle the same content. Surfing through several designs is great inspiration, but it's also a fantastic way to better understand the concept of separating presentation from content.

"CSS Layout Techniques" (*http://www.glish.com/css/*)
> This web page from Glish.com presents one of the first collections of multicolumn layouts created in CSS without the use of HTML tables.

Microformats blog (*http://www.microformats.org/*)
> This blog defines and promotes standards for coding unique pieces of content. Check the microformats listing for methods you can use to code common data such as calendar events, contact information, or even the `abbr` element.

SimpleQuiz Archives (*http://www.simplebits.com/bits/simplequiz/*)
> Web designer and author Dan Cederholm conducted a series of quizzes trying to determine the best methods for marking and styling common web development scenarios. In addition to reading the conclusion to each quiz, you can read each quiz's comments by web designers to get a more informed opinion on coding practices.

Smashing Magazine's CSS articles (*http://www.smashingmagazine.com/category/css/*)
> This online magazine aggregates blog posts and online articles from web developers and designers on the Internet, and publishes summaries of their findings.

Typetester (*http://typetester.maratz.com/*)
> This is a flexible tool that allows web developers to customize three sets of type and then generates the basic CSS for easy copying and pasting. Available features include setting the values for fonts, size, tracking, leading, letter spacing, alignments, and more.

Discussion Groups

Babble List (http://www.babblelist.com/)
> This is a web design and development mailing list which I moderate. Targeting advanced web design issues, the site offers a lively exchange of information, resources, theories, and practices of designers and developers.

css-discuss.org (http://www.css-discuss.org/)
> This mailing list, which is chaperoned by CSS expert Eric Meyer, author of O'Reilly's *CSS: The Definitive Guide*, aims to provide practical discussion about the application of CSS.

WebDesign-L.com (http://www.webdesign-l.com/)
> This mailing list has been around since 1997 and caters to almost all aspects of website building, including (but not limited to) CSS. Earnest questions are met with sage advice.

Usenet Stylesheets Newsgroup (http://news:comp.infosystems.www.authoring.style sheets)
> Founded in 1997, this unmoderated newsgroup covers the theory and application of CSS. Topics for the group can include practical applications, questions about the specification, the benefits of CSS, implementation bugs in browsers, and more.

www-style@w3.org Mail Archives (http://lists.w3.org/Archives/Public/www-style/)
> Maintained by the World Wide Web Consortium (W3C), this mailing list provides a venue for discussing the theories and future of CSS. Questions about the specification or about CSS proposals are welcomed; however, discussions regarding practical applications of the technology are discouraged.

References

CSS Browser Support charts (http://westciv.com/wiki/CSS_Compatibility_Guide/)
> If you run into problems developing with CSS, check the CSS support charts to determine whether there is a problem with the browser(s) you are using.

CSS filters (http://centricle.com/ref/css/filters/)
> Use browser inconsistencies to your advantage. If you want to target CSS rules to a specific browser or set of browsers, refer to this comprehensive list of hacks and filters. It will tell you which CSS rules and declarations work in which browsers— or won't work, as the case may be.

W3C's recommended DTDs (http://www.w3.org/QA/2002/04/valid-dtd-list.html)
Assigning the right DOCTYPE to a web page helps to establish the correct manner in which browsers will render your web page and validators will check your code. On this web page is a listing of the most commonly used DOCTYPEs.

W3C's CSS home page (http://www.w3.org/Style/CSS/)
This is the official site for CSS. At this site you can learn about the history of CSS, investigate resources and authoring tools, and read current CSS news.

CSS 2.1 specification (http://www.w3.org/TR/CSS21/)
Browser implementations of the CSS specification are sometimes a confusing mess. When tracking down how to achieve a certain look or an implementation bug, check the specification (as well as the CSS support charts).

CSS3 specification (http://www.w3.org/Style/CSS/current-work)
The forthcoming CSS3 specification is still being written. Due to the complex nature of the specification, the working draft was split into separate modules; the idea being that work that could proceed in one module could work independently of another without causing a drag on other modules. The result is that there are various aspects of CSS3 at various levels of completion, with most in Working Draft status at the time of this writing.

HTML 4.01 specification (http://www.w3.org/TR/html4/)
To make the most out of using CSS for web design, you need to create your web documents with structured markup instead of using workarounds and hacks. Furthermore, you need to mark up your documents with elements to imply an inherent presentational meaning. For example, you need to highlight important words using the `em` element and not the `b` element. If you need to change your production methods, dig into the HTML specification and get to know the elements all over again.

HTML5 specification (http://dev.w3.org/html5/spec/Overview.html)
Addressing the needs of web application development and the shortcomings of HTML4, the work of HTML5 is ongoing and does not have a timetable for being eligible for candidate recommendation anytime soon. Even with an incomplete specification, web developers can leverage some of HTML5 where applicable in modern browsers.

XHTML 1.0 specification (http://www.w3.org/TR/xhtml1/)
eXtensible HyperText Markup Language (XHTML) is a restructuring of HTML4 in XML 1.0. Although XHTML markup is stricter than that of HTML4, the benefits are simple: more logical markup, increased interoperability, and enhanced accessibility.

Tools

BrowserCam (http://www.browsercam.com/)

BrowserCam is an affordable web-based service that tests a web design in multiple browsers on numerous operating systems. At the time of this writing, a free 24-hour evaluation period is available for web developers who register on the site.

CleanCSS (http://www.cleancss.com/)

An online formatter and compression tool for long, complicated stylesheets, this free tool provides several options and allows you to export compressed CSS files, which eliminates potential character problems when copy and pasting from web browsers to code editors.

Firebug (https://addons.mozilla.org/en-US/firefox/addon/1843)

This free tool for web developers allows quick editing, coding, and debugging of HTML, CSS, and JavaScript of the web page currently being viewed, and it is an excellent tool for debugging Ajax-based web applications. In addition, you can install a plug-in called CodeBurner (see *http://tools.sitepoint.com/codeburner/*) that provides HTML and CSS reference materials inside Firebug's development environment.

IE NetRenderer (http://ipinfo.info/netrenderer/index.php)

This is a free tool that allows web developers to preview web pages as viewed in Internet Explorer. It's also a great site for Macintosh users who don't own a Windows machine but want to test or use virtual machine software to run Windows OS along with OS X.

SelectORacle (http://gallery.theopalgroup.com/selectoracle/)

This free service is designed to help people learn more about complex CSS selectors by translating their meaning into plain English. CSS selectors can be submitted in one of two ways. The first method is to copy and paste a CSS selector into the form. The second method is to enter either a URL of a web page with an embedded stylesheet, or a URL to an external stylesheet. The service then renders the CSS selector into easy-to-understand language.

W3C CSS Validator (http://jigsaw.w3.org/css-validator/)

This free service, provided on the W3C server, checks CSS for proper structure. You can test your markup by uploading files, entering a web address in the form, and then copying and pasting the CSS into a form field. And if you are so inclined, you can download and install the validator on your own server.

W3C HTML Validator (http://validator.w3.org/)

The W3C HTML validator is another free service from the W3C. Similar to the CSS validator, the HTML validator checks to see whether your markup conforms to web standards.

Web Developer browser extension (https://addons.mozilla.org/en-US/firefox/addon/60)
Chris Pedrick has created an indispensable extension for the popular Firefox and Mozilla browsers. Features include the ability to edit a web page's CSS through the browser, send a web page's code directly to a W3C validator, place an outline on block-level elements, as well as many, many other functions with a simple click of the mouse.

xScope (http://iconfactory.com/software/xscope)
xScope is a set of Mac tools designed to save you time when fine-tuning web development. Included in the set is my favorite tool, Dimensions, which determines the distances between objects immediately; the days of taking a screenshot of your browser and opening Photoshop to measure the distance of columns or text leading are over. For Windows users, there are several, separately developed third-party Firefox extensions that replicate some of the tools featured in this package. One example is MeasureIt, an extension for displaying rulers within the browser (see *https://addons.mozilla.org/en-US/firefox/addon/539*).

CSS 2.1 Properties and Proprietary Extensions

This appendix contains several tables. Table B-1 lists CSS properties from W3C's CSS 2.1 specification (see *http://www.w3.org/TR/CSS21*). It lists the property's values, initial value, what the property applies to, whether the values in the property are inherited, whether the property accepts percentages, and the property's media group.

Table B-2 lists Microsoft's proprietary extensions to the specifications. These properties will not validate if you run them through a validator and they will appear successfully only in a browser built by Microsoft. Your mileage will vary with the use of these extensions.

As of this writing, Mozilla's proprietary extensions that are available in Mozilla, Firefox, and Netscape Navigator 6 and later browsers are not fully documented. Table B-3 lists some extensions that are documented from Mozilla's developers' website. Check the site for a complete list of the extensions, at *http://developer.mozilla.org/en/docs/CSS _Reference:Mozilla_Extensions*.

Table B-1. CSS 2.1 properties

Name	Values	Initial value	Applies to (Default: all)	Inherited?	Percentages (Default: N/A)	Media groups
'background-attachment'	scroll|fixed|inherit	scroll		No		Visual
'background-color'	<color>|transparent|inherit	transparent		No		Visual
'background-image'	<uri>|none|inherit	none		No		Visual
'background-position'	[[<percentage>|<length>|left |center|right] [<percentage>| <length>|top|center|bot tom]?]|[[left|center| right]||[top|center|bot tom]]|inherit	0%		No	Refer to the size of the box itself	Visual
'background-repeat'	repeat|repeat-x|repeat-y|no-repeat|inherit	repeat		No		Visual
'background'	['background-color' || 'background-image' || 'background-repeat' || 'background-attachment' || 'background-position']|inherit	See individual properties		No	Allowed on 'back ground-position'	Visual
'border-collapse'	collapse|separate|inherit	separate	'table' and 'inline-table' elements	Yes		Visual
'border-color'	[<color>|transparent]{1,4}| inherit	See individual properties		No		Visual
'border-spacing'	<length> <length>?|inherit	0	'table' and 'inline-table' elements	Yes		Visual

Name	Values	Initial value	Applies to (Default: all)	Inherited?	Percentages (Default: N/A)	Media groups
'border-style'	`<border-style>{1,4}	inherit`	See individual properties		No		Visual
'border-top' 'border-right' 'border-bottom' 'border-left'	`[<border-width> || <border-style> || >'border-top-color']	inherit`	See individual properties		No		Visual
'border-top-color' 'border-right-color' 'border-bottom-color' 'border-left-color'	`<color>	transparent	inherit`	The value of the 'color' property		No		Visual
'border-top-style' 'border-right-style' 'border-bottom-style' 'border-left-style'	`<border-style>	inherit`	none		No		Visual
'border-top-width' 'border-right-width' 'border-bottom-width' 'border-left-width'	`<border-width>	inherit`	medium		No		Visual
'border-width'	`<border-width>{1,4}	inherit`	See individual properties		No		Visual

Name	Values	Initial value	Applies to (Default: all)	Inherited?	Percentages (Default: N/A)	Media groups
'border'	[<border-width> || <border-style> || 'border-top-color'] || inherit	See individual properties		No		Visual
'bottom'	<length> | <percentage> | auto | inherit	auto	Positioned elements	No	Refer to height of containing block	Visual
'caption-side'	top | bottom | inherit	top	'table-caption' elements	Yes		Visual
'clear'	none | left | right | both | inherit	none	Block-level elements	No		Visual
'clip'	<shape> | auto | inherit	auto	Absolutely positioned elements	No		Visual
'color'	<color> | inherit	Depends on user agent		Yes		Visual
'content'	normal | none | [<string> | <uri> | <counter> | attr(<identifier>) | open-quote | close-quote | no-open-quote | no-close-quote]+ | inherit	normal	:before and :after pseudo-elements	No		All
'counter-increment'	[<identifier> <integer>?]+ | none | inherit	none		No		All
'counter-reset'	[<identifier> <integer>?]+ | none | inherit	none		No		All
'cursor'	[[<uri> ,]* [auto | crosshair | default | pointer | move | e-resize | ne-resize | nw-resize | n-resize | se-resize | sw-resize | s-resize | w-	auto		Yes		Visual, interactive

Name	Values	Initial value	Applies to (Default: all)	Inherited?	Percentages (Default: N/A)	Media groups
'direction'	resize|text|wait|help|progress]]|inherit	ltr	All elements, but see prose	Yes		Visual
'display'	inline|block|list-item|run-in|inline-block|table|inline-table|table-row-group|table-header-group|table-footer-group|table-row|table-column-group|table-column|table-cell|table-caption|none|inherit	inline		No		All
'empty-cells'	show|hide|inherit	show	'table-cell' elements	Yes		Visual
'float'	left|right|none|inherit	none	All	No		Visual
'font-family'	[[<family-name>|<generic-family>][,<family-name>|<generic-family>]*]|inherit	Depends on user agent		Yes		Visual
'font-size'	<absolute-size>|<relative-size>|<length>|<percentage>|inherit	medium		Yes	Refer to parent element's font size	Visual
'font-style'	normal|italic|oblique|inherit	normal		Yes		Visual
'font-variant'	normal|small-caps|inherit	normal		Yes		Visual
'font-weight'	normal|bold|bolder|lighter|100|200|300|400|500|600|700|800|900|inherit	normal		Yes		Visual
'font'	[['font-style' || 'font-variant' || 'font-weight']? 'font-	See individual properties		Yes	See individual properties	Visual

Name	Values	Initial value	Applies to (Default: all)	Inherited?	Percentages (Default: N/A)	Media groups

	`size' [ / 'line-height' ]?`
	`'font-family' ]|caption|icon|`
	`menu|message-box|small-caption`
	`|status-bar|inherit`					

`'height'`	`<length>|<percentage>|auto|` `inherit`	auto	All elements except non-replaced inline elements, table columns, and column groups	No	See prose	Visual
`'left'`	`<length>|<percentage>|auto|` `inherit`	auto	Positioned elements	No	Refer to width of containing block	Visual
`'letter-spacing'`	`normal|<length>|inherit`	normal		Yes		Visual
`'line-height'`	`normal|<number>|<length>|<per` `centage>|inherit`	normal		Yes	Refer to font size of element itself	Visual

`'list-style-image'`	`<uri>|none|inherit`	none	Elements with
			`'display:`
			`list-item'`
				Yes		Visual

`'list-style-position'`	`inside|outside|inherit`	outside	Elements with
			`'display:`
			`list-item'`
				Yes		Visual

`'list-style-type'`	`disc|circle|square|decimal|`
	`decimal-leading-zero|lower-`
	`roman|upper-roman|lower-greek|`
	`lower-latin|upper-latin|arme`
	`nian|georgian|lower-alpha|`
	`upper-alpha|none|inherit`
		disc	Elements with
			`'display:`
			`list-item'`
				Yes		Visual

Name	Values	Initial value	Applies to (Default: all)	Inherited?	Percentages (Default: N/A)	Media groups
`'list-style'`	[`'list-style-type'` || `'list-style-position'` || `'list-style-image'`] | inherit	See individual properties	Elements with `'display: list-item'`	Yes		Visual
`'margin-right'` `'margin-left'`	`<margin-width>` | inherit	0	All elements except elements with table display types other than `table` and `inline-table`	No	Refer to width of containing block	Visual
`'margin-top'` `'margin-bottom'`	`<margin-width>` | inherit	0	All elements except elements with table display types other than `table` and `inline-table`	No	Refer to width of containing block	Visual
`'margin'`	`<margin-width>{1,4}` | inherit	See individual properties	All elements except elements with table display types other than `table` and `inline-table`	No	Refer to width of containing block	Visual
`'max-height'`	`<length>` | `<percentage>` | none | inherit	none	All elements except non-replaced inline elements, table columns, and column groups	No	See prose	Visual
`'max-width'`	`<length>` | `<percentage>` | none | inherit	none	All elements except non-replaced inline elements, table rows, and row groups	No	Refer to width of containing block	Visual
`'min-height'`	`<length>` | `<percentage>` | inherit	0	All elements except non-replaced inline elements, table columns, and column groups	No	See prose	Visual

Name	Values	Initial value	Applies to (Default: all)	Inherited?	Percentages (Default: N/A)	Media groups
'min-width'	`<length>` | `<percentage>` | `inherit`	0	All elements except non-replaced inline elements, table rows, and row groups	No	Refer to width of containing block	Visual
'orphans'	`<integer>` | `inherit`	2	Block-level elements	Yes		Visual, paged
'outline-color'	`<color>` | `invert` | `inherit`	invert		No		Visual, interactive
'outline-style'	`<border-style>` | `inherit`	none		No		Visual, interactive
'outline-width'	`<border-width>` | `inherit`	medium		No		Visual, interactive
'outline'	`[ 'outline-color' || 'outline-style' || 'outline-width' ] | inherit`	See individual properties		No		Visual, interactive
'overflow'	`visible` | `hidden` | `scroll` | `auto` | `inherit`	visible	Nonreplaced block-level elements, table cells, and inline-block elements	No		Visual
'padding-top' 'padding-right' 'padding-bottom' 'padding-left'	`<padding-width>` | `inherit`	0	All elements except elements with table display types other than `table`, `inline-table`, and `table-cell`	No	Refer to width of containing block	Visual
'padding'	`<padding-width>{1,4}` | `inherit`	See individual properties	All elements except elements with table display types other than `table`,	No	Refer to width of containing block	Visual

Name	Values	Initial value	Applies to (Default: all)	Inherited?	Percentages (Default: N/A)	Media groups
			inline-table, and table-cell			
'page-break-after'	auto|always|avoid|left|right|inherit	auto	Block-level elements	No		Visual, paged
'page-break-before'	auto|always|avoid|left|right|inherit	auto	Block-level elements	No		Visual, paged
'page-break-inside'	avoid|auto|inherit	auto	Block-level elements	Yes		Visual, paged
'position'	static|relative|absolute|fixed|inherit	static		No		Visual
'quotes'	[<string> <string>]+|none|inherit	Depends on user agent		Yes		Visual
'right'	<length>|<percentage>|auto|inherit	auto	Positioned elements	No	Refer to width of containing block	Visual
'table-layout'	auto|fixed|inherit	auto	'table' and 'inline-table' elements	No		Visual
'text-align'	left|right|center|justify|inherit	'left' if 'direction' is 'ltr'; 'right' if 'direction' is 'rtl'	Block-level elements, table cells, and inline blocks	Yes		Visual
'text-decoration'	none|[underline||overline||line-through||blink]|inherit	none		No (see prose)		Visual

Name	Values	Initial value	Applies to (Default: all)	Inherited?	Percentages (Default: N/A)	Media groups
'text-indent'	<length> | <percentage> | inherit	0	Block-level elements, table cells, and inline blocks	Yes	Refer to width of containing block	Visual
'text-transform'	capitalize | uppercase | lowercase | none | inherit	none		Yes		Visual
'top'	<length> | <percentage> | auto | inherit	auto	Positioned elements	no	Refer to height of containing block	Visual
'unicode-bidi'	normal | embed | bidi-override | inherit	normal	All elements, but see prose	No		Visual
'vertical-align'	baseline | sub | super | top | text-top | middle | bottom | text-bottom | <percentage> | <length> | inherit	baseline	inline-level and 'table-cell' elements	No	Refer to 'line-height' of the element itself	Visual
'visibility'	visible | hidden | collapse | inherit	visible		Yes		Visual
'white-space'	normal | pre | nowrap | pre-wrap | pre-line | inherit	normal		Yes		Visual
'widows'	<integer> | inherit	2	Block-level elements	Yes		Visual, paged
'width'	<length> | <percentage> | auto | inherit	auto	All elements except non-replaced inline elements, table rows, and row groups	No	Refer to width of containing block	Visual
'word-spacing'	normal | <length> | inherit	normal		Yes		Visual
'z-index'	auto | <integer> | inherit	auto	Positioned elements	No		Visual

Table B-2. Microsoft proprietary extensions to CSS

Name	Values	Initial value	Applies to (Default: all)	Inherited?	Percentages (Default: N/A)	Media groups
`'filter'`	See *http://tr.im/filterproperty*	N/A		No		Filter properties
`'overflow-x'`	`visible`|`scroll`|`hidden`|`auto`	`visible` (except for textarea, then initial value is `hidden`)		No		Visual
`'overflow-y'`	`visible`|`scroll`|`hidden`|`auto`	`visible` (except for textarea, then initial value is `auto`)				Visual
`'scrollbar-3dlight-color'`	`<color>`	Default color	Element with scroll bar	Yes		Visual
`'scrollbar-arrow-color'`	`<color>`	Default color	Element with scroll bar	Yes		Visual
`'scrollbar-base-color'`	`<color>`	Default color	Element with scroll bar	Yes		Visual
`'scrollbar-darkshadow-color'`	`<color>`	Default color	Element with scroll bar	Yes		Visual
`'scrollbar-face-color'`	`<color>`	Default color	Element with scroll bar	Yes		Visual
`'scrollbar-highlight-color'`	`<color>`	Default color	Element with scroll bar	Yes		Visual

Name	Values	Initial value	Applies to (Default: all)	Inherited?	Percentages (Default: N/A)	Media groups
`scrollbar-shadow-color`	`<color>`	Default color	Element with scroll bar	Yes		Visual
`zoom`	`normal`|`<number>`|`<percentage>`	normal		No	Yes	Visual
`background-position-x`	`<length>`|`<percentage>`|`left`|`center`|`right`	0%		No	Yes	Visual
`background-position-y`	`<length>`|`<percentage>`|`top`|`center`|`bottom`	0%		No	Yes	Visual
`ime-mode`	`auto`|`active`|`inactive`|`disabled`	auto		Yes		Visual
`layout-grid`	`mode`|`type`|`line`|`char`	both loose none none		Yes		Visual
`layout-grid-char`	`<length>`|`<percentage>`|`none`|`auto`	none		No	Yes	Visual
`layout-grid-line`	`<length>`|`<percentage>`|`none`|`auto`	none		No	Yes	Visual
`layout-grid-mode`	`both`|`none`|`line`|`char`	both		Yes		Visual
`layout-grid-type`	`loose`|`strict`|`fixed`	loose		Yes		Visual
`line-break`	`normal`|`strict`	normal		Yes		Visual
`text-autospace`	`none`|`ideograph-alpha`|`ideograph-numeric`|`ideograph-parenthesis`|`ideograph-space`	none		No		Visual

Name	Values	Initial value	Applies to (Default: all)	Inherited?	Percentages (Default: N/A)	Media groups
'text-justify'	auto|distribute| distribute-all-lines| distribute-center-last| inter-cluster| inter-ideograph|inter- word|kashida|newspaper	auto		Yes		Visual
'text-kashida-space'	<percentage>|inherit	0%		Yes		Visual
'text-underline-position'	above|below|auto|auto-pos	auto		Yes		visual
'word-break'	normal|break-all|keep-all	normal		Yes		Visual
'word-wrap'	normal|break-word	normal		Yes		Visual
'writing-mode'	lr-tb|tb-rl	lr-tb		No		Visual

Table B-3. Mozilla CSS property extensions

Name	Values	Initial value	Applies to (Default: all)	Inherited?
`'-moz-appearance'`	`none|button|button-small|checkbox|checkbox-container|checkbox-small|dialog|listbox|menu|menuitem|menulist|menulist-button|menulist-textfield|progressbar|radio|radio-container|radio-small|resizer|scrollbar|scrollbarbutton-down|scrollbarbutton-left|scrollbarbutton-right|scrollbarbutton-up|scrollbartrack-horizontal|scrollbartrack-vertical|separator|statusbar|tab|tab-left-edge|tabpanels|textfield|toolbar|toolbarbutton|toolbox|tooltip|treeheadercell|treeheadersortarrow|treeitem|treetwisty|treetwistyopen|treeview|window`	none		No
`'-moz-binding'`	`uri|none`	none		No
`'-moz-background-clip'`	`border|padding`	border		No
`'-moz-background-inline-policy'`	`bounding-box|continuous|each-box`	continuous	Inline elements	No
`'-moz-background-origin'`	`border|padding|content`	padding		No
`'-moz-background-image'`	`[ <uri>|none]|[number|<percentage>]` `[ stretch|repeat|round ]`	none		No

Name	Values	Initial value	Applies to (Default: all)	Inherited?
	none|<image>			
	[<number>|<per centage>]			
	[<border-width>]			
	[stretch|repeat| round]			
'-moz-border-bottom-colors' '-moz-border-left-colors' '-moz-border-right-colors' '-moz-border-top-colors'	<color>|transpar ent	N/A		No
'-moz-border-radius'	<length>|<percent age>	0		No
'-moz-border-radius-bottom left' '-moz-border-radius-bottom right' '-moz-border-radius-top left' '-moz-border-radius-top right'	<length>|<percent age>	0		No
'-moz-box-align'	start|center|end| baseline|stretch	stretch	Elements with a CSS display value of -moz-box or -moz-inline-box	No
'-moz-box-direction'	normal|reverse	normal	Elements with a CSS display value of -moz-box or -moz-inline-box	No
'-moz-box-flex'	0|>0	0	Elements with a CSS display value of -moz-box or -moz-inline-box	No
'-moz-box-orient'	horizontal| vertical	horizontal	Elements with a CSS display value of -moz-box or -moz-inline-box	No
'-moz-box-pack'	start|center|end| justify	start	Elements with a CSS display value of -moz-box or -moz-inline-box	No

Name	Values	Initial value	Applies to (Default: all)	Inherited?
'-moz-box-shadow'	none|[<color> <length> <length> <length> <length>| <length> <length> <length> <length>? <color>]|inherit	none		No
'-moz-box-sizing'	content-box|bor der-box|padding box	content-box		No
'-moz-image-region'	For rect() values, a rect consisting of four computed lengths	auto	XUL image elements and :-moz-tree-image, :-moz-tree-twisty, and :-moz-tree-checkbox pseudo-elements	Yes
'-moz-opacity'	0 (or less)|0 < number < 1|1 (or more)	1		No
'-moz-outline'	-moz-outline-color |-moz-outline-style| -moz-outline-width	See individual properties		No
'-moz-outline-color'	<color>|invert	invert		No
'-moz-outline-offset'	<number>	0		No
'-moz-outline-radius'	<length>| <percentage>	0		No
'-moz-outline-radius-bottom left'	<length>| <percentage>	0		No
'-moz-outline-radius-bottom right'				
'-moz-outline-radius-top left'				
'-moz-outline-radius-top right'				
'-moz-outline-style'	none|dotted|dashed| solid|double| groove|ridge|inset| outset	none		No
'-moz-outline-width'	<width>	medium		No
'-moz-user-focus'	ignore|normal			

Name	Values	Initial value	Applies to (Default: all)	Inherited?
'-moz-outline-input'	none|enabled|disa bled|inherit	none		Yes
'-moz-outline-select'	none|text| -moz-none			No
'-moz-window-shadow'	default|none	default	All elements that make native windows; e.g., window, panel	No

CSS 2.1 Selectors, Pseudo-Classes, and Pseudo-Elements

This appendix contains three reference tables that can help you learn how to apply styles to the correct elements.

Table C-1 contains the CSS 2.1 selectors. Selectors help tell the browser where to apply the CSS declarations.

> In the "Generic pattern" column of Table C-1, the values C, R, and S take the place of type selectors.

Table C-2 contains a list of pseudo-classes. A pseudo-class is a device by which a browser applies an invisible class on its own. For example, through the pseudo-class we are able to define properties for various visited, active, and hover states of the ubiquitous link.

Table C-3 contains a list of pseudo-elements. Similar in nature to a pseudo-class, a pseudo-element places invisible tags around content in a web page and then applies styles to that element. Since the structure is more like a typical element than a class, these elements are called pseudo-elements.

Table C-1. CSS 2.1 selectors

Selector	Generic pattern	Description	Sample
Universal	*	Matches with any element	`* { text-decoration: none; }`
Type	C	Matches any element; in this example, all h2 elements	`h2 { font-weight: normal; }`

Selector	Generic pattern	Description	Sample
Descendant	C R S	Matches any S element that is a descendant of element R, which is a descendant of element C	`div#content p em { background-color: yellow; }`
Child	C > S	Selects any S element that is a child of a C element	`li > ul { list-style-type: circle;}`
Adjacent sibling	C + S	Selects any S element that immediately follows a C element	`div#content+p { text-indent: 0;}`
Grouping	C, R, S	Several selectors utilize the same declaration(s)	`h1, h2, h3, h4 { color: #0cf;}`
Class	C.classR	Selects any C element that contains a class attribute with a value of `classR`	`img.content { padding: 4px; border: 1px solid black; }`
ID	C#idR	Selects any C element that contains an `id` attribute with a value of `idR`	`div#content { color: #333;}`
Attribute selector	C[attribute]	Selects any C element that contains the attribute	`a[link] {text-decoration: none;}`
Attribute selector	C[attribute="valueR"]	Selects any C element that contains the attribute with a value of `valueR`	`input[type="text"] { width: 33%; }`
Attribute selector	C[attribute~="valueR"]	Selects any C element that contains the attribute whose value is a space-separated list of words and one of the words in that list matches `valueR`	`div.advertisement form[class~="login"] { float: left; margin-left: 7px; }`

Selector	Generic pattern	Description	Sample
Attribute selector	C[attribute|="valueR"]	Selects any C element that contains the attribute whose value is a hyphen-separated list of words and the first word matches valueR	.warning[lang="uk"]:after { content: " Blimey!"}

Table C-2. CSS 2.1 pseudo-classes

Pseudo-class	Generic pattern	Description	Sample
:first-child	C:first-child	Matches element C that is the first child in another element	divs p:first-child {color: white; background-color: red; }
:link	C:link	Matches any unvisited link of element C	a:link {text-decoration: none; }
:visited	C:visited	Matches any visited link of element C	a:visited {font-weight: normal; }
:hover	C:hover	Matches the C element a user has selected (typically by moving the cursor icon over a link) but not activated	a:hover { background-color: orange; }
:active	C:active	Matches the C element a user has activated	a:active { color: green; }
:focus	C:focus	Matches the C element that contains the focus (typically an input field of a form)	input:focus { background-color: #F7F7D5;}
:lang	C:lang(R)	Matches the C element that uses the language R	p:lang(en) {font-weight: bold;}

Table C-3. CSS 2.1 pseudo-elements

Pseudo-element	Generic pattern	Description	Sample
`:first-line`	`C:first-line`	Selects the first line of text in the C element	`h2+p:first-line {color: #727977;}`
`:first-letter`	`C:first-letter`	Selects the first letter in the C element	`h1:first-letter { font-size: 66%; text-transform: lower case; }`
`:before`	`C:before`	Places generated content before an element; used with the content property	`ul.tracklisting li:before {content: "Song title: ";}`
`:after`	`C:after`	Places generated content after an element; used with the content property	`div#footer p.copyright:after {content: "ouble true!";}`

CSS3 Selectors and Pseudo-Classes

Although CSS3 is still being worked on as this book is being written, some browser vendors are starting to support properties from the unfinished specification. This appendix provides a listing of the new CSS3 selectors for your reference.

To get a solid idea of what tools you have at your disposal to apply styles to a document, consult Appendix C (which covers CSS 2.1 selectors) in conjunction with this listing.

Table D-1 describes the new CSS3 selector, the general sibling selector.

> In the "Generic pattern" column of Table D-1, the values `C`, `R`, and `S` take the place of type selectors.

Table D-1. New CSS3 selector

Selector	Generic pattern	Description	Sample
General sibling combinator	`C ~ R`	Matches any R element that is preceded by a C element	`#content ~ img { padding: 2px; border 2px solid black; }`

Table D-2 contains a list of new attribute selectors. These allow greater control when selecting elements in a document based on an attribute's value or even a small portion of that value.

Table D-2. CSS3 attribute selectors

Selector	Generic pattern	Description	Sample
Attribute selector	`C[attribute^="valueR"]`	Selects any C element that contains the attribute that begins with the value of valueR	`img[alt^="mugshot"] { width: 100px; }`
Attribute selector	`C[attribute$="valueR"]`	Selects any C element that contains the attribute whose value exactly matches valueR	`img[alt$="photo"] { border: 10px solid red; }`
Attribute selector	`C[attribute*="valueR"]`	Selects any C element that contains the substring valueR	`img[alt*="execu tive"] { }`
Attribute selector	`C[attribute|="valueR"]`	Selects any C element that contains an attribute value that's a list of hyphen-separated values	`img[alt|="non"] { opacity: .5;}`

Table D-3 contains a list of structural pseudo-elements. These allow you to pick out elements based on their order. For example, you can pinpoint the odd-numbered `li` elements using the `nth-child` selector instead of using the `class` attribute selector.

Table D-3. CSS3 structural pseudo-classes

Pseudo-class	Generic pattern	Description	Sample
`:last-child`	`C:last-child`	Matches element C that is the last child in another element	`divs p:last-child {color: white; background-color: black; }`
`:target`	`C:target`	Matches the C element when activating a fragment link (e.g., `#section`)	`#section:target {background-color: yellow;}`
`:enabled`	`C:enabled`	Matches the C element when the C element is in an enabled state	`input[type="age"]:enabled {background-color: green;}`
`:disabled`	`C:disabled`	Matches the C element when the C element is in a disabled state	`input[type="password"]:disabled {background-color: #999;}`
`:root`	`:root`	Matches the root of the document; in HTML4 documents, this is the HTML element	`:root {display: block;}`
`:nth-child()`	`C:nth-child(an+b)`	Matches elements in a document tree that have a certain number of siblings before it; where n is an integer, `:nth-child(an+b)` would match the element that has an+b – 1 siblings before it	`tr:nth-child(2n) {background-color: #99ff99;}`
`:nth-last-child()`	`C:nth-last-child(an+b)`	Matches elements in a document tree that have a certain number of siblings after it; where n is an integer, `:nth-last-child(an+b)` would match the element that has an+b – 1 siblings before it	`tr:nth-last-child(-2n) {background-color: #99ff99;}`
`:nth-of-type()`	`C:nth-of-type(an+b)`	Matches elements in a document tree that have a certain number of siblings before it; where n is an integer, `:nth-of-type(an+b)` would match the element that has an+b – 1 siblings before it	`tr:nth-of-type(2n) {float:right;}`

Pseudo-class	Generic pattern	Description	Sample
:nth-last-of-type()	C:nth-last-of-type(an+b)	Matches elements in a document tree that have a certain number of siblings after it; where n is an integer :nth-of-type(an+b) would match the element that has an+b-1 siblings before it	`tr:nth-last-of-type(2n) {float::right;}`
:first-of-type	C:first-of-type	Matches the first child element of the specified element type	`p:first-of-type {font-weight: bold;}`
:last-of-type	C:last-of-type	Matches the last child element of the specified element type	`p:last-of-type {background-color: black;}`
:only-child	C:only-child	Matches the child element if it is the only child element of its parent	`li:only-child {font-size: 2em;}`
:only-of-type	C:only-of-type	Matches the child element if it is the only child element of its parent	`li:only-of-type {font-size: 2em;}`
:empty	C:empty	Matches any element that has no children	`*:empty {background: red; height: 100px;}`
:not()	C:not(R)	Matches all elements within the C element, except the R elements	`*:not(:hover) {opacity: .7;}`

Styling of Form Elements

With the impact that forms have on our day-to-day Internet commerce and lifestyle, forms are always in the foreground of website design.

Web designers want to control the look and feel of form elements in their web page design so that they are more appealing to their audience and also fit in with the rest of the design.

The problem is that browsers manipulate the visual display of form elements from one browser to the next. Even the same browser version can display a form element differently on separate operating systems.

This appendix covers how browsers don't render form controls consistently. Since there are about 10 browsers and 20 CSS properties reviewed, as well as 8 HTML form elements, the entire appendix is too large to print. So, we took it to the Internet and made it available online for free. If you're viewing this appendix as a standalone piece online, you can access the full book here: *http://oreilly.com/catalog/9780596155933/*.

Anatomy of the Appendix

The first part of this appendix lists the properties and their respective values that were tested (as shown in Table E-1).

The second part examines eight form elements and how they can be modified using 20 CSS properties (listed in Table E-1) in 10 different browsers:

- Checkboxes, as shown in Table E-2 and Figure E-1 to Figure E-20
- File upload, as shown in Table E-3 and Figure E-21 to Figure E-40
- Radio buttons, as shown in Table E-4 and Figure E-41 to Figure E-60
- Input text, as shown in Table E-5 and Figure E-61 to Figure E-80
- Select with multiple items, as shown in Table E-6 and Figure E-81 to Figure E-100
- Select with an individual item, as shown in Table E-7 and Figure E-101 to Figure E-120

- Submit button, as shown in Table E-8 and Figure E-121 to Figure E-140
- `Textarea`, as shown in Table E-9 and Figure E-141 to Figure E-160

The values used in Tables E-2 to E-9 include NA, Y, N, and S.

NA stands for *Not Available* (meaning that the CSS property does not apply to the form element), *Y* for *Yes* (meaning that the CSS property's value is properly applied), *N* for *No* (meaning that the CSS property's value was not applied), and *S* for *Somewhat* (meaning that some part of the CSS property's value is being applied).

> The *Somewhat* value marks unusual situations. There are points within the HTML and CSS specifications that do not define a certain behavior, and therefore determination of a CSS rule's successful application becomes difficult.
>
> For example, Firefox expands the width of the input field as well as the space between letters when using the `letter-spacing` property.
>
> In this instance, the discrepancy could be due to Firefox calculating the default width of the input field on a certain number of characters, whereas the other browsers could be basing the width on a predetermined value or an unadjusted number of characters at the font size of the input field.

Tested CSS Properties

Table E-1. The properties and their values used in testing form elements

Property	Value
background-color	#ccff00;
background-image	url(checkerboard_bkgd.gif);
border	0;
border-color	1px solid red;
border-style	groove;
border-width	24px;
color	#00ccff;
font-family	Georgia, Times, 'Times New Roman', serif;
font-size	24px;
font-weight	bold;
height	100px;
letter-spacing	24px;
line-height	1.5;
margin	24px;

Property	Value
padding	24px;
text-align	right;
text-decoration	underline;
text-indent	24px;
width	100px;
word-spacing	24px;

Input Element for Checkboxes

A checkbox element is a form element, which allows on/off selections for one or multiple items for a grouping. An example of a checkbox is one that lets you select which ingredients you would like on a pizza.

Table E-2. A review of the CSS properties on checkboxes

	WinIE6	WinIE7	WinIE8b2	Chrome	MacFF3	WinFF3	MacOP9	WinOP9	MacSF3	WinSF3	MacSF4b	WinSF4b
background-color	S	S	S	N	N	N	Y	Y	N	N	N	N
background-image	S	S	S	N	N	N	Y	Y	N	N	N	N
border	N	N	N	N	N	N	N	N	N	N	N	N
border-color	S	S	S	N	N	N	S	S	N	N	N	N
border-style	S	S	S	N	N	N	Y	Y	N	N	N	N
border-width	N	N	N	N	N	N	N	N	N	N	N	N
color	N	N	N	N	N	N	N	N	N	N	N	N
font-family	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA
font-size	N	N	N	N	N	N	N	N	N	N	N	N
font-weight	N	N	N	N	N	N	N	N	N	N	N	N
height	S	S	S	S	N	S	S	S	S	S	S	S
letter-spacing	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA
line-height	N	N	N	N	N	N	N	N	N	N	N	N
margin	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y
padding	N	N	Y	N	N	N	Y	Y	N	N	N	N
text-align	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA
text-decoration	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA
text-indent	Y	S	N	N	N	S	N	N	N	N	N	N
width	S	S	S	S	N	S	S	S	S	S	S	S
word-spacing	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA

background-color

Figure E-1. Testing the background color of checkboxes

background-image
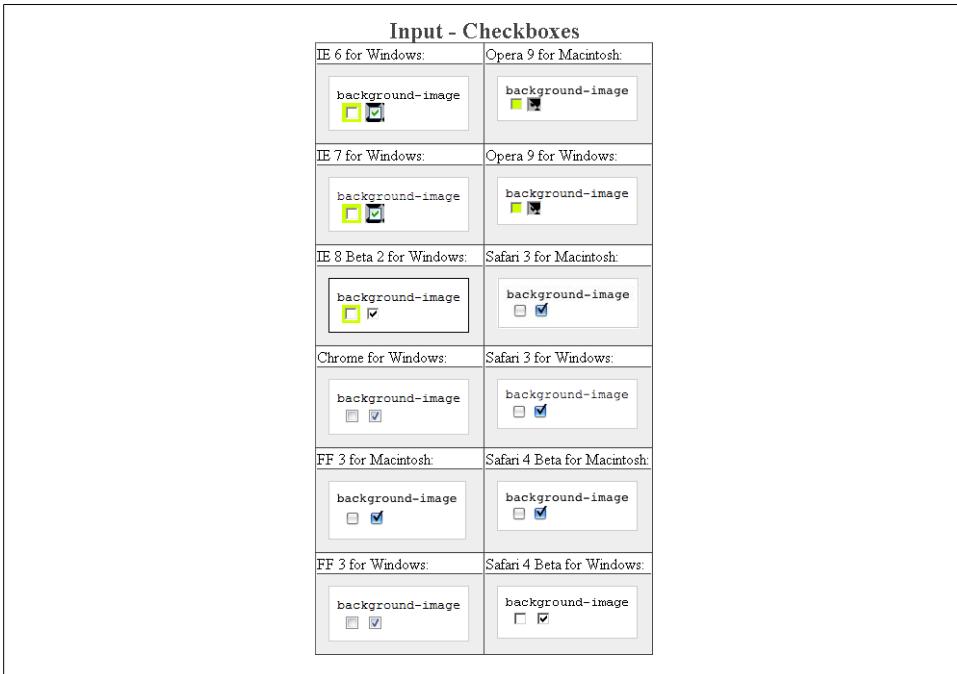
Figure E-2. Testing background images in checkboxes

border: 0;

Figure E-3. Testing the removal of borders from checkboxes

border-color
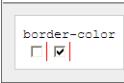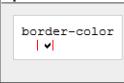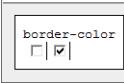
Figure E-4. Testing colors on the checkbox borders

border-style
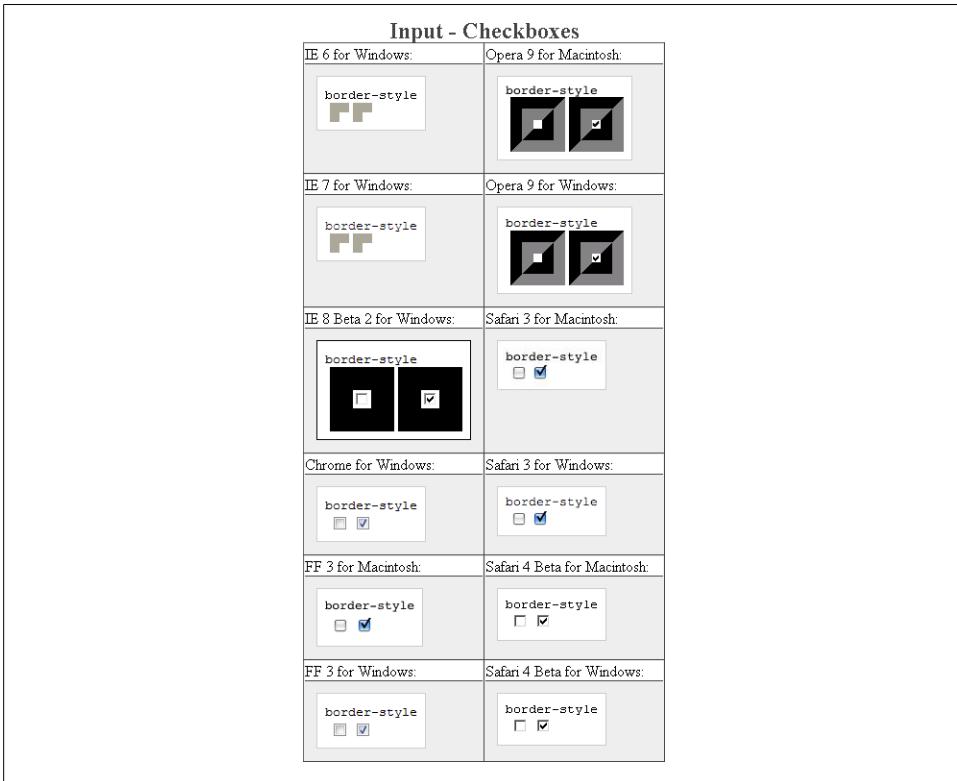
Figure E-5. Testing the styles of borders on checkboxes

border-width
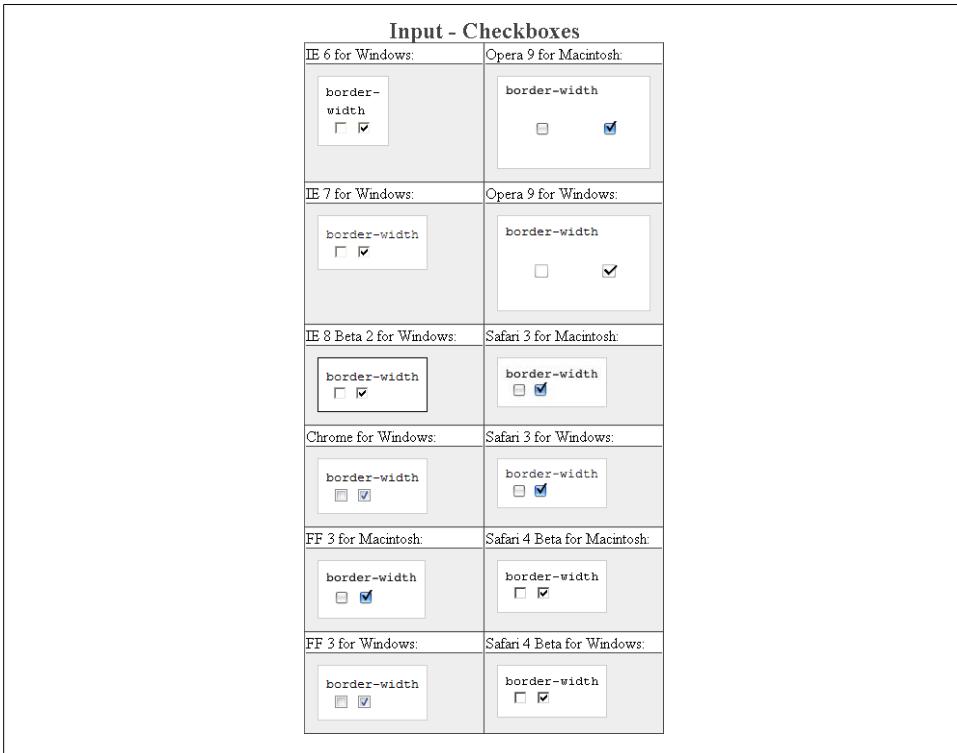
Figure E-6. Testing the widths of borders on checkboxes

color

Figure E-7. Testing the color of checkboxes

font-family

Figure E-8. Testing setting a different font on checkboxes

font-size

Figure E-9. Testing a different size of font on checkboxes

font-weight

Figure E-10. Testing a bold font on checkboxes

height
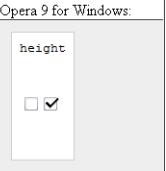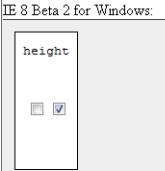
Figure E-11. Testing setting a height on checkboxes

letter-spacing

Figure E-12. Testing the letter spacing on checkboxes

line-height

Figure E-13. Testing setting the spacing between lines of text on checkboxes

margin
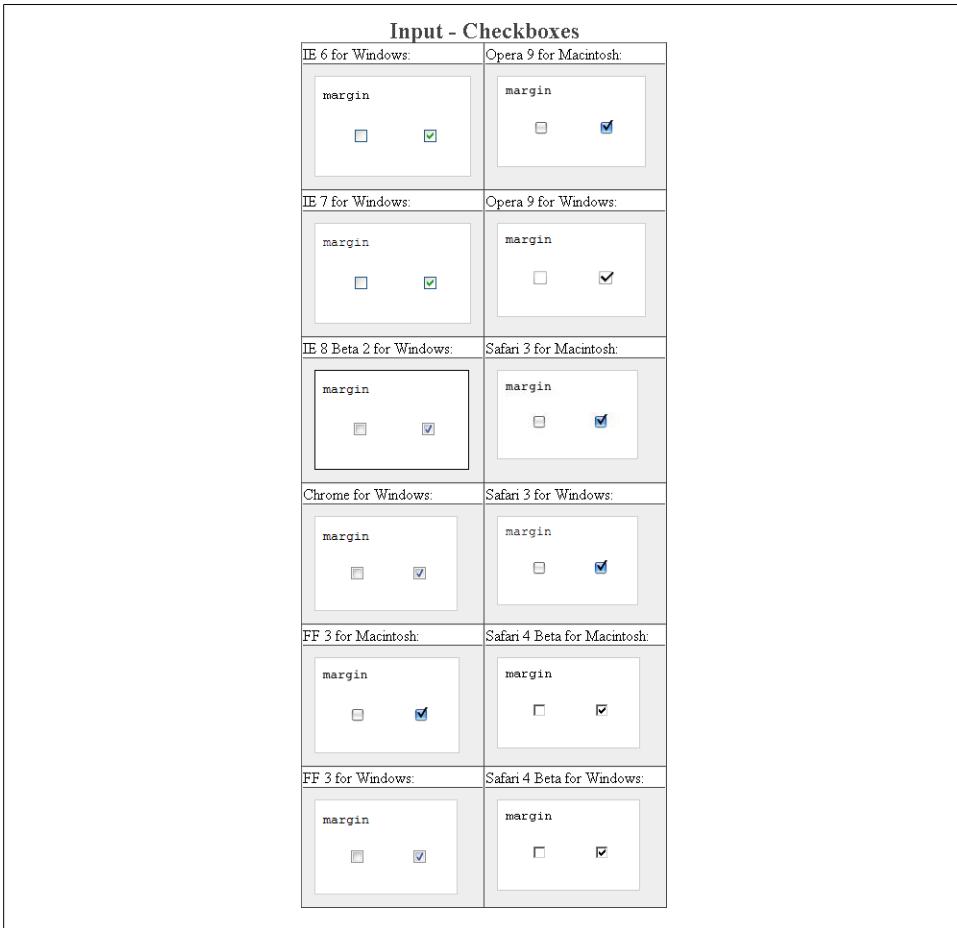
Figure E-14. Testing margins on checkboxes

padding
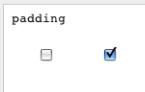
Figure E-15. Testing padding on checkboxes

text-align

Figure E-16. Testing the alignment of text on checkboxes

text-decoration

Figure E-17. Testing setting a different font on checkboxes

text-indent

Figure E-18. Testing indenting the text on checkboxes

width
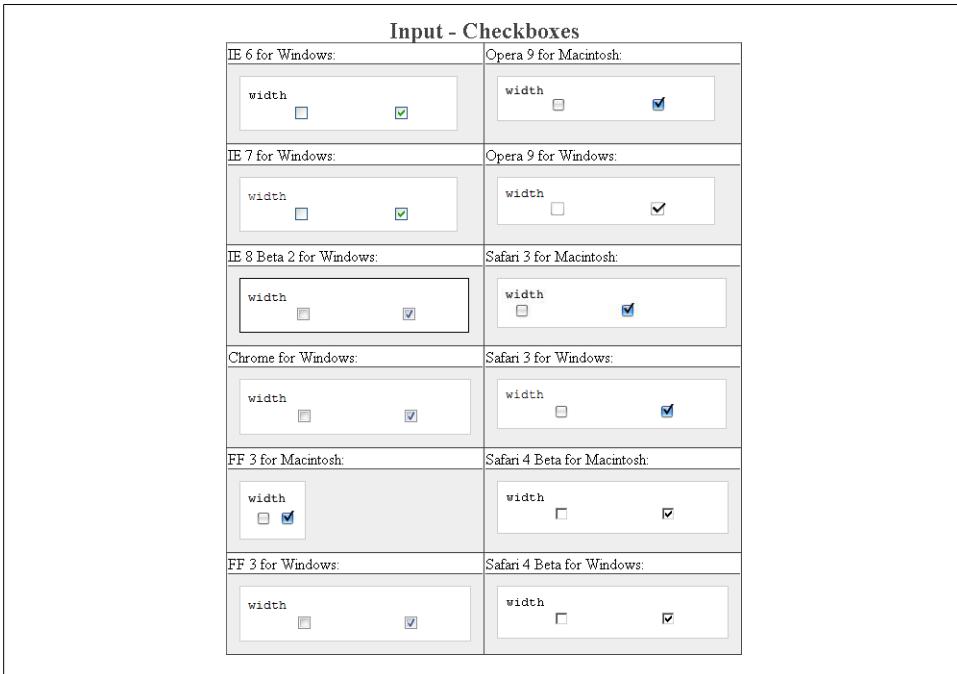
Figure E-19. Testing the width of checkboxes

word-spacing
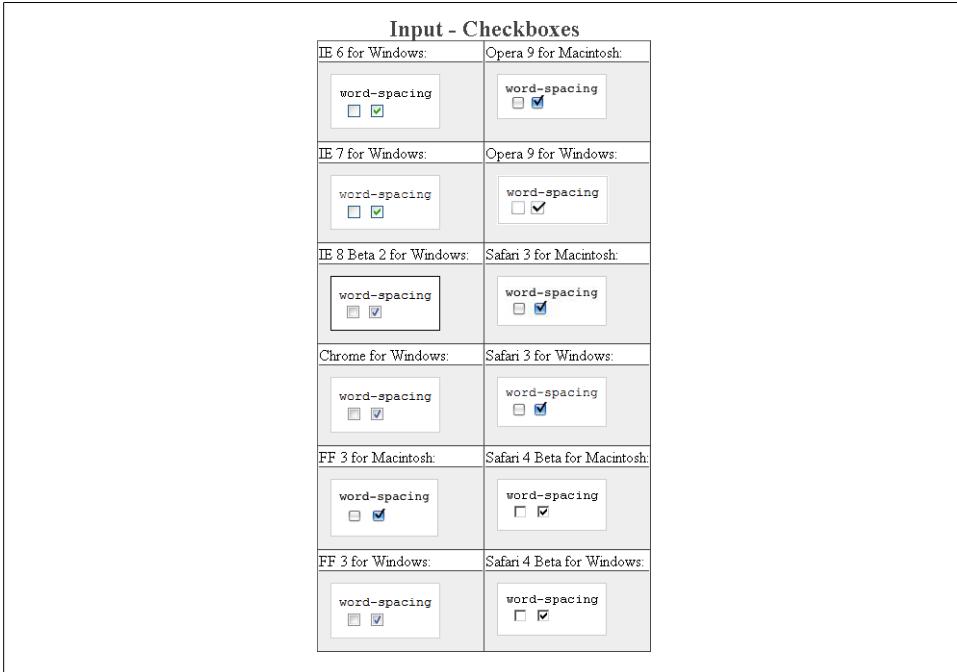
Figure E-20. Testing the spacing between words on checkboxes

Input Element for File Uploads

Also known as file select, this form element allows users to pick a file from their computer for submission along with a form.

Table E-3. A review of the CSS properties on file upload

	WinIE6	WinIE7	WinIE8b2	Chrome	MacFF3	WinFF3	MacOP9	WinOP9	MacSF3	WinSF3	MacSF4b	WinSF4b
background-color	Y	Y	Y	Y	N	N	Y	Y	Y	Y	Y	Y
background-image	Y	Y	Y	Y	N	N	Y	Y	Y	Y	Y	Y
border	Y	Y	Y	N	N	N	Y	Y	N	N	N	N
border-color	Y	Y	Y	S	N	N	S	S	S	S	S	S
border-style	Y	Y	Y	Y	N	N	Y	Y	Y	Y	Y	Y
border-width	Y	Y	Y	N	N	N	S	S	N	N	N	N
color	N	N	N	Y	N	N	Y	N	Y	Y	Y	Y
font-family	N	N	N	Y	N	N	Y	Y	N	N	Y	Y
font-size	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y
font-weight	N	N	N	Y	N	N	Y	Y	Y	Y	Y	Y
height	Y	Y	Y	S	S	S	Y	Y	S	S	S	S
letter-spacing	N	N	N	Y	Y	Y	Y	Y	Y	Y	Y	Y
line-height	N	N	N	N	N	N	N	N	N	N	N	N
margin	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y
padding	Y	Y	Y	N	N	N	Y	Y	N	N	N	N
text-align	N	N	N	N	N	N	N	N	N	N	N	N
text-decoration	N	N	N	Y	N	N	N	N	Y	Y	Y	Y
text-indent	Y	Y	N	Y	Y	Y	N	N	Y	Y	Y	Y
width	Y	Y	Y	Y	S	S	Y	Y	Y	Y	Y	Y
word-spacing	N	N	N	Y	N	N	N	N	Y	Y	S	S
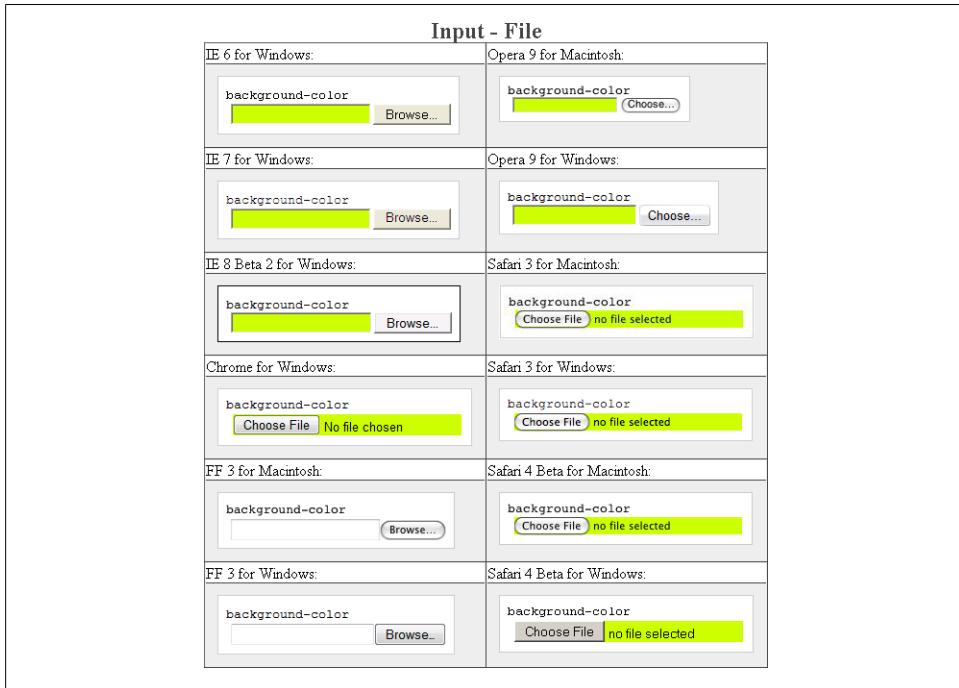
background-color

Figure E-21. Testing the background color of file input

background-image
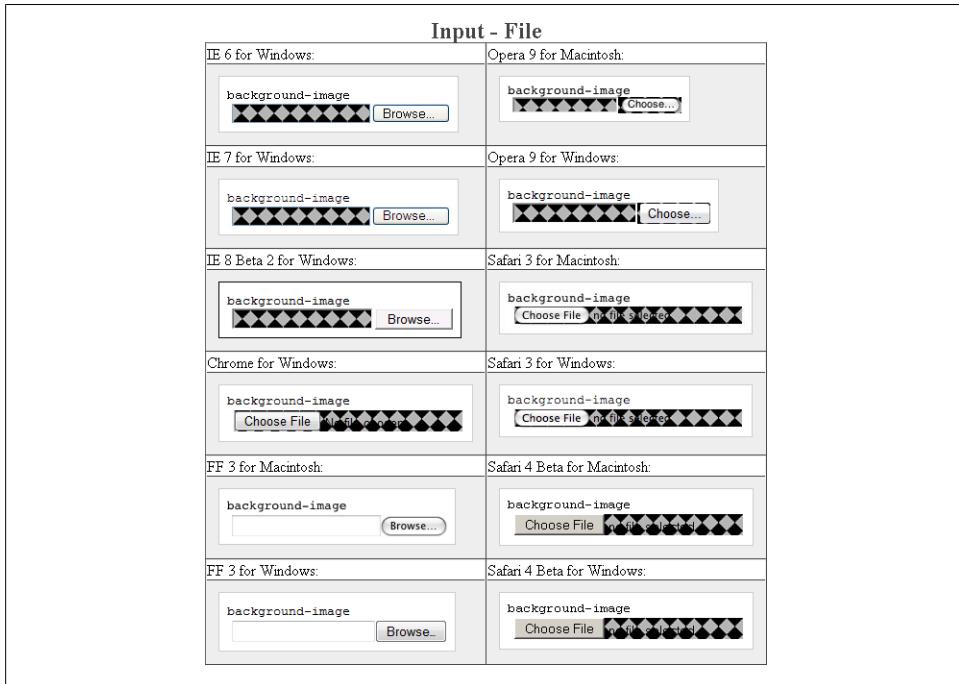
Figure E-22. Testing background images in file input

border: 0;
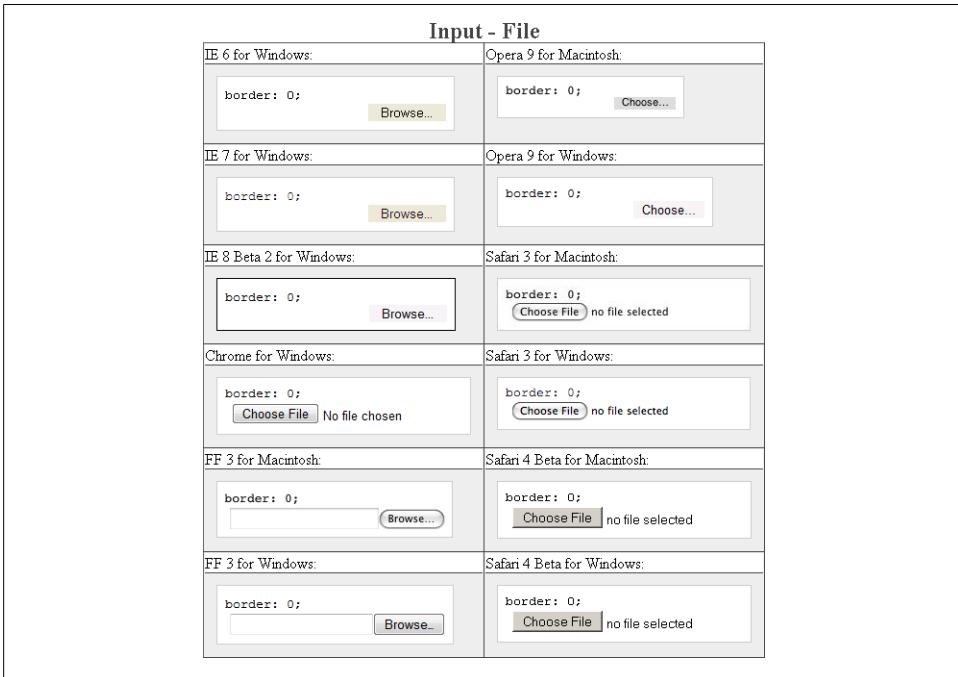
Figure E-23. Testing the removal of borders on file input

border-color
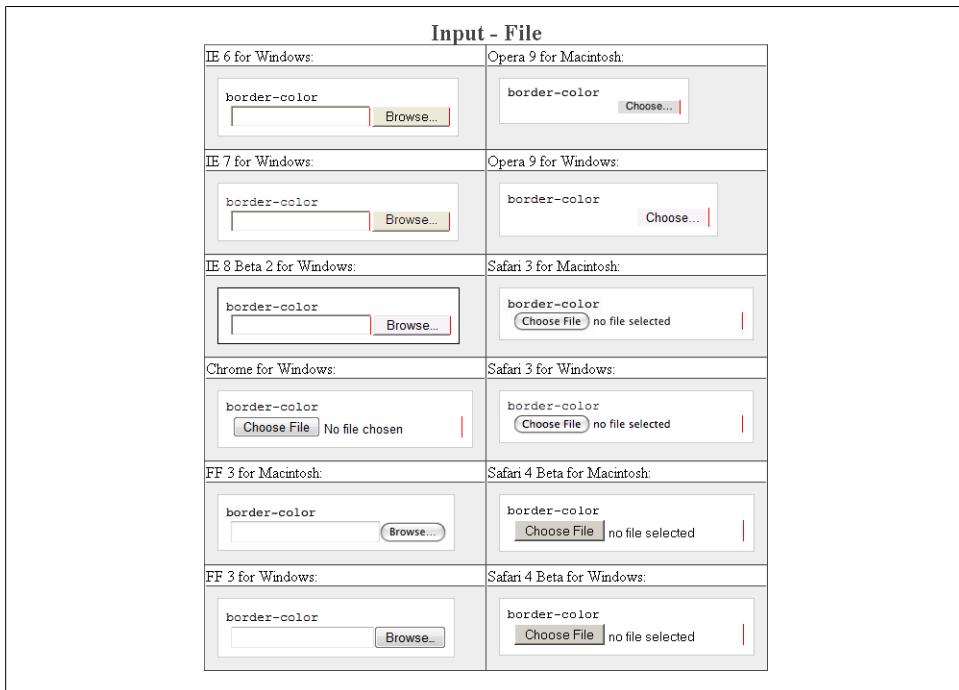
Figure E-24. Testing colors on the file input borders

border-style

Figure E-25. Testing the styles of borders of file input

border-width
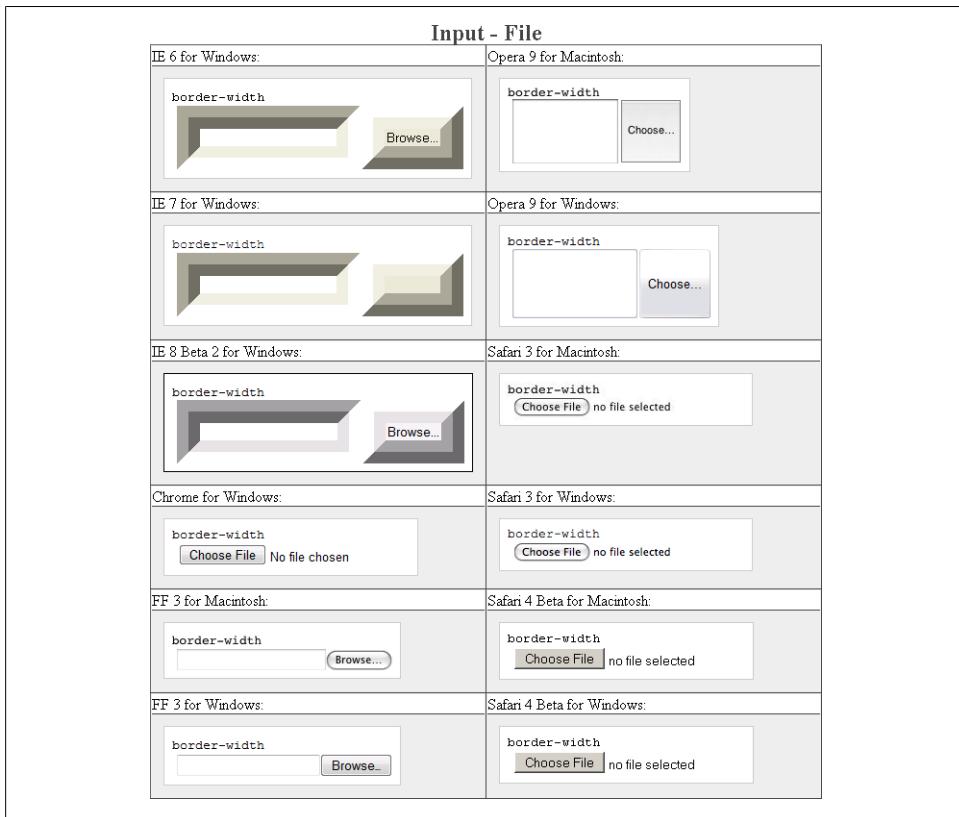
Figure E-26. Testing the widths of borders on file input

color
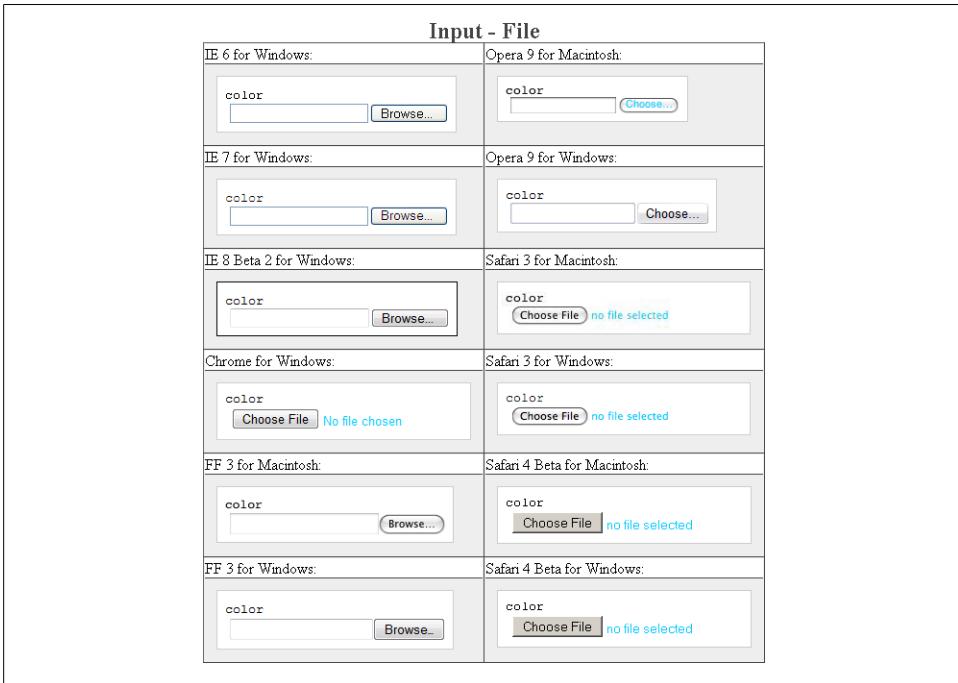
Figure E-27. Testing the color on file input

font-family

Figure E-28. Testing setting a different font on file input

font-size
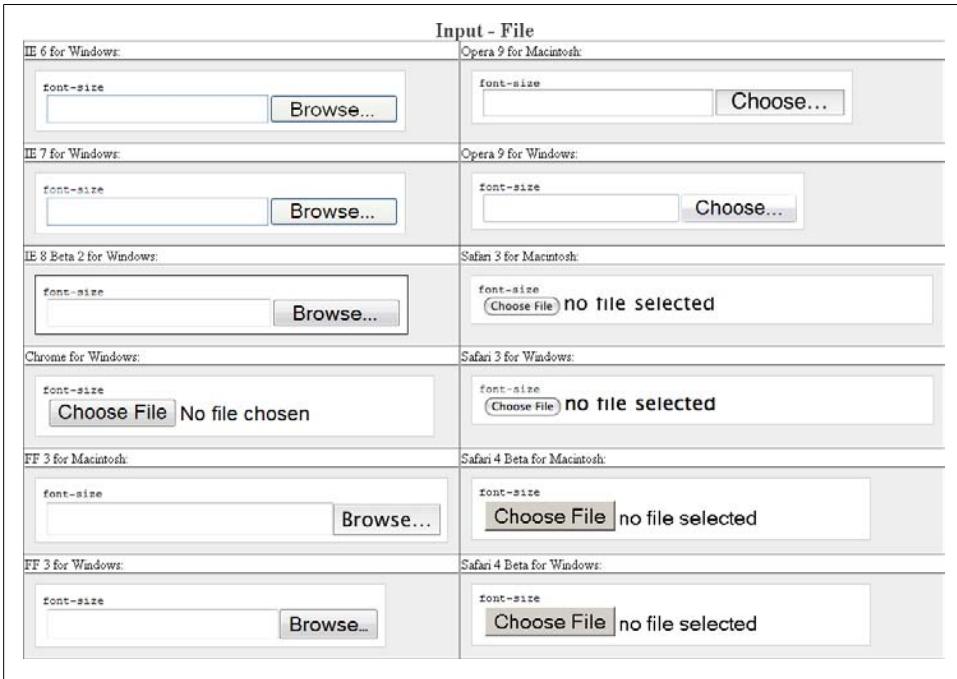
Figure E-29. Testing a different size of font on file input

font-weight
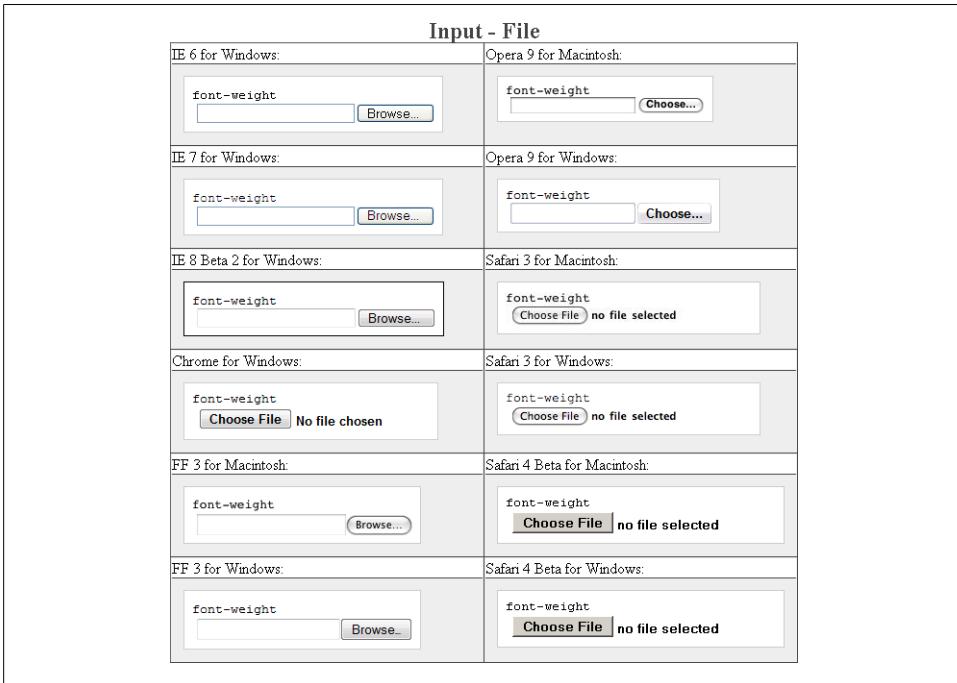
Figure E-30. Testing a bold font on file input

height
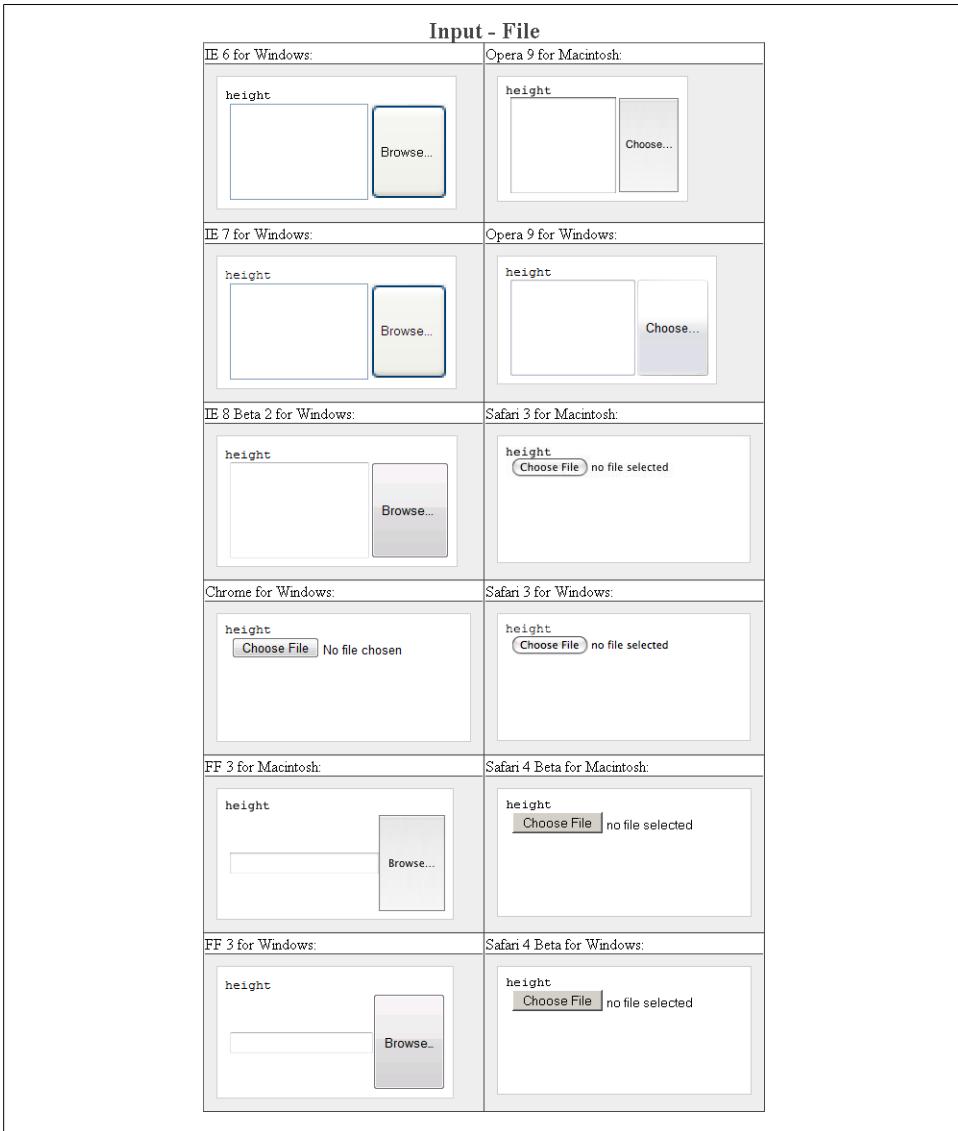
Figure E-31. Testing setting a height for file input

letter-spacing
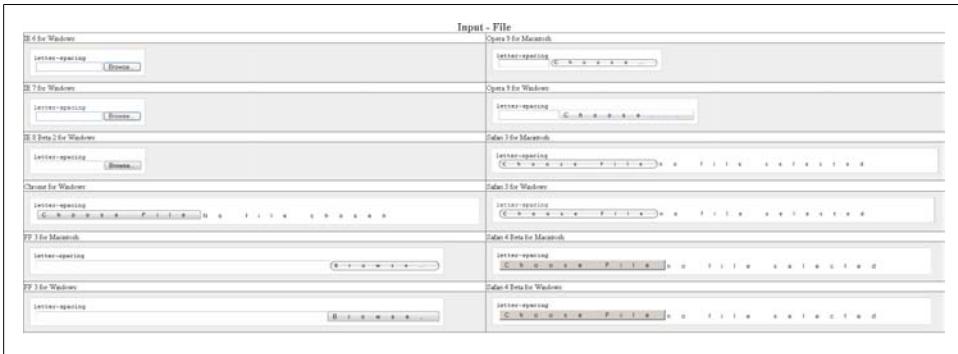
Figure E-32. Testing the letter spacing of file input

line-height
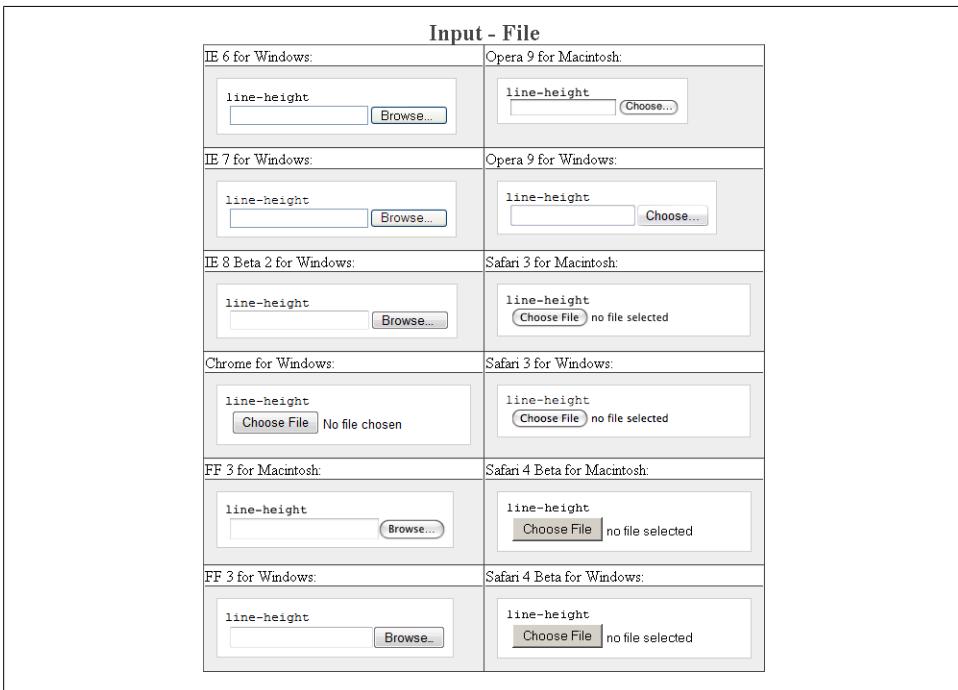
Figure E-33. Testing setting the spacing between lines of text on file input

margin
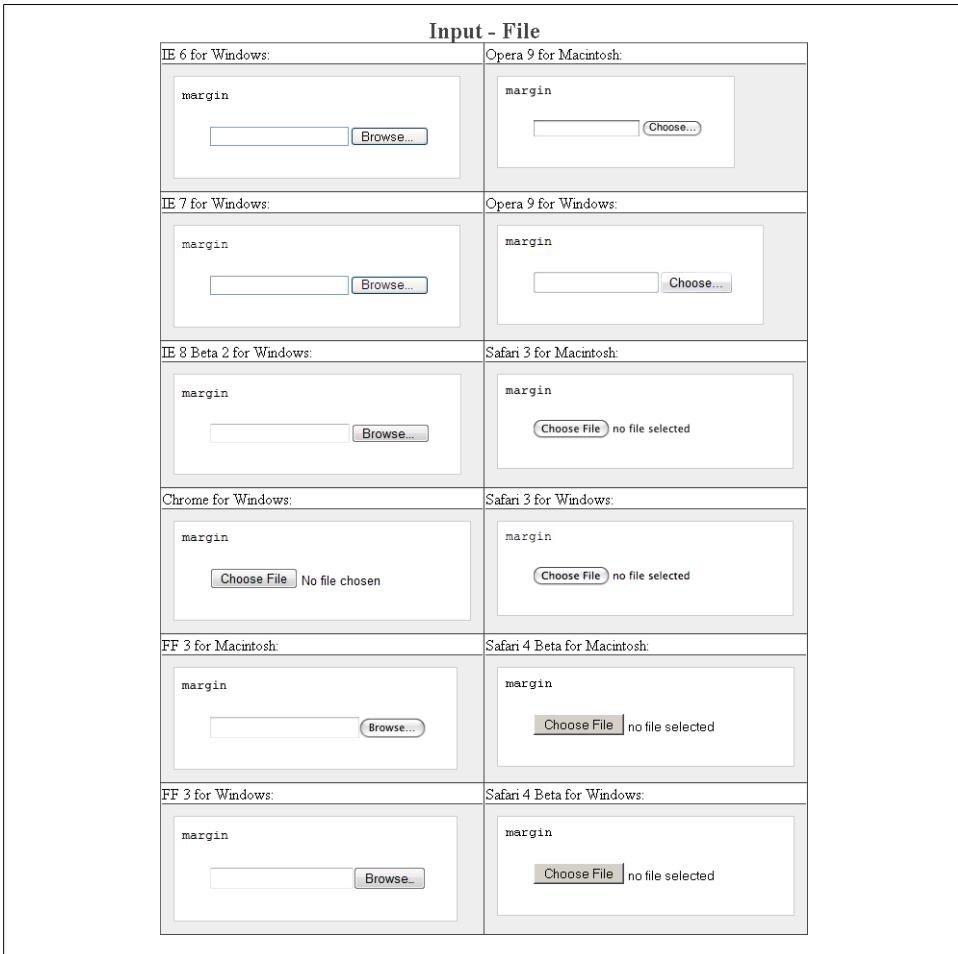
Figure E-34. Testing margins on file input

padding
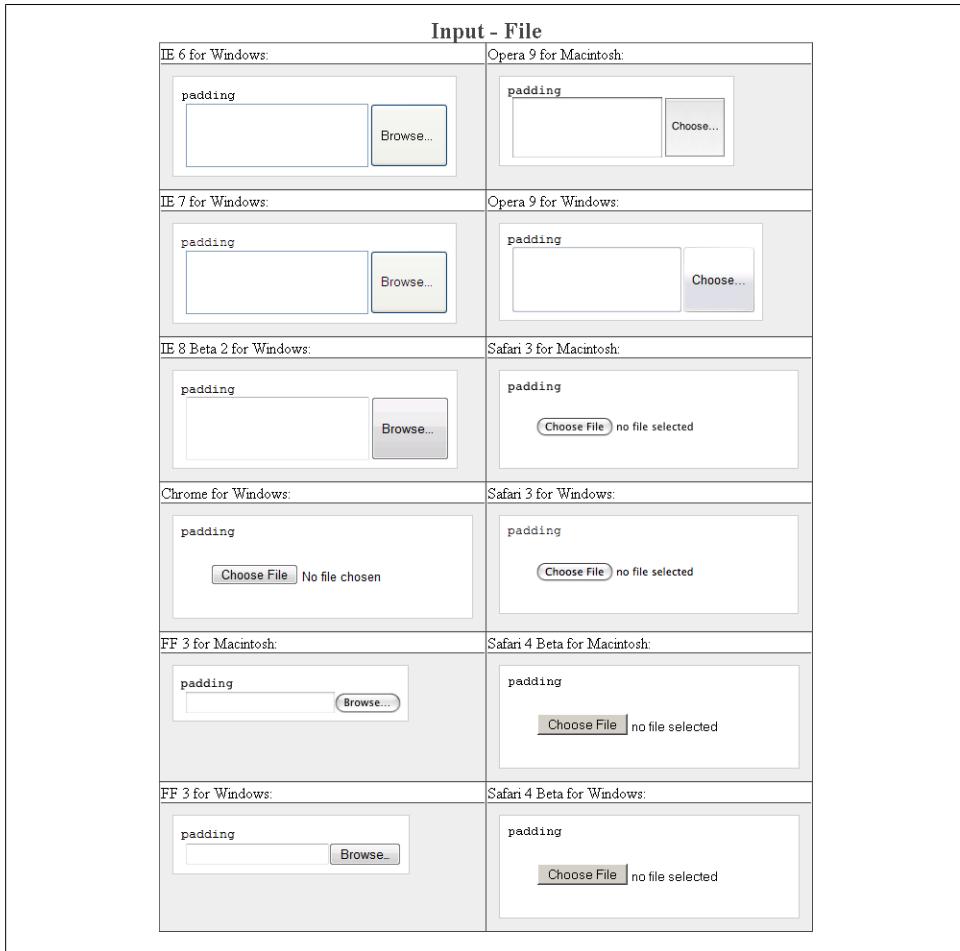
Figure E-35. Testing padding on file input

text-align
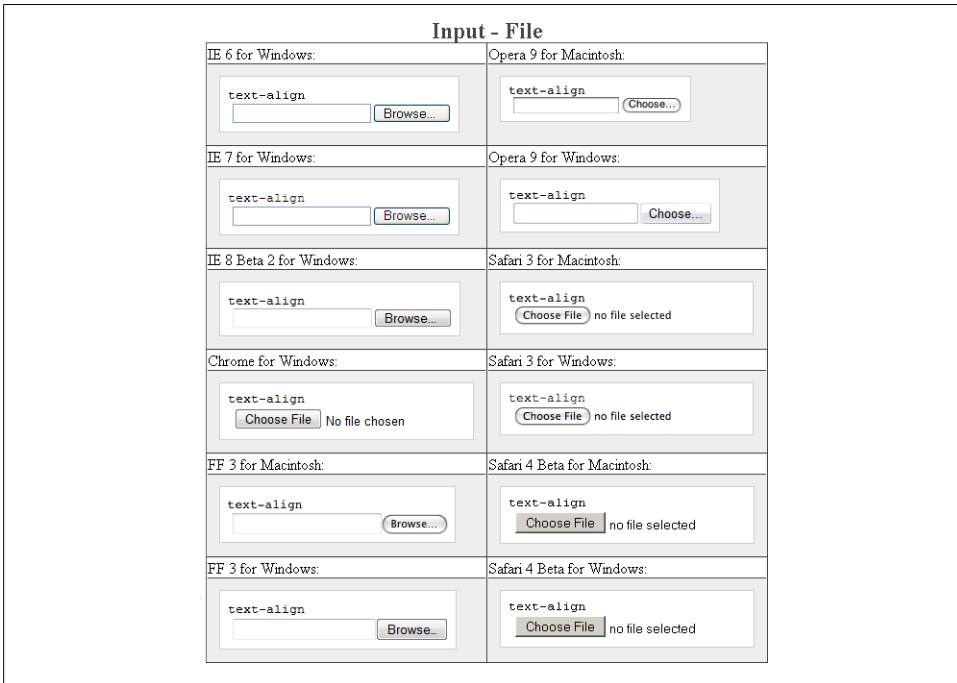
Figure E-36. Testing the alignment of text on file input

text-decoration
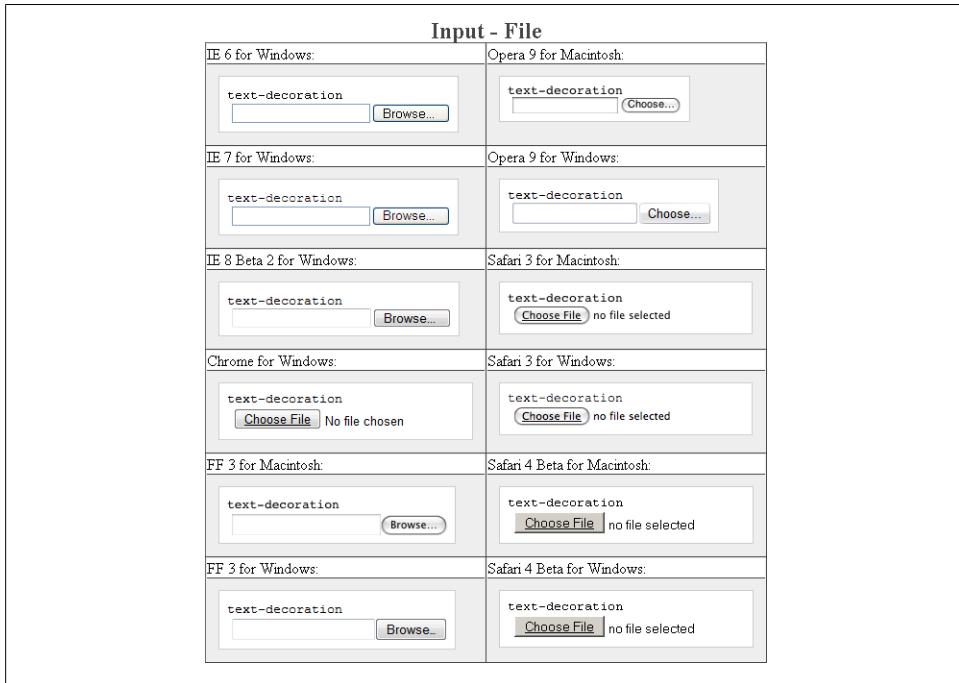
Figure E-37. Testing setting a different font on file input

text-indent
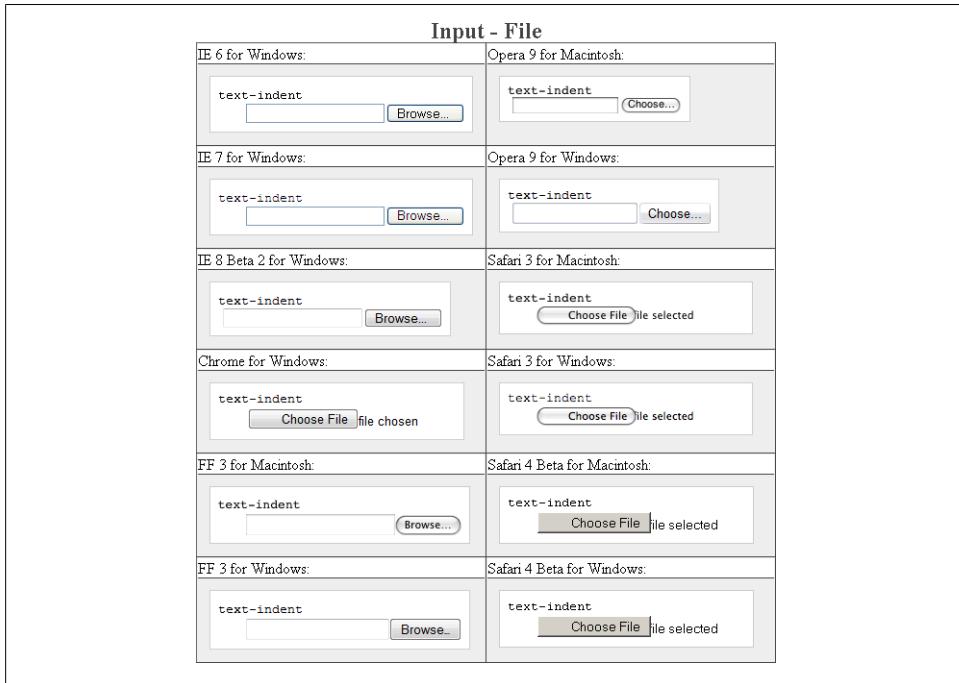
Figure E-38. Testing indenting the text on file input

width
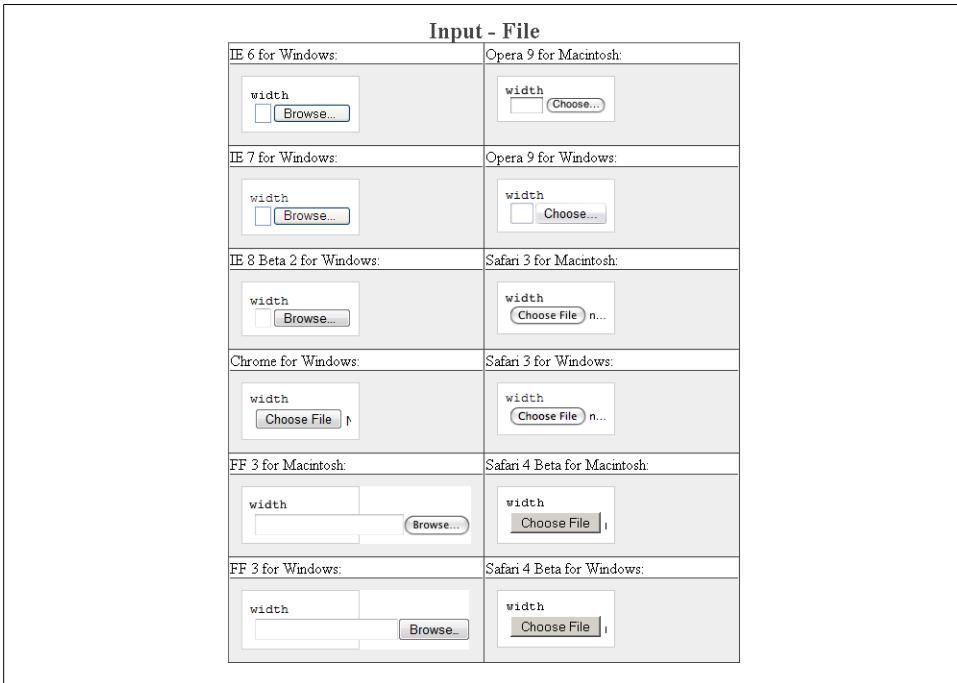
Figure E-39. Testing the width of file input

word-spacing
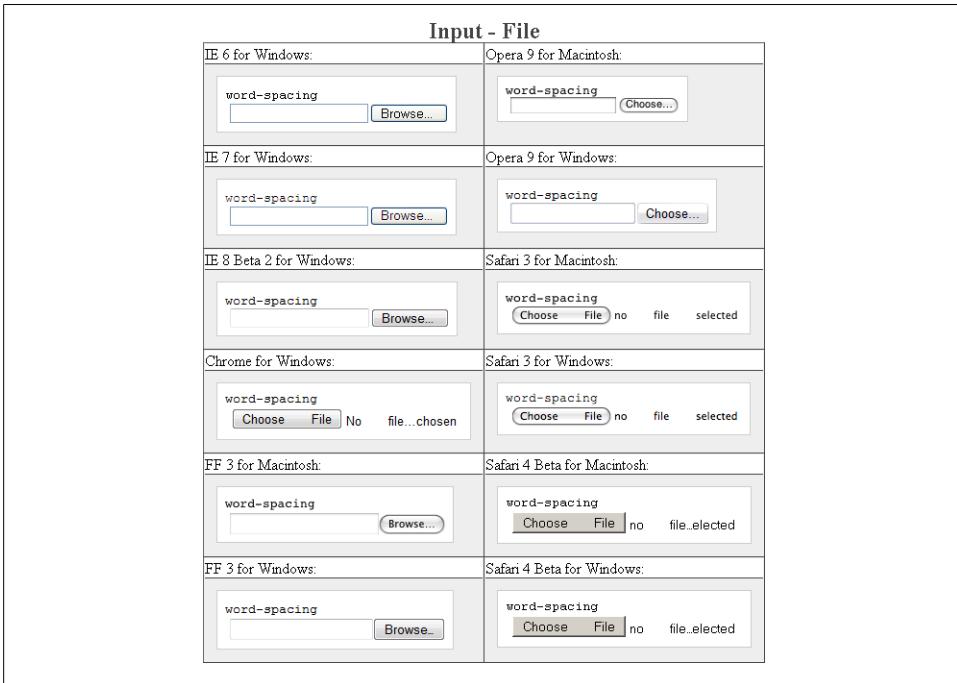
Figure E-40. Testing the spacing between words on file input

Input Element for Radio Buttons

Like checkboxes, radio buttons provide on/off options for a grouping. However, unlike checkboxes, which can take more than one value, a radio button is used when only one option out of a group is allowed to be submitted.

	WinIE6	WinIE7	WinIE8b2	Chrome	MacFF3	WinFF3	MacOP9	WinOP9	MacSF3	WinSF3	MacSF4b	WinSF4b
background-color	S	S	S	N	N	N	Y	Y	N	N	N	N
background-image	S	S	S	N	N	N	S	S	N	N	N	N
border	N	N	N	N	N	N	N	N	N	N	N	N
border-color	S	S	S	N	N	N	S	S	N	N	N	N
border-style	S	S	S	N	N	N	Y	Y	N	N	N	N
border-width	N	N	N	N	N	N	N	N	N	N	N	N
color	N	N	N	N	N	N	N	N	N	N	N	N
font-family	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA
font-size	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA
font-weight	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA
height	S	S	S	S	N	S	S	S	S	S	S	S
letter-spacing	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA
line-height	N	N	N	N	N	N	N	N	N	N	N	N
margin	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y
padding	N	N	Y	N	N	N	Y	Y	N	N	N	S
text-align	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA
text-decoration	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA
text-indent	Y	Y	N	N	N	N	N	N	N	N	N	N
width	S	S	S	S	N	S	S	S	S	S	S	S
word-spacing	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA
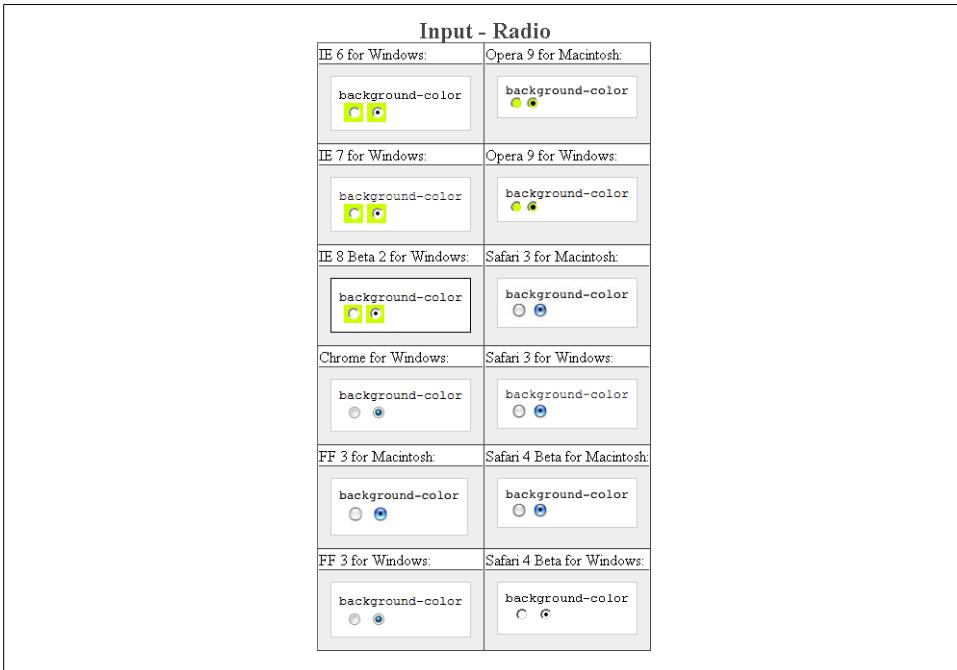
background-color

Figure E-41. Testing the background color of radio buttons

background-image
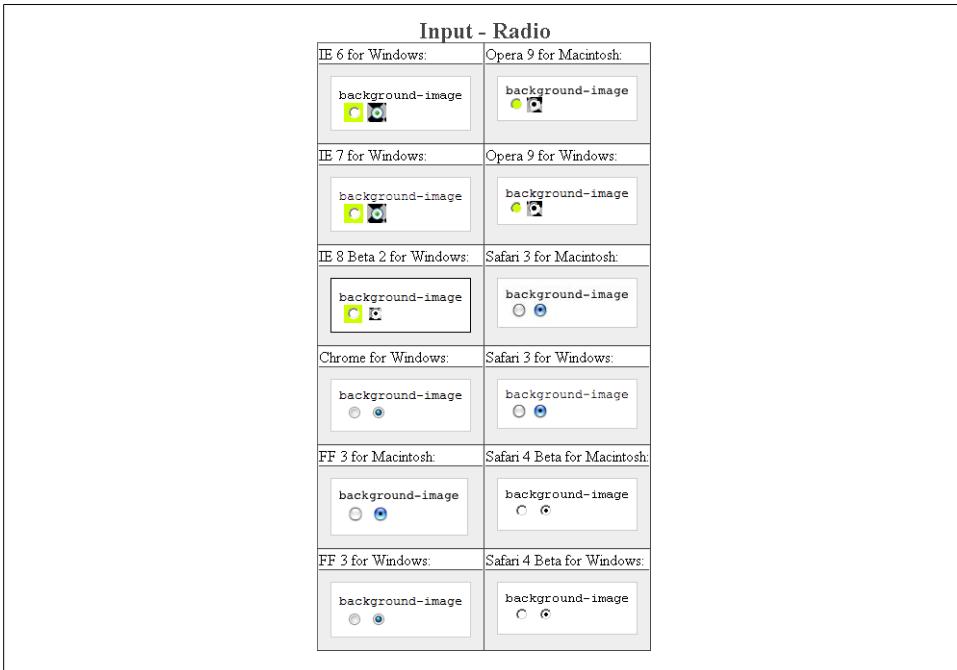
Figure E-42. Testing background images in radio buttons

border: 0;

Figure E-43. Testing the removal of borders on radio buttons

border-color
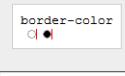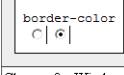
Figure E-44. Testing colors on the radio button borders

border-style
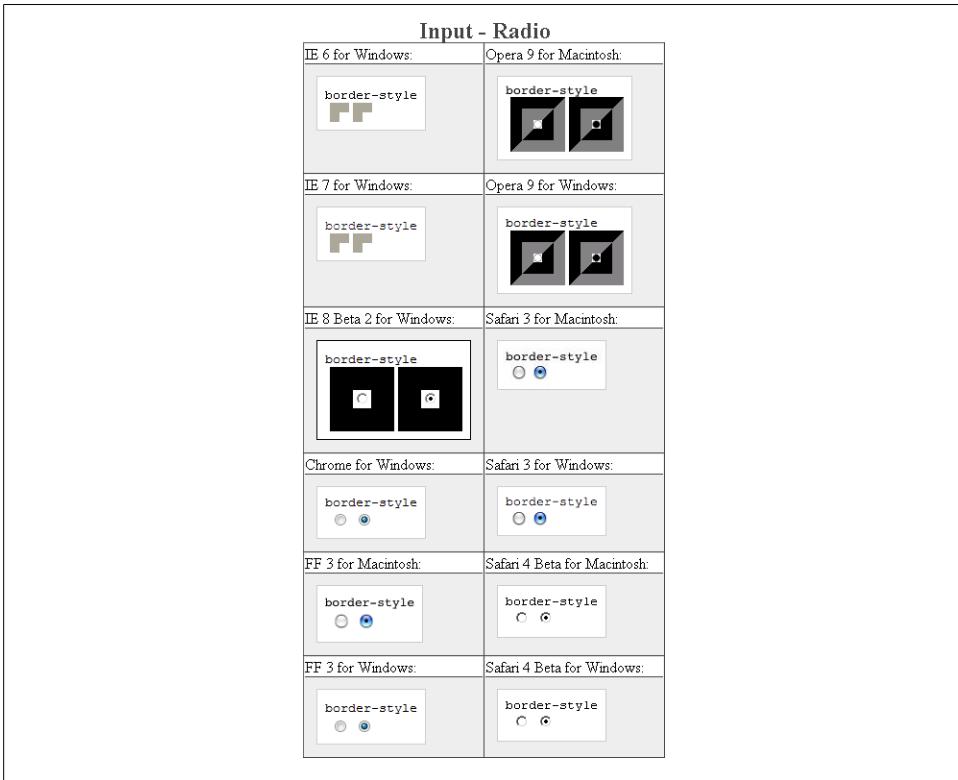
Figure E-45. Testing the styles of borders on radio buttons

border-width
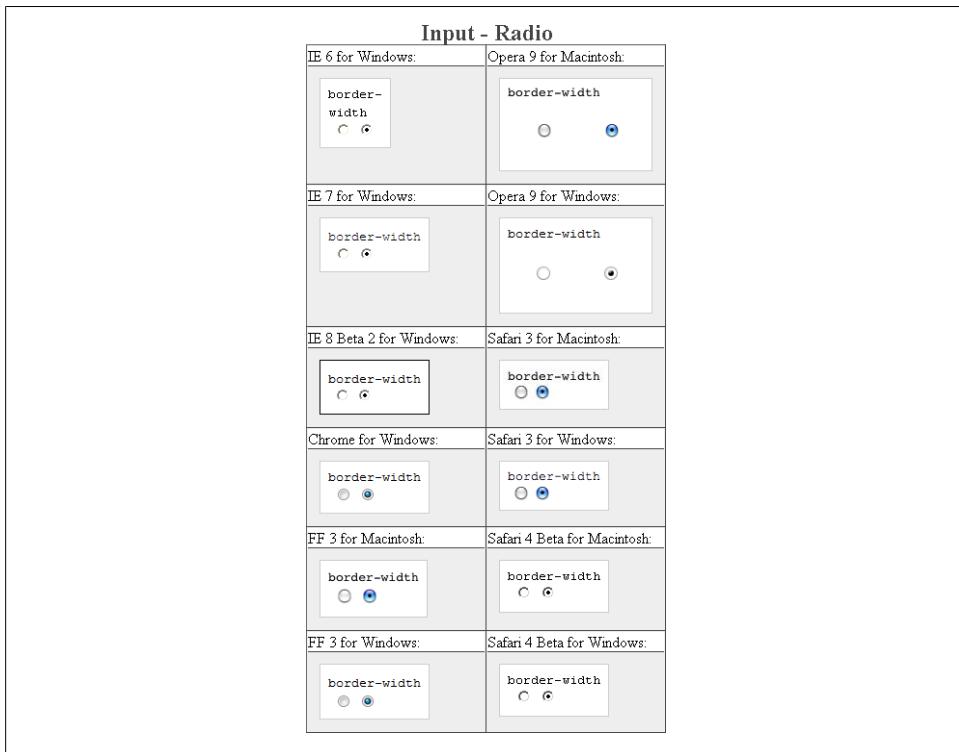
Figure E-46. Testing the widths of borders on radio buttons

color

Figure E-47. Testing the color on radio buttons

font-family
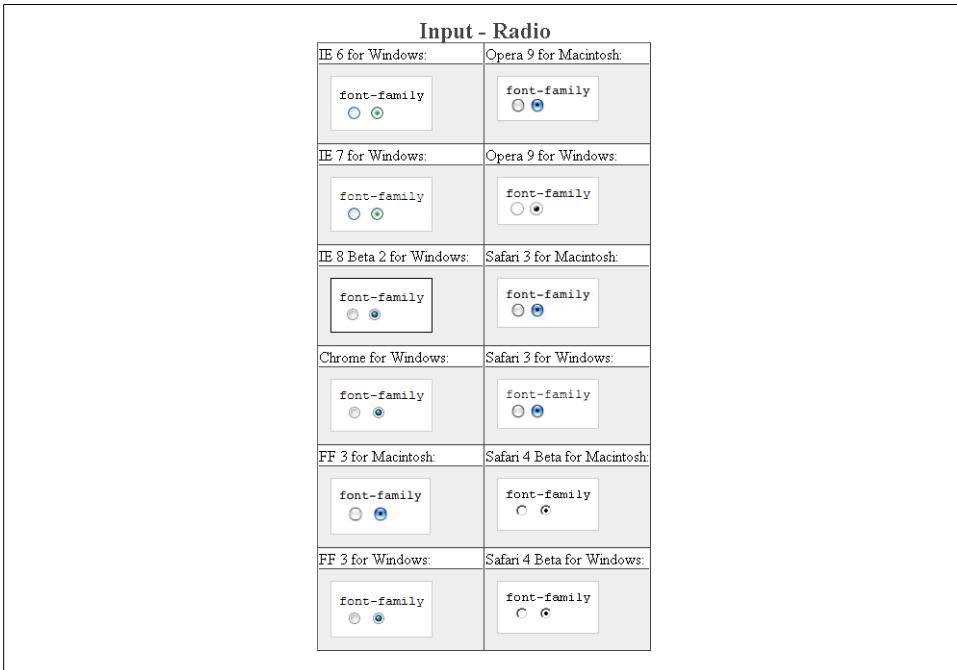
Figure E-48. Testing setting a different font on radio buttons

font-size
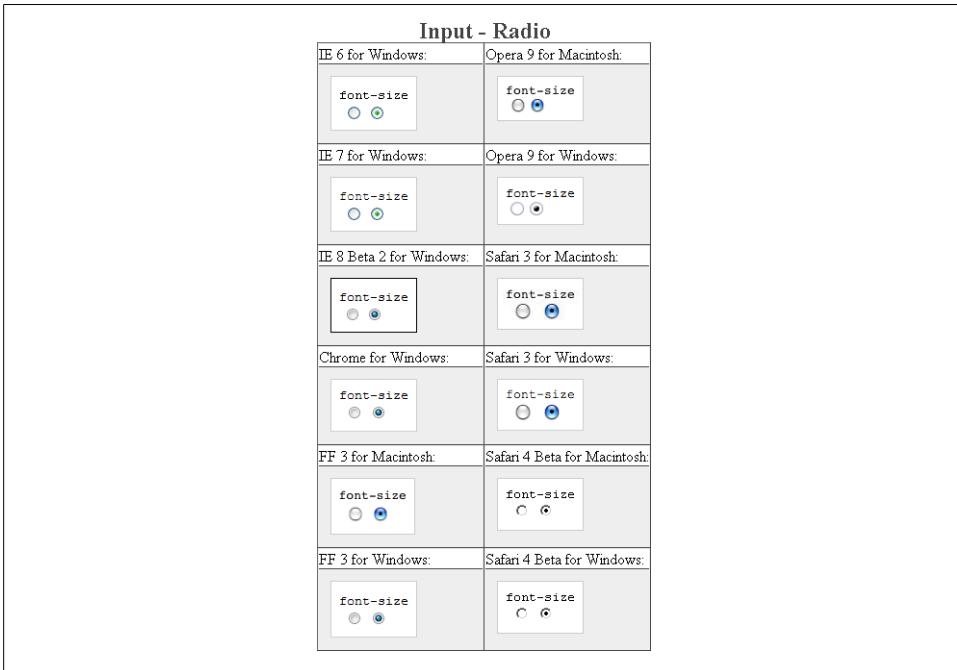
Figure E-49. Testing a different size of font on radio buttons

font-weight
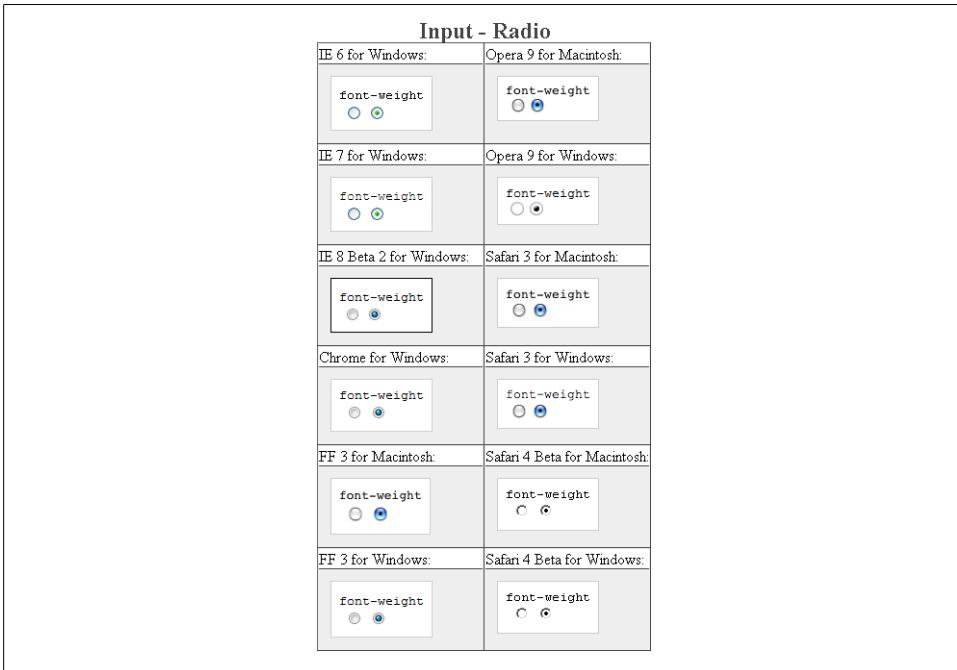
Figure E-50. Testing a bold font on radio buttons

height

Figure E-51. Testing setting a height for radio buttons

letter-spacing

Figure E-52. Testing the letter spacing of radio buttons

line-height
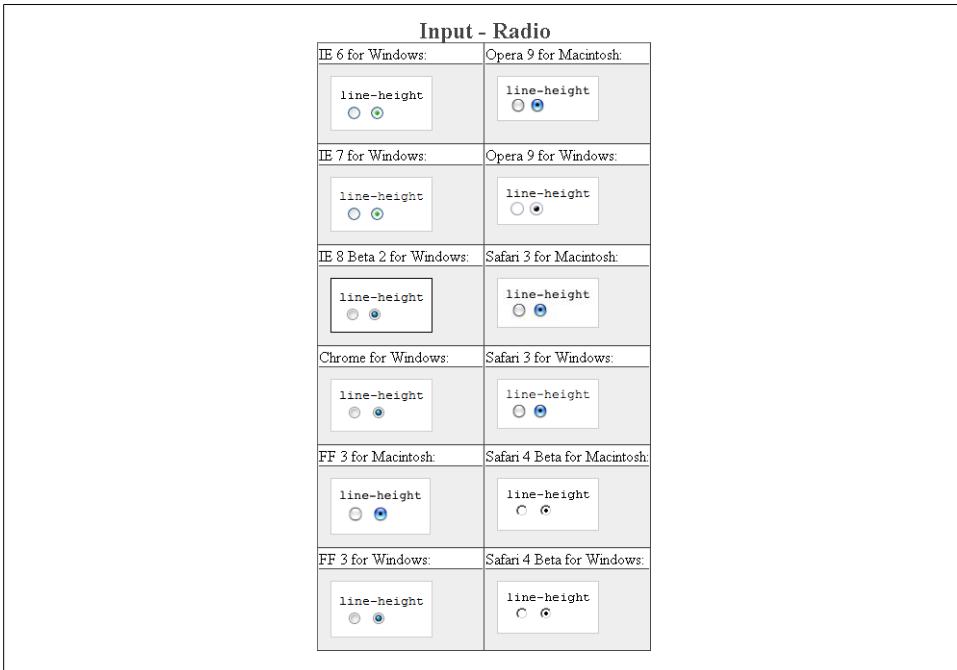
Figure E-53. Testing setting the spacing between lines of text on radio buttons

margin
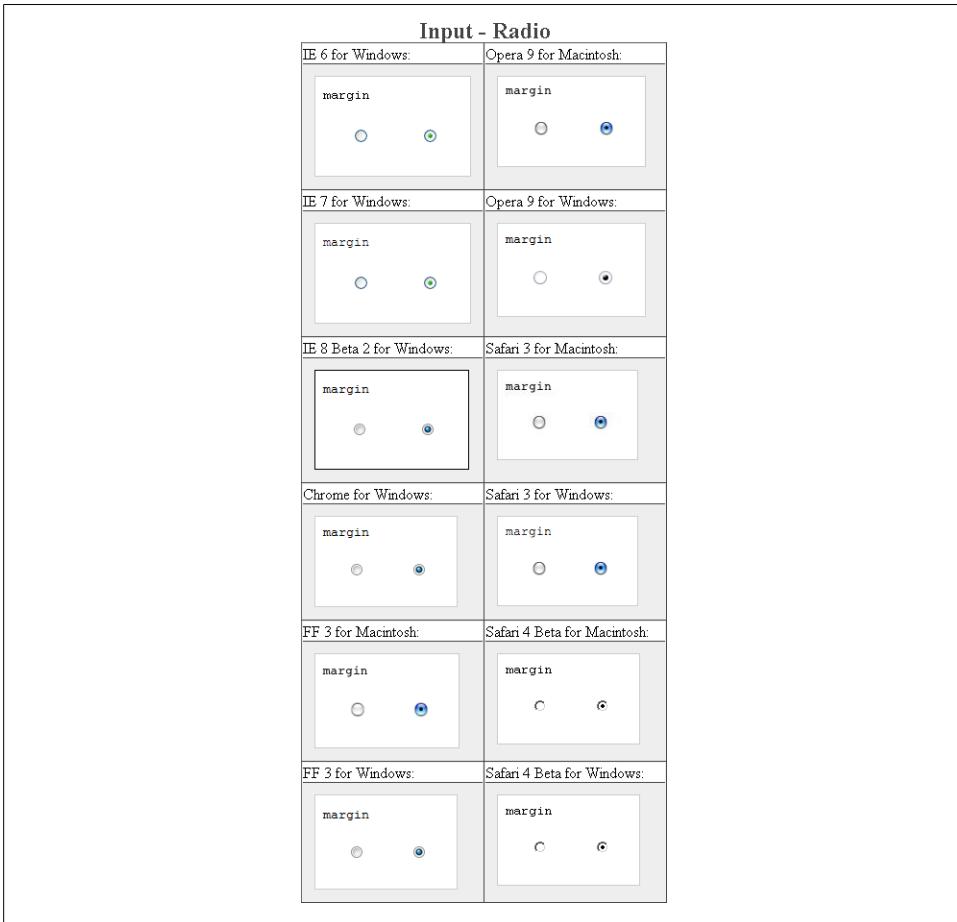
Figure E-54. Testing margins on radio buttons

padding
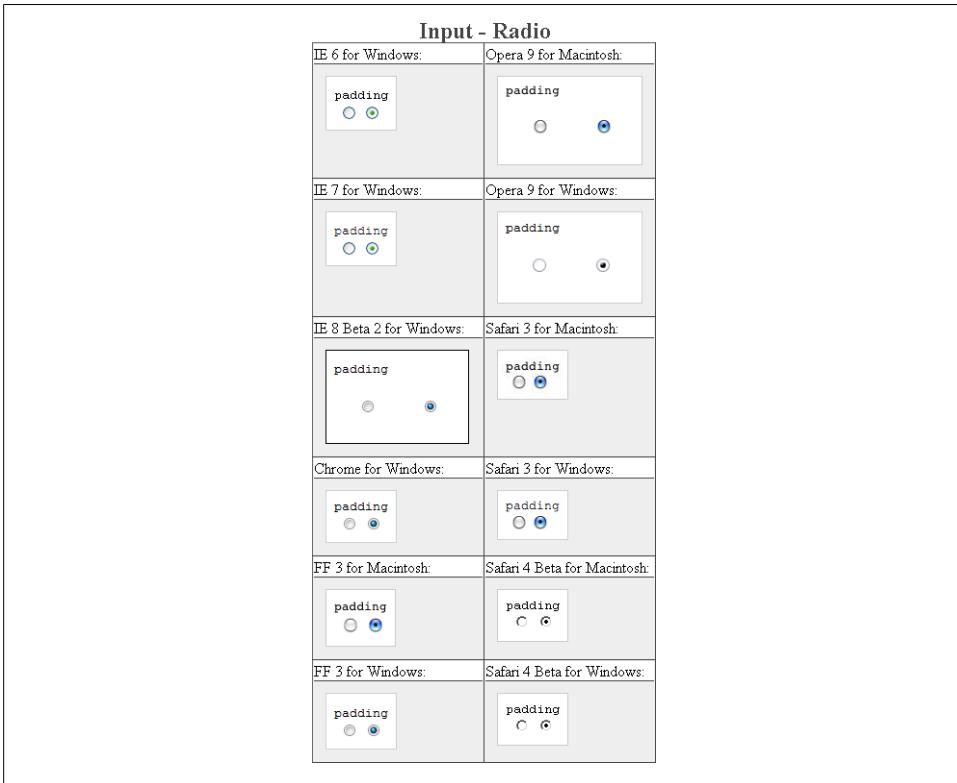
Figure E-55. Testing padding on radio buttons

text-align

Figure E-56. Testing the alignment of text on radio buttons

text-decoration
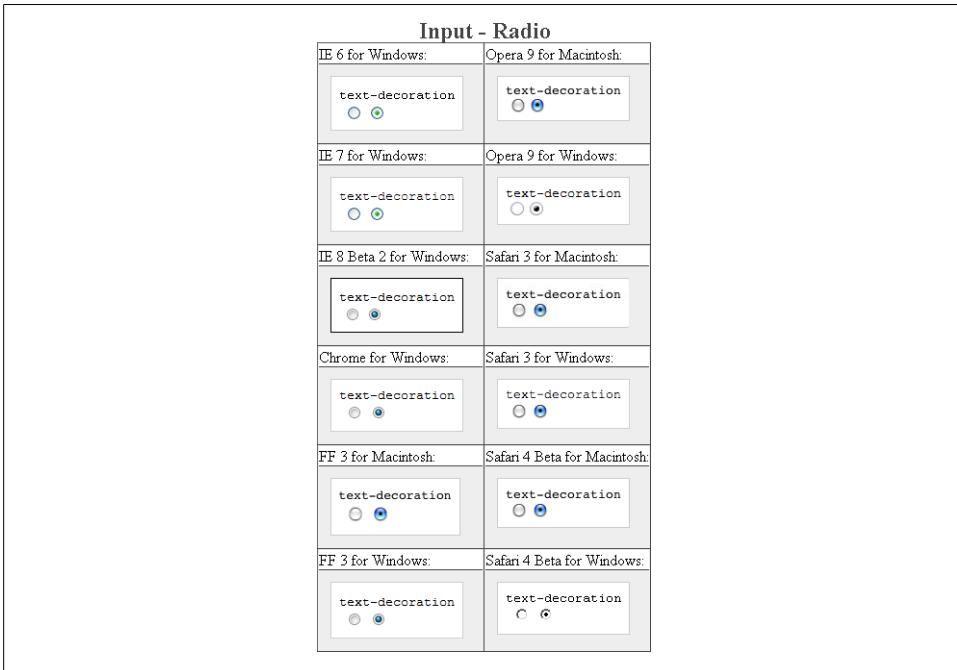
Figure E-57. Testing setting a different font on radio buttons

text-indent
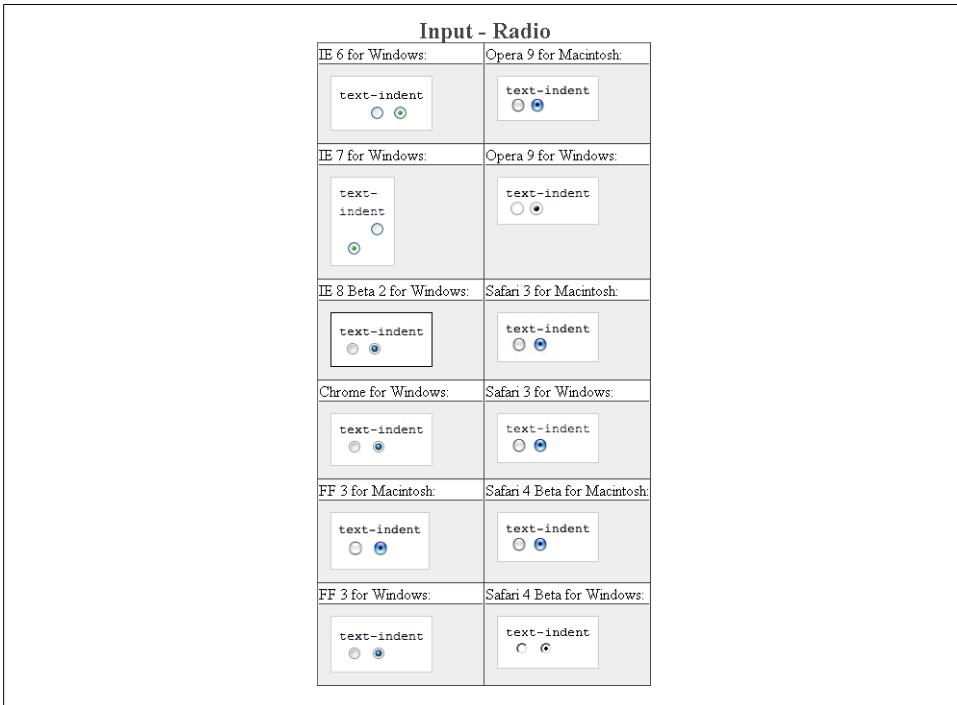
Figure E-58. Testing indenting the text on radio buttons

width

Figure E-59. Testing the width of radio buttons

word-spacing
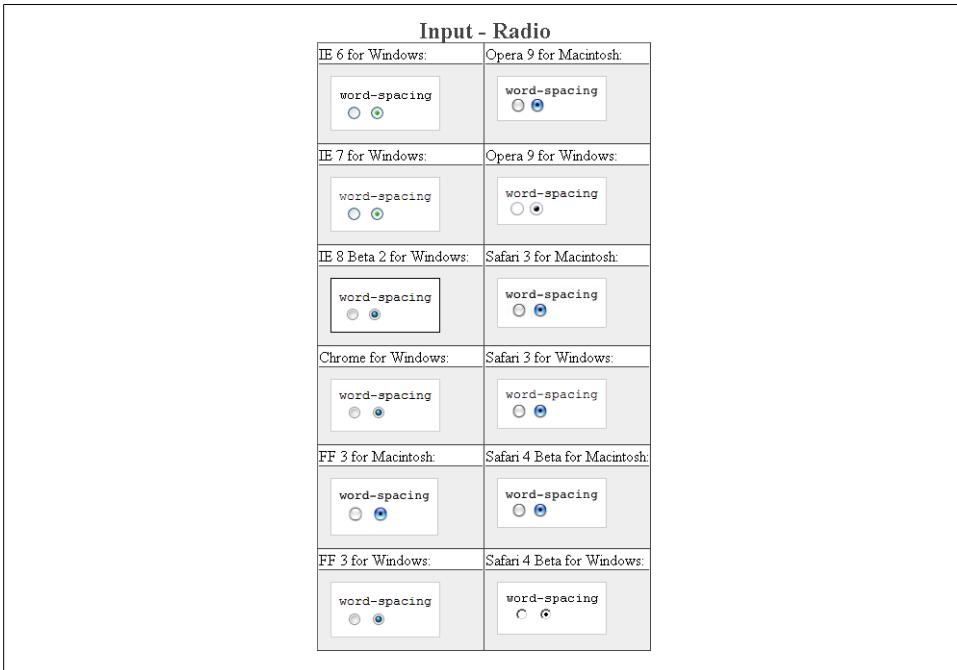
Figure E-60. Testing the spacing between words on radio buttons

Input Element for Text

The input element allows users to submit a single line of text.

Table E-5. A review of the CSS properties on text fields

	WinIE6	WinIE7	WinIE8b2	Chrome	MacFF3	WinFF3	MacOP9	WinOP9	MacSF3	WinSF3	MacSF4b	WinSF4b
`background-color`	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y
`background-image`	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y
`border`	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y
`border-color`	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y
`border-style`	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y
`border-width`	Y	Y	Y	Y	Y	Y	N	N	Y	Y	Y	Y
`color`	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y
`font-family`	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y
`font-size`	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y
`font-weight`	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y
`height`	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y
`letter-spacing`	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y
`line-height`	N	N	N	N	N	N	N	N	N	N	N	N
`margin`	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y
`padding`	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y
`text-align`	N	N	N	N	N	N	N	N	N	N	N	N
`text-decoration`	Y	Y	Y	Y	Y	Y	N	N	Y	Y	Y	Y
`text-indent`	S	S	S	Y	Y	Y	Y	Y	Y	Y	Y	Y
`width`	Y	Y	Y	Y	Y	Y	Y	N	Y	Y	Y	Y
`word-spacing`	Y	Y	Y	Y	Y	Y	N	N	Y	Y	Y	Y

background-color

Figure E-61. Testing the background color of text fields

background-image
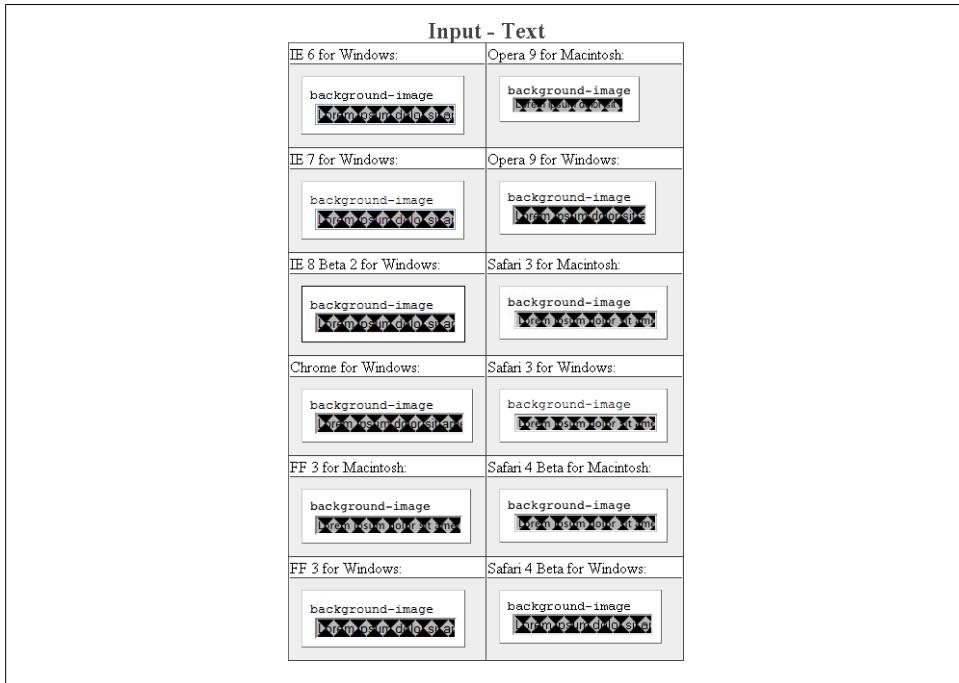
Figure E-62. Testing background images in text fields

border: 0;

Figure E-63. Testing the removal of borders on text fields

border-color
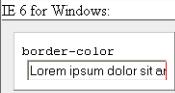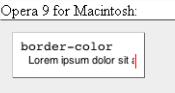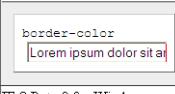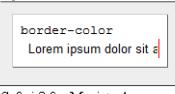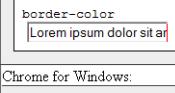
Figure E-64. Testing colors on the text field borders

border-style

Figure E-65. Testing the styles of borders on text fields

border-width
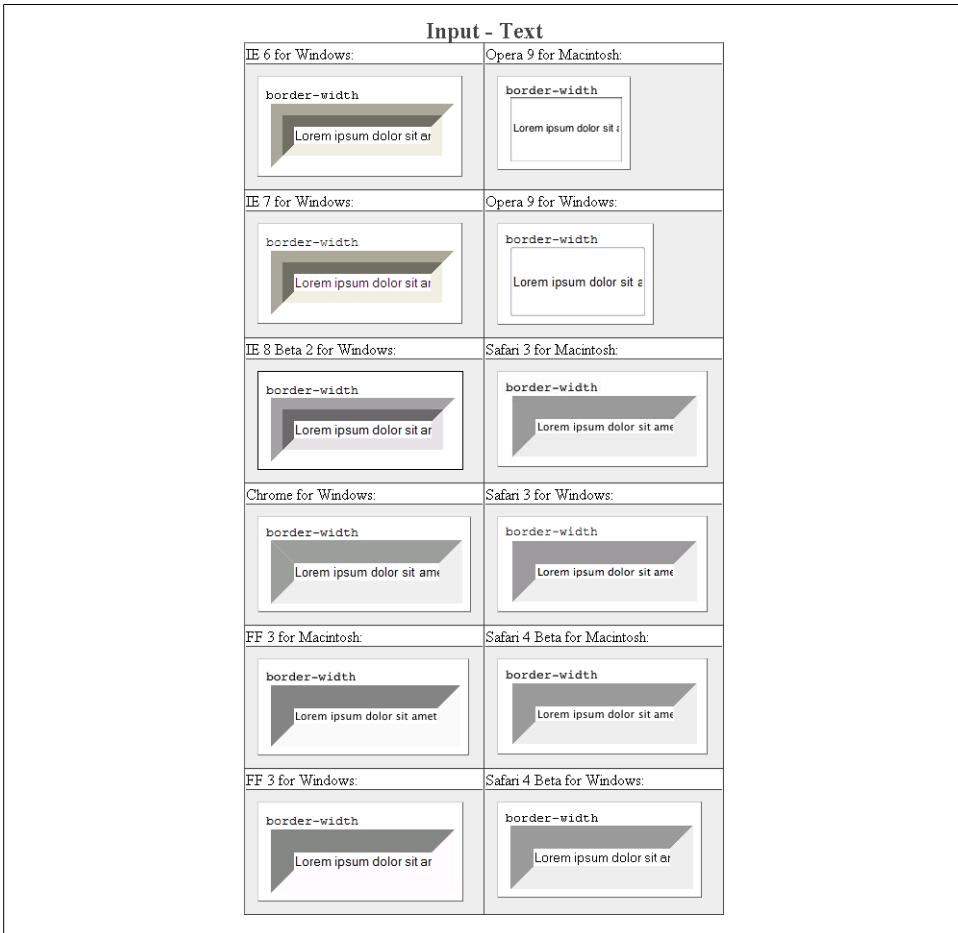
Figure E-66. Testing the widths of borders on text fields

color
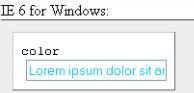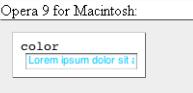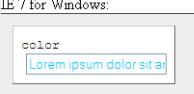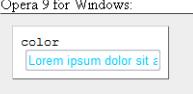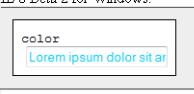
Figure E-67. Testing the color on text fields

font-family
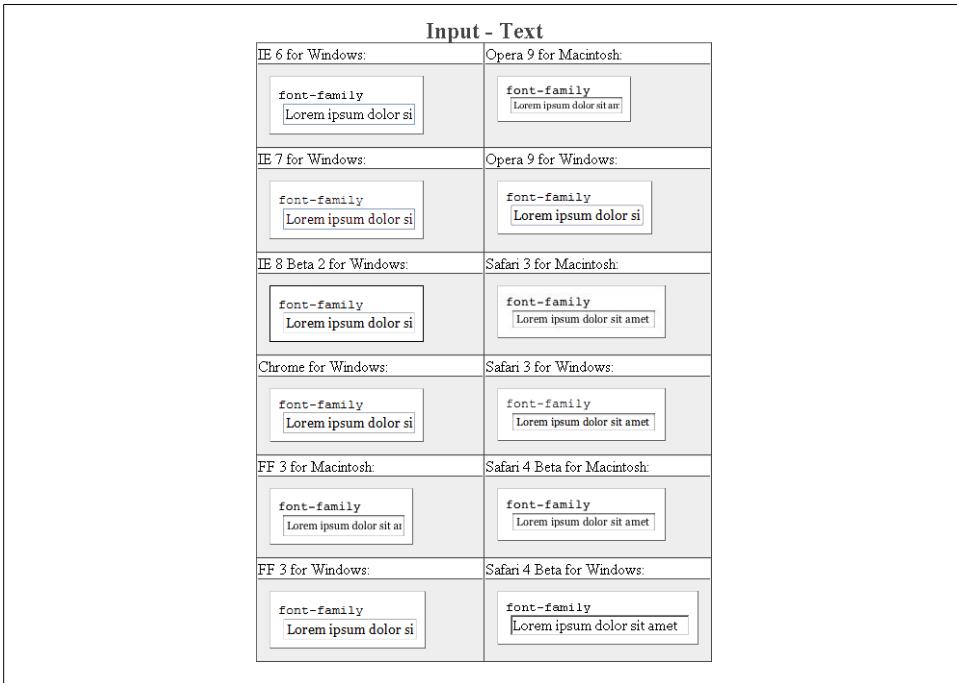
Figure E-68. Testing setting a different font on text fields

font-size
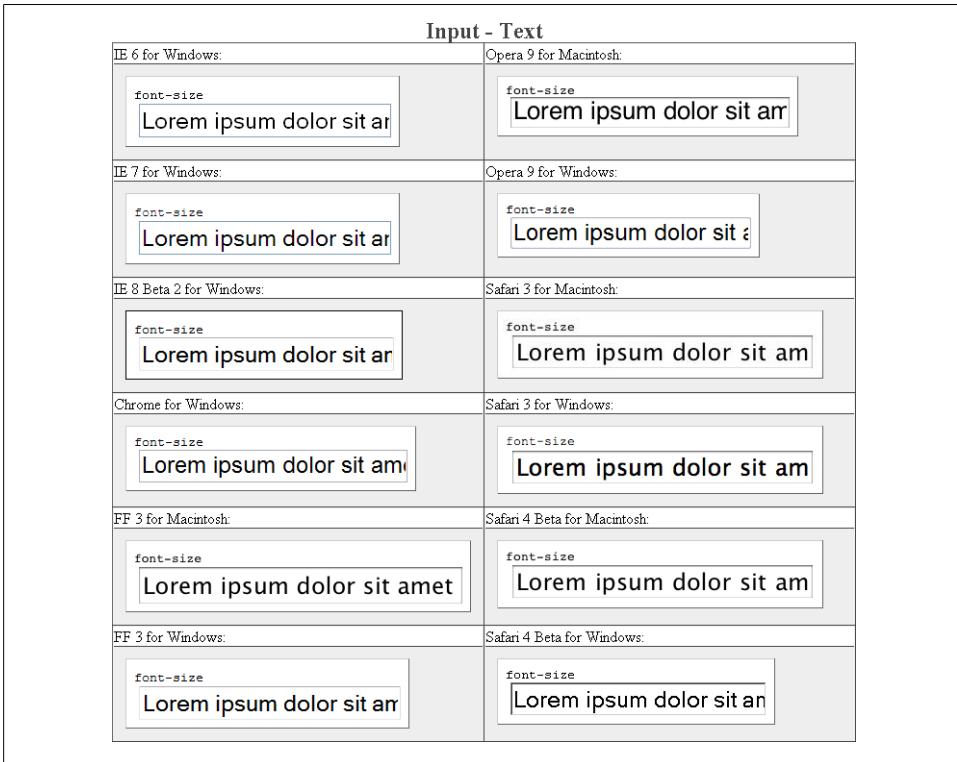
Figure E-69. Testing a different size of font on text fields

font-weight
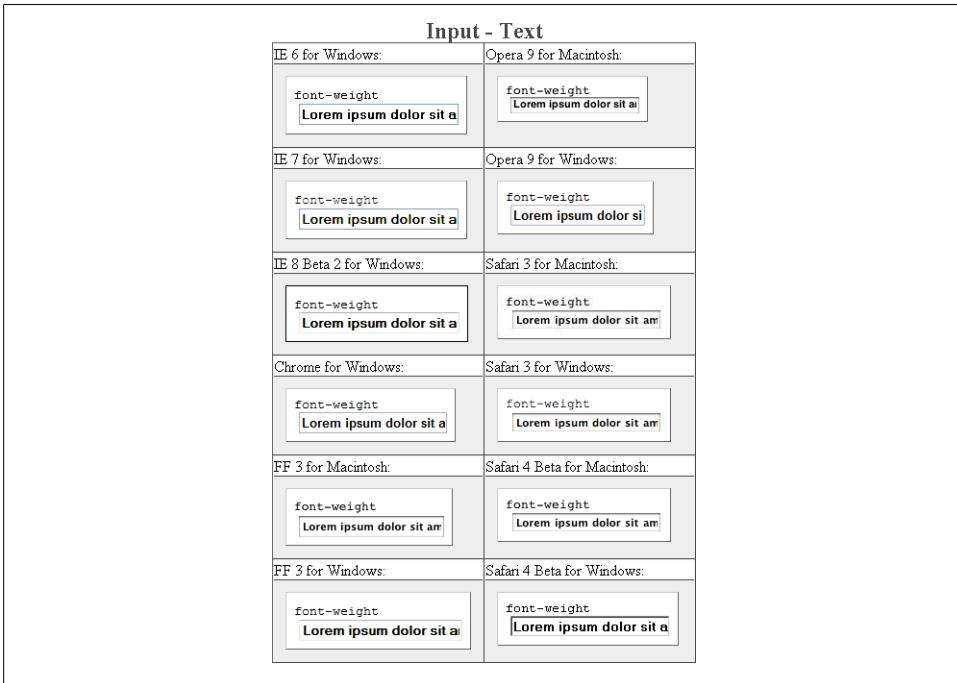
Figure E-70. Testing a bold font on text fields

height
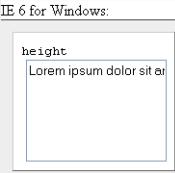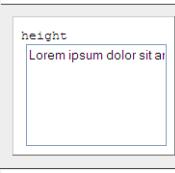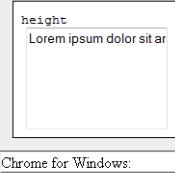
Figure E-71. Testing setting a height for text fields

letter-spacing
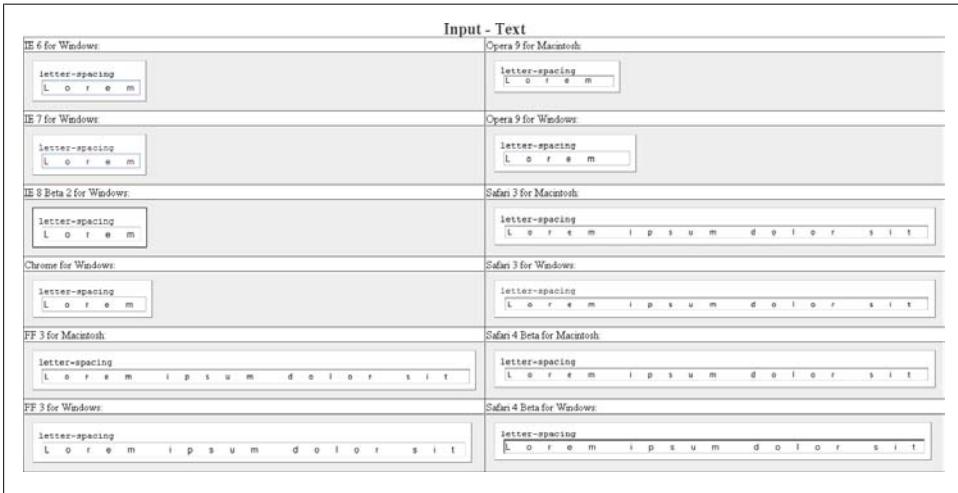
Figure E-72. Testing the letter spacing of text fields

line-height
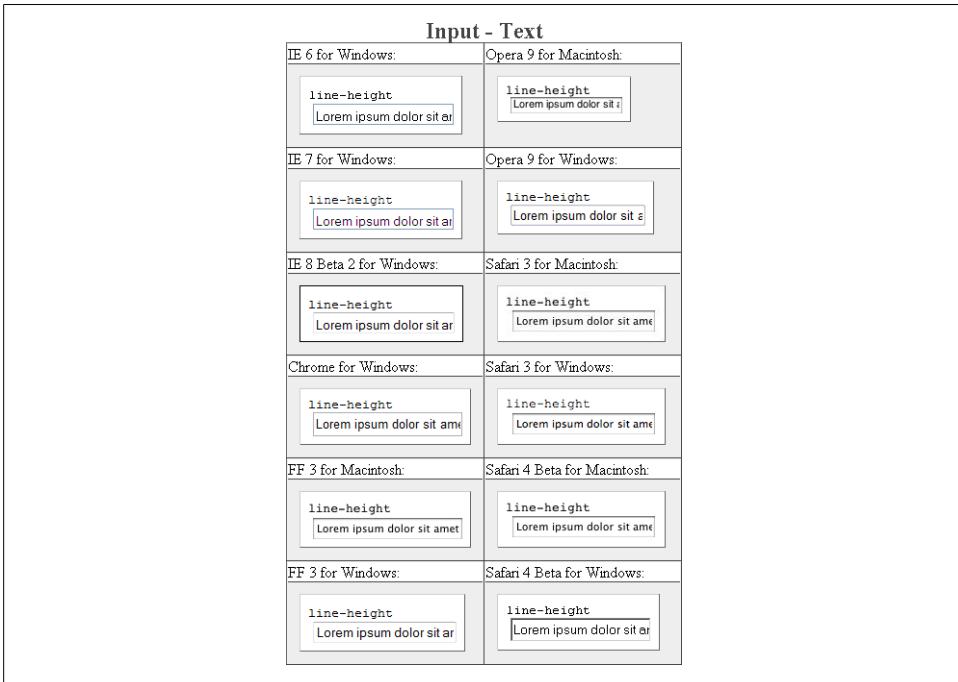
Figure E-73. Testing setting the spacing between lines of text on text fields

margin
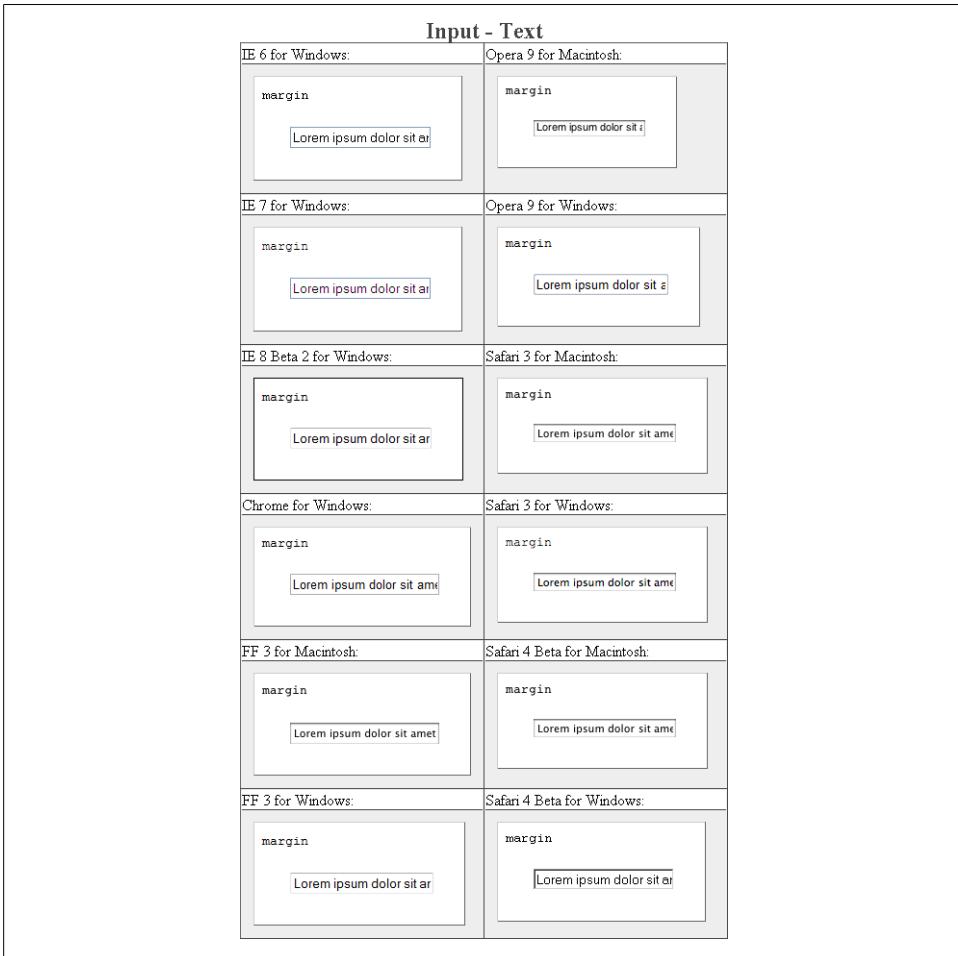
Figure E-74. Testing margins on text fields

padding
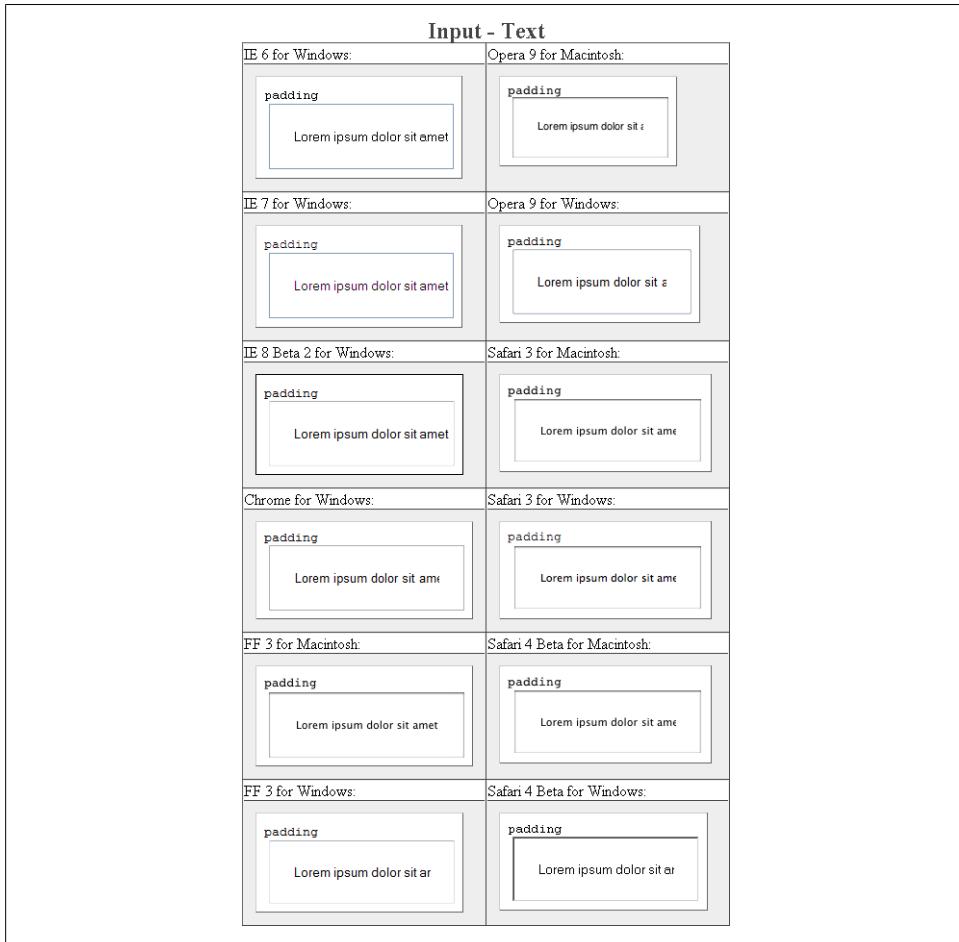
Figure E-75. Testing padding on text fields

text-align
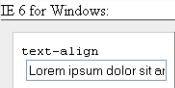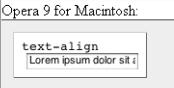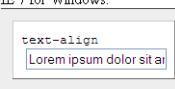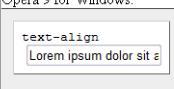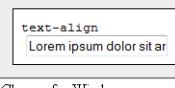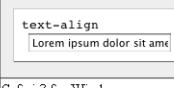
Figure E-76. Testing the alignment of text on text fields

text-decoration

Figure E-77. Testing setting a different font on text fields

text-indent
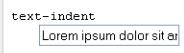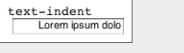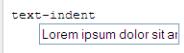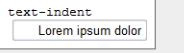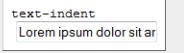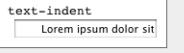
Figure E-78. Testing indenting the text on text fields

width
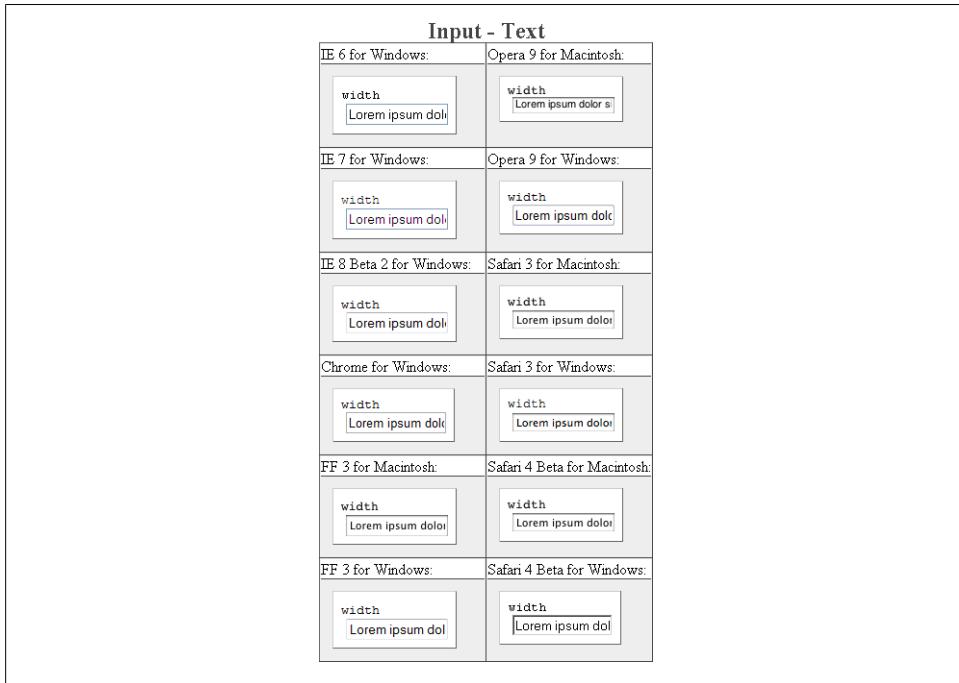
Figure E-79. Testing the width of text fields

word-spacing

Figure E-80. Testing the spacing between words on text fields

Select Element for Multiple Options

The select element allows users to select multiple options presented in a scrollable, vertical listing.

Table E-6. A review of the CSS properties on the select element with multiple options showing

	WinIE6	WinIE7	WinIE8b2	Chrome	MacFF3	WinFF3	MacOP9	WinOP9	MacSF3	WinSF3	MacSF4b	WinSF4b
`background-color`	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y
`background-image`	N	N	Y	Y	Y	Y	N	N	Y	Y	Y	Y
`border`	N	N	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y
`border-color`	N	N	N	Y	Y	Y	Y	Y	Y	Y	Y	Y
`border-style`	N	N	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y
`border-width`	N	N	Y	Y	Y	Y	N	N	Y	Y	Y	Y
`color`	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y
`font-family`	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y
`font-size`	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y
`font-weight`	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y
`height`	S	S	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y
`letter-spacing`	N	N	N	Y	Y	Y	Y	Y	Y	Y	Y	Y
`line-height`	N	N	N	N	N	N	S	S	N	N	N	N
`margin`	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y
`padding`	N	N	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y
`text-align`	N	N	N	N	Y	Y	Y	Y	N	N	N	N
`text-decoration`	Y	Y	Y	N	N	N	N	N	N	N	N	N
`text-indent`	S	S	N	N	N	N	Y	Y	Y	Y	N	N
`width`	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y
`word-spacing`	N	N	N	Y	Y	Y	N	N	Y	Y	Y	Y

background-color

Figure E-81. Testing the background color of a select element with multiple options

background-image
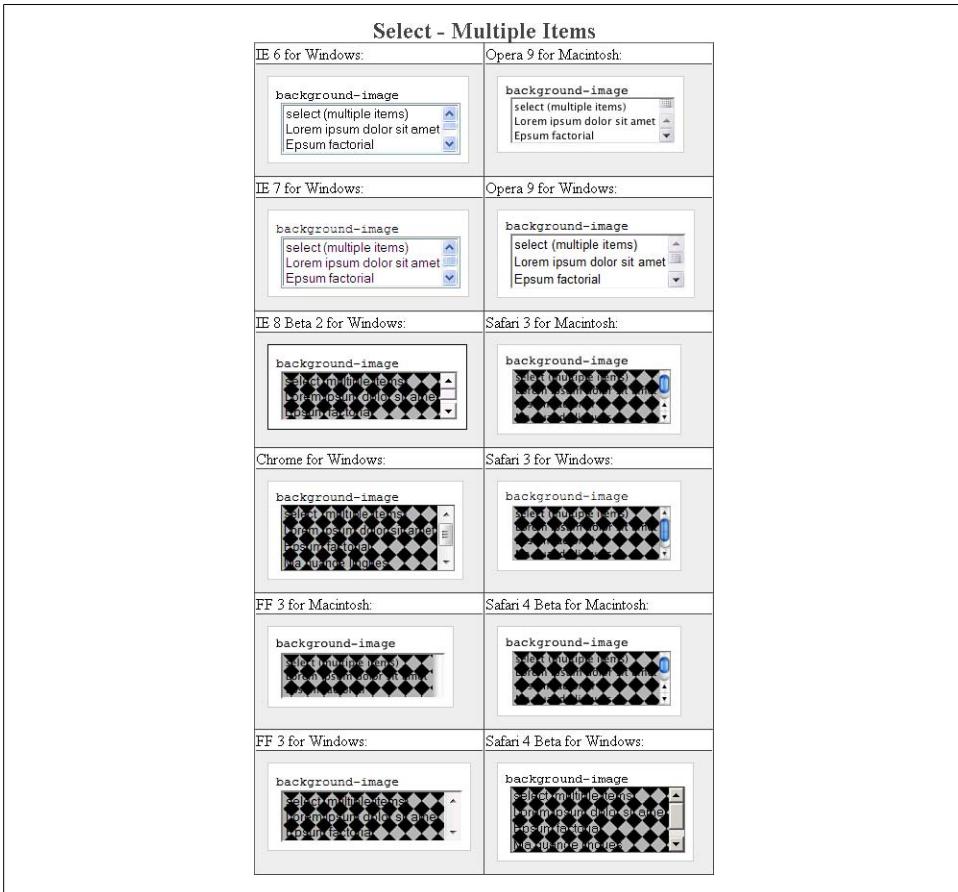
Figure E-82. Testing background images in a select element with multiple options

border: 0;
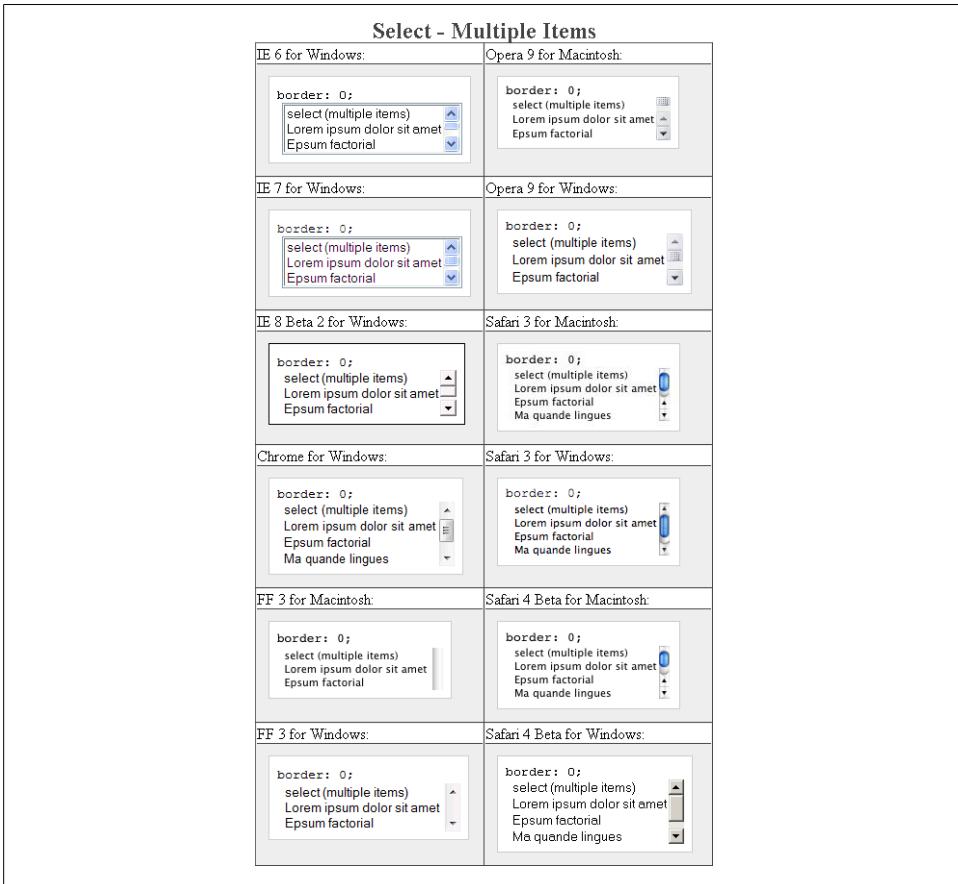
Figure E-83. Testing the removal of borders on a select element with multiple options

border-color
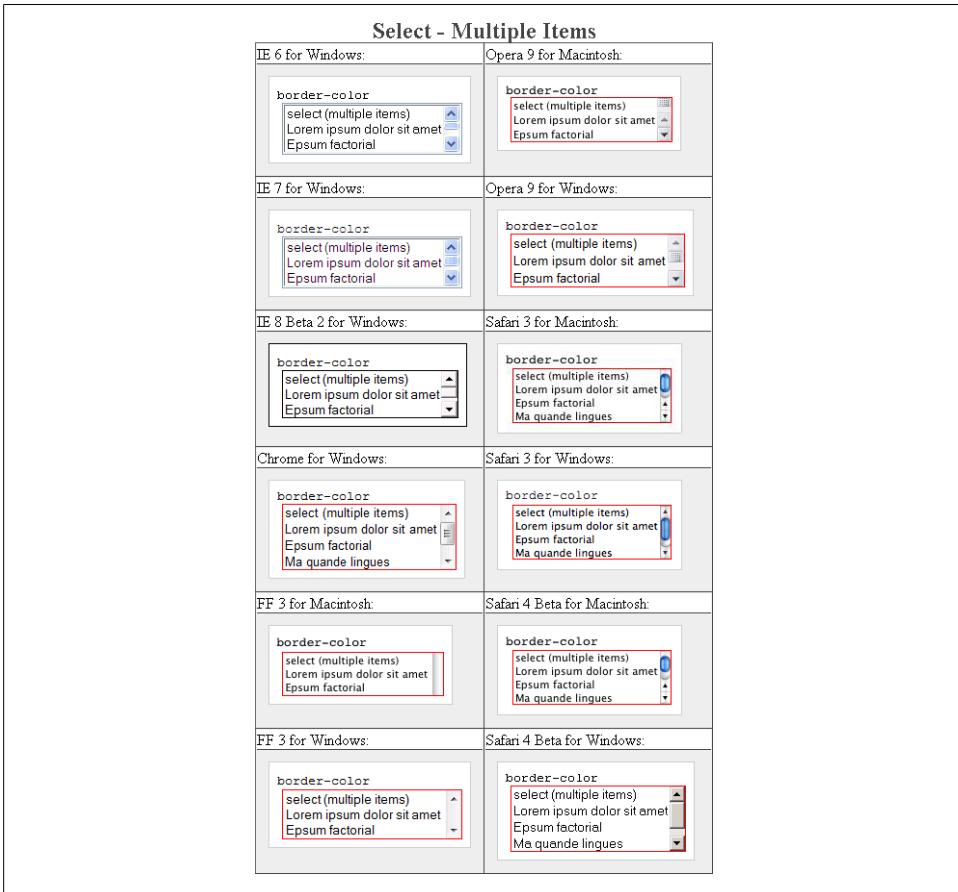
Figure E-84. Testing of border colors on a select element with multiple options

border-style

Figure E-85. Testing the styles of borders on a select element with multiple options

border-width
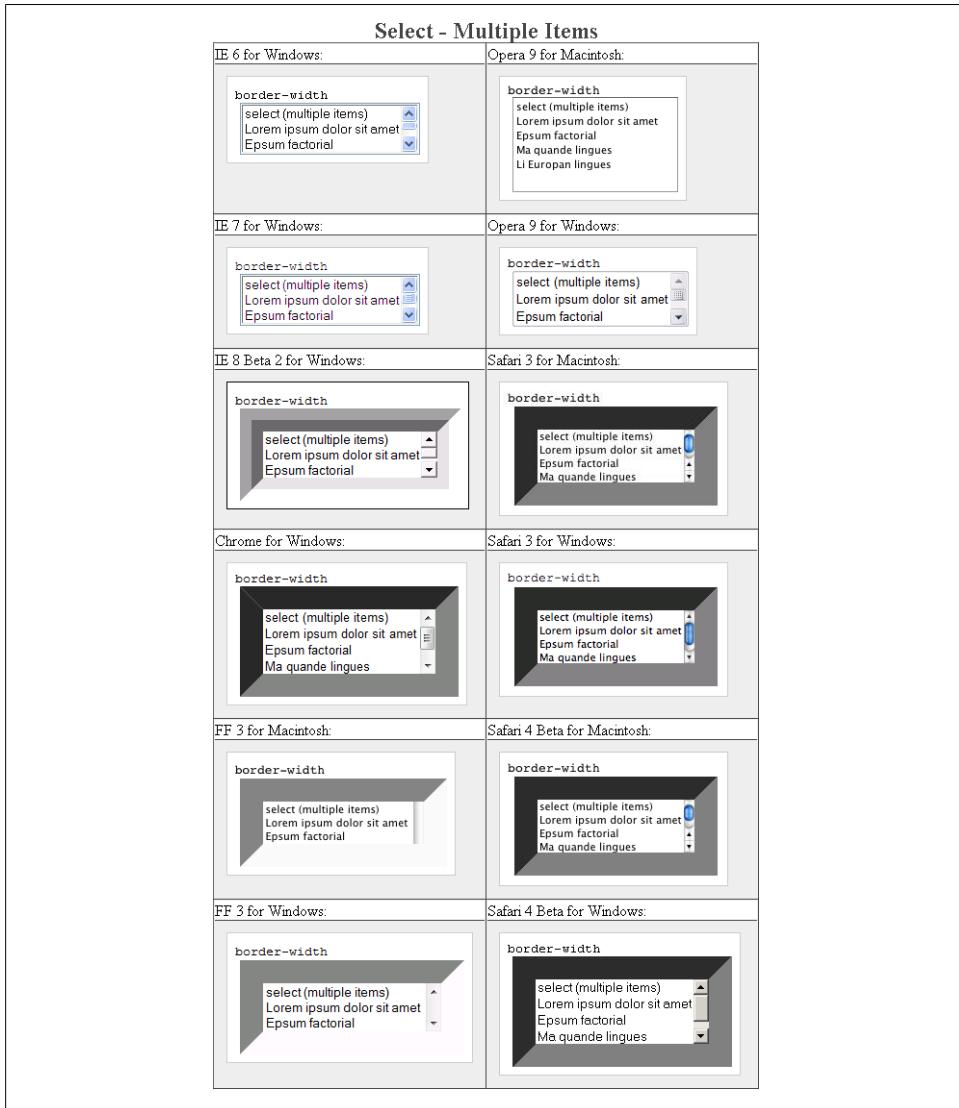
Figure E-86. Testing the widths of borders on a select element with multiple options

color
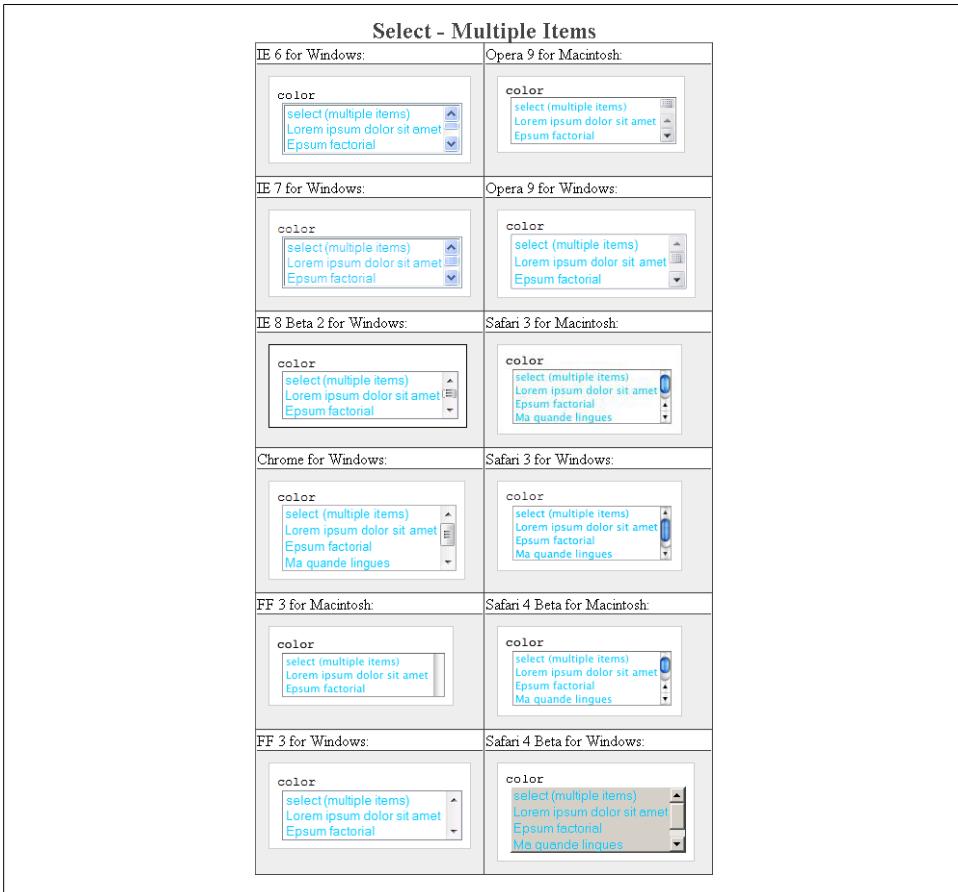
Figure E-87. Testing the color *on a select element with multiple options*

font-family
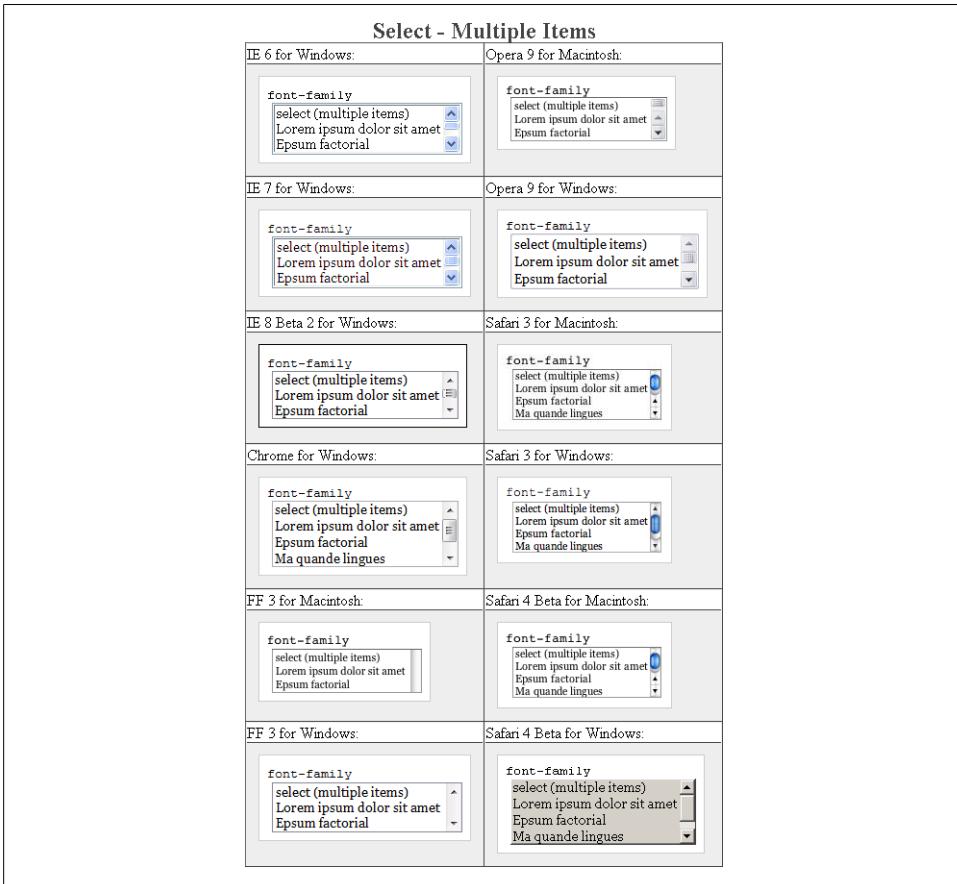
Figure E-88. Testing setting a different font on a select element with multiple options

font-size
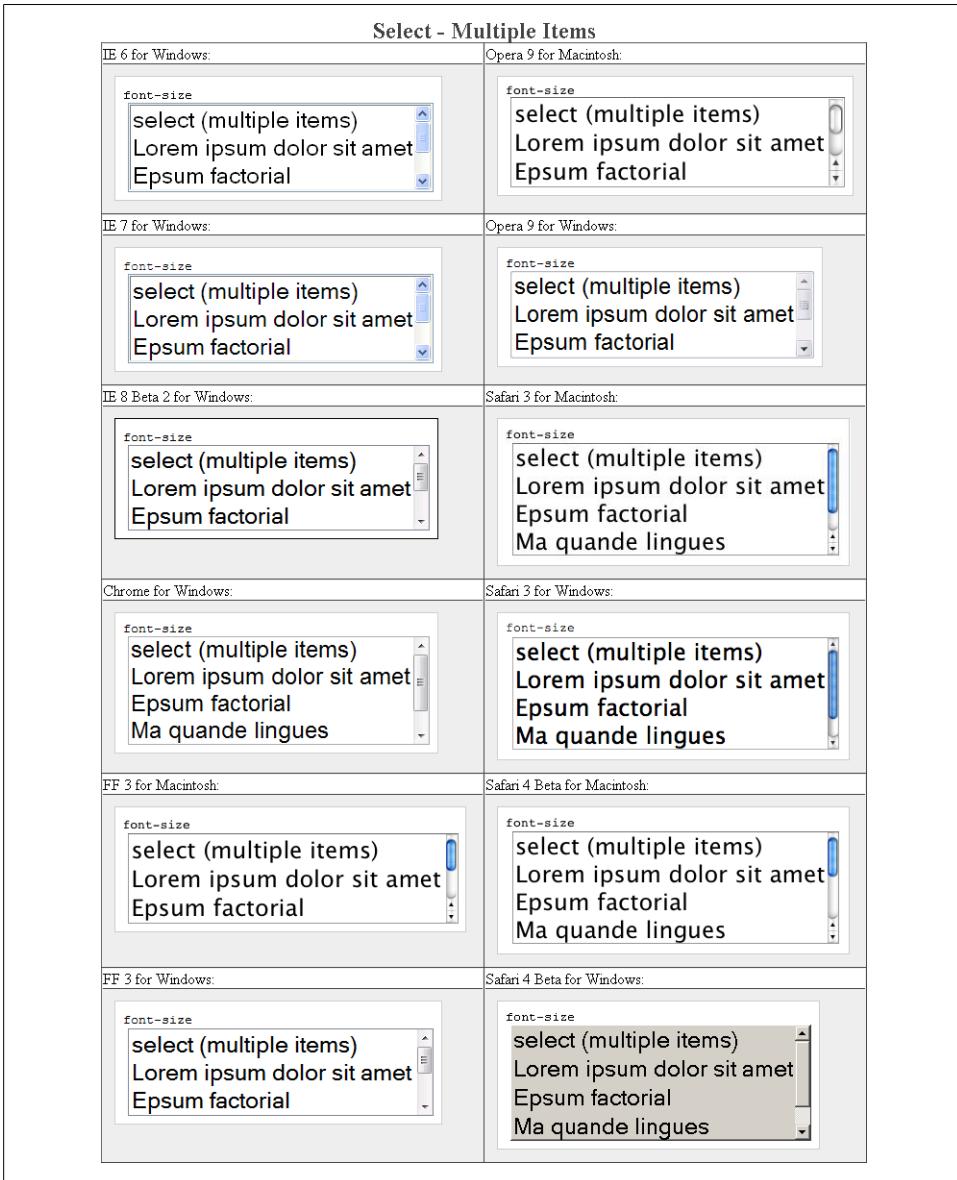
Figure E-89. Testing a different size of font on a select element with multiple options

font-weight
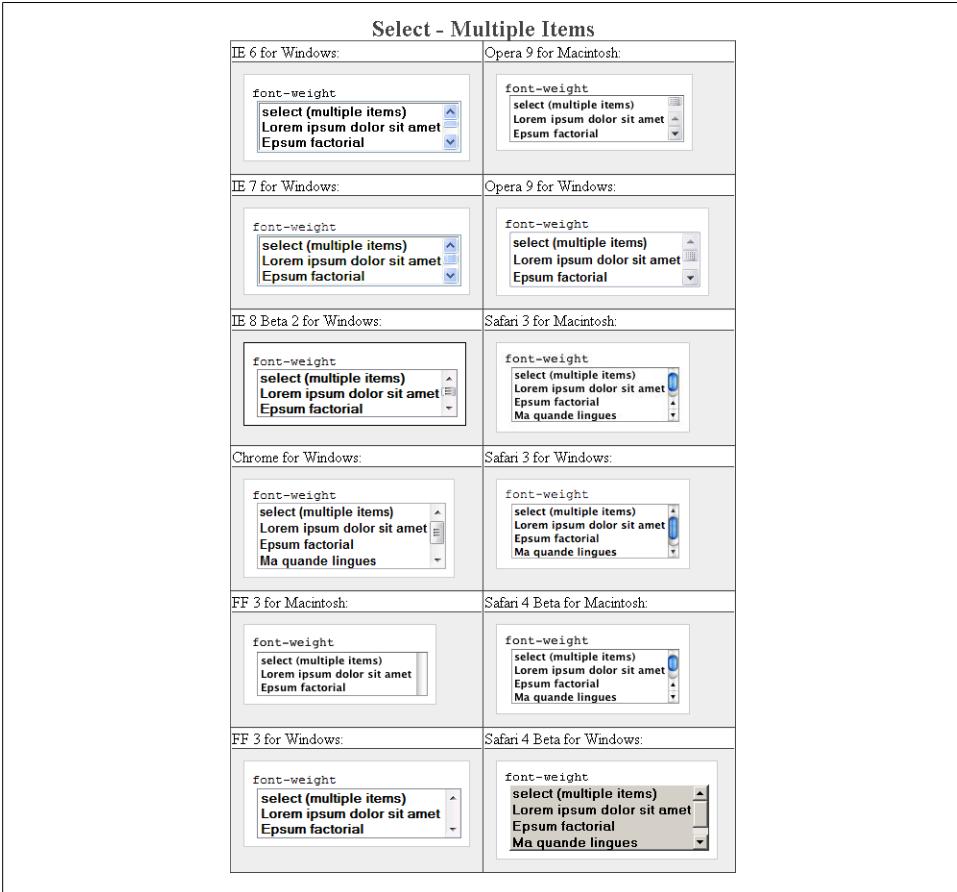
Figure E-90. Testing a bold font on a select element with multiple options

height
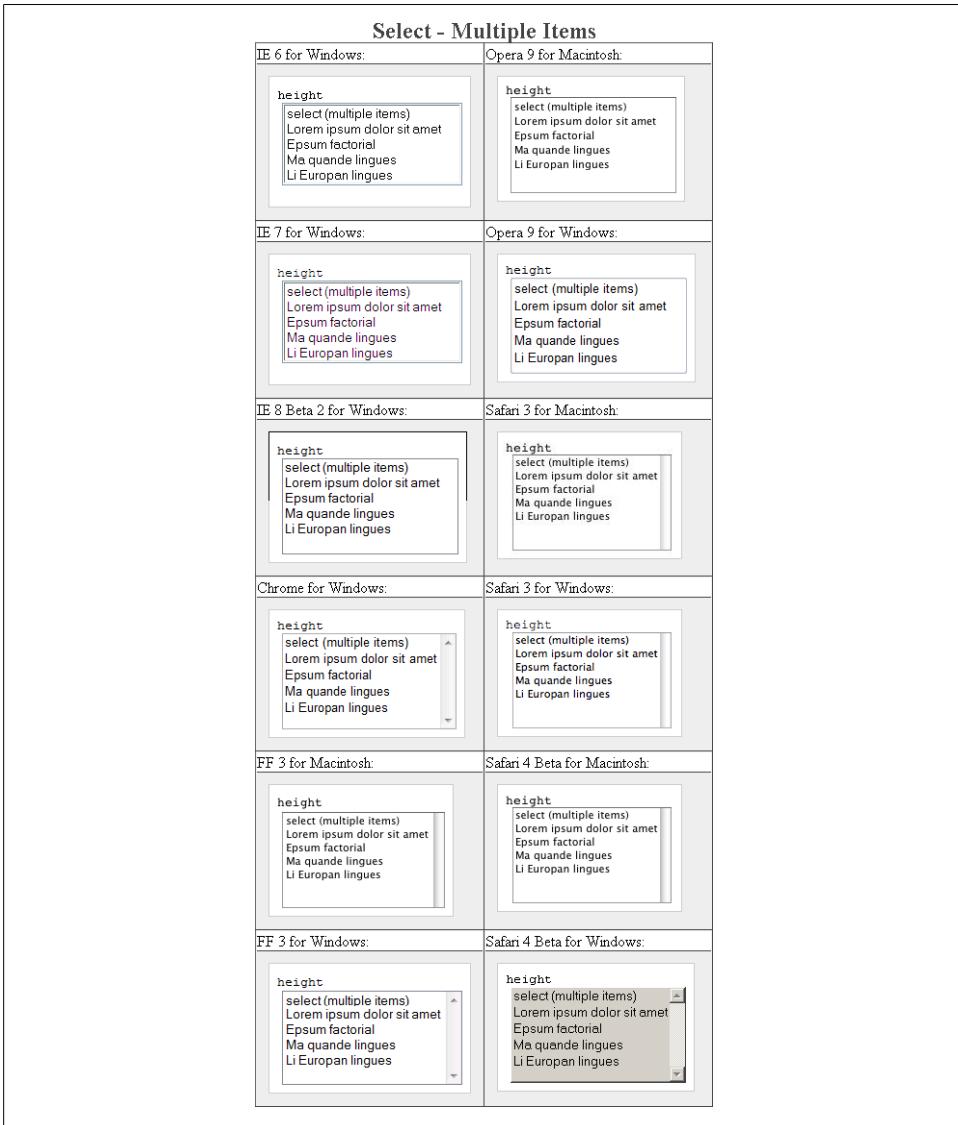
Figure E-91. Testing setting a height for a select element with multiple options

letter-spacing

Figure E-92. Testing the letter spacing of a select element with multiple options

line-height
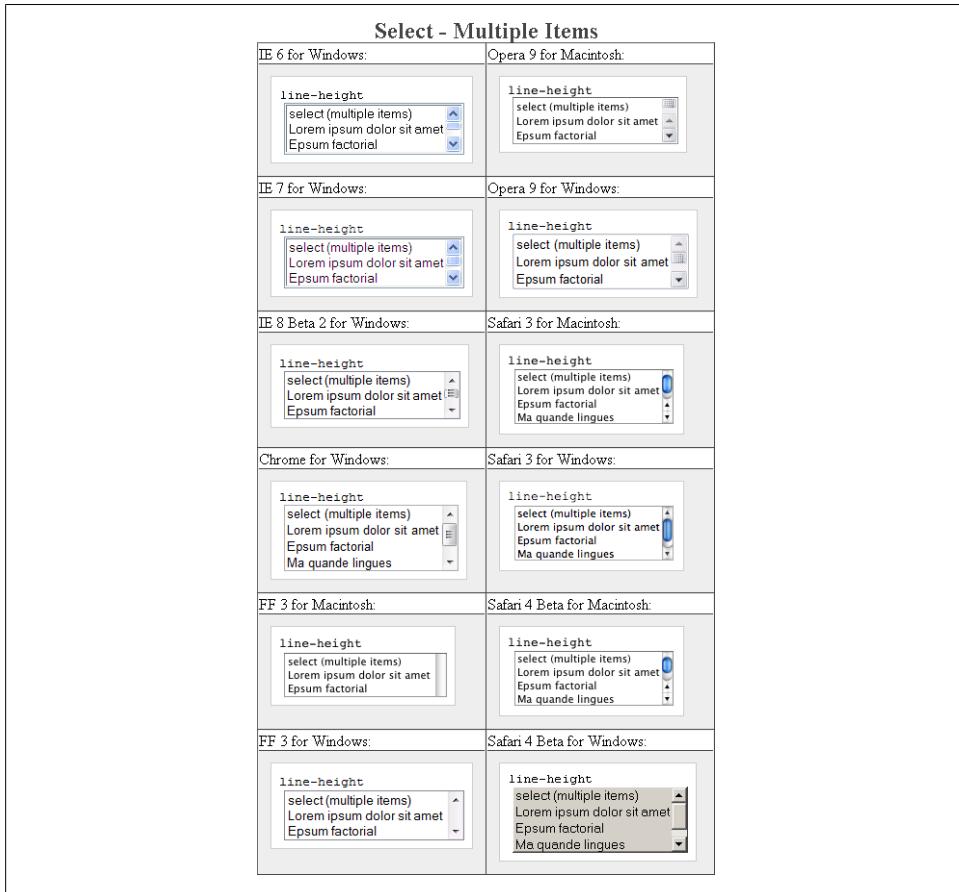
Figure E-93. Testing setting the spacing between lines of text on a select element with multiple options

margin
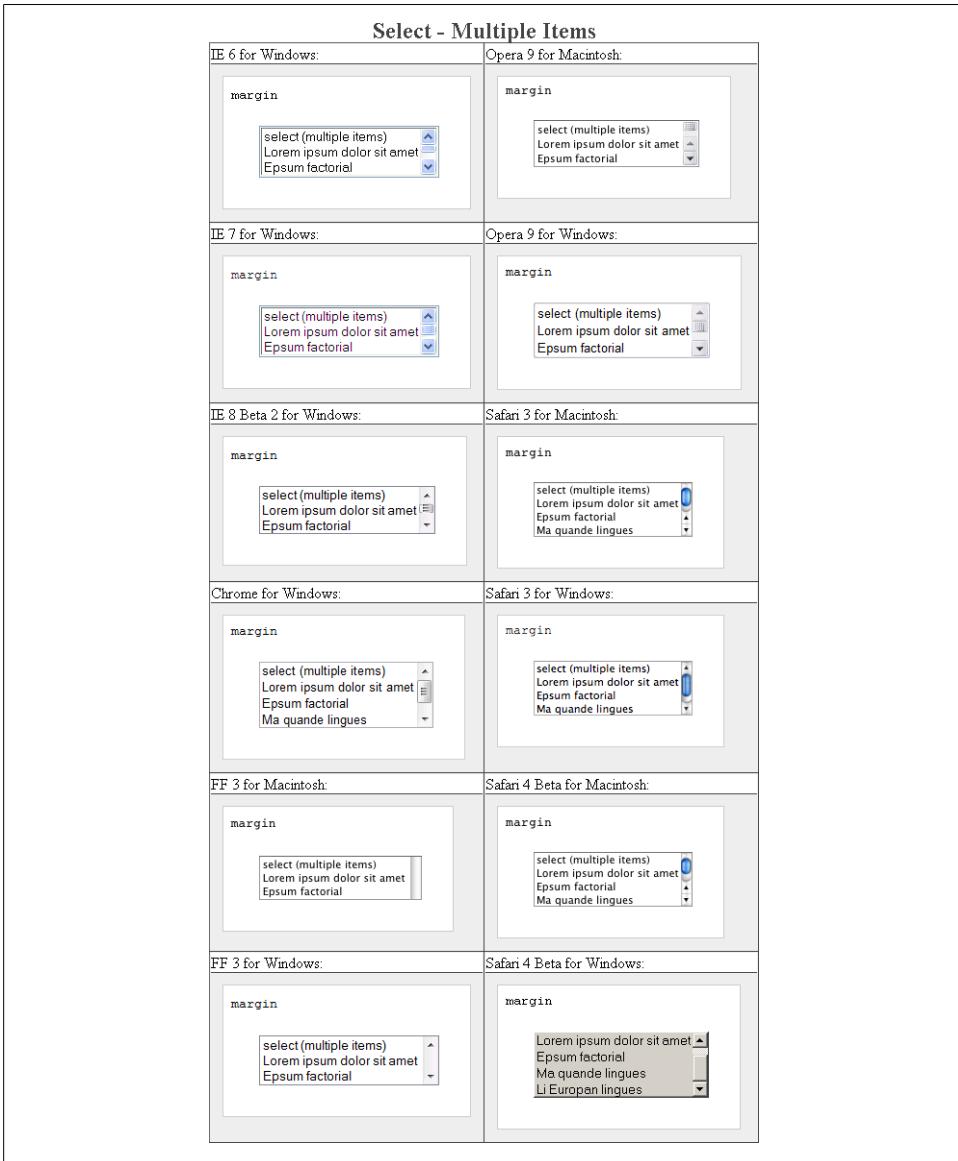
Figure E-94. Testing margins on a select element with multiple options

padding
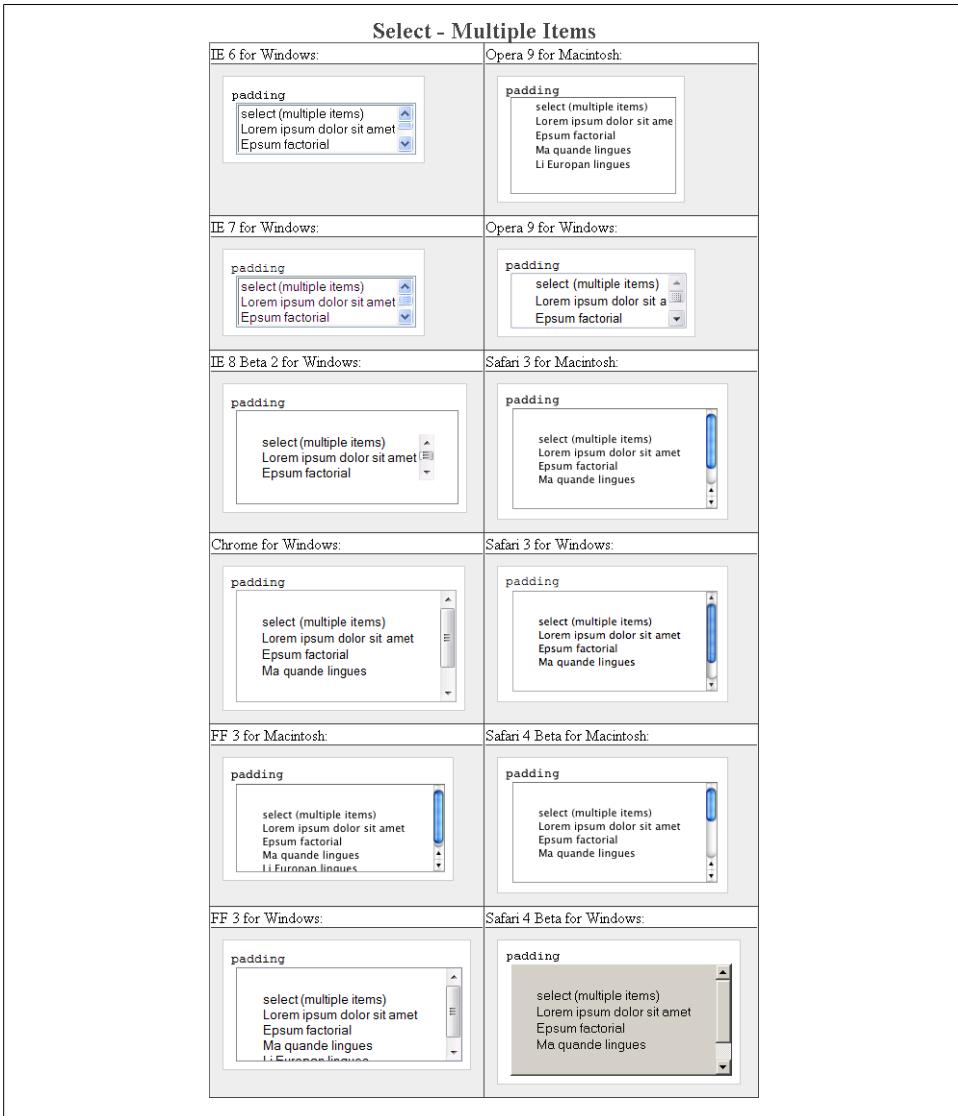
Figure E-95. Testing padding on a select element with multiple options

text-align
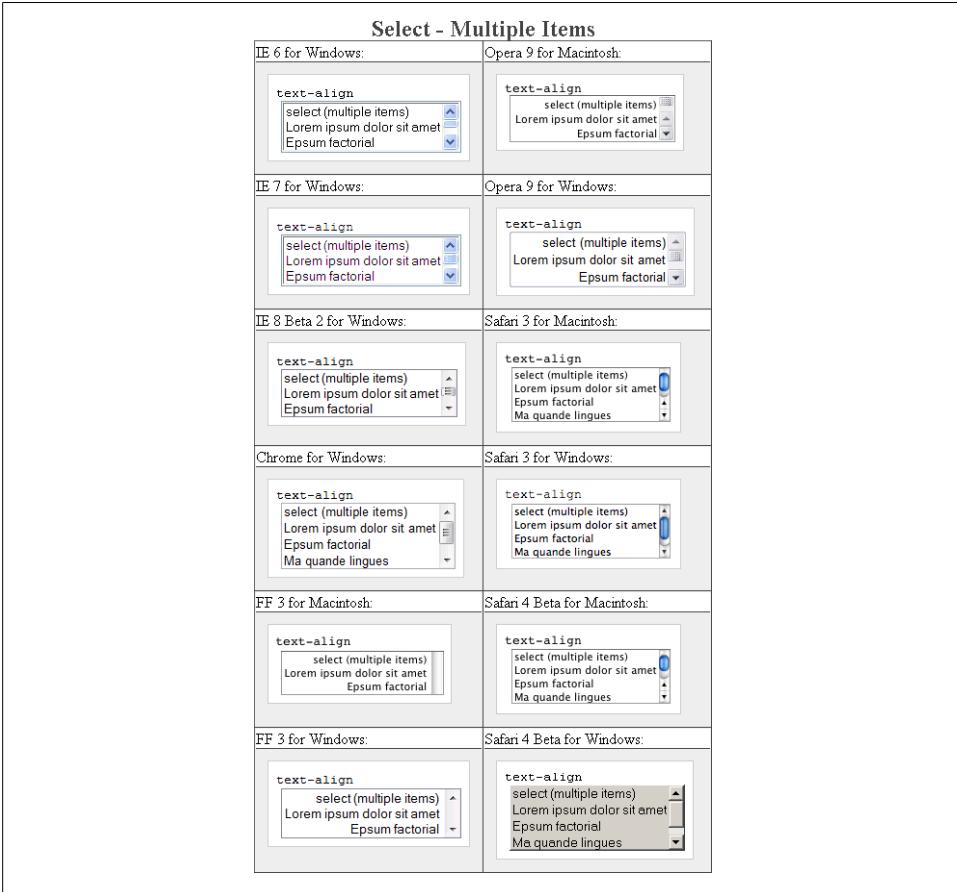
Figure E-96. Testing the alignment of text on a select element with multiple options

text-decoration
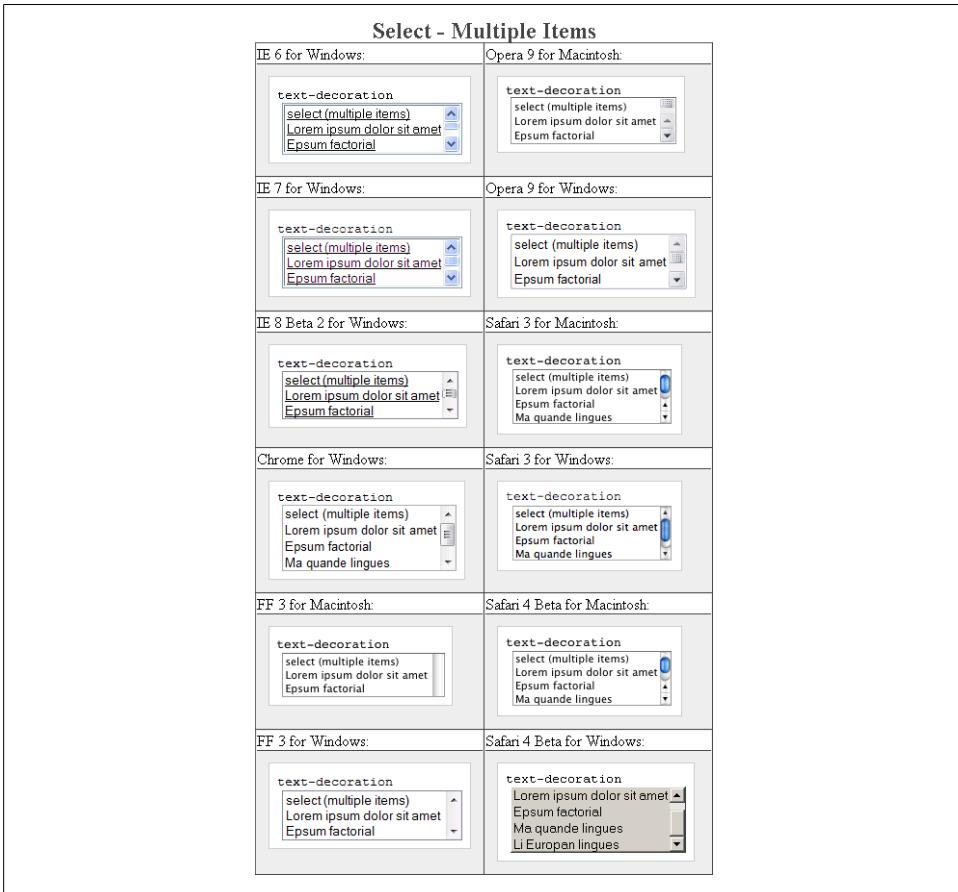
Figure E-97. Testing setting a different font on a select element with multiple options

text-indent
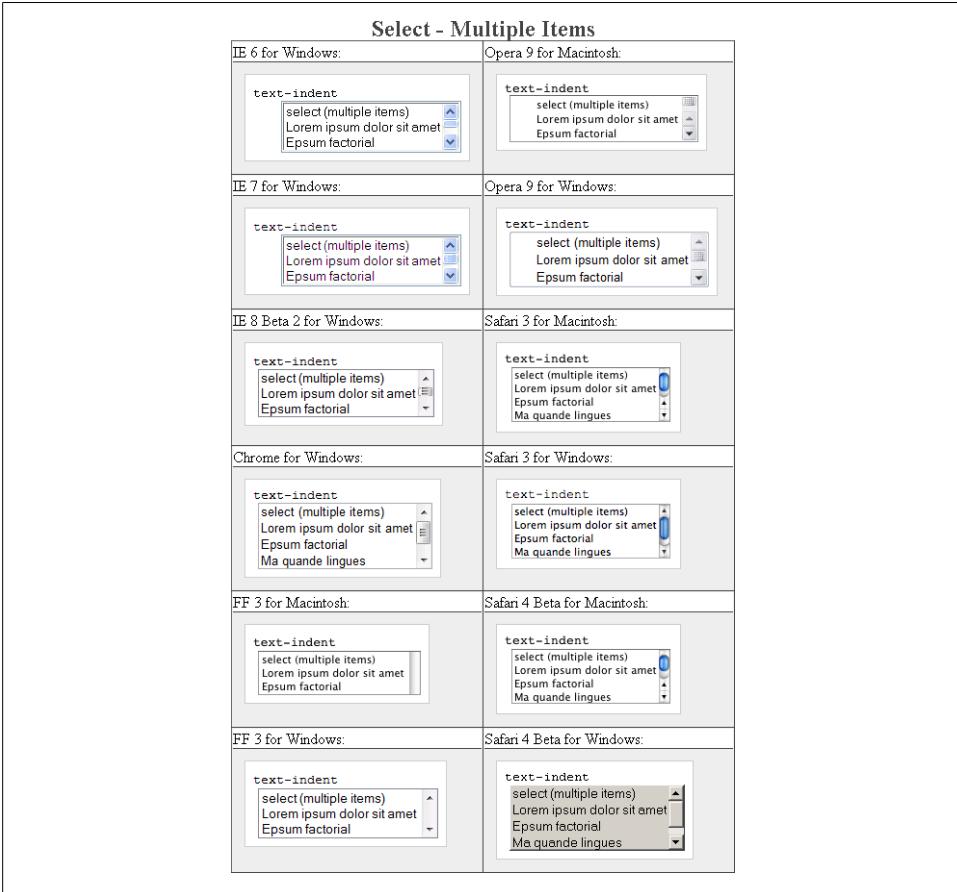
Figure E-98. Testing indenting the text on a select element with multiple options

width
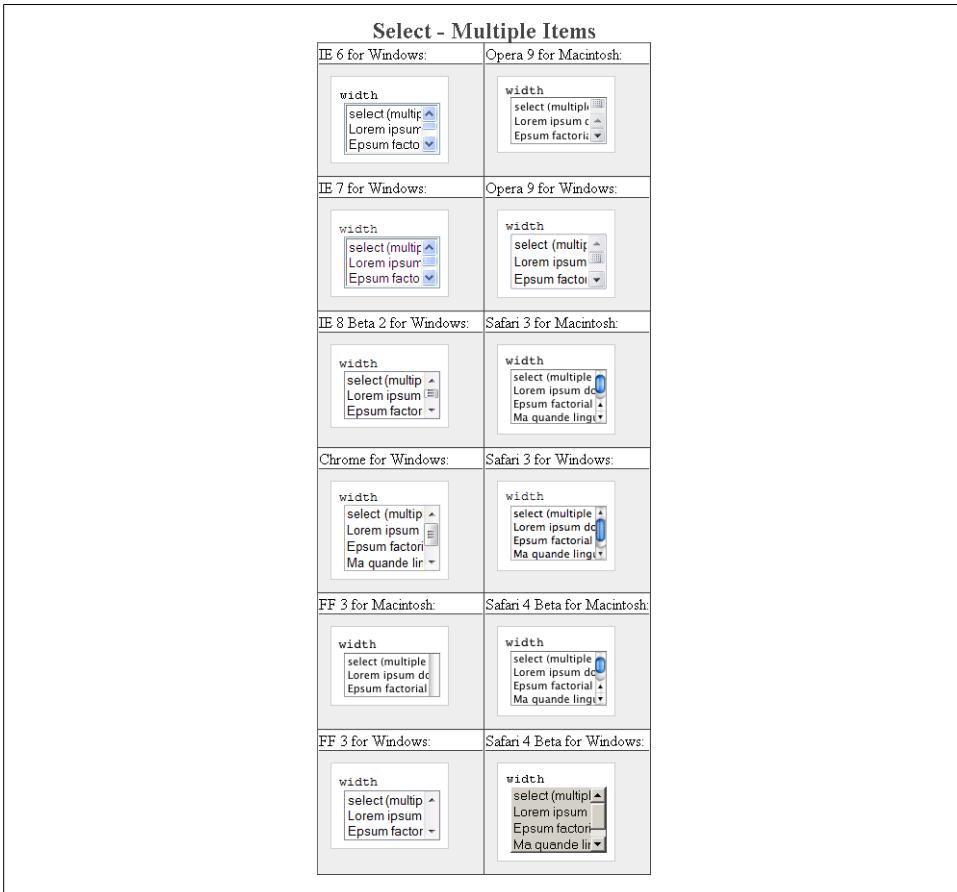
Figure E-99. Testing the width of a select element with multiple options

word-spacing
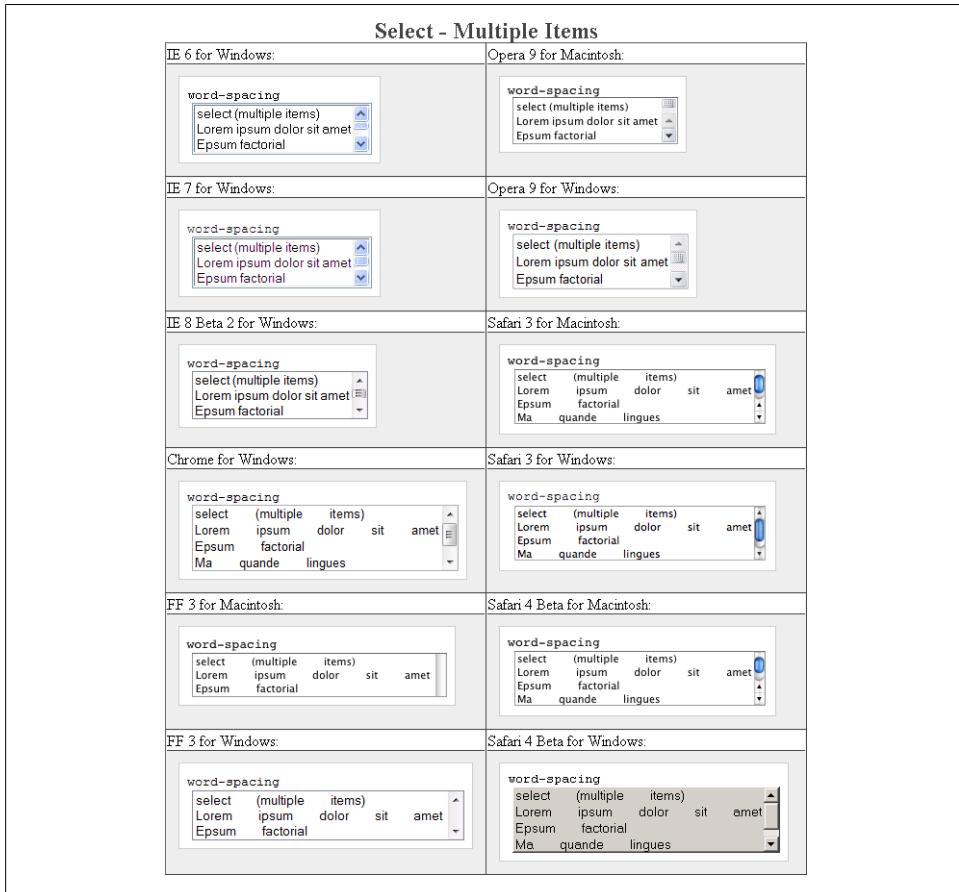
Figure E-100. Testing the spacing between words on a select element with multiple options

Select Element for Single Options

A select element is a form element that shows only one available option at a time. When a user interacts with the element, multiple options are available for selecting, but only one option may be submitted.

Table E-7. A review of the CSS properties on select elements with one option showing

	WinIE6	WinIE7	WinIE8b2	Chrome	MacFF3	WinFF3	MacOP9	WinOP9	MacSF3	WinSF3	MacSF4b	WinSF4b
background-color	Y	Y	N	Y	Y	Y	Y	Y	Y	Y	Y	Y
background-image	N	N	Y	N	N	N	Y	N	Y	Y	Y	Y
border	N	N	Y	Y	Y	Y	Y	Y	S	S	S	Y
border-color	N	N	N	Y	Y	Y	Y	Y	Y	Y	Y	Y
border-style	N	N	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y
border-width	N	N	Y	Y	Y	Y	N	N	Y	Y	Y	Y
color	Y	Y	Y	Y	Y	Y	Y	Y	N	N	N	Y
font-family	Y	Y	Y	Y	Y	Y	Y	Y	N	N	N	Y
font-size	Y	Y	Y	Y	Y	Y	Y	Y	N	N	N	Y
font-weight	Y	Y	Y	Y	Y	Y	Y	Y	N	N	N	Y
height	N	N	Y	Y	Y	Y	Y	Y	N	N	N	N
letter-spacing	N	N	N	Y	Y	Y	Y	Y	Y	Y	Y	Y
line-height	N	N	N	N	N	N	N	N	N	N	N	N
margin	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y
padding	N	N	Y	Y	Y	Y	Y	Y	N	N	N	N
text-align	N	N	N	N	N	N	N	N	N	N	N	N
text-decoration	Y	Y	Y	Y	N	N	N	N	Y	Y	Y	Y
text-indent	S	S	N	Y	Y	Y	Y	Y	Y	Y	Y	Y
width	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y
word-spacing	N	N	N	Y	Y	Y	N	N	Y	Y	Y	Y
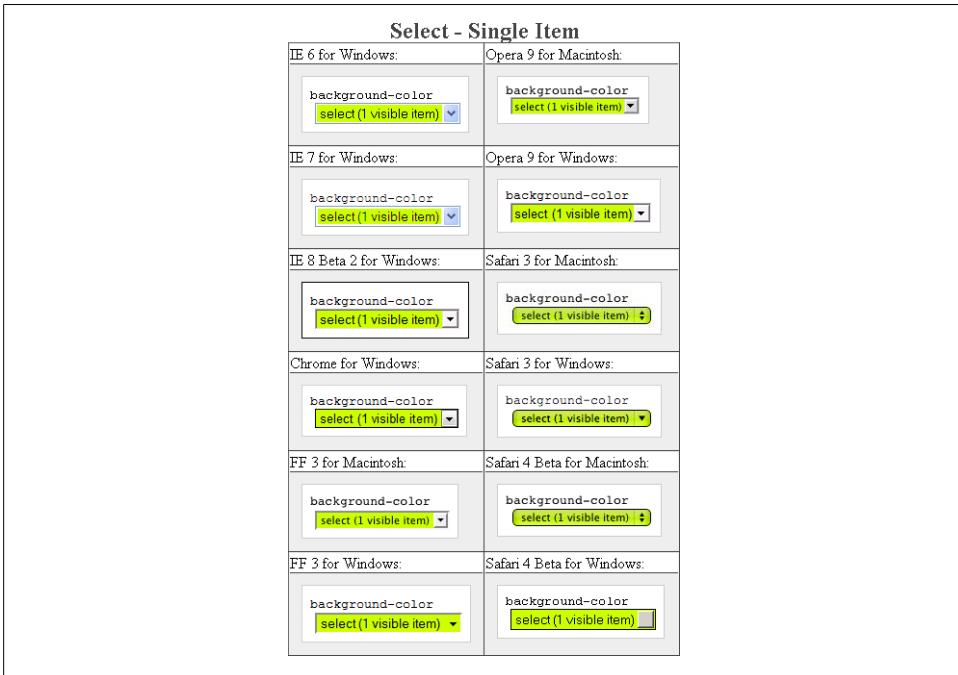
background-color

Figure E-101. Testing the background color of a select element

background-image
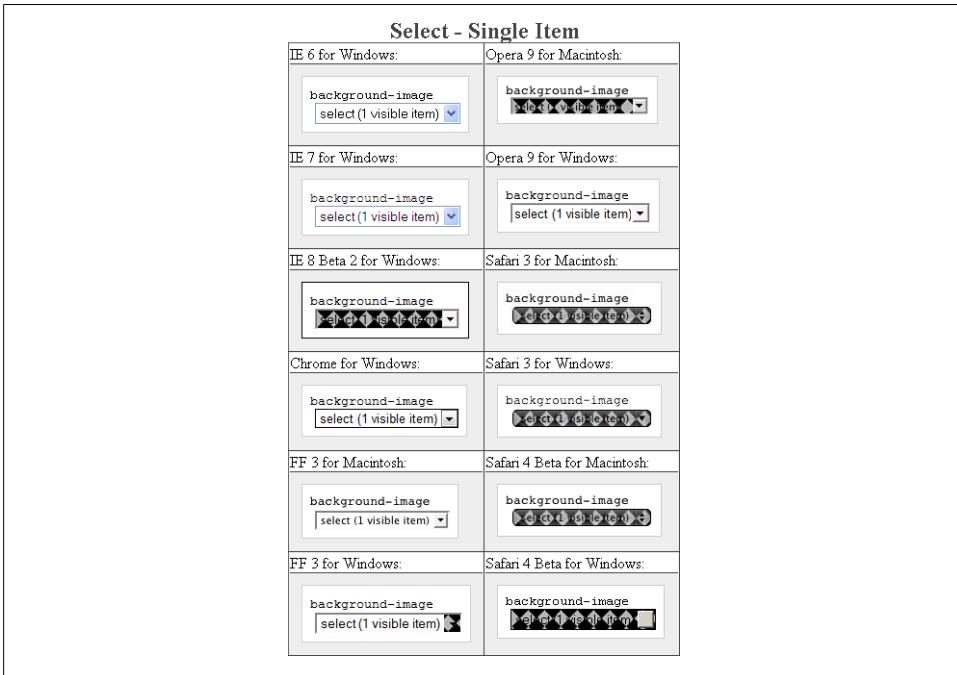
Figure E-102. Testing background images in a select element

border: 0;
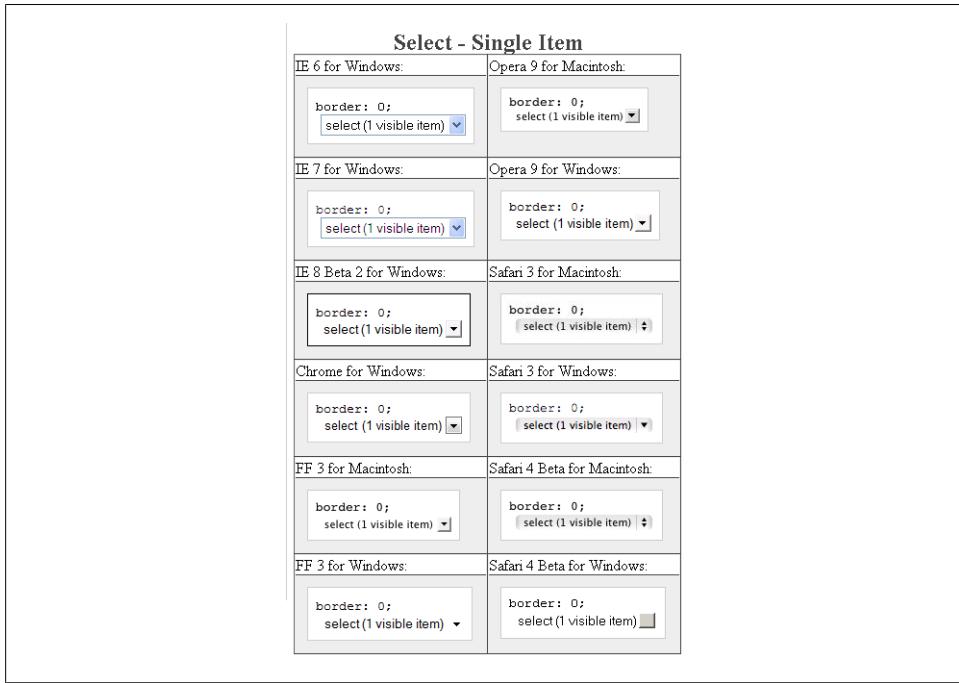
Figure E-103. Testing the removal of borders on a select element

border-color
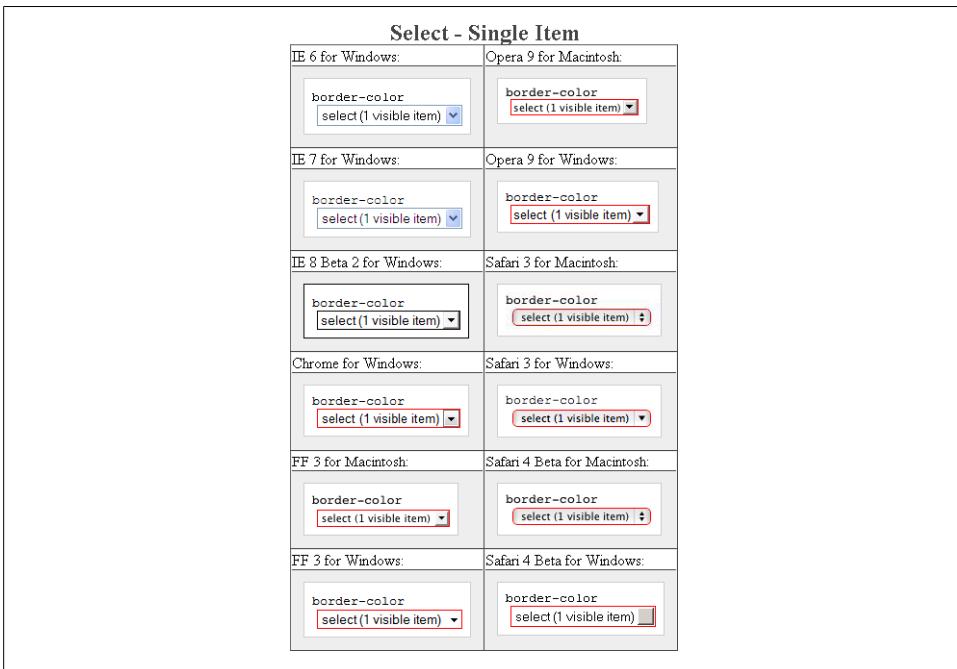
Figure E-104. Testing colors on the select element borders

border-style
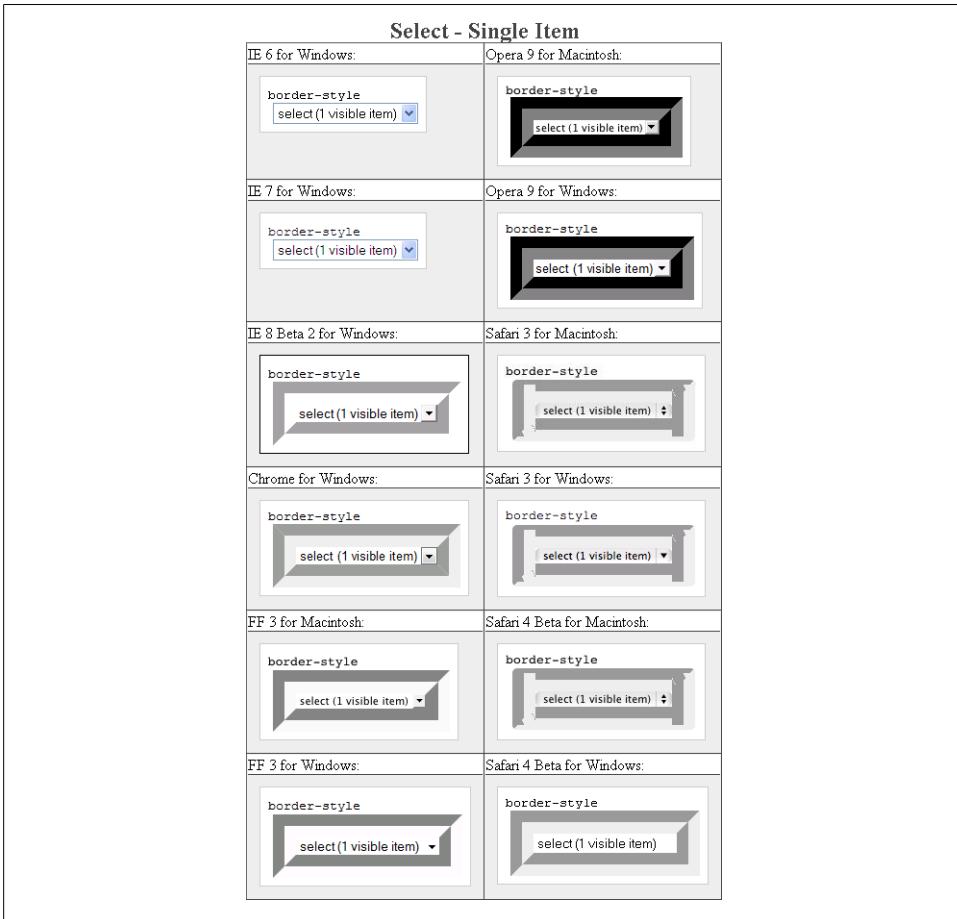
Figure E-105. Testing the styles of borders on a select element

border-width

Figure E-106. Testing the widths of borders on a select element

color
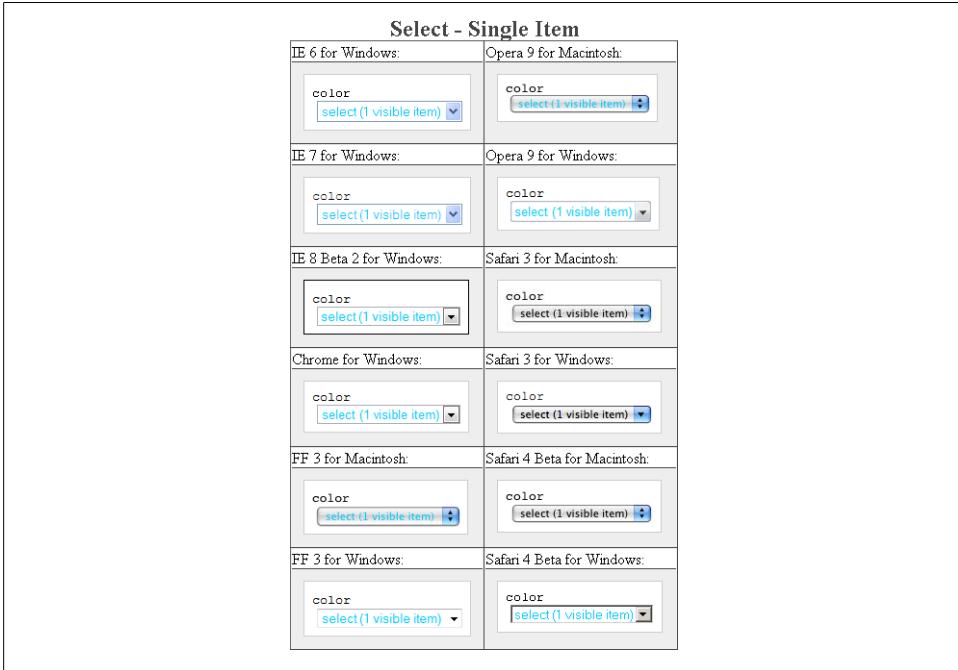
Figure E-107. Testing the color on a select element

font-family
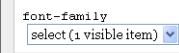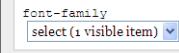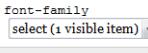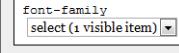
Figure E-108. Testing setting a different font on a select element

font-size

Figure E-109. Testing a different size of font on a select element

font-weight
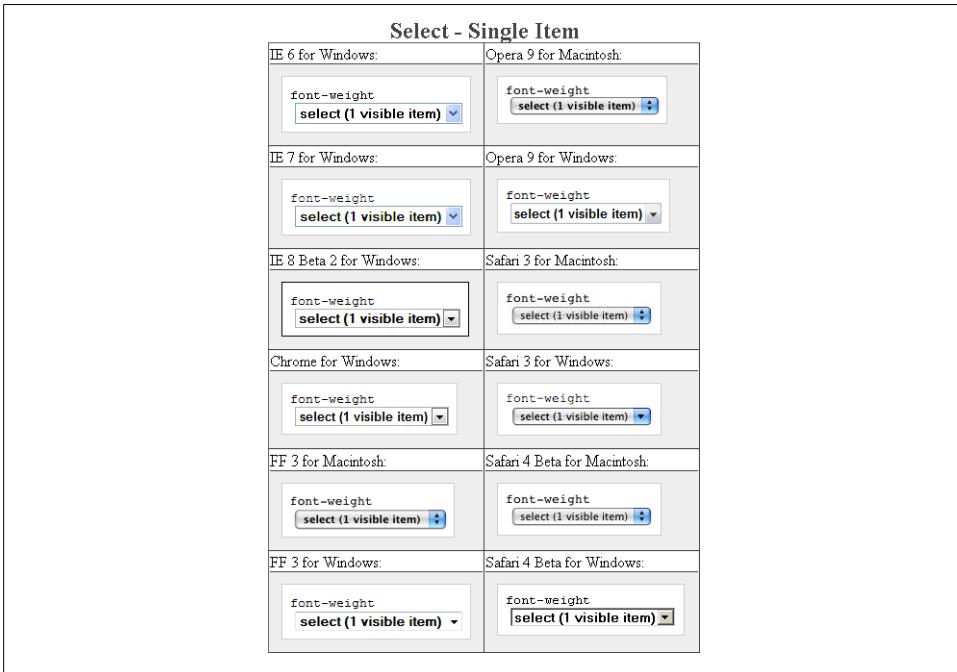
Figure E-110. Testing a bold font on a select element

height
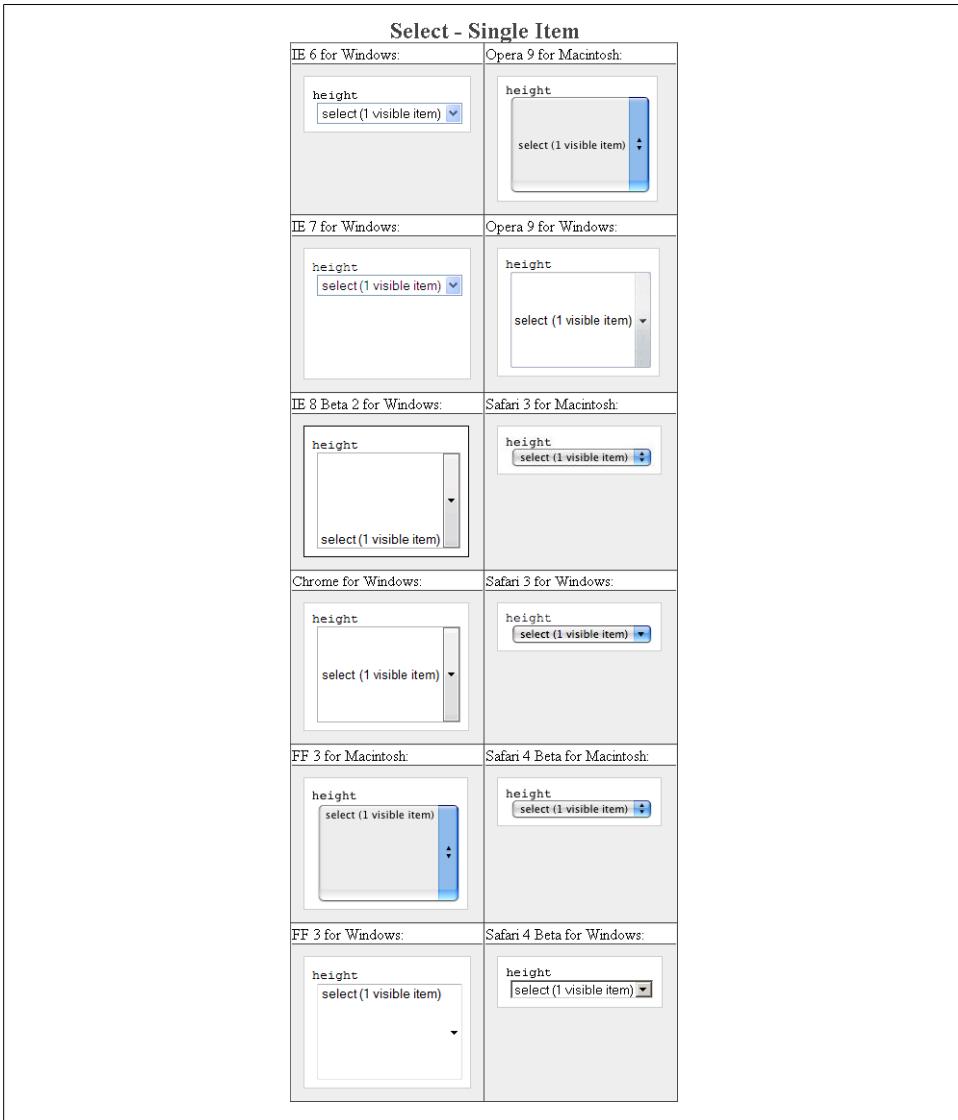
Figure E-111. Testing setting a height for a select element

letter-spacing
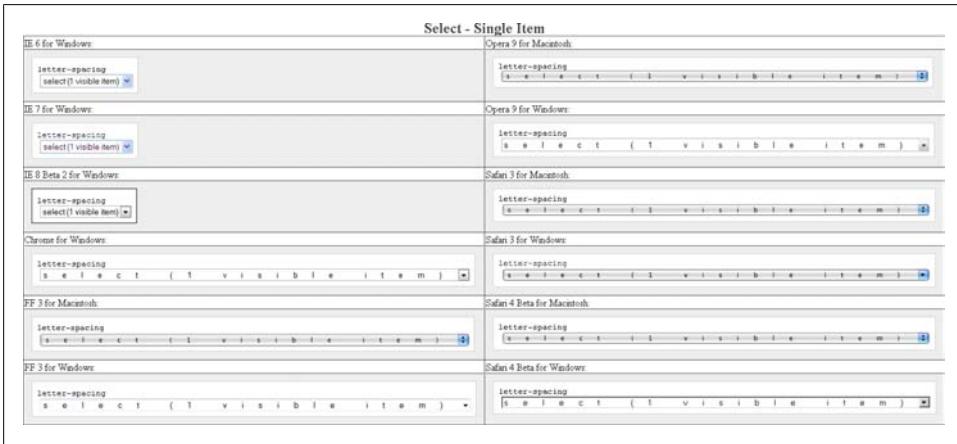
Figure E-112. Testing the letter spacing of a select element

line-height

Figure E-113. Testing setting the spacing between lines of text on a select element

margin
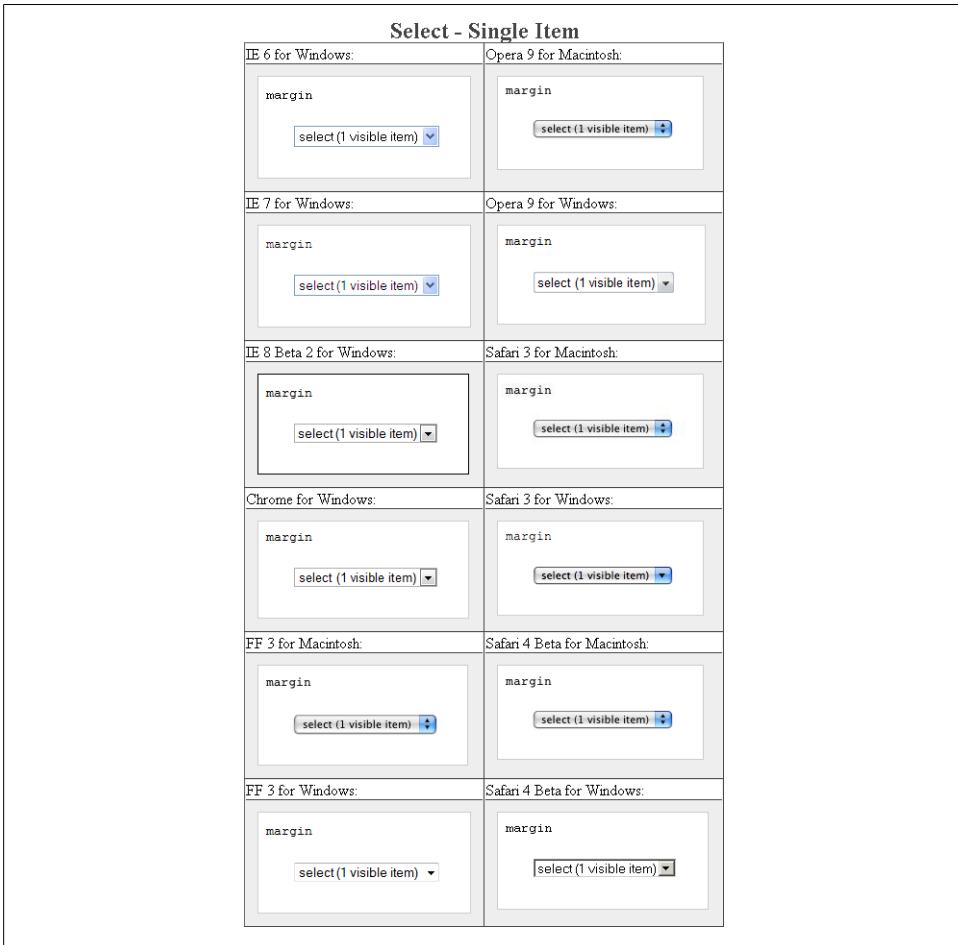
Figure E-114. Testing margins on a select element

padding
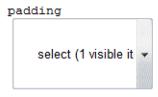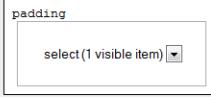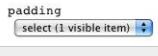
Figure E-115. Testing padding on a select element

text-align

Figure E-116. Testing the alignment of text on a select element

text-decoration
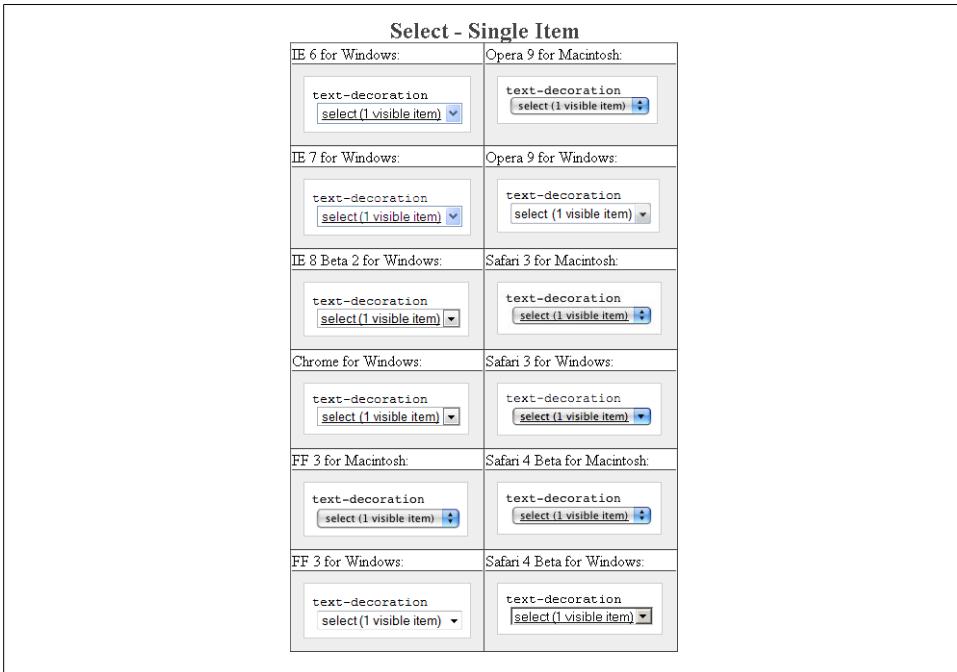
Figure E-117. Testing setting a different font on a select element

text-indent
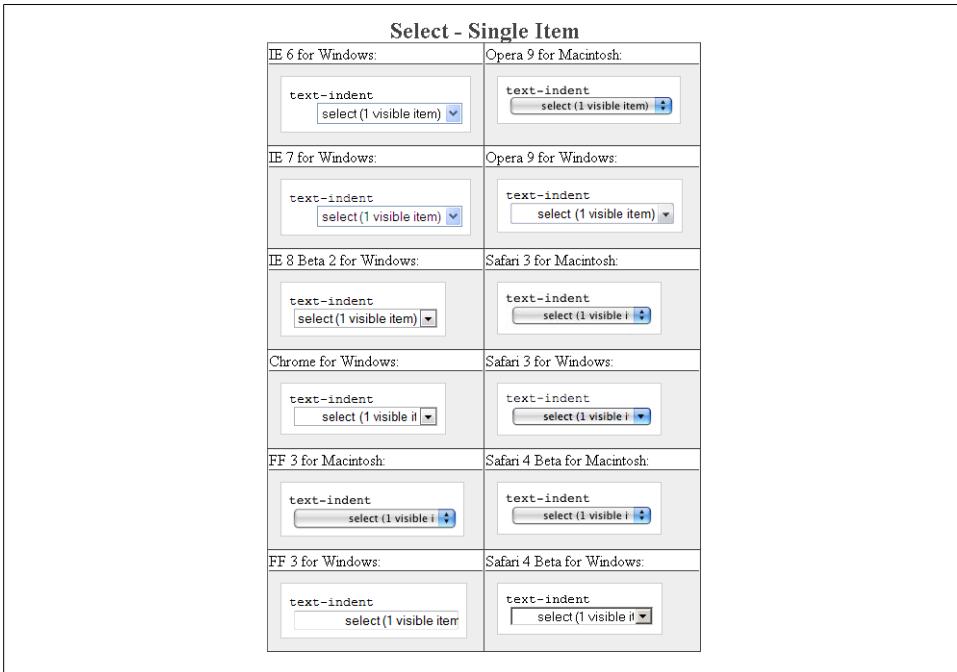
Figure E-118. Testing indenting the text on a select element

width
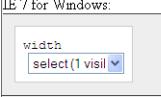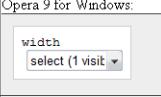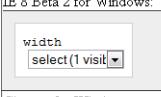
Figure E-119. Testing the width of a select element

word-spacing

Figure E-120. Testing the spacing between words on a select element

Submit Element

When activated, the Submit button simply submits the form.

Table E-8. A review of the CSS properties on the Submit button

	WinIE6	WinIE7	WinIE8b2	Chrome	MacFF3	WinFF3	MacOP9	WinOP9	MacSF3	WinSF3	MacSF4b	WinSF4b
background-color	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y
background-image	N	N	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y
border	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y
border-color	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y
border-style	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y
border-width	Y	Y	Y	Y	Y	Y	N	N	Y	Y	Y	Y
color	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y
font-family	Y	Y	Y	Y	Y	Y	Y	Y	N	N	N	Y
font-size	Y	Y	Y	Y	Y	Y	Y	Y	N	N	N	Y
font-weight	Y	Y	Y	Y	Y	Y	Y	Y	N	N	N	Y
height	Y	Y	Y	Y	Y	Y	Y	Y	N	N	N	Y
letter-spacing	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y
line-height	N	N	N	N	N	N	N	N	N	N	N	N
margin	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y
padding	Y	Y	Y	Y	N	N	Y	Y	N	N	N	Y
text-align	Y	Y	N	N	N	N	N	N	N	N	N	N
text-decoration	Y	Y	Y	N	N	N	N	N	Y	Y	Y	Y
text-indent	S	S	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y
width	Y	Y	Y	Y	Y	Y	Y	N	Y	Y	Y	Y
word-spacing	Y	Y	Y	Y	Y	Y	Y	N	Y	Y	Y	Y
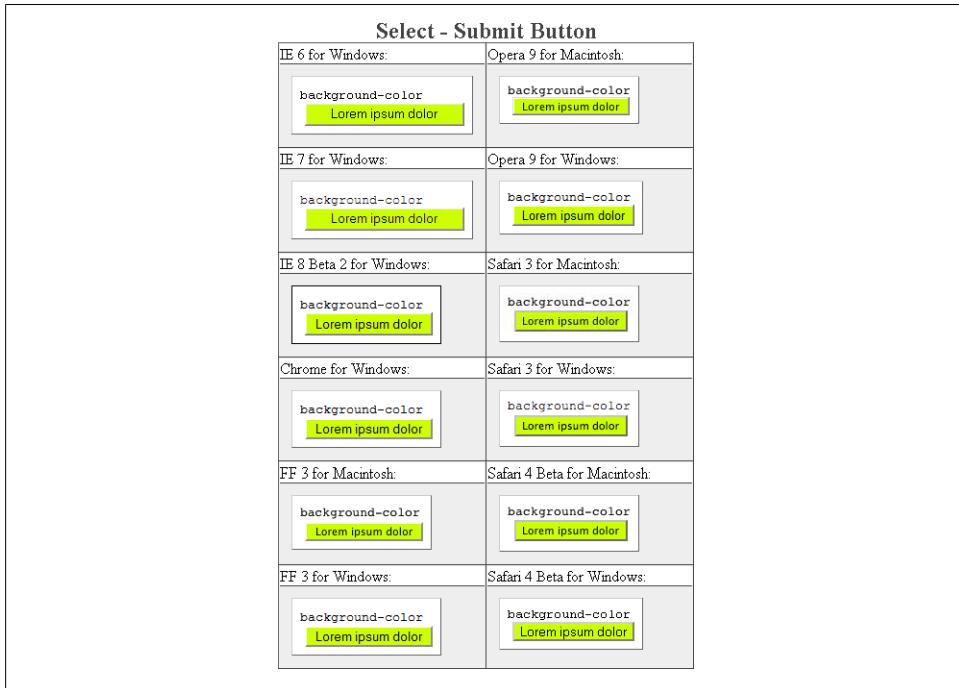
background-color

Figure E-121. Testing the background color of the Submit button

background-image
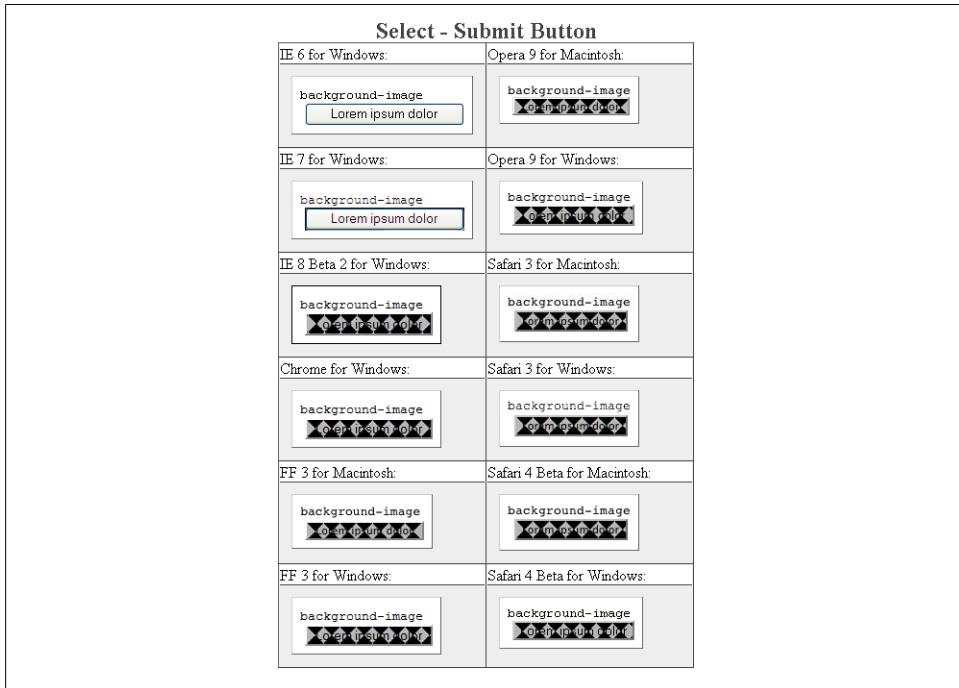
Figure E-122. Testing background images in the Submit button

border: 0;

Figure E-123. Testing the removal of borders on the Submit button

border-color
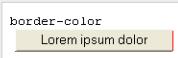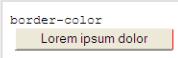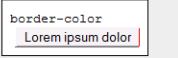
Figure E-124. Testing colors on the Submit button borders

border-style

Figure E-125. Testing the styles of borders on the Submit button

border-width
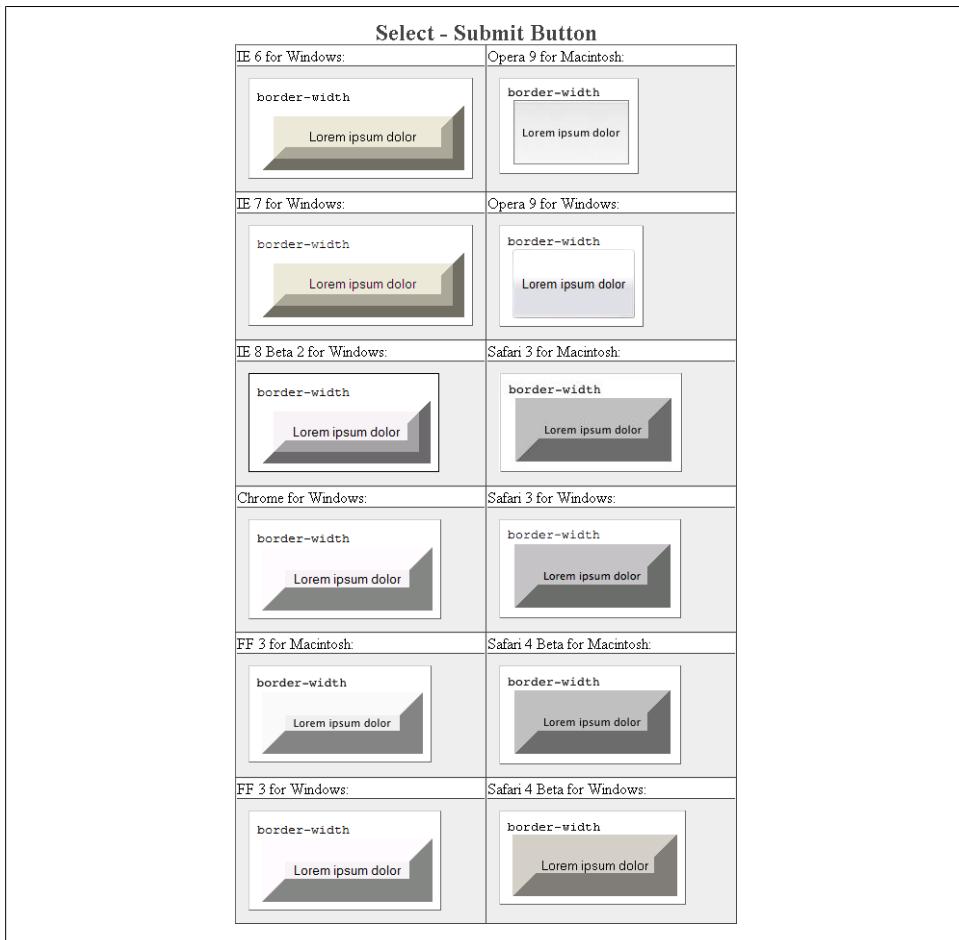
Figure E-126. Testing the widths of borders on the Submit button

color
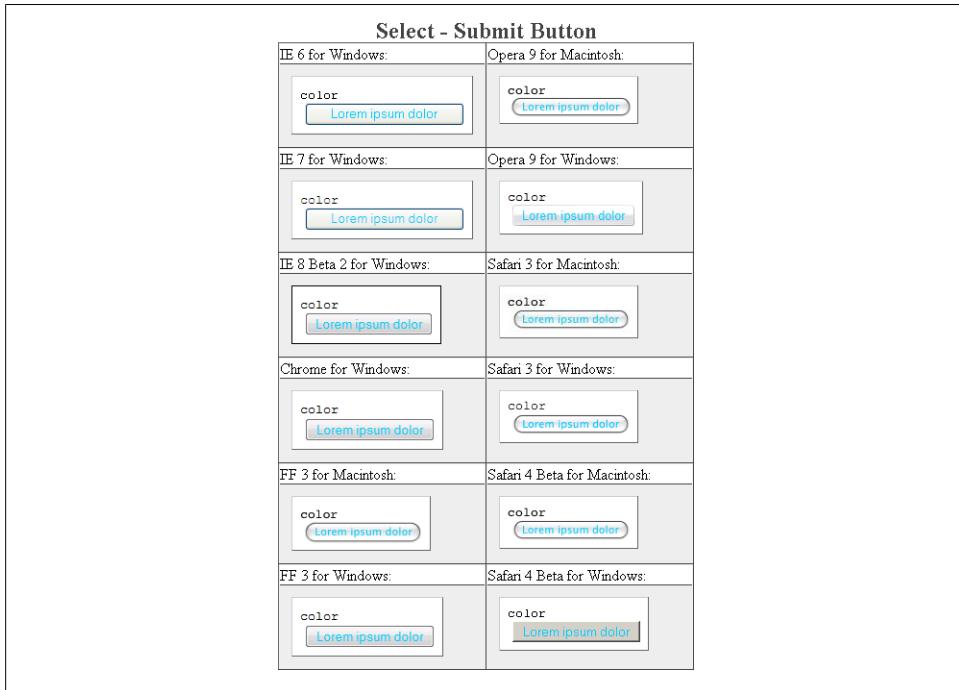
Figure E-127. Testing the color on the Submit button

font-family
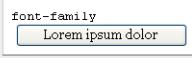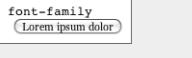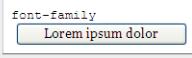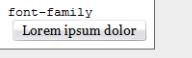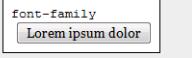
Figure E-128. Testing setting a different font on the Submit button

font-size
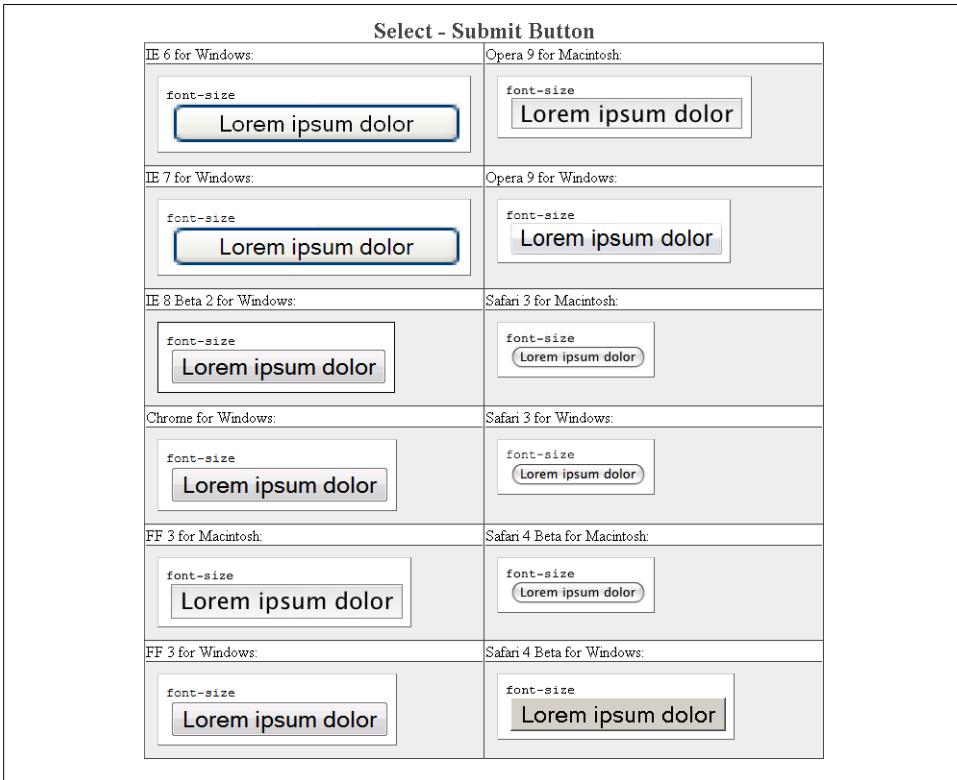
Figure E-129. Testing a different size of font on the Submit button

font-weight
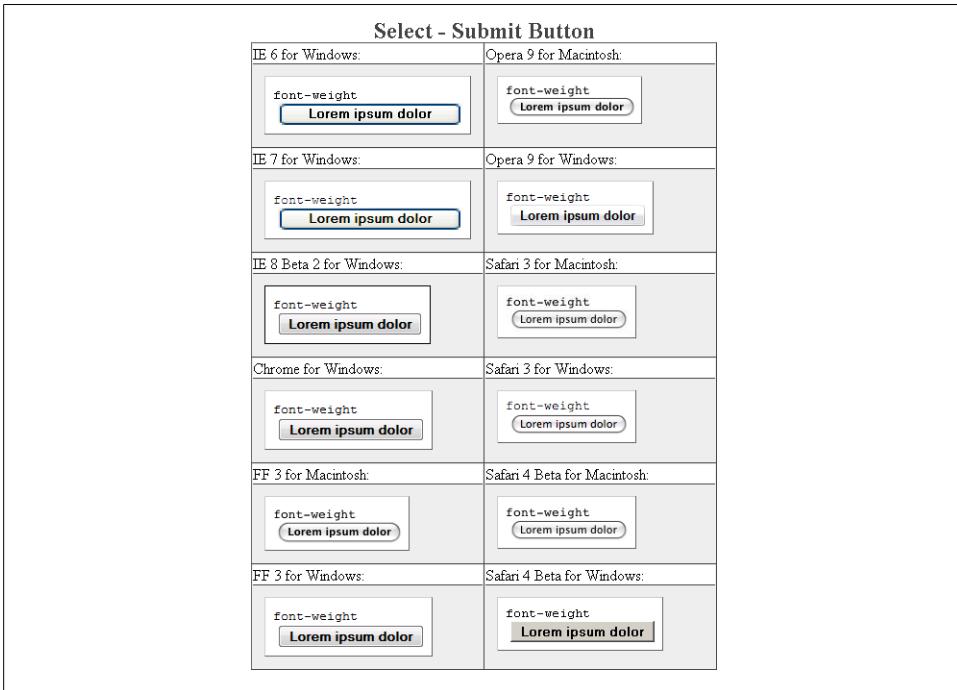
Figure E-130. Testing a bold font on the Submit button

height
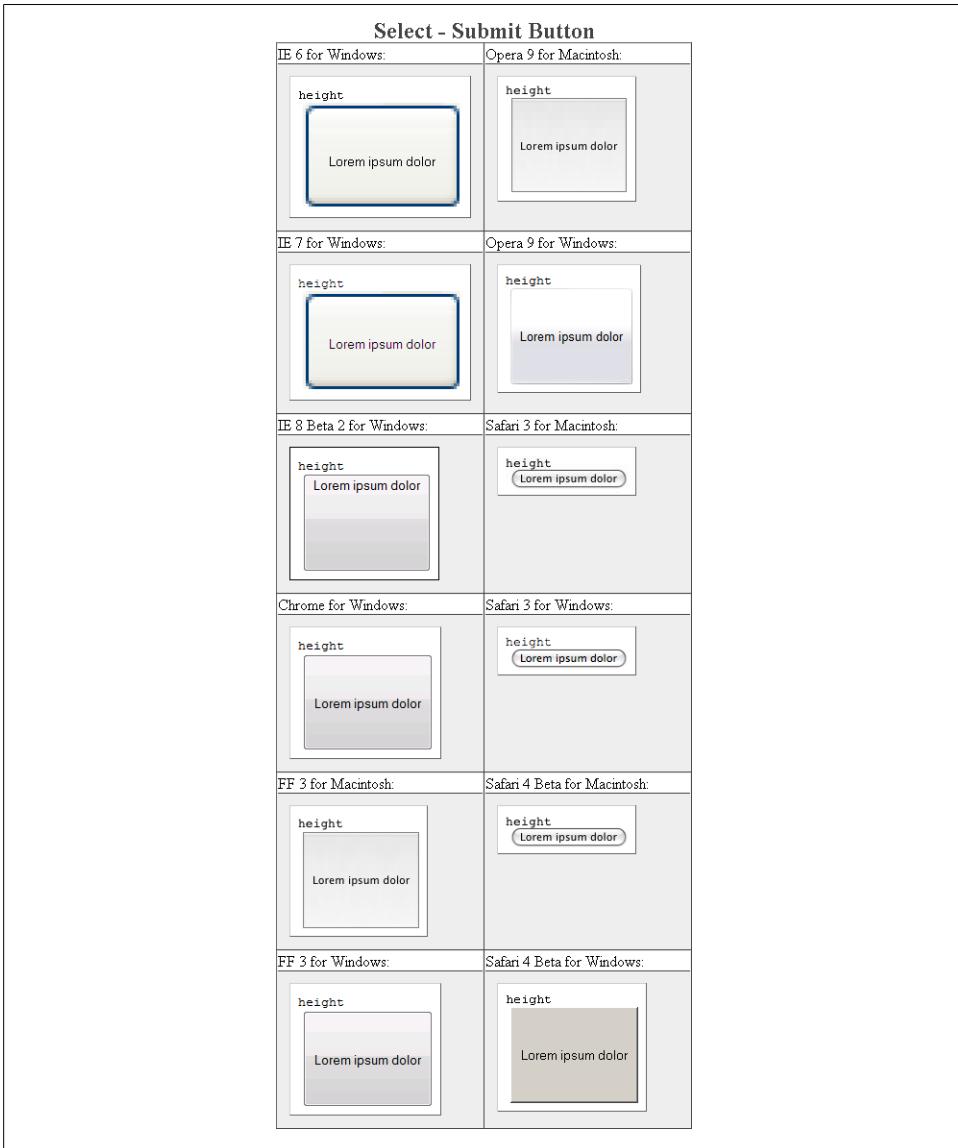
Figure E-131. Testing setting a height for the Submit button

letter-spacing
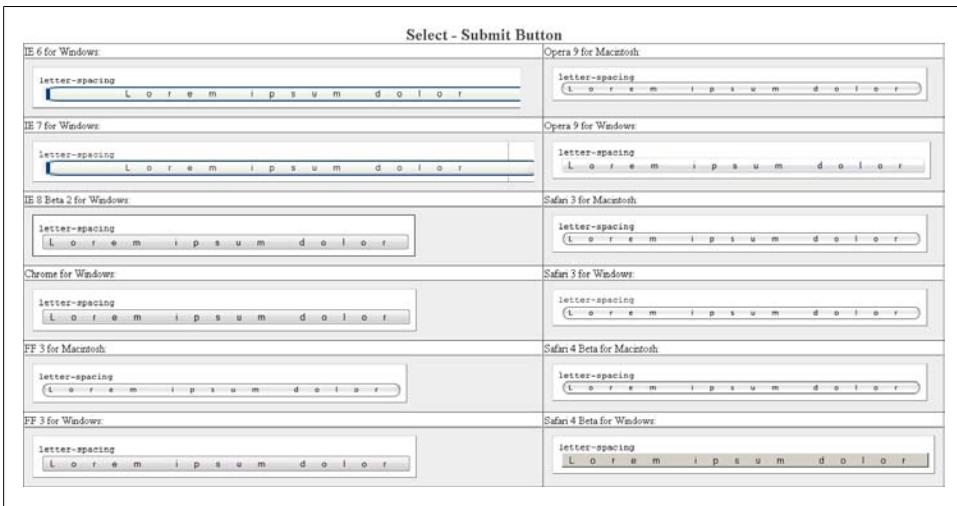
Figure E-132. Testing the letter spacing of the Submit button

line-height
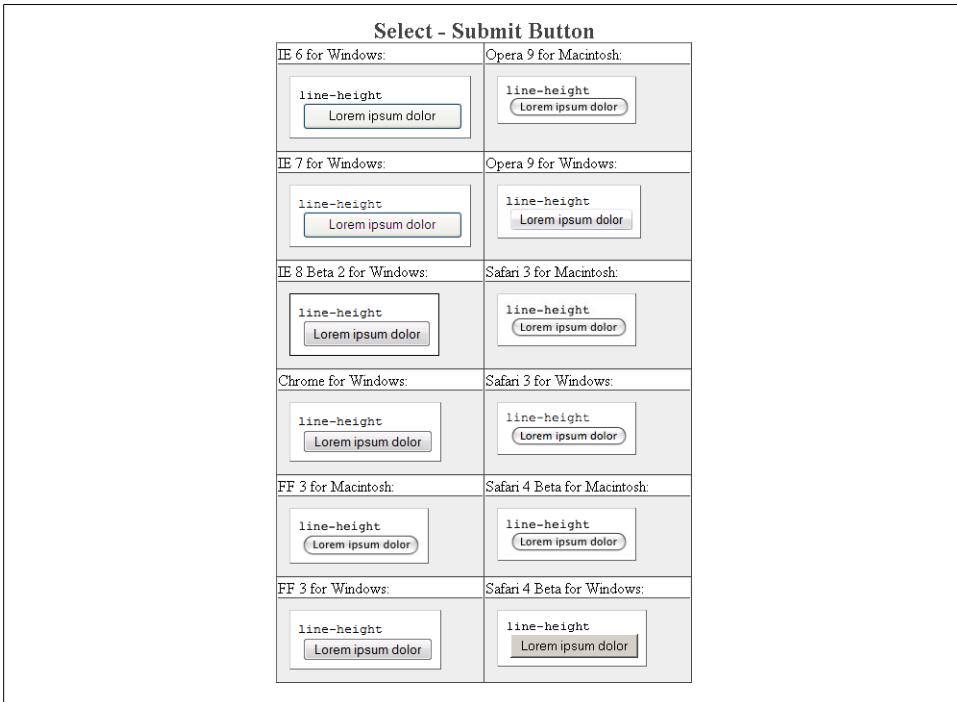
Figure E-133. Testing setting the spacing between lines of text on the Submit button

margin
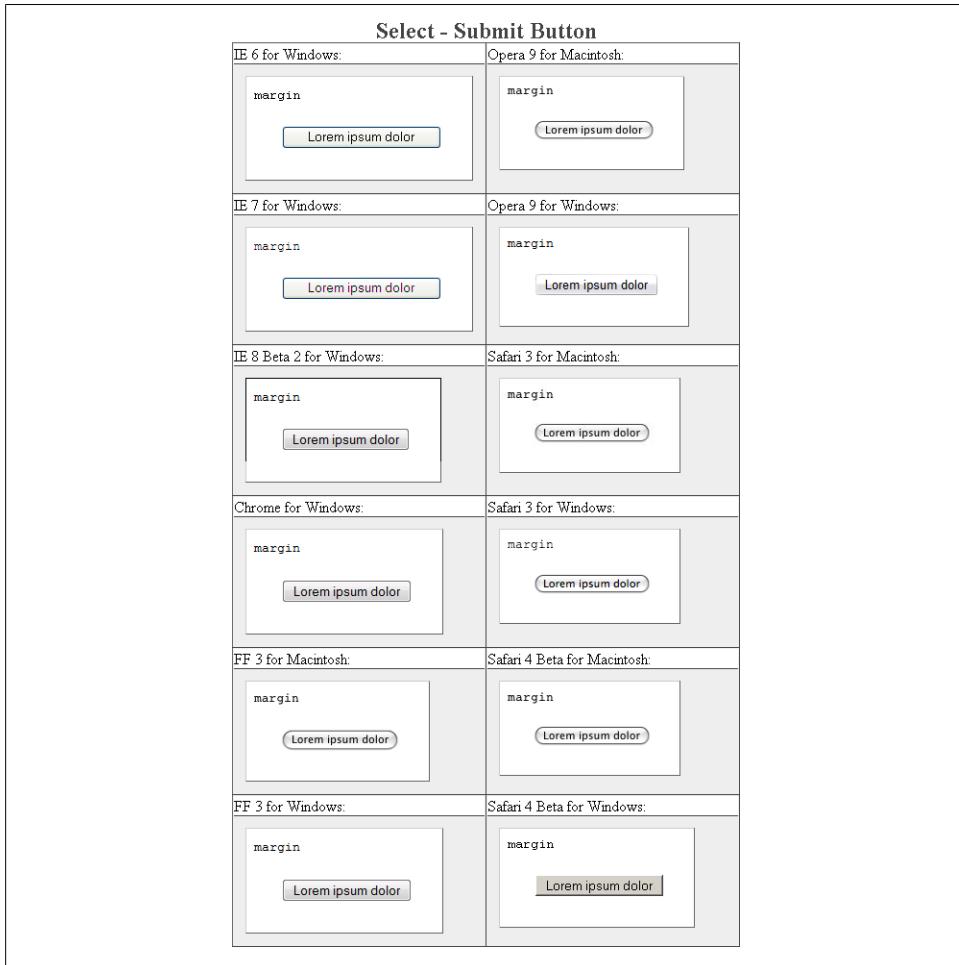
Figure E-134. Testing margins on the Submit button

padding
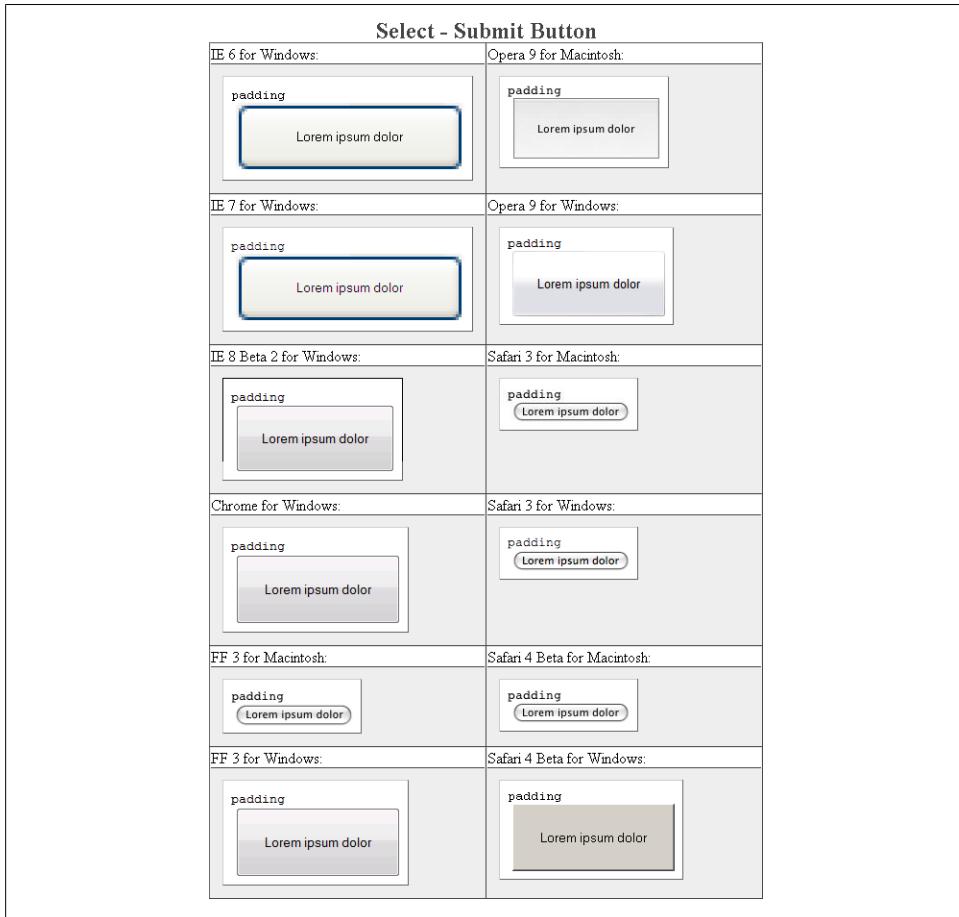
Figure E-135. Testing padding on the Submit button

text-align
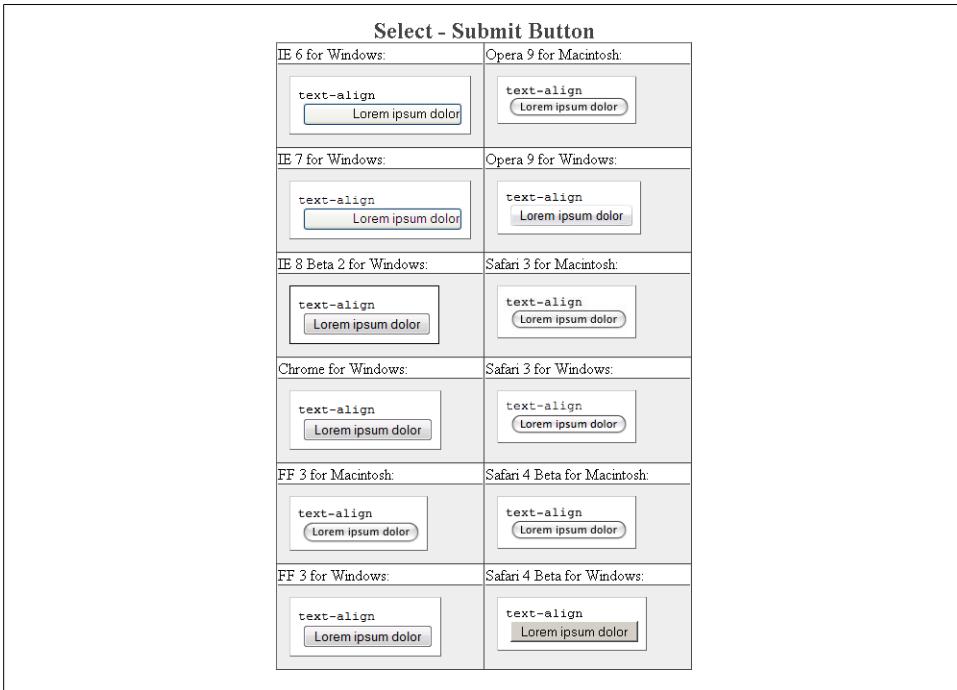
Figure E-136. Testing the alignment of text on the Submit button

text-decoration
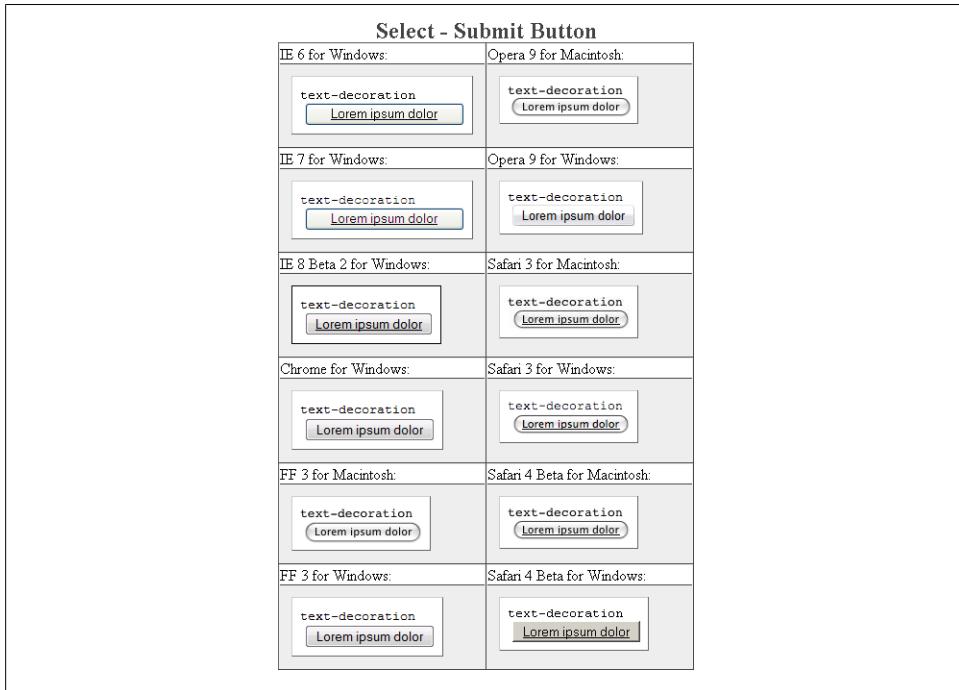
Figure E-137. Testing setting a different font on the Submit button

text-indent
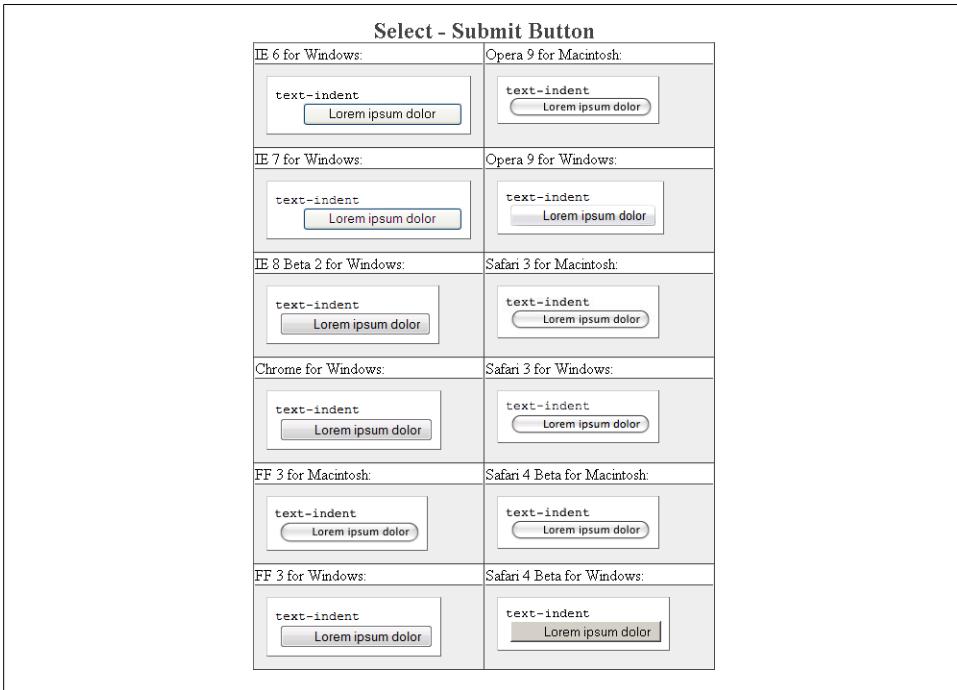
Figure E-138. Testing indenting the text on the Submit button

width
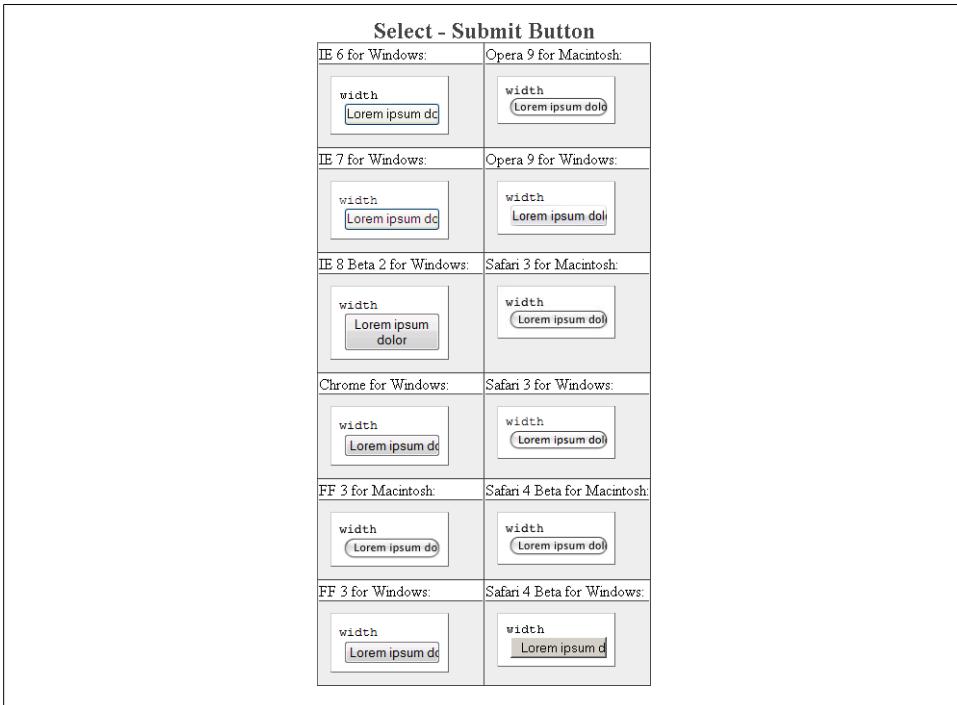
Figure E-139. Testing the width of the Submit button

word-spacing
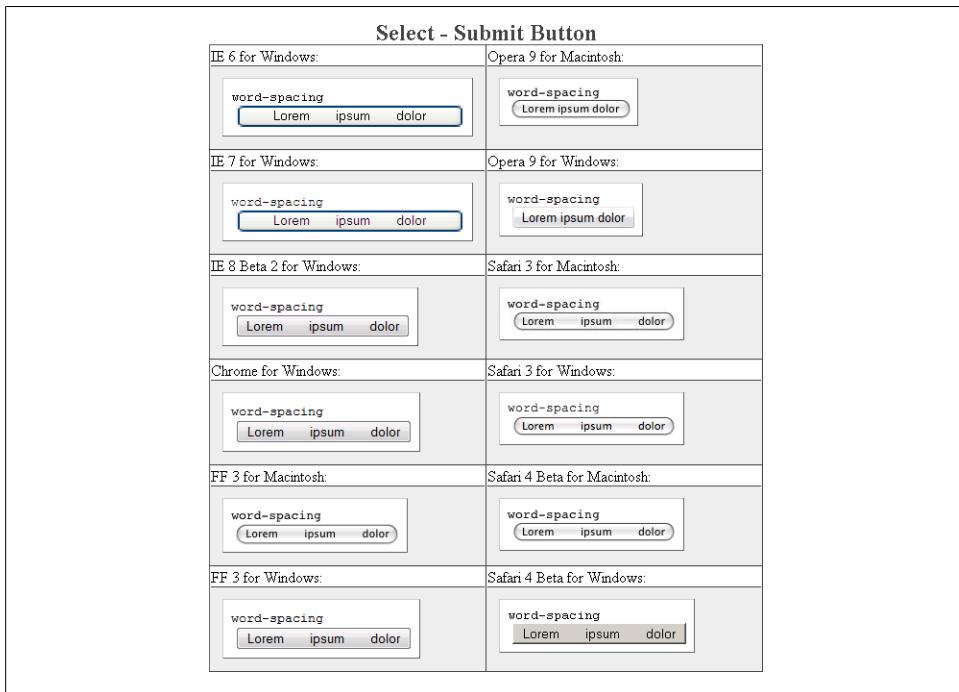
Figure E-140. Testing the spacing between words on the Submit button

Textarea Element

A `textarea` element is an HTML form element that allows for multiline text input by the user.

Table E-9. A review of the CSS properties on textarea

	WinIE6	WinIE7	WinIE8b2	Chrome	MacFF3	WinFF3	MacOP9	WinOP9	MacSF3	WinSF3	MacSF4b	WinSF4b
background-color	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y
background-image	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y
border	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y
border-color	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y
border-style	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y
border-width	Y	Y	Y	Y	Y	Y	N	N	Y	Y	Y	Y
color	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y
font-family	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y
font-size	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y
font-weight	Y	N	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y
height	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y
letter-spacing	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y
line-height	Y	Y	Y	Y	Y	Y	Y	Y	N	N	N	Y
margin	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y
padding	S	S	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y
text-align	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y
text-decoration	Y	Y	Y	Y	Y	Y	N	N	Y	Y	Y	Y
text-indent	S	S	Y	Y	Y	Y	N	N	Y	Y	Y	Y
width	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y
word-spacing	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y

background-color
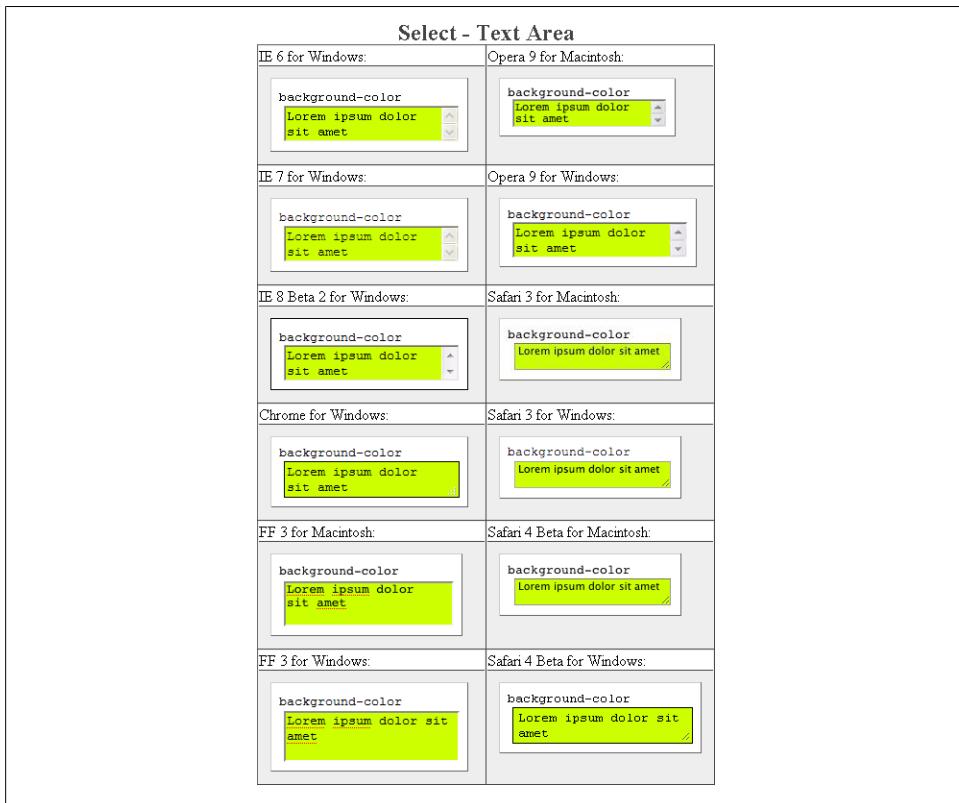
Figure E-141. Testing the background color of the textarea element

background-image
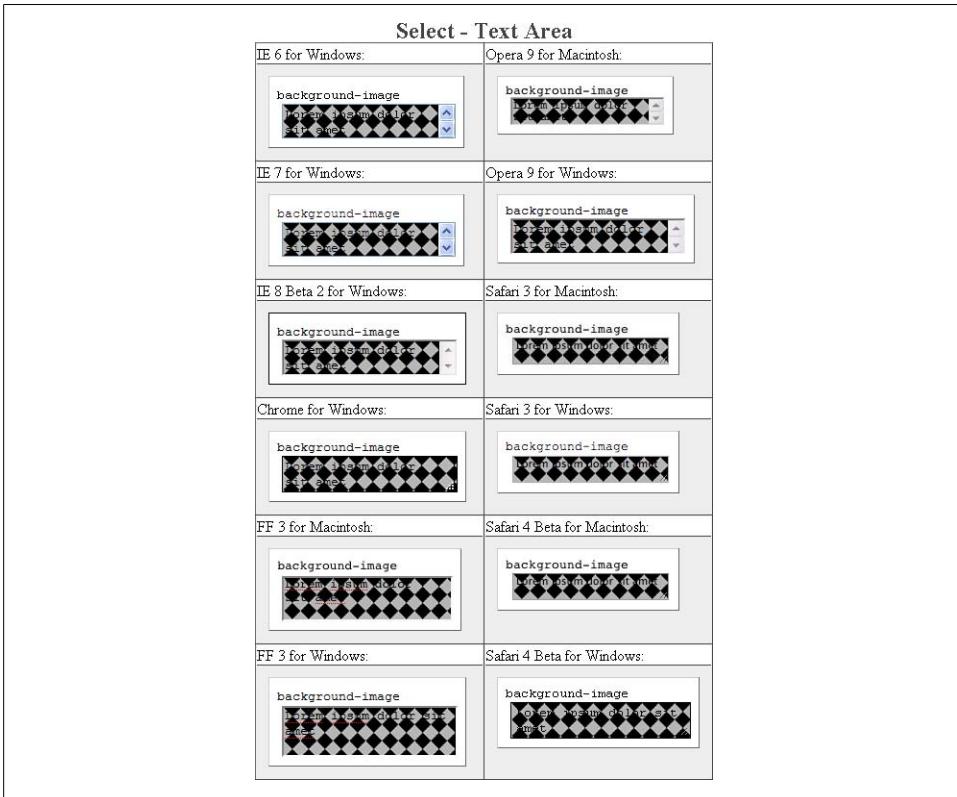
Figure E-142. Testing background images in the textarea element

border: 0;
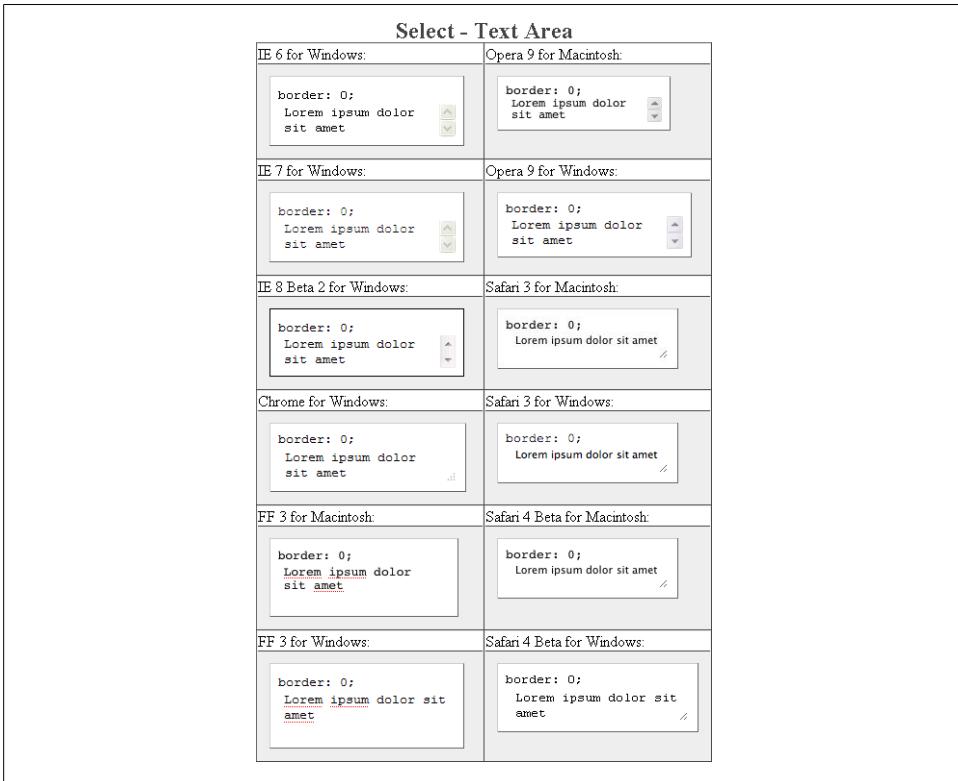
Figure E-143. Testing the removal of borders on the textarea element

border-color
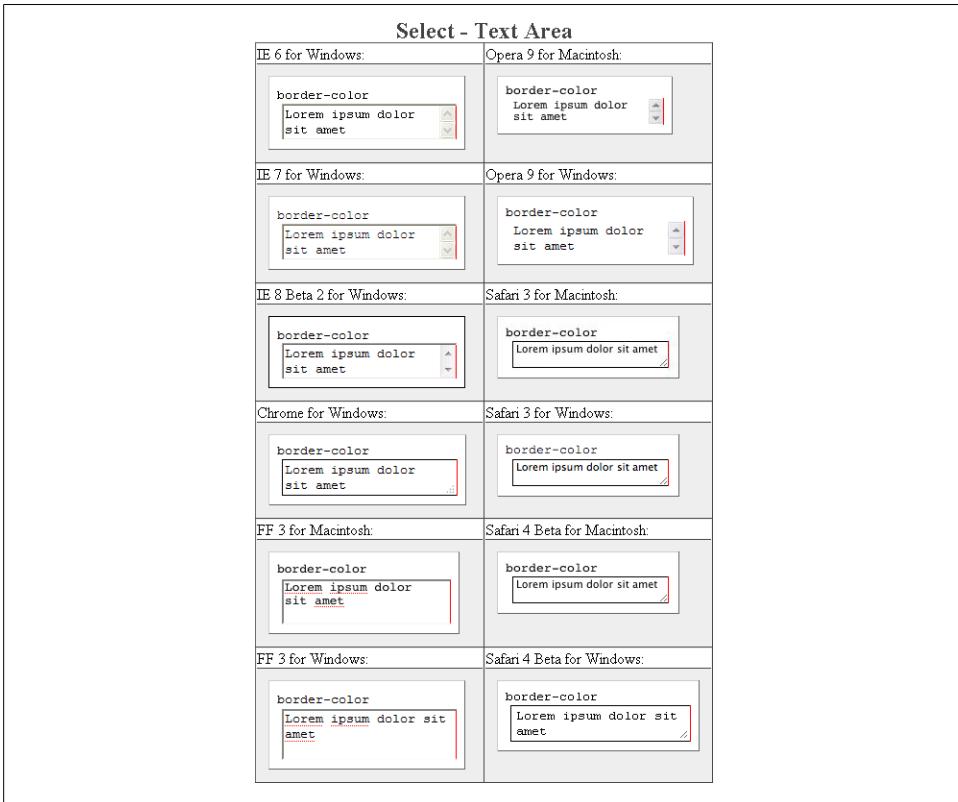
Figure E-144. Testing colors on the textarea element borders

border-style
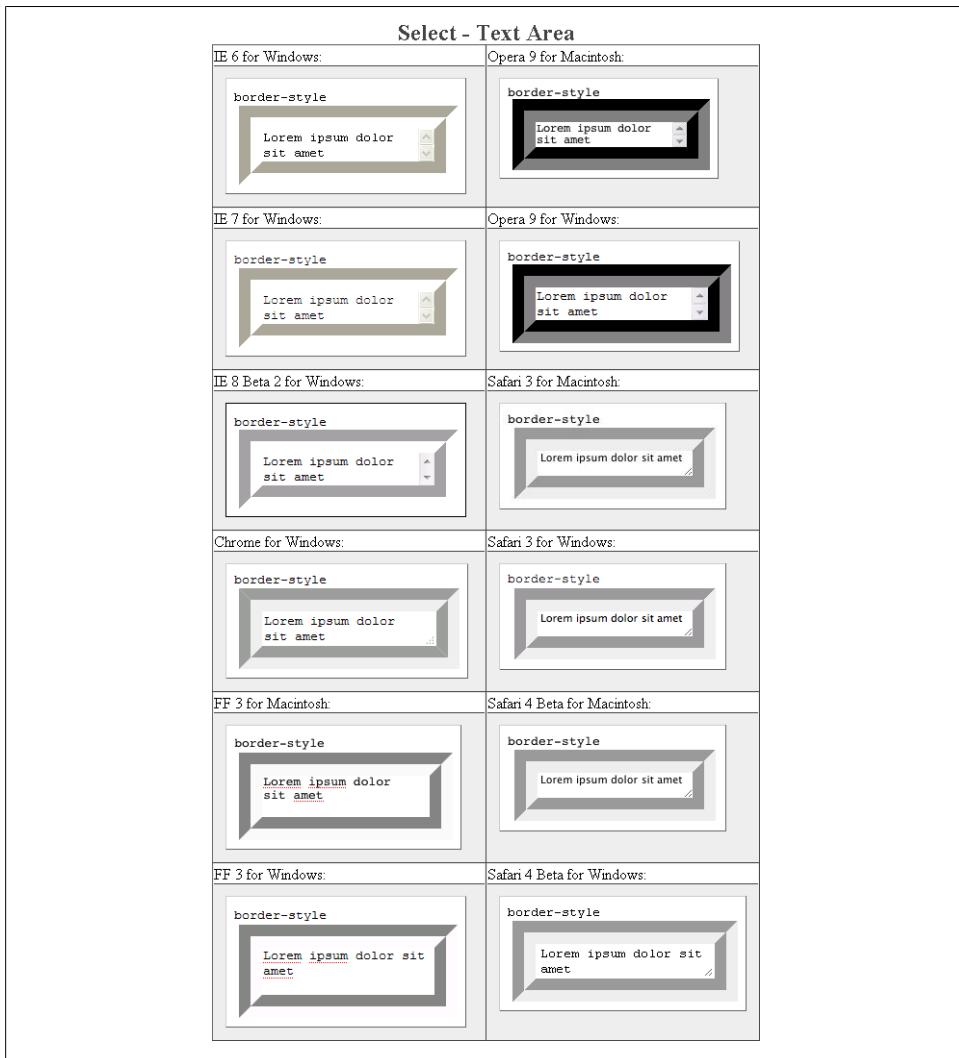
Figure E-145. Testing the styles of borders on the textarea element

border-width

Figure E-146. Testing the widths of borders on the textarea element

color
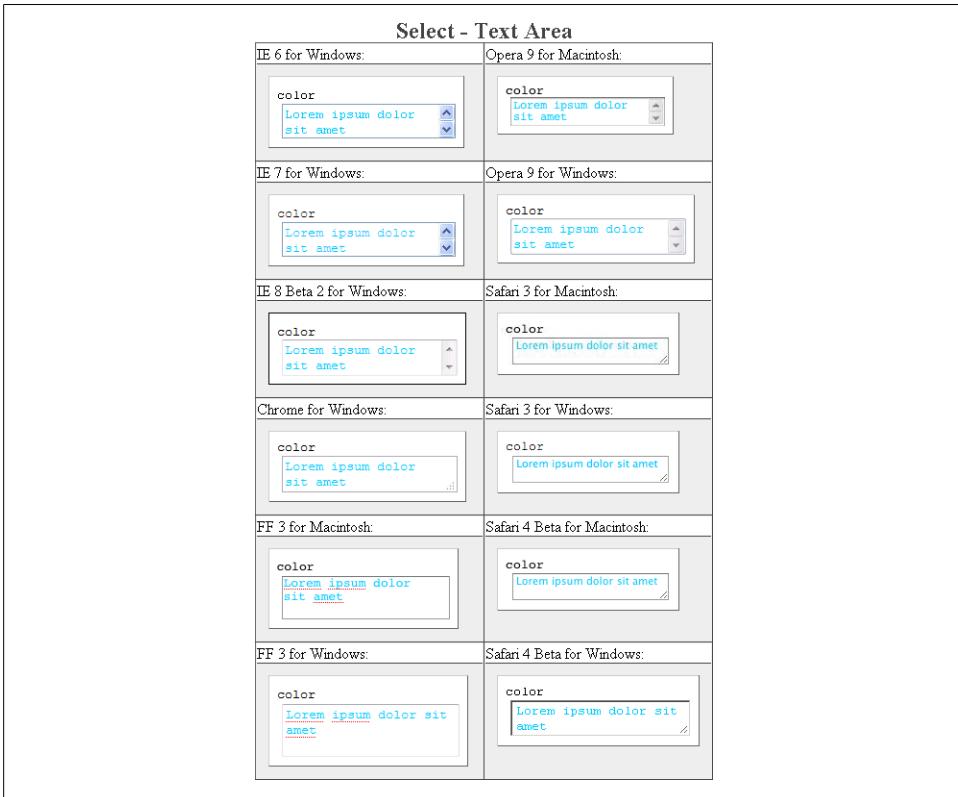
Figure E-147. Testing the color on the textarea element

font-family
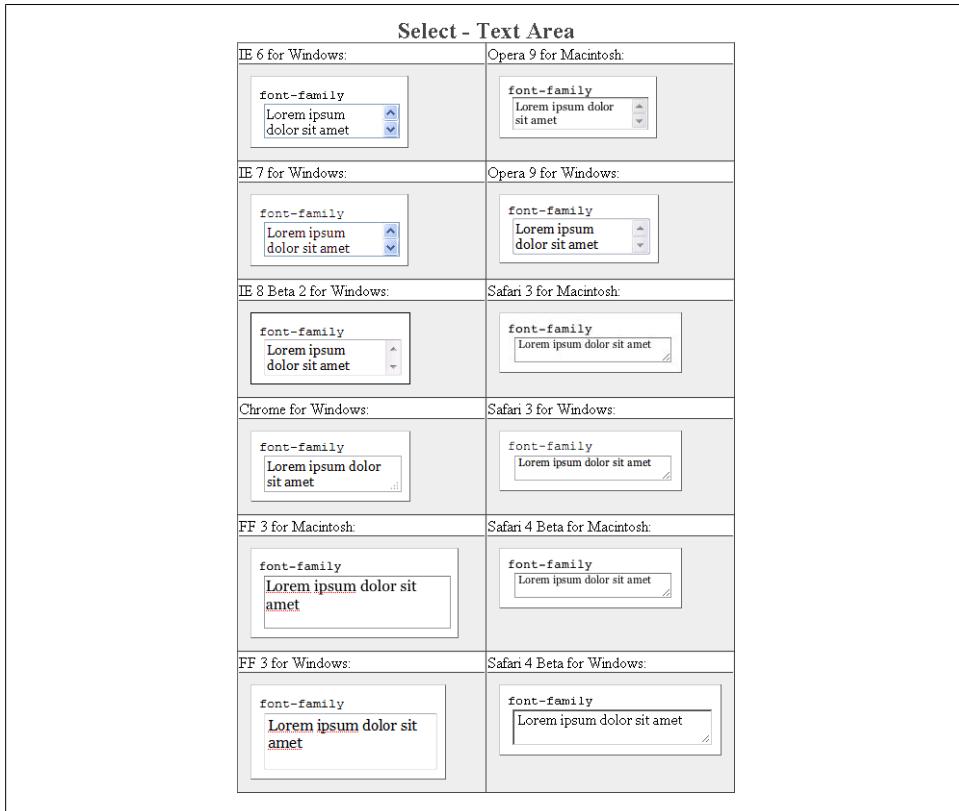
Figure E-148. Testing setting a different font on the textarea element

font-size

Figure E-149. Testing a different size of font on the textarea element

font-weight
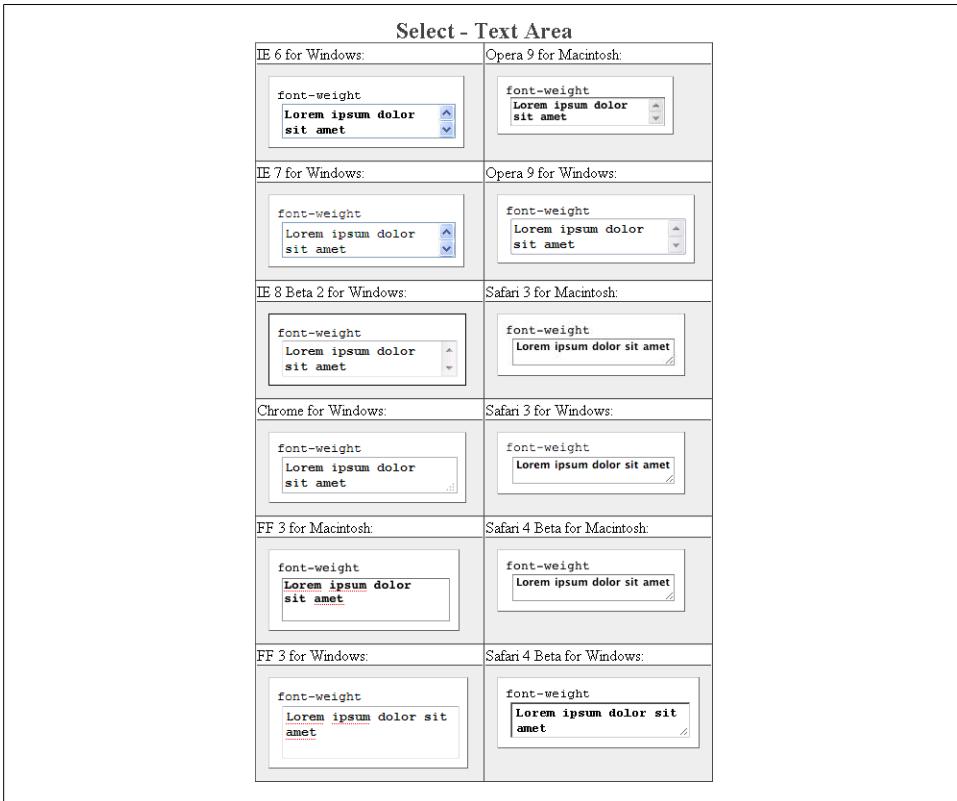
Figure E-150. Testing a bold font on the textarea element

height
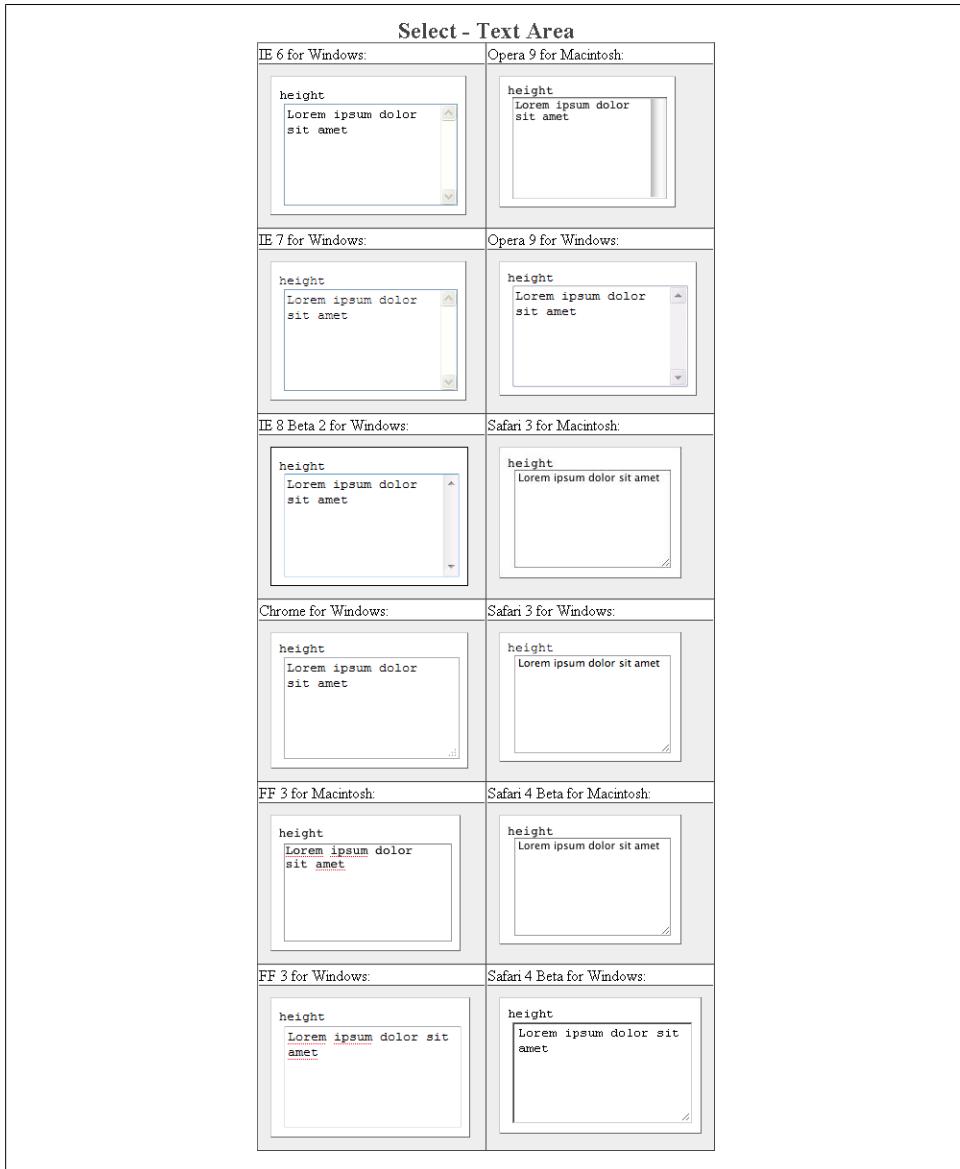
Figure E-151. Testing setting a height for the textarea element

letter-spacing
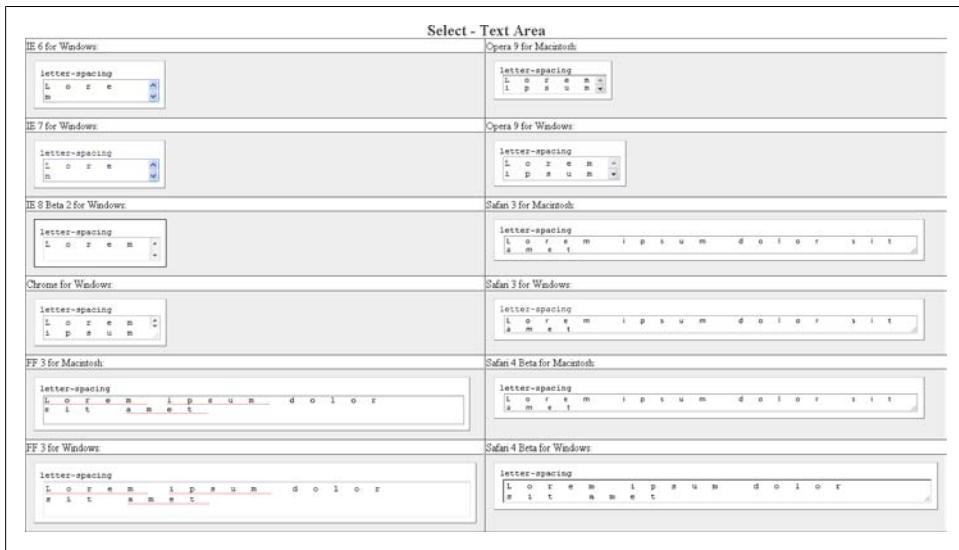
Figure E-152. Testing the letter spacing of the textarea element

line-height
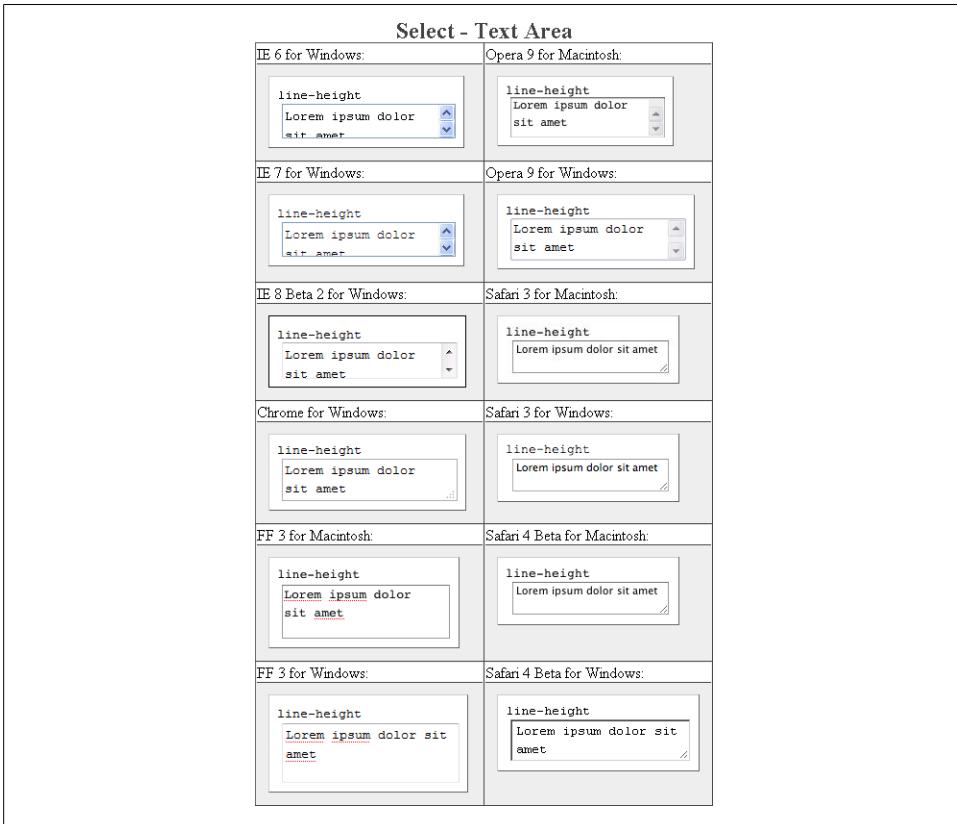
Figure E-153. Testing setting the spacing between lines of text on the textarea element

margin
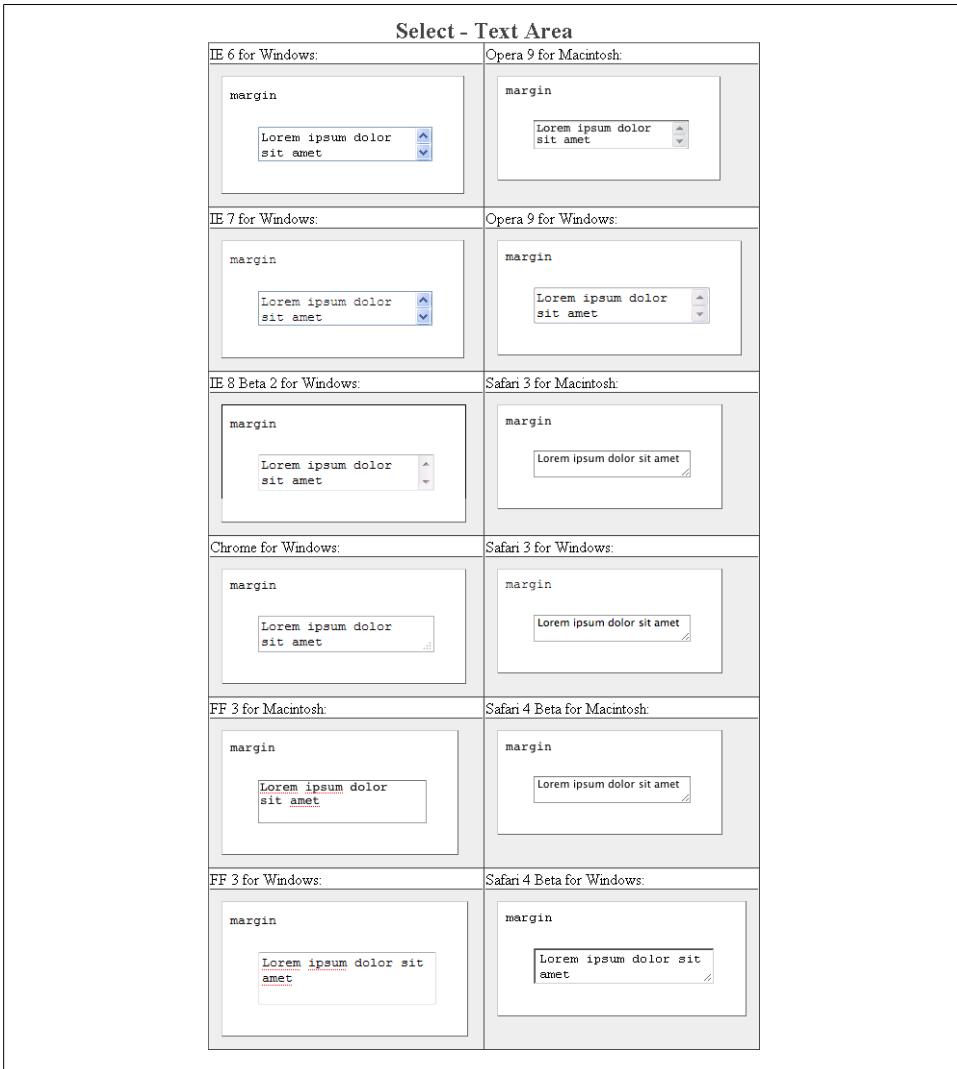
Figure E-154. Testing margins on the textarea element

padding
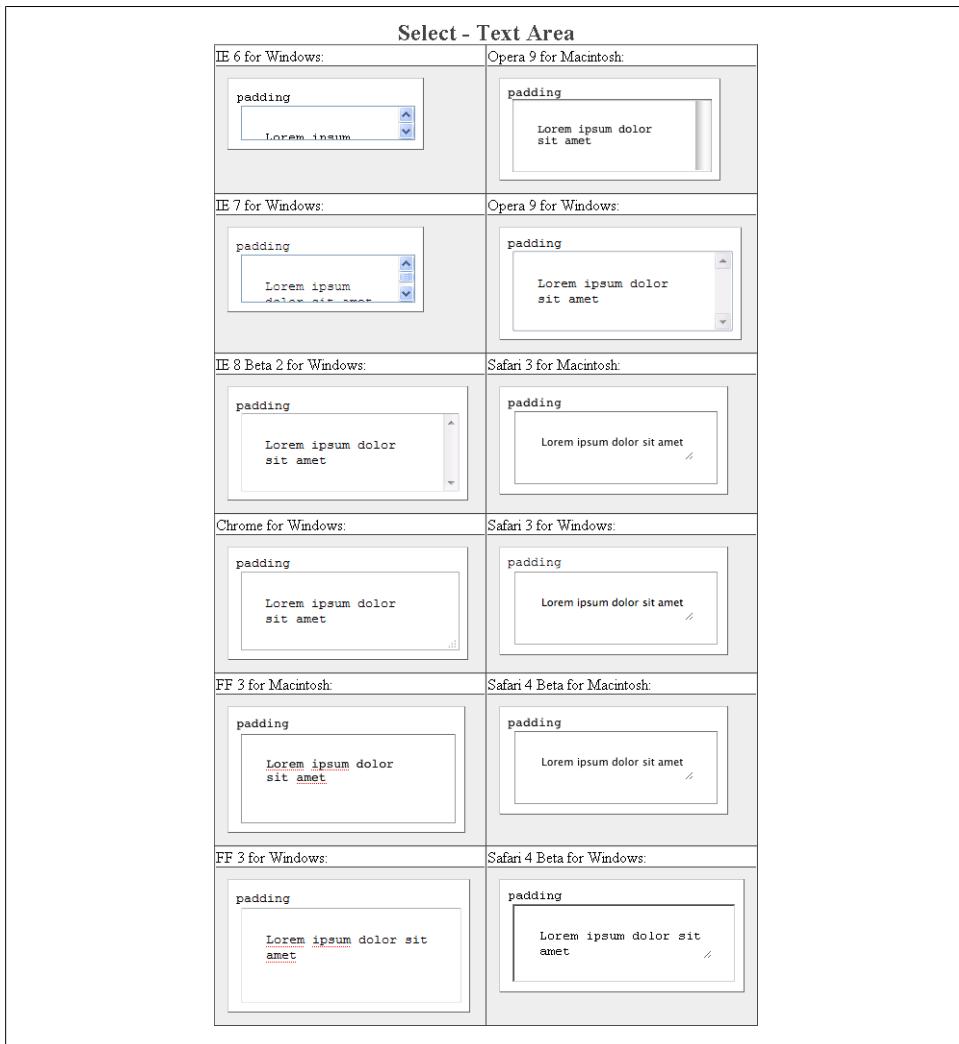
Figure E-155. Testing padding on the textarea element

text-align
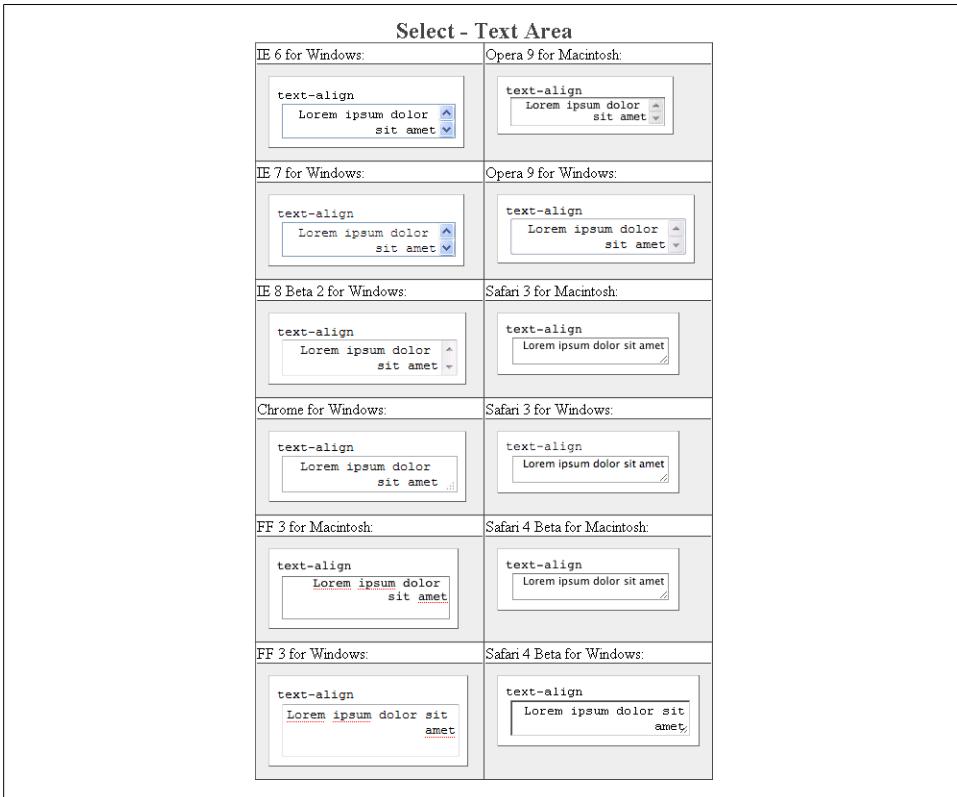
Figure E-156. Testing the alignment of text on the textarea element

text-decoration
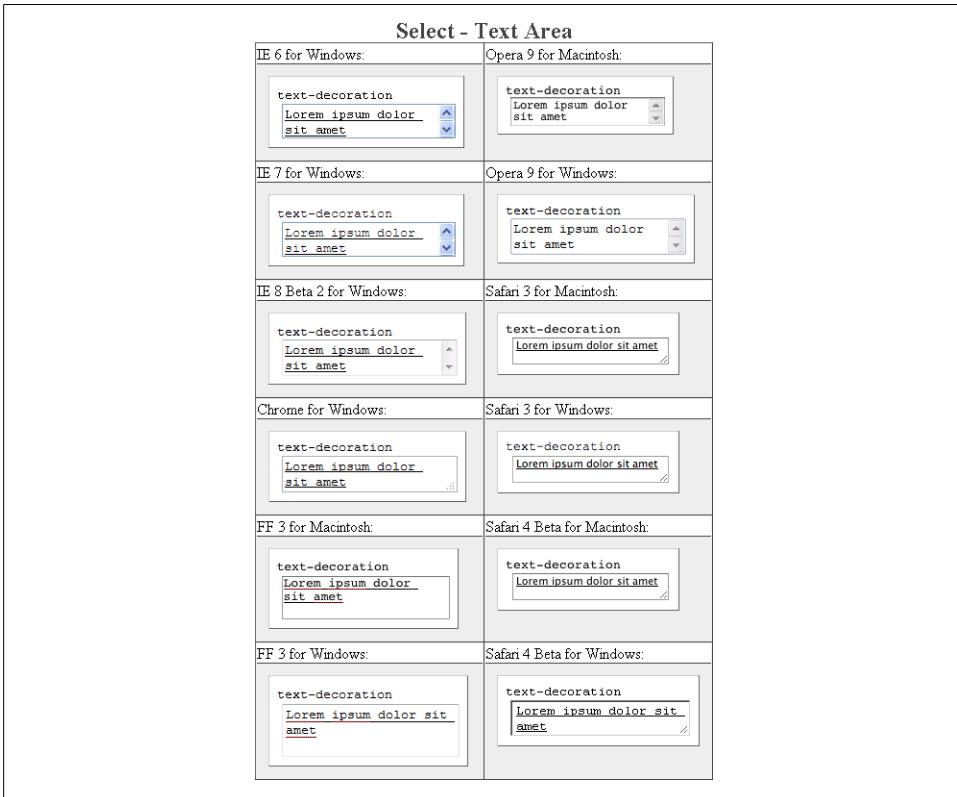
Figure E-157. Testing setting a different font on the textarea element

text-indent
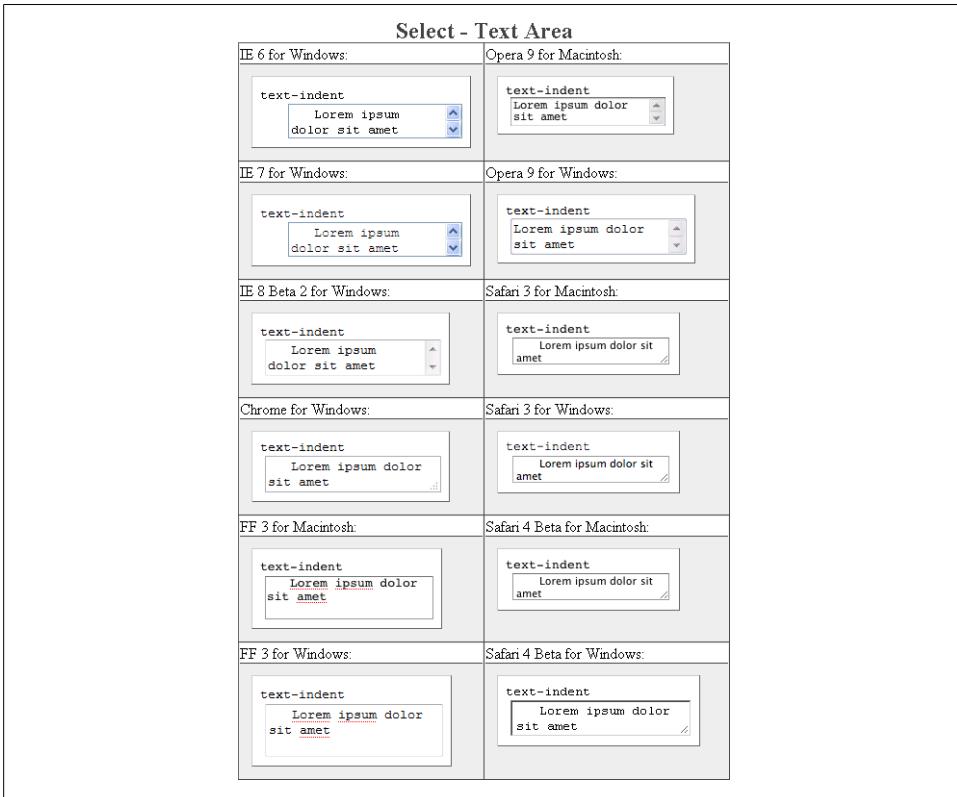
Figure E-158. Testing indenting the text on the textarea element

width
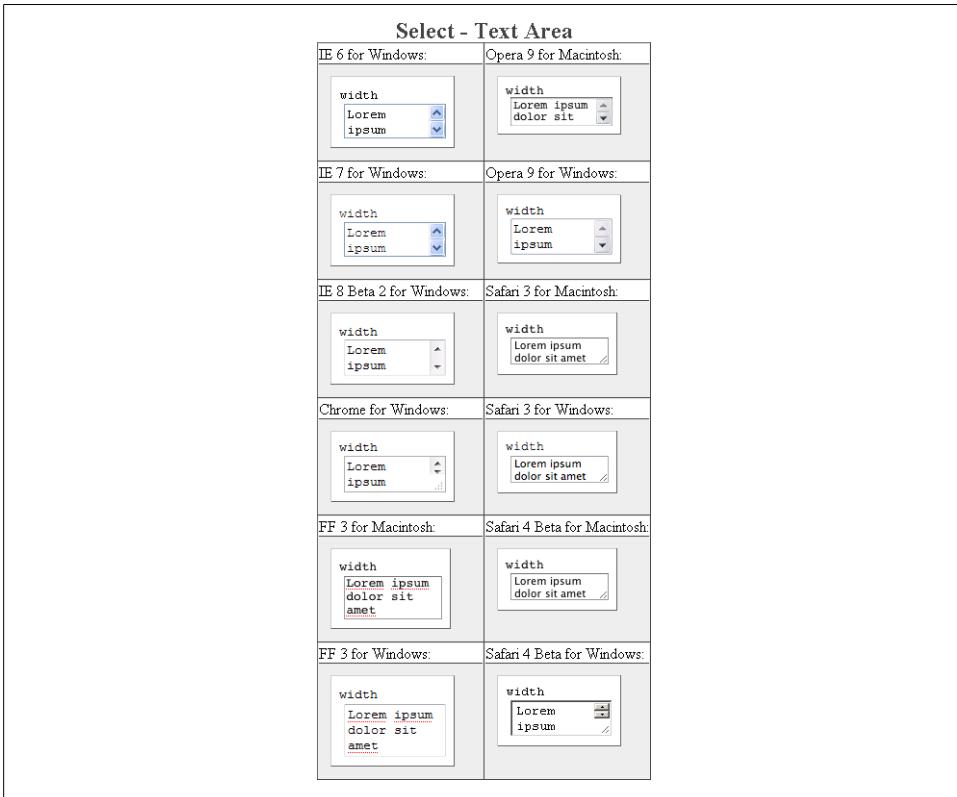
Figure E-159. Testing the width of the textarea element

word-spacing
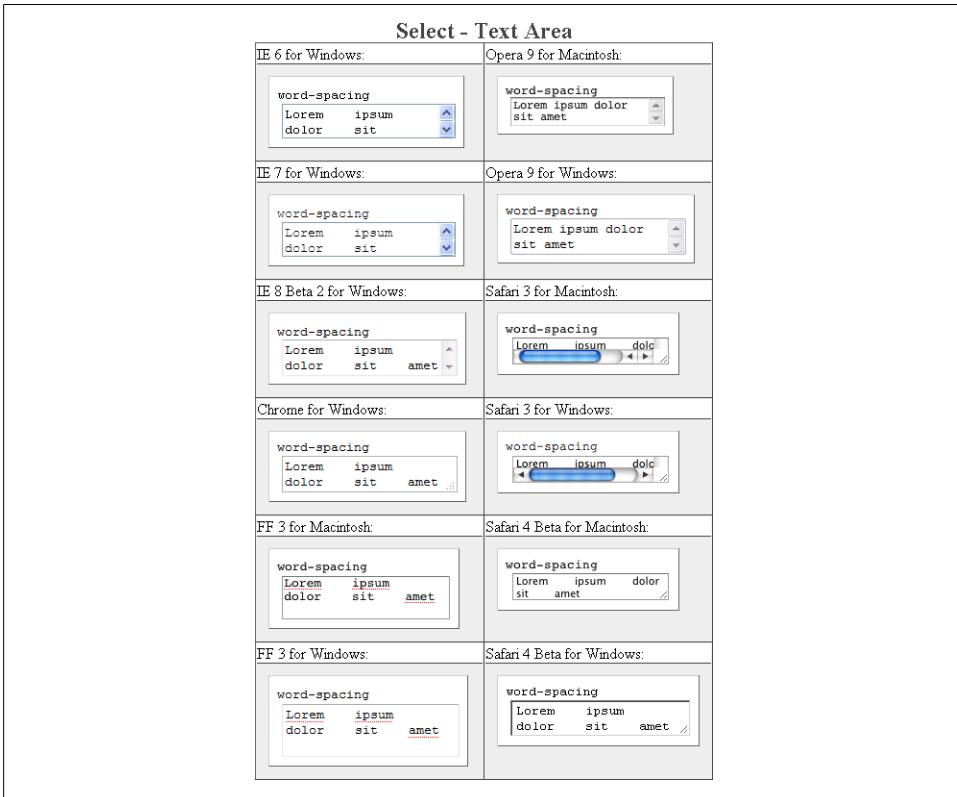
Figure E-160. Testing the spacing between words on the textarea element

Index

Symbols

& (ampersand) flourish, 112–113

< (angle bracket) for child selectors, 47

* (asterisk)
    in attribute selector declarations, 52
    for universal selectors, 45

| (bar) in attribute selector declarations, 51

{ } (braces) for property declarations, 37

[ ] (brackets) for attribute selectors, 51

^ (caret)
    in attribute selector declarations, 52
    for matching, 492

$ (dollar) in attribute selector declarations, 52

... (ellipsis) for text overflow, 128–129

(hash mark)
    for ID selectors, 44, 58
    in links, 25

. (period) for class selectors, 41, 58

.. (dot-dot) for parent folder, 24

+ (plus)
    for adjacent selectors, 49
    in collapsible menus, 383

; (semicolon) for properties lists, 37

/* */ for comments, 83–84

~ (tilde) in attribute selector declarations, 51

A

a (anchor) element, 22–25

abbreviations, returning values of, 491

absolute length units (font size), 119

absolute links, 23
    using domain names in, 491

absolute positioning, 95–97
    (see also alignment)

in columns, 521, 523, 545

access keys
    for form elements, 428–429
    in navigation menus, 374–375

Accessify's List-O-Matic menu builder, 342

accesskey attribute (a element), 374

acronyms, returning values of, 491

:active pseudo-class, 346, 353, 671
    background-image property with, 379

addClass( ) function (JavaScript), 632

adjacent selectors, 49, 152, 605

adjacent sibling selectors, 670

Adobe BrowserLab service, 565

Adobe Fireworks application, 211

Adobe Flash, drop-down menus and, 373

:after pseudo-element, 55, 327, 672
    inserting special characters with links, 492–493
    printing URIs with links, 490–492
    stylizing punctuation, 581

align attribute (hr element), 286

align attribute (img element), 92

alignment
    background images, 188–190
    centering elements on web pages, 275–280
    columns, 175–177
        fixed-width layouts, 523–524
        flexible layouts, 520–523
    definition list items, 323–328, 323–328
    forms, spacing around, 398–399
    list markers
        cross-browser consistency, 303–304
        placing markers inside list, 321–323
        using hanging indents, 319–321
    pull quotes, 143–144

We'd like to hear your suggestions for improving our indexes. Send email to *index@oreilly.com*.

superscripts and subscripts, 173
of text
baseline rhythm, applying, 171–173
centering, 126
double justification, 126–128
into multiple columns, 175–177
U.S. flag sample design, 613, 616
all media type, 83, 482
alpha transparency, 15
background colors, 294–297
CSS gradients, 210
fade effect, with PNGs, 284
masks, applying to images and borders, 262–263
with PNG images, 211–214
supporting transparent PNGs in IE6, 640–642
texture, adding to page, 619
alt attribute (img element), 15
alternate stylesheets, 88–89
alternative text for images, 15
alternative text for video, 19
& (ampersand) flourish, 112–113
an (anchor) element
accesskey attribute, 374
class attribute, 42
id attribute, 386
pseudo-classes for, 344, 346
rel attribute, 638
title attribute, 389
anchors in HTML documents, 25
< (angle bracket) for child selectors, 47
animating elements on page, 584–588
animation-direction property, 588
animation-duration property, 587
animation-iteration-count property, 587
animation-transform property, 586
Any Order Columns method, 524–544
associating styles with web pages, 70–73
* (asterisk)
in attribute selector declarations, 52
for universal selectors, 45
asymmetric page layouts, 544–547
attention, grabbing (see designing with CSS)
attribute selectors, 51–52, 670, 674
for input elements (forms), 402, 404
for placing icons with links, 350
print-readying forms with, 489
substring matching, 52

audio element, 16–17
audio, adding to HTML documents, 16–17
aural media type, 83
auto value (background-size property), 201
auto value (cursor property), 352
auto value (for margin properties), 277
autosave attribute (input element), 410

B

background-attachment property, 197, 589, 652
background-color property, 146, 160, 652
background colors
fade effect, with PNGs, 284
opacity of, 294–297
table rows, 465–468, 630–632
background gradients, 208–210
background-image property, 139, 186, 215, 378, 379, 619, 652
in :first-line pseudo-element, 158
background images
clipping, 260–261
creating columns with, 539–541
fade effect, with PNGs, 284
font inheritance and, 107
for headings, 139–141
for initial caps with decoration, 133
line of (repeating), 187–188
for list item markers, 311–313
for lists, 313–317
moving when window resizes, 582–584
multiple, on one HTML element, 191–193
panoramic presentations, 220–222
positioning, 188–190
setting, 186–187
shadow behind text, 165–168
stationary (locked against scroll), 197–199
stretching
across entire browser window, 202–203
as browser resizes, 199–202
supporting transparent PNGs in IE6, 640–642
for texture, 619
background-position property, 139, 188, 209, 340, 652
for image-based rollovers, 382
with image sprites, 258
background-position-x property, 662
background-position-y property, 662

using multiple in same headline, 575
web-safe, 109–112
footer, adding to pages, 502, 606
with multicolumn layouts, 508
forcing breaks on long words, 118–119
form element, print-ready, 487
formats, audio, 17
formats, images, 15
combining different, 222–227
forms, 397–452
access keys, stylizing, 428–429
buttons on
images for, 415–416
readying for print, 487, 489
stylizing, 411–415
Submit buttons that look like text, 417–419
submit-once-only buttons, 416–417
text links that operate like, 419
designing like spreadsheet tables, 431–434
designing without tables, 419–425
grouping common elements, 429–430
integrated with feedback, 425–427
login form (sample design), 434–440
Macintosh-styled search field, 408–411
print-ready, 486–490
registration form (sample design), 441–452
select and option elements, stylizing, 406–408
spacing around, modifying, 398–399
style changes when user clients, 402–403
styles for input elements, 399–402
different styles for different elements, 403–404
stylizing textarea elements, 404–406
fragment identifiers, 392
frame elements, for stretching images, 202–203
frames, visual (see borders)
framesets, for stretching images, 203
full stop, 41

G

general sibling combinator selectors, 494, 673
getElementbyId method (JavaScript), 385
GIF file format, 15
combining different image formats, 222
hindering theft of your images, 254
percentage-based dimensions and, 204

Gilder/Levin image replacement technique, 217
Google's hosting feature, 627
gradients in element backgrounds, 208–210
graphics (see images)
grayscale, transforming images to, 179–180, 484–486, 495
grids, designing with, 591–593
groove effect (framing), 281
grouping CSS rules in stylesheets, 84–86
grouping form elements, 429–430
grouping selectors, 153, 670
gutters (between columns), 535, 592

H

h1 through h6 elements, 10–12
hacks, separating from correct rules, 562–563
handheld media type, 83, 482
hanging indents
creating, 153–156
in lists, 319–321
paired hanging indent, 155
(hash mark)
for ID selectors, 44, 58
in links, 25
hasLayout element, 296
hCalendar Creator utility, 29
hCalendar microformat, 28–29
hCard Creator utility, 28
hCards (HTML vCards), 27–28, 606
head element, 5
header, table, 26, 460–462
headings, 10–12
ampersand flourish in, 112–113
paired hanging indent, 155
red flame on, 167
reflecting in images, 256–258
replacing with images, 217–220
shadows behind, 165–168
space between paragraphs and, 129–130
stylized borders for, 137–139
stylized images for, 139–141
with stylized text, 135–141
hebrew (counting style), 301
height (see size)
height property, 656
vertical centering on pages and, 279
help value (cursor property), 352

overriding, 552
precedence of, 74
when to use, 72
input element
access keys for, stylizing, 428–429
as block-level element, 420
grouping common, 429–430
print-ready, 487
required, identifying, 425–427
spreadsheet-like forms, 431–434
styles for, 399–402
different styles for different input
elements, 403–404
inset effect (framing), 281
Inspect Element toolbar, 555
internal stylesheets, 70, 71
syntax for, 37
when to use, 72
Internet Explorer 6, patching, 557–559, 609
Internet Explorer Developer Toolbar, 555
Internet Explorer scroll bar, coloring, 272–275
Internet Explorer's conditional comments,
559–561, 609
:invalid pseudo-class, 53
invisible text, 120
italics (see emphasizing certain words)

J

JavaScript, 623–644
adding jQuery framework to pages, 627–
628
availability of, checking for, 623–624
CSS3 and HTML5 to appropriate browsers,
642–644
CSS3 selectors in IE6 and IE7, 628–630
equalizing height for row of elements, 635–
637
highlight table rows with mouseovers, 632–
634
making page elements clickable, 639–640
opening new windows with links, 638–639
rounding corners with, 239–242
screen-width-specific stylesheets, 626–627
simple rollover effects, 634–635
for time-specific stylesheets, 625–626
for transparent PNGs with IE6, 640–642
zebra-striping tables, 630–632
JPEG file format, 15
combining different image formats, 222

percentage-based dimensions and, 204
jQuery framework
associating with pages, 627–628
chaining JavaScript functions, 634
equalizing height for row of elements, 635
plug-ins for, finding, 642
use of CSS selectors, 629
zebra-striping tables, 631

K

katakana (counting style), 301
kerning (typography), 170
keyframes for animation, 585
Knoppix operating system, 564

L

label element
access keys, stylizing, 428–429
as block-level element, 420
identifying required elements, 425–427
web forms without tables, 419–425
label property (web forms), 420
:lang pseudo-class, 53, 671
large centered initial caps (paragraphs), 131–
133
large keyword (font-size property), 123
larger keyword (font-size property), 123
:last-child pseudo-class, 53, 675
:last-of-type( ) pseudo-class, 53, 335, 676
layering
background images, 191–193
text on images, 215–217
layout-grid-char property, 662
layout-grid-line property, 662
layout-grid-mode property, 662
layout-grid property, 662
layout-grid-type property, 662
leading (typography), 163
Leahy-Langridge Image Replacement (LIR)
method, 381
left property, 656
with absolute positioning, 97
elements in multicolumn layouts, 522, 523,
545
with relative positioning, 98
with shackling positioning, 100
left value (background-position property), 189
legend element, 430

M

Macintosh-styled search field, 408–411
mailing address, adding to print forms, 489
margin-bottom property, 657
 drop shadows behind images, 245
margin-left property, 657
 auto value for, 277
 for list items, 303, 304, 324
 percentage values for, 505
margin property, 67, 70, 88, 657
 form buttons, 411
 forms, 398
 for html and body elements, 266
margin-right property, 657
 auto value for, 277
 drop shadows behind images, 245
 percentage values for, 505
margin-top property, 657
margins
 box model, 62–70
 with columns
 around elements within columns, 517,
  535
 using grids (CSS Frameworks), 592
 one-column layouts, 505
 defined, 63
 font inheritance and, 107
 around forms, 398–399
 for headings and paragraphs, 129
 list indentation, 303–304, 319–321
 of pages, eliminating, 265–267
masks, applying to images and borders, 262–
  263
max-height property, 657
max-width property, 204, 221, 657
measurements, font, 119–120
media attribute (link element), 79, 482, 549
@media rule, 79, 483
media-specific stylesheets, 79–83
media types, 482
 different stylesheets for, 79–83
menus for navigation, 341–343
 with access keys, 374–375
 collapsible, 383–386
 contextual, 386–388
 dynamic tabbed, 389–392
 horizontal, 365–371, 599
 with drop-down submenus, 372–374
 print-ready, 487

 rollovers in, 358–362, 597
 vertical, submenus in, 363–365
microformats
 hCalendar events, 28–29
 hCards (HTML vCards), 27–28, 606
Microsoft proprietary extensions (complete
  list), 661–663
Microsoft's Image Toolbar, 255
min-height property, 657
min-width property, 658
mobile-friendly spreadsheets, 626–627
Modernizr JavaScript library, 643
monospace (font family), 106
 web-safe fonts, 109
motion, impression of, 576–577
Mountaintop technique (rounding corners),
  235–238
mouseover( ) and mouseout( ) functions, 633
move value (cursor property), 352
moving background scenes, 582–584
-mox-image-rect property, 260
-moz-appearance property, 664
-moz-background-clip property, 664
-moz-background-image property, 664
-moz-background-inline-policy property, 664
-moz-background-origin property, 664
-moz-binding property, 664
-moz-border-bottom-colors property, 665
-moz-border-left-colors property, 665
-moz-border-radius properties, 183, 665
-moz-border-right-colors property, 665
-moz-border-top-colors property, 665
-moz-box-align property, 665
-moz-box-direction property, 665
-moz-box-flex property, 665
-moz-box-orient property, 665
-moz-box-pack property, 665
-moz-box-shadow property, 666
-moz-box-sizing property, 666
-moz-image-region property, 666
-moz-opacity property, 666
-moz-outline-color property, 666
-moz-outline-input property, 667
-moz-outline-offset property, 666
-moz-outline property, 666
-moz-outline-radius-bottomleft property, 666
-moz-outline-radius-bottomright property,
  666
-moz-outline-radius property, 666

default style for, changing, 299–302
dividers between list items, 304–305
for drop-down navigation, 372
hanging indents in, 319–321
for horizontal navigation menus, 365–371, 599
indentation in, 303–304
as inline lists, 318–319
for navigation, 362
nested, descendant selectors with, 364
placing markers inside list, 321–323
star ranking systems, creating, 335–340
upper-alpha (counting style), 301
upper-latin (counting style), 301
upper-roman (counting style), 301
uppercase (see capitalization)
URIs and URLs (see links)
url( ) function, 196

V

valid HTML pages, understanding, 6–10
:valid pseudo-class, 53
validating CSS rules, 102–103
validating HTML on page, 29–31, 30, 102, 554, 649
validator, HTML, 29–31, 30, 102, 554, 649
value attribute (input element), 409
vCards, 28 (see hCards)
versals, 131
vertical-align property, 464, 660
in :first-line pseudo-element, 158
vertical centering on pages, 278
vertical navigation menus
with access menus, 374–375
collapsible, 383–386
contextual, 386–388
dynamic tabbed, 389–392
submenus in, 363–365
vevent class, 29
vi and vim editors, 4
video element, 17–19
video, adding to HTML documents, 17–19
viewport, browser
border around, 283–284
coloring scroll bar (IE), 272–275
viewsource.css stylesheet, 590
visibility property, 120, 660
:visited pseudo-class, 344, 346, 671
VMware Workstation, 564

W

w-resize value (cursor property), 352
W3C CSS Validator, 29–31, 102, 554, 649
W3C HTML Validator, 30, 554, 649
wait value (cursor property), 352
web browser consistency (see cross-browser consistency)
Web Developer browser extension, 650
web pages
associating styles with, 70–73
borders for, 280–282
elements of (see page elements; specific element by name)
web-safe fonts, 109–112
-webkit-border properties, 183
-webkit-gradient property, 208
-webkit-transform property, 207
-webkit-transition- properties, 354
white-space property, 660
whitespace (see space)
whitespace in CSS programming, 37
widows property, 660
width (see size)
width attribute (hr element), 286
width property, 146, 660
for definition list items, 324
form buttons, 415
percentage values for, 203
windows (see browsers)
Wine software, 565
word balloons, 251–253
word-break property, 663
word-spacing property, 169, 660
in :first-line pseudo-element, 158
word wrap (forcing breaks), 118–119
word-wrap property, 118, 663
words and characters
acronyms, returning values of, 491
emphasizing (see emphasizing certain words)
excessively large, 570–571
forcing breaks on words, 118–119
in general (see text)
highlighting effect for, 159–160
impression of depth or motion, 576–577
inserting characters before links (in print), 492–493
multiple typefaces in one line, 575
red flame on, 167

About the Author

Christopher Schmitt is the founder of Heat Vision, a small new media publishing and design firm, based in Cincinnati, Ohio. An award-winning web designer who has been working with the Web since 1993, Christopher interned for both David Siegel and Lynda Weinman in the mid '90s while he was an undergraduate at Florida State University working on a fine arts degree with an emphasis on graphic design. Afterward, he earned a master's in communication for interactive and new communication technologies while obtaining a graduate certificate in project management from FSU's College of Communication.

He is the author of one of the first books that looked at CSS-enabled designs, *Designing CSS Web Pages* (New Riders). He is also the coauthor of *Adapting to Web Standards* (New Riders), *Professional CSS* (Wrox), *Photoshop in 10 Steps or Less* (Wiley) and *Dreamweaver MX Design Projects* (Wrox), and contributed four chapters to *XML, HTML, and XHTML Magic* (New Riders). Christopher has also written for *New Architect Magazine*, *A List Apart*, *Digital Web*, and *Web Reference*.

He is the list moderator for Babble, a mailing list community devoted to advanced web design and development topics. With the Web Standards Project, Christopher helps co-lead the Adobe Task Force while contributing to its Education Task Force.

On his personal website, Christopher shows his true colors and most recent activities. He is 6'7" and doesn't play professional basketball, but wouldn't mind a good game of chess.

Colophon

The animal on the cover of *CSS Cookbook* is a grizzly bear (*Ursus arctos horribilis*). The grizzly's distinctive features include humped shoulders, a long snout, and long curved claws. The coat color ranges from shades of blond, brown, black, or a combination of these; the long outer guard hairs are often tipped with white or silver, giving the bear a "grizzled" appearance. The grizzly can weigh anywhere from 350 to 800 pounds and reach a shoulder height of 4.5 feet when on all fours. Standing on its hind legs, a grizzly can reach up to eight feet. Despite its large size, the grizzly can reach speeds of 35 to 40 miles per hour.

Some of the grizzly's favorite foods include nuts, berries, insects, salmon, carrion, and small mammals. The diet of a grizzly varies depending on the season and habitat. Grizzlies in parts of Alaska eat primarily salmon, while grizzlies in high mountain areas eat mostly berries and insects.

Grizzlies are solitary, and prefer rugged mountains and forests. They can be found in the Canadian provinces of British Columbia, Alberta, Yukon, and the Northwest Territories, and in the U.S. states of Alaska, Idaho, Wyoming, Washington, and Montana.

The grizzly is considered a threatened species: only about 850 bears exist in the lower 48 states. Before the West was settled, the grizzly bear population was estimated to be between 50,000 and 100,000. Threats to the survival of the grizzly bear include habitat destruction caused by logging, mining, and human development, as well as illegal poaching.

The cover image is a 19th-century engraving from the Dover Pictorial Archive. The cover font is Adobe ITC Garamond. The text font is Linotype Birka; the heading font is Adobe Myriad Condensed; and the code font is LucasFont's TheSansMonoCondensed.