*Quick Solutions to Common CSS Problems*

3rd Edition
Updated for Firefox 3, IE 8, and Chrome

Free Sampler

# CSS Cookbook

*Christopher Schmitt*
*Foreword by Dan Cederholm*

O'REILLY®

## 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*

# O'Reilly Ebooks—Your bookshelf on your devices!


When you buy an ebook through oreilly.com, you get lifetime access to the book, and whenever possible we provide it to you in four, DRM-free file formats—PDF, .epub, Kindle-compatible .mobi, and Android .apk ebook—that you can use on the devices of your choice. Our ebook files are fully searchable and you can cut-and-paste and print them. We also alert you when we've updated the files with corrections and additions.

**Learn more at http://oreilly.com/ebooks/**

You can also purchase O'Reilly ebooks through iTunes, the Android Marketplace, and Amazon.com.

# CSS Cookbook

THIRD EDITION

# CSS Cookbook

*Christopher Schmitt*
*foreword by Dan Cederholm*

O'REILLY®
Beijing • Cambridge • Farnham • Köln • Sebastopol • Taipei • Tokyo

**CSS Cookbook, Third Edition**
by Christopher Schmitt



Published by O'Reilly Media, Inc., 1005 Gravenstein Highway North, Sebastopol, CA 95472.

O'Reilly books may be purchased for educational, business, or sales promotional use. Online editions are also available for most titles (*http://my.safaribooksonline.com*). For more information, contact our corporate/institutional sales department: (800) 998-9938 or *corporate@oreilly.com*.

**Editor:** Simon St.Laurent
**Production Editor:** Sumita Mukherji
**Copyeditor:** Audrey Doyle
**Proofreader:** Kiel Van Horn

**Indexer:** Seth Maislin
**Cover Designer:** Karen Montgomery
**Interior Designer:** David Futato
**Illustrator:** Robert Romano

**Printing History:**

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



ISBN: 978-0-596-15593-3

[SB]

1260562909

# Table of Contents

**Foreword** ..... **xv**

**Preface** ..... **xvii**

**1. Using HTML Basics** ..... **1**
- 1.1 Picking a Text Editor 3
- 1.2 Coding a Basic HTML Page 4
- 1.3 Understanding DOCTYPEs and Effects on Browser Layout 6
- 1.4 Marking Up Headers 10
- 1.5 Making Appropriate Quotations 12
- 1.6 Adding an Image 14
- 1.7 Adding Audio with HTML5 16
- 1.8 Incorporating Video with HTML5 17
- 1.9 Using strong and em Effectively 19
- 1.10 Creating Lists 20
- 1.11 Making a Link to a Web Page 22
- 1.12 Coding Tables 25
- 1.13 Creating an HTML vCard (hCard) 27
- 1.14 Marking Up an Event (hCalendar) 28
- 1.15 Validating HTML 29

**2. CSS Basics** ..... **33**
- 2.1 Applying CSS Rules to a Web Page 35
- 2.2 Using Basic Selectors to Apply Styles 38
- 2.3 Applying Child Selectors 47
- 2.4 Applying Adjacent Selectors 49
- 2.5 Applying Attribute Selectors 51
- 2.6 Using Pseudo-Classes 53
- 2.7 Using Pseudo-Elements 54
- 2.8 Determining When to Use Class and ID Selectors 56
- 2.9 Understanding CSS Properties 61

2.10 Understanding the Box Model 62
2.11 Associating Styles to a Web Page 70
2.12 Understanding the Origin 73
2.13 Understanding the Sort Order Within CSS 73
2.14 Using !important to Override Certain CSS Rules 76
2.15 Clarifying Specificity 77
2.16 Setting Up Different Types of Stylesheets 79
2.17 Adding Comments Within Stylesheets 83
2.18 Organizing the Contents of a Stylesheet 84
2.19 Working with Shorthand Properties 86
2.20 Setting Up an Alternate Stylesheet 88
2.21 Using Floats 89
2.22 Using Self-Clearing Floated Elements 92
2.23 Using Absolute Positioning 95
2.24 Using Relative Positioning 98
2.25 Using Shackling Positioning 99
2.26 Stacking Elements with z-index 101
2.27 Validating CSS Rules 102

**3. Web Typography ..... 105**
3.1 Specifying Fonts 106
3.2 Using Web-Safe Fonts 109
3.3 Setting an Ampersand Flourish 112
3.4 Embedding Font Files 114
3.5 Forcing a Break on Really Long Words 118
3.6 Specifying Font Measurements and Sizes 119
3.7 Gaining More Cross-Browser Consistency with Font Sizes 121
3.8 Setting Hyphens, Em Dashes, and En Dashes 125
3.9 Centering Text 126
3.10 Setting Text to Be Justified 126
3.11 Indicating an Overflow of Text with an Ellipsis 128
3.12 Removing Space Between Headings and Paragraphs 129
3.13 Setting a Simple Initial Cap 130
3.14 Setting a Larger, Centered Initial Cap 131
3.15 Setting an Initial Cap with Decoration (Imagery) 133
3.16 Creating a Heading with Stylized Text 135
3.17 Creating a Heading with Stylized Text and Borders 137
3.18 Stylizing a Heading with Text and an Image 139
3.19 Creating a Pull Quote with HTML Text 141
3.20 Placing a Pull Quote to the Side of a Column 143
3.21 Creating a Pull Quote with Borders 145
3.22 Creating a Pull Quote with Images 146
3.23 Setting the Indent in the First Line of a Paragraph 149

3.24 Setting the Indent of Entire Paragraphs 150
3.25 Creating a Hanging Indent 153
3.26 Styling the First Line of a Paragraph 156
3.27 Styling the First Line of a Paragraph with an Image 158
3.28 Creating a Highlighted Text Effect 159
3.29 Changing the Text Selection Color 160
3.30 Changing Line Spacing 161
3.31 Adding a Graphic Treatment to HTML Text 163
3.32 Placing a Shadow Behind Text 165
3.33 Adjusting the Space Between Letters and Words 168
3.34 Applying Baseline Rhythm on Web Typography 171
3.35 Styling Superscripts and Subscripts Without Messing the Text Baseline 173
3.36 Setting Up Multiple Columns of Text 175

**4. Images ..... 179**
4.1 Transforming Color Images to Black and White in IE with CSS 179
4.2 Setting a Border Around an Image 180
4.3 Setting a Rounded Border Around an Image 182
4.4 Removing Borders Set on Images by Default in Some Browsers 184
4.5 Setting a Background Image 186
4.6 Creating a Line of Background Images 187
4.7 Positioning a Background Image 188
4.8 Using Multiple Background Images on One HTML Element 191
4.9 Setting Images on a Border 194
4.10 Creating a Stationary Background Image 197
4.11 Stretching Images As the Browser Resizes 199
4.12 Stretching an Image Across the Entire Browser Window 202
4.13 Making Images Scalable 203
4.14 Setting How a Browser Renders an Image 205
4.15 Rotating Images with CSS 206
4.16 Setting Gradients with CSS 208
4.17 Creating Transparent PNG Images for IE6 and Later 211
4.18 Using Transparent PNG Images with JavaScript 212
4.19 Overlaying HTML Text on an Image 215
4.20 Replacing HTML Text with an Image 217
4.21 Building a Panoramic Image Presentation 220
4.22 Combining Different Image Formats 222
4.23 Rounding Corners with Fixed-Width Columns 227
4.24 Rounding Corners (Sliding Doors Technique) 230
4.25 Rounding Corners (Mountaintop Technique) 235
4.26 Rounding Corners with JavaScript 239
4.27 Setting a Shadow on an Element with CSS 242

4.28 Placing a Drop Shadow Behind an Image 244
4.29 Placing a Smooth Drop Shadow Behind an Image 247
4.30 Making Word Balloons 251
4.31 Hindering People from Stealing Your Images 254
4.32 Inserting Reflections on Images Automatically 256
4.33 Using Image Sprites 258
4.34 Clipping Background Images 260
4.35 Applying Masks to Images and Borders 262

**5. Page Elements ..... 265**
5.1 Eliminating Page Margins 265
5.2 Resetting Browser-Style Defaults for Elements 268
5.3 Coloring the Scroll Bar in IE 272
5.4 Techniques for Centering Elements on a Web Page 275
5.5 Placing a Page Border 280
5.6 Placing a Border Around the Browser's Viewport 283
5.7 Customizing a Horizontal Rule 285
5.8 Adding a Lightbox 287
5.9 Changing the Opacity on Elements 292
5.10 Adjusting the Opacity of Background Colors 294

**6. Lists ..... 299**
6.1 Changing the Format of a List 299
6.2 Changing the Color of a List Bullet 302
6.3 Writing Cross-Browser Indentation in Lists 303
6.4 Placing Dividers Between List Items 304
6.5 Creating Custom Text Markers for Lists 306
6.6 Creating Custom Image Markers for Lists 308
6.7 Inserting Larger Custom Image Markers for Lists 311
6.8 Making a List Presentation Rich with Imagery 313
6.9 Creating Inline Lists 318
6.10 Making Hanging Indents in a List 319
6.11 Moving the Marker Inside the List 321
6.12 Styling a Definition List 323
6.13 Styling a Screenplay with the HTML5 dialog Element 329
6.14 Turning a List into a Directory Tree 331
6.15 Creating a Star Ranking System 335

**7. Links and Navigation ..... 341**
7.1 Easily Generating Text-Based Menus and Submenus 341
7.2 Removing Underlines from Links (and Adding Other Styles) 343
7.3 Changing Link Colors 346

7.4 Removing Dotted Lines When Clicking on a Link in Internet Explorer 347
7.5 Changing Link Colors in Different Sections of a Page 348
7.6 Placing Icons at the End of Different Kinds of Links 349
7.7 Changing Cursors 351
7.8 Creating Rollovers Without JavaScript 353
7.9 Animating Rollovers on Links with CSS3 Transitions 354
7.10 Creating Text Navigation Menus and Rollovers 358
7.11 Adding Submenus to Vertical Menus 363
7.12 Building Horizontal Navigation Menus 365
7.13 Building Horizontal Navigation Menus with Drop-Down Menus 372
7.14 Building a Navigation Menu with Access Keys 374
7.15 Creating Breadcrumb Navigation 375
7.16 Creating Image-Based Rollovers 379
7.17 Creating Collapsible Menus 383
7.18 Creating Contextual Menus 386
7.19 Making Tool Tips with the title Attribute 389
7.20 Designing a Dynamic Tabbed Menu 389
7.21 Changing Styles on Anchored Links 392

**8. Forms ..... 397**
8.1 Modifying the Spacing Around a Form 398
8.2 Removing the Space Around a Form 399
8.3 Setting Styles for Input Elements 399
8.4 Changing Styles on Form Elements When a User Clicks on Them 402
8.5 Applying Different Styles to Different Input Elements in the Same Form 403
8.6 Setting Styles for textarea Elements 404
8.7 Setting Styles for select and option Elements 406
8.8 Creating a Macintosh-Styled Search Field 408
8.9 Styling Form Buttons 411
8.10 Creating an Image Submit Button 415
8.11 Setting Up a Submit-Once-Only Button 416
8.12 Creating a Submit Button That Looks Like HTML Text 417
8.13 Making an HTML Text Link Operate Like a Submit Button 419
8.14 Designing a Web Form Without Tables 419
8.15 Designing a Two-Column Form Without Tables 422
8.16 Integrating Form Feedback with a Form 425
8.17 Styling Access Keys in Web Forms 428
8.18 Grouping Common Form Elements 429
8.19 Entering Data into a Form That Is Similar to a Spreadsheet 431
8.20 Sample Design: A Login Form 434
8.21 Sample Design: A Registration Form 441

**9. Tables ..... 453**
9.1 Setting the Borders and Cell Padding for Tables 453
9.2 Setting the Cell Spacing 456
9.3 Setting the Style for Captions 457
9.4 Setting the Styles Within Table Cells 458
9.5 Setting the Styles for Table Header Elements 460
9.6 Removing Gaps from Images Placed in Table Cells 462
9.7 Eliminating Gaps Between Table Cells 464
9.8 Creating Alternating Background Colors in Table Rows 465
9.9 Adding a Highlighting Effect on a Table Row 468
9.10 Sample Design: An Elegant Calendar 470

**10. Designing Web Pages for Printing ..... 481**
10.1 Applying a Stylesheet for Printing to a Web Page 481
10.2 Replacing a Color Logo for a Black-and-White Logo When Printing Web Pages 484
10.3 Making a Web Form Print-Ready 486
10.4 Displaying URIs After Links 490
10.5 Inserting Special Characters Before Links 492
10.6 Setting Page Breaks for a Printed Document 493
10.7 Sample Design: A Printer-Friendly Page with CSS 495

**11. Page Layouts ..... 505**
11.1 Building a One-Column Layout 505
11.2 Building a Two-Column Layout 507
11.3 Building a Two-Column Layout with Fixed-Width Columns 511
11.4 Creating a Flexible Multicolumn Layout with Floats 514
11.5 Creating a Fixed-Width Multicolumn Layout with Floats 517
11.6 Creating a Flexible Multicolumn Layout with Positioning 520
11.7 Creating a Fixed-Width Multicolumn Layout with Positioning 523
11.8 Using Floats to Display Columns in Any Order 524
11.9 Designing an Asymmetric Layout 544
11.10 Designing Resolution-Independent Layouts 547

**12. Hacks, Workarounds, and Troubleshooting ..... 551**
12.1 Overriding Inline Styles 552
12.2 Diagnosing CSS Bugs and Browser Issues 552
12.3 Using Bookmarklets to Troubleshoot CSS 554
12.4 Using Browser Extensions to Troubleshoot CSS 555
12.5 Patching Up Internet Explorer 6 557
12.6 Patching Up Internet Explorer 6 with JavaScript 558
12.7 Using Conditional Comments to Deliver Styles to Different Versions of Internet Explorer 559

12.8 Using CSS Filters to Deliver CSS Rules to Almost Any Browser 561
12.9 Setting Up an Intelligent CSS Delivery System for Modern Browsers 562
12.10 Testing a Site Design on More Than One Platform with Only One Computer 564
12.11 Testing a Website with a Text Browser 565

**13. Designing with CSS ..... 569**
13.1 Enlarging Text Excessively 570
13.2 Creating Unexpected Incongruity 571
13.3 Combining Unlike Elements to Create Contrast 574
13.4 Leading the Eye with Contrast 576
13.5 Checking for Enough Color Contrast 578
13.6 Emphasizing a Quotation with Smart Quotes 579
13.7 Setting a Moving Background Scene When a User Resizes the Window 582
13.8 Adding Animation to Elements on a Page 584
13.9 Creating a Fireworks Display As a User Scrolls 588
13.10 Customizing the View Source Stylesheet for Firefox 590
13.11 Designing with Grids (CSS Frameworks) 591
13.12 Sample Design: A Cohesive Web Design 593
13.13 Sample Design: The U.S. Flag 609

**14. Interacting with JavaScript ..... 623**
14.1 Determining Whether JavaScript Is Available Within a Browser 623
14.2 Applying a Different Stylesheet Based on the Time of Day 625
14.3 Redirecting to a Mobile Site Based on the Browser's Screen Width 626
14.4 Adding a JavaScript Framework to a Web Page 627
14.5 Using CSS3 Selectors in IE6 and IE7 628
14.6 Zebra-Striping an HTML Table with JavaScript 630
14.7 Highlighting a Table Row with Mouseovers 632
14.8 Adding Effects to Simple Image Rollovers 634
14.9 Making a Row of Elements with a Variable Amount of Content the Same Height 635
14.10 Setting a Link to Open a New Window 638
14.11 Making an Entire div Element Clickable 639
14.12 Supporting Transparent PNGs in IE6 with JavaScript 640
14.13 Delivering HTML5 and CSS3 to Browsers That Can Handle Them 642

**A. Resources ..... 645**

B. CSS 2.1 Properties and Proprietary Extensions ..... 651

C. CSS 2.1 Selectors, Pseudo-Classes, and Pseudo-Elements ..... 669

D. CSS3 Selectors and Pseudo-Classes ..... 673

E. Styling of Form Elements ..... 677

Index ..... 845

# 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:

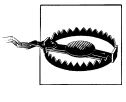
*Italic*
: Indicates new terms, URLs, email addresses, filenames, file extensions, pathnames, directories, and Unix utilities

`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*

CHAPTER 1

# 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:

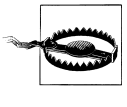
- Notepad++ (Windows OS; *http://notepad-plus.sourceforge.net/uk/site.htm*)
- TextWrangler (Mac OS; *http://www.barebones.com/products/TextWrangler/*)
- jEdit (Windows OS, Mac OS, and Linux OS; *http://www.jedit.org/*)

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.textpad.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.

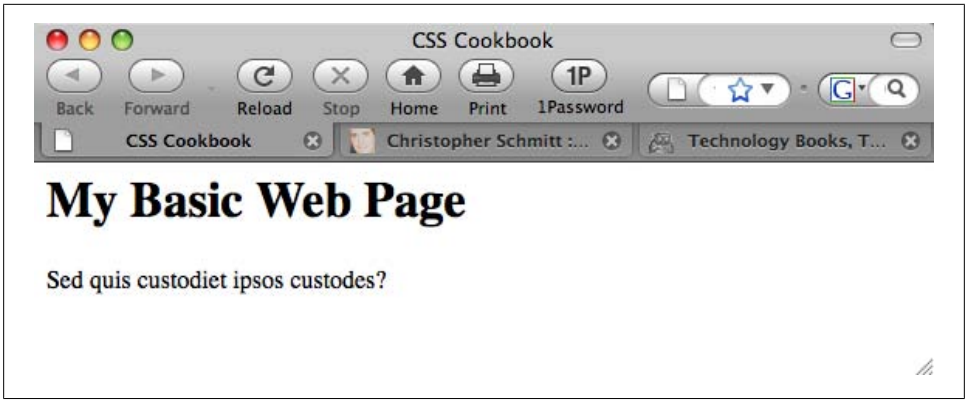



*Figure 1-1. The default rendering of a basic HTML web page*

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.

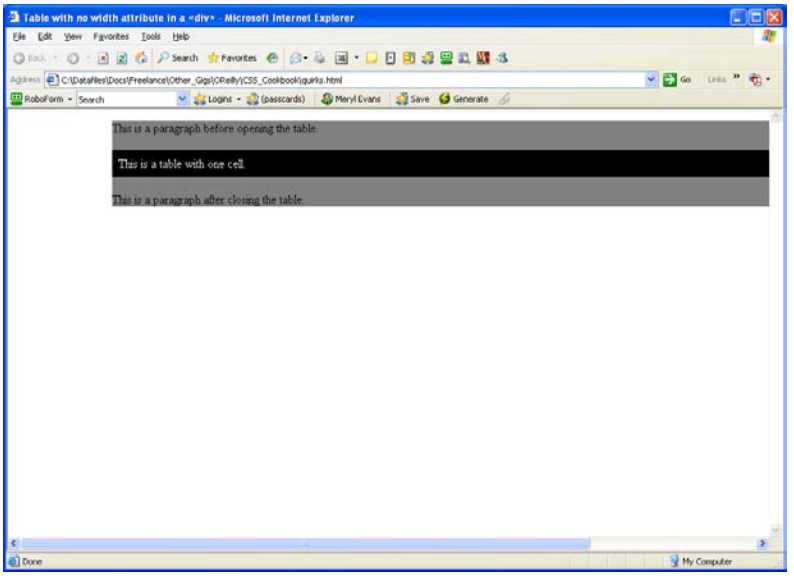



*Figure 1-2. Table width in Internet Explorer 6 in quirks mode with no DOCTYPE included*

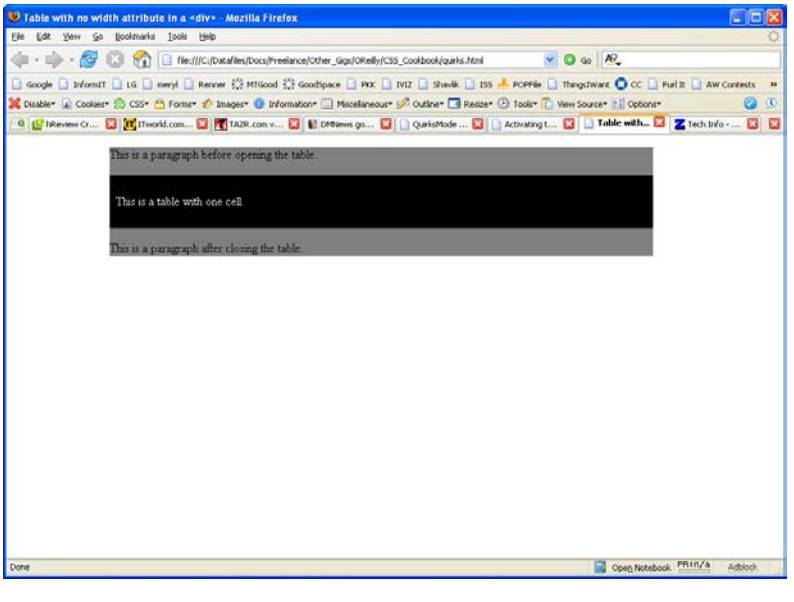



*Figure 1-3. Table width in Firefox 1.5 in standards mode with HTML 4.01 Strict DOCTYPE*

#### 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>
```

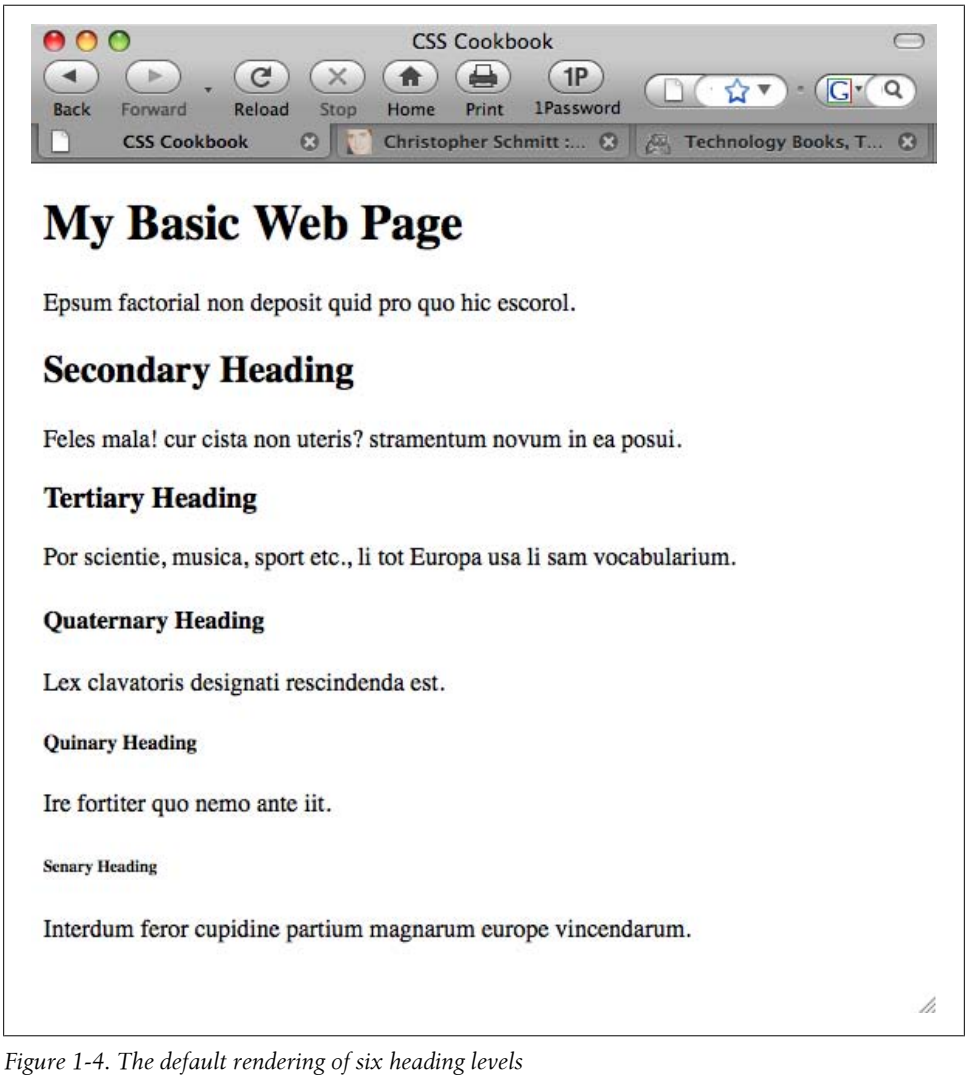



*Figure 1-4. The default rendering of six heading levels*

### 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>
```

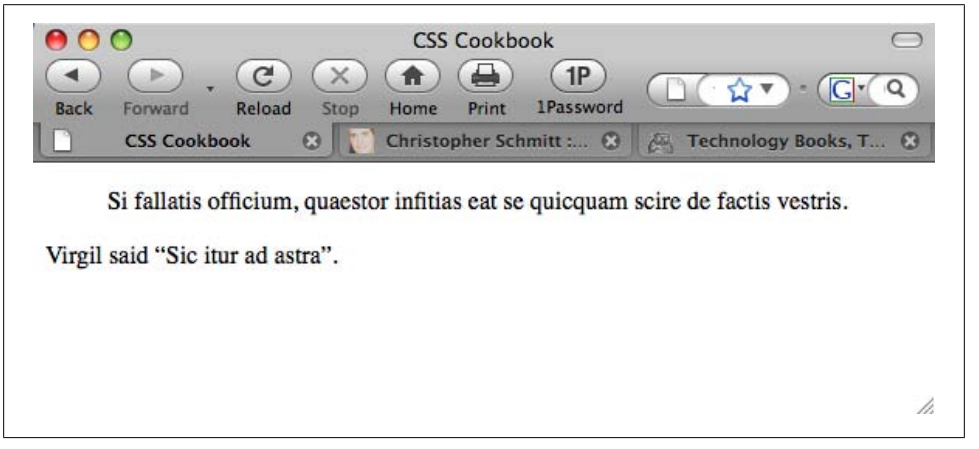



*Figure 1-5. The default rendering of quotations*

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.

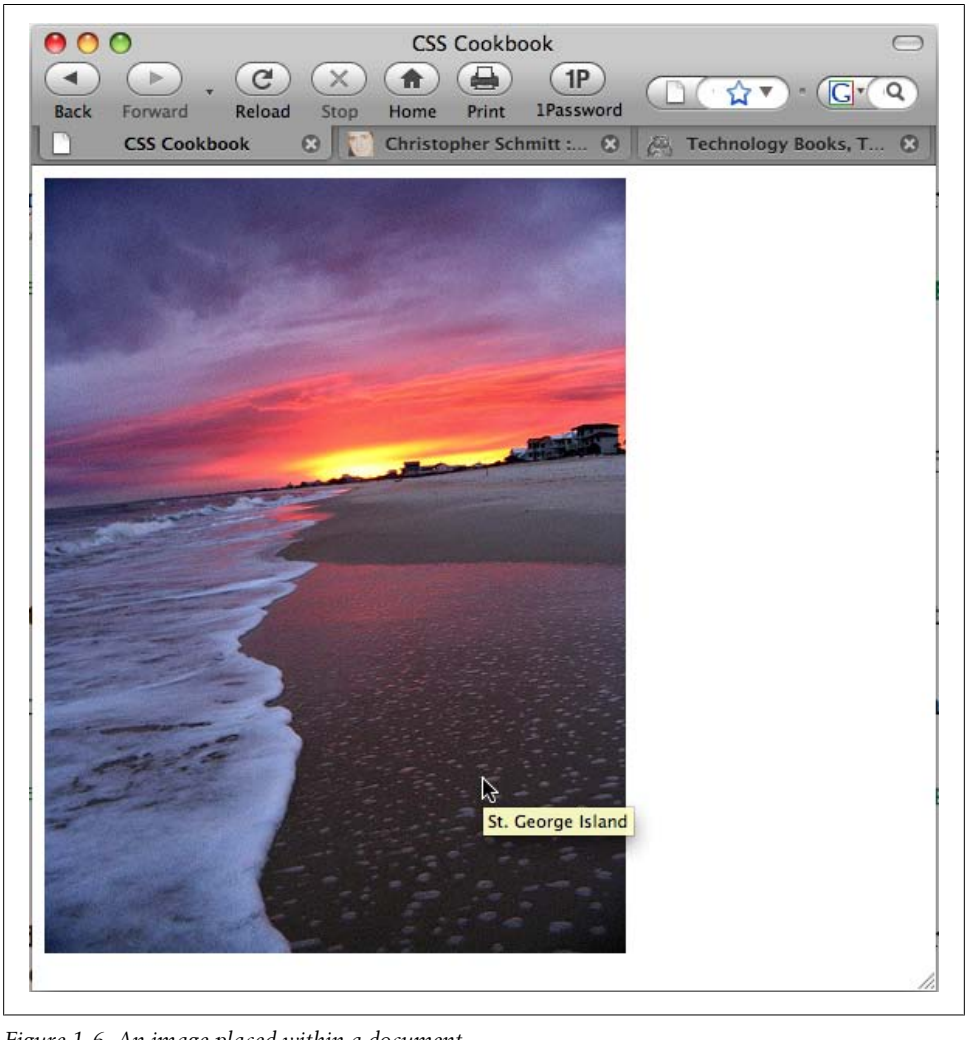



*Figure 1-6. An image placed within a document*

## 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>
```

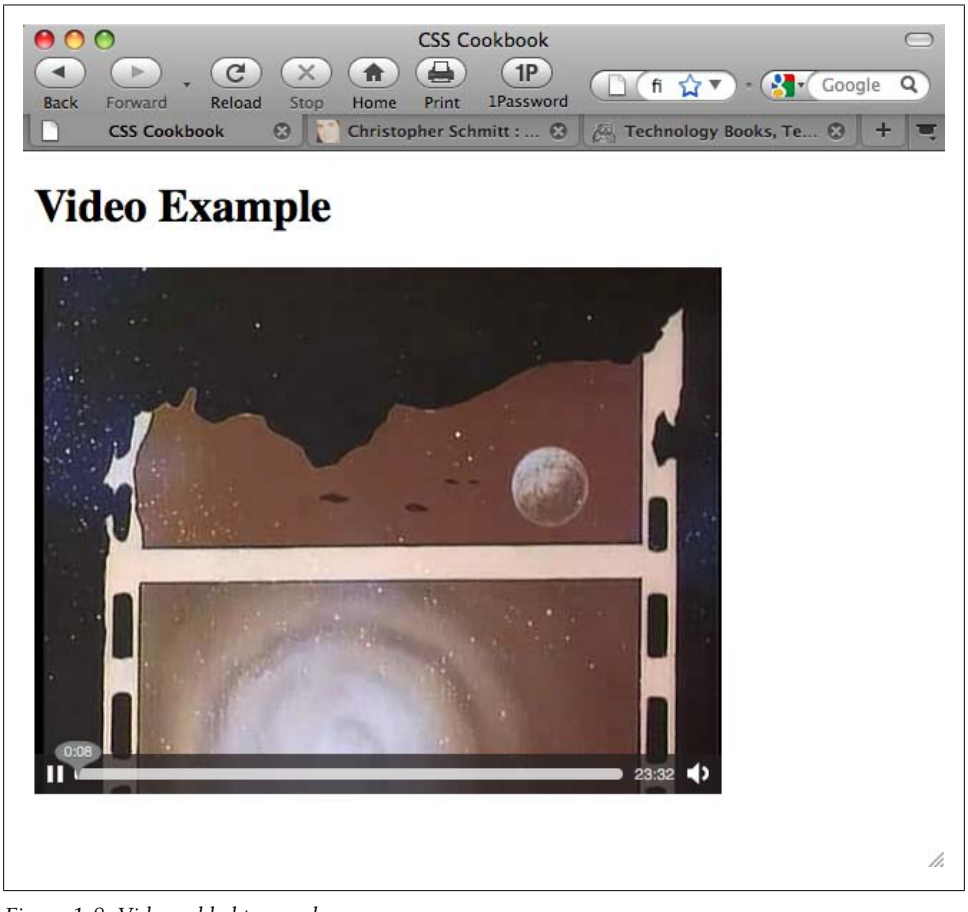



*Figure 1-8. Video added to a web page*

## 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.

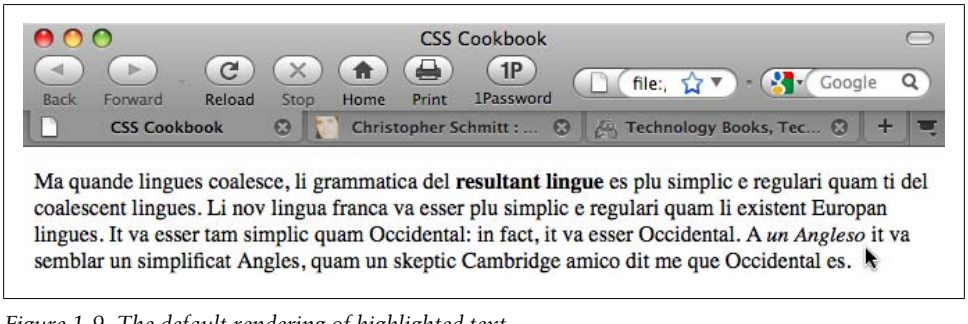



*Figure 1-9. The default rendering of highlighted text*

### 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.

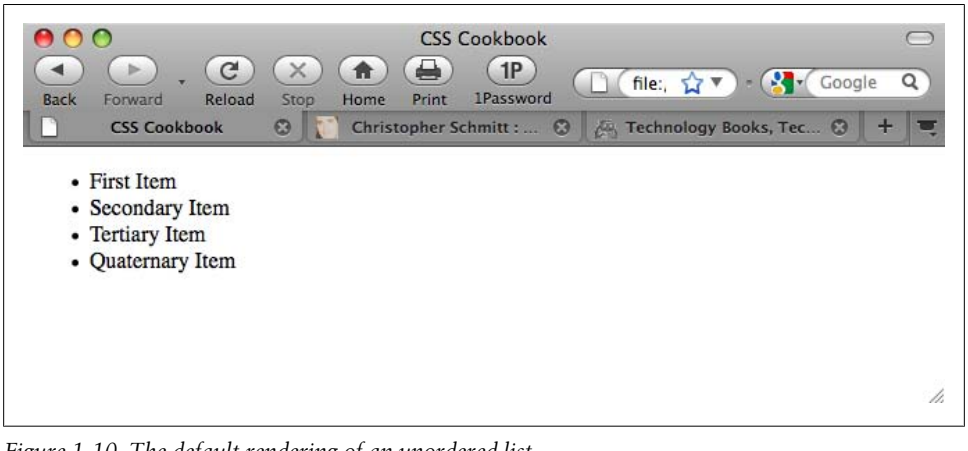



*Figure 1-10. The default rendering of an unordered list*

## 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>
```

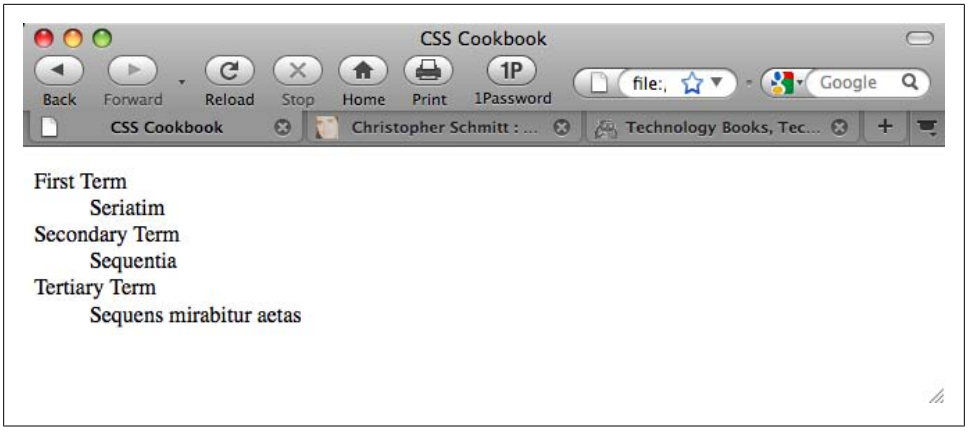



*Figure 1-11. The default rendering of a definition list*

### 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>
```

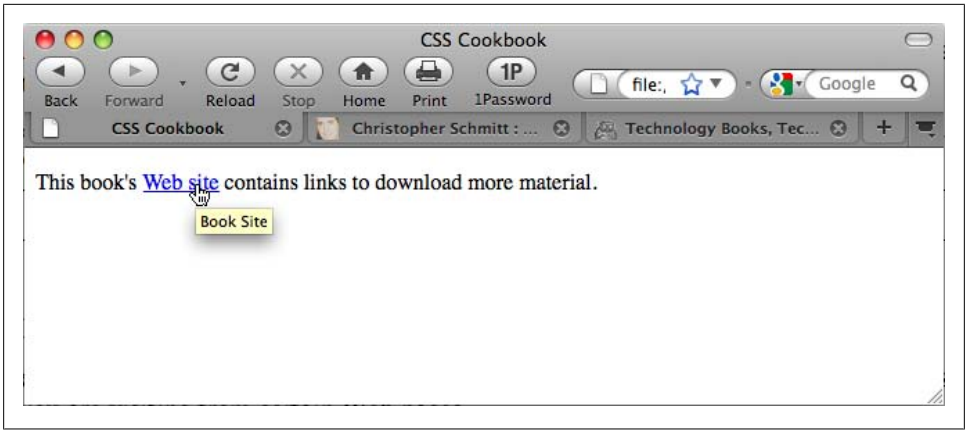



*Figure 1-12. A tool tip displayed over a link*

### 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*

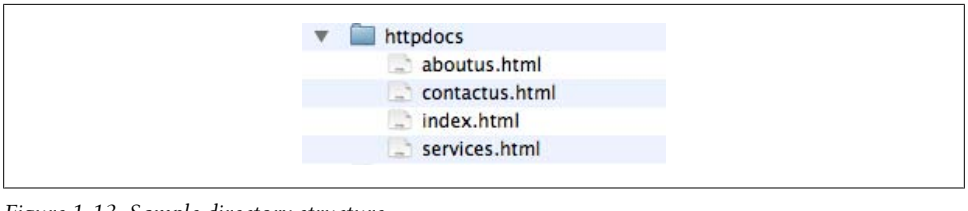



*Figure 1-13. Sample directory structure*

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 shorthand 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.

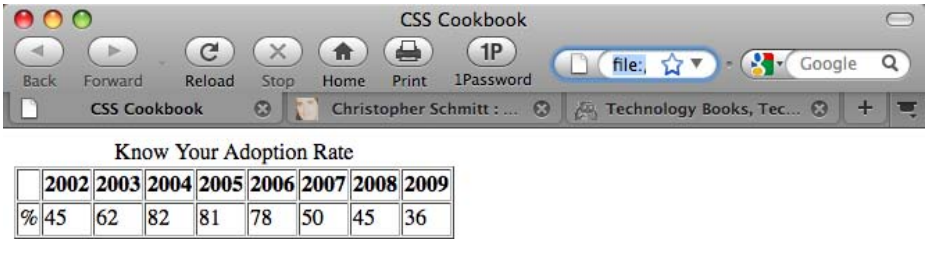



*Figure 1-14. The default rendering of a basic HTML table*

## 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.

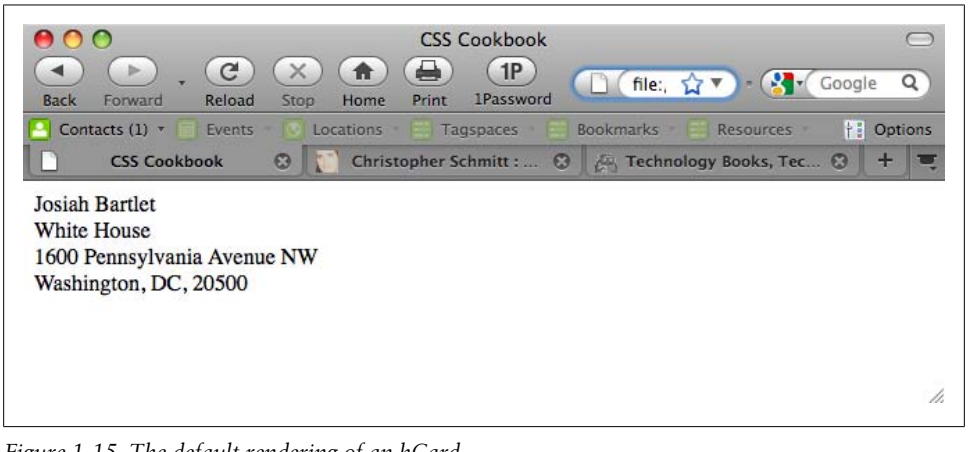



*Figure 1-15. The default rendering of an hCard*

## 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.

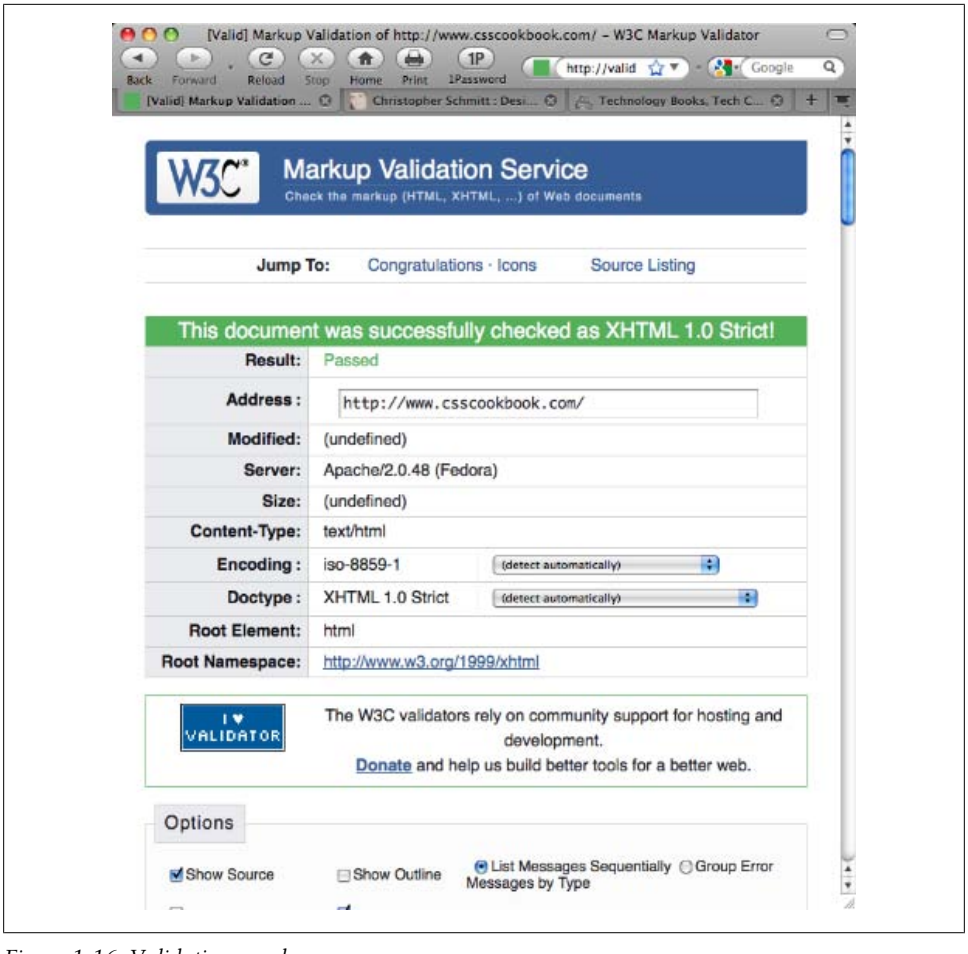



*Figure 1-16. Validating a web page*

## 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

CHAPTER 2

# 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.

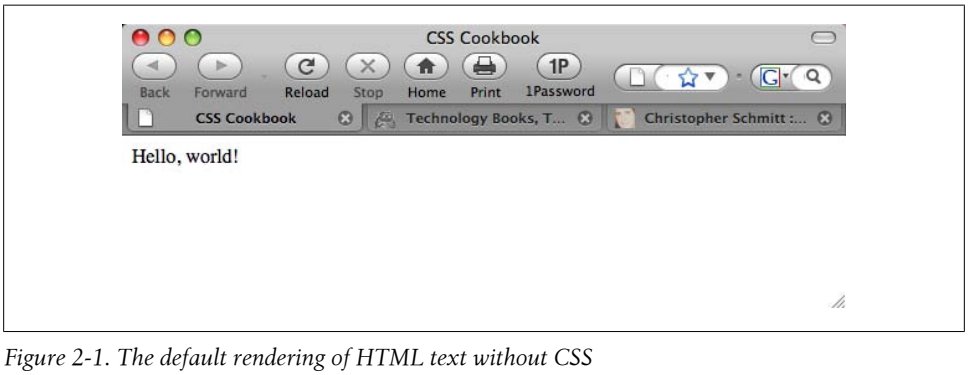



*Figure 2-1. The default rendering of HTML text without CSS*

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>
```

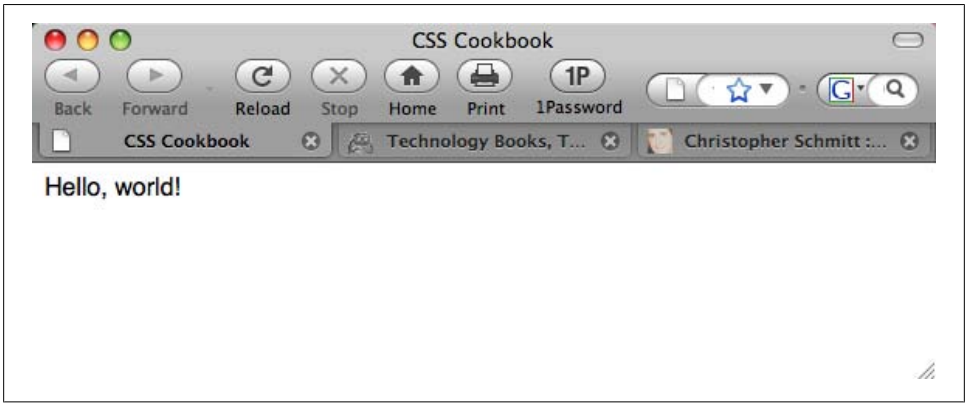



*Figure 2-2. The font, changed to sans serif through CSS*

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>
```

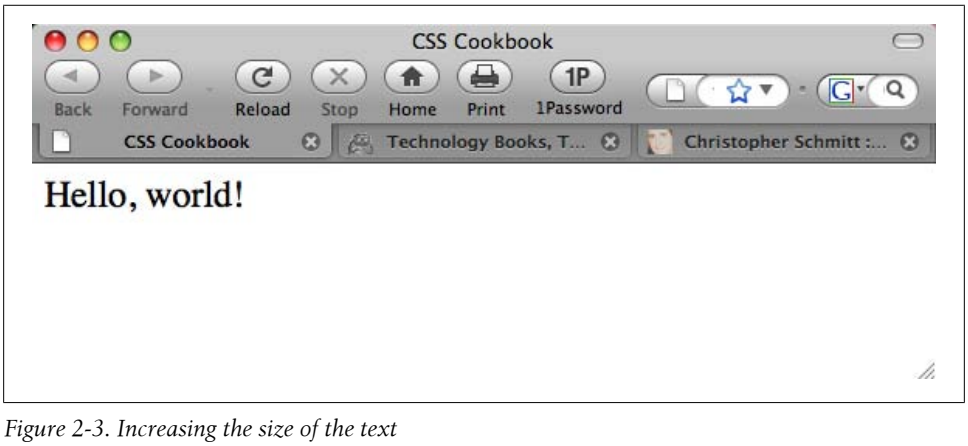



*Figure 2-3. Increasing the size of the text*

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>
```

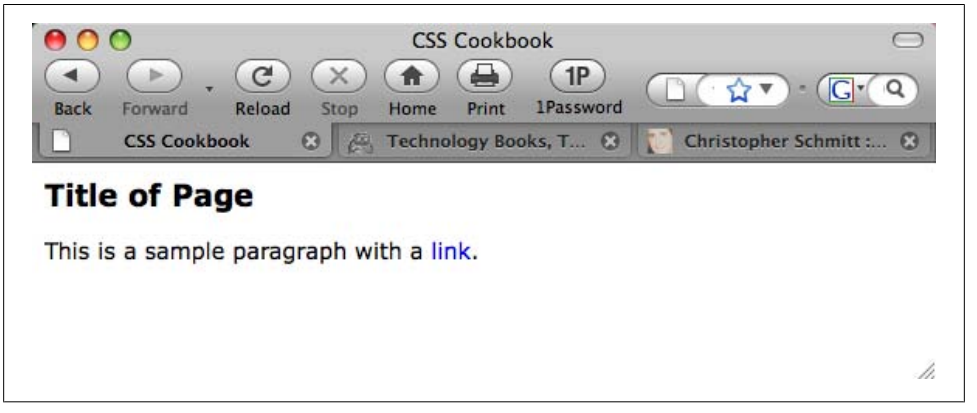



*Figure 2-4. Content rendered differently after adding CSS*

## 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>
```

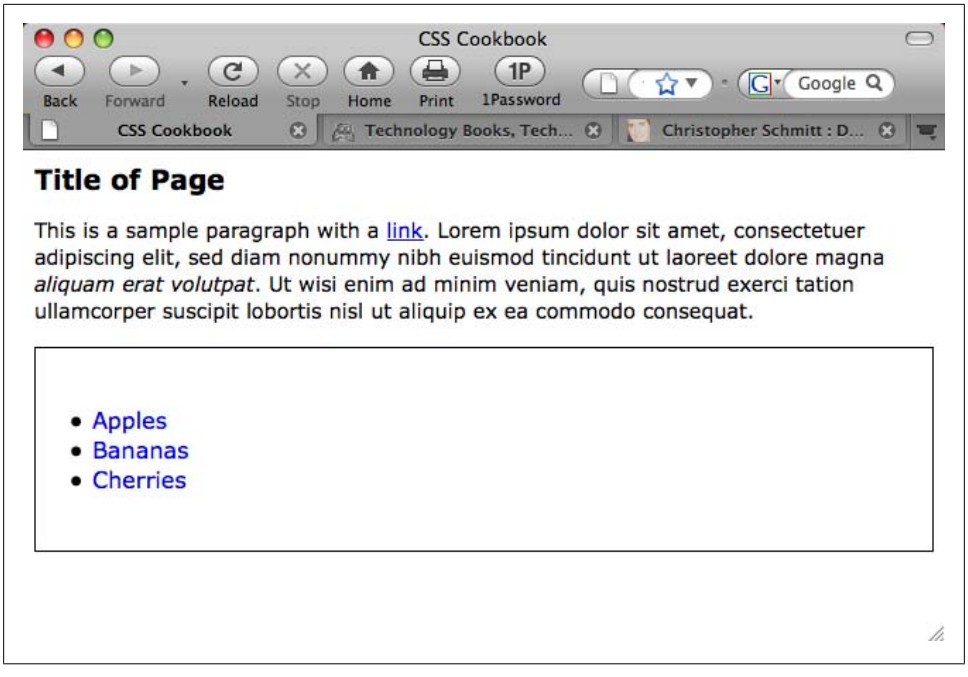



*Figure 2-5. Web page with CSS styles*

## 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.

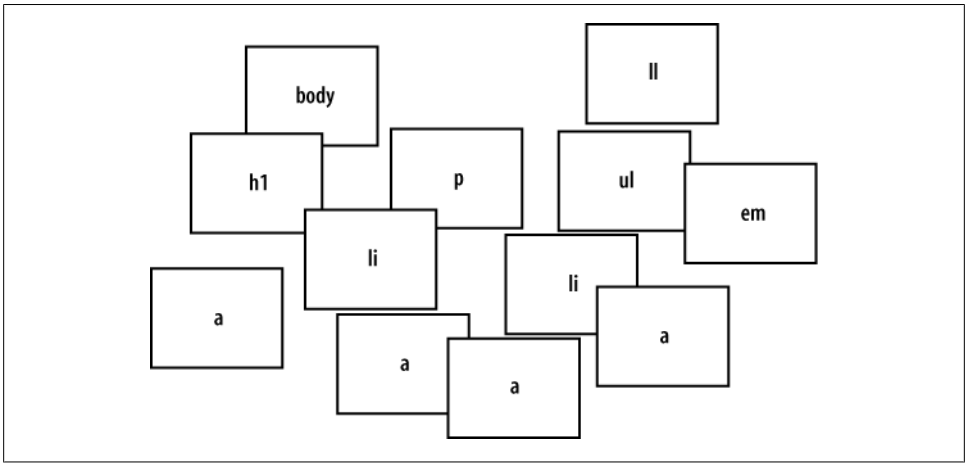



*Figure 2-6. Elements used in the Solution*

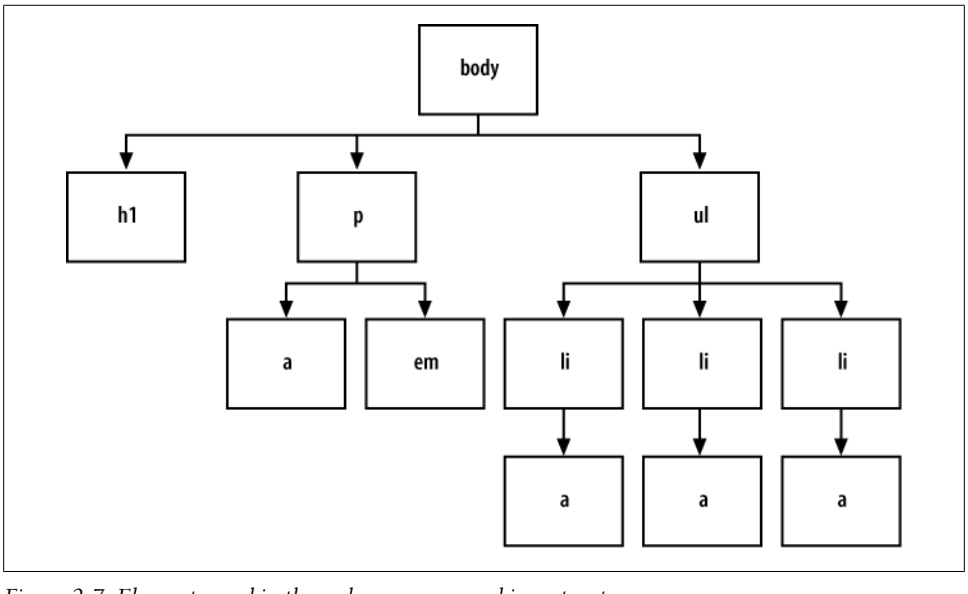



*Figure 2-7. Elements used in the web page arranged in a structure*

### 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;
}
```

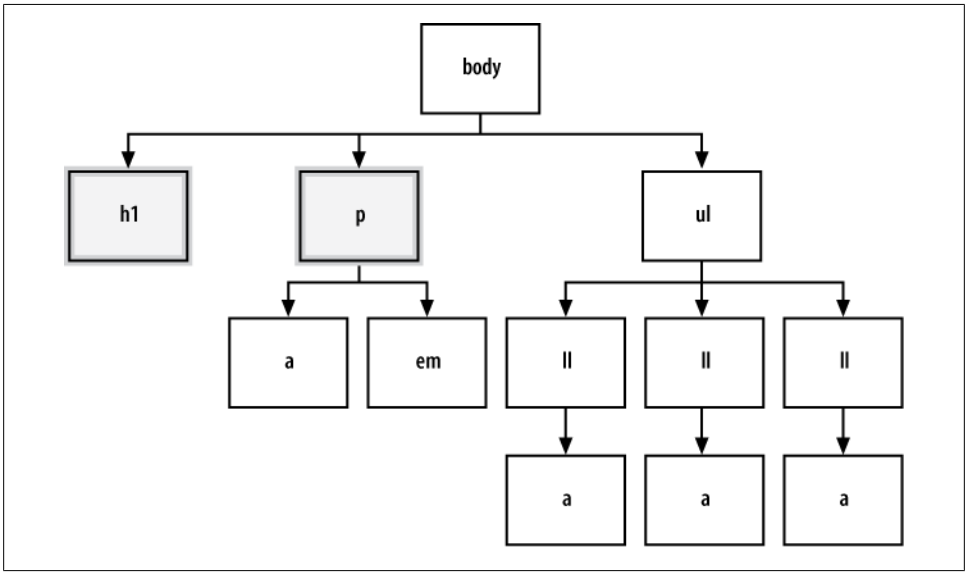



*Figure 2-8. The elements selected from the CSS rules*

### 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>
```

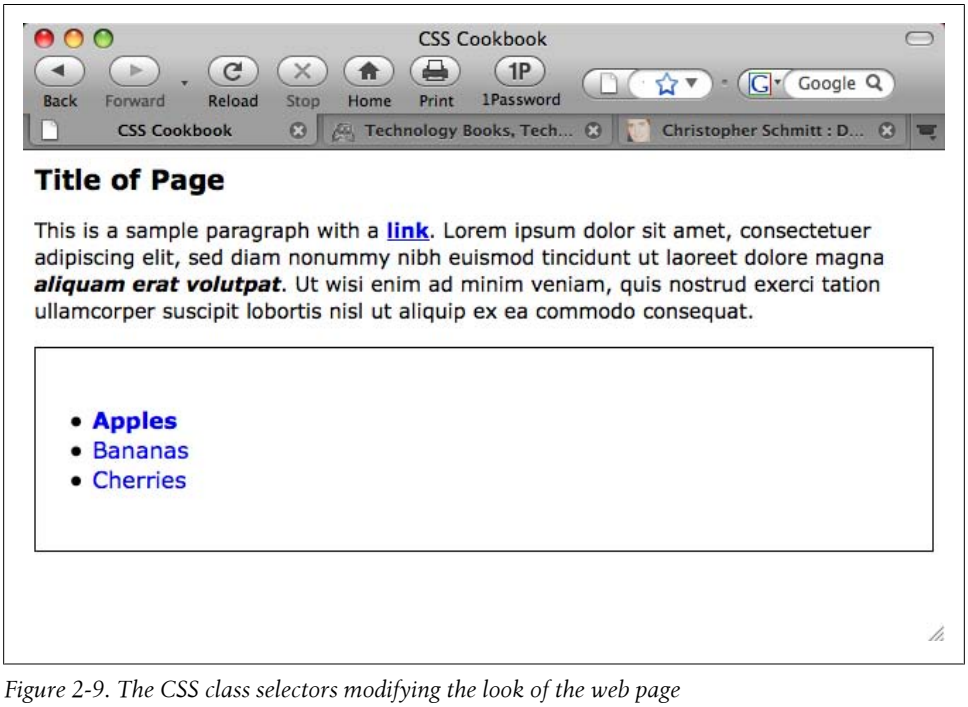



*Figure 2-9. The CSS class selectors modifying the look of the web page*

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

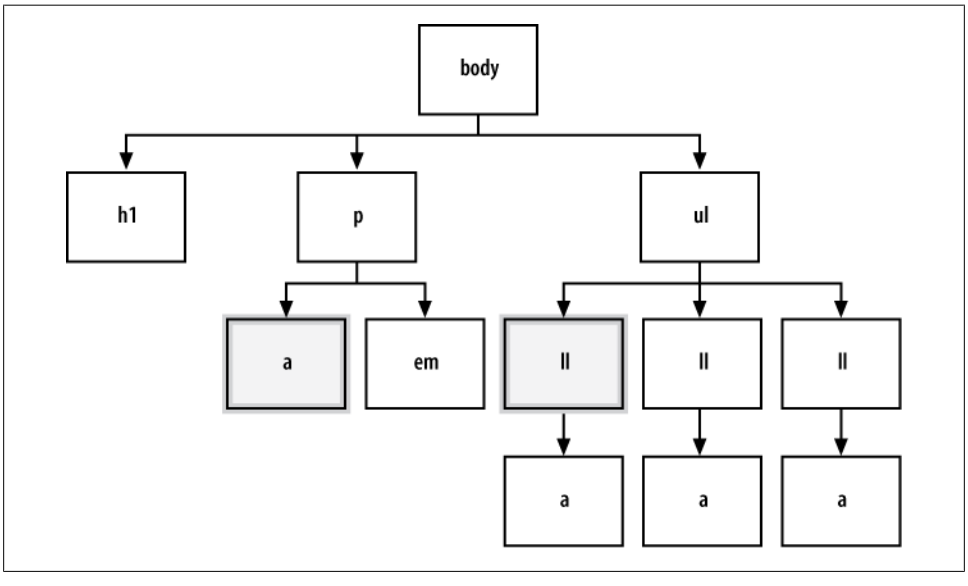



*Figure 2-10. The styled elements within the page structure*

### 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>
```

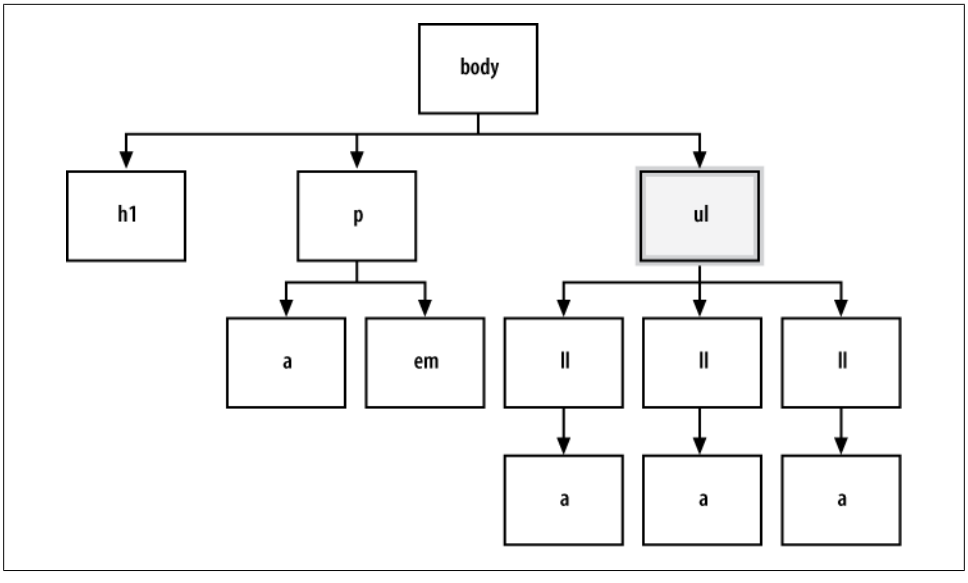



*Figure 2-11. An unordered list element, styled*

### 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;
}
```

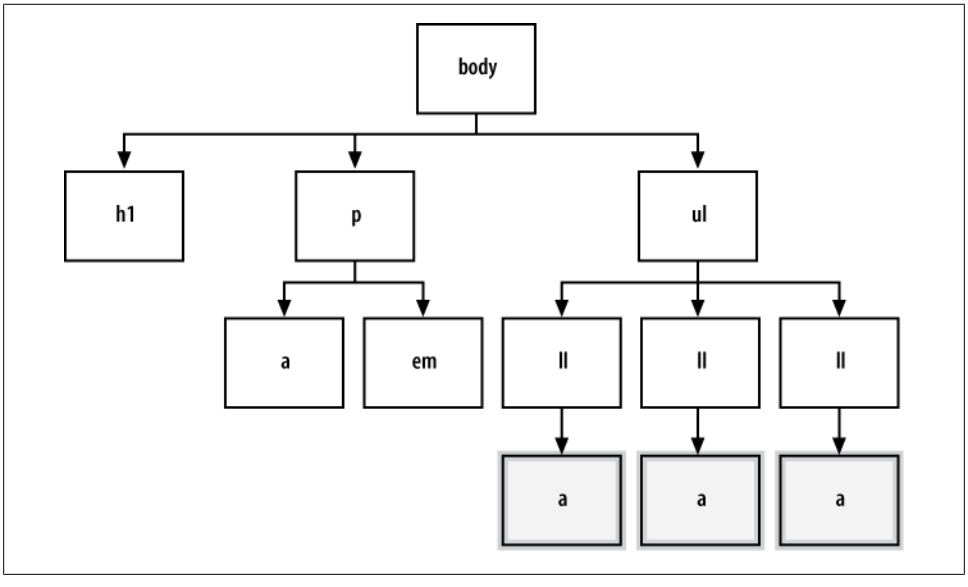



*Figure 2-12. The links within the list items selected*

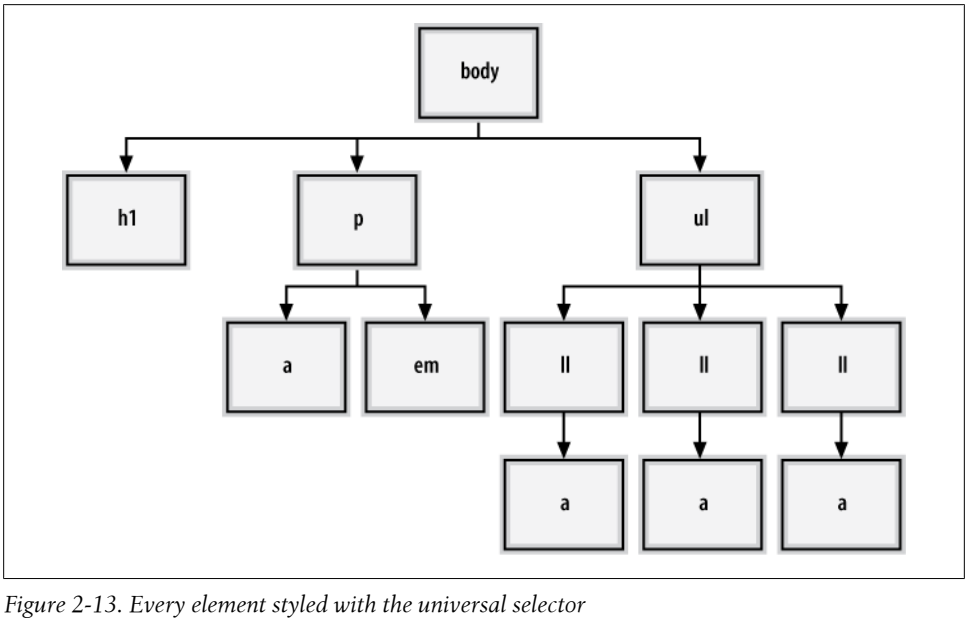



*Figure 2-13. Every element styled with the universal selector*

### 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.

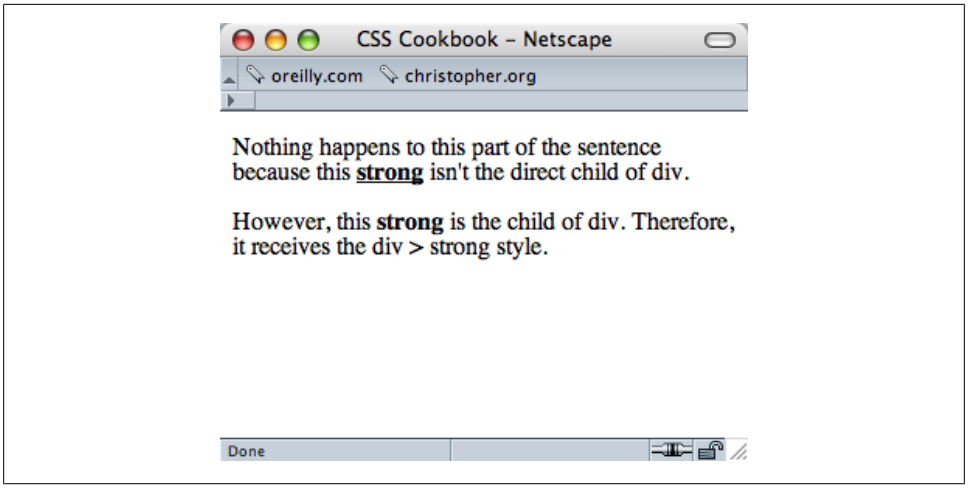



*Figure 2-14. The effect of the child selector rule*

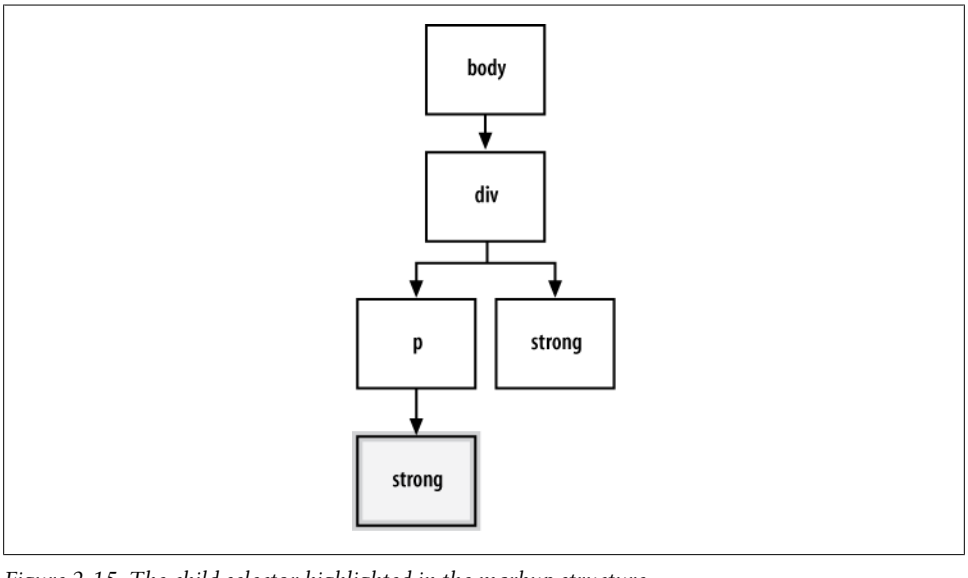



*Figure 2-15. The child selector highlighted in the markup structure*

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*

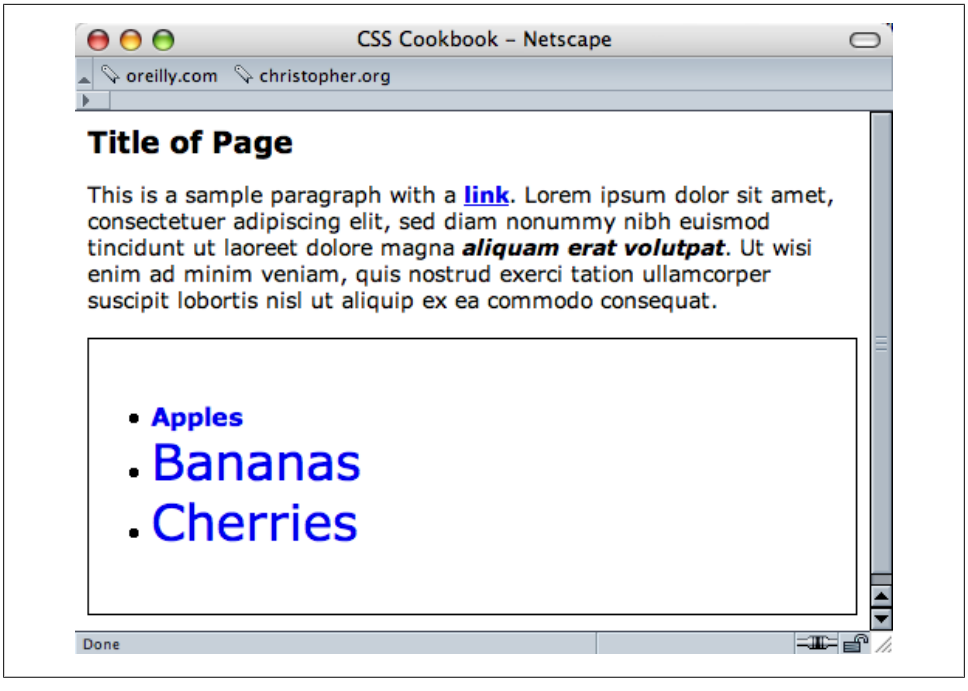



*Figure 2-16. Adjacent sibling selectors, which affect the ordered list because it appears after the unordered list*

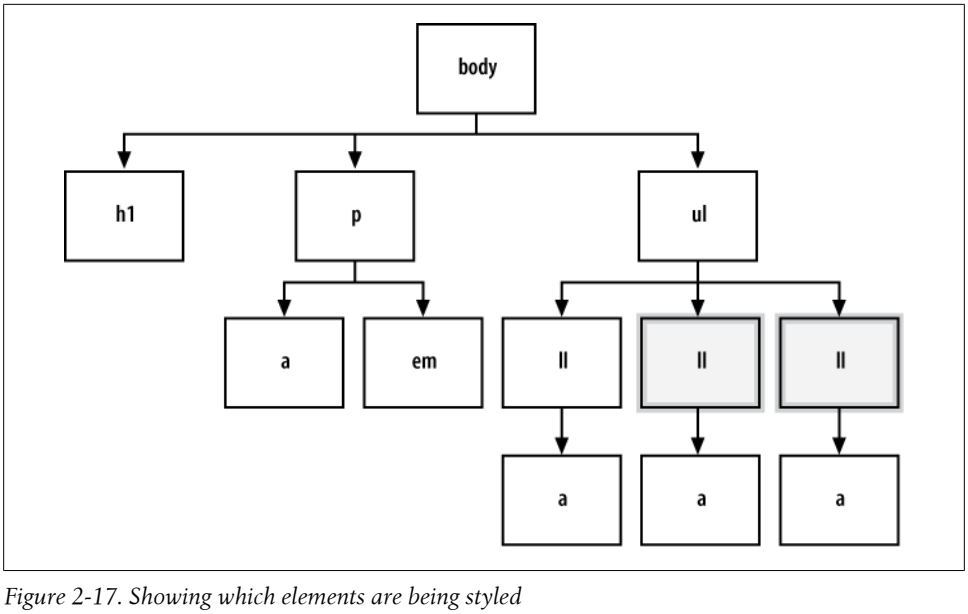



*Figure 2-17. Showing which elements are being styled*

## 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;
}
```

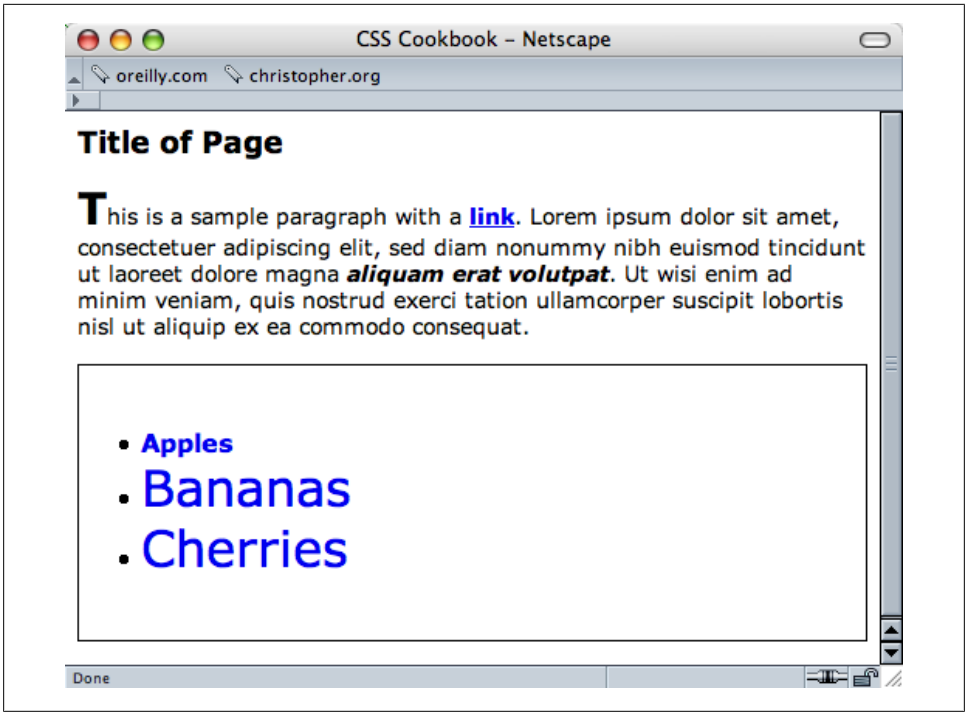



*Figure 2-18. The first letter, styled*

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`.

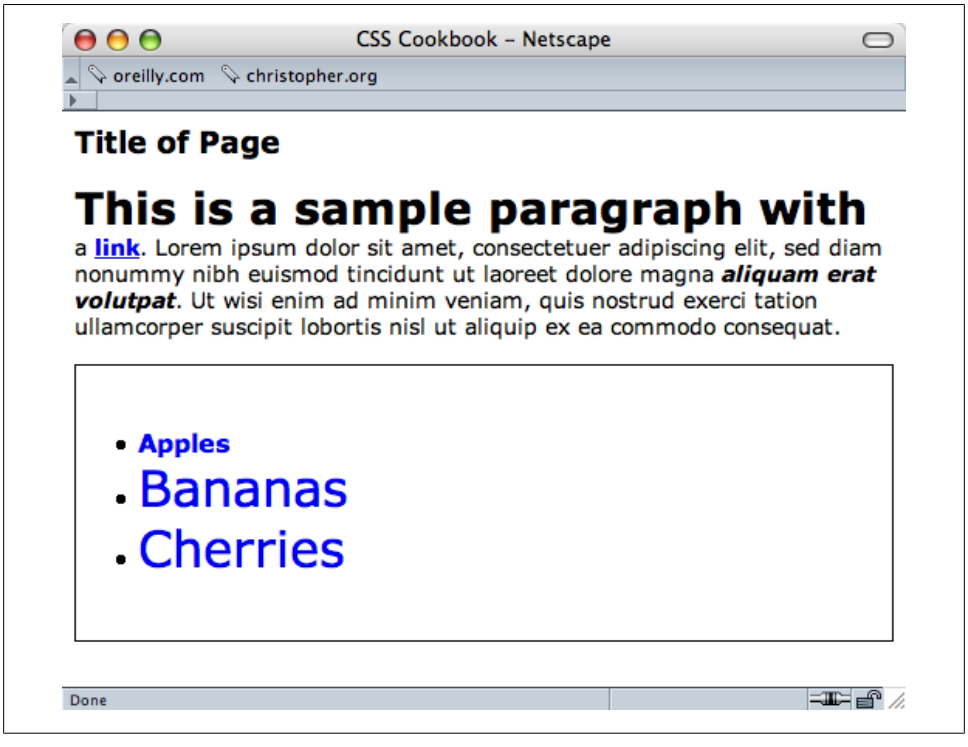



*Figure 2-19. The first line, styled*

### 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-selectors*; 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.

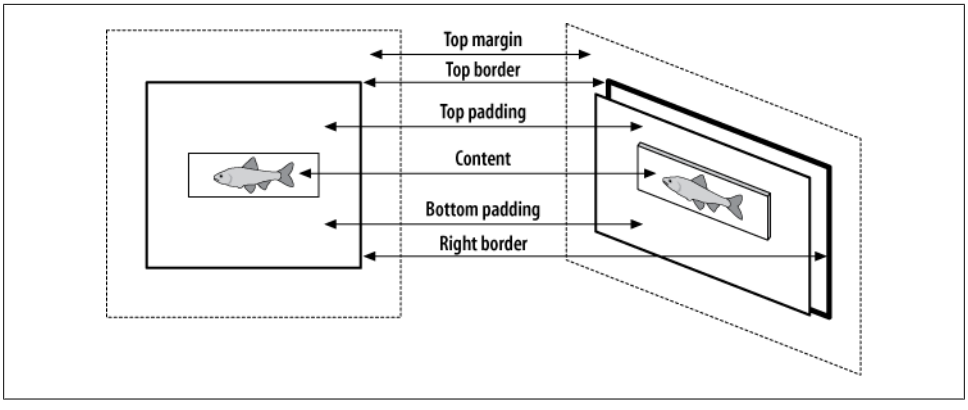



*Figure 2-20. Box model viewed straight on and off to the side*

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;
}
```

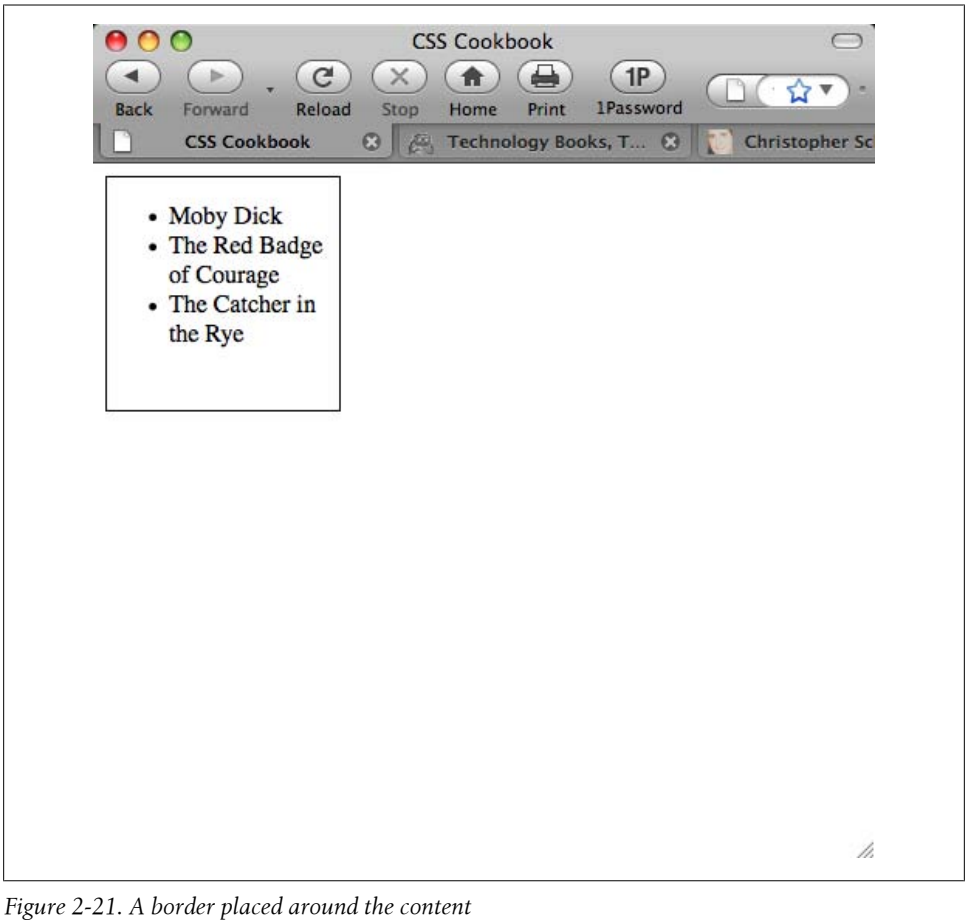



*Figure 2-21. A border placed around the content*

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;
}
```

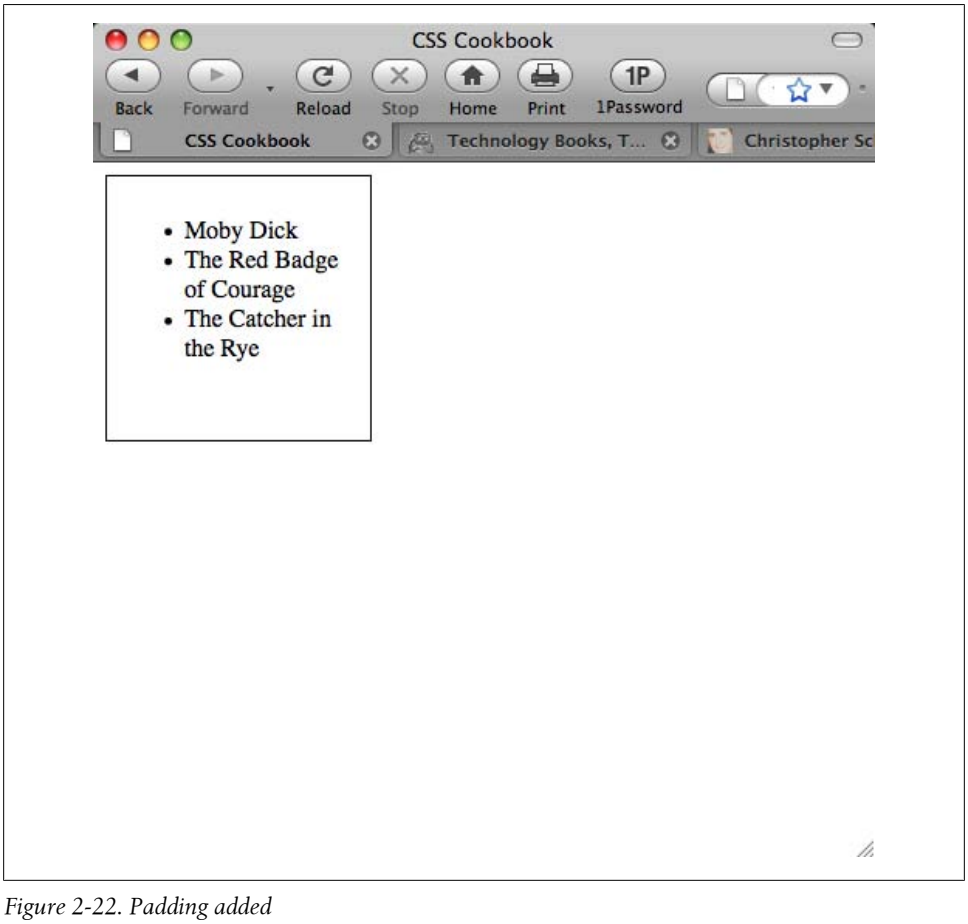



*Figure 2-22. Padding added*

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>
```

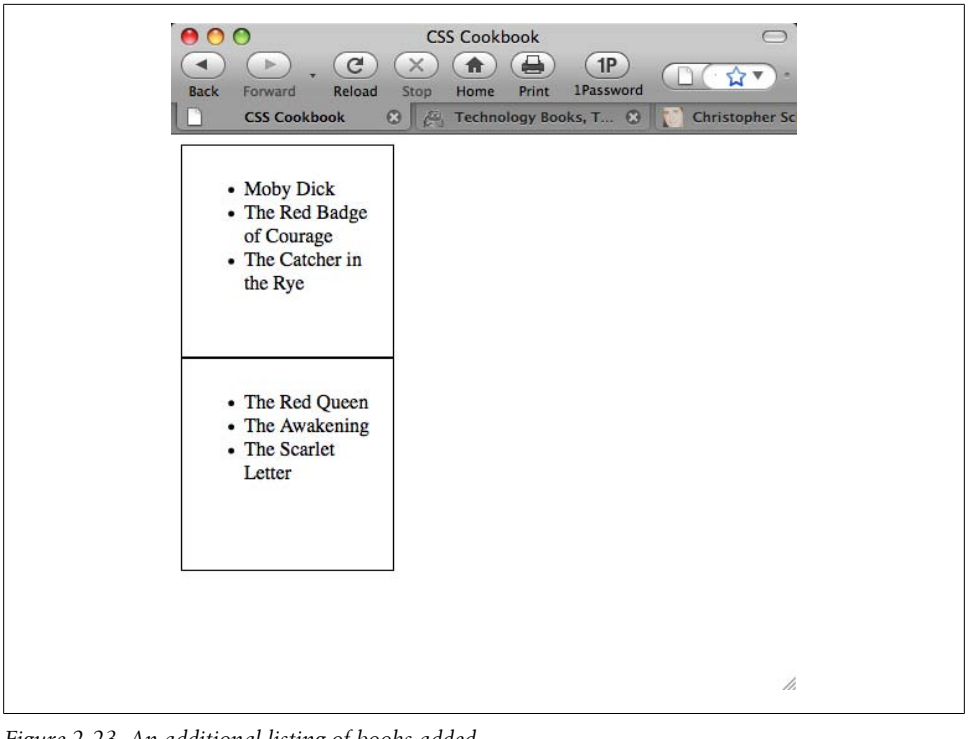



*Figure 2-23. An additional listing of books added*

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;
}
```

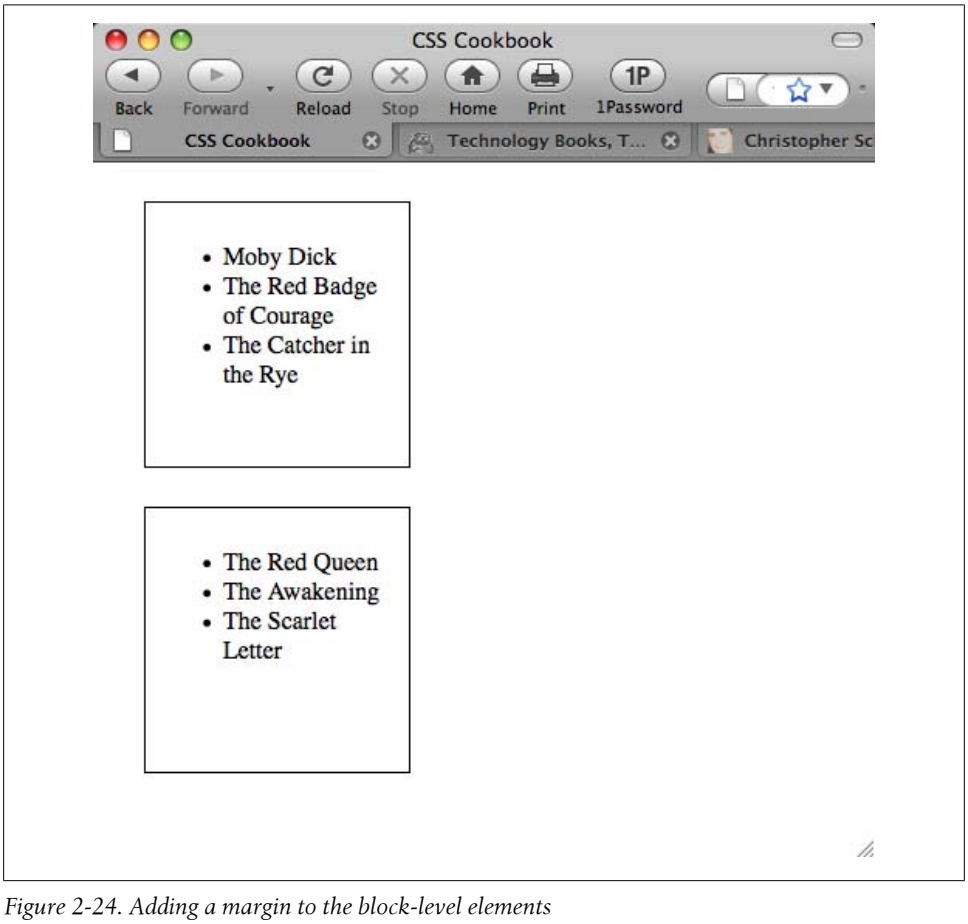



*Figure 2-24. Adding a margin to the block-level elements*

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;
}
```

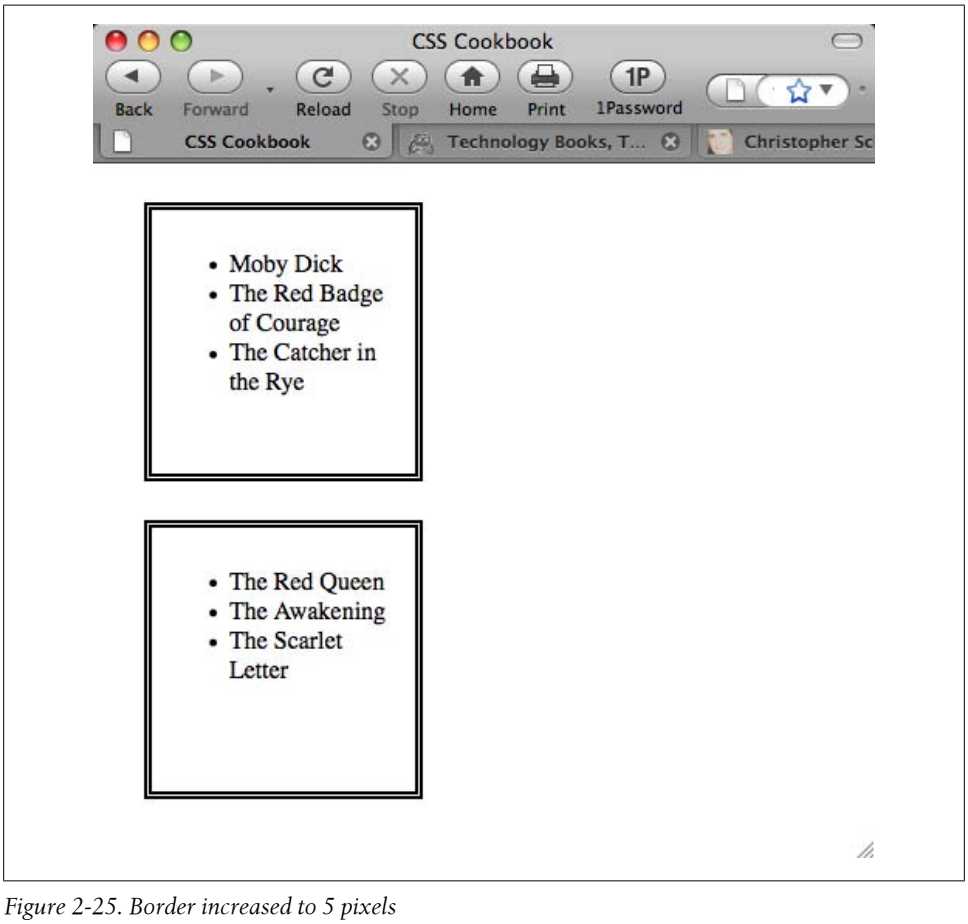



*Figure 2-25. Border increased to 5 pixels*

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;
}
```

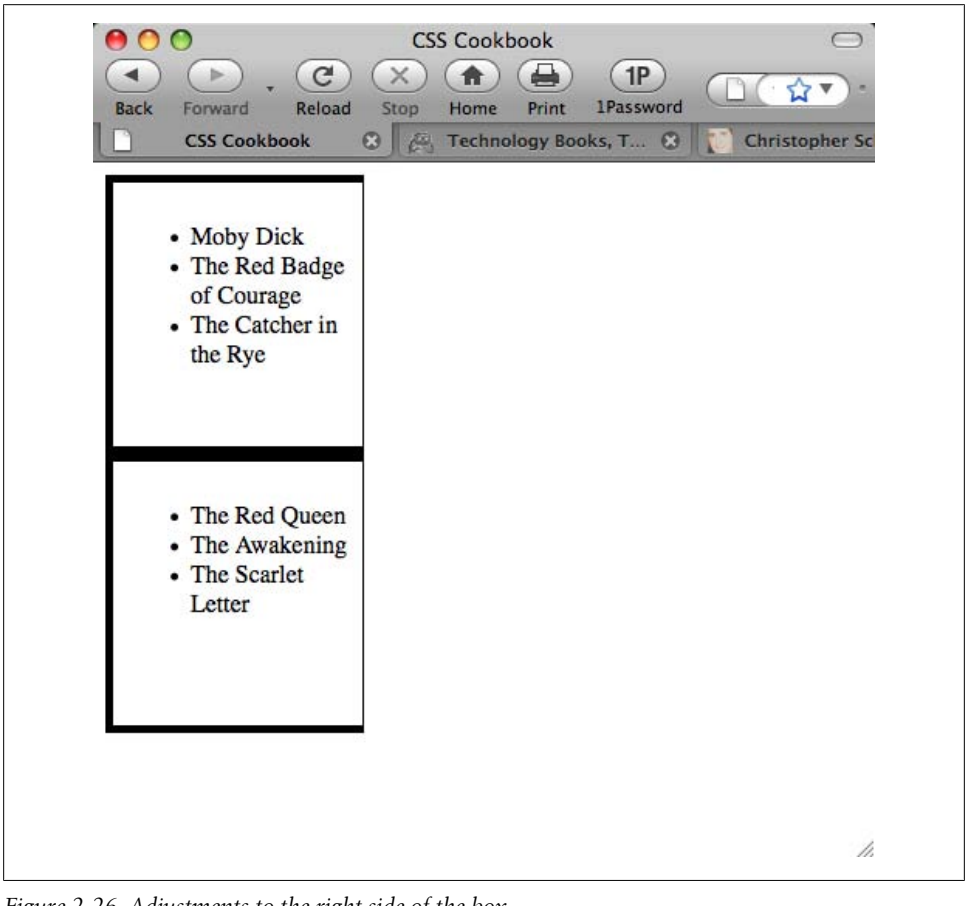



*Figure 2-26. Adjustments to the right side of the box*

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

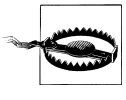
You can apply styles in three ways: externally, internally, and inline. An internal stylesheet appears near the top of the HTML document, within the `head`:

```
<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 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:

```
<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:

```
<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;
}
```

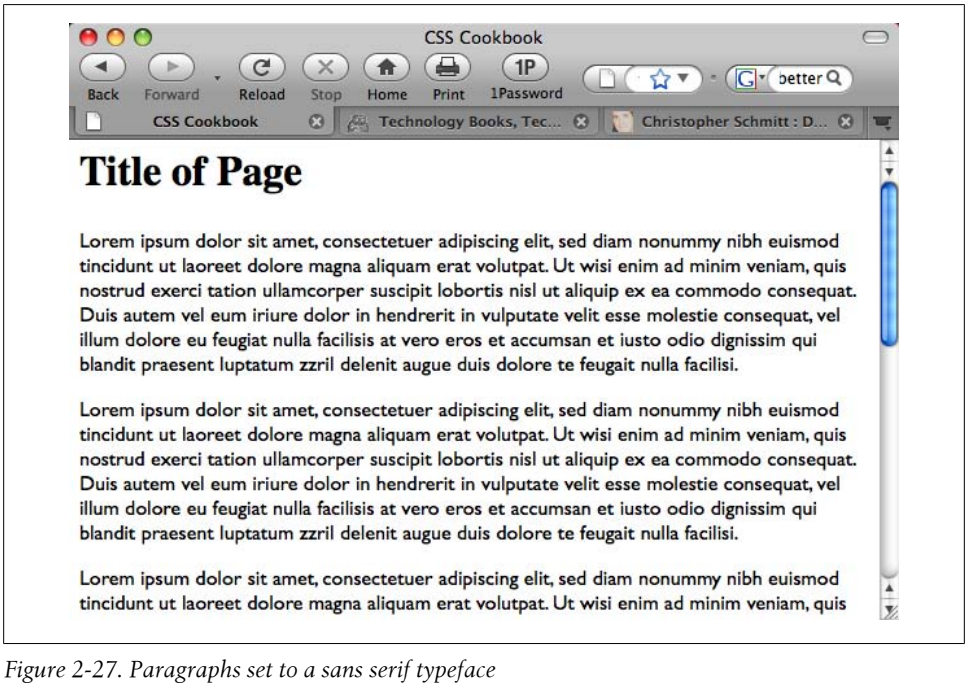



*Figure 2-27. Paragraphs set to a sans serif typeface*

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;
}
```

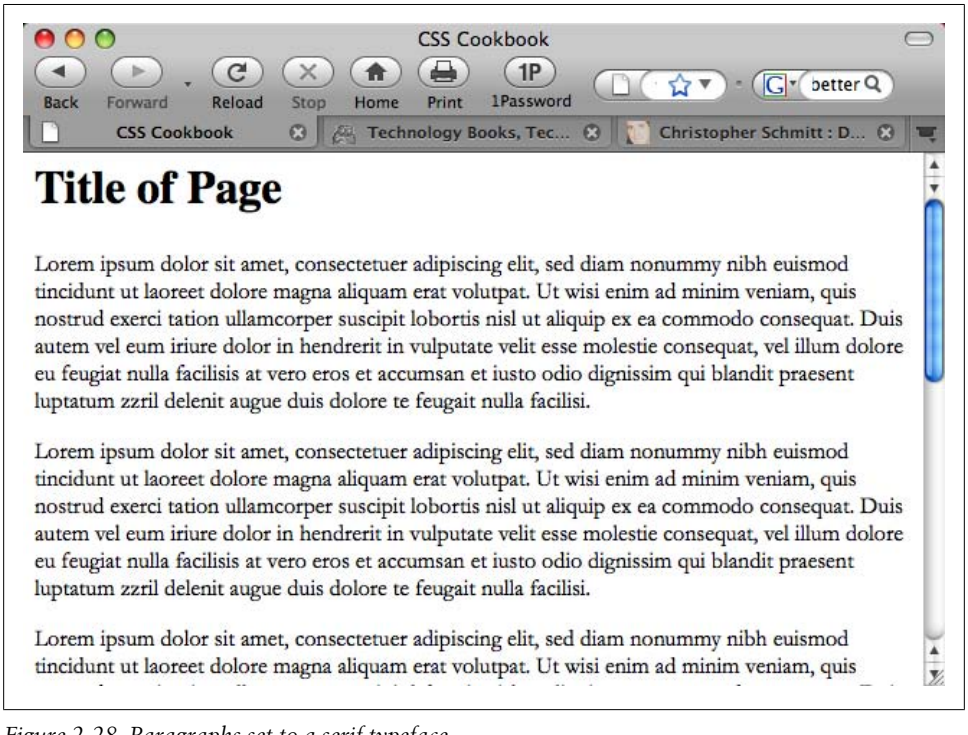



*Figure 2-28. Paragraphs set to a serif typeface*

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.

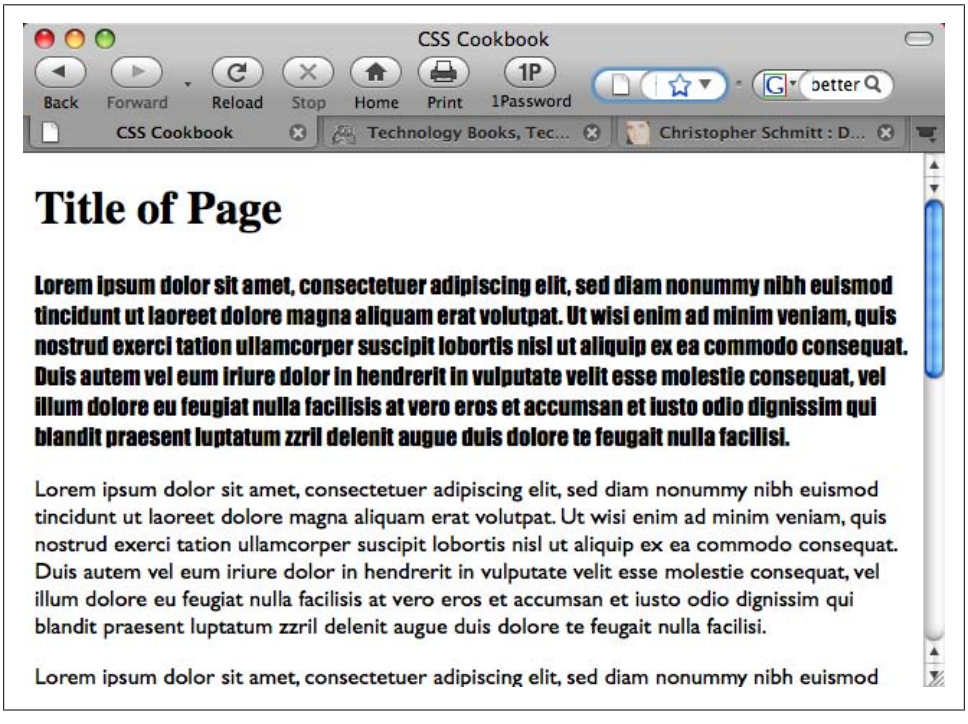



*Figure 2-29. The winning CSS rule*

## 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;
}
```

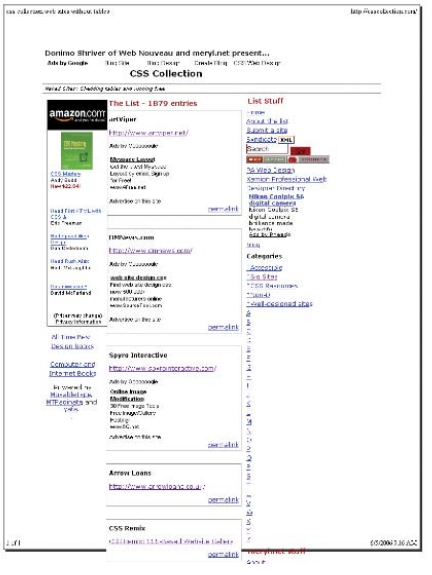



*Figure 2-30. How the page would look if printed without print.css*

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;
}
```

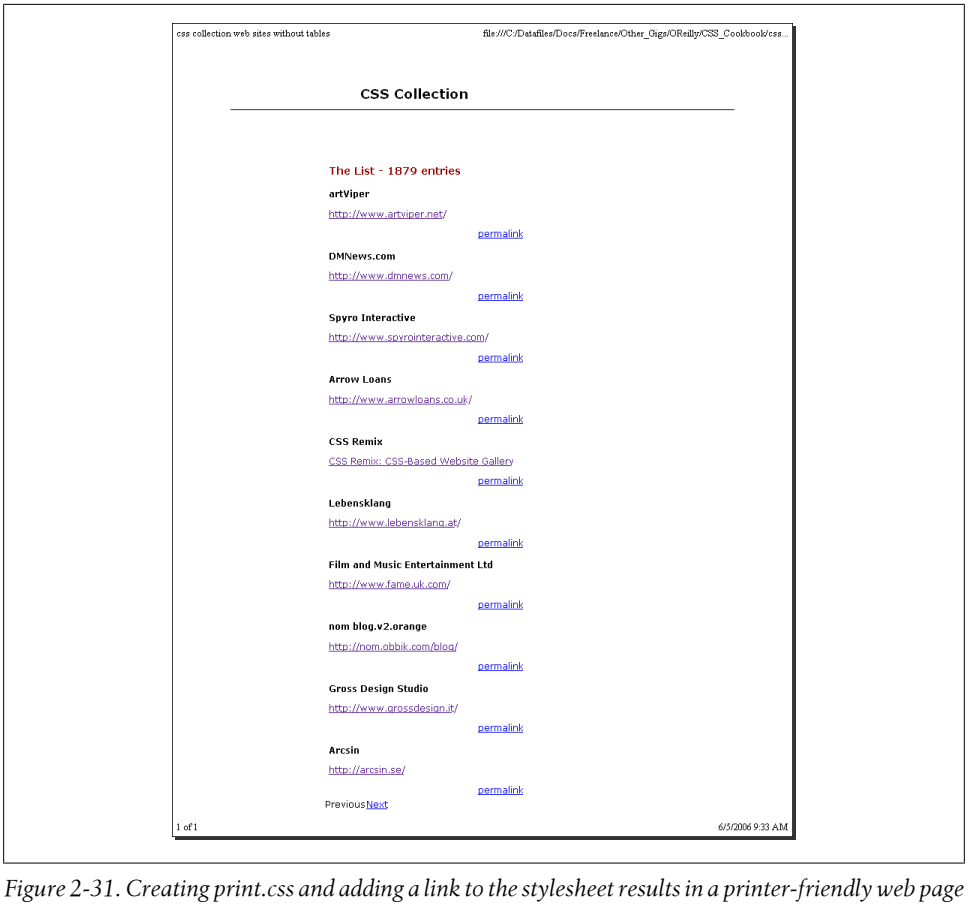



*Figure 2-31. Creating print.css and adding a link to the stylesheet results in a printer-friendly web page*

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

| 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.

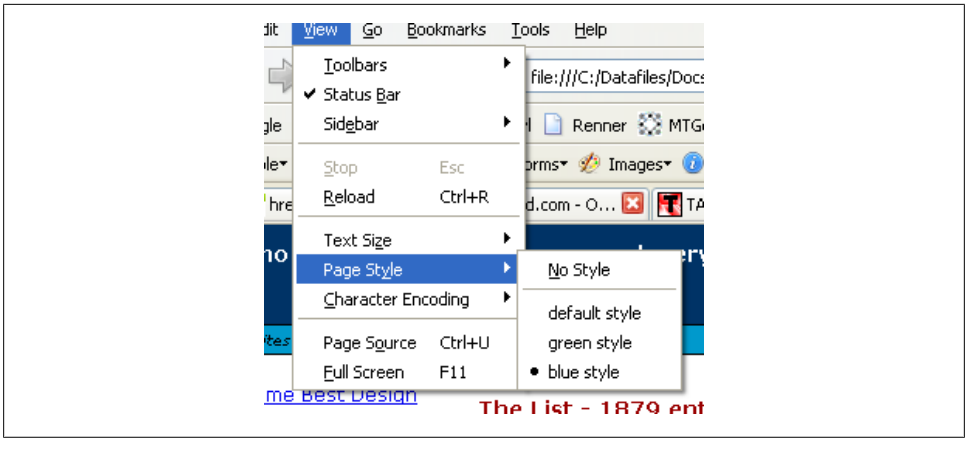



*Figure 2-32. Switching stylesheets within the browser options*

## 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.

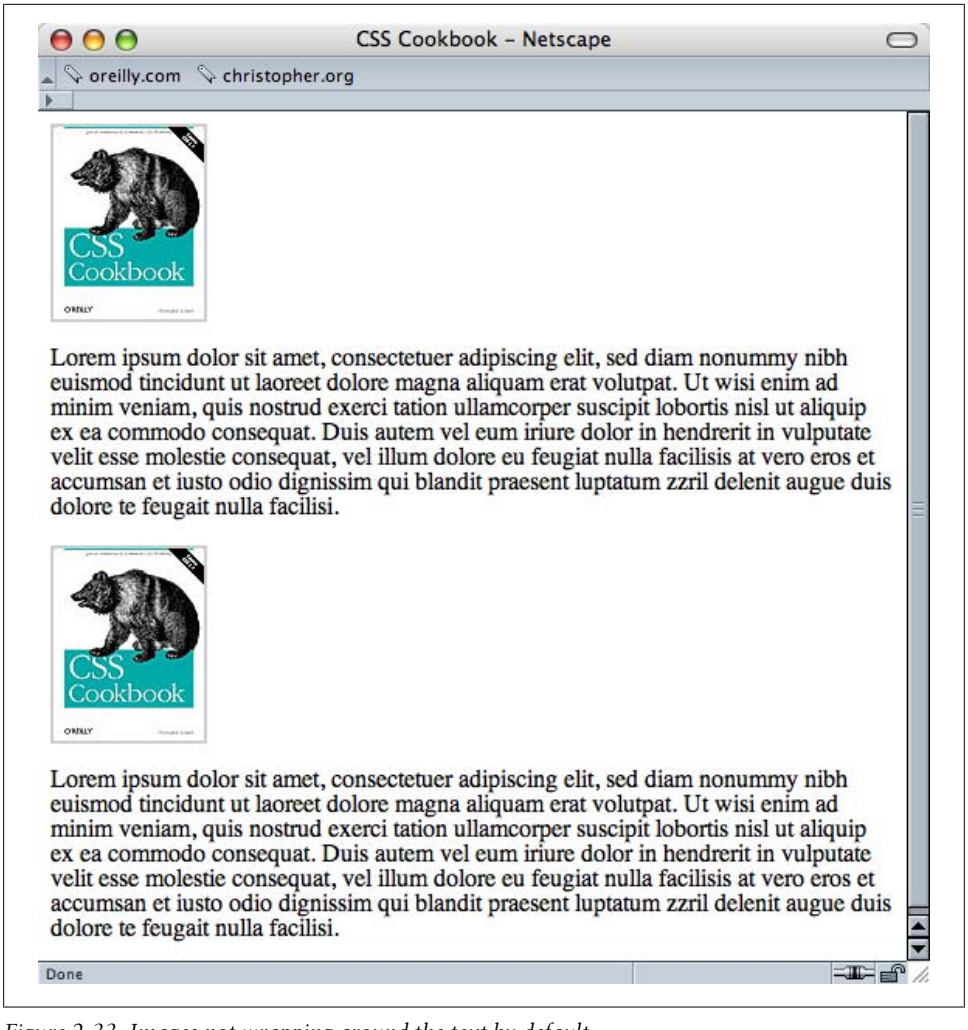
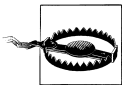



*Figure 2-33. Images not wrapping around the text by default*

## 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%;
}
```

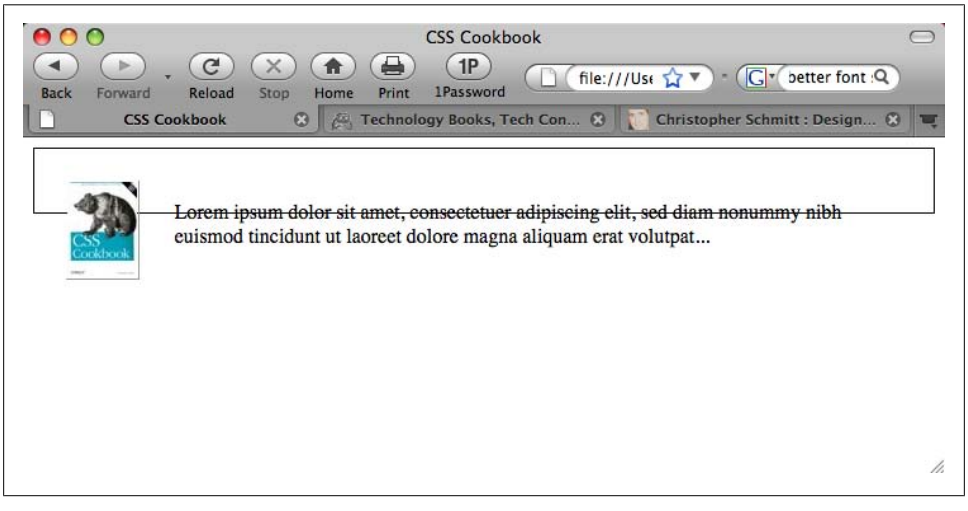



*Figure 2-35. The image and paragraph overlapping the border*

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>
```

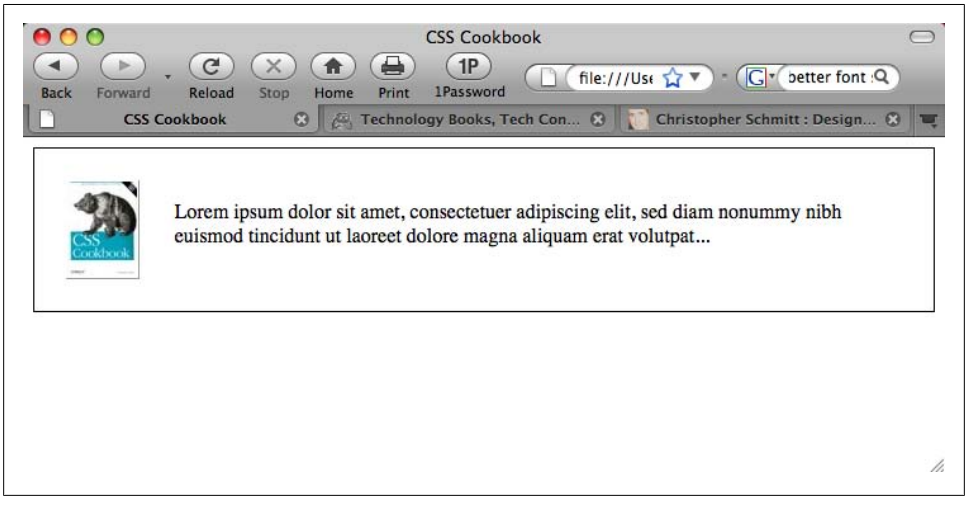



*Figure 2-36. The floated elements, now cleared*

## 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.

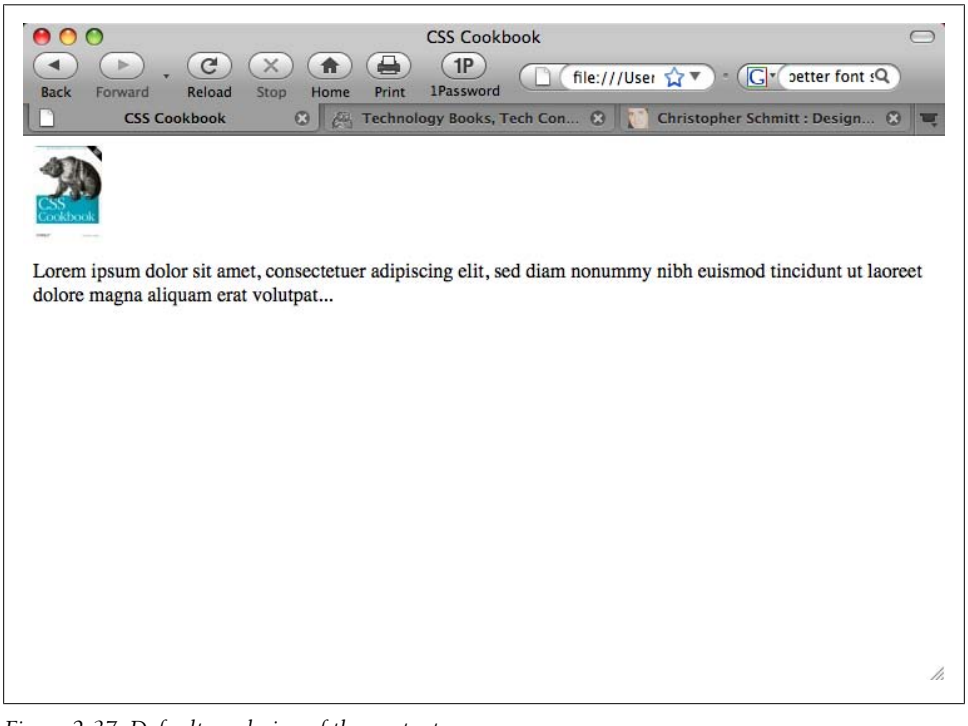



*Figure 2-37. Default rendering of the content*

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>
```

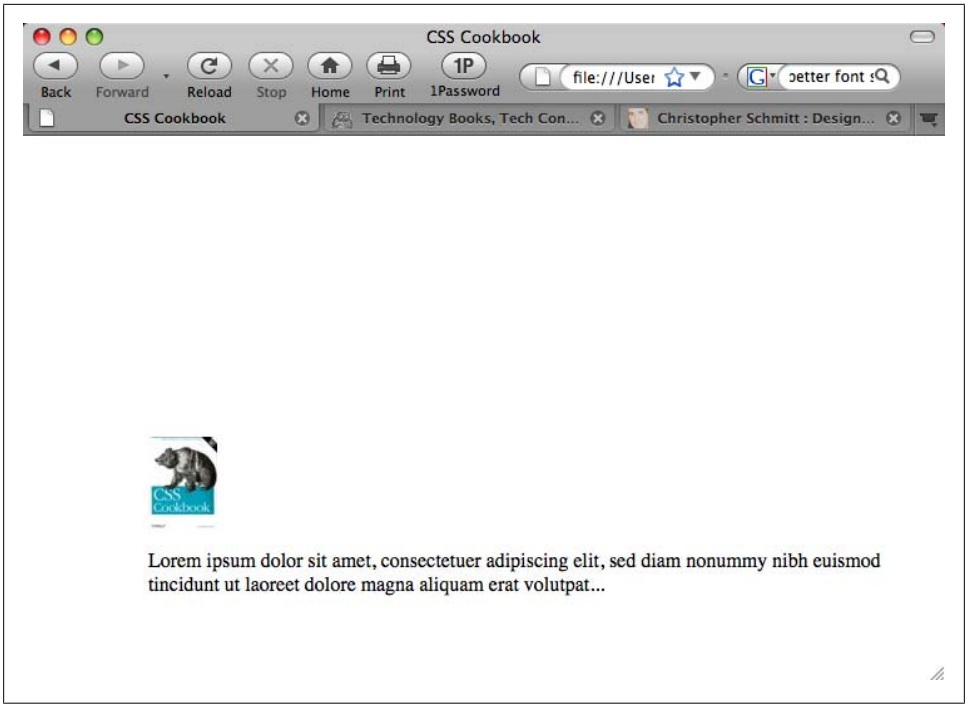
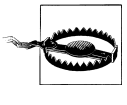



*Figure 2-38. Absolute positioning, which places an element based on its location within a window*

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;
}
```

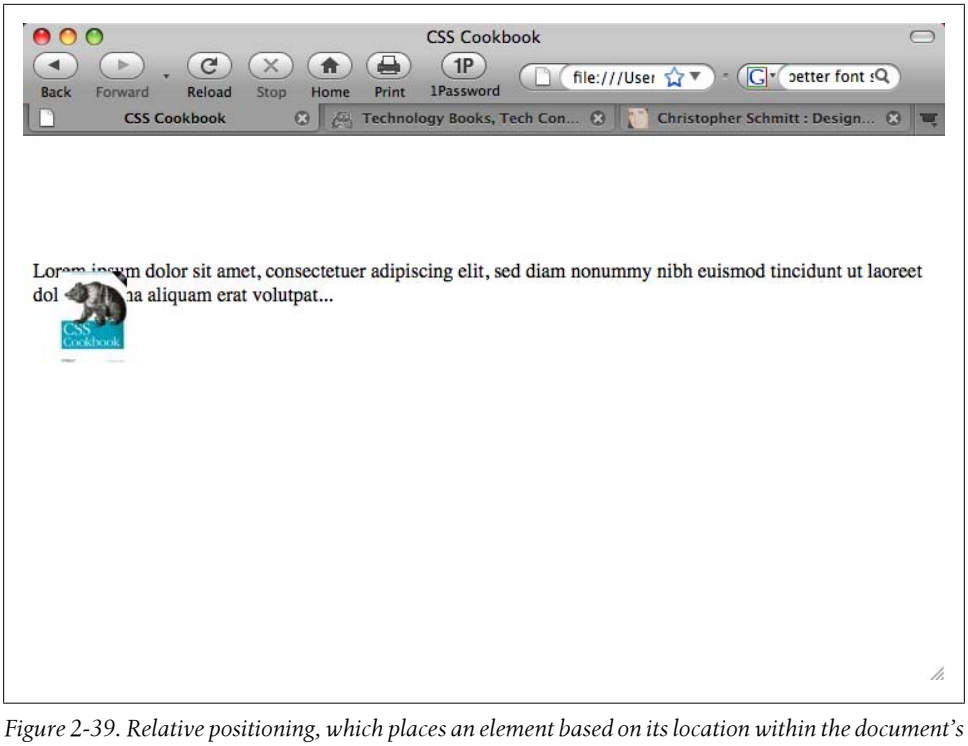



*Figure 2-39. Relative positioning, which places an element based on its location within the document's natural flow*

## 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.

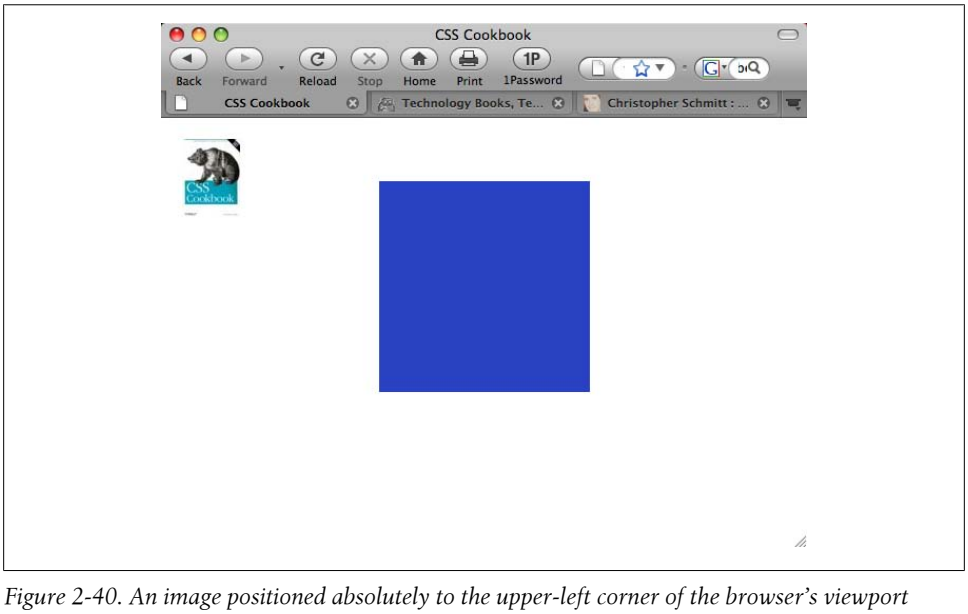



*Figure 2-40. An image positioned absolutely to the upper-left corner of the browser's viewport*

## 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;
}
```

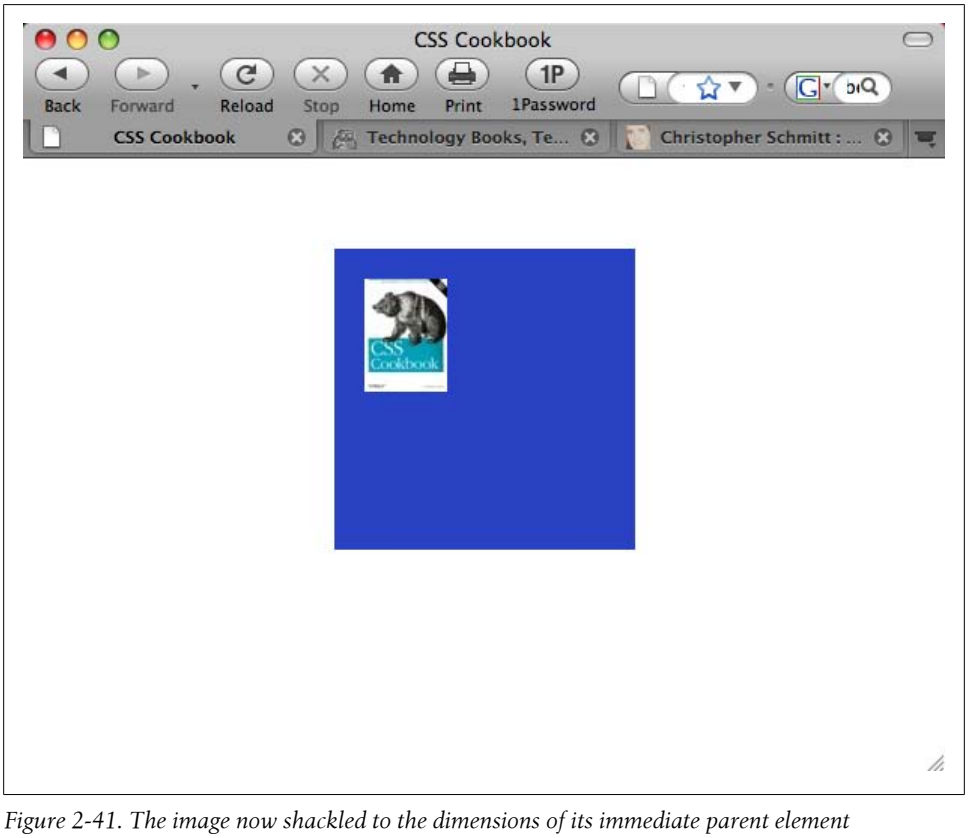



*Figure 2-41. The image now shackled to the dimensions of its immediate parent element*

## 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.

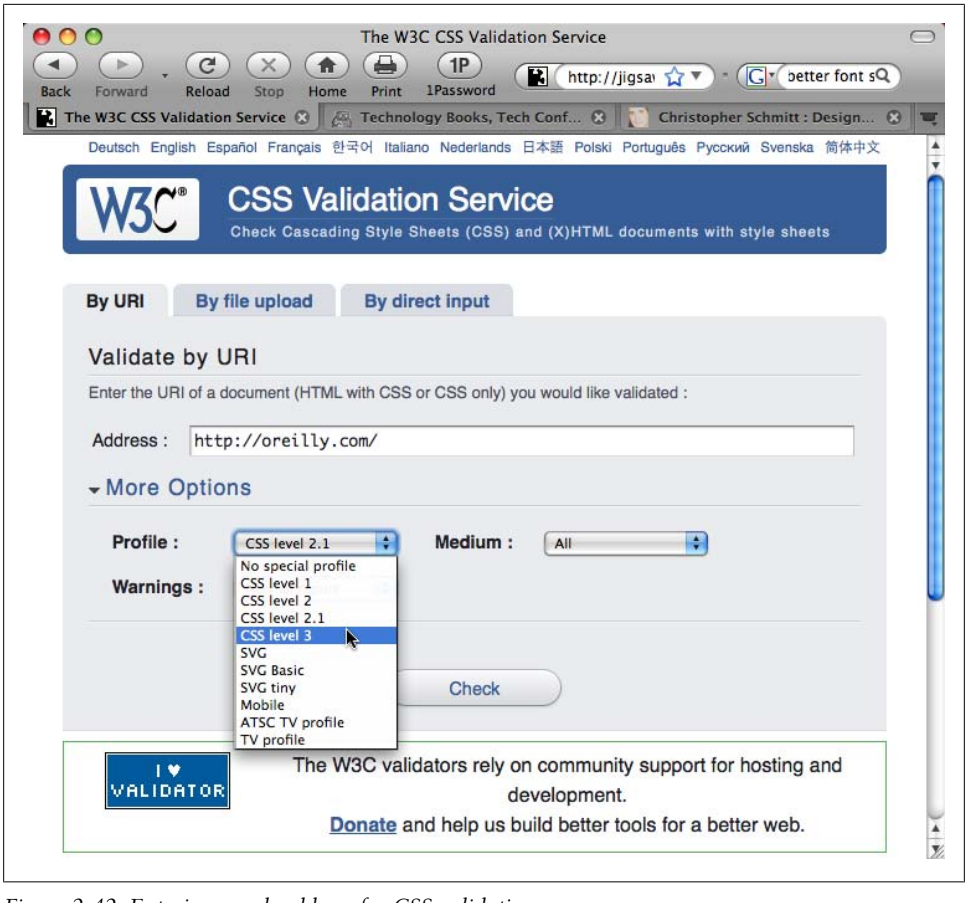



*Figure 2-42. Entering a web address for CSS validation*

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*

CHAPTER 3

# 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.

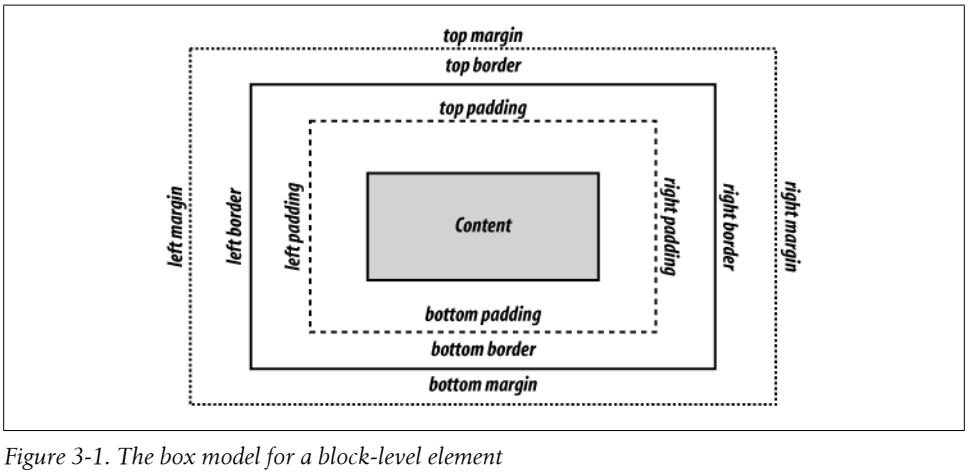



*Figure 3-1. The box model for a block-level element*

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.

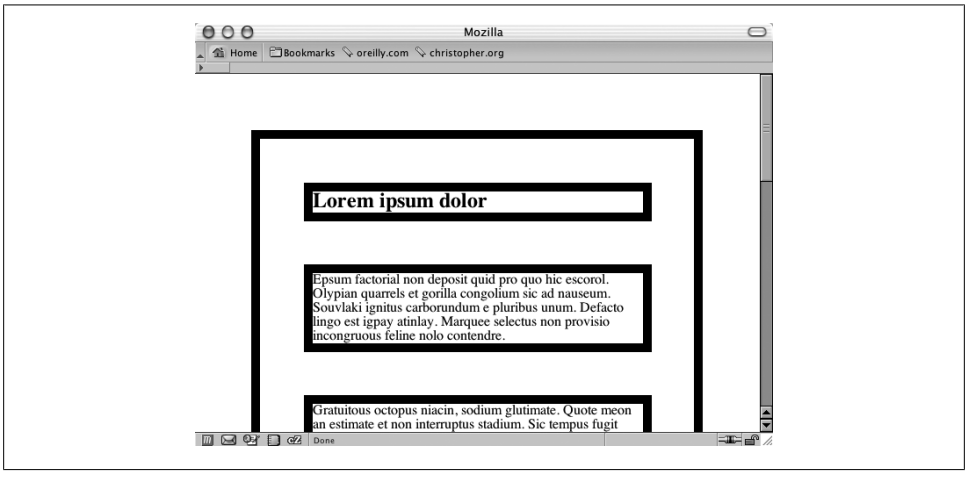



*Figure 3-2. Hypothetical mock-up of margins and border properties being inherited*

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<br>abcdefghijklmnopqrstuvwxyz<br>0123456789 |
| Arial Black | Sans serif | **ABCDEFGHIJKLMNOPQRSTUVWXYZ**<br>**abcdefghijklmnopqrstuvwxyz**<br>**0123456789** |
| Comic Sans MS | Cursive | ABCDEFGHIJKLMNOPQRSTUVWXYZ<br>abcdefghijklmnopqrstuvwxyz<br>0123456789 |
| Courier New | Monospace | ABCDEFGHIJKLMNOPQRSTUVWXYZ<br>abcdefghijklmnopqrstuvwxyz<br>0123456789 |
| Georgia | Serif | ABCDEFGHIJKLMNOPQRSTUVWXYZ<br>abcdefghijklmnopqrstuvwxyz<br>0123456789 |
| Helvetica | Sans serif | ABCDEFGHIJKLMNOPQRSTUVWXYZ<br>abcdefghijklmnopqrstuvwxyz<br>0123456789 |
| Impact | Sans serif | **ABCDEFGHIJKLMNOPQRSTUVWXYZ**<br>**abcdefghijklmnopqrstuvwxyz**<br>**0123456789** |
| Tahoma | Sans serif | ABCDEFGHIJKLMNOPQRSTUVWXYZ<br>abcdefghijklmnopqrstuvwxyz<br>0123456789 |
| Times | Serif | ABCDEFGHIJKLMNOPQRSTUVWXYZ<br>abcdefghijklmnopqrstuvwxyz<br>0123456789 |
| Times New Roman | Serif | ABCDEFGHIJKLMNOPQRSTUVWXYZ<br>abcdefghijklmnopqrstuvwxyz<br>0123456789 |
| Trebuchet MS | Sans serif | ABCDEFGHIJKLMNOPQRSTUVWXYZ<br>abcdefghijklmnopqrstuvwxyz<br>0123456789 |
| Verdana | Sans serif | ABCDEFGHIJKLMNOPQRSTUVWXYZ<br>abcdefghijklmnopqrstuvwxyz<br>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*.

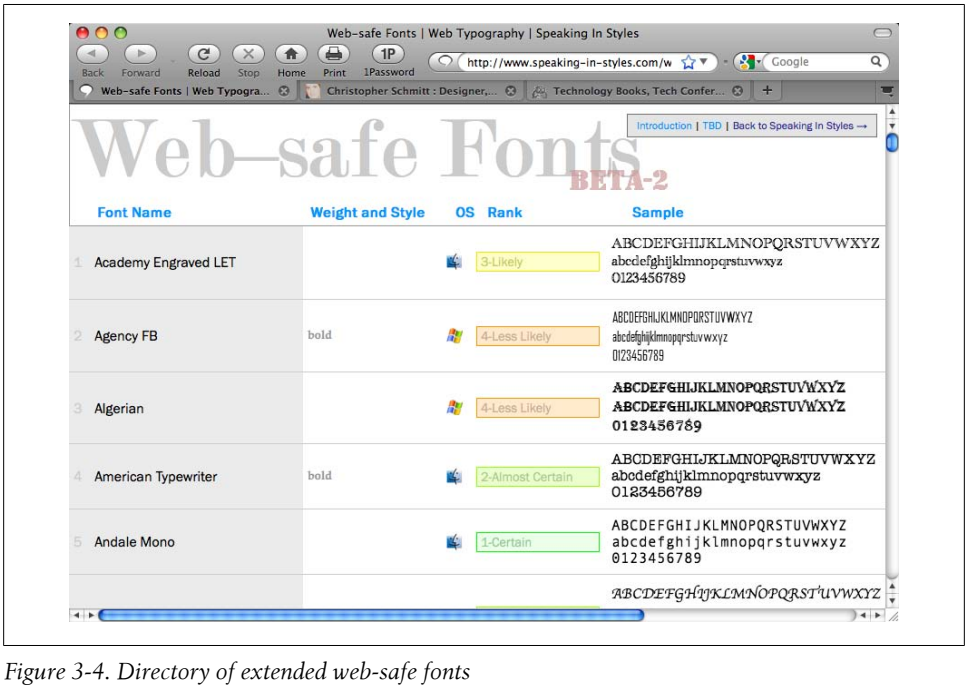



*Figure 3-4. Directory of extended web-safe fonts*

#### 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.

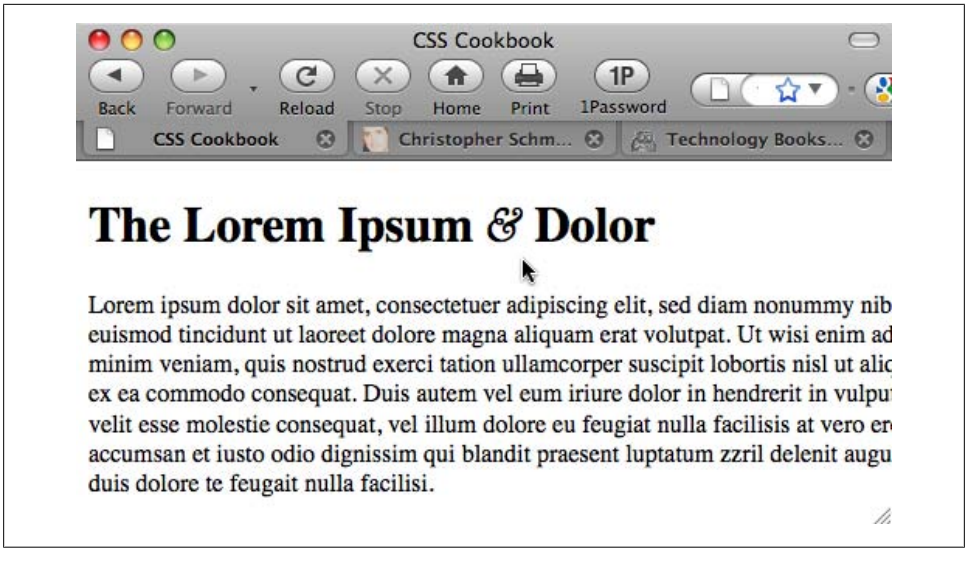



*Figure 3-5. The ampersand style changes to a more distinguished look*

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/*

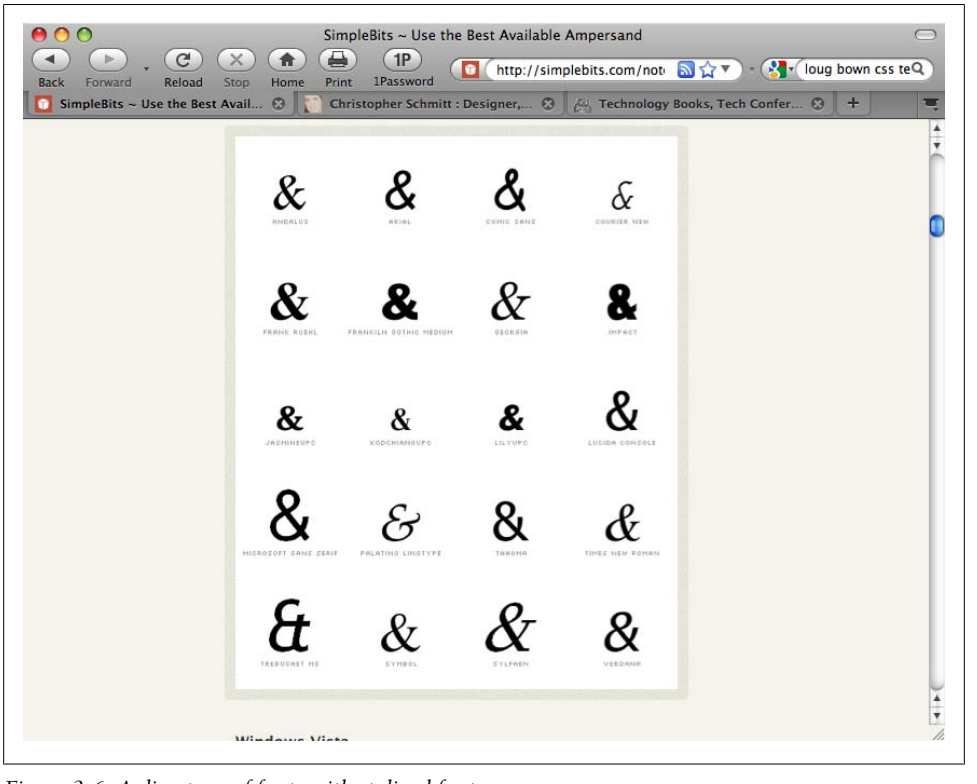



*Figure 3-6. A directory of fonts with stylized fonts*

# 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;
}
```

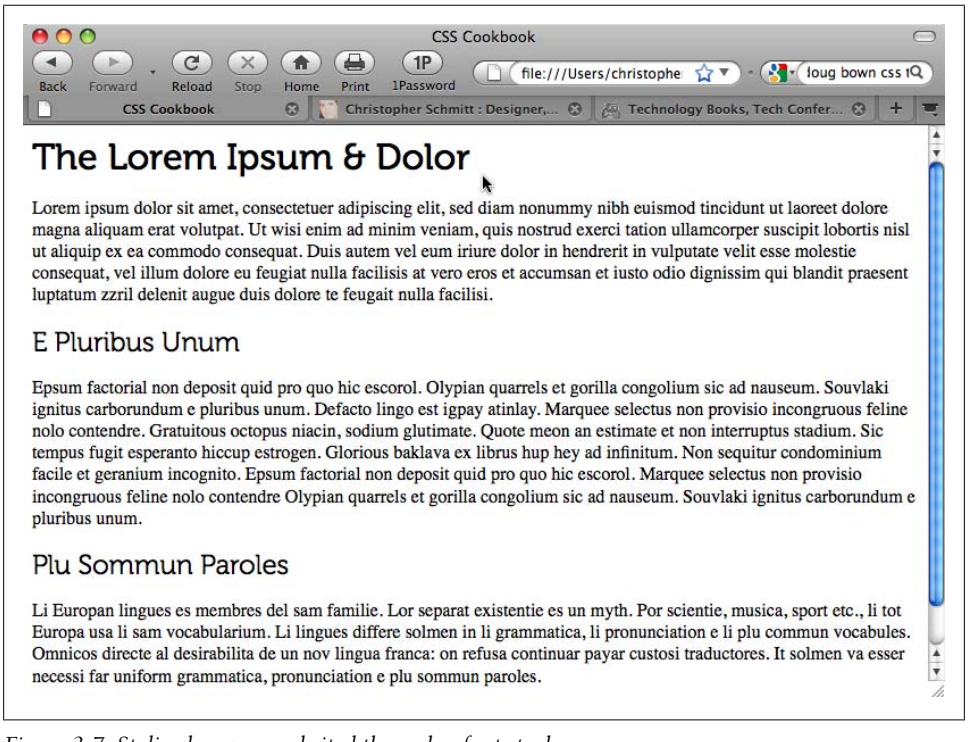



*Figure 3-7. Stylized ampersand cited through a font stack*

## 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.

*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.fontsquirrel.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-protecting-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;
}
```

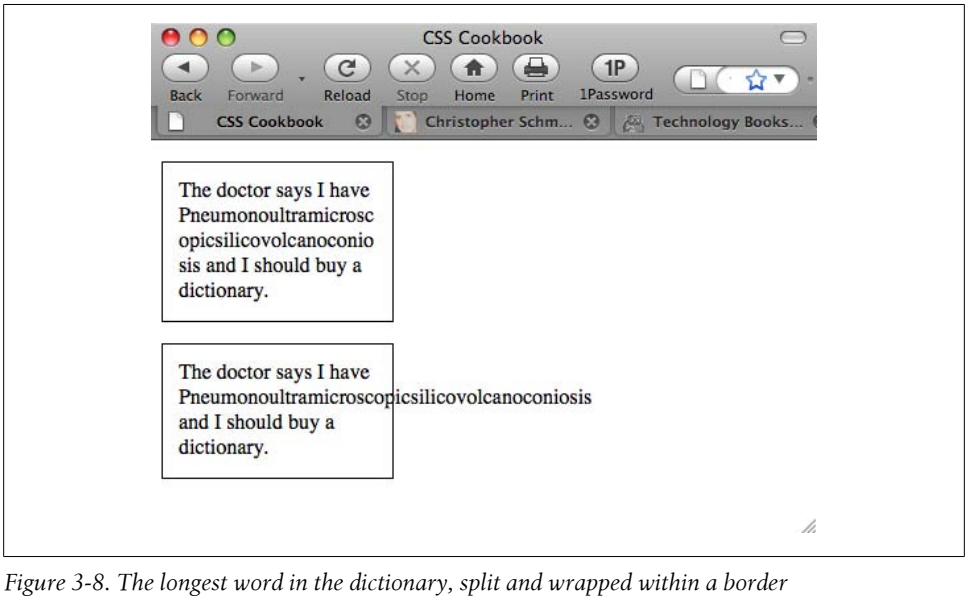



*Figure 3-8. The longest word in the dictionary, split and wrapped within a border*

## 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*

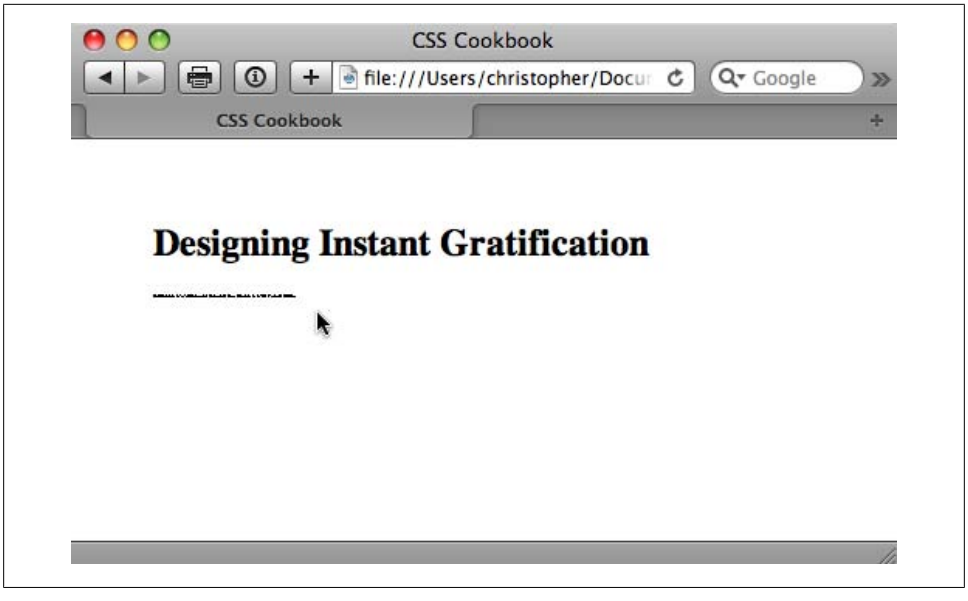



*Figure 3-9. Safari 4 for Macintosh showing illegible type when the font size is set to zero*

# 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.

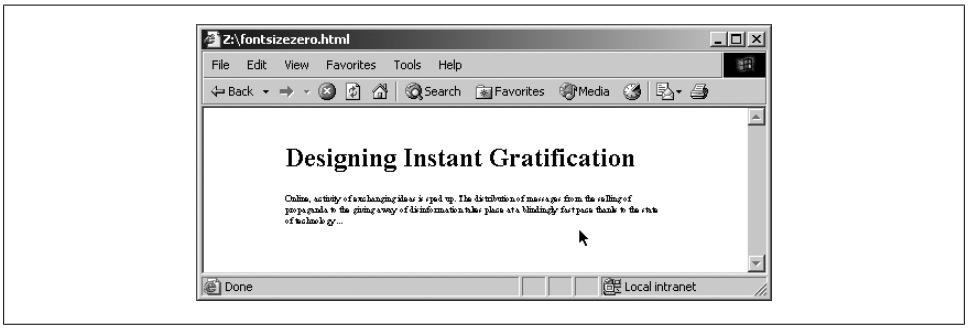



*Figure 3-10. Almost illegible type set with em units*

### 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
```

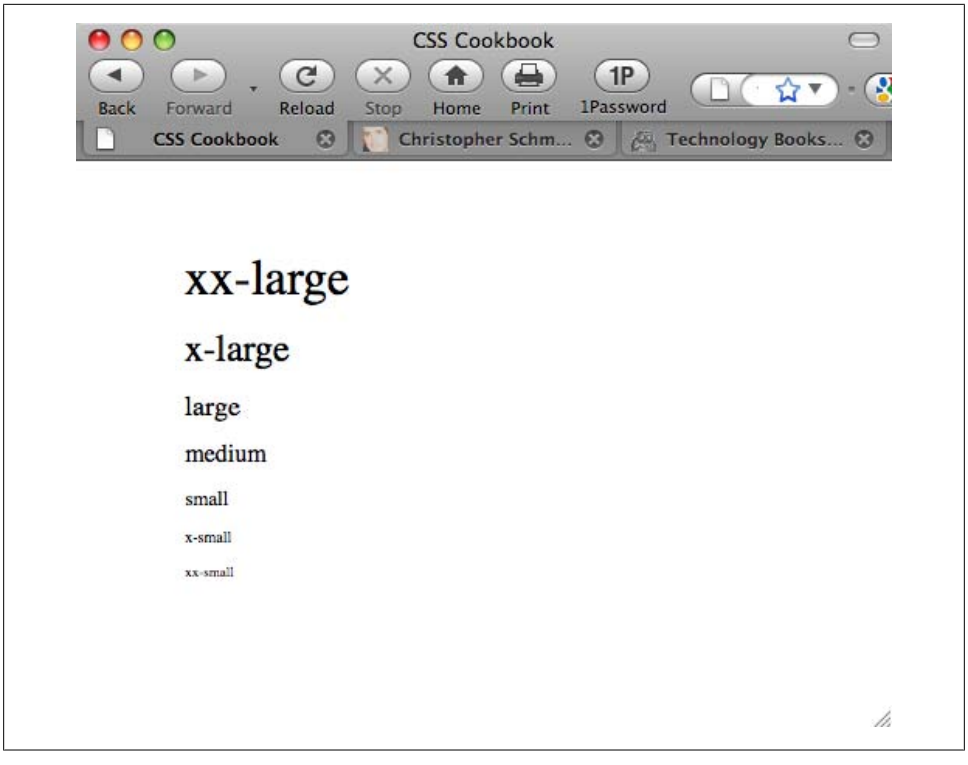



*Figure 3-11. The font-size keywords on display*

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.

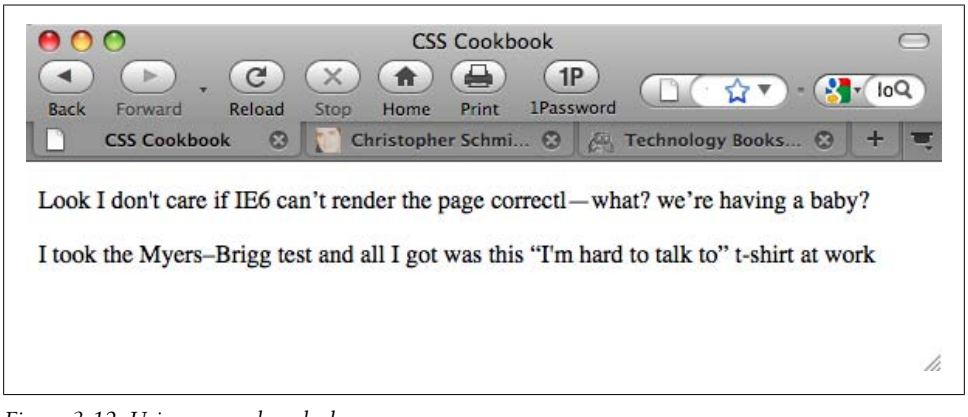



*Figure 3-12. Using em and en dashes*

## Solution

Use the em dash with the decimal representation `&#8212;`:

```
<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 `&#8211;`:

```
<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.

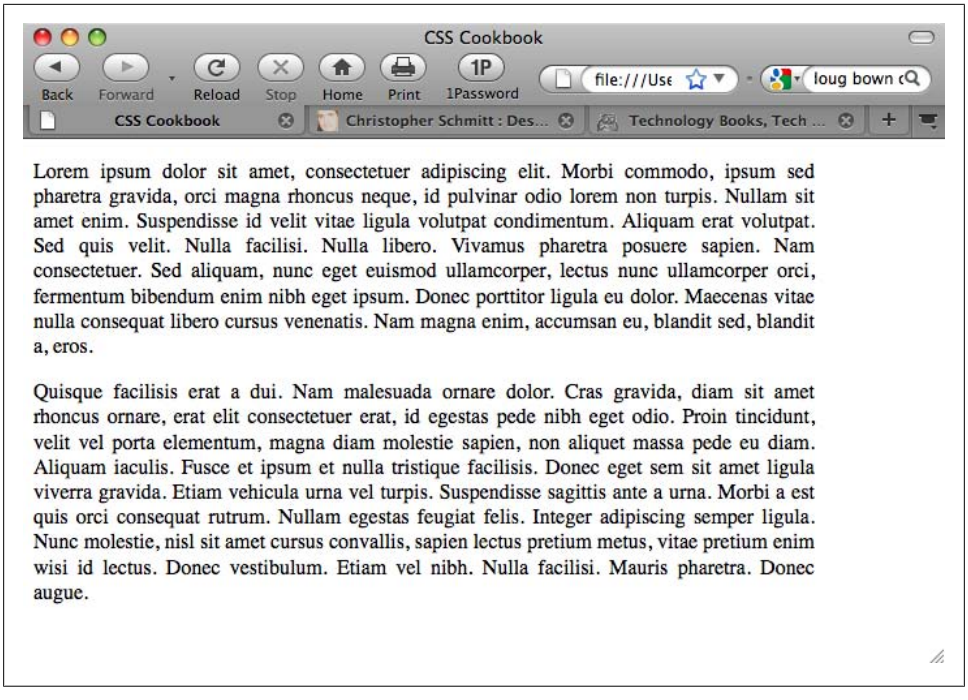



*Figure 3-13. A paragraph justified on both sides*

## 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.

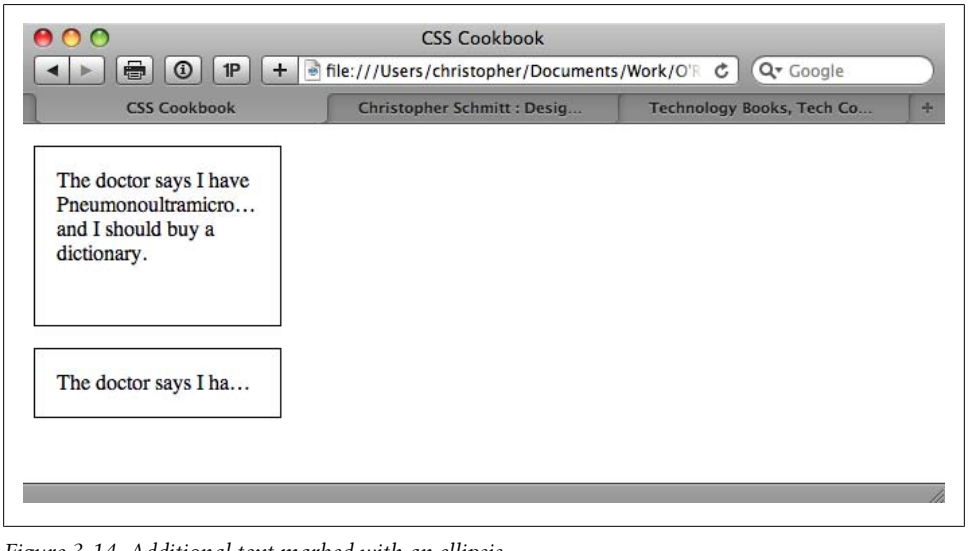



*Figure 3-14. Additional text marked with an ellipsis*

### 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;
}
```

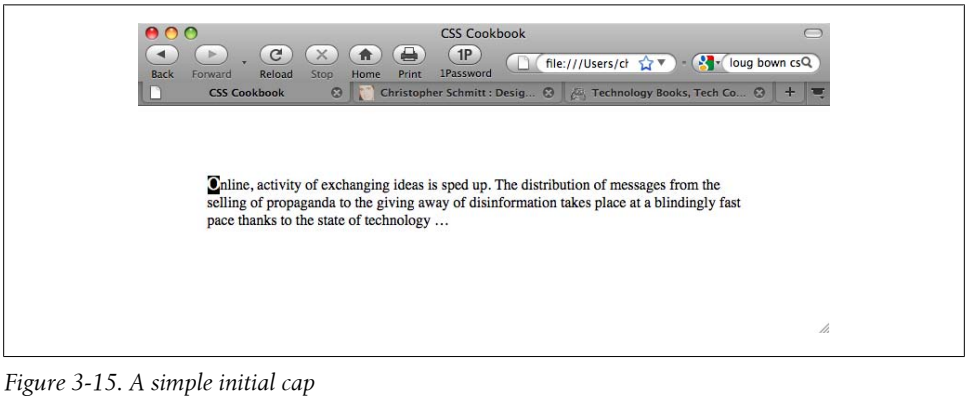



*Figure 3-15. A simple initial cap*

## 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;
}
```

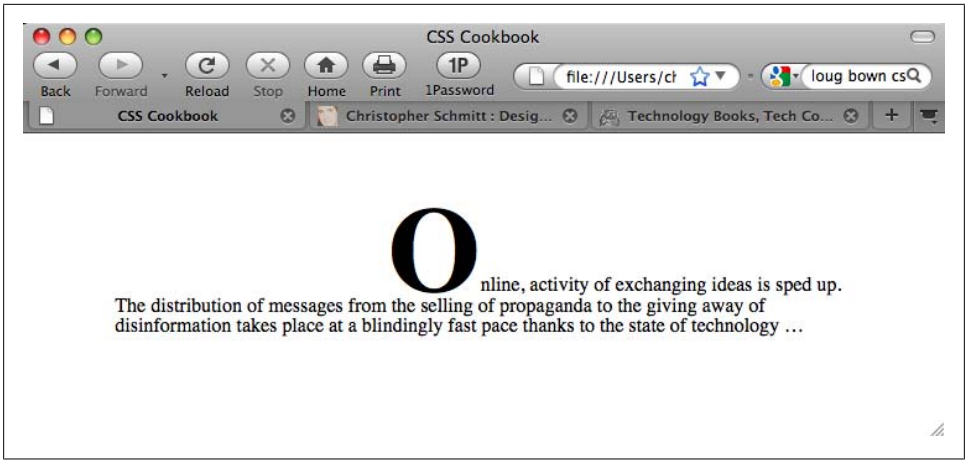



*Figure 3-16. A larger, centered initial cap*

## 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.

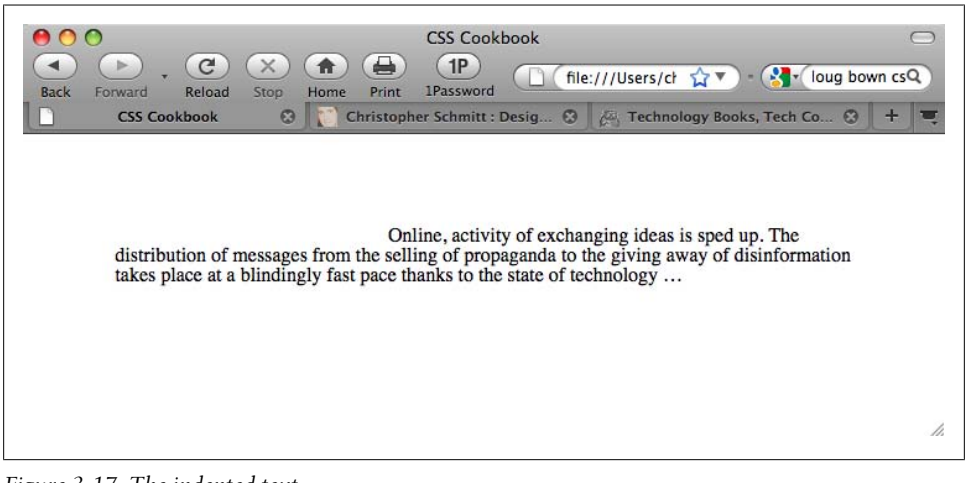



*Figure 3-17. The indented text*

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).

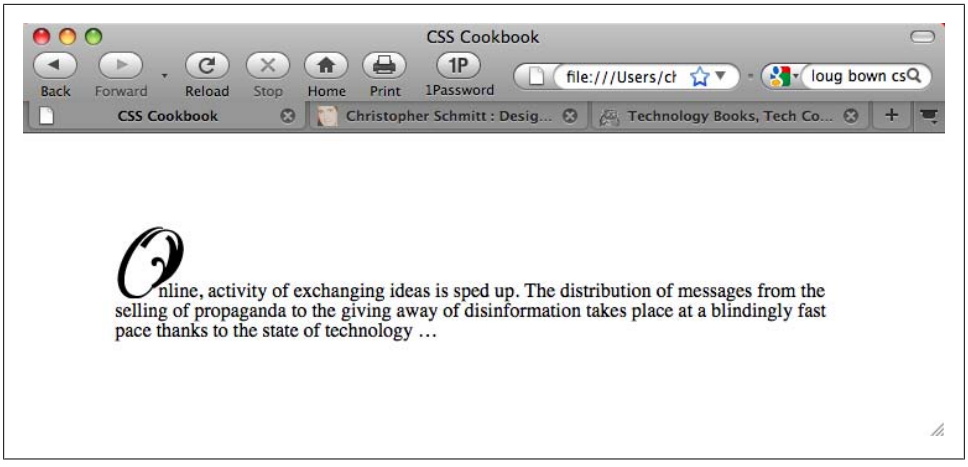



*Figure 3-18. An image used as an initial cap*

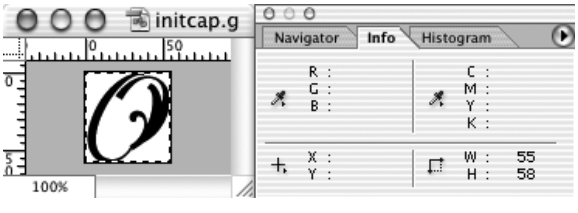



*Figure 3-19. The image of the initial cap*

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.

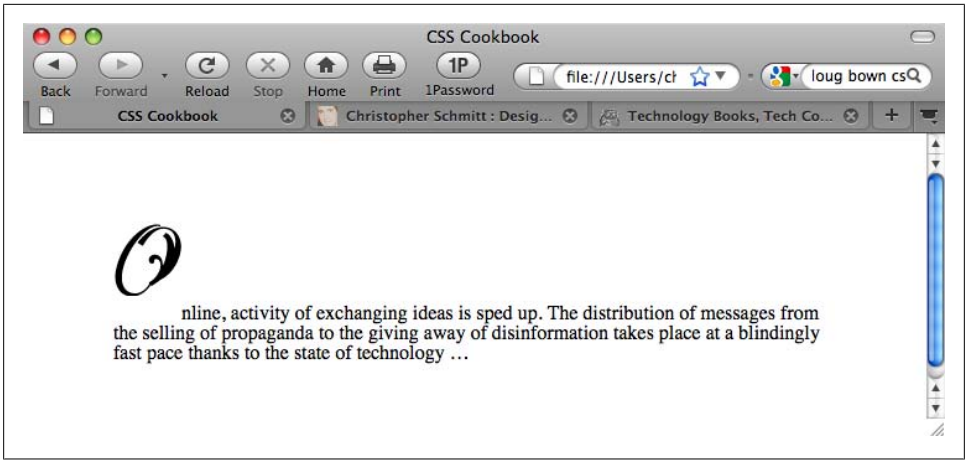



*Figure 3-20. Adjusting the space for the initial cap*

#### 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.

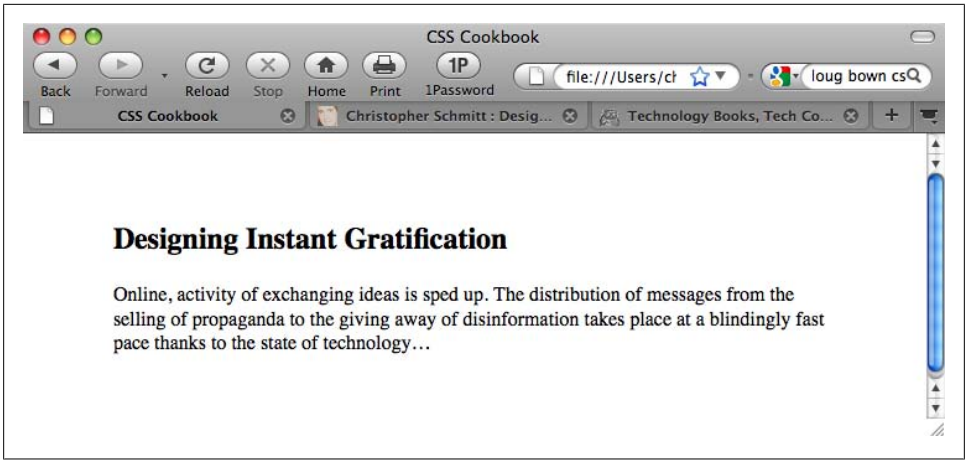



*Figure 3-21. The default rendering of a heading*

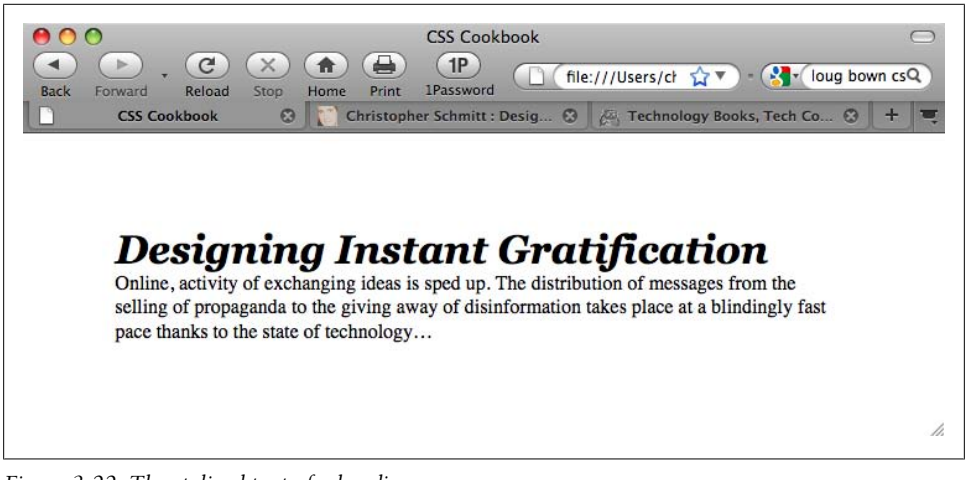



*Figure 3-22. The stylized text of a heading*

## 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.

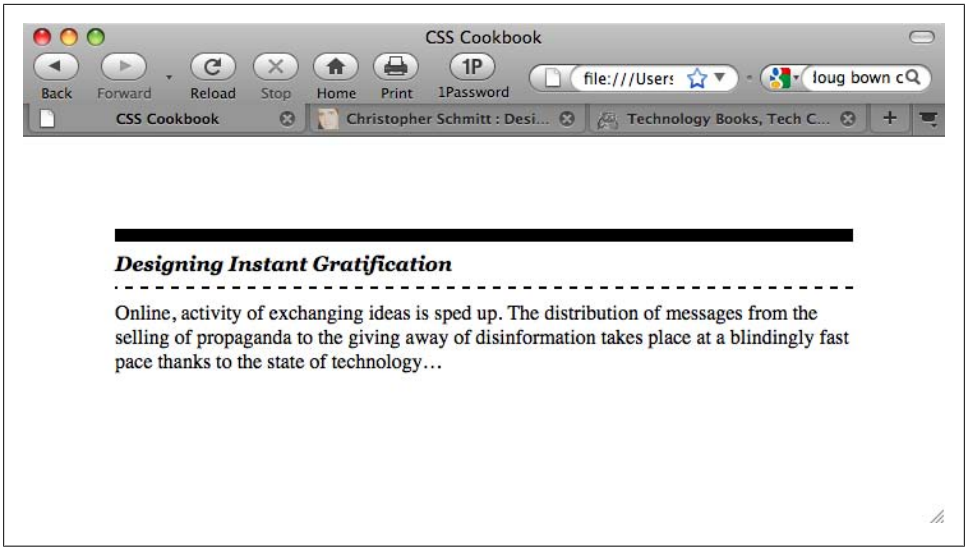



*Figure 3-23. A heading stylized with borders*

## 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;
}
```

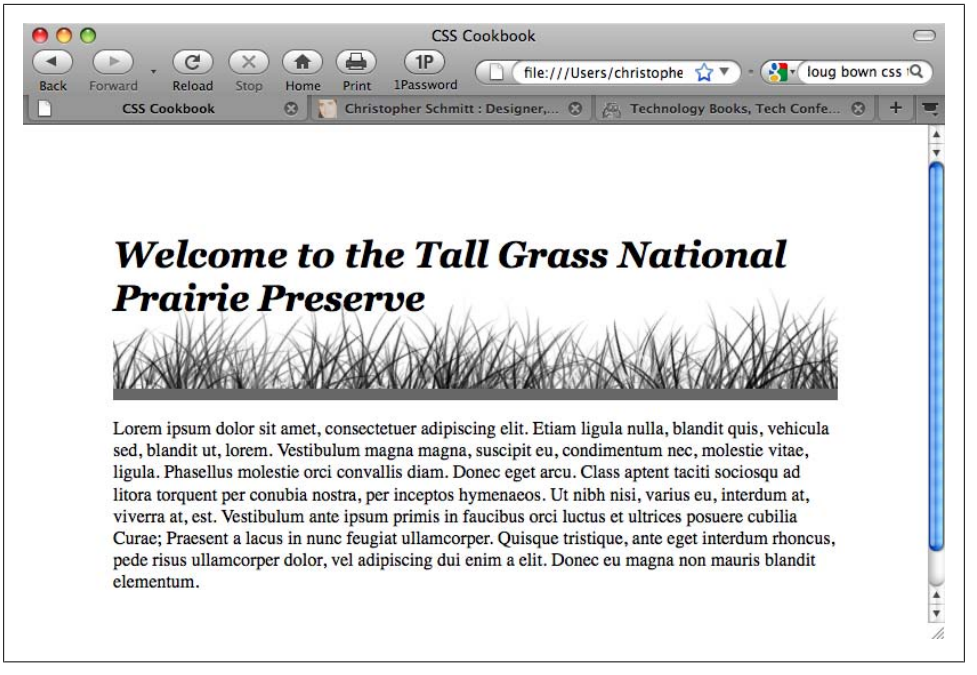



*Figure 3-24. A background image used with a heading*

## 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;
```

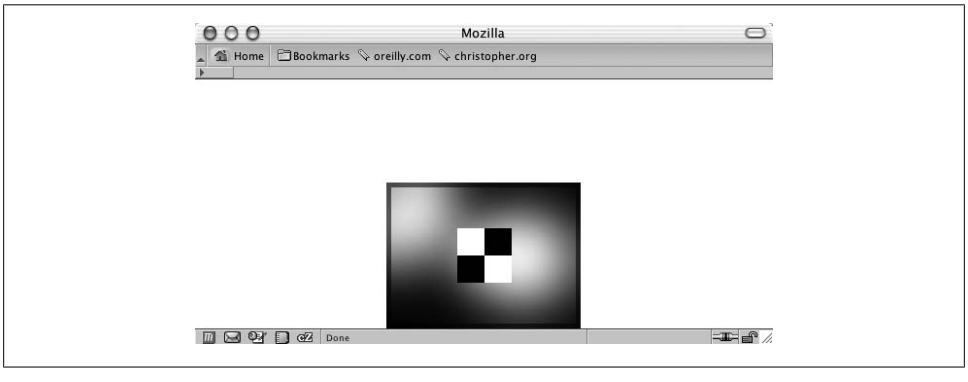



*Figure 3-26. The image aligned on the bottom of the y-axis and in the middle of the x-axis*

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).

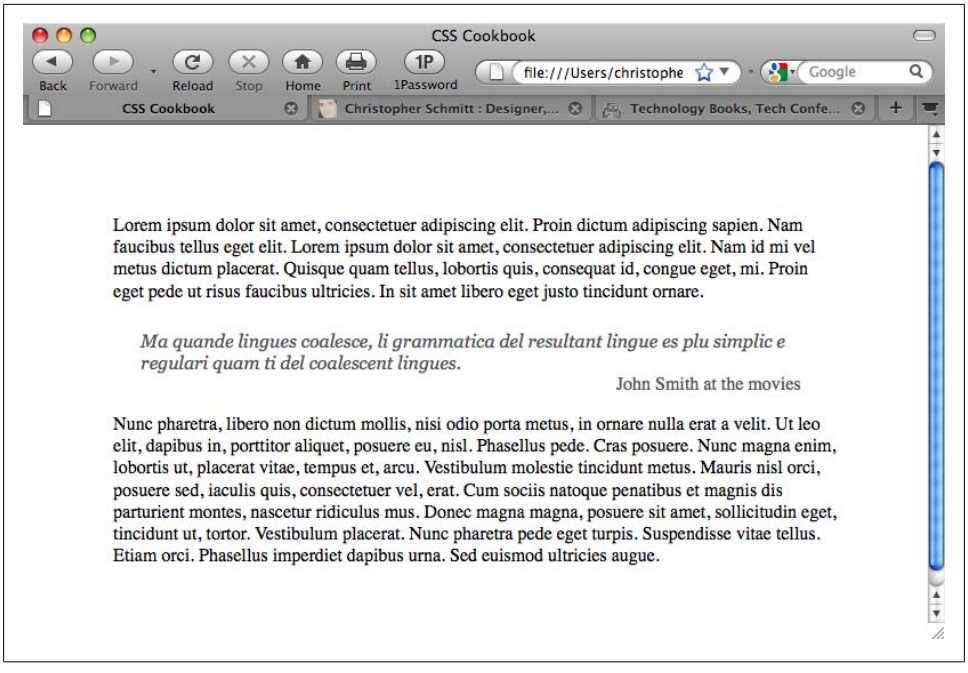



*Figure 3-27. A stylized pull quote*

## 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;
}
```

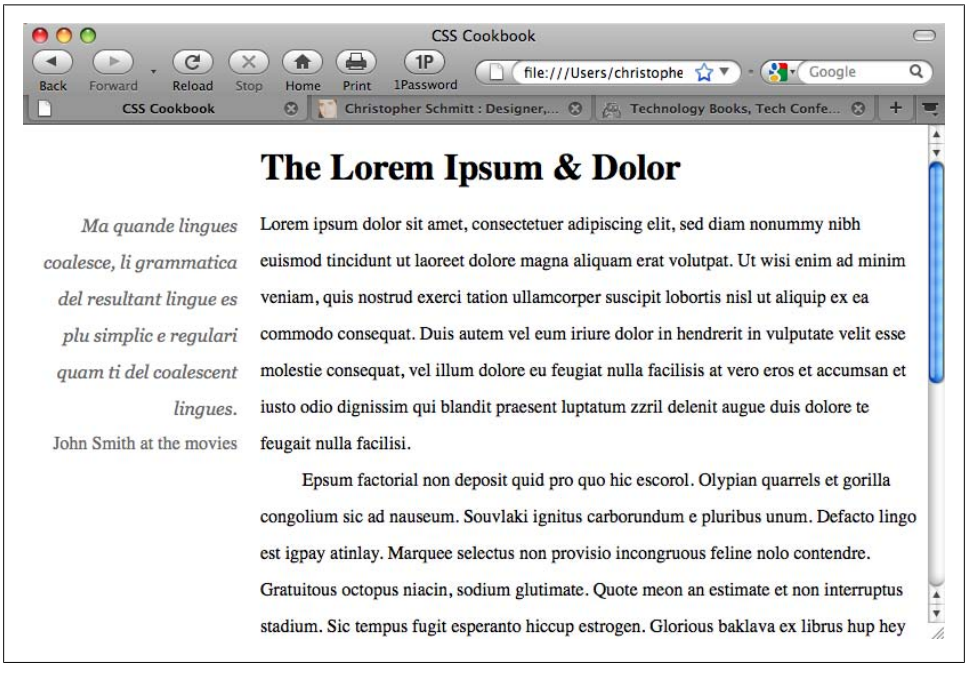



*Figure 3-28. A pull quote to the left of a column*

## 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.

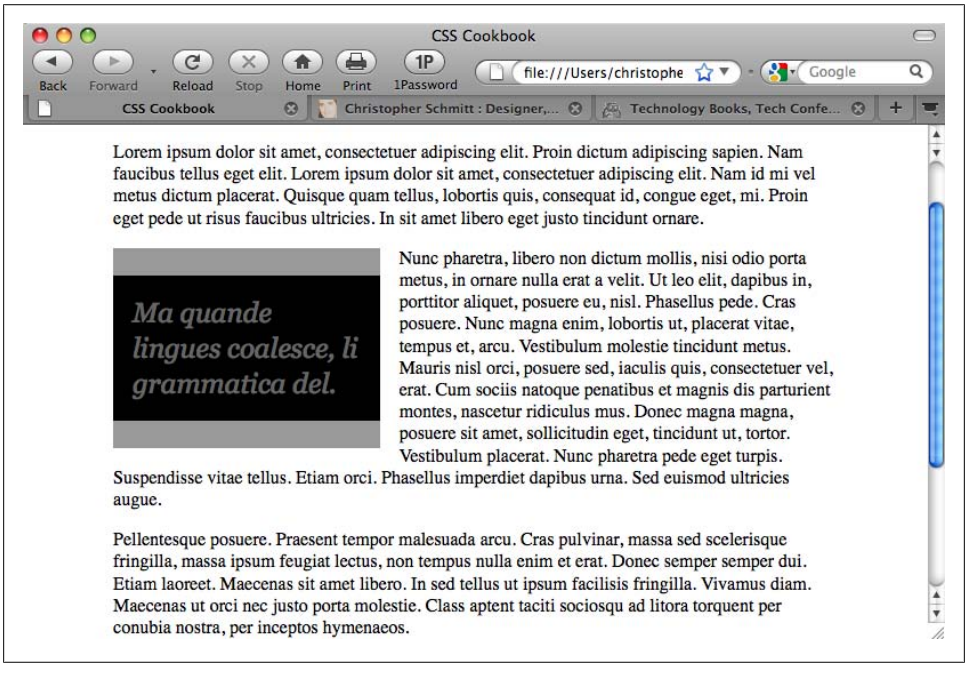



*Figure 3-29. A stylized pull quote using borders*

## 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.

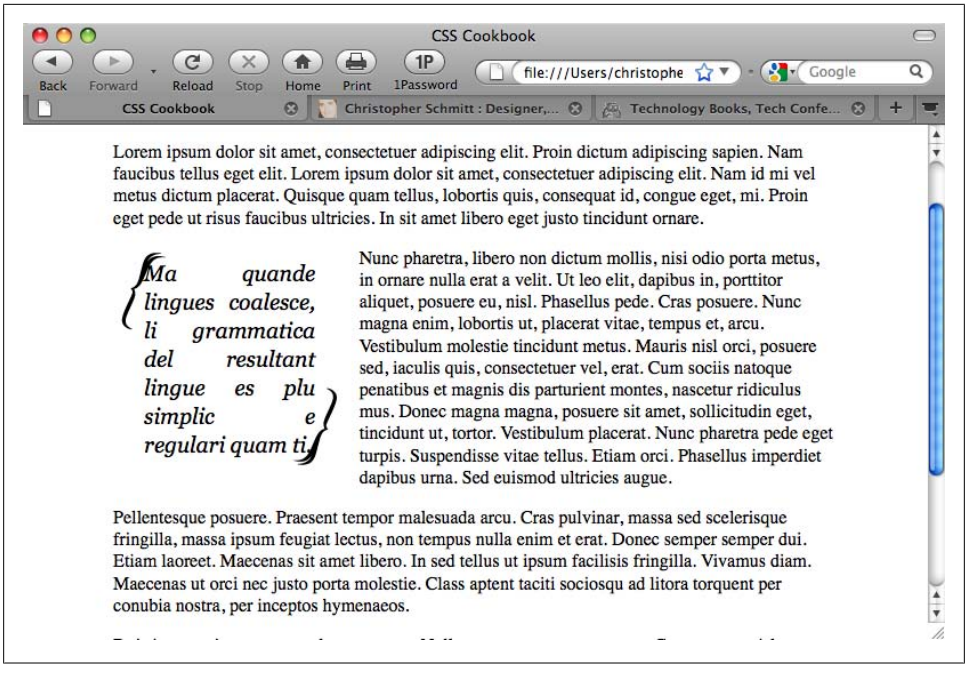



*Figure 3-30. A pull quote with images*

## 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.

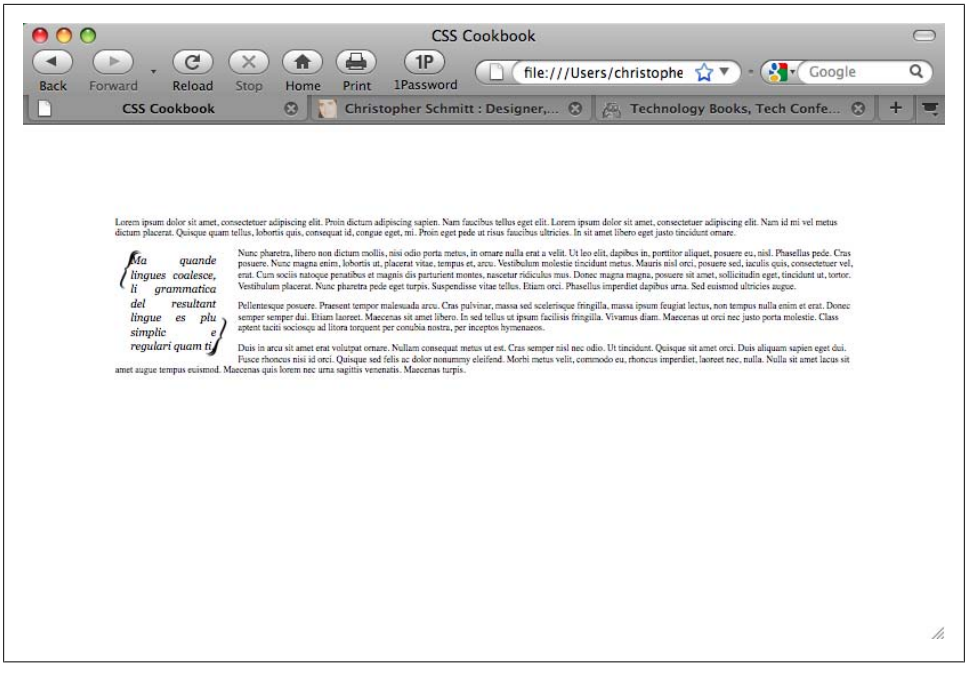



*Figure 3-31. The background images staying in the corners as the text is resized*

## 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;
}
```

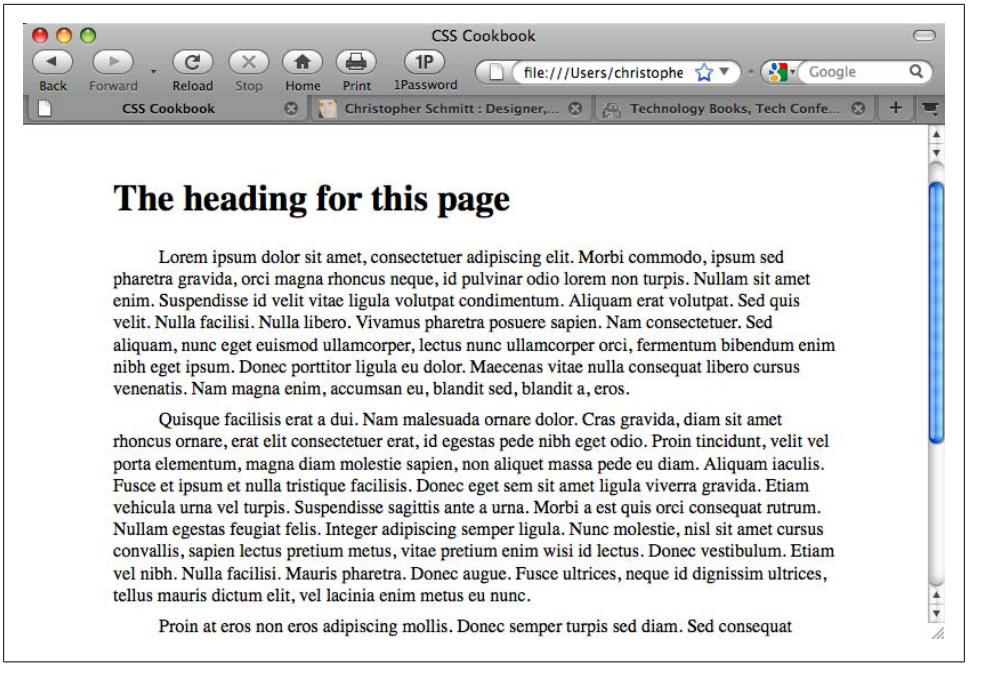



*Figure 3-32. Paragraphs with first lines indented*

## 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.

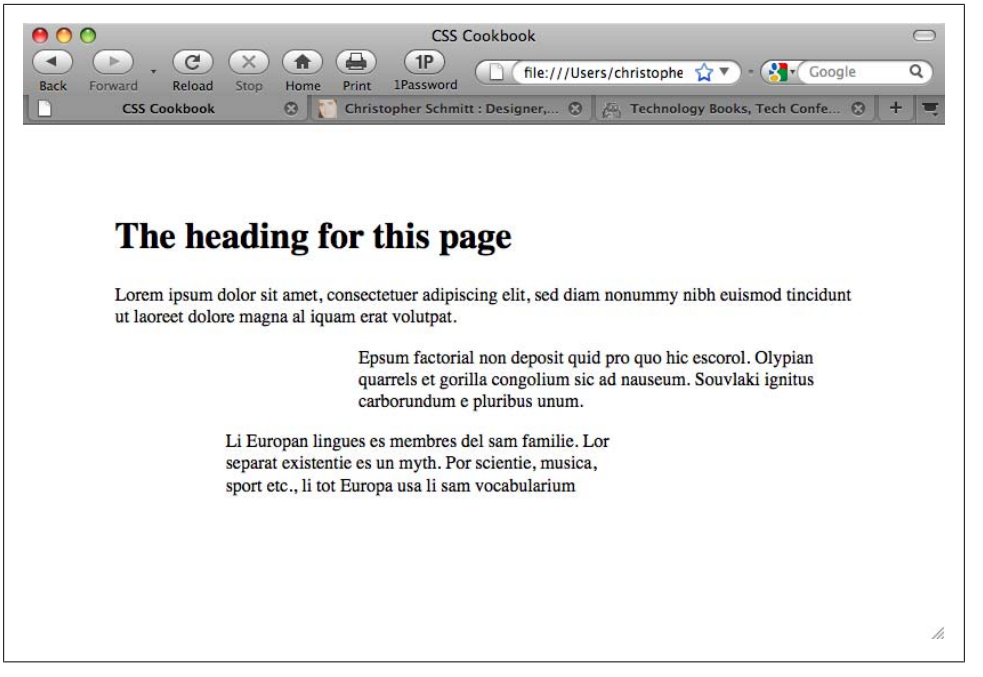



*Figure 3-33. Indented paragraphs*

## 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>
```

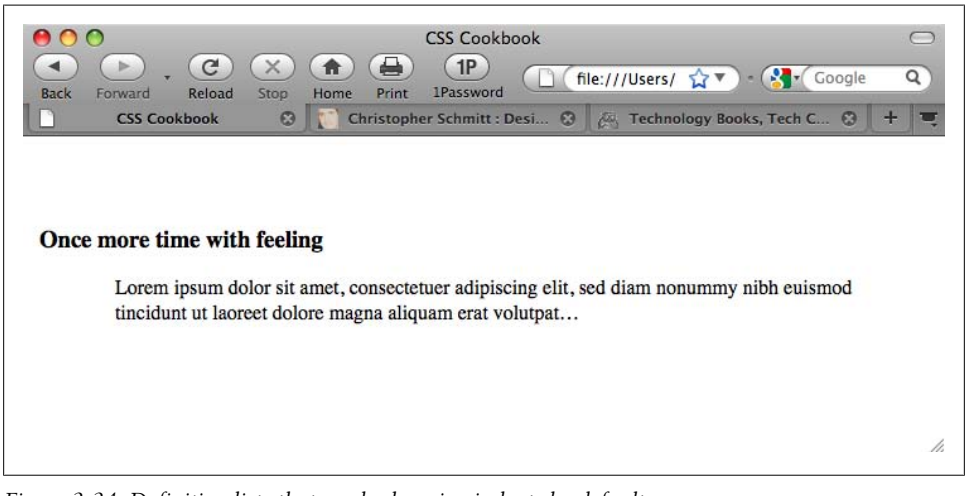



*Figure 3-34. Definition lists that render hanging indents by default*

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:

```
p.hanging {
 text-indent: -5em;
 margin-left: 5em;
}
```

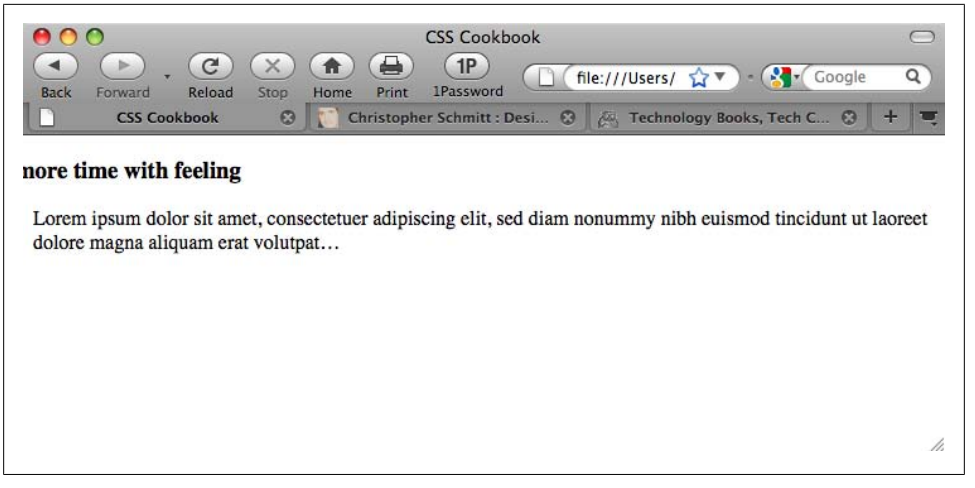



*Figure 3-35. The hanging indent, exiting stage left*

### 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:

```
#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:

```
<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.

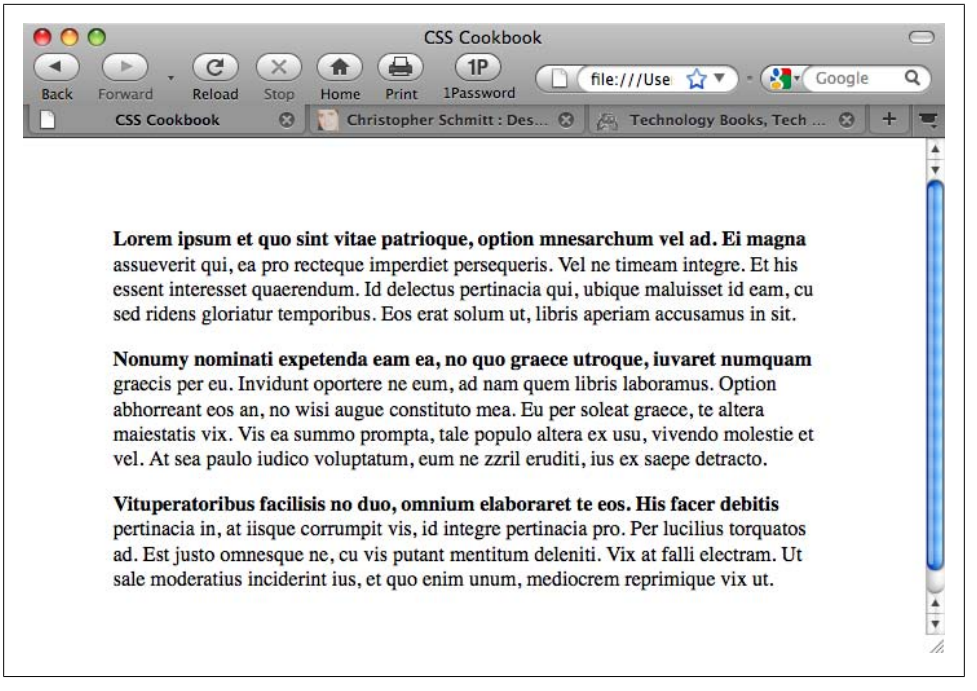



*Figure 3-36. The first line set to bold*

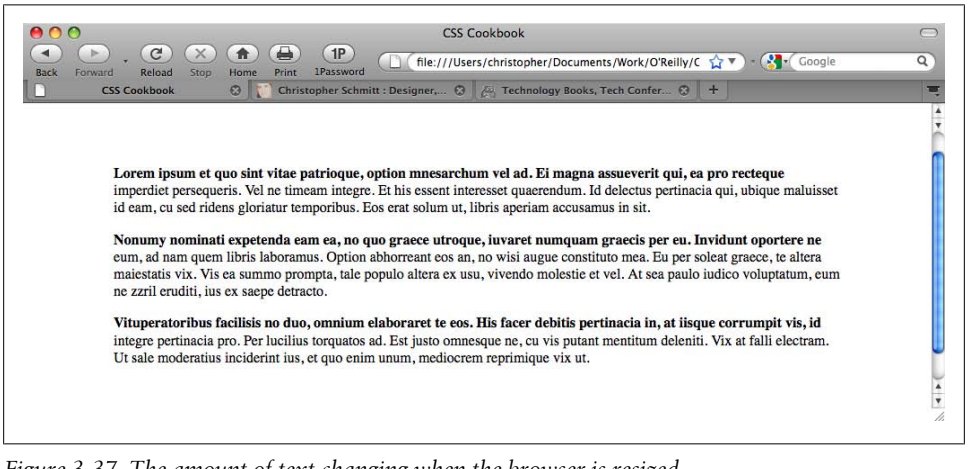



*Figure 3-37. The amount of text changing when the browser is resized*

## 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.

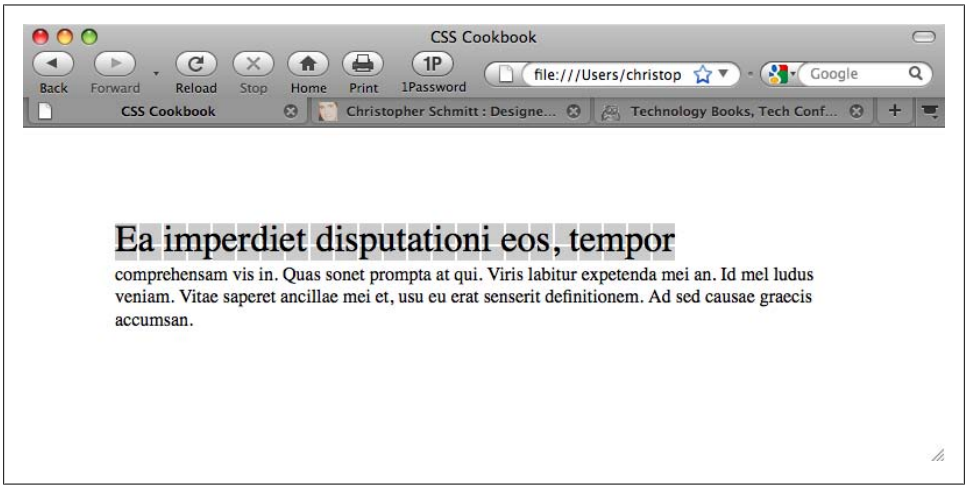



*Figure 3-38. The first line with a background image*

## 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.

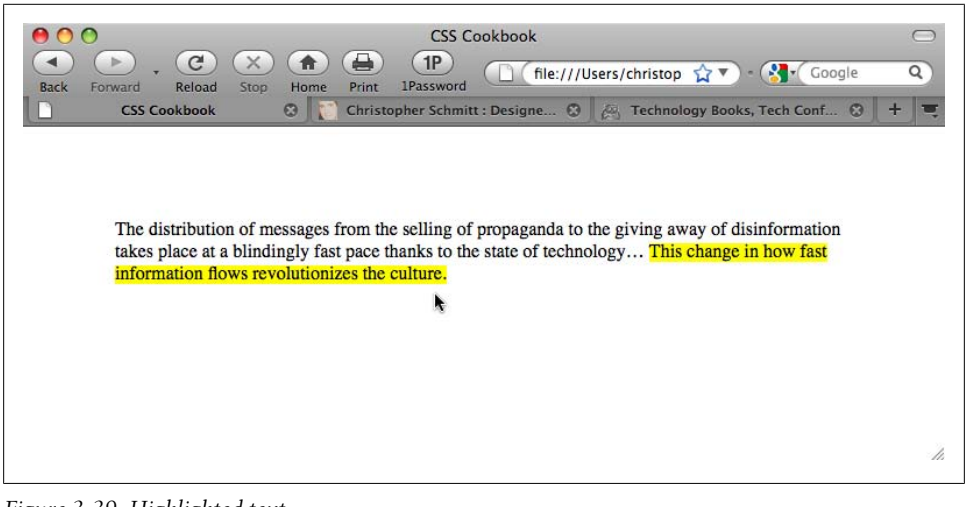



*Figure 3-39. Highlighted text*

### 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.

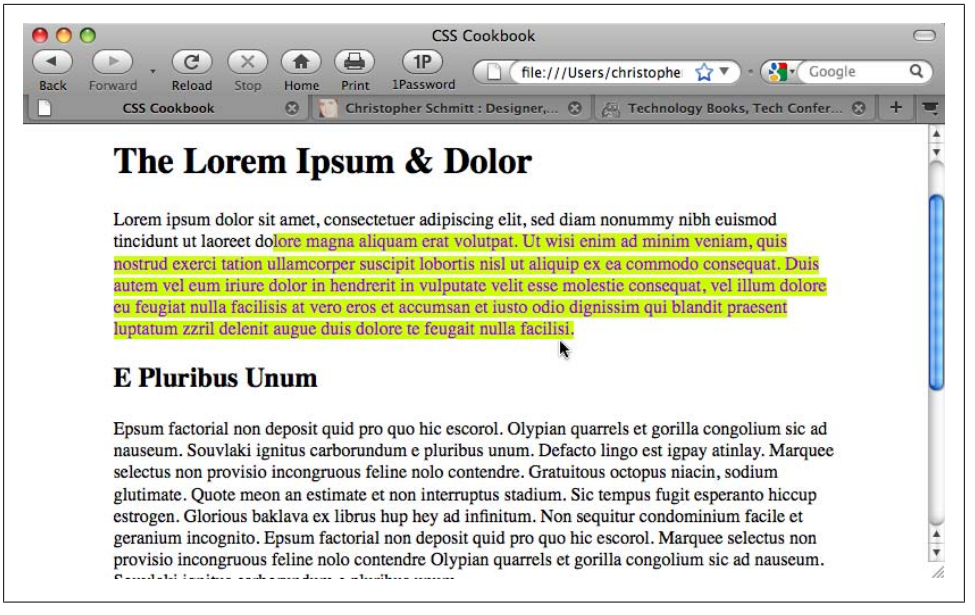



*Figure 3-40. Color set when selecting a passage of text with the mouse*

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.

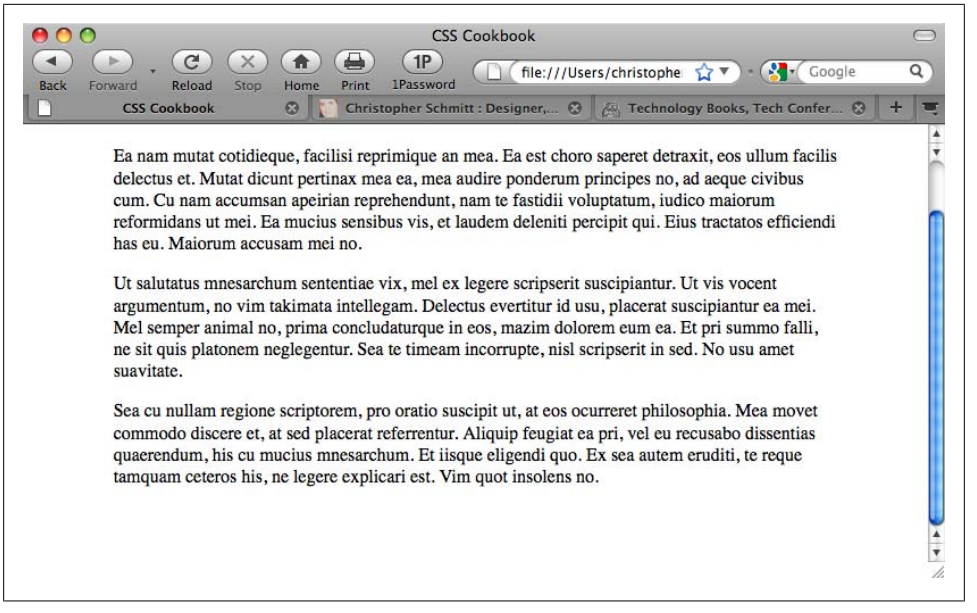



*Figure 3-41. The default leading of a paragraph*

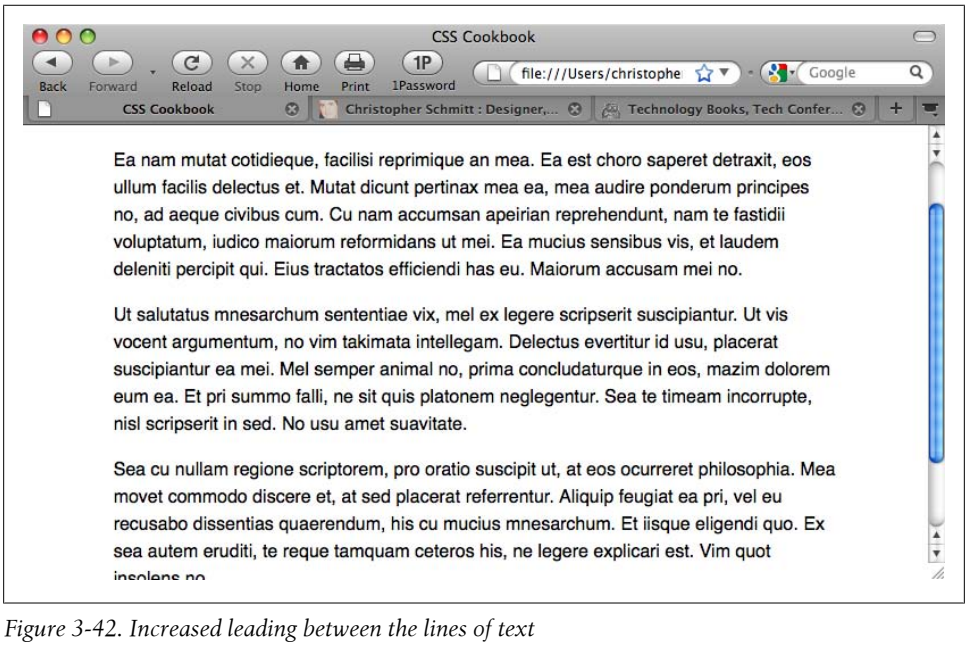



*Figure 3-42. Increased leading between the lines of text*

## 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.

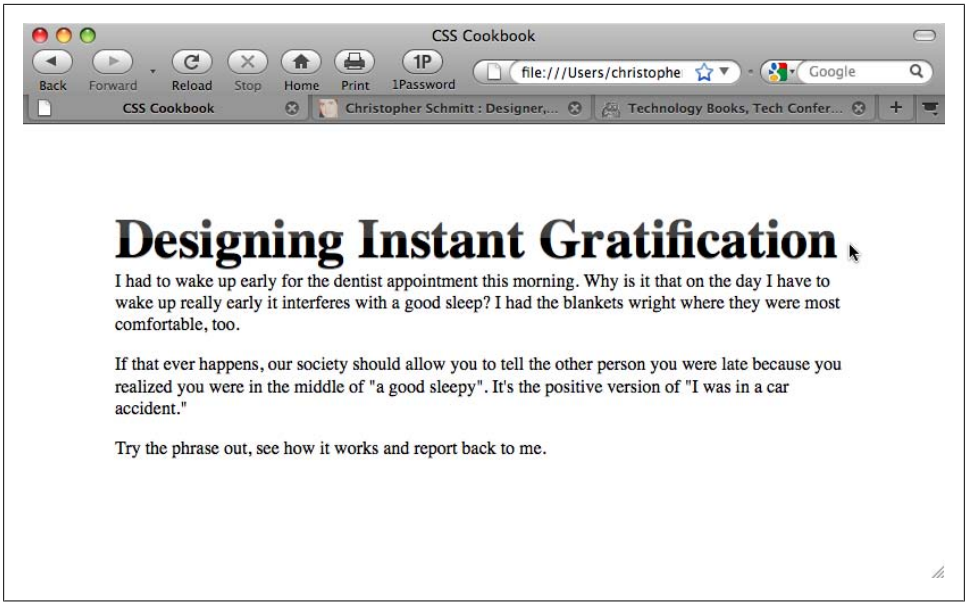



*Figure 3-43. A PNG image repeating over the top half of the HTML text to create a glassy appearance*

## 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.mezzoblue.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.

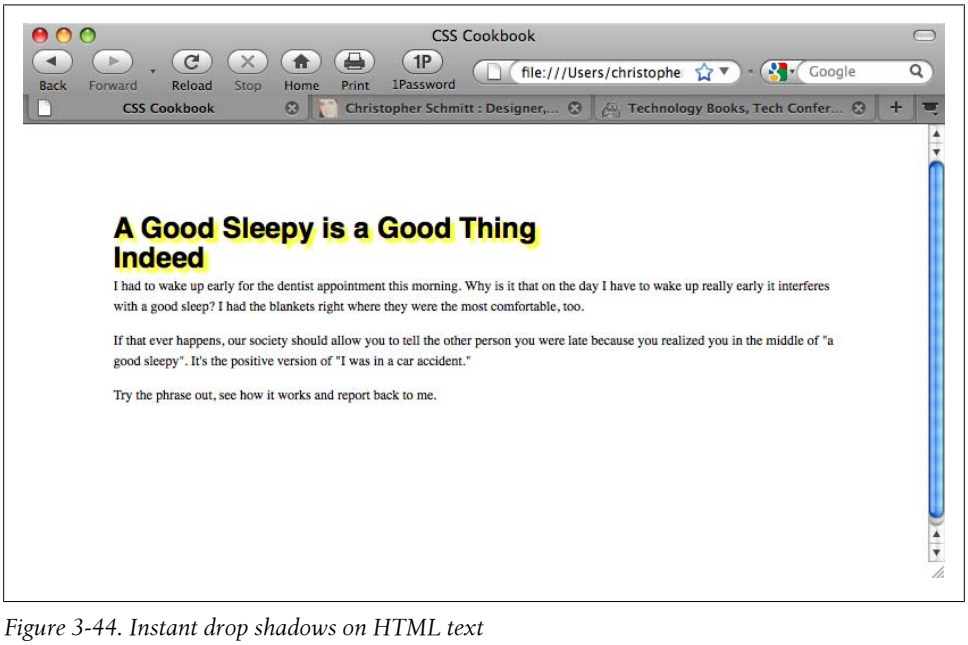



*Figure 3-44. Instant drop shadows on HTML text*

## 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);
}
```

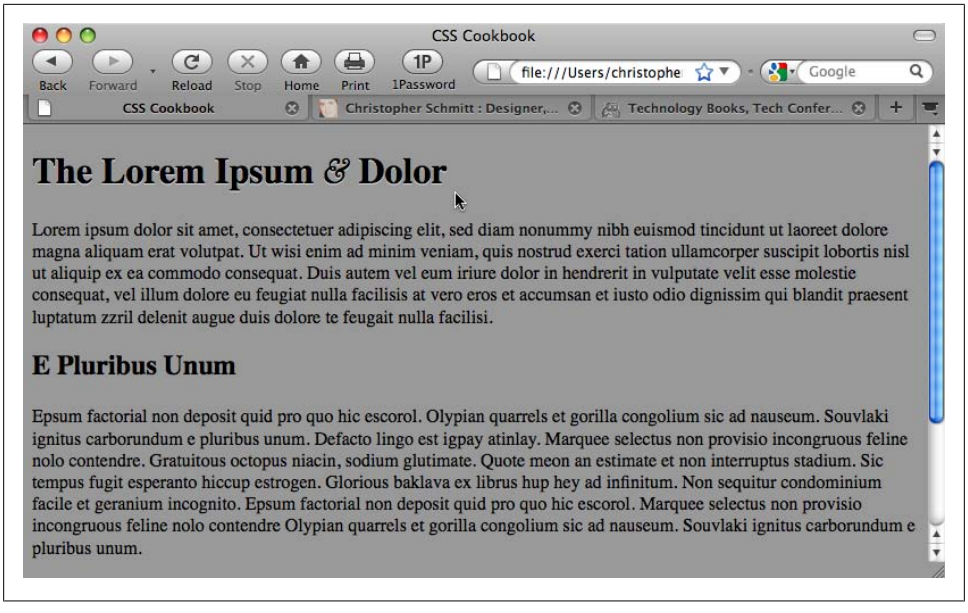



*Figure 3-45. Bevel look with a text-shadow*

### 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);
}
```

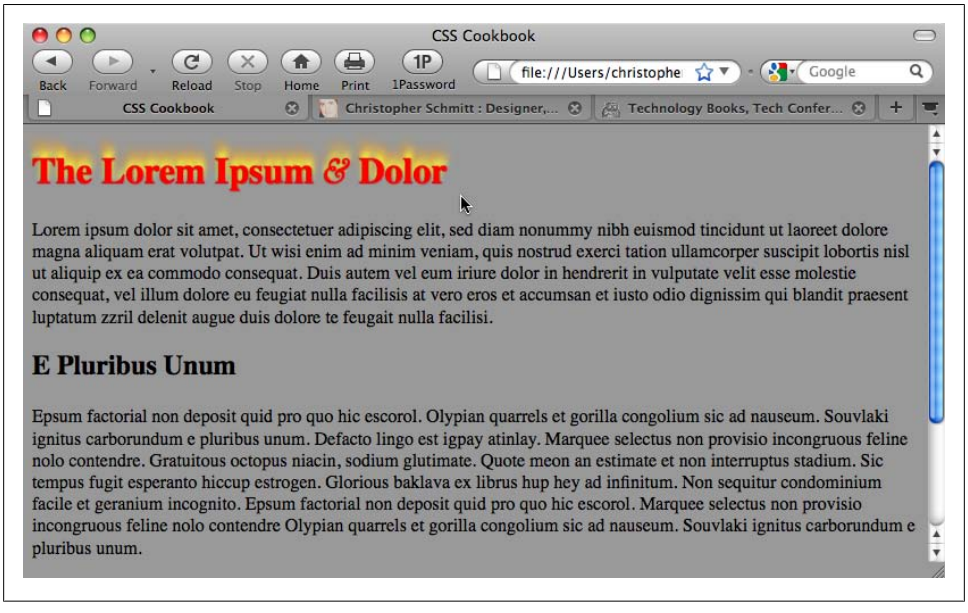



*Figure 3-46. Adding a red flame to text*

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;
}
```

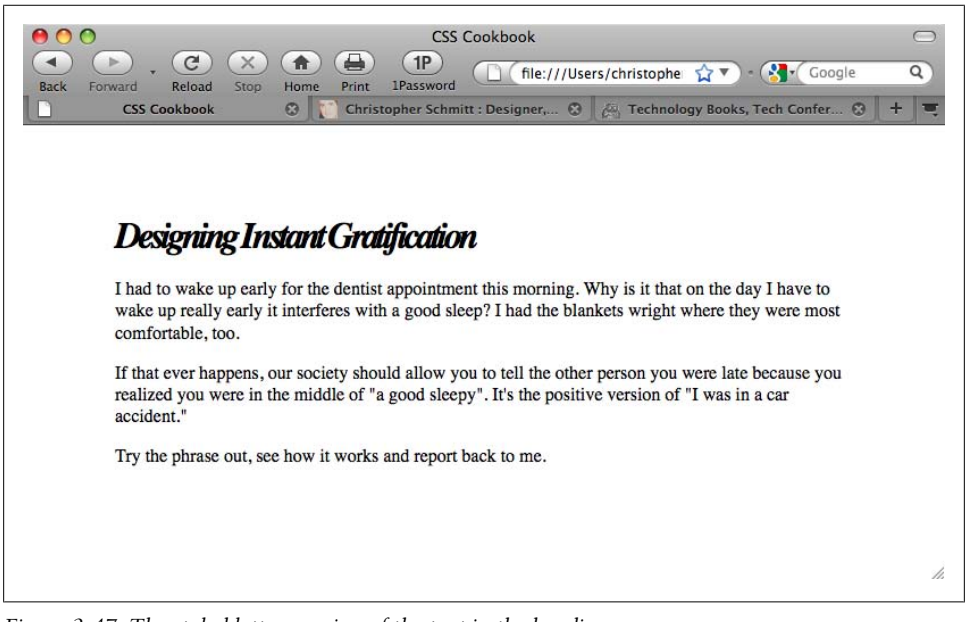



*Figure 3-47. The styled letter spacing of the text in the heading*

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.

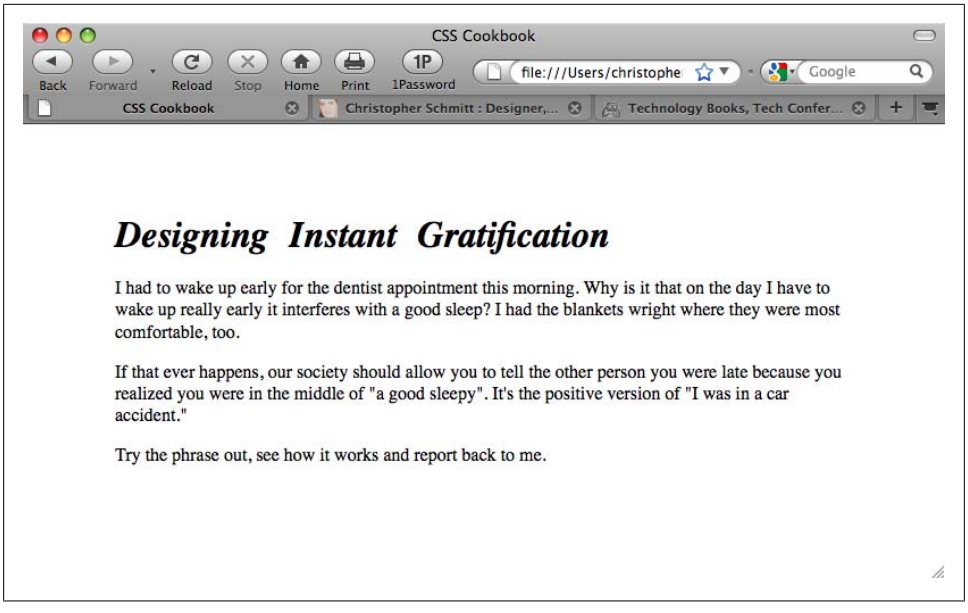



*Figure 3-48. Words in the heading spaced farther apart*

### 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.

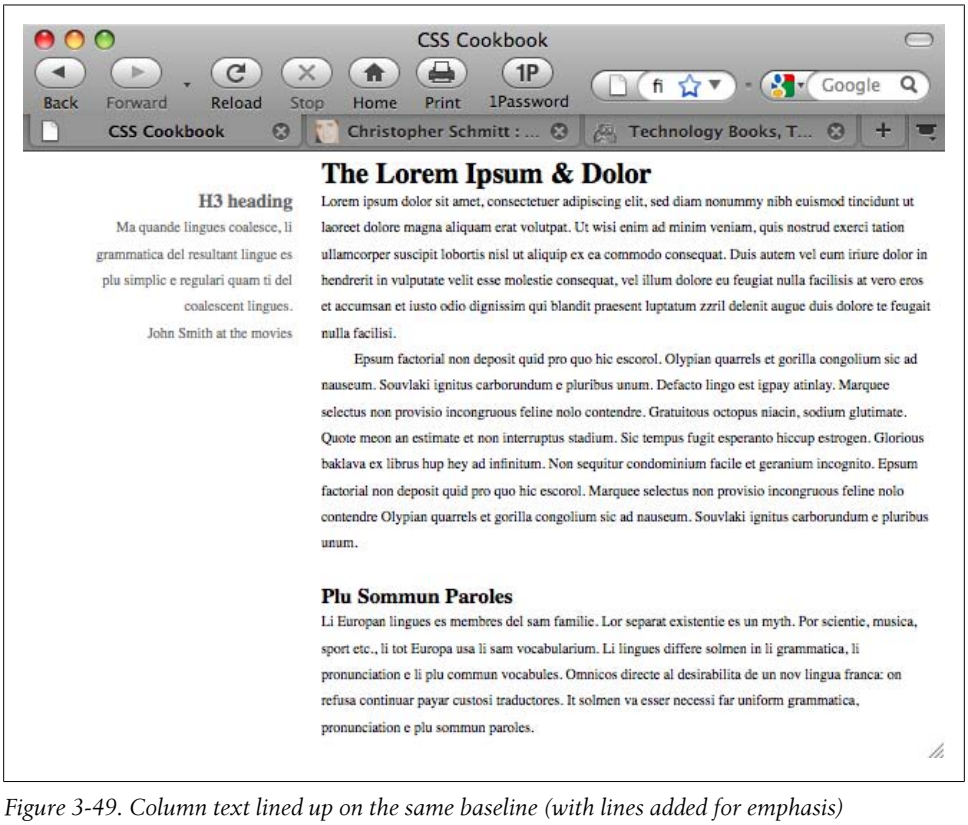



*Figure 3-49. Column text lined up on the same baseline (with lines added for emphasis)*

## 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;
}
```

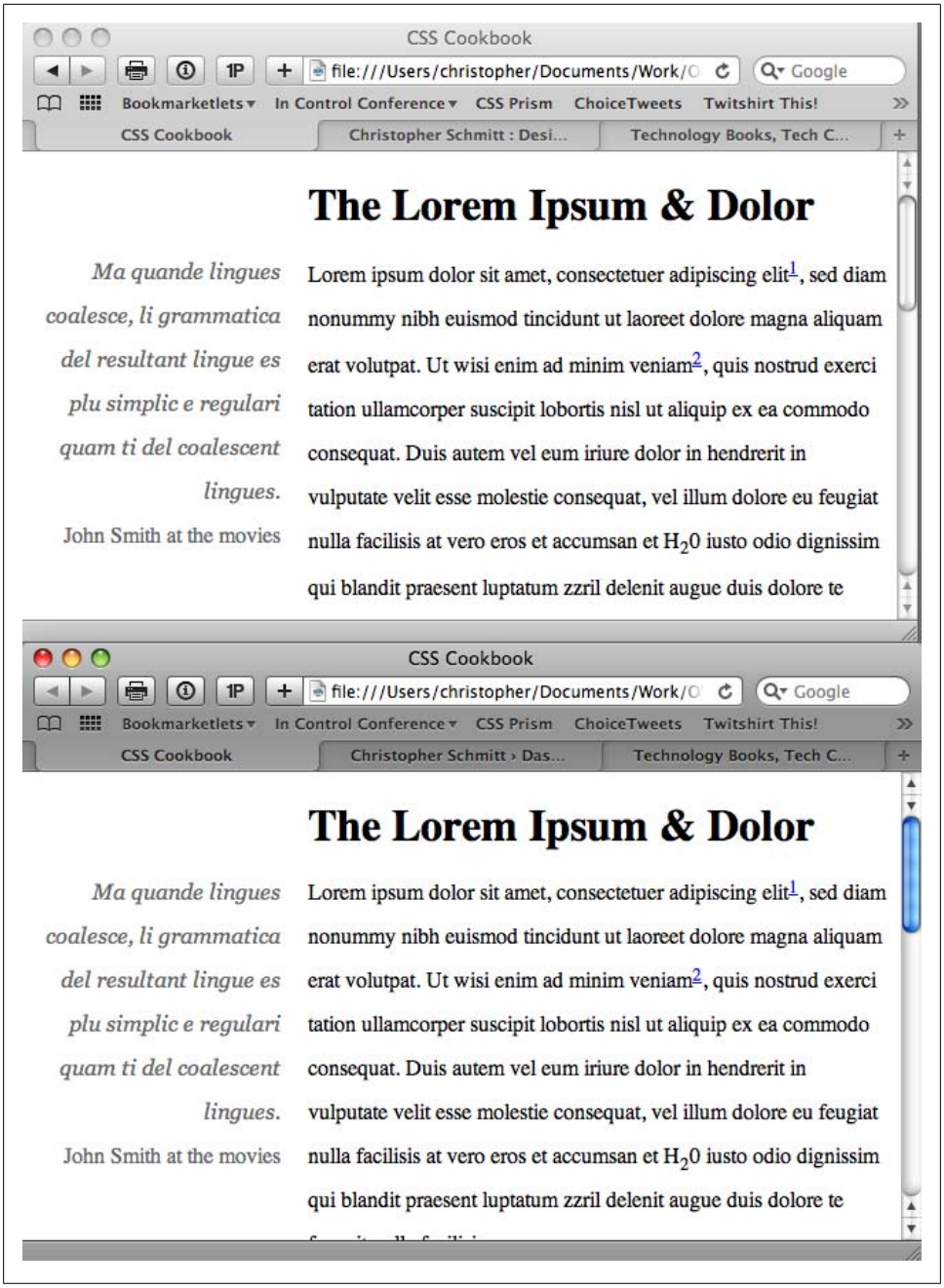



*Figure 3-50. The baseline shifting in the Safari browser with the addition of superscripts and subscripts*

## 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.

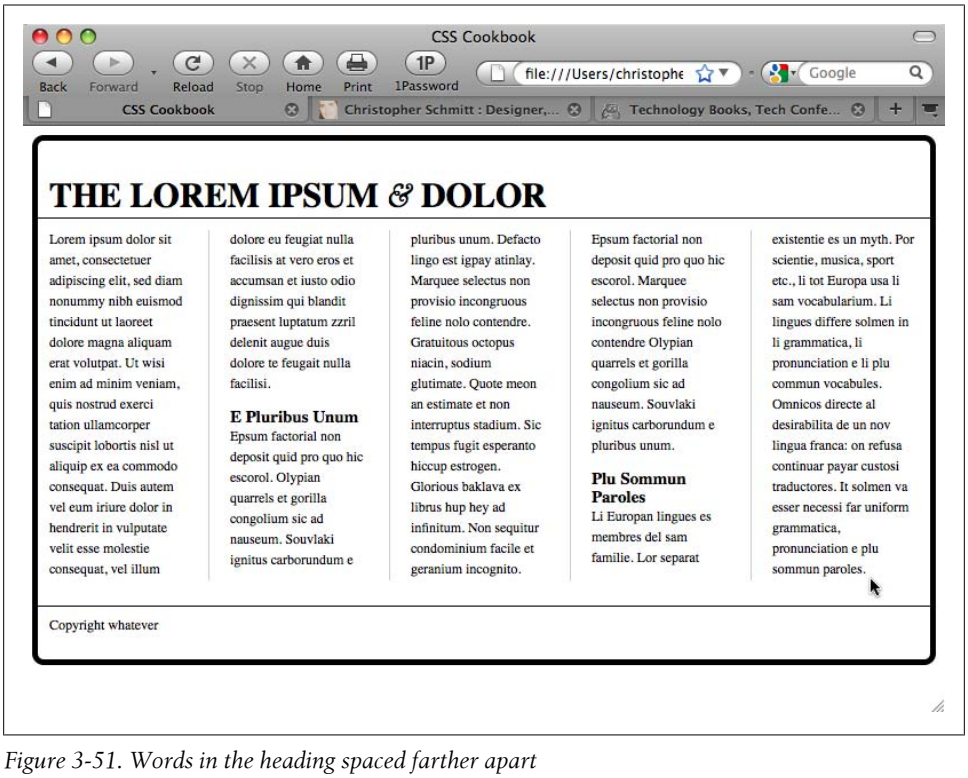



*Figure 3-51. Words in the heading spaced farther apart*

## 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*