Stunning CSS3

A PROJECT-BASED GUIDE
TO THE LATEST IN CSS

Zoe Mickley Gillenwater

Stunning CSS3: A project-based guide to the latest in CSS
Zoe Mickley Gillenwater

New Riders
1249 Eighth Street
Berkeley, CA 94710
(510) 524-2178
Fax: (510) 524-2221

Find us on the Web at www.newriders.com
To report errors, please send a note to errata@peachpit.com
New Riders is an imprint of Peachpit, a division of Pearson Education
Copyright © 2011 by Zoe Gillenwater

Acquisitions Editor: Wendy Sharp
Production Editor: Hilal Sala
Project/Copy Editor: Wendy Katz
Technical Editor: Chris Mills
Cover design: Charlene Charles-Will
Interior design: Mimi Heft, Charlene Charles-Will
Compositor: Danielle Foster
Indexer: Emily Glossbrenner

ISBN 13: 978-0-321-72213-3
ISBN 10: 0-321-72213-2

9 8 7 6 5 4 3 2 1

Printed and bound in the United States of America

This page intentionally left blank

Table of Contents

Introduction xiii

CHAPTER 1 **The CSS3 Lowdown** 1

What is CSS3? 2

Overview of What's New 2

Where CSS3 Stands 4

Use CSS3 Now 5

The State of Browser Support 6

Browser Market Share. 7

How the Major Players Stack Up 8

Progressive Enhancement 11

Advantages 11

Let Me Put it This Way… 13

Benefits of CSS3 15

Reduced Development and Maintenance Time 16

Increased Page Performance 16

Better Search Engine Placement 18

Increased Usability and Accessibility 19

Staying at the Front of the Pack. 19

Case Study: The Highway Safety Research Center 19

Before CSS3. 20

After CSS3. 22

Using CSS3 Wisely 25
Browser Prefixes 25
Dealing with Non-supporting Browsers 30
Filtering IE with Conditional Comments 36
Dealing with Unsupportive Clients or Bosses . . . 41
Don't Tell Them Everything 41
Educate Them About Progressive Enhancement Up Front 42
Manage Expectations from Design Mockups . 43

CHAPTER 2 **Speech Bubbles** 47
The Base Page 48
Corralling Long Text 49
Graphic Effects Sans Graphics 51
Rounding the Corners 51
Adding the Bubble's Tail 55
Semitransparent Backgrounds with RGBA or HSLA 62
Image-free Gradients 72
Image-free Drop Shadows 81
Image-free Text Shadows 85
Transforming the Avatars 89
What are Transforms? 89
Rotating the Avatars 92
The Finished Page 96

CHAPTER 3 **Notebook Paper** 99

The Base Page 100

Beyond the Basic Background 101

Scaling the Background Image 101

Multiple Background Images on One Element 109

Adding a Graphic Border. 114

Adding a Drop Shadow. 125

Embedding Unique Fonts 126

What is `@font-face`? 127

Choosing Acceptable Fonts 128

Browser Support. 134

Converting Fonts. 135

Using `@font-face` 137

The Finished Page 146

CHAPTER 4 **Styling Images and Links by Type** 149

The Base Page 150

What are Attribute Selectors?. 151

Indicating File Types with Dynamically Added Icons 153

Alternative Icon Ideas 156

Fixing IE 6 157

Styling Full-size Photos and Thumbnails Differently. 160

The Trouble with Classes 160

Using Attribute Selectors to Target by Type . . 164

The Finished Page 165

CHAPTER 5 **Improving Efficiency Using Pseudo-classes** 167

Targeting Specific Elements Without Using IDs or Classes 168

New Structural Pseudo-classes 169

Back to the Speech Bubbles: Alternating Colors 171

Back to the Photos: Random Rotation 176

Dynamically Highlighting Page Sections 180

The `:target` Pseudo-class 181

Adding the Table of Contents. 182

Changing Background Color on the Jumped-to Section 188

Animating the Change with Pure CSS 191

CHAPTER 6 **Different Screen Size, Different Design** 205

The Base Page 206

What are Media Queries? 208

Changing the Layout for Large Screens 209

From Horizontal Nav Bar to Vertical Menu . . . 213

Multi-column Text 214

Changing the Layout for Small Screens 220

Changing the Layout for Mobile Devices 226

What is Device Width? 227

The Third Media Query. 228

Improving the Look on High-resolution Displays 233

The Viewport `meta Tag` 235

Workarounds for Non-supporting Browsers. . . 240

The Finished Page .241

CHAPTER 7 **Flexing Your Layout Muscles** 243

Changes on the Horizon 244

Creating Multi-column Layouts Without Floats or Positioning 245

Making Blocks Flex. 248

Adding Columns. 253

Reordering Columns 255

Equal-height Columns 258

Vertical and Horizontal Centering 261

Reality Check: What Works Now 266

Flexible Form Layout 268

Sticky Footers. 272

Alternatives to the Flexible Box Model 277

The `box-sizing` Property 277

Future Layout Systems 284

APPENDIX A **Browser Support** 287

Conclusion 289

Index. 291

Credits. 301

This page intentionally left blank

Introduction

CSS3, the newest version of the style sheet language of the web, is less about creating new effects and more about accomplishing the beautiful web design effects you're familiar with in fantastic new ways—ways that are more efficient and produce more usable and flexible results than the techniques we've been using for the last decade.

CSS3 is still changing and evolving, as are browsers to support it and web designers to figure out how best to use it. CSS3 can create some stunningly beautiful and cool effects, as you'll see throughout this book. But if these effects aren't practical for real-world sites right now, what's the point? In this book, I'll focus on teaching you the cutting-edge CSS techniques that can truly improve your sites and are ready to be used in your work right away.

This book is not an encyclopedia or reference guide to CSS3; it won't teach you every single property, selector, and value that's new to CSS since version 2.1. Instead, it will teach you the most popular, useful, and well-supported pieces of CSS3 through a series of practical but innovative projects. Each chapter (after Chapter 1) walks you through one or more exercises involving the new techniques of CSS3 to produce a finished web page or section of a page. You can adapt these exercises to your own projects, or use them as inspiration for completely different ways to creatively use the new properties, selectors, and values you've learned.

In some ways, CSS3 is a new way of thinking as much as a new way of developing your pages. It can be hard to think of how to use the new `border-image` property, for instance, when you've been making web sites for years and aren't used to having the option of using an image for the border of a box. Because of this, I've included a list of ideas for how to use each CSS3 property, selector, and value I cover, beyond just the single way we use it in the exercise. I hope to provide you with plenty of inspiration for how to put the CSS3 techniques you're learning to work in your own projects, plus the technical know-how to make sure you can use CSS3 comfortably and efficiently.

Who Should Read this Book

This book is meant for anyone who already has experience using CSS, but wants to take their sites and skills to the next level. I assume that you know HTML and CSS syntax and terminology, but you don't need to be a CSS expert, and you certainly don't need to have any experience using anything that's new to CSS3. Whether you've just started using CSS or have been developing sites with it for years, this book will teach you powerful new techniques to add to your CSS toolkit.

Exercise Files

Each of the chapters is made up of at least one exercise where you will have the opportunity to implement the techniques in a real page, step by step. You can download the files for these exercises at the book's companion site at www.stunningcss3.com and work along in them as you go through the steps of each exercise. I've provided both a starter file and final file for each exercise, as well as a few intermediate steps for the longer exercises, so you can check in periodically and make sure you've made the correct changes to your own files.

You can use whatever code editor you like when working with the exercise files. There are no tools in particular that you must have in order to work with and create CSS. I personally use Adobe Dreamweaver, but do all of my CSS development in code view by hand. If you're using Dreamweaver or a similar editor, I recommend you too work on the CSS by hand.

Although a great deal of effort has been made to check the code in this book, there are bound to be a few errors. Please report any errors to me through the email form on the book's web site, and I'll be sure to note them on the site and update the exercise files if needed.

Links

Each chapter contains several links to related resources, articles, tutorials, tools, and examples that I think would be useful for you. And it's certainly easier to click on a live link than painstakingly type out a URL that you're copying from a printed book, so I've provided a compendium of all the links from each chapter on www.stunningcss3.com.

CSS3 is a rapidly changing topic, so in a few cases, I'll be updating these link lists as new resources come out. You'll see a note in the book every time one of these continually updated lists of resources is present, pointing you to the book site to find the latest information.

Browsers

The exercises in this book have been tested in the latest versions of the major browsers. At the time of this writing, these browser versions are Chrome 6, Firefox 3.6, Internet Explorer 8, Opera 10.6, and Safari 5. The exercises were also tested in the beta versions of Internet Explorer 9 and Firefox 4 available at the time of this writing, but behavior may be different from what's described in the book by the time these browsers are finalized and released.

The exercises have also been tested in older browser versions that are still in significant use today (such as Internet Explorer 7 and 6). In many cases, the CSS3 effects we'll be adding that work in the newest browsers also work in older versions of those same browsers; even when they don't, the pages still work, are always perfectly usable, and look fine. We'll always go over possible ways to provide workarounds or fallbacks for non-supporting browsers for each technique.

For information on which browsers a given technique works in, I've provided a table of browser-support information for each property or selector introduced in each chapter. Each browser is set to "yes," "partial," or "no." A value of "yes" means the browser supports all of the syntax and behavior; it may have very minor bugs or inconsistencies with the spec, but overall it's compliant. A value of "partial" means the browser supports some of the syntax and behavior, but not all, or not without significant bugs or inconsistencies.

Some CSS3 properties work only using a vendor-specific prefixed version of the property (you'll learn about these prefixed properties in Chapter 1). I've indicated which browsers require the prefixes on a given property in the browser support tables.

NOTE: On the flip side, I've also occasionally included the browser version number in the support table when it's particularly notable how early the property or selector was supported—for instance, the fact that IE 4 supports `@font-face`!

In cases where support in a given browser is relatively new and there's a chance that some users of the older, non-supporting versions of that browser are still out there, I've provided the version number of the browser in the browser support table, indicating which version was the earliest to support the property or selector. If the browser has supported the property or selector for the last few versions and it's unlikely that there's any significant number of users of the non-supporting versions, I have not included the earliest version number in the support table; you can feel safe that all versions of that browser in use support it.

Conventions Used Throughout this Book

This book uses a few terms that are worth noting at the outset.

- *W3C* refers to the World Wide Web Consortium, the organization that creates the official standards and specifications of the web, including CSS3.
- *IE* refers to the Windows Internet Explorer browser. IE 8 and earlier means IE 8, 7, and 6.
- *Webkit-based browsers* means Safari (both on desktop and on mobile devices), Chrome, and any other browsers that use a recent version of the Webkit browser-rendering engine.
- Occasionally, you'll see a reference to "all browsers." This means all browsers that are in significant use today, not literally every single obscure browser that may have a fractional piece of market share.

All of the exercises in this book are written in HTML5 markup. However, all that means in this case is that I've used the short and sweet HTML5 doctype, `<!DOCTYPE html>`, as well as the shorter `meta` character encoding, `style`, and `script` tags. I haven't included any of the new elements that HTML5 introduces, such as `section` or `article`, so the pages will work without any trouble in IE 8 and earlier, but you're welcome to change the markup for your own pages in whatever way you like. All the exercises will also work in HTML 4.01 or XHTML 1.

All CSS examples shown should be placed in an external style sheet or in the `head` of an HTML or XHTML document. The exercise files have their CSS contained in the `head` of the page, for ease of editing, but it's best to move that CSS to an external style sheet for actual production files.

Some code examples will contain characters or lines colored teal-blue. This indicates that content has been added or changed since the last time you saw that same code snippet, or, in a new code snippet, that there is a particular part that you need to focus on. In some cases. you'll see a ¬ character at the beginning of a line of code, indicating that the text has had to wrap to a new line within the confines of the layout of this book—but this doesn't mean you have to break the line there.

Each property or selector introduced in this book has a “lowdown” sidebar providing a brief overview of its syntax, behavior, and use cases. Not every detail of syntax could be included, of course, but the most essential information you need is there for quick reference. I’ve also provided a link to whichever CSS3 module the property or selector is a part of on the W3C site so you can refer to the full specification when needed.

This page intentionally left blank

3

Notebook Paper

Chapter 2 was all about creating graphic effects without any graphics. In this chapter, we'll use plenty of images, but new CSS3 properties allow us to use them with more streamlined markup and to make them behave in ways not possible with CSS 2.1. You'll also learn how to use unique, non-web-safe fonts in your pages without resorting to Flash, images, or scripting—even in Internet Explorer. Altogether, we'll be able to use these image and font techniques to make a web page look like a realistic piece of notebook paper.

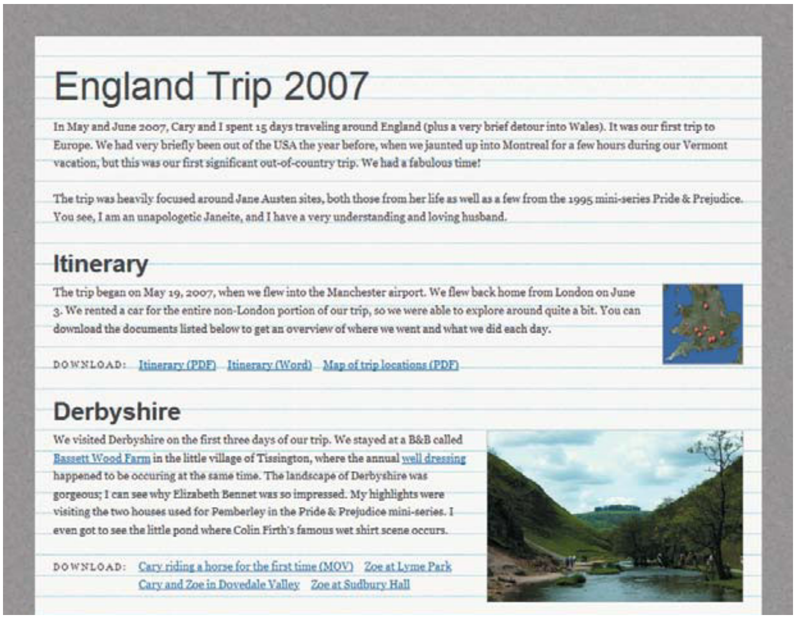
WHAT YOU'LL LEARN

We'll create the appearance of a piece of notebook paper using these CSS3 properties and concepts:

- The `background-size` property to scale a background image with the text
- Multiple background images on one element
- The `border-image` property to create graphic borders
- The `background-clip` property to move a background image out from under a border
- The `@font-face` rule to embed unique fonts in the page

The Base Page

Creating the appearance of real objects, like sticky notes and file folders, has always been popular in web design. If you wanted an article to look like it was written on a piece of real paper, the first step might be to apply a simple lined paper background image to it. **Figure 3.1** shows this starting point.



England Trip 2007

In May and June 2007, Cary and I spent 15 days traveling around England (plus a very brief detour into Wales). It was our first trip to Europe. We had very briefly been out of the USA the year before, when we jaunted up into Montreal for a few hours during our Vermont vacation, but this was our first significant out-of-country trip. We had a fabulous time!

The trip was heavily focused around Jane Austen sites, both those from her life as well as a few from the 1995 mini-series Pride & Prejudice. You see, I am an unapologetic Janeite, and I have a very understanding and loving husband.

Itinerary

The trip began on May 19, 2007, when we flew into the Manchester airport. We flew back home from London on June 3. We rented a car for the entire non-London portion of our trip, so we were able to explore around quite a bit. You can download the documents listed below to get an overview of where we went and what we did each day.

DOWNLOAD: Itinerary (PDF) Itinerary (Word) Map of trip locations (PDF)

Derbyshire

We visited Derbyshire on the first three days of our trip. We stayed at a B&B called Bassett Wood Farm in the little village of Tissington, where the annual well dressing happened to be occuring at the same time. The landscape of Derbyshire was gorgeous; I can see why Elizabeth Bennet was so impressed. My highlights were visiting the two houses used for Pemberley in the Pride & Prejudice mini-series. I even got to see the little pond where Colin Firth's famous wet shirt scene occurs.

DOWNLOAD: Cary riding a horse for the first time (MOV) Zoe at Lyme Park
Cary and Zoe in Dovedale Valley Zoe at Sudbury Hall

FIGURE 3.1 The article with a single background image, before any CSS3 is applied.

Beyond the Basic Background

To make the web page shown in Figure 3.1 look more like a realistic piece of paper, you would want to add some extra graphic details beyond the lined background, like a torn edge or a coffee stain. Without CSS3, it's certainly possible to add these graphic details. But new properties in CSS3 make it easier and keep your markup cleaner. Let's add some of these new properties now to enhance the background.

Scaling the Background Image

One thing that would make the background look more realistic is if the text were aligned to the notebook paper lines, instead of overlapping them indiscriminately. To fix this without CSS3, you would need to set a base `font-size` and `line-height` in pixels, and then adjust the spacing between the lines in your background image to match. This would work for most users. But if anyone resized the text, or had non-standard user settings to override the pixel font sizes, the text would become misaligned. The text could scale, but the background image couldn't.

But that was then—before the CSS3 `background-size` property was introduced. With `background-size`, you can control the horizontal and vertical scaling of a background image as well as how it stretches to cover the background area and gets clipped.

HOW `background-size` WORKS

Before we apply `background-size` to our page, let's look at a couple of simple examples to get a better grip on how the property works.

Figure 3.2 shows an image 200 pixels wide by 120 pixels tall. **Figure 3.3** shows how the image looks when set as a normal repeating background of a `div` that's 500 pixels wide by 200 pixels tall; since the `div`'s dimensions aren't an even multiple of the image's dimensions, some of the image gets cut off on the right and bottom.

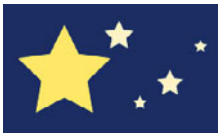


FIGURE 3.2 An image that's 200 pixels wide by 120 pixels tall

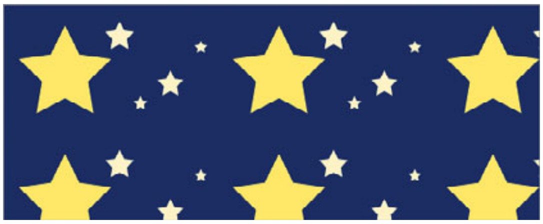


FIGURE 3.3 When the image is repeated across the background of the `div`, some of it gets cut off on the right and bottom.

We can use the `background-size` property to scale the image down from 200 pixels to 100 pixels wide:

```
div {
   width: 500px;
   height: 200px;
   border: 1px solid #999;
   background-image: url(images/stars.gif);
   background-size: 100px auto;
}
```

The first value in the `background-size` property, `100px`, sets the width of the background image. The second value, `auto`, sets the height. A value of `auto` makes the height whatever it needs to be to preserve the aspect ratio of the image. If you leave the second value off, the browser assumes it to be `auto`, so a value of `background-size: 100px;` would have worked identically here. Compare **Figure 3.4** to Figure 3.3 to see how the background image has been shrunk but kept its aspect ratio.

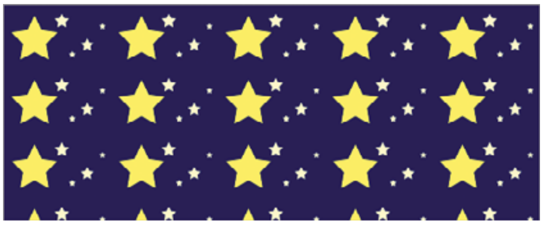


FIGURE 3.4 The browser has scaled the image to 100 pixels wide, so it now fits in the `div` exactly five times and doesn't get cut off on the right.

If you use percentages in the `background-size` property, they're relative to the box the background is on, not to the background image itself. If you wanted exactly two copies of the image to show in the `div`, with neither cut off at all, you could use this CSS:

```
div {
   width: 500px;
   height: 200px;
   border: 1px solid #999;
   background-image: url(images/stars.gif);
   background-size: 50% 100%;
}
```

The image is stretched to fill half the width of the div and all of its height, and then repeated (**Figure 3.5**). In this case, the browser has to both distort the shape of the image and scale it up, making the edges in the image look a little blurry or pixelated. As with any browser-based scaling, background sizing is not going to look good with all images, but can work quite well with grungy, abstract, or very simple images that don't have super-clean edges—such as our lined-paper background.

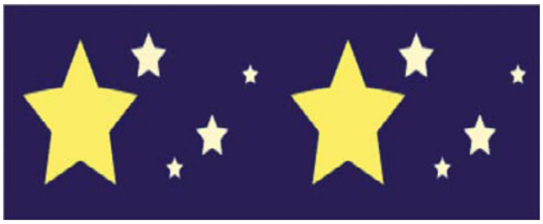


FIGURE 3.5
The browser has scaled the image to fit twice across the width and once across the height, distorting it but keeping it from cutting off.

MORE NEW WAYS TO TILE BACKGROUNDS

Besides setting background-size to a value that fits perfectly within the width of a box, another way to keep background image tiles from getting cut off on one or more sides is to use the values of round and space in the background-repeat property. These values are new to CSS3, and can be used in conjunction with background-size or without it.

A value of round repeats the background image but rescales it so it will fit an even number of times without getting cut off. A value of space repeats the background image as often as it will fit without getting cut off, and then spaces the tiles out to fill any leftover room.

Unfortunately, at the time of this writing, the only browsers that support these values are IE 9 and Opera, but Opera does so incompletely and incorrectly. Until these background-repeat values have better support, background-size is your best bet for ensuring background images don't get cut off, though it's not as flexible as round and space are.

MAKING THE PAPER LINES SCALE WITH THE TEXT

In order to make our paper background image scale with the text, we need to set its dimensions not in percentages or pixels, but in ems. Ems are a relative unit of measurement based on the current font height.

To get started, download the exercise files for this chapter at www.stunningcss3.com, and open paper_start.html in your code editor of choice. Its CSS is contained in a `style` element in the `head` of the page.

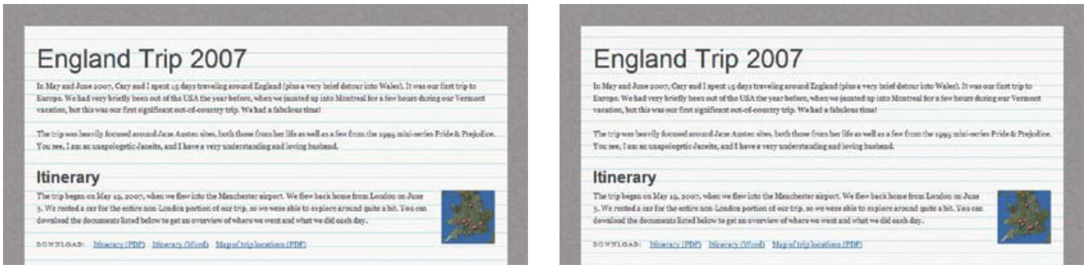
Find the `#paper` rule in the CSS, and add the `background-size` property, plus the Mozilla and Webkit equivalents:

```
#paper {
   float: left;
   margin: 40px;
   padding: 3.2em 1.6em 1.6em 1.6em;
   background: url(images/paperlines.gif) #FBFBF9;
   -moz-background-size: auto 1.6em;
   -webkit-background-size: auto 1.6em;
   background-size: auto 1.6em;
}
```

Opera, Chrome, Safari 5, Firefox 4, and IE 9 use the standard `background-size` property; Firefox 3.6 and Safari 4 and earlier use the `-moz-` and `-webkit-` versions of the property, respectively. In each property, we're telling the browser that we want the height of the image to be 1.6 ems and that we want the width to just size itself proportionally. The image depicts one line on the paper, so that means that the space between every line will now be 1.6 ems tall. Why 1.6 ems? The height of each line of text is 1.6, specified by the `line-height` already in place on the `body` element:

```
body {
   margin: 0;
   padding: 40px;
   background: #CCC url(images/background.gif);
   color: #333;
   font-size: 87.5%;
   font-family: Georgia, "Times New Roman", Times, serif;
   line-height: 1.6;
}
```

Figure 3.6 shows that the background image's size has indeed changed, but the text is still not lining up with the lines in the image. This is because we haven't set all the text sizes and margins to line up with a regular spacing of 1.6 ems. The paragraph and list text have the correct spacing for the background image, since their `line-height` is already 1.6 and their bottom margins are 1.6 ems, as you'll see in the CSS. But the headings need to have their margins tweaked to fall in line.



England Trip 2007

In May and June 2007, Cary and I spent 15 days traveling around England (plus a very brief detour into Wales). It was our first trip to Europe. We had very briefly been out of the USA the year before, when we jaunted up into Montreal for a few hours during our Vermont vacation, but this was our first significant out-of-country trip. We had a fabulous time!

The trip was heavily focused around Jane Austen sites, both those from her life as well as a few from the 1995 mini-series Pride & Prejudice. You see, I am an unapologetic Janeite, and I have a very understanding and loving husband.

Itinerary

The trip began on May 19, 2007, when we flew into the Manchester airport. We flew back home from London on June 5. We rented a car for the entire non-London portion of our trip, so we were able to explore around quite a bit. You can download the documents listed below to get an overview of where we went and what we did each day.

DOWNLOAD: Itinerary (PDF) Itinerary (Word) Map of trip locations (PDF)

England Trip 2007

In May and June 2007, Cary and I spent 15 days traveling around England (plus a very brief detour into Wales). It was our first trip to Europe. We had very briefly been out of the USA the year before, when we jaunted up into Montreal for a few hours during our Vermont vacation, but this was our first significant out-of-country trip. We had a fabulous time!

The trip was heavily focused around Jane Austen sites, both those from her life as well as a few from the 1995 mini-series Pride & Prejudice. You see, I am an unapologetic Janeite, and I have a very understanding and loving husband.

Itinerary

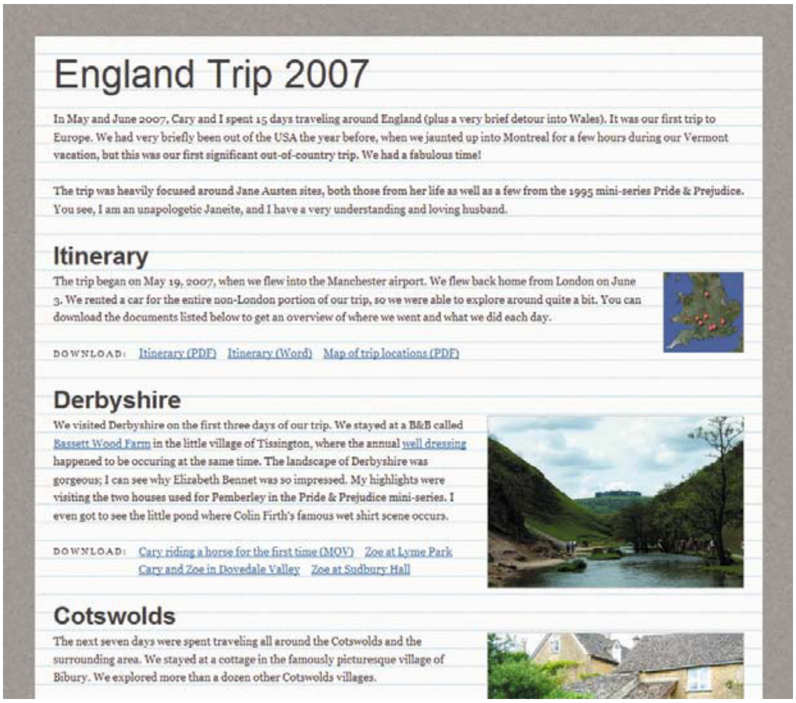
The trip began on May 19, 2007, when we flew into the Manchester airport. We flew back home from London on June 5. We rented a car for the entire non-London portion of our trip, so we were able to explore around quite a bit. You can download the documents listed below to get an overview of where we went and what we did each day.

DOWNLOAD: Itinerary (PDF) Itinerary (Word) Map of trip locations (PDF)

FIGURE 3.6 The background image lines are closer together after applying `background-size`**.**

```
h1 {
   margin: -.3em 0 .14em 0;
   color: #414141;
   font-family: Helvetica, "Helvetica Neue", Arial,
   ¬sans-serif;
   font-size: 3.5em;
   font-weight: normal;
}
h2 {
   clear: left;
   color: #414141;
   margin: 0 0 -.14em 0;
   font-family: Helvetica, "Helvetica Neue", Arial,
   ¬sans-serif;
   font-size: 2.17em;
   font-weight: bold;
}
```

These margin values are based on trial and error. Unlike with absolute pixel-based measurements, you're not going to be able to find values that work perfectly for all browsers; each browser has different ways of rounding and translating relative measurements like ems into the pixels displayed on the screen. In this case, these margin values work well for Firefox, Safari, and Chrome. Everything is spaced out at regular intervals of 1.6 ems, keeping the text aligned to the lines in the background image (**Figure 3.7**).



England Trip 2007

In May and June 2007, Cary and I spent 15 days traveling around England (plus a very brief detour into Wales). It was our first trip to Europe. We had very briefly been out of the USA the year before, when we jaunted up into Montreal for a few hours during our Vermont vacation, but this was our first significant out-of-country trip. We had a fabulous time!

The trip was heavily focused around Jane Austen sites, both those from her life as well as a few from the 1995 mini-series Pride & Prejudice. You see, I am an unapologetic Janeite, and I have a very understanding and loving husband.

Itinerary

The trip began on May 19, 2007, when we flew into the Manchester airport. We flew back home from London on June 3. We rented a car for the entire non-London portion of our trip, so we were able to explore around quite a bit. You can download the documents listed below to get an overview of where we went and what we did each day.

DOWNLOAD: Itinerary (PDF) Itinerary (Word) Map of trip locations (PDF)

Derbyshire

We visited Derbyshire on the first three days of our trip. We stayed at a B&B called Bassett Wood Farm in the little village of Tissington, where the annual well dressing happened to be occuring at the same time. The landscape of Derbyshire was gorgeous; I can see why Elizabeth Bennet was so impressed. My highlights were visiting the two houses used for Pemberley in the Pride & Prejudice mini-series. I even got to see the little pond where Colin Firth's famous wet shirt scene occurs.

DOWNLOAD: Cary riding a horse for the first time (MOV) Zoe at Lyme Park Cary and Zoe in Dovedale Valley Zoe at Sudbury Hall

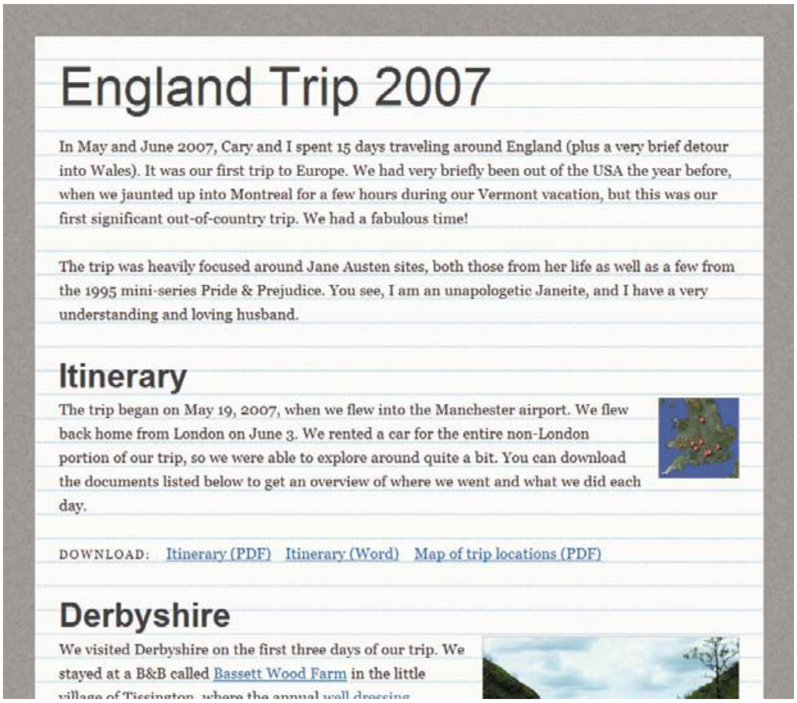
Cotswolds

The next seven days were spent traveling all around the Cotswolds and the surrounding area. We stayed at a cottage in the famously picturesque village of Bibury. We explored more than a dozen other Cotswolds villages.

FIGURE 3.7 The text is now aligned to the lines in the background image, shown here in Firefox 3.6.

But in Opera, the text isn't aligned, as Opera sizes the background image just slightly smaller than the other browsers. If we were to adjust the font sizes and margins to make everything line up in Opera, it would mess up the alignment in the other browsers. You'll have to decide which browsers are more important to you, based on your own site's visitor statistics, and cater your measurements to those.

Once the text is aligned with the background image, if the user has a different default text size from the norm, or scales the text size up or down, the background image scales with it, keeping the lines always aligned with the text (**Figure 3.8**). Also, if you were to later change the base font size on the `body` element, everything would scale to match, without your having to remake the background image.



England Trip 2007

In May and June 2007, Cary and I spent 15 days traveling around England (plus a very brief detour into Wales). It was our first trip to Europe. We had very briefly been out of the USA the year before, when we jaunted up into Montreal for a few hours during our Vermont vacation, but this was our first significant out-of-country trip. We had a fabulous time!

The trip was heavily focused around Jane Austen sites, both those from her life as well as a few from the 1995 mini-series Pride & Prejudice. You see, I am an unapologetic Janeite, and I have a very understanding and loving husband.

Itinerary

The trip began on May 19, 2007, when we flew into the Manchester airport. We flew back home from London on June 3. We rented a car for the entire non-London portion of our trip, so we were able to explore around quite a bit. You can download the documents listed below to get an overview of where we went and what we did each day.

DOWNLOAD: Itinerary (PDF) Itinerary (Word) Map of trip locations (PDF)

Derbyshire

We visited Derbyshire on the first three days of our trip. We stayed at a B&B called Bassett Wood Farm in the little

FIGURE 3.8 Even if the user has a larger text size, the text stays aligned with the background image lines.

WORKAROUNDS FOR IE

The `background-size` property doesn't work in IE 8 and earlier, and there are no workarounds to directly emulate it. In this case, it's a minor visual effect, so I think we can chalk it up as progressive enhancement and not worry about its lack in IE.

You can, however, provide alternate styles using Modernizr, which does detect for support of the `background-size` property. For instance, you could provide a different background image altogether, or you could provide an alternate version of the lined paper background image that has been designed to fit with a particular pixel font size; you would set this pixel font size only for browsers that don't support `background-size`. I don't recommend doing this here, as pixel-based font sizes are bad for accessibility. However, Modernizr is a good option in general for providing alternate styles when you're trying to scale a background image using `background-size`.

THE LOWDOWN ON THE `background-size` PROPERTY

The `background-size` property is part of the Backgrounds and Borders module, found at www.w3.org/TR/css3-background. Its value can be a width and height in any unit, or it can be `auto`.

Alternately, `background-size` can be set to either `contain` or `cover`. Both make the browser scale the image proportionally. A value of `contain` scales it to the largest size where both its width and height will fit inside the background area, so it doesn't get cut off at all, but often leaves some area with no background on it. A value of `cover` scales it to the smallest size where one tile of it will completely cover the background area, but allows it to get cut off where necessary to make sure the whole area has a background image covering it.

Other than scaling lines to match text spacing, you might want to use `background-size` for:

- Making the non-repeating background of the header of a page scale in a liquid or elastic layout to always fill the whole header width
- Making a repeating background image tile a full number of times instead of the tiles getting cut off on the edges of the box
- Making a large background image always fill the entire page; see www.alistapart.com/articles/supersize-that-background-please
- Scaling a faux-columns background image in a liquid layout; see www.css3.info/liquid-faux-columns-with-background-size
- Scaling a link or list item's background image icon with its text
- Scaling background images for the iPhone 4's high-resolution display down by half, so that when it doubles the pixels, as it always does, the images won't look blurry; see http://dryan.com/articles/posts/2010/6/25/hi-res-mobile-css-iphone-4
- Changing the size of background images based on the size of the user's window, using media queries, which you'll learn about in Chapter 6

TABLE 3.1 `background-size` **browser support**

IE	FIREFOX	OPERA	SAFARI	CHROME
Yes, 9+	Yes, 4+; 3.6 with -moz-	Yes	Yes, 5+; 3+ with -webkit-	Yes

Multiple Background Images on One Element

One of the changes to CSS that has brought web designers the most joy is the ability to apply multiple background images to a single element. In our example, we'll be able to use this function to make the paper look a little more realistic—we'll beat it up a bit by adding some stain images, as well as adding a thumbtack at the top.

Before CSS3, only one background image per box was allowed, so you'd have to add an extra `div` for each extra image and apply one image to each `div`. If you could count on other particular blocks already being inside your `div`s, such as a `h3` element always being the first nested element, you could apply background images to these other blocks instead of adding extra `div`s. However, doing so could be risky, as you would be relying on certain types of content always being present and in particular places; if those pieces of content weren't there, of course their background images wouldn't show up.

This nesting `div`s method wasn't difficult, but it was messy. It junked up your markup and increased the pages' file size. To add more images later, you'd need to not only change the CSS, but the HTML as well.

With CSS3, you can leave the HTML alone and instead simply list each background image in the `background-image` or `background` property, separated by commas. Each image can be positioned, repeated, sized, and otherwise controlled independently.

Figure 3.9 shows the extra images we want to apply to our article `div`. To apply them, add a new `background` declaration under the existing one in the `#paper` rule:

```
#paper {
   float: left;
   margin: 40px;
   padding: 3.2em 1.6em 1.6em 1.6em;
   background: url(images/paperlines.gif) #FBFBF9;
               background: url(images/thumbtack.png),
               url(images/stains1.png),
               url(images/stains2.png),
               url(images/stains3.png),
               url(images/stains4.png),
               url(images/paperlines.gif) #FBFBF9;
   -moz-background-size: auto 1.6em;
   -webkit-background-size: auto 1.6em;
   background-size: auto 1.6em;
}
```

NOTE: The line breaks and indentions in the background property are just there to make the CSS easier to read. You can write everything on one line, or not—it works the same regardless.

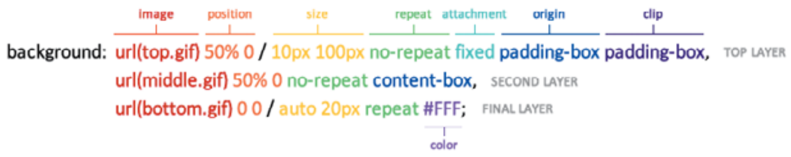
FIGURE 3.9 The five extra background images to add graphic detail to the notebook paper

NOTE: The water stain images shown in Figure 3.9 were created with the Photoshop brushes by Obsidian Dawn from www.obsidiandawn.com/water-stains-photoshop-gimp-brushes.

The first `background` declaration will continue to be used by IE and other browsers that don't support multiple background images. Because they don't understand the syntax of the second `background` declaration, they'll ignore it. Browsers that do support multiple background images will override the first declaration with the second.

The background images are layered on top of each other, with the first declared image put on top of the stack. That's why the thumbtack image is listed first and the lines image is listed last.

We're not quite done yet, though. We haven't told the browser how we want to repeat, position, and size each image. To do this, treat each snippet between the commas as if it were its own standalone `background` shorthand property, and write each of the background-related properties in it accordingly. **Figure 3.10** shows all the pieces that can go in the `background` shorthand property. The order matters for some and not for others, so I recommend sticking with the order shown in 3.10 just so you don't get confused or accidentally make a mistake. (I know *I* would otherwise!)



```
background: url(top.gif) 50% 0 / 10px 100px no-repeat fixed padding-box padding-box,
            url(middle.gif) 50% 0 no-repeat content-box,
            url(bottom.gif) 0 0 / auto 20px repeat #FFF;
```

FIGURE 3.10 **The** background **short-hand property can contain multiple layers; the top layer of this diagram includes all the possible pieces of the property (minus color, which can go only into the final layer).**

Using the order shown in the diagram in Figure 3.10, add the positioning and repeat values after each image in the `background` property:

```
background: url(images/thumbtack.png) 50% 5px no-repeat,
            url(images/stains1.png) 90% -20px no-repeat,
            url(images/stains2.png) 30% 8% no-repeat,
            url(images/stains3.png) 20% 50% no-repeat,
            url(images/stains4.png) 40% 60% no-repeat,
            url(images/paperlines.gif) #FBFBF9;
```

Next, modify the `background-size` properties to tell the browser that each image should be sized using its native dimensions, except for the last (the lines image):

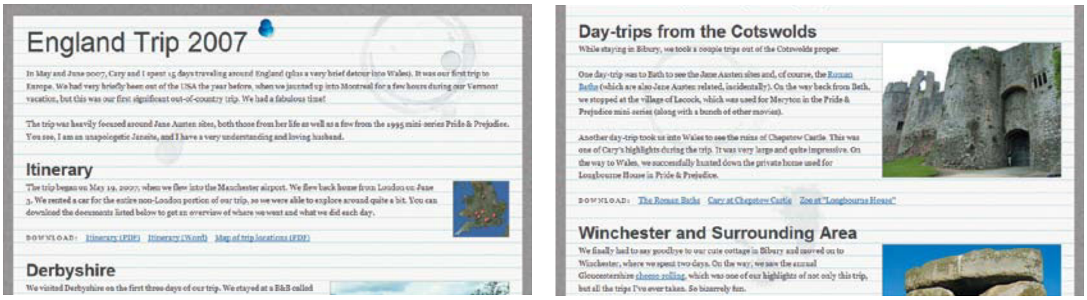
```
-moz-background-size: auto, auto, auto, auto, auto,
                      auto 1.6em;
webkit-background-size: auto, auto, auto, auto, auto,
                        auto 1.6em;
background-size: auto, auto, auto, auto, auto, auto 1.6em;
```

Each comma-separated value matches up with the comma-separated value at the same spot in the `background` property's value list.

Although you can technically include `background-size` information in the `background` shorthand property, it won't work right now. Older versions of Firefox and Safari need `background-size` declared using the vendor-prefixed properties, and although Opera, Chrome, Safari 5, Firefox 4, and IE 9 might accept `background-size` in the `background` property, adding it would break those older versions of Firefox and Safari. So, to keep it working everywhere, and to keep yourself from confusing the values for `background-position` and `background-size` (very easy to do!), keep `background-size` written separately from `background`.

NOTE: The page with all the changes to this point is named paper_1.html in the exercise files that you downloaded for this chapter.

Save your page and view it in an up-to-date browser. You should still see the text aligned with the notebook paper lines, but also see four stains scattered across the paper and a thumbtack at the top (**Figure 3.11**).



England Trip 2007

In May and June 2007, Cary and I spent 15 days traveling around England (plus a very brief detour into Wales). It was our first trip to Europe. We had very briefly been out of the USA the year before, when we jaunted up into Montreal for a few hours during our Vermont vacation, but this was our first significant out-of-country trip. We had a fabulous time!

The trip was heavily focused around Jane Austen sites, both those from her life as well as a few from the 1995 mini-series Pride & Prejudice. You see, I am an unapologetic Janeite, and I have a very understanding and loving husband.

Itinerary

The trip began on May 19, 2007, when we flew into the Manchester airport. We flew back home from London on June 3. We rented a car for the entire non-London portion of our trip, so we were able to explore around quite a bit. You can download the documents listed below to get an overview of where we went and what we did each day.

DOWNLOAD: Itinerary (PDF) Itinerary (Word) Map of trip locations (PDF)

Derbyshire

We visited Derbyshire on the first three days of our trip. We stayed at a B&B called

Day-trips from the Cotswolds

While staying in Bibury, we took a couple trips out of the Cotswolds proper.

One day-trip was to Bath to see the Jane Austen sites and, of course, the Roman Baths (which are also Jane Austen related, incidentally). On the way back from Bath, we stopped at the village of Lacock, which was used for Meryton in the Pride & Prejudice mini-series (along with a bunch of other movies).

Another day-trip took us into Wales to see the ruins of Chepstow Castle. This was one of Cary's highlights during the trip. It was very large and quite impressive. On the way to Wales, we successfully hunted down the private home used for Longbourne House in Pride & Prejudice.

DOWNLOAD: The Roman Baths Cary at Chepstow Castle Zoe at "Longbourne House"

Winchester and Surrounding Area

We finally had to say goodbye to our cute cottage in Bibury and moved on to Winchester, where we spent two days. On the way, we saw the annual Gloucestershire cheese rolling, which was one of our highlights of not only this trip, but all the trips I've ever taken. So bizarrely fun.

FIGURE 3.11 **All six background images show at various points across the `div`.**

The nice thing about setting each of these images independently, instead of combining them into one big image that you set on a single nested `div`, is that the images can move around based on the size of the `div`. No matter what size or dimensions the `div` has, there will be stain images distributed nicely across it, instead of clustered in one spot.

THE LOWDOWN ON MULTIPLE BACKGROUND IMAGES

Multiple background images are a new feature of the `background` and `background-image` properties, not a new property itself. These properties are part of the Backgrounds and Borders module, found at www.w3.org/TR/css3-background.

List each background image in the `background-image` or `background` property, separated by commas. The background images are layered on top of each other, with the first declared image put on top of the stack.

Each image can be positioned, repeated, sized, and otherwise controlled independently. To do so, include this background styling information with each image URL in the `background` property, or add a comma-separated list of values to each independent background property, such as `background-repeat: no-repeat, no-repeat, repeat-x, repeat`. Each value in the list matches up with a value in the list of background images.

Other than layering stain images over a paper background pattern, you might want to use multiple background images for:

- Flexible boxes with fancy or irregular corners or edges that other CSS3 properties like `border-radius` can't handle, such as ornate buttons that would still need images; see http://css-tricks.com/css3-multiple-backgrounds-obsoletes-sliding-doors
- Opening and closing quotation mark images on a blockquote; see http://css.dzone.com/news/multiple-backgrounds-oh-what-beautiful-thing
- The parallax effect, where resizing a window or hovering over a div makes the images appear to move at different speeds in relation to each other; see www.paulrhayes.com/2009-04/auto-scrolling-parallax-effect-without-javascript
- Making what appears to be a single image stretch across the whole width of a box or page, while it's really made up of multiple pieces, such as a landscape image with a sun that you always want to appear in the top right corner and a tree that you always want to appear in the bottom left corner
- Distributing images across the full width or height of a box, using percentage positions to keep them spaced out from each other, such as multiple cloud images over a blue background color
- Creating the appearance of an object from real life, using a top image slice, repeating middle slice, and bottom slice, all on the same box
- Applying a CSS3-generated gradient (remember, it goes in the `background-image` property, not `background-color`) along with a background image, to fade out a texture, blend the edges of an image into a solid color, or reveal portions of an image as the user scrolls down the page; see http://atomicrobotdesign.com/blog/htmlcss/make-the-thinkgeek-background-effect-using-css3

TABLE 3.2 Multiple background images browser support

IE	FIREFOX	OPERA	SAFARI	CHROME
Yes, 9+	Yes, 3.6+	Yes, 10.5+	Yes	Yes

WORKAROUNDS FOR NON-SUPPORTING BROWSERS

IE 8 and earlier plus older versions of Firefox and Opera don't support multiple background images. In a case like this, where the additional images are just extra decoration, you don't have to worry about providing workarounds. They'll still see the lined-paper background, which is a complete image in itself, with no clue that anything's lacking.

However, there may be times when missing out on the extra images would create an overall effect that looks incomplete or broken. For instance, if you're using multiple background images to create a complex button, with a left, middle, and right slice, the button will look broken if only one slice can be seen. Be careful about using multiple background images in cases like these, as you only have a few options for workarounds:

- **Use a single fallback image.** The simplest workaround for non-supporting browsers is to provide it with a single background image, either in a separate `background-image` declaration listed before the one using multiple images (the method we've used here) or by using Modernizr. Make sure this single image can stand on its own. This is easy to implement and doesn't harm supporting browsers, but it won't provide a sufficient appearance in cases where the page truly looks broken without the extra images.
- **Nest `divs` to hold extra images.** A more robust but work-intensive workaround than the single fallback image is to go back to the old method of nesting `divs` and applying separate images to separate boxes. If you do this, you'll need to use Modernizr or IE conditional comments to feed different rules to browsers with different support. Otherwise, you'd get double the backgrounds in browsers that support multiple background images. Of course, if you're going to be adding the extra `divs` and background rules anyway, you might as well stop using multiple background images at all and just use this old technique for all browsers, regardless of support. So I'm not sure that this workaround makes a lot of sense.
- **Generate the extra elements to hold extra images.** A cleaner way of implementing the "Nest `divs`" workaround is to use the `:before`

and `:after` pseudo-elements to generate extra elements, to which you can then apply extra background images. The article "Multiple Backgrounds and Borders with CSS 2.1" by Nicolas Gallagher (http://nicolasgallagher.com/multiple-backgrounds-and-borders-with-css2) explains how to do this. This would work well for IE 8 and Firefox 3.5, for instance, but IE 6 and 7 don't support these pseudo-elements, making this technique fail to work in those browsers—unless you also added a script to force older versions of IE to support these selectors. And you'd need to make sure browsers that do support multiple background images don't see the images on the pseudo-elements. At this point, the workaround would be getting pretty complicated! Again, you'll have to decide if what may amount to simply extra decoration is worth it for you and your users.

- **Use `canvas`.** If you're comfortable with scripting, you can use the HTML5 `canvas` element to draw multiple images on a single element. IE 8 and earlier don't support `canvas`, but Google's "explorer-canvas" script (http://code.google.com/p/explorercanvas) can make it work. Explaining how to use `canvas` is beyond the scope of this book, but Hans Pinckaers' mb.js script (http://github.com/HansPinckaers/mb.js) has already done the work for you, making multiple backgrounds work in IE and older non-IE browsers.

Adding a Graphic Border

Another graphic detail that would be nice to add is a border on the left side of the paper to make it look like it was torn from a spiral notebook (**Figure 3.12**). There are a couple ways we can do this with CSS3.

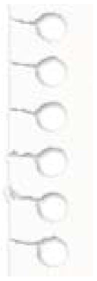


FIGURE 3.12 Torn spiral notebook-paper edge

USING BACKGROUND IMAGES

One way to add the torn paper edge is by adding it as another background image, set to repeat down only. But the edge image has transparent areas in it (the holes in the paper), so the lines background image below it will show through. If our page had a solid background color instead of a pattern, we could fill the transparent areas of the edge image with that solid color, obscuring the lines background image and blending into the page background color seamlessly. But that won't work in our page.

Without a solid background color on the page, your only option is to wrap another `div` around the paper `div`, and set the edge image as the background on this wrapper `div`. You could then give the wrapper enough left padding to keep the inner `div` from overlapping the edge image and obscuring it. This wouldn't be ideal, since it would add extra markup, but it would work in all browsers and with all page backgrounds.

One small disadvantage of setting the edge image as a background is that we can't control how it gets clipped at the bottom of the `div`. It's possible that the `div` would end in the middle of one of the holes in the edge, which isn't what a full sheet of real spiral notebook paper looks like (**Figure 3.13**). I will admit this is hardly a tragedy—it's a very minor, nitpicky problem. But if we can fix the problem easily with CSS, why not fix it?

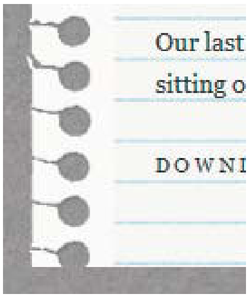


FIGURE 3.13
With the edge image as a repeating background image, it can get cut off in the middle of one of the holes.

The CSS3 solution is to set `background-repeat` on the edge image to `round`—a new value for the property introduced in CSS3. This makes the browser repeat the image as many times as it will fit, and if it doesn't fit a whole number of times, the browser rescales the image so that it will fit without clipping off at the end.

Unfortunately, only IE 9 and Opera support the `round` value at the time of this writing, and Opera does so imperfectly. So, `background-repeat: round` is not a usable solution right now. Luckily, we can forgo using a background image entirely and use the new `border-image` property instead.

USING border-image

CSS3 allows you to assign an image to a border, in addition to (or instead of) a color and line style. The browser will take a single image, slice it into pieces, and stretch or tile each of those pieces across each border.

For instance, let's say that **Figure 3.14** is the image we want to use for the borders on a `div`. We want to use the top 30 pixels of the image for the top border, the right 25 pixels for the right border, the bottom 27 pixels for the bottom border, and the left 34 pixels for the left border (**Figure 3.15**). We need to use these values as both our border widths and our border image slice locations.

NOTE: You can actually use different values for the border widths and the corresponding image slice locations. The browser will scale each image slice to fit the border width it's applied to.

```
.clouds {
   width: 400px;
   height: 150px;
   border-width: 30px 25px 27px 34px;
   border-image: url(clouds.png) 30 25 27 34 stretch;
}
```

FIGURE 3.14 This image has irregular borders that can be stretched and tiled using `border-image`.

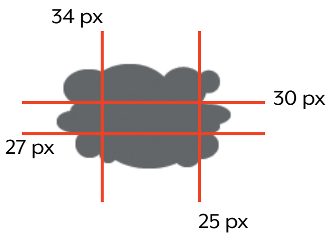


FIGURE 3.15 The lines indicate where we want to virtually slice the image into pieces that can be tiled or stretched across the borders.

NOTE: Strangely, you must leave the "px" unit off the slice values in the `border-image` property. Or, you can use percentages for slice values, relative to the image itself; in this case, you must include the % sign after the number.

The first part of the `border-image` value is the path to the image, which works just like any other path in CSS.

Next comes one or more numbers to specify where the browser should slice the image. In this case, we're using four numbers, since we want four different amounts sliced off from each edge. The first number, `30`, is the inward offset from the top edge of the image, in pixels. The second number, `25`, is the inward offset from the right edge, the third is the offset from the bottom, and the fourth is the offset from the left. The browser will slice the image at each of these lines, creating nine images that it applies to each border, each corner, and the middle of the box.

THE CENTER SLICE

The center slice of the border image is used to cover the entire middle area of the box, inside the border area. This doesn't seem very intuitive, but it does give you more styling options. If you don't want the middle of the border image to obscure the background image or color beneath it, use your graphics program to make the middle portion of the image you're using transparent, and save the image as a transparent GIF or PNG.

The spec says that this center slice should be discarded by default, unless you add the word `fill` to your `border-image` value. However, right now no browser seems to support the `fill` keyword, and they all "fill" by default, with no option to "not fill."

THE LOWDOWN ON THE `border-image` PROPERTY

The `border-image` property is part of the Backgrounds and Borders module, found at www.w3.org/TR/css3-background. It's a shorthand property, but you can't use the individual properties right now, since no browser supports them declared outside of the shorthand `border-image` property.

In the `border-image` property, you specify an image, how far in from each edge you want the browser to slice the image, and how to repeat each image (except the corners) across its border.

You can use one to four slice values, depending on whether each side needs to be sliced differently. One value applies the same slice offset to all four sides; two values applies the first to the top and bottom and the second to the right and left; three values applies the first to the top, second to the right and left, and third to bottom; and four values applies each to an individual side, starting at the top edge and going clockwise. See Figure 3.15 for a diagram of where the browser slices a border image.

The repeat value can be set to `stretch`, `repeat`, `round`, or `space`. Using one repeat value will apply the value to all four sides, while two repeat values applies the first value to the top and bottom borders and the second value to the left and right sides. A value of `repeat` will tile all four edges plus the center; `stretch` will stretch them to fill the area; `round` will tile and scale them so each fits a whole number of times; and `space` will tile them so each fits a whole number of times and then evenly distribute the extra space between the tiles.

Remember to always set `border-width` in conjunction with `border-image` to create a border area for the image to draw onto. There is also a `border-image-width` property, but no browser currently supports it, nor does any browser currently support `border-image-outset`.

Sadly, border images don't conform to curved borders created by `border-radius`.

Other than creating a torn-edge look, you might want to use `border-image` for:

- Buttons; see http://ejohn.org/blog/border-image-in-firefox
- Gradient backgrounds
- Scalloped edges to create the effect of a stamp or raffle ticket
- Graphic edges to create the effect of a picture frame or certificate; see www.norabrowndesign.com/css-experiments/border-image-frame.html
- A curved or angled edge of a box
- Box drop shadows that are curved or angled (`box-shadow` can do only straight drop shadows, but you can create an image of an irregular shadow and apply it as a border image)

TABLE 3.3 `border-image` **browser support**

IE	FIREFOX	OPERA	SAFARI	CHROME
No	Partial with -moz-, 3.5+	Partial, 10.5+	Partial with -webkit-	Partial

How exactly the browser applies these images depends on the third part of the `border-image` property: the repeat value. In this example, we're using a value of `stretch`, which will make the browser stretch all four border images, plus the center (but not the corners), to fill the available space (**Figure 3.16**). You can also set it to `repeat` (**Figure 3.17**), `round` (**Figure 3.18**), or `space`. (The `round` value is supported only by Firefox and Opera currently.) No browser currently supports the `space` value, so I can't show you a screenshot!

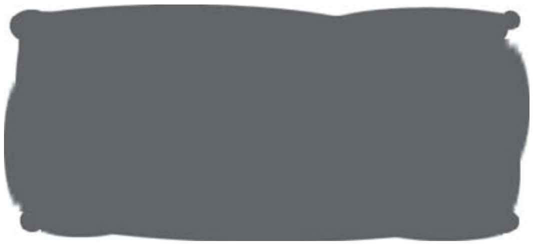


FIGURE 3.16 This `border-image` has been stretched.

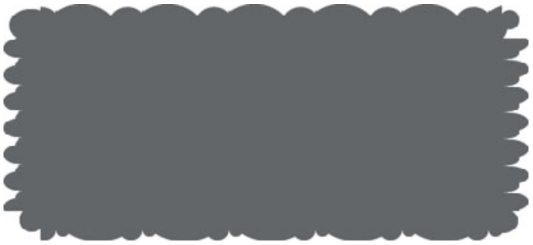


FIGURE 3.17 This `border-image` has been repeated.

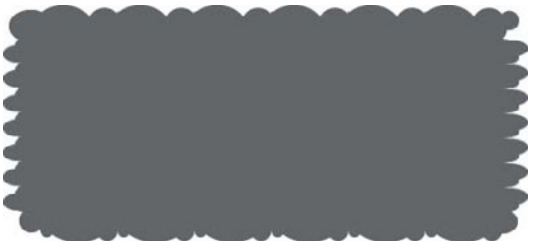


FIGURE 3.18 This `border-image` has been rounded.

APPLYING THE TORN-EDGE IMAGE

Let's put `border-image` to use in our page to apply the torn-paper edge image, shown in Figure 3.12, to the article `div`. We want to apply the image only to the left border, so we'll make that border 50 pixels wide—the width of the image—and set the other borders to zero:

```
#paper {
   float: left;
   margin: 40px;
   padding: 3.2em 1.6em 1.6em 1.6em;
   border-width: 0 0 0 50px;
   background: url(images/paperlines.gif) #FBFBF9;
   background: url(images/thumbtack.png) 50% 5px no-repeat,
               url(images/stains1.png) 90% -20px no-repeat,
               url(images/stains2.png) 30% 8% no-repeat,
               url(images/stains3.png) 20% 50% no-repeat,
               url(images/stains4.png) 40% 60% no-repeat,
               url(images/paperlines.gif) #FBFBF9;
   -moz-background-size: auto, auto, auto, auto, auto,
                         auto 1.6em;
   -webkit-background-size: auto, auto, auto, auto, auto,
                            auto 1.6em;
   background-size: auto, auto, auto, auto, auto,
                    auto 1.6em;
}
```

PLAYING WITH BORDER IMAGES

The `border-image` property is pretty confusing—I won't deny it. If, after walking through the examples provided, you're still feeling a little unsure, I highly recommend you check out these border image web tools:

- "border-image-generator" by Kevin Decker (http://border-image.com) allows you to upload any image to see how it will look when applied as a border image. You can change the slice offsets, border widths, and repeat method and instantly see how your border image changes.
- "Grokking CSS3 border-image" by Nora Brown (www.norabrowndesign.com/css-experiments/border-image-anim.html) uses five preset images and lets you change between a few preset `border-image` values to see how the images are affected.

Getting to change values on the fly and see how they affect the visual output is one of the best ways to learn how a piece of CSS works.

Next, we'll apply the border image, using the standard `border-image` property for Chrome and Opera and the prefixed properties for Firefox and Safari:

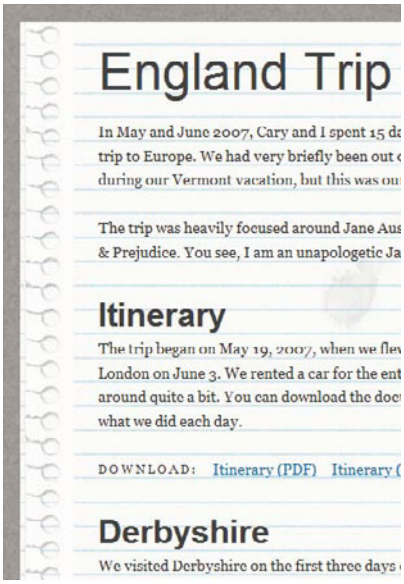
```
#paper {
   float: left;
   margin: 40px;
   padding: 3.2em 1.6em 1.6em 1.6em;
   border-width: 0 0 0 50px;
   -moz-border-image: url(images/edge.png) 0 0 0 50 round;
   -webkit-border-image: url(images/edge.png) 0 0 0 50
   ¬ round;
   border-image: url(images/edge.png) 0 0 0 50 round;
   background: url(images/paperlines.gif) #FBFBF9;
   background: url(images/thumbtack.png) 50% 5px no-repeat,
               url(images/stains1.png) 90% -20px no-repeat,
               url(images/stains2.png) 30% 8% no-repeat,
               url(images/stains3.png) 20% 50% no-repeat,
               url(images/stains4.png) 40% 60% no-repeat,
               url(images/paperlines.gif) #FBFBF9;
   -moz-background-size: auto, auto, auto, auto, auto,
                         auto 1.6em;
   -webkit-background-size: auto, auto, auto, auto, auto,
                            auto 1.6em;
   background-size: auto, auto, auto, auto, auto,
                    auto 1.6em;
}
```

We've set each of the slice locations to zero except for the left one; we don't want to slice off any from the top, right, or bottom, but we do want to slice in from the left edge by 50 pixels so that the entire 50-pixel-width of the image is used for the left border.

For the repeat value, we've used `round` to repeat the image but keep it from ending in the middle of a hole. Since Safari and Chrome don't support this value, they treat it as `repeat` instead, which is an acceptable second choice.

USING `background-clip` TO POSITION IMAGES

Our edge image is now repeating down the left side of the `div`, but the background image is showing through it (**Figure 3.19**). That's because, by default, borders are drawn on top of the background area. You may have never noticed it before, because usually your borders are just solid lines, without any transparent pieces. But change your `border-style` to `dashed` and you'll see what I mean. Border images are placed the same way.



England Trip

In May and June 2007, Cary and I spent 15 da
trip to Europe. We had very briefly been out o
during our Vermont vacation, but this was ou

The trip was heavily focused around Jane Aus
& Prejudice. You see, I am an unapologetic Ja

Itinerary

The trip began on May 19, 2007, when we flew
London on June 3. We rented a car for the ent
around quite a bit. You can download the doc
what we did each day.

DOWNLOAD: Itinerary (PDF) Itinerary (

Derbyshire

We visited Derbyshire on the first three days

FIGURE 3.19 The torn-edge image repeats down the left side, but overlaps the background.

ORDER OF THE BACKGROUND PROPERTIES

Normally, the order I write the properties in each rule is irrelevant; it's just a standard order that I always use, and you can feel free to reorder the properties however you like. In the case of `background-clip`, however, make sure to write it *after* the shorthand `background` property, as shown, because `background-clip` can be included in the shorthand `background` property (see Figure 3.10). If you write `background-clip` separately first, and then write the `background` property without any `background-clip` information in it, you're effectively telling the browser you want to use the default value of `border-box`, overriding the earlier `background-clip` values.

So why not just include the `background-clip` value we want in the shorthand `background` property? We can't, for the same reasons we can't include the `background-size` values in the `background` property right now: some browsers need prefixes and don't yet understand the standard property, by itself or in the `background` shorthand property.

THE LOWDOWN ON THE `background-clip` PROPERTY

The `background-clip` property is part of the Backgrounds and Borders module, found at www.w3.org/TR/css3-background. It controls under which sections of a box the background is painted.

The allowed values are `border-box` (the default value to paint backgrounds under borders), `padding-box` (to clip backgrounds at the outer edge of the padding area and not extend under borders), and `content-box` (to clip backgrounds at the outer edge of the content area and not extend under padding or borders). Firefox 3.6 and earlier don't support `content-box`, and use values of `border` and `padding`, not `border-box` and `padding-box`; Firefox 4 doesn't have these issues. Safari 5 supports the `border-box` and `padding-box` values in the standard `background-clip` property, but supports only `content-box` in the `-webkit-background-clip` property.

Webkit also supports a value of `text`, available only in the `-webkit-` prefixed property, which makes the text act like a mask on the background image, obscuring whatever parts of the background image are not behind the text. It's a cool effect, but probably won't make it into the spec. For more information and examples, see www.css3.info/webkit-introduces-background-cliptext, http://trentwalton.com/2010/03/24/css3-background-clip-text, and http://trentwalton.com/2010/04/06/css3-background-clip-font-face.

Other than moving a background out from under a border image, you might want to use `background-clip` for:

- Moving a background color or image out from under a dashed or dotted border
- Creating the appearance of a double border, one made from the actual border and one made from the padding, by using `content-box`
- Keeping the background color from bleeding outside the edges of rounded corners, as sometimes happens in Webkit-based browsers, by using `padding-box`; see http://tumble.sneak.co.nz/post/928998513/fixing-the-background-bleed

TABLE 3.4 `background-clip` **browser support**

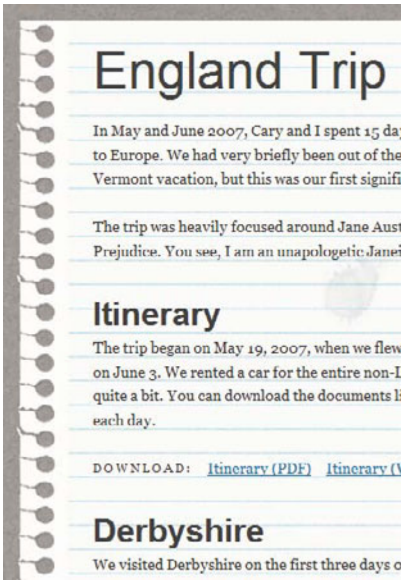
IE	FIREFOX	OPERA	SAFARI	CHROME
Yes, 9+	Yes, 4+; Partial, 1+, with -moz-	Yes	Yes, 3+, with -webkit-; Partial, 5+	Yes

Luckily, we can change this default behavior with CSS3. CSS3 lets you control where backgrounds are placed relative to the borders with the new `background-clip` property. The default value, `border-box`, makes backgrounds extend under the borders as they've always done. Setting `background-clip` to `padding-box` starts the backgrounds inside the borders, under the padding area:

```
#paper {
   float: left;
   margin: 40px;
   padding: 3.2em 1.6em 1.6em 1.6em;
   border-width: 0 0 0 50px;
   -moz-border-image: url(images/edge.png) 0 0 0 50 round;
   -webkit-border-image: url(images/edge.png) 0 0 0 50
   ¬ round;
   border-image: url(images/edge.png) 0 0 0 50 round;
   background: url(images/paperlines.gif) #FBFBF9;
   background: url(images/thumbtack.png) 50% 5px no-repeat,
               url(images/stains1.png) 90% -20px no-repeat,
               url(images/stains2.png) 30% 8% no-repeat,
               url(images/stains3.png) 20% 50% no-repeat,
               url(images/stains4.png) 40% 60% no-repeat,
               url(images/paperlines.gif) #FBFBF9;
   -moz-background-size: auto, auto, auto, auto, auto,
                         auto 1.6em;
   -webkit-background-size: auto, auto, auto, auto, auto,
                            auto 1.6em;
   background-size: auto, auto, auto, auto, auto,
                    auto 1.6em;
   -moz-background-clip: padding;
   -webkit-background-clip: padding-box;
   background-clip: padding-box;
}
```

Chrome, Safari 5, Firefox 4, and Opera use the standard property, while Firefox 3.6 and earlier and Safari 4 and earlier use the prefixed versions. Note also that the `-moz-background-clip` property takes a value of `padding` instead of the standard `padding-box`.

Making this change moves the lines background image out from under the border image (**Figure 3.20**).



England Trip

In May and June 2007, Cary and I spent 15 da
to Europe. We had very briefly been out of the
Vermont vacation, but this was our first signifi

The trip was heavily focused around Jane Aust
Prejudice. You see, I am an unapologetic Janei

Itinerary

The trip began on May 19, 2007, when we flew
on June 3. We rented a car for the entire non-I
quite a bit. You can download the documents li
each day.

DOWNLOAD: Itinerary (PDF) Itinerary (\

Derbyshire

We visited Derbyshire on the first three days o

FIGURE 3.20
The `background-clip` **property moves the background image out from under the border image.**

WORKAROUNDS FOR NON-SUPPORTING BROWSERS

Browsers that don't support `border-image` won't know what they're missing, in this case, as they'll still see the regular lined-background image. If you must have the torn edge, you can go back to using a background image for it on an additional wrapper `div`, as described earlier.

If you do this, you'll either need to remove the `border-image` from all the other browsers, or you'll need to hide the background image from the browsers that support `border-image`. I like the second approach, as it allows you the extra flexibility of having border images without too much extra work. Simply use Modernizr or IE conditional comments to create a wrapper rule that only certain browsers can see. This rule would assign left padding and the edge background image:

```
#wrapper {
   padding-left: 50px;
   background: url(images/edge.png) repeat-y;
}
```

The other browsers wouldn't see this rule at all. They'd still see the wrapper `div` in the HTML, of course, but they wouldn't apply any styles to it.

Alternately, you could combine the lined paper image with the torn edge image and apply this merged image to the existing `div` named paper. That would allow you to do away with the extra wrapper `div`, but it may be more work-intensive to have to maintain different images for different browsers. Again, you'd need to make sure that browsers that do support `border-image` continue to use the two separate images—one as the background and one as the border image.

There are a few ways to make `border-image` work through scripting, rather than ditching it in favor of background images. However, the scripting solutions work only when you're stretching the border images, not repeating or rounding them, so a script won't do in our case. But if your own project just needs stretched border images, check out:

- PIE by Jason Johnston (http://css3pie.com), described in Chapter 2. PIE also includes limited support for the `border-image` property in IE 6 through 8.
- borderImage by Louis-Rémi Babé (http://github.com/lrbabe/borderimage), a jQuery plugin that emulates `border-image` using VML for IE and `canvas` for non-IE browsers. You can find more description of how to use it at www.lrbabe.com/sdoms/borderImage.

Adding a Drop Shadow

In Chapter 2, you learned about the `box-shadow` property to create drop shadows beneath boxes. Our notebook-paper article seems like a good place for it as well, so let's add it. But we have to be careful—the drop shadow won't conform to the ragged edge of the border image, but rather to the box as a whole. That means that if the drop shadow shows on the left side of the box, you'll end up with a strange-looking straight-edged shadow that's slightly offset from the jagged paper edge (**Figure 3.21**).

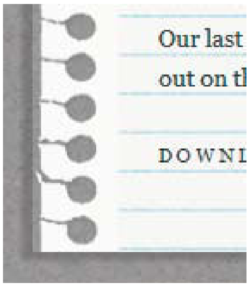


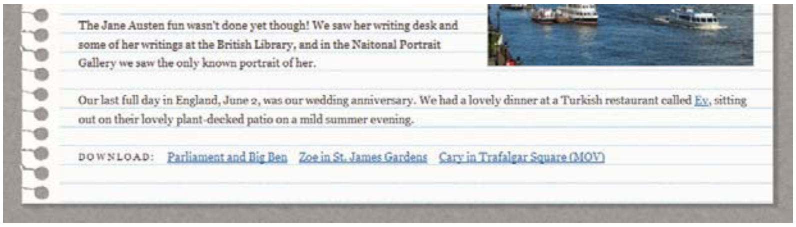
FIGURE 3.21
Drop shadows conform to the `div`'s straight edge, not to any jagged lines within border or background images.

NOTE: The page with all the changes to this point is named paper_2.html in the exercise files that you downloaded for this chapter.

To avoid this problem, place the shadow far enough to the right to not peek out at all on the left edge. Add the following three lines to the `#paper` rule:

```
-moz-box-shadow: 6px 5px 3px hsla(0,0%,0%,.2);
-webkit-box-shadow: 6px 5px 3px hsla(0,0%,0%,.2);
box-shadow: 6px 5px 3px hsla(0,0%,0%,.2);
```

This creates a shadow below the right and bottom edges of the paper (**Figure 3.22**).

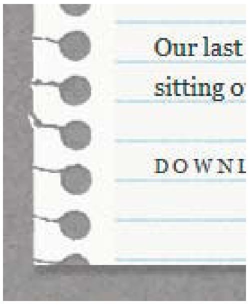


The Jane Austen fun wasn't done yet though! We saw her writing desk and some of her writings at the British Library, and in the Naitonal Portrait Gallery we saw the only known portrait of her.

Our last full day in England, June 2, was our wedding anniversary. We had a lovely dinner at a Turkish restaurant called Ev, sitting out on their lovely plant-decked patio on a mild summer evening.

DOWNLOAD: Parliament and Big Ben Zoe in St. James Gardens Cary in Trafalgar Square (MOV)

FIGURE 3.22 This drop shadow works better, showing on the right side of the `div`.



Our last
sitting o
DOWNL

FIGURE 3.23 The drop shadow in Safari or Chrome might show up under an empty hole.

Since Safari and Chrome don't support `round` for the repeat value on the border image, it's possible to get a cut-off hole at the bottom of the paper, making the shadow underneath it look a little strange (**Figure 3.23**). It's not very noticeable, but if this really bothers you, remove the `-webkit-box-shadow` declaration. (Sometimes it's nice having each browser declared separately, isn't it!)

Of course, now the drop shadow will be gone in Webkit-based browsers. To create a drop shadow in Safari and Chrome without using `-webkit-box-shadow`, you could create an image of a shadow and apply it as a border image to the right and bottom borders, using the `-webkit-border-image` property.

Embedding Unique Fonts

We've done a lot of work on the background of the article so far. Now let's apply some extra styling to the actual content. We can use `@font-face` rules to make the headings look like they are handwritten—and this trick even lets IE in on the fun.

What is @font-face?

The `@font-face` rule is a way of linking to fonts on your server (just as you can link to images) that the browser downloads into its cache and uses to style the text on the page. It's often called *font embedding* (though the fonts aren't truly embedded anywhere), and the fonts that are "embedded" are called *web fonts*.

The `@font-face` rule was actually part of CSS 2 back in 1998, but was removed from the CSS 2.1 specification. It's now back, in CSS3, and finally has widespread browser support.

Until now, without web fonts, web designers have been limited to the small handful of common fonts installed on all users' computers, called *web-safe fonts*. Designers who didn't want to use just Arial, Verdana, or Georgia (among a few more) would have to resort to images, Flash, or scripting to create their text using unique fonts. These font-replacement techniques all suffer from accessibility and usability problems to varying degrees. They're also much more work-intensive to implement and maintain, and they can degrade the performance of your pages.

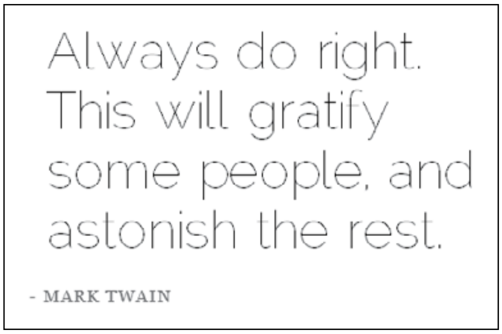
Using `@font-face`, on the other hand, keeps real text in the page. You don't have to depend on the user having the Flash plugin installed or JavaScript operating. You don't have to create any images or scripts, and your users don't have to download them. The work involved to implement it can be as simple as writing CSS like this:

```
@font-face {
   font-family: Raleway;
   src: url(fonts/raleway_thin.otf);
}
h1 {
   font-family: Raleway, "HelveticaNeueLt Std Thin",
   ¬"Helvetica Neue Light", "HelveticaNeue-Light",
   ¬"Helvetica Neue", Helvetica, Arial, sans-serif;
}
```

This tells the browser to use the raleway_thin.otf font file to render the text inside the `h1` element (**Figure 3.24**). If the user's browser doesn't support `@font-face` or can't download the file for some reason, the browser simply works through the font stack for a fallback. The *font stack* is the list of fonts declared in the `font-family` property, which the browser tries to load from the user's machine, in order, until it finds a font it can use.

NOTE: For the considerations that should go into crafting a good font stack, as well as many links to proven font stacks and other resources, see http://nicewebtype.com/notes/2009/04/23/css-font-stacks.

FIGURE 3.24
Using `@font-face`, you can display a non-standard font.


Always do right. This will gratify some people, and astonish the rest.
- MARK TWAIN

As you might have suspected, however, using `@font-face` is more complicated in the real world.

Choosing Acceptable Fonts

One of the big issues with web fonts is that not every font ought to be used in web pages. Some fonts have licensing restrictions that forbid such a use, while others simply don't look good on the web.

LICENSING ISSUES

NOTE: There are ways you can make your font files more secure. See http://subjectiveobject.com/2009/10/28/securing-font-face for a brief discussion of options, as well as http://typefront.com.

When choosing a font to use, read its license—often called an end-user license agreement (EULA) or terms of use—to see if it allows web font embedding. Many fonts' licenses don't, because when you use `@font-face`, the font file is downloaded into the user's cache, just like images. The user could go into her cache, take the font file, and install it on her system. Most font vendors are not interested in simply giving their products away to the thousands of people who browse your web site.

Of course, not many users are really going to go to this trouble. But Richard Fink describes the bigger problem font vendors have with font embedding in his article "Web Fonts at the Crossing" (www.alistapart.com/articles/fonts-at-the-crossing):

> *The fear is that once fonts are on the web, they will become a commodity, the current model will break, and a devaluation of fonts, in general, will occur. The fear is that font designers will no longer be able to charge a print customer, say, $420 for a four-style font family with a 6–10 user license in a world where fonts are being delivered on web sites to virtually unlimited numbers of "users" who don't have to pay anything at all. What if the web drives down prices in the print sector and doesn't generate much revenue on its own?*

Unfortunately, most fonts' licenses were not written with `@font-face` in mind, so when you read through a font's license, it may not say anything about not embedding fonts. Lack of a restriction doesn't mean you have a free pass to use the font. It's best to err on the side of caution and not use the font unless it explicitly says that web embedding or redistribution is OK.

NOTE: Although you still ought to look at each font's license to be sure, a general guide as to which foundries allow font embedding and font replacement methods is available at http://webfonts.info/wiki/index.php?title=Web_fonts_licensing_overview.

This is the case even with free fonts. Just because the font vendor gave you the font for free doesn't mean you can redistribute it. Same thing with the fonts that came with your computer. Again, you have to check the license to be sure.

Luckily, there are many places online to find fonts whose licenses allow web font embedding:

- **The League of Moveable Type** (www.theleagueofmoveabletype.com) is a small but growing collection of free, open-source fonts that are specifically provided for `@font-face` use. The Raleway font used in Figure 3.24 is one of these fonts.
- The Webfonts.info wiki has a page called **"Fonts available for @font-face embedding"** (http://webfonts.info/wiki/index.php?title=Fonts_available_for_%40font-face_embedding) that lists fonts (mostly free) whose licenses permit embedding. But like most wiki pages, it's not always as up-to-date and comprehensive as it could be.
- **Font Squirrel** (www.fontsquirrel.com) provides a large collection of free fonts whose licenses allow embedding. It also provides some handy tools for working with `@font-face`, as we'll talk about in a bit.
- Google has a library of free fonts for embedding called **Google Font Directory** (http://code.google.com/webfonts). You link to one of the fonts on their server using the Google Fonts API, which has a number of advantages (see http://mindgarden.de/benefit-of-the-google-font-api). But you can also download the fonts at http://code.google.com/p/googlefontdirectory/source/browse and host them yourself.
- Most of the fonts available at **Kernest** (www.kernest.com) are free, and all are specifically provided for `@font-face` use. Some are hosted by Kernest, but most you can download and host yourself.
- **exljbris** (www.josbuivenga.demon.nl) and **Fontfabric** (http://fontfabric.com) both provide a number of fonts for free that

can be embedded on the web, as long as you provide attribution according to the terms in the EULAs.

NOTE: As you can imagine, the list of resources shown here is likely to change and grow. To see the most up-to-date list of web font sources, go to www.stunningcss3.com/resources.

- All of the fonts at **Fonthead** (www.fonthead.com) are allowed to be used with `@font-face` as well as other text replacement methods.
- **FontSpring** (www.fontspring.com/fontface) sells fonts that can be used both in a traditional way on your computer and in print work, as well as embedded on the web with `@font-face`.
- FontShop has created web versions of several fonts, called **Web FontFonts** (www.fontshop.com/fontlist/n/web_fontfonts), that you can buy separately from the traditional versions.

LETTING OTHERS DO THE HEAVY LIFTING

All the sources I listed for `@font-face`-ready fonts are places where you can download fonts to host on your own servers and then do the coding yourself. Another option is to let others do all this work for you using a font-embedding service, also called a type delivery service or font hosting and obfuscation service (FHOS).

These services offer a collection of fonts that their distributors have approved for web use through the service, getting around the licensing issues of `@font-face`. These fonts are hosted by the service, making them difficult or impossible to download and redistribute.

Font-embedding services are easy to use because they provide all the different font file formats needed for different browsers, as well as the code for you to add the fonts to your sites. This code may include JavaScript in addition to CSS in order to make the real fonts impossible to reuse or speed up their rendering. Most of these services are not free, though some have free options, and the pricing models vary, such as subscribing to a collection or paying per font and per site.

These services are popping up all over the place—many type vendors are creating their own services for their fonts only—but here are the major players:

- Typekit (http://typekit.com) is a subscription-based service where you pay yearly for access to a collection of fonts, which come from multiple foundries. The smallest collection is free, but has other use restrictions.

- Fontdeck (http://fontdeck.com) is a subscription-based service, but you pay for each font you want per year and per site, instead of paying a yearly fee for a collection of fonts. The fonts come from multiple foundries.
- Kernest (www.kernest.com) has a subscription model similar to Fontdeck, but nearly all of the fonts are free. The fonts come from multiple foundries. Some are hosted by Kernest, and most you can download and host yourself.
- Ascender offers two services: Web Fonts from Ascender (www.ascenderfonts.com/webfonts) and FontsLive (www.fontslive.com). Both have a subscription model similar to Fontdeck, and the fonts come from multiple foundries.
- WebINK (www.extensis.com/en/WebINK) has a subscription model similar to Typekit, but you pay a monthly fee based on the fonts' pricing tier and your bandwidth usage. The fonts come from multiple foundries.
- Webtype (www.webtype.com) has a subscription model similar to Fontdeck, but pricing varies based on the bandwidth you use. The fonts come from multiple foundries. You can also purchase traditional versions of the fonts to download and use on your desktop.
- Typotheque (www.typotheque.com/webfonts) offers a service for fonts from only its foundry, where you pay a one-time fee per font.
- Just Another Foundry (http://justanotherfoundry.com/webfonts) also offers a service for fonts from its foundry only, but you pay a yearly subscription fee.
- Fonts.com Web Fonts (http://webfonts.fonts.com) has a subscription model similar to Typekit, but you pay monthly. The highest-priced plan allows you to also download fonts to use on your desktop, but you can use the installed font only so long as it's being used in a web site through their service.

If you're thinking about using one of these services, read and use the list of buyer considerations at the end of the article "Web Fonts at the Crossing" at www.alistapart.com/articles/fonts-at-the-crossing before choosing. To see the most up-to-date list of font-embedding services, go to www.stunningcss3.com/resources.

READABILITY AND RENDERING ISSUES

Once you've cleared the licensing hurdle, don't go crazy and start loading up your pages with all sorts of bizarre fonts. Every time you choose to use a web font, have a specific reason for picking that font, beyond just that it looks cool. Make sure that the font truly enhances the text and doesn't make it less readable.

TIP: The Soma FontFriend bookmarklet (http://somadesign.ca/projects/fontfriend) lets you easily test out the fonts in your font stacks, including web fonts, so you can quickly see how each one will look on your page.

Test your web fonts with your actual content to make sure they will work. The Raleway font shown in Figure 3.24 might work well for large headings but be too thin to render well and be readable for body copy. Most commercial fonts were not designed to be viewed at small sizes on a screen, so in many cases it makes the most sense to reserve `@font-face` for headings and continue to use web-safe fonts like Georgia and Lucida for body copy.

Another aspect of web fonts that can affect legibility is how they are anti-aliased and hinted. Right now, web fonts are generally more jagged around the edges than traditional fonts, even when anti-aliased, usually because most were not designed to be viewed on screen. Higher quality fonts, as well as fonts that were designed for the web, have better *hinting*, which, in a nutshell, is a set of instructions in the font file that adjusts the edges of the characters to line up better with the pixel grids of our computer screens so they look better to the human eye. Font format plays a role in this too; TrueType fonts are generally better hinted than OpenType CFF fonts. The degree of jaggedness depends not only on the font but on the operating system and sometimes the browser; Mac is generally smoother than Windows, but can look blurry. Windows XP in particular can look quite bad if the user hasn't enabled ClearType (Microsoft's current technology for improving text rendering on screen).

NOTE: It's possible to force differently sized fonts to match up in size using the `font-size-adjust` property, but currently only Firefox supports it. See http://webdesignernotebook.com/css/the-little-known-font-size-adjust-css3-property, as well as the links at the end of the article, for more information.

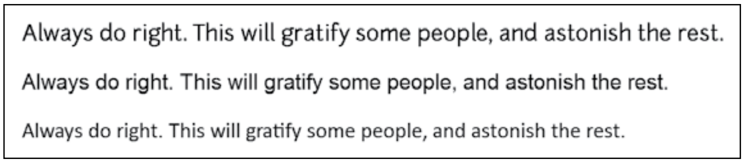
Not only is the readability of your web fonts important, but so too is the readability of the fallback fonts in your font stacks. Make sure to test the fallback fonts so that if the web font doesn't load, the user still gets readable and attractive text. You usually want to choose fallback fonts that have similar proportions to the web font you're putting at the front of your font stack. That way, the font size, weight, and other styles you apply to the text will work well with whatever font the user sees.

MORE ON FONT HINTING AND ANTI-ALIASING

Font hinting and anti-aliasing is a big, technical topic beyond the scope of this book, but if you'd like to learn more about it, check out these articles:

- "The Ails Of Typographic Anti-Aliasing" by Thomas Giannattasio (www.smashingmagazine.com/2009/11/02/the-ails-of-typographic-anti-aliasing) gives a good overview of anti-aliasing, hinting, sub-pixel rendering, and how various operating systems and browsers handle rendering web fonts.
- "Font Hinting Explained By A Font Design Master" by Richard Fink (http://readableweb.com/font-hinting-explained-by-a-font-design-master) and "Font Hinting" by Peter Bil'ak (www.typotheque.com/articles/hinting) give more detail on how hinting works.
- "Font smoothing, anti-aliasing, and sub-pixel rendering" by Joel Spolsky (www.joelonsoftware.com/items/2007/06/12.html) compares Apple and Microsoft's methods for smoothing on-screen text.
- "Browser Choice vs Font Rendering" by Thomas Phinney (www.thomasphinney.com/2009/12/browser-choice-vs-font-rendering) explains how browsers' text rendering is dependent on the operating system.
- Webkit-based browsers let you control the anti-aliasing mode using their proprietary `-webkit-font-smoothing` property. See "-webkit-font-smoothing" by Tim Van Damme (http://maxvoltar.com/archive/-webkit-font-smoothing) for examples and "Font Smoothing" by Dmitry Fadeyev (www.usabilitypost.com/2010/08/26/font-smoothing) for an argument against the property.

NOTE: Luckily, web font rendering is improving. For instance, IE 9 uses Microsoft's DirectWrite API to handle text rendering, making web fonts look very smooth; Firefox has said it will use DirectWrite in its Windows versions as well. Also, more and more font vendors are now selling web fonts, so as `@font-face` grows in popularity we will undoubtedly see more fonts for sale that are hinted aggressively for web use.

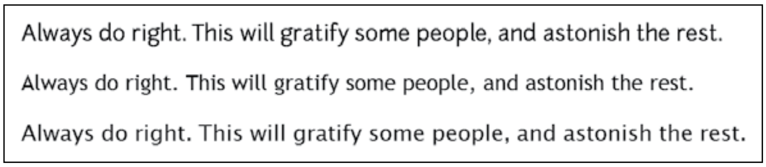


Always do right. This will gratify some people, and astonish the rest.

Always do right. This will gratify some people, and astonish the rest.

Always do right. This will gratify some people, and astonish the rest.

FIGURE 3.25 **Arial (center) and Calibri (bottom) are too small to be the best fall-backs for the Junction (top) web font.**



Always do right. This will gratify some people, and astonish the rest.

Always do right. This will gratify some people, and astonish the rest.

Always do right. This will gratify some people, and astonish the rest.

FIGURE 3.26 **Trebuchet MS matches up well with Junction, with Lucida Sans Unicode being a good runner-up.**

Browser Support

So once you've chosen a font that has the correct license and is legible on the web, all you need to do is link to it in an `@font-face` rule as shown earlier and you're done, right? Well, not quite. The `@font-face` rule has good browser support, but different browsers want you to use different font file types.

TrueType (TTF) and OpenType (OTF) font files, such as the ones you probably already have on your computer, work in most browsers.

IE supports `@font-face` as far back as version 4, but IE 4 through 8 support it only if you use a proprietary font format called Embedded OpenType (EOT). EOT is technically not a font format; it's a compressed copy of a TTF font that uses digital rights management (DRM) to keep the font from being reused.

NOTE: For more information on SVG fonts, see "About Fonts in SVG" by Divya Manian (http://nimbupani.com/about-fonts-in-svg.html).

The only type of font file that works on Safari on iOS (the browser on the iPhone, iPod Touch, and iPad, and often called "Mobile Safari") is SVG (Scalable Vector Graphics). SVG also works on Chrome, desktop Safari, and Opera, but not Firefox. You're probably most familiar with SVG as a vector graphics format, but an SVG file can contain font information too—after all, each character in a font is really just a vector drawing.

Using these three formats—TTF or OTF, EOT, and SVG—will make your unique fonts show up in every browser that supports `@font-face`. But you should also include a fourth format, WOFF, for future compatibility.

NOTE: Learn more about WOFF at www.w3.org/Fonts/WOFF-FAQ.

WOFF, which stands for Web Open Font Format, was introduced in 2009. Like EOT, WOFF is not technically a font format, but rather a compressed wrapper for delivering TTF or OTF fonts. Unlike EOT, however, WOFF contains no DRM. So far, the only browsers that support WOFF are Firefox 3.6 and later, Chrome 6, and IE 9, but the other major browsers are all working on adding support for it, and many font vendors have also expressed support. The WOFF specification became a W3C working draft in July 2010, so it's now officially on its way to becoming the standard web font format. Going forward, it's the one to use.

But don't get too overwhelmed by all these acronyms and browsers. As you'll learn in the next section, it's easy to create all the different formats you need. Check out Table 3.5 for a summary of which browsers support which font types.

NOTE: The compression in EOT and WOFF files is lossless. These fonts should look just as good as their TTF or OTF originals.

NOTE: This is a rapidly changing area of CSS support. See http://webfonts.info/wiki/index.php?title=%40font-face_browser_support and www.stunningcss3.com/resources for the latest information.

TABLE 3.5 `@font-face` **file types browser support**

	WOFF	OTF	TTF	SVG	EOT
IE	9	no	9	no	4
Firefox	3.6	3.5	3.5	no	no
Opera	no	10	10	10	no
Opera Mobile	no	9.7	9.7	9.7	no
Safari	no	3.1	3.1	3.1	no
Chrome	6	4*	4*	0.3	no
Safari on iOS	no	no	no	3.1	no

* Chrome 3 supported OTF and TTF fonts, but not by default—you had to do a command-line switch to enable it.

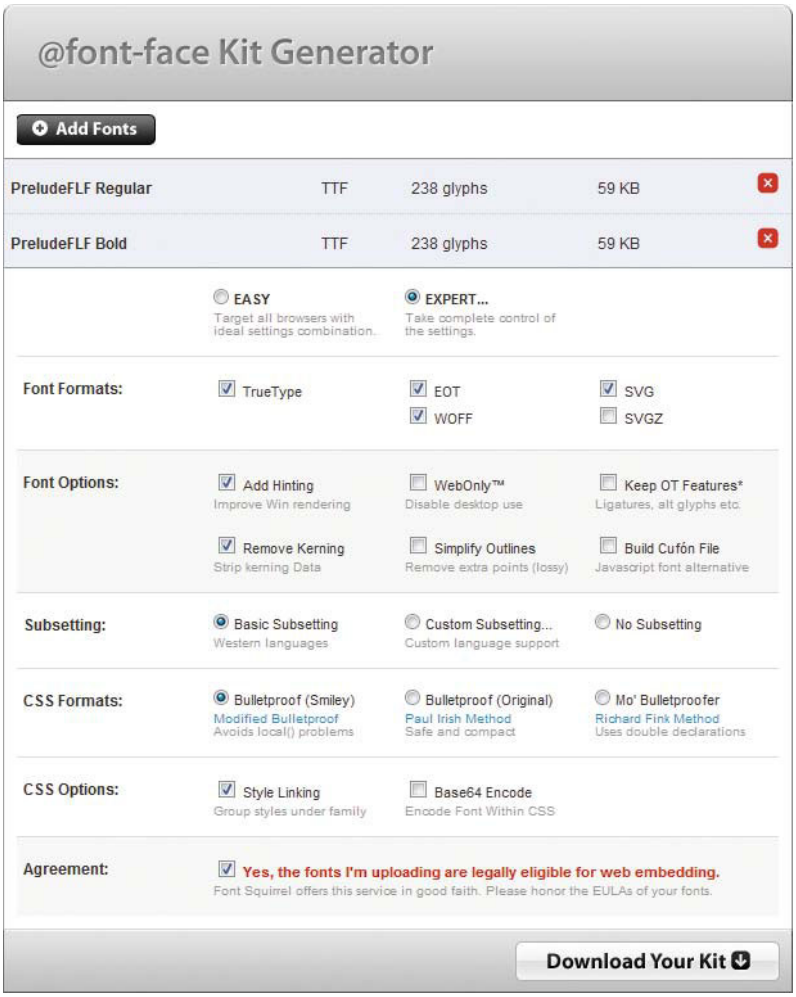
Each browser version number noted in Table 3.5 is the earliest—not the only—version of that browser to support that type.

Converting Fonts

Some providers of `@font-face`-ready fonts supply you with all the different font formats you need for the different browsers. For instance, Font Squirrel offers something they call "@font-face kits," each of which includes the original TTF or OTF font, an SVG version, a WOFF version, an EOT version, and a sample style sheet and HTML page showing the `@font-face` rules you need to place in your CSS. You can download these kits at www.fontsquirrel.com/fontface.

Even better is Font Squirrel's @font-face Kit Generator (www.fontsquirrel.com/fontface/generator). You can upload your font and convert it to whichever formats you wish. You can also control the CSS syntax it outputs, subset the characters to reduce file size, and use more options to fine-tune the fonts (**Figure 3.27**).

FIGURE 3.27
Font Squirrel's @font-face Kit Generator


@font-face Kit Generator

Add Fonts

PreludeFLF Regular	TTF	238 glyphs	59 KB
PreludeFLF Bold	TTF	238 glyphs	59 KB

EASY — Target all browsers with ideal settings combination.
[✓ EXPERT...] — Take complete control of the settings.

Font Formats: [✓ TrueType] [✓ EOT] [✓ WOFF] [✓ SVG] SVGZ

Font Options:
[✓ Add Hinting] — Improve Win rendering
WebOnly™ — Disable desktop use
Keep OT Features* — Ligatures, alt glyphs etc.
[✓ Remove Kerning] — Strip kerning Data
Simplify Outlines — Remove extra points (lossy)
Build Cufón File — Javascript font alternative

Subsetting:
[✓ Basic Subsetting] — Western languages
Custom Subsetting... — Custom language support
No Subsetting

CSS Formats:
[✓ Bulletproof (Smiley)] — Modified Bulletproof — Avoids local() problems
Bulletproof (Original) — Paul Irish Method — Safe and compact
Mo' Bulletproofer — Richard Fink Method — Uses double declarations

CSS Options:
[✓ Style Linking] — Group styles under family
Base64 Encode — Encode Font Within CSS

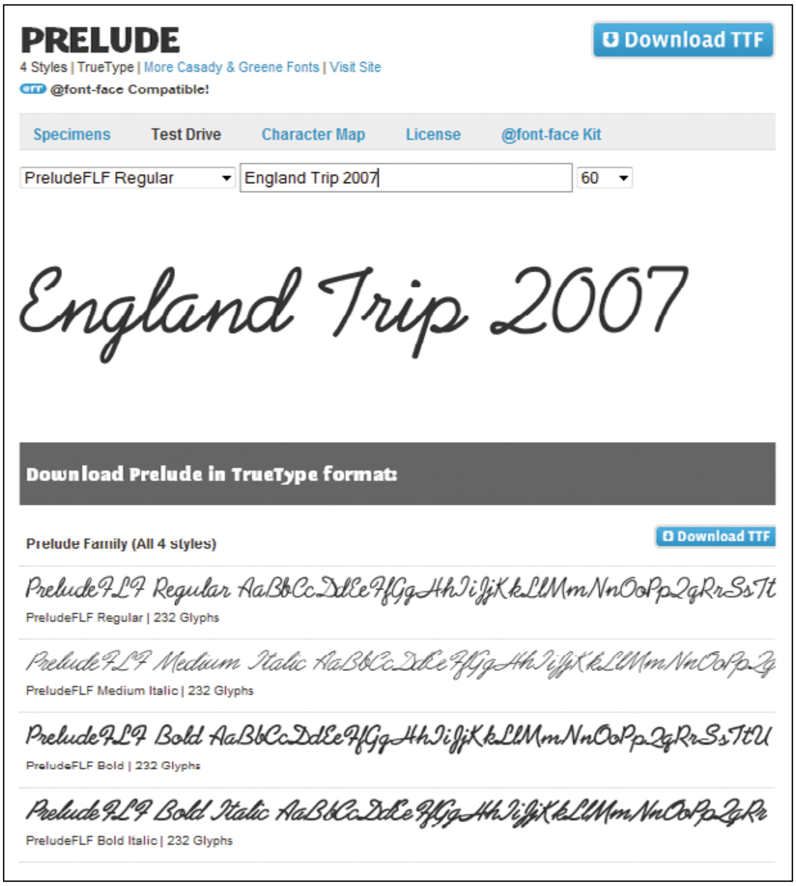
Agreement: [✓ Yes, the fonts I'm uploading are legally eligible for web embedding.]
Font Squirrel offers this service in good faith. Please honor the EULAs of your fonts.

Download Your Kit

The files Font Squirrel produces are usually all you'll need, but there are a couple of other tools worth mentioning that will optimize your EOT and SVG files even further. EOTFAST is free desktop software (download at http://eotfast.com) that converts TTF files into compressed but lossless EOT files; the EOT files that Font Squirrel produces are not compressed. The command-line tool ttf2svg (http://xmlgraphics.apache.org/batik/tools/font-converter.html) converts TTF files into same size or smaller SVG files; you need to have Java and the Java SVG toolkit Batik installed on your system to run it.

Using `@font-face`

Let's finally put `@font-face` to use in our page. Since it looks like notebook paper, a font that simulates handwriting seems appropriate. I picked Prelude, a casual cursive font, for the headings (**Figure 3.28**). We're not going to apply a casual cursive font to the body copy, however, as that kind of font at small sizes doesn't look very good and decreases legibility.


PRELUDE

4 Styles | TrueType | More Casady & Greene Fonts | Visit Site

@font-face Compatible!

PreludeFLF Regular | England Trip 2007 | 60

England Trip 2007

Download Prelude in TrueType format:

Prelude Family (All 4 styles)

PreludeFLF Regular | 232 Glyphs

PreludeFLF Medium Italic | 232 Glyphs

PreludeFLF Bold | 232 Glyphs

PreludeFLF Bold Italic | 232 Glyphs

FIGURE 3.28
The Prelude font on the Font Squirrel site

In the exercise files for this chapter, you'll find a folder named "fonts" that contains all the eight versions of Prelude that we'll need for our page: EOT, SVG, TTF, and WOFF files for both the regular and bold weight of the font. I created these versions using Font Squirrel's Generator tool, using the settings shown in Figure 3.27. I then remade the EOT files using EOTFAST to cut the file size of each EOT roughly in half.

LINKING TO THE FONTS WITH THE @font-face RULES

You may notice in Figure 3.27 that there are three choices in the Font Squirrel Generator for CSS Formats. These refer to three variations of the `@font-face` syntax used in the CSS. As with almost everything in CSS, there are multiple ways to code `@font-face` to get the same effect; all three syntaxes use valid, standards-compliant CSS and will work in the same browsers.

NOTE: If you want the details, click the blue links under each CSS format name in the Font Squirrel Generator to read the three articles explaining all the whys and hows.

The rationale behind each of these three syntaxes is too complicated to fully explain here, and not terribly important. Any of the three will work for our purposes, and the choice really boils down to personal preference. My preference is the "Bulletproof Smiley" version.

Here's what the Bulletproof Smiley syntax for the Prelude font looks like:

```
@font-face {
   font-family: 'Prelude';
   src: url('fonts/preludeflf-webfont.eot');
   src: local('☺'),
      url('fonts/preludeflf-webfont.woff') format('woff'),
      url('fonts/preludeflf-webfont.ttf')
      ¬format('truetype'),
      url('fonts/preludeflf-webfont.svg#webfont')
      ¬format('svg');
}
@font-face {
   font-family: 'Prelude';
   src: url('fonts/preludeflf-bold-webfont.eot');
   src: local('☺'),
      url('fonts/preludeflf-bold-webfont.woff')
      ¬format('woff'),
      url('fonts/preludeflf-bold-webfont.ttf')
      ¬format('truetype'),
      url('fonts/preludeflf-bold-webfont.svg#webfont')
      ¬format('svg');
   font-weight: bold;
}
```

Put this Bulletproof Smiley syntax before any of the other CSS rules; it will work anywhere you put it, but you'll learn later in the chapter how putting it at the top of your styles can improve your page's performance." You can copy and paste it from paper_final.html in the exercise files.

These two `@font-face` rules group the regular and bold font faces into a single font family by declaring them with the same `font-family` name, Prelude. Each `@font-face` rule gives the path to the font files and, optionally, the style characteristics of an individual face (such as `font-weight: bold` or `font-style: italic`).

NOTE: There are more nitty-gritty details about how this syntax works in Paul Irish's original article at http://paulirish.com/2009/bulletproof-font-face-implementation-syntax. They're not essential to know, but are interesting if you're a web geek like me.

Let's look at just the first `@font-face` rule for now and go through it line by line.

The first part of the rule—`font-family: 'Prelude';`—assigns a name to the font you're linking to so that you can later refer to this font in your font stacks. You can make the name whatever you want; it's just a shorthand way of referring to a whole bunch of font information at once.

The second part of the rule—`src: url('fonts/preludeflf-webfont.eot');`—gives the path to the EOT version of the font for IE 8 and earlier. This is separated out from the other versions of the fonts because IE can't understand a `src` descriptor with multiple comma-separated values. It thinks it's one big path, preventing it from noticing the EOT and being able to use it when grouped with the other files.

The next part of the rule is a second `src` value that lists all the font files for non-IE browsers. Each browser will go through the list until it finds a format it can use, and then download that file, and only that file, to display the text. Each font includes a path to the font file, such as `url('fonts/preludeflf-webfont.woff')`, and a format hint, such as `format('woff')`. The format hint is optional, but including it alerts the browsers about the format of each font to keep them from downloading ones they can't use, which would waste bandwidth and slow page loading.

NOTE: The WOFF format is listed first because it's the standard that we want all browsers to use when they can. It's also the smallest file, so you want to make sure browsers that can use it see it first and therefore do use it.

HITTING THE SERVER

All browsers but IE8 and earlier don't actually download any font files until one is called for in a font stack elsewhere in the CSS. So you can declare lots of `@font-face` rules in your style sheet, but if one particular page doesn't have elements that use most of those fonts, for instance, you won't incur the hit of a bunch of extra HTTP requests.

IE8 and earlier, on the other hand, download every EOT file as soon as they encounter it. While you're testing font embedding, it's common to include a lot of extra `@font-face` rules in your style sheet so you can compare fonts. Be sure to remove any `@font-face` rules that you don't end up using so IE 8 and earlier don't download the EOT files unnecessarily.

But you probably noticed that at the start of the second `src` value is `local('☺')`. What in the world does this smiley face do?

The `local('☺')` part of the `src` value is there to protect IE. Without it there, IE would try to read the second `src` descriptor as one big path, as explained earlier, which would lead it to get a 404 error. While this doesn't stop `@font-face` from working—IE can still use the separate EOT—it's an extra, pointless hit on your server that you don't want. IE doesn't understand the `local()` syntax, and putting it at the start of the `src` value stops it from moving any further into the `src` value, seeing the `url()` value, and then trying to parse the path.

PROBLEMS WITH `local()`

Letting users skip downloading a font they already have installed sounds like such a good and helpful idea—so why not put the real font name in `local()` instead of a smiley face character? This is certainly an option. It's what Paul Irish's original "Bulletproof @font-face syntax" did, and you can still choose to download this syntax from the Font Squirrel Generator.

But before you use the real font name in `local()`, you should be aware of a few problems you might run into:

- Different fonts sometimes have the same names. It's possible that the user will end up seeing a completely different font from the one you intend. (See http://typophile.com/node/63992 for a discussion of this.) It's a very small chance, but some argue that, regardless, giving control over type to the user's machine and browser is not wise.
- In Chrome, all characters might be displayed as As in boxes if the local font that you're referring to was installed on the user's system using the font management software FontExplorer X. (Go to http://snook.ca/archives/html_and_css/font-face-in-chrome to see a screenshot of this weirdness.)
- In Safari, the user might get a dialog box asking permission to use the local font if it's being managed by FontExplorer X.

None of these problems are likely to happen very often, but if they do happen, the effect could be pretty bad. Many web font experts recommend never using `local()`, or using it only when the font file you're trying to keep the user from downloading is particularly large.

The local() syntax is perfectly valid CSS, by the way. Its real purpose in a @font-face rule is to point to a locally installed version of the font on the user's machine, so that if the user has the same font as you're embedding, he doesn't have to download the extra file. That's why Paul Irish, who came up with the syntax, recommends using a smiley face: we don't want to call for a font that might actually exist, and it's very unlikely that anyone will ever release a font named ☺.

The second @font-face rule declares the bold versions of the Prelude font family. It gives the paths to all the bold font files and also sets the font-weight to bold inside the rule. But the font-family name is Prelude (not PreludeBold or some other variation), matching the first @font-face rule. Assigning the same name tells the browser that the file is the bold version of the same Prelude font family. Now, any time the browser needs to have bold Prelude text (because of a strong element in the HTML or font-weight: bold in the CSS), it doesn't have to synthesize the boldness by making the characters thicker, but can instead use the true bold font files. Using a true bold or italic font face looks better than having the browser simulate it for you.

NOTE: IE doesn't always handle this font style switching within the @font-face rules correctly. IE 8 and earlier don't use the font when font-style: italic is in the @font-face rule. IE 9 does, but it synthesizes italic rendering anyway, even if the font you're calling isn't actually italic.

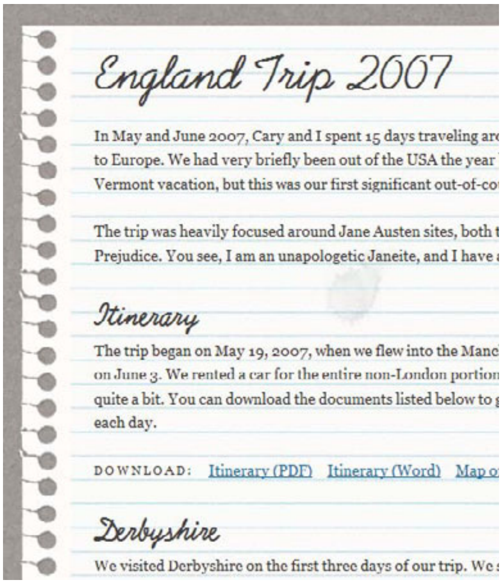
DECLARING THE FONT

Adding @font-face rules to your CSS doesn't actually make the fonts show up anywhere; it simply links them, so they're ready to be downloaded and used when you need them. Let's call them up in our h1 and h2 elements. Add Prelude, the name of the font we assigned in the @font-face rule, to the start of the existing font-family values in the h1 and h2 rules:

```
h1 {
   margin: -.3em 0 .14em 0;
   color: #414141;
   font-family: Prelude, Helvetica, "Helvetica Neue",
                Arial, sans-serif;
   font-size: 3.5em;
   font-weight: normal;
}
h2 {
   clear: left;
   margin: 0 0 -.14em 0;
   color: #414141;
   font-family: Prelude, Helvetica, "Helvetica Neue",
                Arial, sans-serif;
   font-size: 2.17em;
   font-weight: bold;
}
```

The sans-serif fallback fonts in the font stacks don't look anything like the cursive Prelude script, of course. I chose to do this because there aren't really any cursive web-safe fonts we can rely on as fallbacks. If someone is using a browser that can't do font embedding, I'd rather they see some nice, clean Helvetica or Arial text than whatever random cursive font they might have on their computers.

Note that the `h1` rule sets the `font-weight` to `normal` and the `h2` rule sets it to `bold`. This tells the browser to use the regular member of the Prelude font family (the first `@font-face` rule) for the `h1` elements and the bold member of the Prelude font family (the second `@font-face` rule) for the `h2` elements (**Figure 3.29**).



England Trip 2007

In May and June 2007, Cary and I spent 15 days traveling ar
to Europe. We had very briefly been out of the USA the year
Vermont vacation, but this was our first significant out-of-co

The trip was heavily focused around Jane Austen sites, both t
Prejudice. You see, I am an unapologetic Janeite, and I have

Itinerary

The trip began on May 19, 2007, when we flew into the Manc
on June 3. We rented a car for the entire non-London portion
quite a bit. You can download the documents listed below to g
each day.

DOWNLOAD: Itinerary (PDF) Itinerary (Word) Map o

Derbyshire

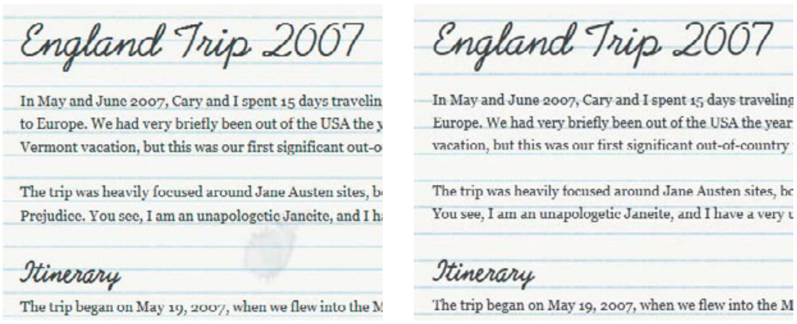
We visited Derbyshire on the first three days of our trip. We

FIGURE 3.29
The cursive Prelude font in the headings on our page

We now have handwritten cursive text showing in our headings that is resizable, selectable, and indexable. There are differences in the anti-aliasing and hinting of the text between browsers and between Windows and Mac, but the advantages of real text outweigh the inconvenience of its slight jaggedness in some browsers (**Figure 3.30**).

TABLE 3.6 `@font-face` **browser support**

IE	FIREFOX	OPERA	SAFARI	CHROME
Yes	Yes, 3.5+	Yes, 10+	Yes	Yes



England Trip 2007

In May and June 2007, Cary and I spent 15 days travelin
to Europe. We had very briefly been out of the USA the y
Vermont vacation, but this was our first significant out-o

The trip was heavily focused around Jane Austen sites, b
Prejudice. You see, I am an unapologetic Janeite, and I h

Itinerary

The trip began on May 19, 2007, when we flew into the M

England Trip 2007

In May and June 2007, Cary and I spent 15 days traveling
Europe. We had very briefly been out of the USA the year
vacation, but this was our first significant out-of-country

The trip was heavily focused around Jane Austen sites, bc
You see, I am an unapologetic Janeite, and I have a very u

Itinerary

The trip began on May 19, 2007, when we flew into the M

FIGURE 3.30 Different platforms and browsers, such as Firefox 3.6 (left) and IE 9 (right), display the anti-aliasing of the headings differently.

THE LOWDOWN ON THE `@font-face` RULE

The `@font-face` rule is part of the Fonts module, found at www.w3.org/TR/css3-fonts.

A `@font-face` rule gives a font family name (using the `font-family` descriptor) that you make up and the path to one or more font files (using the `src` descriptor). Optionally, it can also include the style characteristics of an individual face (using `font-weight`, `font-style`, and `font-stretch`). You can use multiple `@font-face` rules with the same `font-family` name to group faces into one family.

To access the fonts in your `@font-face` rules, simply add each font family name to your font stacks in the `font-family` property.

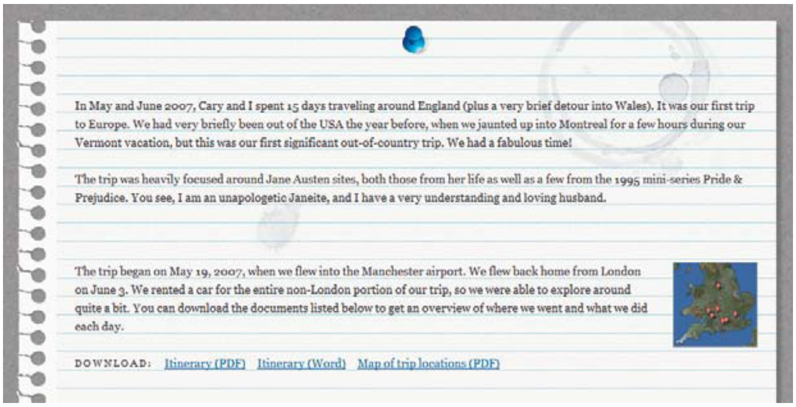
Other than making text look like handwriting, you might want to use `@font-face` for:

- Creating a look and feel not possible with standard web-safe fonts
- Keeping branding consistent between printed materials (such as a logo or brochure) and their related web site
- Displaying text using non-Latin characters, which often don't render well in browser default fonts. Using a font designed for the language ensures all the characters display correctly.

A tempting use of `@font-face` is to use dingbat fonts to create icons without images. But this has serious accessibility problems. See http://filamentgroup.com/lab/dingbat_webfonts_accessibility_issues and http://jontangerine.com/log/2010/08/web-fonts-dingbats-icons-and-unicode for a discussion of the problems and potential solutions.

IMPROVING PERFORMANCE

If you view your page in a browser now, you may notice a lag between when most of the page loads and when the handwritten font displays. Webkit-based browsers don't show the `@font-face`-styled text until they've finished downloading the font file (**Figure 3.31**).



In May and June 2007, Cary and I spent 15 days traveling around England (plus a very brief detour into Wales). It was our first trip to Europe. We had very briefly been out of the USA the year before, when we jaunted up into Montreal for a few hours during our Vermont vacation, but this was our first significant out-of-country trip. We had a fabulous time!

The trip was heavily focused around Jane Austen sites, both those from her life as well as a few from the 1995 mini-series Pride & Prejudice. You see, I am an unapologetic Janeite, and I have a very understanding and loving husband.

The trip began on May 19, 2007, when we flew into the Manchester airport. We flew back home from London on June 3. We rented a car for the entire non-London portion of our trip, so we were able to explore around quite a bit. You can download the documents listed below to get an overview of where we went and what we did each day.

DOWNLOAD: Itinerary (PDF) Itinerary (Word) Map of trip locations (PDF)

FIGURE 3.31
The headings are invisible while Safari or Chrome downloads the font files it needs.

In Firefox and Opera, the fallback fonts show for a moment while the font file is downloaded, and then the browser re-renders the text with the new font. This is called the Flash of Unstyled Text, or FOUT, a term quippishly coined by Paul Irish.

TIP: The `font-size-adjust` property, mentioned earlier, doesn't lessen the FOUT, but it makes it less noticeable because it makes the size of the fallback font match up better with the web font. Again, though, it works only in Firefox.

These font-loading lags are usually a minor annoyance, but in some cases they can be quite noticeable and problematic. Fonts for non-Western languages, such as Chinese and Japanese, can contain thousands of characters and be several megabytes in size; these huge font files take a long time to download, of course. Also, users on mobile devices in areas with poor coverage, or at hotels with notoriously slow connection speeds, may be left waiting for the web fonts to appear for quite a while.

There are a number of things you can do to minimize or do away with the FOUT or the invisible text problem in Webkit:

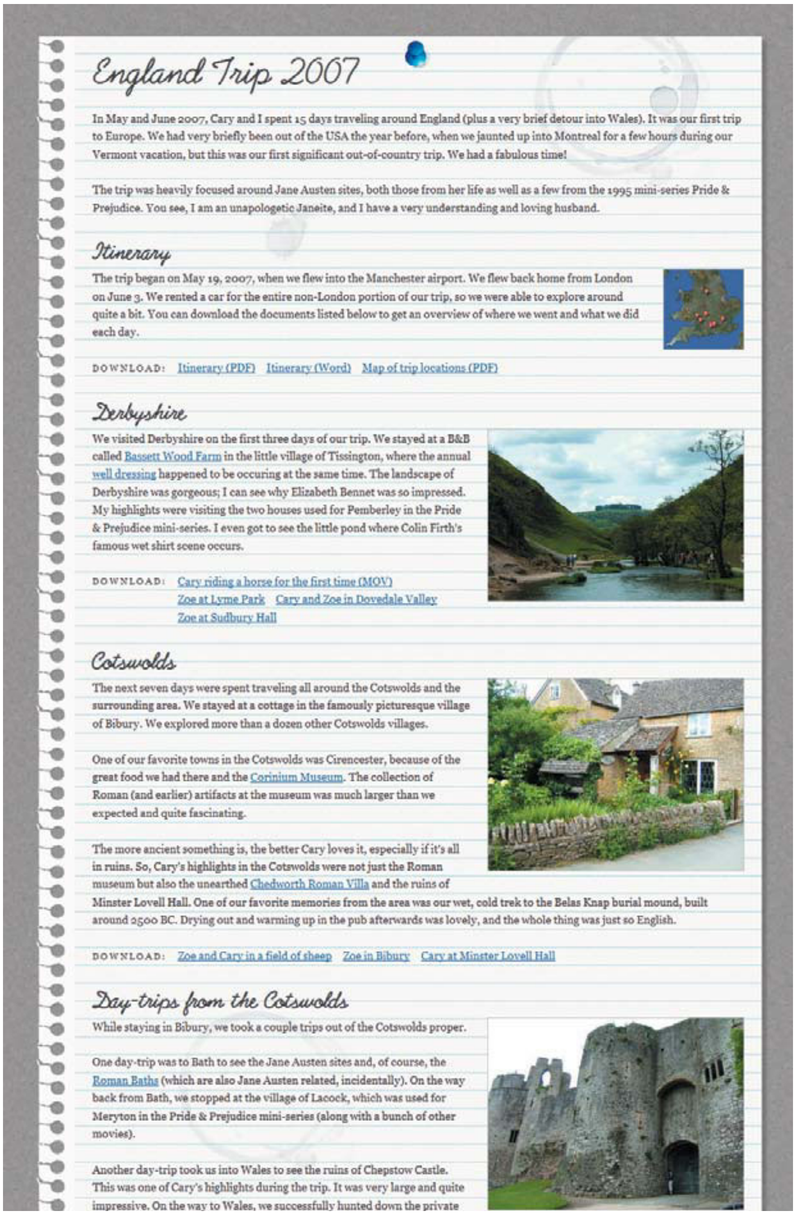
- Keep your font file sizes as small as possible to begin with. Subsetting the characters within each font to include only the characters that you need can really help in this regard; the Font Squirrel Generator lets you do this.
- Put your `@font-face` rules at the top of your style sheets. This increases the chance that the browser will download them before the other files called for in your CSS, such as background images.

- Get the browser to download the font file as soon as possible by, for instance, calling it on a hidden element at the very start of your page. You can adapt many image preloading techniques, such as the many listed at http://perishablepress.com/press/tag/preload-images, to work with font files.
- Host your fonts elsewhere. By serving your fonts from one common location, you increase the chance that the visitor already has the font file in his or her cache, instead of having to download the same exact font file again from a new location. The font-embedding services listed earlier allow you to do this, as does Google's Font Directory, but you can also upload fonts you personally own to the TypeFront service (http://typefront.com). TypeFront hosts the fonts you give it, converts them to all the needed formats, and serves them only to the sites you specify.
- Set the Expires header in .htaccess to a date far in the future so that when a font is downloaded once, it's cached by the browser and not requested again for a very long time. This doesn't help with the initial page load when the browser first downloads the font, but it should help on subsequent loads. (See "HTTP Caching" by Steve Lamm at http://code.google.com/speed/articles/caching.html for more information.)
- Gzip your font files. Stoyan Stefanov found average file-size savings to be from 40 to 45 percent (see www.phpied.com/gzip-your-font-face-files). But he also found that this doesn't really help WOFF files, which are already very compressed, so this may not help you much with the FOUT in Firefox (see www.phpied.com/font-face-gzipping-take-ii). However, gzipping should help Opera avoid or minimize the FOUT and Safari and Chrome show the text sooner.
- Use scripting to hide all the content for a couple seconds while the browser downloads the fonts. This doesn't actually speed up downloading the fonts, of course, but it keeps the user from ever seeing the FOUT's disorienting shift in fonts. Paul Irish provides two different JavaScript options to do this, one of which uses Google's WebFont Loader JavaScript library (http://paulirish.com/2009/fighting-the-font-face-fout).

Our font files already use a subset of all characters and are called at the top of the CSS, so we've covered the most basic `@font-face` performance best practices. It's beyond the scope of this book to add any of the scripting or server-side techniques to our page, but this gives you a number of things to try if you're having trouble with web font loading times.

The Finished Page

We've completed all the styling for the article to make it look like a piece of notebook paper. In any up-to-date, non-IE browser, you should see something like **Figure 3.32**. Compare it to Figure 3.1.



England Trip 2007

In May and June 2007, Cary and I spent 15 days traveling around England (plus a very brief detour into Wales). It was our first trip to Europe. We had very briefly been out of the USA the year before, when we jaunted up into Montreal for a few hours during our Vermont vacation, but this was our first significant out-of-country trip. We had a fabulous time!

The trip was heavily focused around Jane Austen sites, both those from her life as well as a few from the 1995 mini-series Pride & Prejudice. You see, I am an unapologetic Janeite, and I have a very understanding and loving husband.

Itinerary

The trip began on May 19, 2007, when we flew into the Manchester airport. We flew back home from London on June 3. We rented a car for the entire non-London portion of our trip, so we were able to explore around quite a bit. You can download the documents listed below to get an overview of where we went and what we did each day.

DOWNLOAD: Itinerary (PDF) Itinerary (Word) Map of trip locations (PDF)

Derbyshire

We visited Derbyshire on the first three days of our trip. We stayed at a B&B called Bassett Wood Farm in the little village of Tissington, where the annual well dressing happened to be occuring at the same time. The landscape of Derbyshire was gorgeous; I can see why Elizabeth Bennet was so impressed. My highlights were visiting the two houses used for Pemberley in the Pride & Prejudice mini-series. I even got to see the little pond where Colin Firth's famous wet shirt scene occurs.

DOWNLOAD: Cary riding a horse for the first time (MOV) Zoe at Lyme Park Cary and Zoe in Dovedale Valley Zoe at Sudbury Hall

Cotswolds

The next seven days were spent traveling all around the Cotswolds and the surrounding area. We stayed at a cottage in the famously picturesque village of Bibury. We explored more than a dozen other Cotswolds villages.

One of our favorite towns in the Cotswolds was Cirencester, because of the great food we had there and the Corinium Museum. The collection of Roman (and earlier) artifacts at the museum was much larger than we expected and quite fascinating.

The more ancient something is, the better Cary loves it, especially if it's all in ruins. So, Cary's highlights in the Cotswolds were not just the Roman museum but also the unearthed Chedworth Roman Villa and the ruins of Minster Lovell Hall. One of our favorite memories from the area was our wet, cold trek to the Belas Knap burial mound, built around 2500 BC. Drying out and warming up in the pub afterwards was lovely, and the whole thing was just so English.

DOWNLOAD: Zoe and Cary in a field of sheep Zoe in Bibury Cary at Minster Lovell Hall

Day-trips from the Cotswolds

While staying in Bibury, we took a couple trips out of the Cotswolds proper.

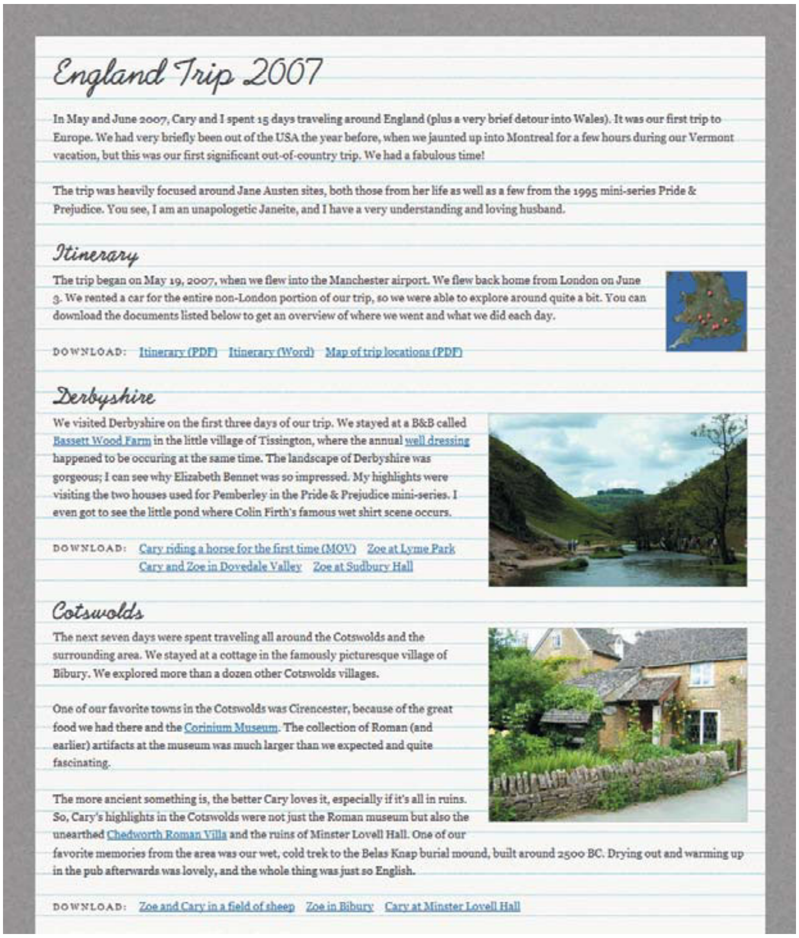
One day-trip was to Bath to see the Jane Austen sites and, of course, the Roman Baths (which are also Jane Austen related, incidentally). On the way back from Bath, we stopped at the village of Lacock, which was used for Meryton in the Pride & Prejudice mini-series (along with a bunch of other movies).

Another day-trip took us into Wales to see the ruins of Chepstow Castle. This was one of Cary's highlights during the trip. It was very large and quite impressive. On the way to Wales, we successfully hunted down the private

FIGURE 3.32 The page with all CSS3 applied, shown here in Firefox 3.6.

The preview version of IE 9 doesn't show the torn paper edge, but otherwise looks like Figure 3.32. IE 8 and earlier are missing most of the graphic effects, but since they show the lined background image and the handwritten font, the overall appearance is still attractive and notebook-paper-like (**Figure 3.33**). Also, in this case, all versions of IE up to 8 display almost identically—even 5.5 looks like the screenshot shown in Figure 3.33.

NOTE: The completed page showing all of these effects is named paper_final.html in the exercise files for this chapter.



England Trip 2007

In May and June 2007, Cary and I spent 15 days traveling around England (plus a very brief detour into Wales). It was our first trip to Europe. We had very briefly been out of the USA the year before, when we jaunted up into Montreal for a few hours during our Vermont vacation, but this was our first significant out-of-country trip. We had a fabulous time!

The trip was heavily focused around Jane Austen sites, both those from her life as well as a few from the 1995 mini-series Pride & Prejudice. You see, I am an unapologetic Janeite, and I have a very understanding and loving husband.

Itinerary

The trip began on May 19, 2007, when we flew into the Manchester airport. We flew back home from London on June 3. We rented a car for the entire non-London portion of our trip, so we were able to explore around quite a bit. You can download the documents listed below to get an overview of where we went and what we did each day.

DOWNLOAD: Itinerary (PDF) Itinerary (Word) Map of trip locations (PDF)

Derbyshire

We visited Derbyshire on the first three days of our trip. We stayed at a B&B called Bassett Wood Farm in the little village of Tissington, where the annual well dressing happened to be occuring at the same time. The landscape of Derbyshire was gorgeous; I can see why Elizabeth Bennet was so impressed. My highlights were visiting the two houses used for Pemberley in the Pride & Prejudice mini-series. I even got to see the little pond where Colin Firth's famous wet shirt scene occurs.

DOWNLOAD: Cary riding a horse for the first time (MOV) Zoe at Lyme Park Cary and Zoe in Dovedale Valley Zoe at Sudbury Hall

Cotswolds

The next seven days were spent traveling all around the Cotswolds and the surrounding area. We stayed at a cottage in the famously picturesque village of Bibury. We explored more than a dozen other Cotswolds villages.

One of our favorite towns in the Cotswolds was Cirencester, because of the great food we had there and the Corinium Museum. The collection of Roman (and earlier) artifacts at the museum was much larger than we expected and quite fascinating.

The more ancient something is, the better Cary loves it, especially if it's all in ruins. So, Cary's highlights in the Cotswolds were not just the Roman museum but also the unearthed Chedworth Roman Villa and the ruins of Minster Lovell Hall. One of our favorite memories from the area was our wet, cold trek to the Belas Knap burial mound, built around 2500 BC. Drying out and warming up in the pub afterwards was lovely, and the whole thing was just so English.

DOWNLOAD: Zoe and Cary in a field of sheep Zoe in Bibury Cary at Minster Lovell Hall

FIGURE 3.33
IE (version 8 shown here) doesn't show all the CSS3 graphic effects, but does show the handwritten fonts.

This page intentionally left blank

This page intentionally left blank

Index

:: (double colons), 57
# (pound sign), 181
: (single colon), 57
&nbsp, 57
* attribute selector, 164
2D transforms, 35, 90, 91
3D buttons, 84
3D cubes, 91
3D transforms, 90, 91
37signals, 191

A

absolute positioning, 60
accessibility, 19, 92, 143
access keys, 163
Acrobat Reader, 153
::after pseudo-element, 57, 169
alistapart.com articles
  font embedding, 128, 131
  mobile style sheets, 240
  Modernizr, 34
  prefix or posthack, 26, 27
  supersizing background images, 108
  zebra striping, 172
Alpha filter, 35
alpha transparency, 62
alpha-transparent PNGs, 62, 69–70, 79
alt attribute, 151
alternating colors, 171–176
alternative text, 163
ancestor elements, 169
Android, 25, 226, 235, 236
angled elements, 91, 92
animatable properties, 196
animated GIFs, 204
animation property, 200, 202
animations, 3, 96, 191–204
anti-aliasing, 132, 133, 143
Ascender, 131
Ateş, Faruk, 34
attribute presence and value selectors, 152, 162
attribute selectors
  adding file-type indicators with, 153–156
  assigning classes to, 160, 161
  browser support, 163
  combining multiple, 157
  combining with other selectors, 151
  exercise files, 153, 155, 158, 165
  finished page, 165–166
  for “not matching”, 153
  purpose of, 151
  quotation marks in, 154
  styling photos with, 160–165
  targeting image types with, 164–165
  types of, 152, 163
  values in, 162
  ways of using, 162–163
  workarounds for IE 6, 157–158, 161
avatars, 89, 92–96

B

Babé, Louis-Rémi, 125
background
  changing color of, 188–191
  fading color in/out, 191–204
  images (*See* background images)
  semitransparent, 31, 62–72
background-clip property, 35, 120–125
background images
  applying multiple, 109–114
  filling entire page with, 108
  scaling, 101–107, 108
  for spiral-notebook effect, 114–125
  tiling, 62, 103, 108
background-origin property, 35
background-position property, 21
background-repeat property, 103, 115
Backgrounds and Borders module, 53, 84, 108, 112, 117, 122
background shorthand property, 69, 110, 111, 121, 196
background-size property, 101–108
  browser support, 104, 108, 111
  how it works, 101–103
  purpose of, 101
  vs. background-repeat property, 103
  ways of using, 108, 156
  workarounds for IE, 107
backslash, 58
Bakaus, Paul, 96
bakery site
  base page for, 206–207
  finished page for, 241–242
  large-screen layout, 209–219
  mobile-device layout, 226–240
  small-screen layout, 220–225
bar charts, 53
Basecamp, 191
BasicImage filter, 35
Basic User Interface module, 283
::before pseudo-element, 57, 59, 169
Bester, Michael, 71
Bil’ak, Peter, 133
blocks, aligning, 3
blog comments, 47, 48–49
  See also speech bubbles
Blur filter, 35, 85
bold font, 141
border-box value, 122, 123, 280, 283
border-image generator, 119
borderImage plugin, 125
border-image property, 115–120
border images, 115, 116, 121, 125
border-radius property, 22, 23, 35, 52–54, 89, 186
borders
  creating triangles out of, 55–56
  flowing across columns, 217
  simulating, 84
bosses, dealing with unsupportive, 41–45
box-align property, 258–260, 261, 263, 277
box-direction property, 255–258, 277
boxes
  angled, 91, 92
  disappearing, 264

boxes *(continued)*
  flexible, 112, 246
    See also flexible box layout model
  skewed, 91
  stretching, 283
  transforming, 2
    See also transforms
  wrapping children of, 265
box-flex-group property, 277
box-flex property, 248–252, 272–275, 277
box-lines property, 265, 277
box model, 244, 277
box-ordinal-group property, 256–257, 277
box-orient property, 277
box-pack property, 261, 265, 266, 277
box-shadow generator, 83
box-shadow property
  adding drop shadow with, 81–85
  benefits of using, 16
  browser support, 84, 164
  declaring color in, 82
  emulating, 35
  ways of using, 84
  workarounds for IE, 85
box-sizing property, 9, 280, 282–283
break-word value, 49
Brown, Nora, 119
browser-based color pickers, 64
browser prefixes, 25–30
browsers. *See also* specific browsers
  and animations, 192, 202
  and attribute selectors, 163
  and background-clip property, 122, 123, 124–125
  and background-repeat property, 115
  and background-size property, 104, 108, 111
  and border-image property, 118
  and border-radius property, 54
  and box-shadow property, 84
  and box-sizing property, 282
  and calc function, 281
  and CSS3, 8–11
  designing in, 43–45
  and flexible box layout model, 276
  and @font-face rules, 9, 135, 139, 142
  and font rendering, 133
  and generated content, 187
  and gradients, 77
  market share, 7–8
  and media queries, 212
  mobile, 236
  and multi-column layouts, 218, 219
  and multiple background images, 113
  and :nth-child( ) pseudo-class, 174
  and :nth-of-type( ) pseudo-class, 179
  providing workarounds for older, 11–14, 24, 30–41
  and RGBA/HSLA, 67
  and :target pseudo-class, 190, 191
  and text-shadow property, 88
  and transforms, 91
  and transitions, 192, 198
  and word-wrap property, 49–50
browser sniffing, 33
bulleted lists, 163
Bulletproof Smiley syntax, 138–141
buttons, 53, 84, 113, 192

C

calc function, 281
Camino, 25
Campaign Monitor, 10
Candidate Recommendation status, 5, 25
canvas element, 114
cards, folding, 91
Casciano, Chris, 85
centering, 261–266, 268
Chao, Ingo, 71
child elements, 169
Chrome. *See also* browsers
  and anti-aliasing, 92
  and background-size property, 104, 111
  and border-image property, 120
  and box-shadow property, 164
  browser prefix, 25
  CSS3 support, 8–9
  and drop shadows, 126
  and float property, 264
  and @font-face rules, 135
  and local( ) syntax, 140
  market share, 7
  and round value, 126
  and spread radius, 164
circles, 52
circular badges, 53
citations, 163
Clark, Keith, 34, 159
Clarke, Andy, 44, 218
classes, 160–161
ClearType, 132
clients, dealing with unsupportive, 41–45
CMS, 154, 160
Code Converter, Unicode, 57
colons, single *vs.* double, 57
color converter tools, 64
Color module, 67
color picker tools, 63–64
colors, alternating, 171–176
Colors app, 63
column-count property, 214–215, 219
column-gap property, 215, 219
Columnizer plugin, 218
columns. *See also* multi-column layouts
  adding, 253–255
  breaking text into, 214–215
  creating equal-height, 258–260
  problems with, 217–218
  reordering, 255–258
Column script, 218
column-width property, 214, 219
comments, 30. *See also* blog comments; conditional comments
comps, 43–45

conditional comments, 36–41, 55
content
  breaking across columns, 217
  CSS-generated, 56–57, 61, 156, 184
  overflowing, 217
content-box model, 277
`content-box` value, 122, 280, 283
content management system, 154. *See also* CMS
Converter, Unicode Code, 57
counters, 184–186
Coyier, Chris, 185
CSS. *See also* specific versions
  browser prefixes, 25–30
  properties. *See* properties
  pseudo-elements, 169
  putting comments in, 30
  sprites. 21, 22, 185
  transforming objects with. 89
  vs. other technologies. 17
CSS 2, 127
CSS 2.1
  attribute selectors, 152, 162
  box model, 244, 277
  CSS3 as extension of, 2
  and `@font-face` rules, 127
  and generated content, 56–57
  layout tools, 243
CSS3
  attribute selectors, 152, 162
  benefits of, 15–19
  best practices, 25–41
  browser support, 7, 8–11, 30
  case study, 19–24
  creating speech bubbles with, 47–48
  declaring color in, 31
  detecting support for, 32–34
  emulating, 34–35
  and `@font-face` rules, 127
  gradients. *See* gradients
  layout tools, 243, 277, 284–285 *See also* layouts
  managing expectations about, 43–45
  as marketable career skill, 19
  maturity levels, 4–6
  modules, 2, 4
  new features, 2–3
  and non-supporting browsers, 30–41
  and readability, 19
  recommendations on using, 5–6, 25–41
  RGBA/HSLA syntax, 62–64
  strategies for gaining acceptance of, 39–45
  structural pseudo-classes, 169–171
  styling images/links with, 149–166
  transforms. *See* transforms
CSS-generated content, 56–57, 61
CSS Infos, 10
cssSandpaper script, 34, 96
Cuadra, Jimmy, 192
cursive font, 137
"curved corner" script, 54

D

"DD_roundies" script, 55
Decker, Kevin, 119
degradation, graceful, 12, 15
descendants, 169
Designing with Progressive Enhancement, 11, 15
design mockups, 43–45
Deveria, Alexis, 285
device width, 227
dialog boxes, 53
digital rights management, 134
Diller, Drew, 55
dingbat fonts, 143
DirectWrite, 133
disappearing boxes, 264
`display` property, 246, 284
document object model, 170
document tree, 169–170
DOM, 170
DOMAssistant, 176
Dorward, David, 191
double colons, 57
Doughnut Color Picker, 64
DRM, 134
DropShadow filter, 35, 85, 89
drop shadows, 81–85, 125–126
dynamic highlighting, 180–181

E

eCSStender, 27
Edwards, Dean, 34, 157, 176, 284
elliptical corners, 52
email
  links, 163
  newsletters, 10
  sign-up form, 268–272
`:empty` pseudo-class, 170
ems, 104, 105
Enders, Jessica, 172
end-user licensing agreements, 128–129
EOTFAST, 136, 137
EOT font files, 134, 135, 136
equal-height columns, 258–260
escape character, 58
EULAs, 128–129
exercise files
  attribute selectors, 153, 155, 158, 165
  `box-sizing` property, 283
  flexible box layout model, 245, 265, 283
  media queries, 209, 224, 241
  notebook paper, 104
  speech bubbles, 49, 61, 70, 81, 87, 96
  sticky footers, 275
  `:target` pseudo-class, 189
  zebra striping, 172
exljbris, 129–130
Expires header, 145
"explorer-canvas" script, 114

F

fade-out animation, 198–204
Fadeyev, Dmitry, 133
famfamfam icons, 155
faux columns. *See* equal-height columns
Fetchak, Nick, 55
FHOS, 130
file folders, 100
file-type indicators, 154
file types, styling links for specific, 153–159
`fill` keyword, 116
`filter` property, 35, 94
filters IE, 35–36
FindMeByIP, 10
Fink, Richard, 128, 133

Firefox. *See also* browsers
and anti-aliasing, 143
and `background-clip` property, 122
and `background-size` property, 104, 111
and `border-radius` property, 89
browser prefix, 25
and `calc` function, 281
CSS3 support, 9
and `@font-face` rules, 135, 144
and gradients, 73
market share, 7
and multiple background images, 113–114
Rainbow extension, 64
and transitions, 192
and W3C syntax, 73–74
Web Developer extension, 238
`:first-child` pseudo-class, 169, 170, 171
`::first-letter` pseudo-element, 169
`::first-line` pseudo-element, 169
`:first-of-type` pseudo-class 170
Fisher, Meagan, 44
Flash, 194
Flash of Unstyled Text, 144
flexible boxes, 112, 246
flexible box layout model
adding columns with, 253–255
alternatives to, 277–285
browser support, 247, 266–268, 276
creating equal-height columns with, 258–260
creating layouts with, 245–266
disadvantages of, 246
exercise files, 245, 257, 265
laying out forms with, 268–272
properties, 277
purpose of, 244, 276
reordering columns with, 255–258
and sticky footers, 272–275
vertical/horizontal centering with, 261–266
ways of using, 268, 276
and wrapping of box's children, 265
Flexible Box Layout module, 3, 276
Flexible Web Design, 207, 214
floats, 217, 229, 245, 264, 265
Flock, 25
fluid layouts, 206
folding cards, 91
Fontdeck, 131
Font Directory, Google, 129, 145
font embedding, 127, 129
font-embedding services, 130–131, 145
FontExplorer X, 140
Fontfabric, 129–130
@font-face Kit Generator, 135–136, 137
`@font-face` rules
accessing fonts in, 143
adding to CSS, 141
browser support, 9, 134–135, 142
and font licenses, 128–129
linking to fonts with, 138–141
performance issues, 17–18, 144–145
purpose of, 3, 127, 143
syntax variations, 138
ways of using, 143
and Webkit-based browsers, 144
`font-family` descriptor, 127, 141, 143
font-generator tool, 135–136, 137, 144
Fonthead, 130
font hinting, 132, 133
font hosting, 130, 145
font-loading lags, 144
Font module, 143
font-replacement techniques, 127, 129
fonts. *See also* text
and anti-aliasing, 132, 133, 142
choosing acceptable, 128–134
compressing, 145
converting, 135–136
declaring, 141–145
embedding, 127
gzipping, 145
hosting of, 130, 145
licensing issues, 128–129
linking to, 127, 134, 138–141
for non-Western languages, 144
performance issues, 144–145
readability issues, 132–134
rendering issues, 132, 133
security issues, 128
sources of free, 129–130
testing, 132
web, 127, 128, 132
web-safe, 127
Fonts.com Web Fonts, 131
FontShop, 130
`font-size-adjust` property, 132
FontsLive, 131
Fonts module, 3
font smoothing, 133
`font-smoothing` property, 133
FontSpring, 130
Font Squirrel, 129, 135–136, 137, 144
font stacks, 127
footers, sticky, 272–275
form fields, 162
form layouts, 268–272
FOUT, 144, 145
"Fragment Highlight" script, 191
fragment identifiers, 181–182
Franquet, Alix, 75

G

Gallagher, Nicolas, 114
Gasston, Peter, 75, 79
generated content, 56–57, 61, 156, 184
Giannattasio, Thomas, 133
GIFs, animated, 204
Glow filter, 35, 85, 89
Google
downloading IE7 script from, 157
"explorer-canvas" script, 114
Font Directory, 129, 145
and page-loading speed, 18
WebFont Loader JavaScript library, 145

graceful degradation, 12, 15
Gradient filter, 35, 70, 71, 81
gradient-generator tools, 78, 112
gradients
  browser support, 77
  image-free, 72–81
  linear, 35, 72–73, 76–77
  radial, 72–73, 75
  simulating, 84
  using prefixed versions of, 29
  ways of using, 77
  workarounds, 78–81
graphic effects, 51
graphics programs, 44
grid positioning, 3, 285
Griffiths, Patrick, 191
Gzip, 145

H

h2:target selector, 188–189
hacks, 36
handheld media type, 240
handwritten font, 126, 137, 142, 144, 147
hash mark, 181
hasLayout, 71
Hawryluk, Zoltan, 34, 93, 96
Hazaël-Massieux, Dominique, 240
headings
  fading in/out, 191–204
  targeting, 188–189
hex color values, 63
highlighting, dynamic, 180–181
Highway Safety Research Center, 19–24
hinting, font, 132, 133
horizontal centering, 261, 263–266
horizontal nav bars, 213–214
href attribute, 155, 158
HSB values, 63
HSLA
  browser support, 67
  converting to Gradient filter, 71
  emulating, 35
  hue values cheat sheet, 65
  meaning of acronym, 62
  semitransparent backgrounds with, 31, 62–72
  syntax, 62–64
  vs. RGBA, 65–67
  ways of using, 67
  workarounds for IE, 69–72
HSL color picker, 63
HSL values, 63, 64, 65
HSV values, 63
.htaccess, 145
HTML
  adding classes to, 160
  comments, 36–37
  hierarchical structure, 169
  targeting, 3, 16, 149, 168–180
html tag, 39–41
"HTTP Caching" article, 145
HTTP requests, 16–18, 21, 38

I

Icon Eden, 223
icons
  creating number, 184–187
  indicating file types with, 153–159
  resizing, 231
  scaling, 156
  using dingbat fonts to create, 143
IE. *See also* specific versions
  and animation property, 202
  and attribute selectors, 163
  and background-clip property, 123
  and background-size property, 107, 108
  and border-image property, 118
  and border-radius property, 54
  and box-shadow property, 84–85
  and box-sizing property, 283–284
  browser prefix, 25
  CSS3 support, 9
  feeding rules/scripts to, 36
  filtering with conditional comments, 36–41
  filters, 35–36, 70, 71, 81, 93, 96
  and flexible box layout model, 276
  and @font-face rules, 9, 134, 135, 141, 142
  and generated content, 61
  and gradients, 77
  hacks, 36
  hiding content from, 38–39, 55
  market share, 7–8
  and media queries, 212
  and multi-columns, 219
  and multiple background images, 113–114
  and :nth-child( ) pseudo-class, 174, 176
  and :nth-of-type( ) pseudo-class, 179
  and RGBA/HSLA, 67, 69
  and rounded corners, 24, 54–55
  and :target pseudo-class, 190
  and text-shadow property, 88, 89
  and transforms, 91, 92–96
  and transition property, 198
  and word-wrap property, 49
IE 5, 8, 49
IE 6, 7–8, 9, 157–158, 161, 163
IE 7, 7–8, 9, 61, 163, 283–284
IE7 script, 34, 157–158, 176, 284
IE 8. *See also* IE
  and background-size property, 107
  and box-shadow property, 85
  and conditional comments, 38
  CSS3 support, 9
  and @font-face rules, 139, 141
  and HSLA/RGBA, 69–72
  market share, 7
  and multiple background images, 113–114
  and :nth-child( ) pseudo-class, 176
  and rounded corners, 54–55
IE 9. *See also* IE
  and anti-aliasing, 143
  and background-repeat property, 103, 115
  and background-size property, 104, 111
  and border-radius property, 186
  CSS3 support, 9

IE 9 *(continued)*
and flexible box layout model, 247
and @font-face rules, 141
and :nth-child( ) pseudo-class, 176
and :nth-of-type( ) pseudo-class, 179
and web font rendering, 133
IE-CSS3 script, 55, 85
IE-only style sheets, 37
image-free gradients, 72–81
image-free text shadows, 85–89
image-free visual effects, 2
image galleries, 91, 190, 276
images
angled, 91
assigning to borders, 115
background. *See* background images
border. *See* border images
displaying alternative text for, 163
positioning, 120–125
repeating, 118
revealing portions of, 112
rounding, 118
scaling, 101–107
slicing, 116, 117
stretching, 112, 118
styling, 149, 150, 151, 160–165
targeting by type, 164–165
image slideshows, 91
image tabs, 22
img[alt] selector, 151–152
Internet Explorer. *See* IE
intrinsic sizing, 248
invisible content, 57
iPad, 227, 239
iPhone, 108, 226, 227, 233–234, 235, 239
iPod Touch, 227, 233, 239
Irish, Paul, 39, 139, 141, 144, 145
italic font, 141

J

Jaeger, Jason J. 218
James, Mark 155
JavaScript
and box-sizing property, 284
dynamic highlighting with, 181
emulating CSS3 with, 34–35
for fading background color, 191–192
libraries, 145, 159, 176
and media queries, 241
and :nth-child( ) pseudo-class, 176
and :target pseudo-class, 190, 191, 204
and template layout, 285
Johanssen, Roger, 218
Johnston, Jason, 35, 54, 125
jQuery, 159, 176, 241, 285
Just Another Foundry, 131

K

Kernest, 129, 131
@keyframes rule, 199
KHTML, 25
Koch, Peter-Paul, 235, 236
Konqueror, 25

L

Lamm, Steve, 145
Last Call status, 5
:last-child pseudo-class, 170, 183–184
:last-of-type pseudo-class, 170
Lawson, Bruce, 191
layouts
for large screens, 209–219
liquid, 206–207
for mobile devices, 226–240
multi-column, 3, 245–266
for small screens, 220–225
layout viewports, 235–236
Lazaris, Louis, 39, 185
League of Moveable Type, 129
left value, 60
licensing agreements, font, 128–129
linear gradients, 35, 72–73, 76–77
line breaks, 73
lined-paper background image, 100
line length, 214
links. *See also* URLs
checking for empty, 163
displaying access key for, 163
styling, 149, 150, 151, 153–156, 163
liquid layouts, 206
lists, 184
Little Pea Bakery, 206
See also bakery site
local( ) syntax, 140–141

M

Mac
color picker, 63
font considerations, 132
Manion, Divya, 134
margins, 217
Matrix filter, 93, 96
maturity levels, CSS3 module, 4–5
mb.js script, 114
media features, 208, 212, 227
media queries
exercise files, 209, 224, 241
improving usability with, 19
for large-screen layouts, 209–219
for mobile-device layouts, 226–240
purpose of, 3, 208–209
for small-screen layouts, 220–225
testing, 238
ways of using, 209, 212
workarounds, 240–241
Media Queries module, 3
@media rule, 212
menus, vertical, 213–214
meta tag, 235–240
Meyer, Eric, 27–28
Microsoft
ClearType, 132
DirectWrite, 133
filters, 35–36
Internet Explorer. *See* IE
and word-wrap property, 49
min-height attribute, 154, 156
mobile browsers, 236
mobile devices, 207, 209, 226–240
Mobile Safari, 227, 236, 239
mobile viewports, 236
mockups, 43–45
Modernizr, 32–34, 55, 107, 113, 203, 266

MooColumns, 218
MooTools, 89, 153, 159, 176
Moveable Type, League of, 129
Mozilla, 10, 25. *See also* Firefox
-ms-filter property, 94, 95
multicol elements, 215
Multi-column Layout module, 3, 219
multi-column layouts
  adding columns to, 253–255
  browser support, 219
  creating, 245–252
  equal-height columns in, 258–260
  horizontal centering in, 263–266
  and line length, 214
  problems with, 217–218
  reordering columns in, 255–258
  vertical centering in, 261–263
  workarounds for non-supporting browsers, 218–219
Multi-column script, 218
multi-column text, 214–219. *See also* multi-column layouts
multi-row layouts, 245

N

nav bars, 213–214, 263–266, 276
nested divs, 109, 111, 113, 246
Net Applications, 7
newspaper layouts, 3
Nintendo, 25
nodes, 169
non-breaking space, 57–59
notebook paper, 99–147
  adding drop shadow to, 125–126
  adding graphic border to, 114–125
  adding stain images to, 109–111
  aligning text to lines in, 101
  applying multiple background images to, 109–114
  base page for, 100
  embedding fonts in, 126–145
  exercise files, 104, 111
  finished page for, 146–147
  scaling background image for, 101–108
"not matching" selector, 153
:not selector, 153
:nth-child( ) pseudo-class
  how it works, 170, 171–172
  negative values in, 175
  online tools, 172
  rotating photos with, 176–180
  ways of using, 168, 175
  workarounds for IE, 176
  zebra striping with, 172–176
:nth-last-child( ) pseudo-class, 168, 170
:nth-last-of-type( ) pseudo-class, 168
:nth-of-type( ) pseudo-class, 168, 170, 179
number icons, 184–187
number sign, 181
Nyman, Robert, 71

O

Obsidian Dawn, 110
octothorps, 181
ol element, 184
online color tools, 64
:only-child pseudo-class, 170
:only-of-type pseudo-class, 170
opacity property, 63
OpenType, 132, 134
Opera. *See also* browsers
  and background-repeat property, 103, 115
  and background-size property, 104, 106, 111
  and border-image property, 120
  browser prefix, 25
  CSS3 support, 9, 10
  and @font-face rules, 135
  market share, 7
  and multiple background images, 113–114
  navigation div bug, 210
  and transitions, 192
Opera Mobile, 135
ordered lists, 184
OTF font files, 132, 134–135
overflows, 217, 264, 265
overlapping elements, 232–233

P

padding, 217
padding-box value, 122, 123, 283
page performance, 16–18, 21
pagination, 217
parallax effect, 112
parent elements, 169
Patenaude, Matt, 63
PDF files, 153, 156
Phinney, Thomas, 133
photos. *See also* images
  angled, 91
  rotating, 176–180
  styling full-size *vs.* thumbnails, 160–165
Photoshop, 63
photo slideshows, 91
PIE script, 25, 35, 54, 70, 80, 85
Pinckaers, Hans, 114
pixel-based measurements, 105
PNGs, alpha-transparent, 62, 69–70, 79
Polaroid-style photos, 164–165
pound sign, 181
prefixes, browser, 25–30
Prelude font, 137, 141–142
presentational-behavior effects, 192–193
Presto, 25
progressive enhancement, 11–15, 40–41
properties. *See also* specific properties
  animatable, 196
  browser-specific prefixes for, 25
properties
  flexible box layout model, 277
  providing more than one value for, 32
  using browser-specific, 28–30
Proposed Recommendation status, 5
ProtoFluid, 238
pseudo-classes, 168–171
pseudo-elements, 57, 168, 169

Q

QuirksMode, 10
quotation marks, 112, 154

R

radial gradients, 72–73, 75, 76
Rahnas, Remiz, 54
Rainbow Firefox extension, 64, 65
Raleway font, 127, 129, 132
readability, 19, 86, 89, 92, 132–134
Recommendation status, 5
rendering engines, 25, 30
repeat value, 118
retina display, 233
RGBA
- browser support, 67
- converting to Gradient filter, 71
- emulating, 35
- meaning of acronym, 62
- semitransparent backgrounds with, 31, 62–72
- syntax, 62–64
- *vs.* HSLA, 65–67
- ways of using, 67
- workarounds for IE, 69–72

RGB values, 63, 64
Roberts, Harry, 179
:root pseudo-class, 170
rotate transform function, 90, 93
rotation, random, 176–180
rounded corners, 51–55
round value, 118, 126
Ruter, Weston, 80
Rutter, Richard 218

S

Safari. *See also* browsers
- and background-clip property, 122
- and background-size property, 104, 111
- and border-image property, 120
- and box-shadow property, 164
- browser prefix, 25
- CSS3 support, 8–9, 10
- and drop shadows, 126
- and float property, 264
- and @font-face rules, 135
- and local( ) syntax, 140
- market share, 7
- and round value, 126
- and spread radius, 164

Safari on iOS, 135
Sass, 27
Savarese, Cédric, 218
scale transform function, 90
scaling
- background images, 101–107, 108
- border images, 115
- icons, 156

screen sizes
- accommodating different, 205, 206–207
- targeting specific, 226

scripts
- for attribute selectors, 157–158, 159
- for box-sizing property, 284
- CSS3 emulation, 34–35
- for media queries, 241
- for multi-column text, 218
- for rounded corners, 54–55
- for stretching border images, 125
- for :target pseudo-class, 191, 204
- for zebra striping, 176

search engine placement, 18
Selectivizr, 34, 159
SelectORacle, 152
selectors, 3, 149, 159, 167. *See also* attribute selectors
Selectors module, 3, 162, 175, 179, 190
Selector Utility, YUI, 159
semitransparent backgrounds, 31, 62–72, 82
Shadow filter, 35, 85, 89
shadows
- drop, 81–85
- text, 85–89

sibling elements, 169
sideways text, 91
Silk icon set, 155
Simons, Randy, 218
Sizzle, 159
skewed boxes, 91
skew transform function, 90
slideshows, 91, 190
Sly, 159
smartphones, 235. *See also* specific devices
smiley faces, 53
Soma Fontfriend, 132
space value, 118
special character code, 58
speech bubbles, 47–97
- adding alternating colors to, 171–176
- adding drop shadows behind, 81–85
- adding tails to, 55–61
- applying gradients to, 72–81
- base page for, 48–49
- creating semitransparent backgrounds for, 68–72
- exercise files, 49, 61, 70, 81, 87, 96
- finished page for, 96–97
- increasing 3D appearance of, 83
- making text stand out in, 86–89
- rotating photos in, 176–180
- rounding corners of, 51–55
- transforming avatars for, 89–96
- wrapping text in, 49–51

spiral notebook-paper effect, 114–116
Spolsky, Joel, 133
spread radius, 164
sprites, 21, 22, 185
src attribute, 164
stability status, CSS3, 4
stain images, 109–111
Standardista, 10
star html hack, 36
Stefanov, Stoyan, 145
sticky footers, 272–275
sticky notes, 100
Storey, David, 185
stretch value, 258, 259, 260
structural pseudo-classes, 169–171
stunningcss3.com
- CSS3 emulation scripts, 35
- flex-box exercise files, 245
- font-embedding resources, 131
- media queries exercise files, 209

notebook paper exercise files, 104
selectors exercise files, 153
speech bubbles exercise files, 49
web font resources, 130, 135
zebra striping exercise files, 172
style sheets, 37, 163, 212
styling
numbers, 186
table of contents, 183–184
type-based, 151
subscription form, 268–272
substring matching attribute selectors, 152, 162
“Suckerfish :target” script, 191
SuperSelectors plugin, 159
SVG fonts, 134, 135, 136
Swan, Elliot, 193

T

table of contents, 182–187, 204
tabs, 53
tag clouds, 91
tails, speech bubble, 55–61
targeting HTML elements, 3, 16, 149, 168–180. *See also* selectors
`:target` pseudo-class, 181–182, 188–191, 204
template layout, 284–285
Template Layout module, 3
text. *See also* fonts
displaying alternative, 163
multi-column, 214–219
real *vs.* images of, 19
sideways, 91
wrapping, 9, 49–51
Text module, 88
text-shadow generator, 87
`text-shadow` property, 35, 85–89
text-shadow script, 89
`text-wrap` property, 50
thumbnails, 160–165
thumbtack image, 110
`title` attribute, 163
TOC, 182–187, 204
`top` value, 60
torn-edge effect, 114–125, 147
transform functions, 89, 90
Transformie script, 96
`transform-origin` property, 90, 92
`transform` property, 92
transforms
benefits of using, 92
browser support, 91
defined, 89
purpose of, 2
rotating avatars with, 92–93
scripting changes in, 96
syntax for, 92
using multiple, 90
ways of using, 91
workarounds for IE, 92–96
transforms generator, 92
Transforms Translator, 93, 94
`transition` property, 194–198
transitions, 3, 192–204
Transitions module, 197
`translate` transform function, 90
transparency, 62. *See also* semitransparent backgrounds
triangles, 55–56
Trident, 25
TrueType, 132, 134–135
ttf2svg, 136
TTF fonts, 132, 134–135
Twitter, 31
type-based styling, 151
type delivery services, 130–131
TypeFront, 128, 129
Typekit, 130
typography, 3. *See also* fonts
Typotheque, 131, 133

U

`ul` element, 184
UNC Highway Safety Research Center, 19–24
Unicode Code Converter, 57
Unicode code points, 58
unordered lists, 184
URLs, 49, 163, 181–182. *See also* links
usability, 19, 92, 172, 218
user style sheets, 163

V

Van Damme, Tim, 133
van der Graaf, Wouter, 241
van Ouwerkerk, Michael, 218
vendor prefixes, 25–26
Verou, Lea, 172
vertical centering, 261–263
vertical menus, 213–214
viewport `meta` tag, 235–240
virtual viewports, 235
visual effects, 2
visual viewports, 235

W

W3C
and attribute selectors, 152
box model, 277
and CSS3 maturity levels, 4–5
and Firefox, 73
and WOFF specification, 134, 135
water stain images, 110
Webb, Dan, 191
web-based color tools , 64
web browsers. *See* browsers
Web Developer extension, Firefox, 238
Web FontFonts, 130
WebFont Loader JavaScript library, 145
web fonts, 127, 128, 132. *See also* fonts
Web Fonts from Ascender, 131
Webfonts.info, 129
WebINK, 131
Webkit
and anti-aliasing, 133
and `background-clip` property, 122
browser prefix, 25
and `@font-face` rules, 144
and font smoothing, 133
and gradients, 73
linear gradient syntax, 75, 79
and long URLs, 49
and transitions, 192
Web Open Font Format, 134–135

web pages
  choosing fonts for, 128–134
  hiding ads on, 163
  highlighting sections of, 180–204
  how people view, 205
  improving search engine placement for, 18
  progressive enhancement of, 11
  speeding up, 16, 21
web-safe fonts, 127
Webtype, 131
Wii, 25
Wikipedia, 7, 10, 180–182
Windows fonts, 132
wireframes, 44
WOFF files, 134–135, 139, 145
`word-wrap` property, 9, 49–50
Working Draft status, 5
Wroblewski, Luke, 234
Wulf, Adam, 218

X

XHTML Character Entity Reference, 58

Y

Yahoo, 159
Yellow Fade Technique, 191–192, 204
YUI Selector Utility, 159
Yummy icon set, 223

Z

zebra striping, 172–176