

“The Missing Manual series is simply the most intelligent and usable series of guidebooks...”  
—KEVIN KELLY, CO-FOUNDER OF WIRED

# CSS3

**the missing manual<sup>®</sup>**

The book that should have been in the box<sup>®</sup>

Fully Revised  
3rd Edition  
Covers HTML5

Free Sampler

TM

O'REILLY<sup>®</sup>

David Sawyer McFarland

# Want to read more?

You can [buy this book](#) at **oreilly.com**  
in print and ebook format.

**Buy 2 books, get the 3rd FREE!**

Use discount code: OPC10

All orders over \$29.95 qualify for **free shipping** within the US.

---

It's also available at your favorite book retailer,  
including the iBookstore, the [Android Marketplace](#),  
and [Amazon.com](#).



**O'REILLY®**

Spreading the knowledge of innovators

[oreilly.com](#)

# **CSS3: The Missing Manual, Third Edition**

*by David Sawyer McFarland*

Copyright © 2013 Sawyer McFarland Media, Inc. All rights reserved.  
Printed in the United States of America.

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

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

August 2006: First Edition.  
August 2009: Second Edition.  
December 2012: Third Edition.

## **Revision History for the Third Edition:**

2012-12-07 First release

See <http://oreilly.com/catalog/errata.csp?isbn=0636920024996> for release details.

The Missing Manual is a registered trademark of O'Reilly Media, Inc. The Missing Manual logo, and “The book that should have been in the box” are trademarks of O'Reilly Media, Inc. Many of the designations used by manufacturers and sellers to distinguish their products are claimed as trademarks. Where those designations appear in this book, and O'Reilly Media is aware of a trademark claim, the designations are capitalized.

While every precaution has been taken in the preparation of this book, the publisher assumes no responsibility for errors or omissions, or for damages resulting from the use of the information contained in it.

ISBN-13: 978-1-449-32594-7

[V]

# Contents

<b>The Missing Credits</b> .....	<b>vii</b>
----------------------------------	------------

<b>Introduction</b> .....	<b>1</b>
---------------------------	----------

## Part One: **CSS Basics**

---

CHAPTER 1:	<b>HTML for CSS</b> .....	<b>17</b>
	HTML: Past and Present .....	17
	Writing HTML for CSS .....	20
	The Importance of the Doctype .....	30
	Making Sure Internet Explorer Is Up-to-Date .....	32
CHAPTER 2:	<b>Creating Styles and Style Sheets</b> .....	<b>35</b>
	Anatomy of a Style .....	35
	Understanding Style Sheets .....	38
	Internal Style Sheets .....	40
	External Style Sheets .....	41
	Tutorial: Creating Your First Styles .....	43
CHAPTER 3:	<b>Selectors: Identifying What to Style</b> .....	<b>55</b>
	Tag Selectors: Page-Wide Styling .....	56
	Class Selectors: Pinpoint Control .....	57
	ID Selectors: Specific Page Elements .....	60
	Styling Groups of Tags .....	62
	Styling Tags Within Tags .....	63
	Pseudo-Classes and Pseudo-Elements .....	68
	Attribute Selectors .....	72
	Child Selectors .....	74
	Siblings .....	79
	The :not( ) Selector .....	80
	Tutorial: Selector Sampler .....	81
CHAPTER 4:	<b>Saving Time with Style Inheritance</b> .....	<b>93</b>
	What Is Inheritance? .....	93
	How Inheritance Streamlines Style Sheets .....	95
	The Limits of Inheritance .....	95
	Tutorial: Inheritance .....	97

CHAPTER 5:	<b>Managing Multiple Styles: The Cascade</b> .....	<b>103</b>
	How Styles Cascade .....	104
	Specificity: Which Style Wins .....	108
	Controlling the Cascade .....	111
	Tutorial: The Cascade in Action .....	117

## Part Two: **Applied CSS**

---

CHAPTER 6:	<b>Formatting Text</b> .....	<b>127</b>
	Using Fonts .....	127
	Using Web Fonts .....	132
	Discovering Google Web Fonts .....	146
	Adding Color to Text .....	154
	Changing Font Size .....	157
	Formatting Words and Letters .....	162
	Adding Text Shadow .....	166
	Formatting Entire Paragraphs .....	167
	Styling Lists .....	173
	Tutorial: Text Formatting in Action .....	178
CHAPTER 7:	<b>Margins, Padding, and Borders</b> .....	<b>193</b>
	Understanding the Box Model .....	193
	Controlling Space with Margins and Padding .....	195
	Adding Borders .....	202
	Coloring the Background .....	206
	Creating Rounded Corners .....	207
	Adding Drop Shadows .....	210
	Determining Height and Width .....	212
	Wrapping Content with Floating Elements .....	220
	Tutorial: Margins, Backgrounds, and Borders .....	225
CHAPTER 8:	<b>Adding Graphics to Web Pages</b> .....	<b>239</b>
	Discovering CSS and the <img> Tag .....	239
	Adding Background Images .....	240
	Controlling Repetition .....	244
	Positioning a Background Image .....	245
	Using Background Property Shorthand .....	255
	Using Multiple Background Images .....	257
	Utilizing Gradient Backgrounds .....	259
	Applying Easy Gradients with Colorzilla .....	269
	Tutorial: Enhancing Images .....	272
	Tutorial: Creating a Photo Gallery .....	278
	Tutorial: Using Background Images .....	283

CHAPTER 9:	<b>Sprucing Up Your Site's Navigation</b>	<b>293</b>
	Selecting Which Links to Style	293
	Styling Links	297
	Building Navigation Bars	303
	CSS-Style Preloading Rollovers	313
	Styling Particular Types of Links	315
	Tutorial: Styling Links	318
	Tutorial: Creating a Navigation Bar	324
CHAPTER 10:	<b>CSS Transforms, Transitions, and Animations</b>	<b>335</b>
	Transforms	335
	Transitions	345
	Animations	354
	Tutorial	366
CHAPTER 11:	<b>Formatting Tables and Forms</b>	<b>375</b>
	Using Tables the Right Way	375
	Styling Tables	378
	Styling Forms	384
	Tutorial: Styling a Table	390
	Tutorial: Styling a Form	395
 <b>Part Three: CSS Page Layout</b>		
CHAPTER 12:	<b>Introducing CSS Layout</b>	<b>405</b>
	Types of Web Page Layouts	405
	How CSS Layout Works	408
	Layout Strategies	412
CHAPTER 13:	<b>Building Float-Based Layouts</b>	<b>417</b>
	Applying Floats to Your Layouts	421
	Overcoming Float Problems	425
	Tutorial: Multiple Column Layouts	440
CHAPTER 14:	<b>Responsive Web Design</b>	<b>453</b>
	Responsive Web Design Basics	454
	Setting Up a Web Page for RWD	455
	Media Queries	457
	Flexible Grids	465
	Fluid Images	470
	Responsive Web Design Tutorial	474

CHAPTER 15:	<b>Positioning Elements on a Web Page</b>	<b>487</b>
	How Positioning Properties Work	488
	Powerful Positioning Strategies	502
	Tutorial: Positioning Page Elements	507

## Part Four: **Advanced CSS**

---

CHAPTER 16:	<b>CSS for the Printed Page</b>	<b>519</b>
	How Media Style Sheets Work	519
	How to Add Media Style Sheets	522
	Creating Print Style Sheets	523
	Tutorial: Building a Print Style Sheet	530
CHAPTER 17:	<b>Improving Your CSS Habits</b>	<b>539</b>
	Adding Comments	539
	Organizing Styles and Style Sheets	541
	Eliminating Browser Style Interference	548
	Using Descendent Selectors	553
	Trying Different CSS for Internet Explorer	558

## Part Five: **Appendixes**

---

APPENDIX A:	<b>CSS Property Reference</b>	<b>563</b>
	CSS Values	563
	Text Properties	567
	List Properties	572
	Padding, Borders, and Margins	573
	Backgrounds	579
	Page Layout Properties	582
	Animation, Transform, and Transition Properties	588
	Table Properties	593
	Miscellaneous Properties	595
APPENDIX B:	<b>CSS Resources</b>	<b>599</b>
	References	599
	CSS Help	600
	CSS Tips, Tricks, and Advice	600
	CSS Navigation	601
	CSS Layout	602
	Showcase Sites	603
	CSS Books	604
	CSS Software	604
	<b>Index</b>	<b>607</b>

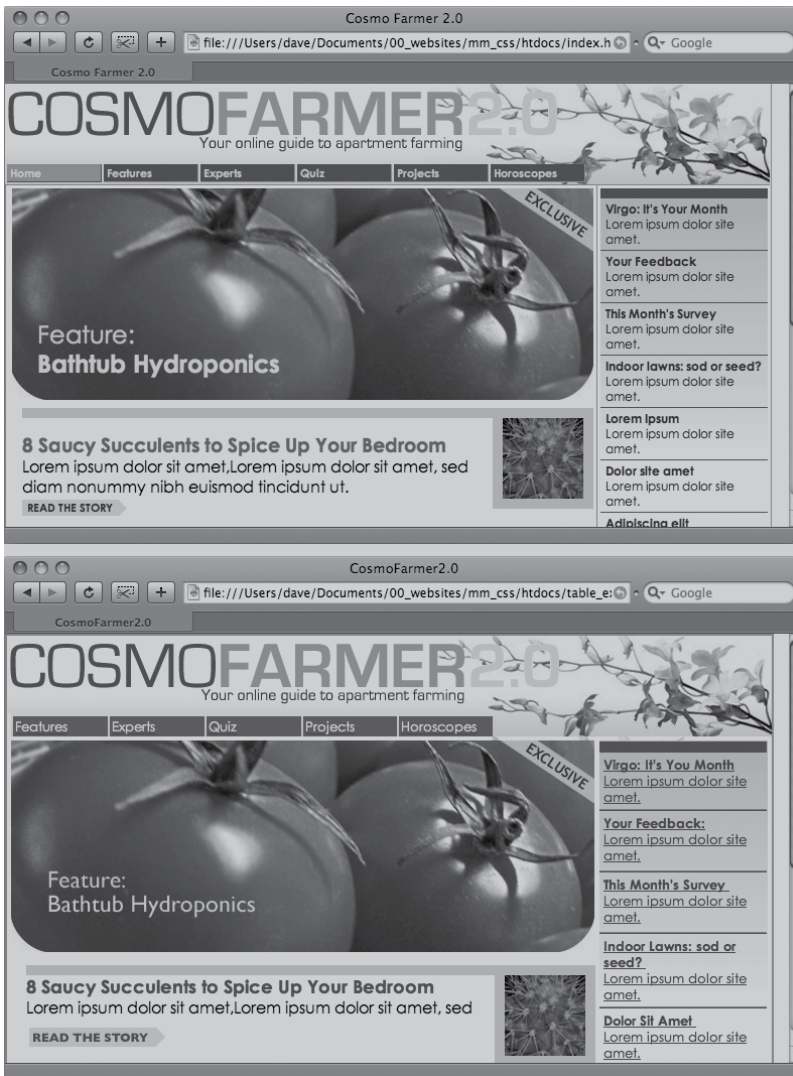
# HTML for CSS

To get the most out of CSS, your HTML code needs to provide a solid, well-built foundation. This chapter shows you how to write better, more CSS-friendly HTML. The good news is that when you use CSS throughout your site, HTML actually becomes *easier* to write. You no longer need to worry about trying to turn HTML into the design maven it was never intended to be. Instead, CSS offers most of the graphic design touches you'll likely ever want, and HTML pages written to work with CSS are easier to create since they require less code and less typing. They'll also download faster—a welcome bonus your site's visitors will appreciate (see Figure 1-1).

## ■ HTML: Past and Present

As discussed in the Introduction, HTML provides the foundation for every page you encounter on the World Wide Web. When you add CSS into the mix, the way you use HTML changes. Say goodbye to repurposing awkward HTML tags merely to achieve certain visual effects. Some HTML tags and attributes—like the old `<font>` tag—you can forget completely.





**FIGURE 1-1**

*CSS-driven web design makes writing HTML easier. The two designs pictured here look similar, but the top page is styled completely with CSS, while the bottom page uses only HTML. The size of the HTML file for the top page is only 4k, while the HTML-only page is nearly 4 times that size at 14k. The HTML-only approach requires a lot more code to achieve nearly the same visual effects: 213 lines of HTML code compared with 71 lines for the CSS version.*

## HTML Past: Whatever Looked Good

When a bunch of scientists created the Web to help share and keep track of technical documentation, nobody called in the graphic designers. All the scientists needed HTML to do was structure information for easy comprehension. For example, the `<h1>` tag indicates an important headline, while the `<h2>` tag represents a lesser heading,

usually a subheading of the `<h1>` tag. Another favorite, the `<ol>` (ordered list) tag, creates a numbered list for things like “Top 10 reasons not to play with jellyfish.”

But as soon as people besides scientists started using HTML, they wanted their web pages to look good. So web designers started to use tags to control appearance rather than structure information. For example, you can use the `<blockquote>` tag (intended for material that’s quoted from another source) on any text that you want to indent a little bit. You can use heading tags to make any text bigger and bolder—regardless of whether it functions as a heading.

In an even more elaborate workaround, designers learned how to use the `<table>` tag to create columns of text and accurately place pictures and text on a page. Unfortunately, since that tag was intended to display spreadsheet-like data—research results, train schedules, and so on—designers had to get creative by using the `<table>` tag in unusual ways, sometimes nesting a table within a table within a table to make their pages look good.

Meanwhile, browser makers introduced new tags and attributes for the specific purpose of making a page look better. The `<font>` tag, for example, lets you specify a font color, typeface, and one of seven different sizes. (If you’re keeping score at home, that’s about 100 fewer sizes than you can get with, say, Microsoft Word.)

Finally, when designers couldn’t get exactly what they wanted, they often resorted to using graphics. For example, they’d create a large graphic to capture the exact font and layout for web page elements and then slice the Photoshop files into smaller files and piece them back together inside tables to recreate the original design.

While all of the preceding techniques—using tags in creative ways, taking advantage of design-specific tag attributes, and making extensive use of graphics—provide design control over your pages, they also add a lot of additional HTML code (and more wrinkles to your forehead).

## HTML Present: Scaffolding for CSS

No matter what content your web page holds—the fishing season calendar, driving directions to the nearest IKEA, or pictures from your kid’s birthday party—it’s the page’s design that makes it look like either a professional enterprise or a part-timer’s hobby. Good design enhances the message of your site, helps visitors find what they’re looking for, and determines how the rest of the world sees your website. That’s why web designers went through the contortions described in the previous section to force HTML to look good. By taking on those design duties, CSS lets HTML go back to doing what it does best—structure content.

Using HTML to control the look of text and other web page elements is obsolete. Don’t worry if HTML’s `<h1>` tag is too big for your taste or bulleted lists aren’t spaced just right. You can take care of that later using CSS. Instead, think of HTML as a method of adding structure to the content you want up on the Web. Use HTML to organize your content and CSS to make that content look great.

## ■ Writing HTML for CSS

If you're new to web design, you may need some helpful hints to guide your forays into HTML (and to steer clear of well-intentioned, but out-of-date HTML techniques). Or if you've been building web pages for a while, you may have picked up a few bad HTML-writing habits that you're better off forgetting. The rest of this chapter introduces you to some HTML writing habits that will make your mother proud—and help you get the most out of CSS.

### Think Structure

HTML adds meaning to text by logically dividing it and identifying the role that text plays on the page: For example, the `<h1>` tag is the most important introduction to a page's content. Other headers let you divide the content into other, less important, but related sections. Just like the book you're holding, for example, a web page should have a logical structure. Each chapter in this book has a title (think `<h1>`) and several sections (think `<h2>`), which in turn contain smaller subsections. Imagine how much harder it would be to read these pages if every word just ran together as one long paragraph.

#### NOTE

For a tutorial on HTML, visit [www.w3schools.com/html/html\\_intro.asp](http://www.w3schools.com/html/html_intro.asp). For a quick list of all available HTML tags, visit Sitepoint.com's HTML reference at <http://reference.sitepoint.com/html>.

HTML provides many other tags besides headers for *marking up* content to identify its role on the page. (After all, the M in HTML stands for *markup*.) Among the most popular are the `<p>` tag for paragraphs of text and the `<ul>` tag for creating bulleted (non-numbered) lists. Lesser-known tags can indicate very specific types of content, like `<abbr>` for abbreviations and `<code>` for computer code.

When writing HTML for CSS, use a tag that comes close to matching the role the content plays in the page, not the way it looks (see Figure 1-2). For example, a bunch of links in a navigation bar isn't really a headline, and it isn't a regular paragraph of text. It's most like a bulleted list of options, so the `<ul>` tag is a good choice. If you're saying, "But items in a bulleted list are stacked vertically one on top of the other, and I want a horizontal navigation bar where each link sits next to the previous link," don't worry. With CSS magic you can convert a vertical list of links into a stylish horizontal navigation bar as described in Chapter 9.

```
<p>
<strong>
<font color="#0066FF" size="5" face="Verdana,
Arial, Helvetica, sans-serif">Urban Agrarian
Lifestyle</font></strong>
<br />
<font color="#FF3300" size="4" face="Georgia,
Times New Roman, Times, serif">
<em>
<strong>A Revolution in Indoor Agriculture
<br /></strong></em></font>
Lorem ipsum dolor sit amet...</p>
```

## The Urban Agrarian Lifestyle

### *A Revolution in Indoor Agriculture*

Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed diam nonummy nibh euismod tincidunt ut laoreet dolore magna aliquam erat volutpat. Ut wisi enim ad minim veniam, quis nostrud exerci tation ullamcorper suscipit lobortis nisl ut aliquip ex ea commodo consequat. Duis autem vel eum iriure.

```
<h1>The Urban Agrarian Lifestyle</h1>
<h2>A Revolution in Indoor Agriculture</h2>
<p>Lorem ipsum dolor sit amet...</p>
```

**FIGURE 1-2**

*Old school, new school. Before CSS, designers had to resort to the <font> tag and other extra HTML to achieve certain visual effects (top). You can achieve the same look (and often a better one) with a lot less HTML code (bottom). In addition, using CSS for formatting frees you up to write HTML that follows the logical structure of the page's content.*

## Two HTML Tags to Keep in Mind

HTML's motley assortment of tags doesn't cover the wide range of content you'll probably add to a page. Sure, <code> is great for marking up computer program code, but most folks would find a <recipe> tag handier. Too bad there isn't one. Fortunately, HTML provides two generic tags that let you better identify content, and, in the process, provide "handles" that let you attach CSS styles to different elements on a page.

## UP TO SPEED

### Simple HTML Is Search Engine Friendly

Once you take the mental leap of picturing HTML as the way to structure a document's content, and CSS as the tool for making that content look good, you'll discover additional benefits to writing lean, mean HTML. For one thing, you may boost your search-engine ranking as determined by sites like Google, Yahoo, and Bing. That's because when search engines crawl the Web, indexing the content on websites, they must go through *all* the HTML on a page to discover the actual content. The old HTML way of using special tags (like `<font>`) and lots of tables to design a page gets in the way of a search engine's job. In fact, some search engines stop reading a page's HTML after a certain number of characters. When you use a lot of HTML just

for design, the search engine may miss important content on the page or even fail to rank it at all.

By contrast, simple, structured HTML is easy for a search engine to read and index. Using an `<h1>` tag to indicate the most important topic of the page (as opposed to just making the text big and bold) is smart strategy: Search engines give greater weight to the contents inside that tag while indexing the page.

To see Google's suggestions for building search-friendly websites, visit <http://support.google.com/webmasters/bin/answer.py?hl=en&answer=35769>.

### ■ UNDERSTANDING THE `<div>` AND `<span>` TAGS

The `<div>` tag and the `<span>` tag are like empty vessels that you fill with content. A `div` is a block, meaning it has a line break before it and after it, while a `span` appears inline, as part of a paragraph. Otherwise, `div`s and `span`s have no inherent visual properties, so you can use CSS to make them look any way you want. The `<div>` (for *division*) tag indicates any discrete block of content, much like a paragraph or a headline. But more often it's used to group any number of *other* elements, so you can insert a headline, a bunch of paragraphs, and a bulleted list inside a single `<div>` block. The `<div>` tag is a great way to subdivide a page into logical areas, like a banner, footer, sidebar, and so on. Using CSS, you can later position each area to create sophisticated page layouts (a topic that's covered in Part III of this book).

The `<span>` tag is used for *inline* elements; that is, words or phrases that appear inside of a larger paragraph or heading. Treat it just like other inline HTML tags such as the `<a>` tag (for adding a link to some text in a paragraph) or the `<strong>` tag (for emphasizing a word in a paragraph). For example, you could use a `<span>` tag to indicate the name of a company, and then use CSS to highlight the name by using a different font, color, and so on. Here's an example of those tags in action, complete with a sneak peek of a couple of attributes—`id` and `class`—frequently used to attach styles to parts of a page.

```
<div id="footer">
  <p>Copyright 2006, <span class="bizName">CosmoFarmer.com</span></p>
  <p>Call customer service at 555-555-5501 for more information</p>
</div>
```

This brief introduction isn't the last you'll see of these tags. They're used frequently in CSS-heavy web pages, and in this book you'll learn how to use them in combination with CSS to gain creative control over your web pages.

## ■ ADDITIONAL TAGS IN HTML5

The `<div>` tag is rather generic—it's simply a block-level element used to divide a page into sections. One of the goals of HTML5 is to provide other, more *semantic* tags for web designers to choose from. Making your HTML more semantic simply means using tags that accurately describe the content they contain. As mentioned earlier in this section, you should use the `<h1>` (heading 1) tag when placing text that describes the primary content of a page. Likewise, the `<code>` tag tells you clearly what kind of information is placed inside—programming code.

HTML5 includes many different tags whose names reflect the type of content they contain, and can be used in place of the `<div>` tag. The `<article>` tag, for example, is used to mark off a section of a page that contains a complete, independent composition...in other words, an “article” as in a blog post, an online magazine article, or simply the main text of the page. Likewise, the `<header>` tag indicates a header or banner, the top part of a page usually containing a logo, site-wide navigation, page title and tagline, and so on.

### NOTE

To learn more about the new HTML tags, visit HTML5 Doctor (<http://html5doctor.com>) and [www.w3schools.com/html5/html5\\_intro.asp](http://www.w3schools.com/html5/html5_intro.asp).

Many of the new HTML5 tags are intended to expand upon the generic `<div>` tag. Here are a few other HTML5 tags frequently used to structure the content on a page:

- The **`<section>`** tag contains a grouping of related content, such as the chapter of a book. For example, you could divide the content of a home page into three sections: one for an introduction to the site, one for contact information, and another for latest news.
- The **`<aside>`** tag holds content that is related to content around it. A sidebar in a print magazine, for example.
- The **`<footer>`** tag contains information you'd usually place in the footer of a page, such as a copyright notice, legal information, some site navigation links, and so on. You're not limited, however, to just a single `<footer>` per page; you can put a footer inside an `<article>`, for example, to hold information related to that article, like footnotes, references, or citations.
- The **`<nav>`** element is used to contain primary navigation links.
- The **`<figure>`** tag is used for an illustrative figure. You can place an `<img>` tag inside it, as well as another new HTML5 tag—the `<figcaption>` tag, which is used to display a caption explaining the photo or illustration within the `<figure>`.

**TIP** Understanding which HTML5 tag to use—should your text be an `<article>` or a `<section>`?—can be tricky. For a handy flowchart that makes sense of HTML5’s new sectioning elements, download the PDF from the HTML5doctor.com at <http://html5doctor.com/downloads/h5d-sectioning-flowchart.pdf>.

There are other HTML5 elements, and many of them simply provide a more descriptive alternative to the `<div>` tag. This book uses both the `<div>` tag and the new HTML5 tags to help organize web-page content. The downside of HTML5 is that Internet Explorer 8 and earlier don’t recognize the new tags without a little bit of help (see the box below).

In addition, aside from feeling like you’re keeping up with the latest web design trends, there’s really no tangible benefit to using some of these HTML5 tags. For example, simply using the `<article>` tag to hold the main story on a web page, doesn’t make Google like you any better. You can comfortably continue using the `<div>` tag, and avoid the HTML5 sectioning elements if you like.

## WORKAROUND WORKSHOP

### Getting IE 8 to Understand HTML5

HTML5 provides many new HTML tags to play with. From tags that clearly describe the kind of content they hold, like the `<nav>` tag, to ones that provide added functionality, like the `<video>` tag for embedding videos and the `<audio>` tag for embedding sound and music. As you learn more about HTML5, you’ll probably start applying these new tags to your web pages.

Unfortunately, Internet Explorer 8 and earlier don’t recognize these new tags, and won’t respond to any CSS you apply to them. That’s right—if you’re using HTML5 and viewing web pages in IE 8, this book is useless to you. Well...not exactly. There is a way to kick those old versions of IE into gear, so they’ll understand all the CSS that applies to HTML5 tags.

Simply place the following code before the closing `</head>` tag at the top of your HTML file:

```
<!--[if lt IE 9]>
```

```
<script src="//html5shiv.googlecode.com/
svn/trunk/html5.js"></script>
<![endif]-->
```

This tricky bit of code uses what’s called an “Internet Explorer conditional comment” (IECC for short) to embed a bit of JavaScript code that’s only visible to versions of Internet Explorer earlier than IE 9. In other words, only IE 6, 7, and 8 respond to this code, and all other browsers (including newer versions of IE) simply ignore it. This code makes earlier versions of IE load a small JavaScript program that forces the browser to recognize HTML5 tags and apply the CSS that applies to those tags.

This code only affects how the browser displays and prints HTML5 tags; it doesn’t make the browser “understand” an HTML5 tag that actually does something. For example, IE 8 and earlier don’t understand the `<video>` tag and can’t play HTML5 video (even with the added JavaScript code).

## Keep Your Layout in Mind

While you’ll use the `<h1>` tag to identify the main topic of the page and the `<p>` tag to add a paragraph of text, you’ll eventually want to organize a page’s content into a pleasing layout. As you learn how to use CSS to lay out a page in Part Three of this book, it doesn’t hurt to keep your design in mind while you write the page’s HTML.



You can think of web page layout as the artful arrangement of boxes (see Figure 1-3 for an example). After all, a two-column design consisting of two vertical columns of text is really just two rectangular boxes sitting side-by-side. A header consisting of a logo, tagline, search box, and site navigation is really just a wide rectangular box sitting across the top of the browser window. In other words, if you imagine the groupings and layout of content on a page, you'd see boxes sitting on top of one another, next to each other, and below each other.



In your HTML, you create these boxes, or structural units, using the `<div>` tag. Simply wrap the HTML tags that make up the banner area, for example, in one div, a column's worth of HTML in another, and so on. If you're HTML5 savvy, you might create the design pictured in Figure 1-3, with a `<header>` tag for the top banner, an `<article>` tag for the main text, an `<aside>` or `<section>` tag for the sidebar, and a `<footer>` tag for the page's footer. In other words, if you plan to place a group of HTML tags together, somewhere on a page, then you'll need to wrap those tags in a sectioning element like a `<div>`, `<article>`, `<section>`, or `<aside>`.



## HTML to Forget

CSS lets you write simpler HTML for one big reason: You can stop using a bunch of tags and attributes that only make a page better looking. The `<font>` tag is the most glaring example. Its sole purpose is to add a color, size and font to text. It doesn't do anything to make the structure of the page more understandable.

Here's a list of tags and attributes you can easily replace with CSS:

- **Ditch `<font>` for controlling the display of text.** CSS does a much better job with text. (See Chapter 6 for text-formatting techniques.)
- **Don't use the `<b>` and `<i>` tags to emphasize text.** If you want text to really be emphasized, use the `<strong>` tag, which browsers normally display as bold. For a slightly less emphatic point, use the `<em>` tag, which browsers display as italic.

While HTML4 tried to phase the `<b>` and `<i>` tags out, HTML5 has brought them back. In HTML5 the `<b>` tag is meant to merely make text bold without adding any meaning to that text (that is, you just want the text to be bold looking but you don't want people to treat that text like you're shouting it). Likewise, the `<i>` tag is used for italicizing text, but not emphasizing its meaning.

### NOTE

To italicize a publication's title, the `<cite>` tag kills two birds with one stone. It puts the title in *italics* and tags it as a cited work for search engines' benefit. This one's a keeper.

- **Skip the `<table>` tag for page layout.** Use it only to display tabular information like spreadsheets, schedules, and charts. As you'll see in Part Three of this book, you can do all your layout with CSS for much less time and code than the table-tag tango.
- **Avoid the awkward `<body>` tag attributes** that enhance only the presentation of the content: `background`, `bgcolor`, `text`, `link`, `alink`, and `vlink` set colors and images for the page, text, and links. CSS gets the job done better (see Chapter 7 and Chapter 8 for CSS equivalents of these attributes).
- **Don't abuse the `<br>` tag.** If you grew up using the `<br>` tag (`<br />` in XHTML) to insert a line break without creating a new paragraph, then you're in for a treat. (Browsers automatically—and sometimes infuriatingly—insert a bit of space between paragraphs, including between headers and `<p>` tags. In the past, designers used elaborate workarounds to avoid paragraph spacing they didn't want, like replacing a single `<p>` tag with a bunch of line breaks and using a `<font>` tag to make the first line of the paragraph *look like* a headline.) Using CSS's margin controls, you can easily set the amount of space you want to see between paragraphs, headers, and other block-level elements.

**NOTE**

In the next chapter, you'll learn about a technique called a "CSS Reset," which eliminates the gaps browsers normally insert between paragraphs and other tags (see page 115).

As a general rule, adding attributes to tags that set colors, borders, background images, or alignment—including attributes that let you format a table's colors, backgrounds, and borders—is pure old-school HTML. So is using alignment properties to position images and center text in paragraphs and table cells. Instead, look to CSS to control text placement (page 169), borders (page 202), backgrounds (page 206), and image alignment (page 239).

**UP TO SPEED**

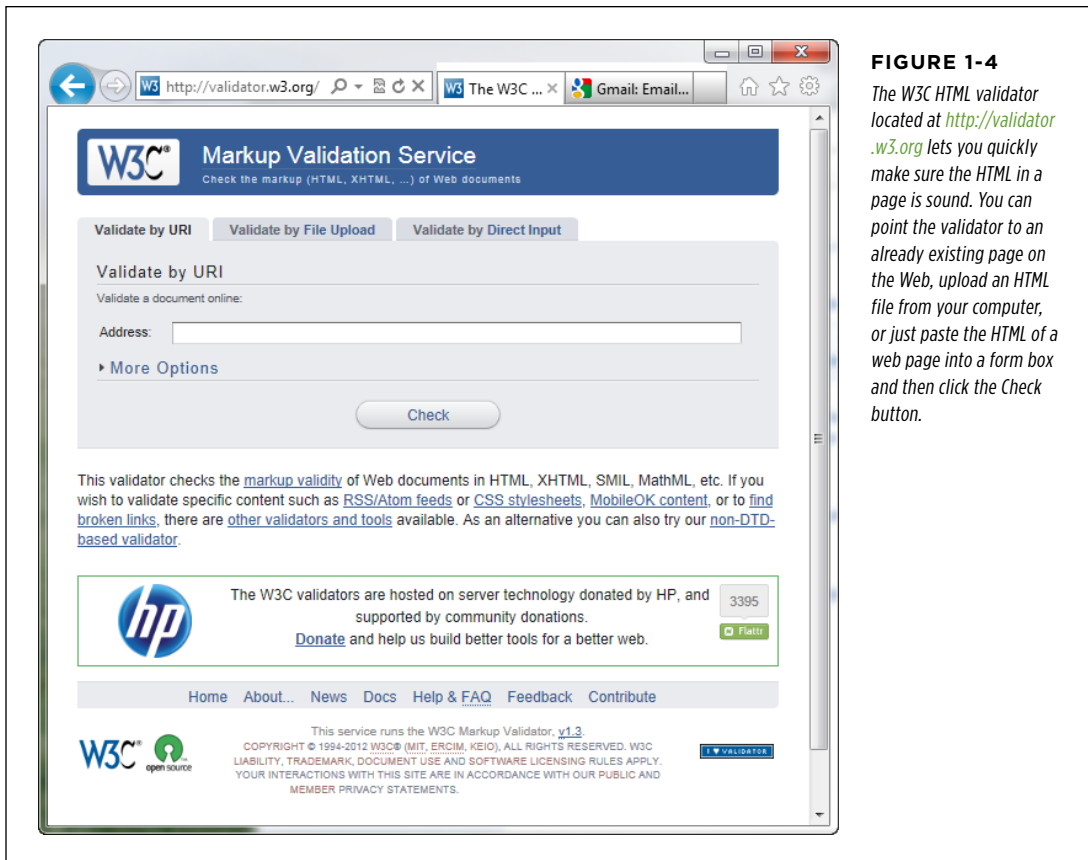
### Validate Your Web Pages

HTML follows certain rules: For example, the `<html>` tag wraps around the other tags on a page, and the `<title>` tag needs to appear within the `<head>` tag. XHTML provides an even more strict set of rules to follow, while HTML5 is a bit more lax. It's easy to forget these rules or simply make a typo. Incorrect (or *invalid*, as the geeks would say) HTML causes problems like making your page look different in different web browsers. More importantly, even valid CSS may not work as expected with invalid HTML. Fortunately, there are tools for checking whether the HTML in your web pages is correctly written.

The easiest way to check—that is, *validate*—your pages is on the W3C's website at <http://validator.w3.org> (see Figure 1-4). Get the Web Developer extension for Firefox (<http://chrispederick.com/work/web-developer>); it provides a quick way to test a page in the W3C validator.

The W3C, or World Wide Web Consortium, is the organization responsible for determining the standards for many of the technologies and languages of the Web, including HTML and CSS.

If the W3C validator finds any errors in your page, it tells you what those errors are. If you use Firefox, you can download an extension that lets you validate a web page directly in that browser, without having to visit the W3C site. It can even attempt to fix any problems it encounters. You can get the extension here: <http://users.skynet.be/mgueury/mozilla/>. Chrome users can download the HTML Tidy extension for Chrome from <http://bit.ly/SCONRY>. A similar tool is available for the Safari browser as well: [www.zappatic.net/safaritidy](http://www.zappatic.net/safaritidy).

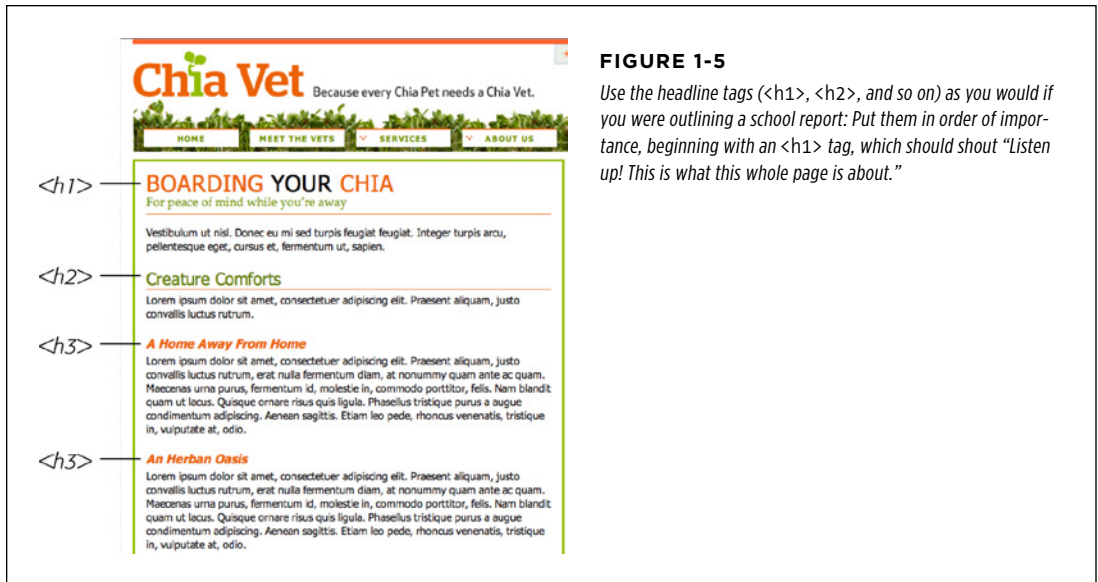


**FIGURE 1-4**  
The W3C HTML validator located at <http://validator.w3.org> lets you quickly make sure the HTML in a page is sound. You can point the validator to an already existing page on the Web, upload an HTML file from your computer, or just paste the HTML of a web page into a form box and then click the Check button.

## Tips to Guide Your Way

It's always good to have a map for getting the lay of the land. If you're still not sure how to use HTML to create well-structured web pages, then here are a few tips to get you started:

- Use headings to indicate the relative importance of text. Again, think outline. When two headings have equal importance in the topic of your page, use the same level header on both. If one is less important or a subtopic of the other, then use the next level header. For example, follow an `<h2>` with an `<h3>` tag (see Figure 1-5). In general, it's good to use headings in order and try not to skip heading numbers. For example, don't follow an `<h2>` tag with an `<h5>` tag.



**FIGURE 1-5**

Use the headline tags (<h1>, <h2>, and so on) as you would if you were outlining a school report: Put them in order of importance, beginning with an <h1> tag, which should shout “Listen up! This is what this whole page is about.”

- Use the **<p>** tag for paragraphs of text.
- Use unordered lists (**<ul>**) when you’ve got a list of several related items, such as navigation links, headlines, or a set of tips like these.
- Use numbered lists (**<ol>**) to indicate steps in a process or define the order of a set of items. The tutorials in this book are a good example, as is a list of rankings like “Top 10 websites popular with monks.”
- To create a glossary of terms and their definitions or descriptions, use the **<dl>** (definition list) tag in conjunction with the **<dt>** (definition term) and **<dd>** (definition description) tags. (For an example of how to use this combo, visit [www.w3schools.com/tags/tryit.asp?filename=tryhtml\\_list\\_definition](http://www.w3schools.com/tags/tryit.asp?filename=tryhtml_list_definition).)
- If you want to include a quotation like a snippet of text from another website, a movie review, or just some wise saying of your grandfather’s, try the **<blockquote>** tag for long passages or the **<q>** tag to place a short quote within a longer paragraph, like this:
 

```
<p>Mark Twain is said to have written <q>The The coldest winter I ever spent
was a summer in San Francisco</q>. Unfortunately, he never actually wrote that
famous quote.</p>
```
- Take advantage of obscure tags like the **<cite>** tag for referencing a book title, newspaper article, or website, and the **<address>** tag to identify and supply contact information for the author of a page (great for a copyright notice).

- As explained in full on page 26, steer clear of any tag or attribute aimed just at changing the appearance of a text or image. CSS, as you'll see, can do it all.
- When there just isn't an HTML tag that fits the bill, but you want to identify an element on a page or a bunch of elements on a page so you can apply a distinctive look, use the **<div>** and **<span>** tags (see page 21). You'll get more advice on how to use these in later chapters.
- Don't overuse **<div>** tags. Some web designers think all they need are **<div>** tags, ignoring tags that might be more appropriate. For example, to create a navigation bar, you could add a **<div>** tag to a page and fill it with a bunch of links. A better approach would be to use a bulleted list (**<ul>** tag). After all, a navigation bar is really just a list of links. As discussed on page 23, HTML5 provides several new tags that can take the place of the **<div>** tag, like the **<article>**, **<section>**, and **<footer>** tags. For a navigation bar, you could use the HTML5 **<nav>** tag.
- Remember to close tags. The opening **<p>** tag needs its partner in crime (the closing **</p>** tag), as do all other tags, except the few self-closers like **<br>** and **<img>** (**<br />** and **<img />** in XHTML).
- Validate your pages with the W3C validator (see Figure 1-4 and the box on page 27). Poorly written or typo-ridden HTML causes many weird browser errors.

## ■ The Importance of the Doctype

HTML follows certain rules—these rules are contained in a *Document Type Definition* file, otherwise known as a DTD. A DTD is a text file that explains what tags, attributes, and values are valid for a particular type of HTML. And for each version of HTML, there's a corresponding DTD. By now you may be asking, "But what's all this got to do with CSS?"

Everything—if you want your web pages to appear correctly and consistently in web browsers. You tell a web browser which version of HTML or XHTML you're using by including what's called a *doctype declaration* at the beginning of a web page. This doctype declaration is the first line in the HTML file, and defines what version of HTML you're using (such as HTML5 or HTML 4.01 Transitional). When you mistype the doctype declaration or leave it out, you can throw most browsers into an altered state called *quirks mode*.

Quirks mode is browser manufacturers' attempt to make their software behave like browsers did circa 1999 (in the Netscape 4 and Internet Explorer 5 days). If a modern browser encounters a page that's missing the correct doctype, then it thinks, "Gee, this page must have been written a long time ago, in an HTML editor far, far away. I'll pretend I'm a really old browser and display the page just as one of those

buggy old browsers would display it.” That’s why, without a correct doctype, your lovingly CSS-styled web pages may not look as they should, according to current standards. If you unwittingly view your web page in quirks mode when checking it in a browser, you may end up trying to fix display problems that are related to an incorrect doctype and not the incorrect use of HTML or CSS.

**NOTE**

For more (read: technical) information on quirks mode, visit [www.quirksmode.org/css/quirksmode.html](http://www.quirksmode.org/css/quirksmode.html) and [https://developer.mozilla.org/en/Mozilla%2527s\\_Quirks\\_Mode](https://developer.mozilla.org/en/Mozilla%2527s_Quirks_Mode).

Fortunately, it’s easy to get the doctype correct. All you need to know is what version of HTML you’re using. If you’re using HTML5, things are easy. The doctype is simply:

```
<!doctype html>
```

Put this at the top of your HTML file and you’re good to go. If you’re still using older versions of HTML or XHTML such as HTML 4.01 Transitional and XHTML 1.0 Transitional, then the doctype is a lot more convoluted.

If you’re using HTML 4.01 Transitional, for example, type the following doctype declaration at the very beginning of every page you create:

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

The doctype declaration for XHTML 1.0 Transitional is similar. It’s also necessary to add a little code to the opening `<html>` tag that’s used to identify the file’s XML type—in this case, it’s XHTML—like this:

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
```

If this entire discussion is making your head ache and your eyes slowly shut, keep your life simple by using the HTML5 doctype. It’s short, easy to remember, and works in all browsers. You can use this doctype even if you don’t touch any of the new HTML5 tags.

**NOTE**

Most visual web page tools like Dreamweaver and Expression Web automatically add a doctype declaration whenever you create a new web page, and many HTML-savvy text editors have shortcuts for adding doctypes.

## FREQUENTLY ASKED QUESTION

### Should I Care About IE 6, 7, or 8?

*I keep hearing that Internet Explorer 6 is dead and we don't need to worry about it anymore. Is this true? What about other versions of IE?*

If you're a web designer, you've probably got the latest version of Internet Explorer, Firefox, Safari, Chrome, or Opera on your computer. Previous versions of this book talked quite a bit about Internet Explorer 6 and even provided a section on how to deal with IE 6 bugs. That's all been removed, since IE 6 is rapidly disappearing.

If you're building sites with a US audience in mind, as of August 2012, less than .6 percent of US web surfers use that outdated browser; in the UK, it's 1.1%. The exact figure varies depending on whom you ask (for example, <http://gs.statcounter.com> has worldwide IE 6 usage at .57 percent and [www.ie6countdown.com](http://www.ie6countdown.com) says it's at 6 percent as of August 2012). In China, IE 6 is still widely used.

However, even statistics that include the geographic region of your site's audience don't truly reflect what visitors to your site use. If you build a site aimed at tech-savvy web designers, odds are that IE 6 hasn't been to your site in a long time. However, if your site's aimed at people in China, you may need to deal with IE 6. The best way to find out how much of your traffic comes via IE 6 is to look at your web server's log files or sign

up for Google Analytics ([www.google.com/analytics](http://www.google.com/analytics)) so you can track your visitors' browsers (among many other things).

For the great majority of web design projects, IE 6 is effectively gone. In addition, IE 7 is also rapidly disappearing, with just 1.16 percent worldwide market share as of August 2012 ([http://gs.statcounter.com/#browser\\_version-ww-monthly-201108-201208](http://gs.statcounter.com/#browser_version-ww-monthly-201108-201208)).

IE 8, however, you still need to contend with. Windows XP is still a popular operating system, and Microsoft chose not to bring Internet Explorer 9 or later to that operating system. So as long as XP is around, Internet Explorer 8 will be around, too. (To keep up with advances in web development, XP users can switch to Chrome or Firefox.) According to StatCounter, in August 2012, IE 8 accounted for 13.65 percent of all web browsers ([http://gs.statcounter.com/#browser\\_version-ww-monthly-201108-201208](http://gs.statcounter.com/#browser_version-ww-monthly-201108-201208)). According to NetMarketShare, IE 8 accounts for a whopping 26 percent of web browsers (<http://marketshare.hitslink.com/browser-market-share.aspx?qprid=2&qpcustomid=0>). Since IE 8 doesn't understand HTML5 tags, you can't directly format them with CSS either—you have to turn to a JavaScript workaround as described in the box on page 24.

## ■ Making Sure Internet Explorer Is Up-to-Date

Thanks to Microsoft's auto-update feature, Windows PCs now update Internet Explorer to the latest version. Windows 7 and 8 users have the latest version of Internet Explorer installed—IE 10 at the time of this writing. IE 10 supports many of the exciting and powerful new HTML5 and CSS3 properties. As you'll learn in this book, CSS3 provides many exciting design possibilities like drop shadows (page 210), gradient backgrounds (page 259), and rounded corners (page 207), to name a few.

Unfortunately, not all Windows users will be able to take advantage of these advancements in web design. As explained in the box on page 32, the widely used Windows XP operating system can run only Internet Explorer 8 or earlier. In fact, IE 8 is still the most common version of Internet Explorer used on the Web. This book will point out CSS properties that won't work in IE 8, as well as possible workarounds.

Because IE 8 is still very popular, you need to keep a couple of things in mind. IE 8 is sort of like a chameleon: It can take on the appearance of a different version. If you're not careful, it may not display your web pages the way you want it to. For example, and most importantly, you must include a proper doctype. As mentioned in the previous section, without a doctype, browsers switch into quirks mode. Well, when IE 8 goes into quirks mode, it tries to replicate the look of IE 5 (!?).

But wait—there's more! IE 8 can also pretend to be IE 7. When someone viewing your site in IE 8 clicks a "compatibility view" button, IE 8 goes into IE 7 mode, displaying pages without IE 8's full CSS 2.1 goodness. The same thing happens if Microsoft puts your website onto its Compatibility View List—a list of sites that Microsoft has determined look better in IE 7 than in IE 8. If you're designing a site using the guidelines in this book, you won't want IE 8 to act like IE 7...ever.

Fortunately, there's a way to tell IE 8 to stop all this nonsense and just be IE 8. Adding a single META tag to a web page instructs IE 8 to ignore Compatibility View and the Compatibility View List and always display the page by using its most standards-compliant mode:

```
<meta http-equiv="X-UA-Compatible" content="IE=edge" />
```

Put this instruction in the page's <head> section (below the <title> tag is a good place). This tag works for current versions of IE, too: The "IE=edge" part of the tag will instruct versions of Internet Explorer later than IE 8 to also display web pages in their standard mode. Unfortunately, you must do this on every page of your site.

Now that your HTML ship is steering in the right direction, it's time to jump into the fun stuff (and the reason you bought this book): Cascading Style Sheets.



## UP TO SPEED

### Cross-Browser Testing

There are a lot of web browsers out there. If you use Windows, you automatically get Internet Explorer and can install additional browsers like Firefox, Safari, Opera, or Google's Chrome. On a Mac, you can stick with the already-installed Safari browser, or run Firefox or Chrome. While the very latest browsers are mostly comparable when displaying CSS-driven web pages, you can't say the same for IE 8, which is still in widespread use.

To really make sure your sites work for the largest audience, you need a way to test your designs in as wide a range of browsers as possible. Here are a few techniques:

**Windows.** Normally, Windows computers can only run one version of IE—you can't test in IE 6, IE 7, and IE 8 on the same computer. Well, normally you can't, but *you* can if you run *virtual machines* on your Windows computer to create completely separate versions of Internet Explorer that you can then test in. It's a fair amount of work, but it gives you the most accurate way of testing a page in multiple versions of IE. Visit [www.howtogeek.com/102261/how-to-run-internet-explorer-7-8-and-9-at-the-same-time-using-virtual-machines](http://www.howtogeek.com/102261/how-to-run-internet-explorer-7-8-and-9-at-the-same-time-using-virtual-machines) to learn more about how to do this.

You should also install the other major browsers on your computer: Firefox, Opera, and Chrome. Apple has discontinued Safari for Windows, so you can simply rely on Chrome (which has a similar underlying rendering engine) or use one of the services listed later in this box (like Adobe Browserlab) to test your designs in Safari.

**Mac.** Testing is a bit trickier for Mac people. You *have* to test in Internet Explorer—it's still one of the most used browsers in the world, and the lack of most CSS3 properties in IE 8 means that your painstakingly crafted design might look great for you, but fall apart in IE 8. You have a few options: First, you can buy (or borrow) a Windows machine; second, if you have an Intel Mac, you can install Windows using Apple's Boot Camp software ([www.apple.com/macosx/features/bootcamp.html](http://www.apple.com/macosx/features/bootcamp.html)); a third option is to install virtualization software like VMware Fusion or Parallels Desktop on your Mac. This software lets you run a virtual Windows machine side-by-side with the Mac OS. You can jump between Mac and Windows to test in various browsers in

both operating systems. It's the most efficient testing technique for Mac users. Both Boot Camp and virtualization software require a copy of the Windows OS.

**Everyone.** Another option that works for both Windows and Macs and doesn't require installing extra software is one of the many cross-browser testing services that let you see what your pages look like in many different operating systems and browsers.

- **CrossBrowserTesting.com** (<http://CrossBrowserTesting.com>) is \$30 a month (ouch) for 150 minutes of use and offers an added benefit—interactive testing. You get to see your page running remotely on a PC under your control—you can test features that a screenshot can't capture, like Flash movie playback, animation, and JavaScript interactions.
- **BrowserStack** ([www.browserstack.com](http://www.browserstack.com)) is another service that lets you test your pages interactively via your web browser. For \$19 a month, it's like renting multiple Windows and Mac machines with multiple versions of IE, Chrome, Firefox, and Safari installed.
- **Browsershots** ([www.browsershots.org](http://www.browsershots.org)) is a free alternative that provides screenshots for a wide range of browsers on Windows and Linux.
- You can also see what a page looks like in IE 9, 8 7, 6, and even 5.5 for free with **NetRenderer** (<http://ipinfo.info/netrenderer>), though. You need to have the pages you want to test up on a publicly accessible website. Visit <http://ipinfo.info/netrenderer>, type the URL of the page, and in a few moments you'll see a screen shot. Unfortunately, it doesn't take a picture of a complete web page—just the top part that you'd normally see on a screen. On the other hand, it's free.
- Finally, you can use Adobe's **BrowserLab** tool (<https://browserlab.adobe.com>) to get snapshots of a web page in all major browsers. If you use Dreamweaver to build sites, for example, you can launch BrowserLab from within the program and preview the page you're currently working on.

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



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

Learn more at [ebooks.oreilly.com](http://ebooks.oreilly.com)

You can also purchase O'Reilly ebooks through the iBookstore, the [Android Marketplace](http://AndroidMarketplace.com), and [Amazon.com](http://Amazon.com).

# O'REILLY®

Spreading the knowledge of innovators

[oreilly.com](http://oreilly.com)