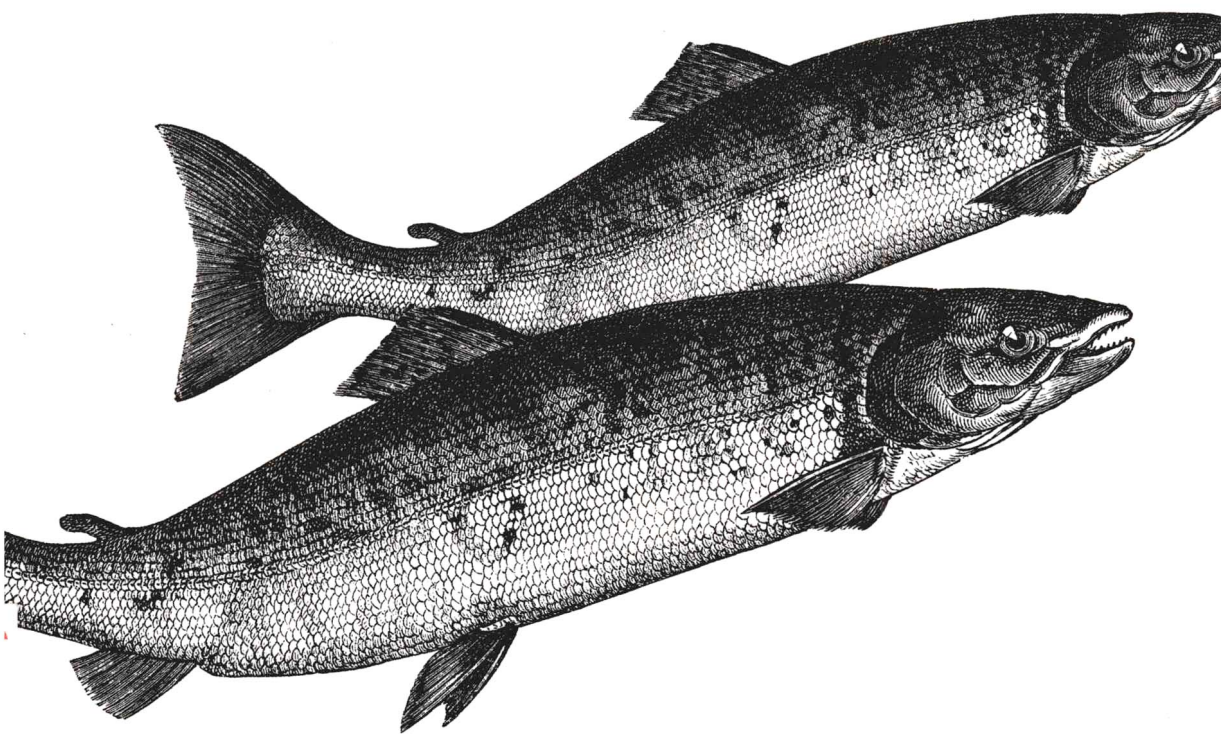
CSS权威指南（影印版）

3rd Edi

CSS

The Definitive Guide



O'REILLY®
東南大學出版社

Eric A. Meyer

第三版

CSS权威指南(影印版)

CSS: The Definitive Guide

O'REILLY®

Beijing • Cambridge • Farnham • Köln • Paris • Sebastopol • Taipei • Tokyo

O'Reilly Media, Inc.授权东南大学出版社出版

东南大学出版社

图书在版编目（CIP）数据

CSS 权威指南：第 3 版：英文 /（美）迈耶（Meyer, E.A.）著. — 影印本. — 南京：东南大学出版社，2007.6

书名原文：CSS: The Definitive Guide, Third Edition

ISBN 978-7-5641-0777-2

Ⅰ. C... Ⅱ. 迈... Ⅲ. 主页制作－软件工具，CSS －英文 Ⅳ. TP393.092

中国版本图书馆 CIP 数据核字（2007）第 072071 号

江苏省版权局著作权合同登记

图字：10-2007-093 号

书　　名 / CSS 权威指南 第三版（影印版）

责任编辑 / 张烨

封面设计 / Edie Freedman，张健

出版发行 / 东南大学出版社（press.seu.edu.cn）

地　　址 / 南京四牌楼 2 号（邮政编码 210096）

印　　刷 / 扬中市印刷有限公司

开　　本 / 787 毫米 × 980 毫米　16 开本　33.75 印张

版　　次 / 2007 年 6 月第 1 版　2007 年 6 月第 1 次印刷

印　　数 / 0001-3500 册

书　　号 / ISBN 978-7-5641-0777-2/TP · 125

定　　价 / 68.00 元（册）

O'Reilly Media, Inc.介绍

O'Reilly Media, Inc.是世界上在 UNIX、X、Internet 和其他开放系统图书领域具有领导地位的出版公司，同时是联机出版的先锋。

从最畅销的《The Whole Internet User's Guide & Catalog》（被纽约公共图书馆评为二十世纪最重要的50本书之一）到GNN（最早的Internet门户和商业网站），再到WebSite（第一个桌面PC的Web服务器软件），O'Reilly Media, Inc.一直处于Internet发展的最前沿。

许多书店的反馈表明，O'Reilly Media, Inc.是最稳定的计算机图书出版商——每一本书都一版再版。与大多数计算机图书出版商相比，O'Reilly Media, Inc.具有深厚的计算机专业背景，这使得O'Reilly Media, Inc.形成了一个非常不同于其他出版商的出版方针。O'Reilly Media, Inc.所有的编辑人员以前都是程序员，或者是顶尖级的技术专家。O'Reilly Media, Inc.还有许多固定的作者群体——他们本身是相关领域的技术专家、咨询专家，而现在编写著作，O'Reilly Media, Inc.依靠他们及时地推出图书。因为O'Reilly Media, Inc.紧密地与计算机业界联系着，所以O'Reilly Media, Inc.知道市场上真正需要什么图书。

出版说明

随着计算机技术的成熟和广泛应用，人类正在步入一个技术迅猛发展的新时期。计算机技术的发展给人们的工业生产、商业活动和日常生活都带来了巨大的影响。然而，计算机领域的技术更新速度之快也是众所周知的，为了帮助国内技术人员在第一时间了解国外最新的技术，东南大学出版社和美国 O'Reilly Meida, Inc.达成协议，将陆续引进该公司的代表前沿技术或者在某专项领域享有盛名的著作，以影印版或者简体中文版的形式呈献给读者。其中，影印版书籍力求与国外图书“同步”出版，并且“原汁原味”展现给读者。

我们真诚地希望，所引进的书籍能对国内相关行业的技术人员、科研机构的研究人员和高校师生的学习和工作有所帮助，对国内计算机技术的发展有所促进。也衷心期望读者提出宝贵的意见和建议。

最新出版的影印版图书，包括：

- 《深入浅出面向对象分析与设计》（影印版）
- 《Ajax on Rails》（影印版）
- 《Java 与 XML 第三版》（影印版）
- 《学习 MySQL》（影印版）
- 《Linux Kernel 技术手册》（影印版）
- 《Dynamic HTML 权威参考 第三版》（影印版）
- 《ActionScript 3.0 Cookbook》（影印版）
- 《CSS:The Missing Manual》（影印版）
- 《Linux 技术手册 第五版》（影印版）
- 《Ajax on Java》（影印版）
- 《WCF Service 编程》（影印版）
- 《JavaScript 权威指南 第五版》（影印版）
- 《CSS 权威指南 第三版》（影印版）
- 《嵌入式系统编程 第二版》（影印版）
- 《学习 JavaScript》（影印版）
- 《Rails Cookbook》（影印版）

Preface

If you are a web designer or document author interested in sophisticated page styling, improved accessibility, and saving time and effort, this book is for you. All you really need before starting the book is a decent knowledge of HTML 4.0. The better you know HTML, of course, the better prepared you'll be. You will need to know very little else to follow this book.

This third edition of *CSS: The Definitive Guide* covers CSS2 and CSS2.1 (up through the 11 April 2006 Working Draft), the latter of which is, in many ways, a clarification of the first. While some CSS3 modules have reached Candidate Recommendation status as of this writing, I have chosen not to cover them in this edition (with the exception of some CSS3 selectors). I made this decision because implementation of these modules is still incomplete or nonexistent. I feel it's important to keep the book focused on currently supported and well-understood levels of CSS, and to leave any future capabilities for future editions.

Conventions Used in This Book

The following typographical conventions are used in this book:

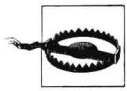
Italic
: Indicates new terms, URLs, variables in text, user-defined files and directories, commands, file extensions, filenames, directory or folder names, and UNC pathnames.

`Constant width`
: Indicates command-line computer output, code examples, Registry keys, and keyboard accelerators.

`Constant width bold`
: Indicates user input in examples.

`Constant width italic`
: Indicates variables in examples and in Registry keys. It is also used to indicate variables or user-defined elements within italic text (such as pathnames or filenames). For instance, in the path *\Windows\username*, replace username with your name.

This icon signifies a tip, suggestion, or general note.



This icon indicates a warning or caution.

Property Conventions

Throughout this book, there are boxes that break down a given CSS property. These have been reproduced practically verbatim from the CSS specifications, but some explanation of the syntax is in order.

Throughout, the allowed values for each property are listed with the following syntax:

Value: [<length> | `thick` | `thin`]{1,4}
Value: [<family-name> ,]* <family-name>
Value: <url>? <color> [/ <color>]?
Value: <url> || <color>

Any words between "<" and ">" give a type of value or a reference to another property. For example, the property `font` will accept values that actually belong to the property `font-family`. This is denoted by the text <font-family>. Any words presented in `constant width` are keywords that must appear literally, without quotes. The forward slash (/) and the comma (,) must also be used literally.

Several keywords strung together means that all of them must occur in the given order. For example, `help me` means that the property must use those keywords in that exact order.

If a vertical bar separates alternatives (X | Y), then any one of them must occur. A vertical double bar (X || Y) means that X, Y, or both must occur, but they may appear in any order. Brackets ([...]) are for grouping things together. Juxtaposition is stronger than the double bar, and the double bar is stronger than the bar. Thus "V W | X || Y Z" is equivalent to "[V W] | [X || [Y Z]]".

Every word or bracketed group may be followed by one of the following modifiers:

- An asterisk (*) indicates that the preceding value or bracketed group is repeated zero or more times. Thus, `bucket*` means that the word `bucket` can be used any number of times, including zero. There is no upper limit defined on the number of times it can be used.
- A plus (+) indicates that the preceding value or bracketed group is repeated one or more times. Thus, `mop+` means that the word `mop` must be used at least once, and potentially many more times.

- A question mark (?) indicates that the preceding value or bracketed group is optional. For example, `[pine tree]?` means that the words `pine tree` need not be used (although they must appear in that exact order if they are used).
- A pair of numbers in curly braces (`{M,N}`) indicates that the preceding value or bracketed group is repeated at least `M` and at most `N` times. For example, `ha{1,3}` means that there can be one, two, or three instances of the word `ha`.

Some examples follow:

`give || me || liberty`
: At least one of the three words must be used, and they can be used in any order. For example, `give liberty`, `give me`, `liberty me give`, and `give me liberty` are all valid.

`[ I | am ]? the || walrus`
: Either the word `I` or `am` may be used, but not both, and use of either is optional. In addition, either `the` or `walrus`, or both, must follow in any order. Thus, you could construct `I the walrus`, `am walrus the`, `am the`, `I walrus`, `walrus the`, and so forth.

`koo+ ka-choo`
: One or more instances of `koo` must be followed by `ka-choo`. Therefore, `koo koo ka-choo`, `koo koo koo ka-choo`, and `koo ka-choo` are all legal. The number of `koo`s is potentially infinite, although there are bound to be implementation-specific limits.

`I really{1,4}? [love | hate] [Microsoft | Netscape | Opera | Safari]`
: This is the all-purpose web designer's opinion expresser. This example can be interpreted as `I love Netscape`, `I really love Microsoft`, and similar expressions. Anywhere from zero to four `really`s may be used. You also get to pick between `love` and `hate`, even though only `love` was shown in this example.

`[[Alpha || Baker || Cray],]{2,3} and Delphi`
: This is a potentially long and complicated expression. One possible result would be `Alpha, Cray, and Delphi`. The comma is placed because of its position within the nested bracket groups.

Using Code Examples

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

We appreciate, but do not require, attribution. An attribution usually includes the title, author, publisher, and ISBN. For example: "*CSS: The Definitive Guide*, Third Edition, by Eric A. Meyer. Copyright 2007 O'Reilly Media, Inc., 978-0-596-52733-4."

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

How to Contact Us

We at O'Reilly have tested and verified the information in this book to the best of our ability, but you may find that features have changed (or even that we have made mistakes!). Please let us know about any errors you find, as well as your suggestions for future editions, by writing to:

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

There is a web page for this book, which lists errata, examples, or any additional information. You can access this page at:

http://www.oreilly.com/catalog/csstdg3

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

bookquestions@oreilly.com

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

http://www.oreilly.com

Safari® Enabled

When you see a Safari® Enabled icon on the cover of your favorite technology book, that means the book is available online through the O'Reilly Network Safari Bookshelf.

Safari offers a solution that's better than e-books. It's a virtual library that lets you easily search thousands of top tech books, cut and paste code samples, download chapters, and find quick answers when you need the most accurate, current information. Try it for free at *http://safari.oreilly.com*.

Acknowledgments

I'd like to take a moment to thank the people who have backed me up during the long process of getting this book to its readers.

First, I'd like to thank everyone at O'Reilly for all they've done over the years, giving me my break into publishing and continuing to give me the opportunity to produce a book that matters. For this third edition, I'd like to thank Tatiana Apandi for her good humor, patience, and understanding as I played chicken with my deadlines.

I'd also like to thank most profoundly my technical reviewers. For the first edition, that was David Baron and Ian Hickson, with additional input from Bert Bos and Håkon Lie. The second edition was reviewed by Tantek Çelik and Ian Hickson. The fine folks who performed technical review on the third edition, the one you hold in your hands, were Darrell Austin, Liza Daly, and Neil Lee. All lent their considerable expertise and insight, keeping me honest and up-to-date on the latest changes in CSS as well as taking me to task for sloppy descriptions and muddled explanations. None of the editions, least of all this one, could have been as good as it is without their collective efforts, but of course whatever errors you find in the text are my fault, not theirs. That's kind of a cliché, I know, but it's true nonetheless.

Similarly, I'd like to thank everyone who pointed out errata that needed to be addressed. I may not have always been good about sending back email right away, but I read all of your questions and concerns and, when needed, made corrections. The continued feedback and constructive criticism will only help the book get better, as it always has.

There are a few personal acknowledgments to make as well.

To the staff of WRUW, 91.1 FM Cleveland, thank you for nine years of support, great music, and straight-out fun. Maybe one day I'll bring Big Band back to your airwaves, and maybe not; but either way, keep on keepin' on.

To Jeffrey Zeldman, thanks for being a great colleague and partner; and to the whole Zeldman family, thanks for being such wonderful friends.

To "Auntie" Molly, thanks for always being who you are.

To "Uncle" Jim, thanks for everything, both professionally and personally. It's no exaggeration to say I wouldn't be where I am without your influence, and our lives would be a good deal poorer without you around.

To the Bread and Soup Crew—Jim, Genevieve, Jim, Gini, Ferrett, Jen, Jenn, and Molly—thanks for all your superb cooking and tasty conversation.

To my extended family, thank you as always for your love and support.

To anyone I should have thanked, but didn't: my apologies. And my thanks.

And to my wife and daughter, more thanks than I can ever express for making my days richer than I have any right to expect, and for showering me with more love than I could ever hope to repay. Though I'll keep trying, of course.

—Eric A. Meyer
Cleveland Heights, Ohio
1 August 2006

Table of Contents

CHAPTER 1

CSS and Documents

Cascading Style Sheets (CSS) are a powerful way to affect the presentation of a document or a collection of documents. Obviously, CSS is basically useless without a document of some sort, since it would have no content to present. Of course, the definition of "document" is extremely broad. For example, Mozilla and related browsers use CSS to affect the presentation of the browser chrome itself. Still, without the content of the chrome—buttons, address inputs, dialog boxes, windows, and so on—there would be no need for CSS (or any other presentational information).

The Web's Fall from Grace

Back in the dimly remembered, early years of the Web (1990–1993), HTML was a fairly lean language. It was composed almost entirely of structural elements that were useful for describing things like paragraphs, hyperlinks, lists, and headings. It had nothing even remotely approaching tables, frames, or the complex markup we assume is necessary to create web pages. HTML was originally intended to be a structural markup language, used to describe the various parts of a document; very little was said about how those parts should be displayed. The language wasn't concerned with appearance—it was just a clean little markup scheme.

Then came Mosaic.

Suddenly, the power of the World Wide Web was obvious to almost anyone who spent more than 10 minutes playing with it. Jumping from one document to another was no more difficult than pointing the cursor at a specially colored bit of text, or even an image, and clicking the mouse. Even better, text and images could be displayed together, and all you needed to create a page was a plain-text editor. It was free, it was open, and it was cool.

Web sites began to spring up everywhere. There were personal journals, university sites, corporate sites, and more. As the number of sites increased, so did the demand for new HTML elements that would each perform a specific function. Authors started demanding that they be able to make text boldfaced or italicized.

At the time, HTML wasn't equipped to handle those sorts of desires. You could declare a bit of text to be emphasized, but that wasn't necessarily the same as being italicized—it could be boldfaced instead, or even normal text with a different color, depending on the user's browser and preferences. There was nothing to ensure that what the author created was what the reader would see.

As a result of these pressures, markup elements like `<FONT>` and `<BIG>` started to creep into the language. Suddenly, a structural language started to become presentational.

What a Mess

Years later, we have inherited the problems of this haphazard process. Large parts of HTML 3.2 and HTML 4.0, for example, were devoted to presentational considerations. The ability to color and size text through the `font` element, to apply background colors and images to documents and tables, to use `table` attributes (such as `cellspacing`), and to make text blink on and off are all the legacy of the original cries for "more control!"

For an example of the mess in action, take a quick glance at almost any corporate web site's markup. The sheer amount of markup in comparison to actual useful information is astonishing. Even worse, for most sites, the markup is almost entirely comprised of tables and `font` elements, neither of which conveys any real semantic meaning as to what's being presented. From a structural standpoint, these pages are little better than random strings of letters.

For example, let's assume that for page titles, an author uses `font` elements instead of heading elements like `h1`:

```
<font size="+3" face="Helvetica" color="red">Page Title</font>
```

Structurally speaking, the `font` tag has no meaning. This makes the document far less useful. What good is a `font` tag to a speech-synthesis browser, for example? If an author uses heading elements instead of `font` elements, though, the speaking browser can use a certain speaking style to read the text. With the `font` tag, the browser has no way to know that the text is any different from other text.

Why do authors run roughshod over structure and meaning this way? Because they want readers to see the page as they designed it. To use structural HTML markup is to give up a lot of control over a page's appearance, and it certainly doesn't allow for the kind of densely packed page designs that have become so popular over the years. But consider the following problems with such an approach:

- Unstructured pages make content indexing inordinately difficult. A truly powerful search engine would allow users to search only page titles, or only section headings within pages, or only paragraph text, or perhaps only those paragraphs that are marked as important. To accomplish such a feat, however, the page contents must be contained within some sort of structural markup—exactly the sort of markup most pages lack. Google, for example, does pay attention to markup structure when indexing pages, so a structural page will increase your Google rank.

- Lack of structure reduces accessibility. Imagine that you are blind and rely on a speech-synthesis browser to search the Web. Which would you prefer: a structured page that lets your browser read only section headings so that you can choose which section you'd like to hear more about; or a page that is so lacking in structure that your browser is forced to read the entire thing with no indication of what's a heading, what's a paragraph, and what's important? Let's return to Google—the search engine is, in effect, the world's most active blind user, with millions of friends who accept its every suggestion about where to surf and shop.
- Advanced page presentation is possible only with some sort of document structure. Imagine a page in which only the section headings are shown, with an arrow next to each. The user can decide which section heading applies to him and click on it, thus revealing the text of that section.
- Structured markup is easier to maintain. How many times have you spent several minutes hunting through someone else's HTML (or even your own) in search of the one little error that's messing up your page in one browser or another? How much time have you spent writing nested tables and `font` elements, only to get a sidebar with white hyperlinks in it? How many linebreak elements have you inserted trying to get exactly the right separation between a title and the following text? By using structural markup, you can clean up your code and make it easier to find what you're looking for.

Granted, a fully structured document is a little plain. Due to that one single fact, a hundred arguments in favor of structural markup won't sway a marketing department from using the type of HTML that was so prevalent at the end of the 20th century, and which persists even today. What we need is a way to combine structural markup with attractive page presentation.

CSS to the Rescue

Of course, the problem of polluting HTML with presentational markup was not lost on the World Wide Web Consortium (W3C), which began searching for a quick solution. In 1995, the consortium started publicizing a work-in-progress called CSS. By 1996, it had become a full Recommendation, with the same weight as HTML itself. Here's why.

Rich Styling

In the first place, CSS allows for much richer document appearances than HTML ever allowed, even at the height of its presentational fervor. CSS lets you set colors on text and in the background of any element; permits the creation of borders around any element, as well as the increase or decrease of the space around them; lets you change the way text is capitalized, decorated (e.g., underlining), spaced, and even whether it is displayed at all; and allows you to accomplish many other effects.

Take, for example, the first (and main) heading on a page, which is usually the title of the page itself. The proper markup is:

```
<h1>Leaping Above The Water</h1>
```

Now, suppose you want this title to be dark red, use a certain font, be italicized and underlined, and have a yellow background. To do all of that with HTML, you'd have to put the h1 into a table and load it up with a ton of other elements like font and U. With CSS, all you need is one rule:

```
h1 {color: maroon; font: italic 2em Times, serif; text-decoration: underline;
    background: yellow;}
```

That's it. As you can see, everything you did in HTML can be done in CSS. There's no need to confine yourself to only those things HTML can do, however:

```
h1 {color: maroon; font: italic 2em Times, serif; text-decoration: underline;
    background: yellow url(titlebg.png) repeat-x;
    border: 1px solid red; margin-bottom: 0; padding: 5px;}
```

You now have an image in the background of the h1 that is only repeated horizontally, and a border around it, separated from the text by at least five pixels. You've also removed the margin (blank space) from the bottom of the element. These are feats that HTML can't even come close to matching—and that's just a taste of what CSS can do.

Ease of Use

If the depth of CSS doesn't convince you, then perhaps this will: style sheets can drastically reduce a web author's workload.

First, style sheets centralize the commands for certain visual effects in one handy place, instead of scattering them throughout the document. As an example, let's say you want all of the h2 headings in a document to be purple. Using HTML, the way to do this would be to put a font tag in every heading, like so:

```
<h2><font color="purple">This is purple!</font></h2>
```

This must be done for every heading of level two. If you have 40 headings in your document, you have to insert 40 font elements throughout, one for each heading! That's a lot of work for one little effect.

Let's assume that you've gone ahead and put in all those font elements. You're done, you're happy—and then you decide (or your boss decides for you) that those h2 headings should really be dark green, not purple. Now you have to go back and fix every single one of those font elements. Sure, you might be able to find-and-replace, as long as headings are the only purple text in your document. If you've put other purple font elements in your document, then you *can't* find-and-replace because you'd affect those, too.