Covers CSS 2.1

sitepoint

The CSS Anthology

101 Essential Tips, Tricks & Hacks

Practical Solutions to Common Problems

By Rachel Andrew

The CSS Anthology
101 Essential Tips, Tricks & Hacks
(First 4 Chapters)

Thank you for downloading the first four chapters of Rachel Andrew's book, *The CSS Anthology: 101 Essential Tips, Tricks & Hacks*, published by SitePoint.

This excerpt includes the Summary of Contents, Information about the Author, Editors and SitePoint, Table of Contents, Preface, the first four chapters of the book and the index.

We hope you find this information useful in evaluating this book.

Summary of Contents of this Excerpt

Summary of Additional Book Contents

The CSS Anthology
101 Essential Tips, Tricks & Hacks

by Rachel Andrew

The CSS Anthology: 101 Essential Tips, Tricks & Hacks

by Rachel Andrew

Editor: Georgina Laidlaw
Managing Editor: Simon Mackie
Expert Reviewer: Simon Willison
Technical Director: Kevin Yank
Printing History:
First Edition: November 2004

Index Editor: Bill Johncocks
Cover Designer: Julian Carroll
Cover Illustrator: Lucas Licata

Notice of Rights

Notice of Liability

The author and publisher have made every effort to ensure the accuracy of the information herein. However, the information contained in this kit is sold without warranty, either express or implied. Neither the authors and SitePoint Pty. Ltd., nor its dealers or distributors will be held liable for any damages to be caused either directly or indirectly by the instructions contained in this kit, or by the software or hardware products described herein.

Trademark Notice

Rather than indicating every occurrence of a trademarked name as such, this book uses the names only in an editorial fashion and to the benefit of the trademark owner with no intention of infringement of the trademark.

Published by SitePoint Pty. Ltd.

424 Smith Street Collingwood
VIC Australia 3066.
Web: www.sitepoint.com
Email: business@sitepoint.com

ISBN 0–9579218–8–8
Printed and bound in the United States of America

About The Author

Rachel Andrew is Web developer and director of Web solutions provider edgeofmyseat.com. When not writing code, she writes *about* writing code and is the coauthor of several books promoting the practical usage of Web standards alongside other everyday tools and technologies. Rachel takes a common sense, real world approach to Web standards, with her writing and teaching being based on the experiences she has in her own company every day.

Rachel lives in the UK with her partner Drew and daughter Bethany. When not working, they can often be found wandering around the English countryside hunting for geocaches and nice pubs that serve Sunday lunch and a good beer.

About SitePoint

SitePoint specializes in publishing fun, practical, and easy-to-understand content for Web professionals. Visit http://www.sitepoint.com/ to access our books, newsletters, articles and community forums.

For Bethany

Table of Contents

Preface

When I'm not writing books like this one, I'm writing code. I make my living by building Websites and applications, as, I'm sure, will many readers of this book. I use CSS to get jobs done every day. And I know what it's like to struggle to get something to work when the project needs to be finished the next morning.

When I talk to designers and developers who don't use CSS, or use CSS only for simple text styling, one thing that I hear over and over again is that they just don't have time to learn this whole new way of doing things. After all, tables and spacer GIFs work, they get the job done, and they pay the bills.

I was lucky. I picked up CSS very early in the piece, and started to play with it because it interested me. As a result of that early interest, my knowledge grew as the CSS techniques themselves were developed, and I can now draw on three years' experience building CSS layouts every time I tackle a project.

This book is my attempt to pass on the tricks and techniques that allow me to quickly and easily develop Websites and applications using CSS.

You won't find pages and pages of theory in this book. What you will find are solutions that will enable you to do the cool stuff today, but which should also act as a starting point for your own creativity. In my experience, it's far easier to learn by doing than by reading, so while you can use this book to find solutions that will help you get that client Website up and running by the deadline, please do experiment with these examples and use them as a way to learn new techniques.

The book was designed to let you quickly find the answer to the particular CSS problem with which you're struggling at any given point in time. You don't need to read it from cover to cover—just grab the technique that you need, or that interests you, and you're set to go. Along with each solution, I've provided an explanation to help you to understand why the technique works. This knowledge will allow you to expand on, and experiment with the technique in your own time.

I hope you enjoy this book! It has been great fun to write, and my hope is that it will be useful as a day-to-day reference, as well as a tool that helps give you the confidence to explore new CSS techniques.

Who Should Read This Book?

This book is aimed at people who need to work with CSS—Web designers and developers who have seen the cool CSS designs out there, but don't have the time to wade through masses of theory and debate in order to create a site. Each problem is solved with a working solution that can be implemented as-is or used as a starting point.

This book isn't a tutorial; while Chapter 1 covers the very basics of CSS, and the early chapters cover simpler techniques than those that follow, you will find the examples easier to grasp if you have a basic grounding in CSS.

What's Covered in This Book?

Chapter 1: ***Getting Started with CSS***
This chapter does not follow the same format as the rest of the book—it's simply a quick CSS tutorial for anyone who needs to brush up on the basics of CSS. If you've been using CSS in your own projects, you might want to skip this chapter and refer back to it on a needs basis, if you find you want to look into basic concepts in more detail.

Chapter 2: ***Text Styling and Other Basics***
This chapter covers techniques for styling and formatting text in your documents; font sizing, colors, and the removal of annoying extra white space around page elements are explained as the chapter progresses. Even if you're already using CSS for text styling, you will find some useful tips here.

Chapter 3: ***CSS and Images***
Combining CSS and images can create powerful visual effects. This chapter looks at the ways in which you can do this, and covers background images (not just on the body), and positioning text with images, among other topics.

Chapter 4: ***Navigation***
We all need navigation, and this chapter explains how to do it, CSS-style. The questions of CSS replacements for image-based navigation, CSS "tab" navigation, combining background images with CSS text to create attractive and accessible menus, and using lists to structure navigation in an accessible way are addressed in this chapter.

Chapter 5: ***Tabular Data***
While the use of tables for layout is to be avoided wherever possible, tables should be used for their real purpose: the display of tabular data, such as that contained in a spreadsheet. This chapter will demonstrate techniques for the application of tables to create attractive and usable tabular data displays.

Chapter 6: ***Forms and User Interfaces***
Whether you're a designer or a developer, it's likely that you'll spend a fair amount of time creating forms for data entry. CSS can help you to create forms that are attractive and more usable; this chapter shows how we can do that while bearing the key accessibility principles in mind.

Chapter 7: ***Browser and Device Support***
How can we deal with older browsers, browsers with CSS bugs, and alternate devices? These questions form the main theme of this chapter. We'll also see how to troubleshoot CSS bugs—and where to go for help—and discuss the ways you can test your site in as many browsers as possible.

Chapter 8: ***CSS Positioning and Layout***
In this chapter, we explore the use of CSS to create beautiful and accessible pages. We cover a range of different CSS layouts, and a variety of techniques, which can be combined and extended upon to create a range of different page layouts.

Chapter 9: ***Experimentation, Browser Specific CSS, and Future Techniques***
Through the text so far, I've presented techniques that will work well cross-browser, and don't involve rafts of CSS "hacks" in order to work. But, in this chapter, we look at newer techniques that don't work so well cross-browser, or that need quite a lot of extra effort on your part, in order to do so.

The Book's Website

Located at http://www.sitepoint.com/books/cssant1/, the Website that supports this book will give you access to the following facilities:

The Code Archive

As you progress through this book, you'll note file names above most of the code listings. These refer to files in the code archive, a downloadable ZIP archive that contains all of the finished examples presented in this book. Simply click the Code Archive link on the book's Website to download it.

Updates and Errata

No book is error-free, and attentive readers will no doubt spot at least one or two mistakes in this one. The Errata page on the book's Website will provide the latest information about known typographical and code errors, and will offer necessary updates for new releases of browsers and related standards.

The SitePoint Forums

If you'd like to communicate with me or anyone else on the SitePoint publishing team about this book, you should join SitePoint's online community.[2] The CSS forum,[3] in particular, offers an abundance of information above and beyond the solutions in this book.

In fact, you should join that community even if you *don't* want to talk to us. There are a lot of fun and experienced Web designers and developers hanging out there. It's a good way to learn new stuff, get questions answered in a hurry, and just have a good time.

The SitePoint Newsletters

In addition to books like this one, SitePoint publishes free email newsletters including *The SitePoint Tribune*, *The SitePoint Tech Times*, and *The SitePoint Design View*. Reading them will keep you up to date on the latest news, product releases, trends, tips, and techniques for all aspects of Web development. If nothing else, you'll get useful CSS articles and tips. If you're interested in learning other technologies, you'll find them especially valuable. Sign up to one or more SitePoint newsletters at http://www.sitepoint.com/newsletter/.

Your Feedback

If you can't find your answer through the forums, or if you wish to contact us for any other reason, the best place to write is `books@sitepoint.com`. We have an email support system set up to track your inquiries. If our support staff members can't answer your question, they'll send it straight to me. Suggestions

[2] http://www.sitepointforums.com/
[3] http://www.sitepoint.com/forums/forumdisplay.php?f=53

for improvements as well as notices of any mistakes you may find are especially welcome.

Acknowledgements

Firstly, I'd like to thank the SitePoint team for making this book a reality, and for being easy to communicate with despite the fact that our respective time zones saw me going to bed as they started work each day. Particular thanks must go to Simon Mackie, whose encouragement throughout the writing process was a great support.

Thanks also to Technical Editor Simon Willison, who picked up on the slightest error or inconsistency, and whose attention to detail and well thought-out comments ensured that this book is accurate and clear.

To those people who are really breaking new ground in the world of CSS, those whose ideas are discussed throughout this book, and those who share their ideas and creativity with the wider community, thank you.

Thanks to Drew for his support and encouragement, for being willing to discuss CSS concepts as I worked out my examples for the book, for making me laugh when I was growing annoyed, and for putting up with our entire lack of a social life. Finally, thanks must go to my daughter Bethany, who is very understanding of the fact that her mother is constantly at a computer, and who reminds me of what is important every day. You both make so many things possible, thank you.

1 Getting Started with CSS

Cascading Style Sheets sound intimidating. The name alone conjures up images of cryptic code and syntax too difficult for the layperson to grasp. In reality, however, CSS is one of the simplest and most convenient tools available to Web developers. In this first chapter, which takes a different format than the rest of the book, I'll guide you through the basics of CSS and show you how it can be used to simplify the task of managing a consistently formatted Website. If you've used CSS to format text on your sites, you may want to skip this chapter and jump straight to the solutions that begin in Chapter 2.

The Problem with HTML

CSS is a language that's used to define the formatting applied to a Website, including colors, background images, typefaces (fonts), margins, and indentation. If you've never used CSS before, you could be forgiven for thinking, "Well, I do all that now with HTML tags. Why would I need CSS?" It's a valid question that's best answered with an illustration of the problems that can arise when we define styles using HTML.

At present, a popular design choice is to use a sans-serif font (such as Arial, Verdana, Tahoma, etc.) for the main body text of a site. Since most Web browsers default to a serif font like Times New Roman, creating a complex Web page layout using a sans-serif font will often involve a lot of `<font>` tags. In a complex layout,

you might see ten or twenty <font> tags dedicated to applying the same font to all text on a page. Multiply this by five—the number of pages on a modest site—and we're in the neighborhood of one hundred tags. A beefier site might have fifty pages or more, in which case you're looking at one thousand <font> tags, all of them dedicated to applying that one basic, consistent style to your document's text.

Now here's the kicker: your client calls you late one Friday afternoon to say, "Verdana is nice, but everyone uses it. Let's use Tahoma instead." Fancy search-and-replace tools aside, you're now faced with the task of adjusting one hundred, one thousand, or even more <font> tags to make what, from your client's perspective, seems like a very simple change. You can kiss that ski weekend you had planned goodbye. And, try not to groan aloud—it doesn't go over well with most customers.

If you know your HTML, you may be thinking that the <basefont> tag, which lets you set the default font to be used throughout a page, provides a nice solution to this problem. But even then, you'd have to adjust one tag for each page of your site. Add another font style to the equation (if, say, you wanted to use a different font for that fancy navigation bar of yours), and the problem returns in full.

Another reason why you shouldn't use HTML to format your site is that these presentational elements are deprecated (flagged to be removed in future specifications) and can't be used if you wish to follow a Strict DOCTYPE such as HTML 4.01 Strict or XHTML 1.0 Strict. While your page will still be valid if you use a transitional DOCTYPE, it's good practice to avoid using these deprecated elements where possible. As you'll discover through the examples in this book, CSS allows you to do lots of things that you can't do with HTML alone, so there are many reasons why you should use CSS—but let's stop talking and see some CSS in action!

Defining Styles with CSS

The basic purpose of CSS is to allow the designer to define a style (a list of formatting details such as fonts, sizes, and colors) and then, to apply it to one or more portions of one or more HTML pages using a selector. Let's look at a basic example to see how this is done.

Consider the following HTML document outline:

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
    "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
<title>A Simple Page</title>
<meta http-equiv="content-type"
    content="text/html; charset=iso-8859-1" />
</head>
<body>
<h1><font face="sans-serif" color="#3366CC">First Title</font>
</h1>
<p>…</p>

<h2><font face="sans-serif" color="#3366CC">Second Title</font>
</h2>
<p>…</p>

<h2><font face="sans-serif" color="#3366CC">Third Title</font>
</h2>
<p>…</p>
</body>
</html>
```

This document contains three headings, created using `<h1>` and `<h2>` tags. To make these headings stand out more, I used `<font>` tags to display them in a light blue, sans-serif font (Windows browsers will display them in Arial, for example). Notice the repetition involved, even at this basic level. I had to specify the details of the font I wanted three separate times. Wouldn't it make sense to define the font just once, then apply it to my headings? Here's the CSS version:

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
    "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
<title>A Simple Page</title>
<meta http-equiv="content-type"
    content="text/html; charset=iso-8859-1" />
<style type="text/css">
h1, h2 {
 font-family: sans-serif;
 color: #3366CC;
}
</style>
</head>
<body>
<h1>First Title</h1>
```

```
<p>…</p>

<h2>Second Title</h2>
<p>…</p>

<h2>Third Title</h2>
<p>…</p>
</body>
</html>
```

All the magic lies between the `<style>` tags in the `<head>` of the document, where we define our light blue, sans-serif font and apply it to all `<h1>` and `<h2>` tags in the document. Don't worry about the syntax; I'll explain it in detail in a moment. Meanwhile, the `<font>` tags have completely disappeared from the `<body>`, leaving our document looking a lot less cluttered. Changes to the style definition at the top of the page will affect all three headings, as well as any other headings that are added to the page.

Now that you have an idea of what CSS does, let me explain the different ways of using CSS styles in your HTML documents. The simplest way of putting CSS styles into your Web pages is to use the `<style>` tag, as I did in the example above. This lets you declare any number of CSS styles by placing them inside the `<style>` tag, as follows:

```
<style type="text/css">
CSS Styles here
</style>
```

The `type` attribute specifies the language that you're using to define your styles. CSS is the only language in wide use as of this writing, and is indicated with the value `text/css`.

While it's nice and simple, the `<style>` tag has one major disadvantage. Specifically, if you want to use a particular set of styles throughout your site, you'll have to repeat those style definitions in a `<style>` tag at the top of every one of your site's pages.

A more sensible alternative is to put those definitions in a plain text file (usually given a `.css` filename), then link your documents to that file. Any changes to the style definitions in that one file will affect all the pages that link to it. This achieves the objective of site-wide style definitions mentioned earlier.

To link a document to a CSS text file (say, `styles.css`), place a `<link>` tag in the document's header:


```
<link rel="stylesheet" type="text/css" href="styles.css" />
```

Let's return to the original example in which three headings shared a single style; that document would now look 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">
<head>
<title>A Simple Page</title>
<meta http-equiv="content-type"
    content="text/html; charset=iso-8859-1" />
<link rel="stylesheet" type="text/css" href="styles.css" />
</head>
<body>
<h1>First Title</h1>
<p>…</p>

<h2>Second Title</h2>
<p>…</p>

<h2>Third Title</h2>
<p>…</p>
</body>
</html>
```

The `styles.css` file would contain the style definition:

```
h1, h2 {
  font-family: sans-serif;
  color: #3366CC;
}
```

As with an image file, you can reuse this `styles.css` file across as many pages as you need. Not only will it save you typing, it also ensures a consistent look to the headings across your entire site.

CSS Selectors

Every CSS style definition has two components: the **selector**, which defines the tags to which the style will be applied, and the **properties**, which specify what the style actually does. In the previous example, the selector was `h1, h2`, specifying that the style should apply to all `<h1>` and `<h2>` tags. The remainder of the style definition comprised the attributes, specifying the font and color that should be

applied by the style. In this section, I'll describe the basic CSS selector types and give examples of each.

Tag Selectors

The most basic form of selector is that which we have already seen. By naming a particular HTML tag, you can apply a style definition to every occurrence of that tag in the document. This is often used to set the basic styles that will appear throughout a Website. For example, the following might be used to set the default font for a Website:

```
body, p, td, th, div, blockquote, dl, ul, ol {
  font-family: Tahoma, Verdana, Arial, Helvetica, sans-serif;
  font-size: 1em;
  color: #000000;
}
```

This rather long selector is a list of tags, all of which will take on the style definition (font, size, and color). In theory, the `<body>` tag is all that's needed (as all the other tags appear inside the `<body>` tag, and would thus inherit its properties), but many browsers don't properly carry style properties into tables and other elements. Thus, I specified the other elements for the sake of completeness.

Pseudo-Class Selectors

The formatting of the `<a>` tag in HTML is more versatile than is the formatting of most other tags. By specifying `link`, `vlink`, and `alink` attributes in the `<body>` tag, you can set the colors for the various states of the links in your page (unvisited, visited, and being clicked on, respectively). CSS provides its own way of doing this, and adds a fourth state that's applied when the mouse hovers over the link. Consider the following example:

```
a:link { color: #0000FF; }
a:visited { color: #FF00FF; }
a:hover { color: #00CCFF; }
a:active { color: #FF0000; }
```

This code contains four CSS style definitions. Each of the selectors uses what is termed a **pseudo-class** of the `<a>` tag. The first, `link`, applies to unvisited links only, and specifies that they should be blue. The second, `visited`, applies to visited links, and makes them magenta. The third style definition, `hover`, overrides the first two by making links light blue when the mouse is moved over them,

whether they've been visited or not. The final style definition makes links red when they're clicked on. Because `active` appears last, it overrides the first three, so it will take effect whether the links have been visited or not, and whether the mouse is over them or not.

Class Selectors

Assigning styles to tags is all well and good, but what happens if you want to assign different styles to identical tags occurring in different places within your document? This is where CSS **classes** come in. Consider the following style, which makes all paragraph text in the page blue:

```
p { color: #0000FF; }
```

Now, what if you had a sidebar on your page with a blue background? You wouldn't want text in the sidebar to be blue as well, because it would be invisible! What you need to do is define a class for your sidebar text, then assign a CSS style to that class:

```
p { color: #0000FF; }
.sidebar { color: #FFFFFF; }
```

This second rule uses a class selector that indicates that the style should be applied to any tag of the `sidebar` class. The period indicates that a class is being named, instead of a tag. To create a paragraph of text of the `sidebar` class, you add a `class` attribute to the tag:

```
<p class="sidebar">This text will be white, as specified by the
  CSS style definitions above.</p>
```

Now, what if there were links in your sidebar? By default, they'd be rendered just like any other links in your page; however, add a `class="sidebar"` attribute to the tag, and they'll turn white, too:

```
<p class="sidebar">This text will be white, <a class="sidebar"
      href="link.html">and so will this link</a>.</p>
```

That's pretty neat, but what if you wanted to make the links stand out a bit more by displaying them in bold text? Adding the `bold` text attribute to the `sidebar` class will make your whole sidebar bold. You need a CSS selector that selects links of the `sidebar` class only. By combining a tag selector with a class selector, you can do exactly that:

```
p { color: #0000FF; }
.sidebar { color: #FFFFFF; }
a.sidebar:link, a.sidebar:visited { font-weight: bold; }
```

Note that we've also used the `:link` and `:visited` pseudo-classes to specify `<a>` tags that are links (as opposed to `<a name="…">` tags that are not).

Note also that our sidebar links are still white—both of the styles pertaining to the sidebar class apply to our sidebar links. If we specified a different color in the third style, however, links would adopt that new color, because the third selector is more specific, and CSS styles are applied in order of increasing selector specificity.

Incidentally, the process of applying multiple styles to a single page element is called **cascading**, and is where Cascading Style Sheets got their name.

Contextual Selectors

If your sidebar happens to contain a lot of links, it becomes tedious to assign the `sidebar` class to every single `<a>` tag. Wouldn't it be nice to use a selector to select any link that appeared inside a paragraph of the `sidebar` class? That's what contextual selectors are for: selecting a tag based on its **context**, that is, based on the tag(s) that contain it.

Here's the new CSS:

```
p { color: #0000FF; }
.sidebar { color: #FFFFFF; }
p.sidebar a:link, p.sidebar a:visited {
  font-weight: bold;
  color: #FFFFFF;
}
```

And here's the updated HTML:

```
<p class="sidebar">This text will be white,
  <a href="link.html">and so will this link</a>.</p>
```

As you can see, a contextual selector provides a list of selectors separated by spaces that must match tags "from the outside in." In this case, since the link (matched by the `a:link` or `a:visited` selector) occurs inside a sidebar paragraph (matched by the `p.sidebar` selector), the style applies to the link.

Note that, to keep the link white, we had to add that color to the CSS attributes for the links, as the style for the `sidebar` class no longer applies to the links.

ID Selectors

Similar to class selectors, ID selectors are used to select one particular tag, rather than a group of tags. The tag that you select is identified by an ID attribute as follows:

```
<p id="sidebar1">This paragraph is uniquely identified by the ID
  "sidebar1".</p>
```

An ID selector is simply the ID preceded by a hash (`#`). Thus, the following style will make the above paragraph white:

```
#sidebar1 { color: #FFFFFF; }
```

ID selectors can be used in combination with the other selector types. The following style, for example, applies to unvisited links appearing in the `sidebar1` paragraph:

```
#sidebar1 a:link {
  font-weight: bold;
  color: #FFFFFF;
}
```

Other selector types exist, but they're not widely used. Support for them in some browsers (especially Netscape 4) is buggy at best.

CSS Properties

In the examples used so far in this chapter, we've used several common CSS properties like `color`, `font-family`, `font-size`, and `font-weight`. In the rest of the book, we'll be using these properties—and a lot more.

Summary

This chapter has given you a taste of CSS and its usage at the most basic level. If you haven't used CSS before, but have a basic understanding of the concepts discussed in this chapter, you should be able to start using the examples in this book. The examples in the early chapters of this book are somewhat simpler than those found near the end, so, if you haven't worked with this technology before,

you might want to begin with the earlier chapters. These will build on the knowledge you gained in this chapter to get you using and, I hope, enjoying CSS.

2

Text Styling and Other Basics

This chapter explores the applications of CSS for styling text, and covers a lot of CSS basics as well as answering some of the more frequently asked questions. If you're new to CSS, these examples will introduce you to a variety of properties and usages, and will give you a broad overview from which to start your own experiments with CSS. For those who are already familiar with CSS, this chapter will serve as a quick refresher for those moments when you can't quite remember how to achieve a certain effect.

The examples provided here are well supported across a variety of browsers and versions. As always, testing your code across different browsers is important; however, the text styling properties have good support. While there may be small inconsistencies or some lack of support in older browsers, nothing here should cause you any serious problems.

How do I replace font tags with CSS?

The styling of text with CSS is supported by version 4 browsers and above, so there is no compelling reason to continue to use `<font>` tags to style text. At its very simplest, you can use CSS to replace the font tags in your site.

Using font tags, you would need to set the style for each paragraph on your page.

```
<p><font color="#800080"
      face="Verdana,Geneva,Arial,Helvetica,sans-serif">These
    stuffed peppers are lovely as a starter, or as a side dish
    for a Chinese meal. They also go down well as part of a
    buffet and even children seem to like them.</font></p>
```

Solution

Using CSS, you would simply define in the style sheet that the `color` property of the `<p>` tag is `#800080`, and that the `font-family` should be `Verdana,Geneva,Arial,Helvetica,sans-serif`:

```
p {
  color: #800080;
  font-family: Verdana,Geneva,Arial,Helvetica,sans-serif;
}
```

Now, every time you add to your document text enclosed in a `<p>` tag, it will take on this style, and save you from needing to add reams of extra markup to your document. It also makes life a lot easier if your client suddenly wants to change the font from Verdana to Times on 100 documents!

Should I use pixels, points, ems or something else for font sizes?

You can size text in CSS using the `font-size` property. For example:

```
font-size: 12px;
```

However, there are a variety of other ways to set the size of your fonts. Deciding which to use requires you to know a little about the relative merits of each.

Solution

The Units of Font Sizing

Table 2.1 describes the units that you can use to size fonts.

Table 2.1. Font Sizing Units

Unit identifier	Corresponding Units
pt	Points
pc	Picas
px	Pixels
em	Ems
ex	Exes
%	Percentages

Points and Picas

```
p {
  font-size: 10pt;
}
```

You should avoid using points and picas to style text for display on screen. These units are an excellent way to set sizes for print design, as their measurements were designed for that purpose. A point has a fixed size of one seventy-second of an inch, while a pica is one sixth of an inch. A printed document specified using these units will come out exactly as you intended. However, computers cannot accurately predict the actual size at which elements will appear on the monitor, so they guess—and guess badly—at the size of a point or pica, with varying results across platforms. If you're creating a print style sheet, or a document that's intended for print, not on-screen, viewing, these are the units to use. However, a general rule of thumb is to avoid them when designing for the Web.

Pixels

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

Many designers like to measure font sizes in pixels, as this unit of measurement makes it easy to achieve consistent displays across various browsers and platforms. However, pixels ignore any preferences users may have set in their own browsers and, in most browsers, font sizes that the designer has dictated in pixels cannot be resized by users. This is a serious accessibility problem for those who need to make text larger in order to read it clearly. Therefore, while pixels may seem like the easiest option, pixel measurements should be avoided if another method can

be used, particularly for large blocks of content. If you're creating a document for print, or using a print style sheet, you should avoid pixels entirely. Pixels have no meaning in the world of print and, similar to the application of points to the on-screen environment, print applications provided with a pixel measurement will simply try to guess the size at which the font should appear on paper, with erratic results.

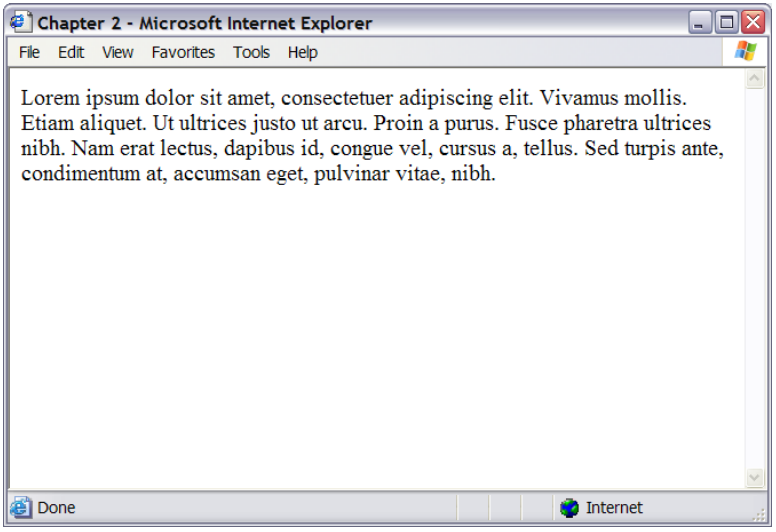
Ems

The em is a relative font measurement, where one em is equal to the height of the letter "M" in the default font size. Where CSS is concerned, `1em` is seen to be equal to the user's default font size, or the font size of the parent element when it differs. If you use ems (or any other relative unit) for all your font sizing, users will be able to resize the text, which will comply with the text size preferences they have set in their browsers. For example, imagine I create a declaration that sets text within the `<p>` tag to `1em`, as is shown in the following CSS:

```
p {
  font-size: 1em;
}
```

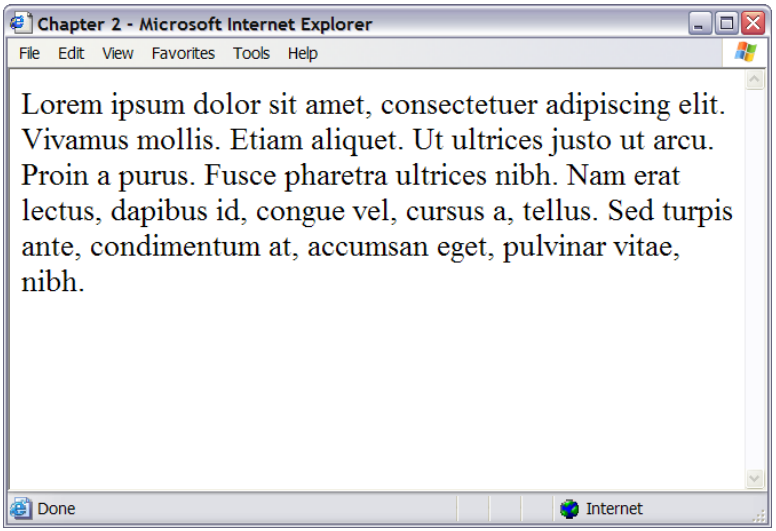
A user with an Internet Explorer 6 browser in which text size is set to Medium will see the paragraph shown in Figure 2.1.

Figure 2.1. The `font-size` is set to `1em` and text size is Medium.



If the users have their text set to Largest, the `1em` text will display as shown in Figure 2.2.

Figure 2.2. The `font-size` is set to `1em` and text size is set to Largest.



As a designer, this gives you less control over the way users view the document. However, it means that users who need a very large font size, for instance, can read your content.

Ems values can be set using decimal numbers. For example, to display text at a size 10% smaller than the user's default (or the font size of its parent element) you could use:

```
p {
  font-size: 0.9em;
}
```

To display the text 10% larger than the default or inherited size:

```
p {
  font-size: 1.1em;
}
```

Exes

The ex is a relative unit measurement that corresponds to the height of the lowercase letter "x" in the default font size. In theory, if you set the `font-size` of a paragraph to `1ex`, the uppercase letters of the text should be the same height as the letter "x" would have been if the font size had not been specified.

Unfortunately, modern browsers don't yet support the typographical features needed to determine the size of an ex precisely. They usually make a rough guess for this measurement. For this reason, exes are rarely used at this time.

Percentages

```
p {
  font-size: 100%;
}
```

As with ems and exes, font sizes that are set in percentages will honor users' text size settings and are resizable by the user. Setting the `<p>` tag to `100%` would display your text at users' default settings for font size (as would setting the font size to `1em`). Decreasing the percentage will make the text smaller:

```
p {
  font-size: 90%;
}
```

Increasing the percentage will make the text larger:

```
p {
  font-size: 150%;
}
```

Keywords

You can also size text using absolute and relative keywords.

Absolute Keywords

There are seven absolute keyword sizes for use in CSS, as follows:

- `xx-small`
- `x-small`

- `small`
- `medium`
- `large`
- `x-large`
- `xx-large`

These keywords are defined relative to each other, and browsers implement them in different ways. Most browsers display `medium` at the same size as unstyled text, with the other keywords resizing text to varying degrees. Internet Explorer 5 (and version 6, depending on the document type), however, treats `small` as being the same size as unstyled text.

These keyword measurements are considered absolute in that they don't inherit from any parent element. Yet, unlike the absolute values provided for height, such as pixels and points, they do allow the text to be resized in the browser, and will honor the user's browser settings. The main problem with using these keywords is the fact that, for example, `x-small`-sized text may be perfectly readable in one browser, and miniscule in another.

Relative Keywords

Relative keywords—`larger` and `smaller`—take their size from the parent element, in the same way that `em` and `%` do. Therefore, if you set your `<p>` tag to `small` using absolute keywords, and you simply want emphasized text to display comparatively larger, you'd add the following to the style sheet:

File: **relative.css**

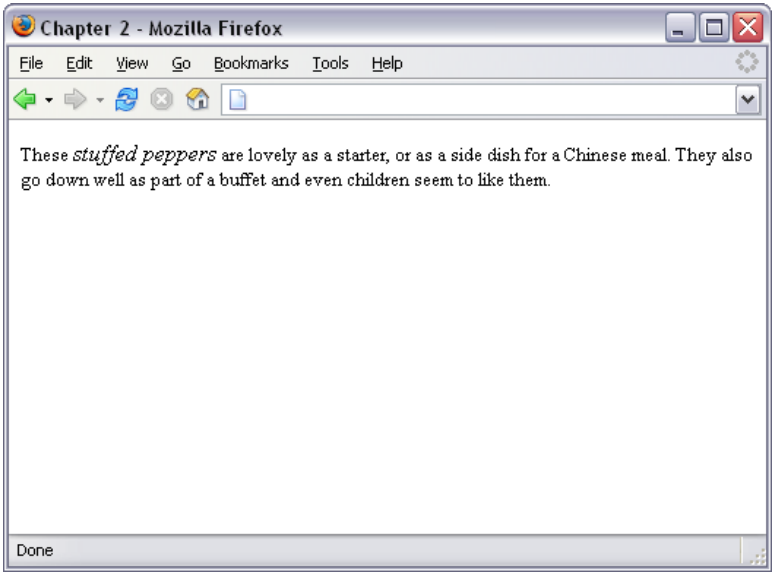
```
p {
  font-size: small;
}
em {
  font-size: larger;
}
```

The following markup would display as shown in Figure 2.3, because the text wrapped in the `<em>` tag will display larger than its parent—the `<p>` tag.

File: **relative.html (excerpt)**

```
<p>These <em>stuffed peppers</em> are lovely as a starter, or as a
  side dish for a Chinese meal. They also go down well as part of
  a buffet and even children seem to like them.</p>
```

Figure 2.3. The emphasized text displays larger than its containing paragraph.



Relative Sizing and Inheritance

When you use any kind of relative sizing, remember that the element inherits its starting size from its parent element, then adjusts its size accordingly. This is fairly easy to understand in layouts in which elements are not nested in a complex manner; however, this inheritance pattern can become problematic in nested table layouts in which the parent element is not always obvious—things can seem to inherit very strangely indeed! The following example demonstrates this.

My style sheet contains the following code, which sets `td` to display text at 80%. This is slightly smaller than the user's default font size, but they will be able to resize it:

File: **nesting.css**

```
td {
  font-size: 80%;
}
```

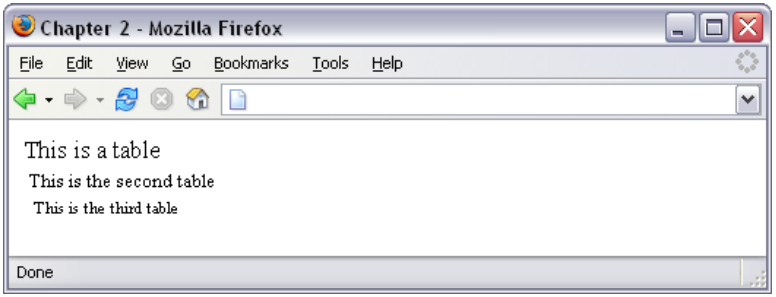
On a page in which there are no nested table cells, the text will display consistently at that slightly smaller size. However, in a nested table layout, the text within each nested table will display at 80% of the font size of its containing table.

File: **nesting.html (excerpt)**

```
<table>
  <tr>
    <td>This is a table
      <table>
        <tr>
          <td>This is the second table
            <table>
              <tr>
                <td>This is the third table</td>
              </tr>
            </table>
          </td>
        </tr>
      </table>
    </td>
  </tr>
</table>
```

The example markup above will display as in Figure 2.4. As you can see, the text becomes progressively smaller in each nested table.

Figure 2.4. The display demonstrates the use of relative font sizing within nested tables.



Discussion

When choosing which method of sizing text to use, it's best to select one that allows all users to resize the text, and that ensure that the text complies with the settings users have chosen in their browsers. Relative font sizing tends to work well with CSS layouts and simple table-based layouts, but it can be tricky to implement in a complex nested table layout because of the way the elements inherit sizing. If you decide that your only option is to size fonts using an absolute unit of measurement, consider developing a style sheet switcher to allow users to change the font size from within the interface of your site.

How do I specify that my text is shown in a certain font?

Solution

Specify the typeface that your text will adopt using the `font-family` property like so:

```
p {
  font-family: Verdana;
}
```

Discussion

As well as specific fonts, such as Verdana or Times, CSS allows the specification of some more generic font families, which are:

- `serif`
- `sans-serif`
- `monospace`
- `cursive`
- `fantasy`

When you specify fonts, it's important to remember that users probably don't have the same fonts you have on your computer. If you define a font that they don't have, your text will display in their Web browser's default font, regardless of what you'd have preferred, or your site design.

You can simply use generic font names and let users' systems decide which font to apply. For instance, if you want your document to appear in a sans-serif font such as Arial, you could simply use the following code:

```
p {
  font-family: sans-serif;
}
```

However, you'll probably want to have a little more control over the way your site displays—and you can. It's possible to specify font names as well as generic fonts. Take, for example, the following CSS declaration for the `<p>` tag. Here, we've specified that if Verdana is installed on the system, it should be used; if it's not installed, the computer is directed to see if the user has Geneva installed; failing that, the computer will look for Arial, then Helvetica. If none of these fonts is available, the computer is instructed to use that system's default sans-serif font.

```
p {
  font-family: Verdana, Geneva, Arial, Helvetica, sans-serif;
}
```

How do I remove underlines from my links?

The default indication that text on a Web page is a link to another document is that it's underlined and is a different color from the rest of the text. There may be instances in which you want to remove that underline.

Solution

We use the `text-decoration` property to remove the underlines. By default, the browser will set the `text-decoration` of an `<a>` tag to `underline`. To remove the underline, simply set the following property for the link:

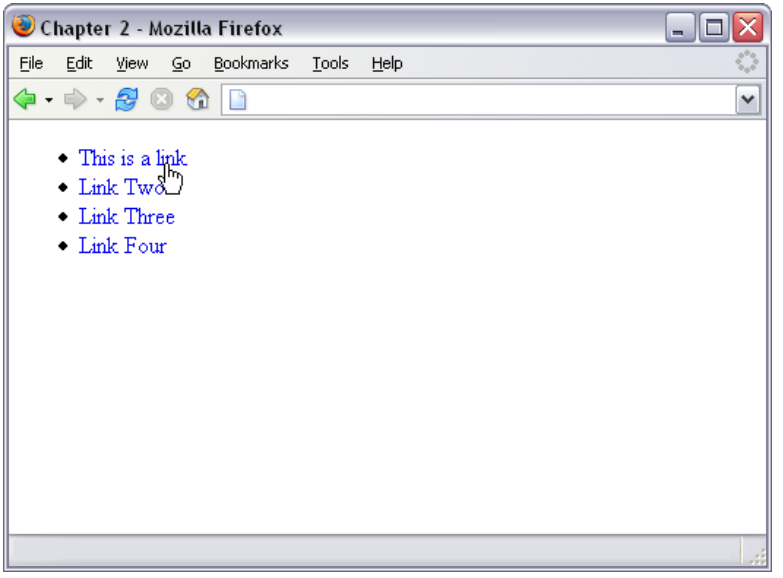
```
text-decoration: none;
```

The CSS used to create the effect shown in Figure 2.5 is as follows:

File: **textdecoration.css**

```
a:link, a:visited {
  text-decoration: none;
}
```

Figure 2.5. Use `text-decoration` to create links that aren't underlined.



Discussion

In addition to `underline` and `none`, there are other values for `text-decoration` that you can try out:

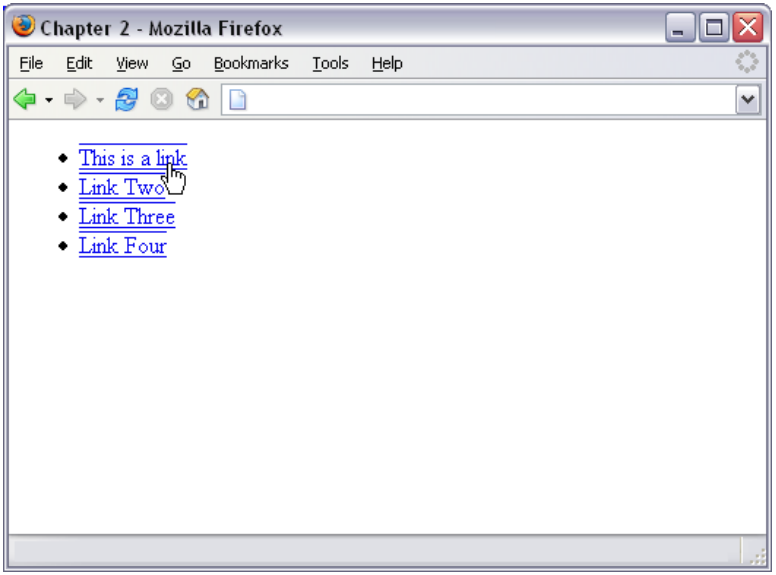
- `overline`
- `line-through`
- `blink`

You can also combine these values. For instance, should you wish to have an underline and overline on a particular link, as shown in Figure 2.6, you'd use the following code:

File: **textdecoration2.css**

```
a:link, a:visited {
  text-decoration: underline overline;
}
```

Figure 2.6. Combine `text-decoration` values to create links with underlines and overlines.



Misleading Lines

You can use the `text-decoration` property on text that's not a link, however, be wary of this. The underlining of links is such a widely-accepted convention that users tend to think that any underlined text is a link to another document.

When is Removing Underlines a Bad Idea?

Underlining links is a standard convention followed by all Web browsers and, consequently, users expect to see links underlined. Removing the underline from

links that are within text can make it very difficult for people to realize that these words are in fact links—not just highlighted text. I'd advise against removing the underlines from links within text. There are other ways in which you can style links so they look attractive and removal of the underline is rarely, if ever, necessary.

Links that are used as part of a menu, or in some other situation in which the text is quite obviously a link—for instance, where the text is styled with CSS to resemble a graphical button—are a different story. If you wish, you can remove the underline from these kinds of links because it's obvious from their context that they're links.

How do I create a link that changes color on mouseover?

An attractive link effect changes the color or otherwise alters the appearance of links when the cursor moves across them. This effect can be applied to great advantage when CSS is used to replace navigation buttons; however, it can also be used on links within your document.

Solution

To create this effect, we style the `:hover` and `:active` pseudo-classes differently than the other pseudo-classes of the anchor tag.

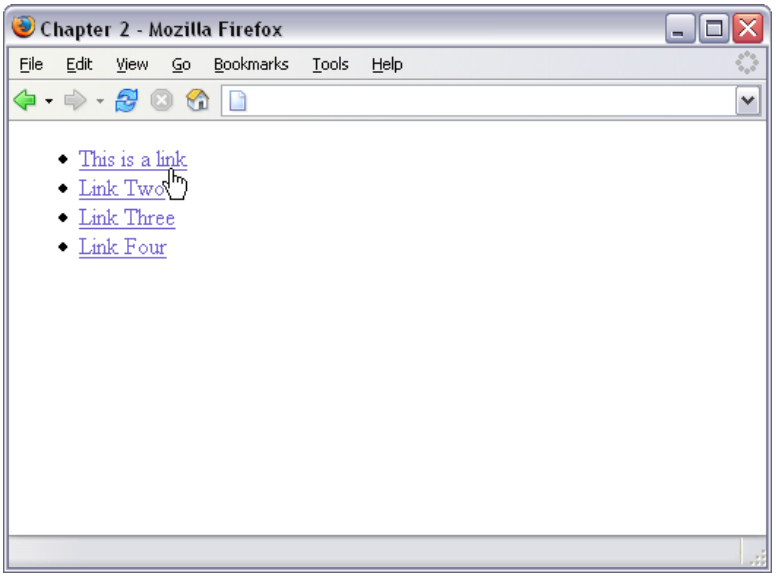
Our links are styled with the following declaration in the style sheet:

File: **textdecoration3.css**

```
a:link, a:visited, a:hover, a:active {
  text-decoration: underline;
  color: #6A5ACD;
  background-color: transparent;
}
```

When this style sheet is applied, our links will display in the blue color `#6A5ACD` with an underline, as shown in Figure 2.7.

Figure 2.7. The same declaration is used for all pseudo-classes of these links.

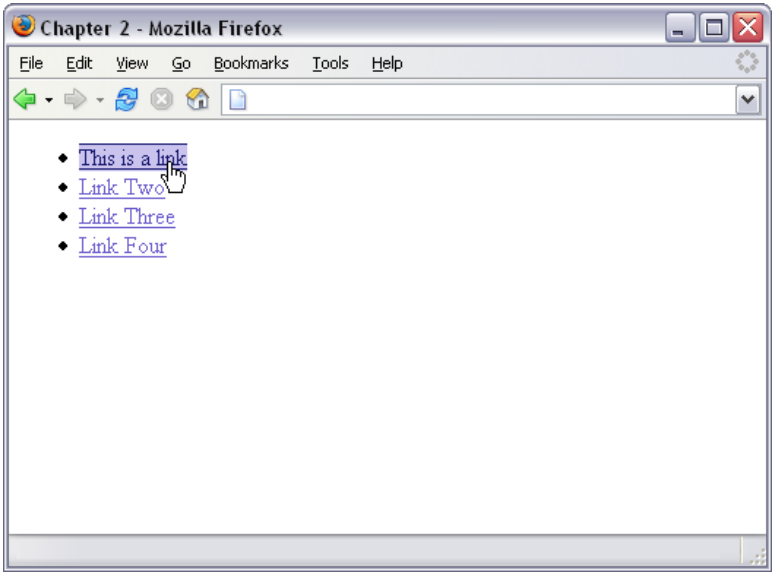


To make our `:hover` and `:active` pseudo-classes different, we need to remove them from the declaration with the other pseudo-classes and give them their own declaration. I decided to apply an overline in addition to the underline, a background color, and make the text a darker color. The resulting CSS is below (see also Figure 2.8):

File: **textdecoration4.css**

```
a:link, a:visited {
  text-decoration: underline;
  color: #6A5ACD;
  background-color: transparent;
}
a:hover, a:active {
  text-decoration: underline overline;
  color: #191970;
  background-color: #C9C3ED;
}
```

Figure 2.8. The effect of our CSS is evident when we roll over a link with a hover style.



As you've probably realized, you can style the other pseudo-classes separately, too. In particular, you might like to style differently links that the user has visited. To do this, you'd simply style the `:visited` pseudo-class separately.

When styling pseudo-classes, take care that you don't change either the size or weight (boldness) of the text. If you do, you'll find that your page appears to "jiggle," as the content has to move to make way for the larger text to appear on hover.

Pseudo Order

The link pseudo-classes should be declared in the following order: `link`, `visited`, `hover`, `active`. If they aren't, you may find that they don't work as you intended. One way to remember this order is the mnemonic: LoVe-HAte.

How do I display two different styles of link on one page?

The previous tip explained how to style the different selectors of the anchor tag, but what if you want to use different link styles within the same document? Perhaps you want to display links without underlines in your navigation menu, yet make sure that links within the body content are easily identifiable. Maybe part of your document has a dark background color, so you need to use a light-colored link style there.

Solution

To demonstrate how to create multiple styles for links, let's take an example in which we've already styled the links.

File: **linktypes.css (excerpt)**

```
a:link, a:visited {
  text-decoration: underline;
  color: #6A5ACD;
  background-color: transparent;
}

a:hover, a:active {
  text-decoration: underline overline;
  color: #191970;
  background-color: #C9C3ED;
}
```

These should be taken as the default style—this is how links will normally be styled within your documents. This link style makes the link blue, so, if you have an area with a blue background on your page, the links will be unreadable. You need to create a second set of styles for any link within that area.

First, you need to give a class or an ID to the area that will contain the differently colored links. If the area is already styled with CSS, it will already have a class or ID that you can use. Imagine your document contains the following markup:

File: **linktypes.html (excerpt)**

```
<div class="boxout">
  <p>Visit our <a href="store.html">online store</a>, for all your
```

```
    widget needs.</p>
</div>
```

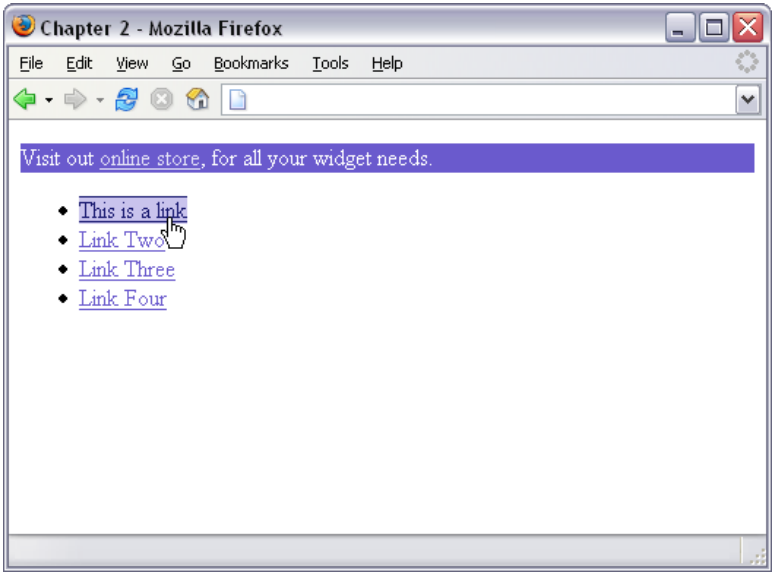
You will need to create a CSS rule to affect any link appearing within an area for which the parent element has the class `boxout` applied:

File: **linktypes.css (excerpt)**

```
.boxout {
  color: #FFFFFF;
  background-color: #6A5ACD;
}
.boxout a:link, .boxout a:visited {
  text-decoration: underline;
  color: #E4E2F6;
  background-color: transparent;
}

.boxout a:hover, .boxout a:active {
  background-color: #C9C3ED;
  color: #191970;
}
```

Figure 2.9. Two different styles of links can be used in one document.



As shown in Figure 2.9, this will display all links in the document as per the first style except those that appear in the `boxout`—these links will be displayed in the lighter color.

How do I add a background color to a heading?

You can add a background color to any element, including a heading.

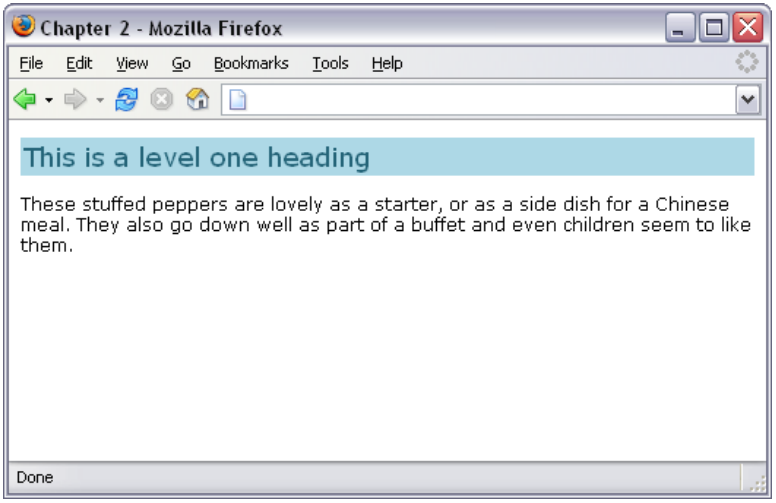
Solution

Below, we've created a CSS rule for all level one headings in a document. The result is shown in Figure 2.10.

File: **headingcolor.css (excerpt)**

```
h1 {
  background-color: #ADD8E6;
  color: #256579;
  font: 18px Verdana, Geneva, Arial, Helvetica, sans-serif;
  padding: 2px;
}
```

Figure 2.10. Headings can display with a background color.



Make Way for Color

When adding a background to a heading, you may also want to adjust the padding so that there's space between the heading text and the edge of the colored area, as we did in the above example.

How do I style headings with underlines?

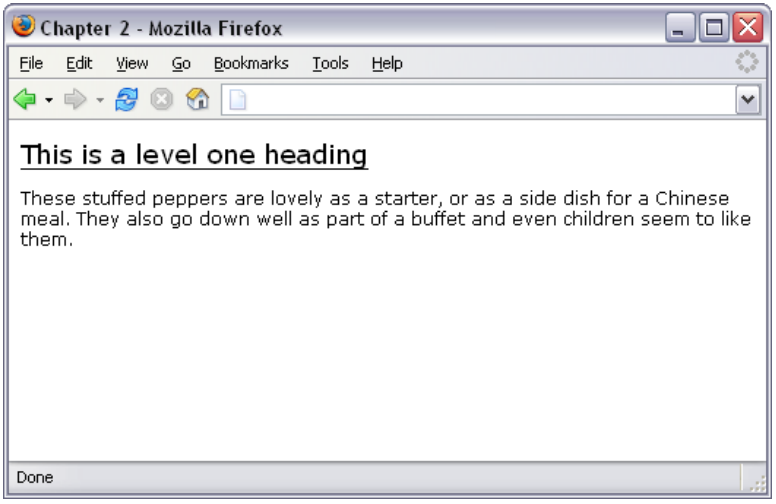
Solution

There are two ways in which you can add an underline to your text. The simplest way is to use the `text-decoration` property that we encountered when styling links above. This method will allow you to underline the text in the same color as the text itself, as shown in Figure 2.11.

File: **`headingunderline.css` (excerpt)**

```
h1 {
  font: 18px Verdana, Geneva, Arial, Helvetica, sans-serif;
  text-decoration: underline;
}
```

Figure 2.11. Add an underline to a heading using `text-decoration`.

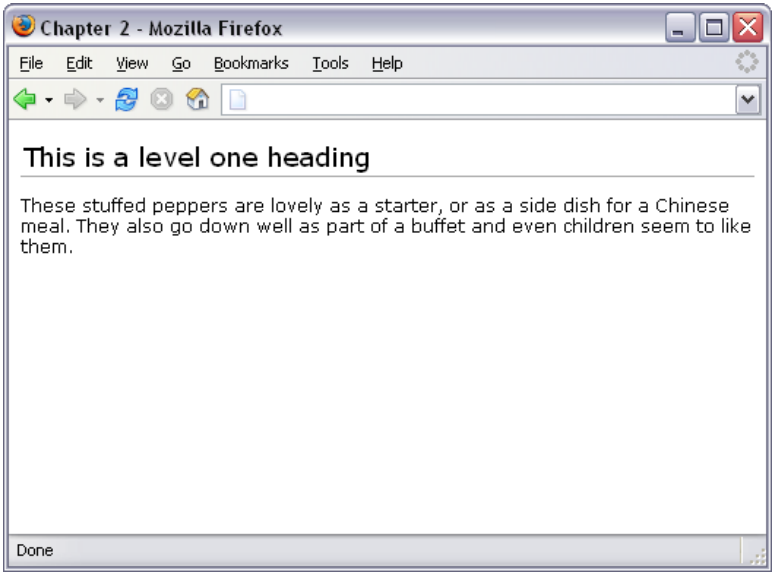


You can also create an underline effect by adding a bottom border to the heading. This solution is more flexible, in that you can separate the underline and the heading with the use of padding, and you can change the color of the underline to be different than the text. However, the effect may display slightly differently in different browsers, so you'll need to test it to make sure the effect looks reasonable on the browsers your visitors may use (see Figure 2.12).

File: **headingunderline2.css**

```
h1 {
  font: 18px Verdana, Geneva, Arial, Helvetica, sans-serif;
  padding: 2px;
  border-bottom: 1px solid #aaaaaa;
}
```

Figure 2.12. Add an underline effect using a bottom border.



How do I get rid of the large gap between an <h1> tag and the following paragraph?

Solution

By default, browsers render a gap between heading and paragraph tags. This is produced by a default top and bottom margin that browsers apply to these tags.

To remove all space between a heading and the paragraph that follows it, you must not only remove the bottom margin from the heading, but also the top margin from the paragraph. But since it can be inconvenient to target the first paragraph following a heading with a CSS selector, it's easier to simply assign a negative bottom margin to the heading. Margins can be set to negative values, though padding cannot.

The margin of the heading shown in Figure 2.13 has not been changed.

Figure 2.13. This heading and paragraph show the default spacing.

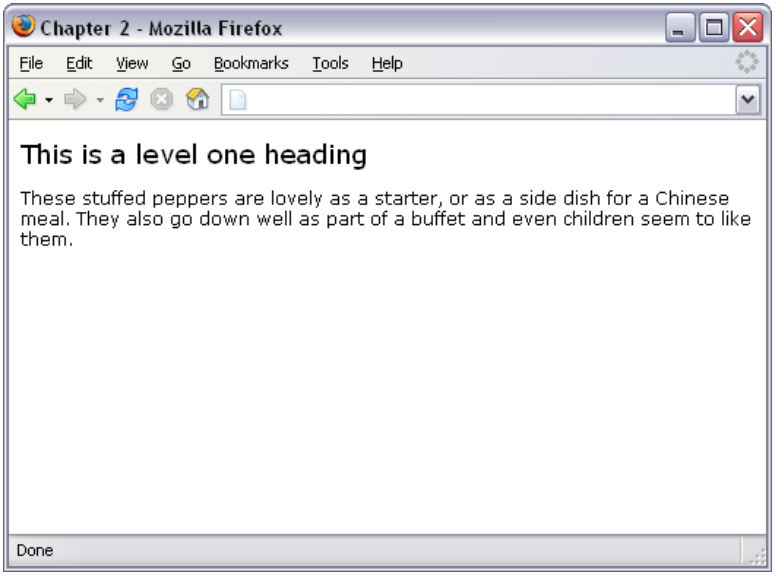
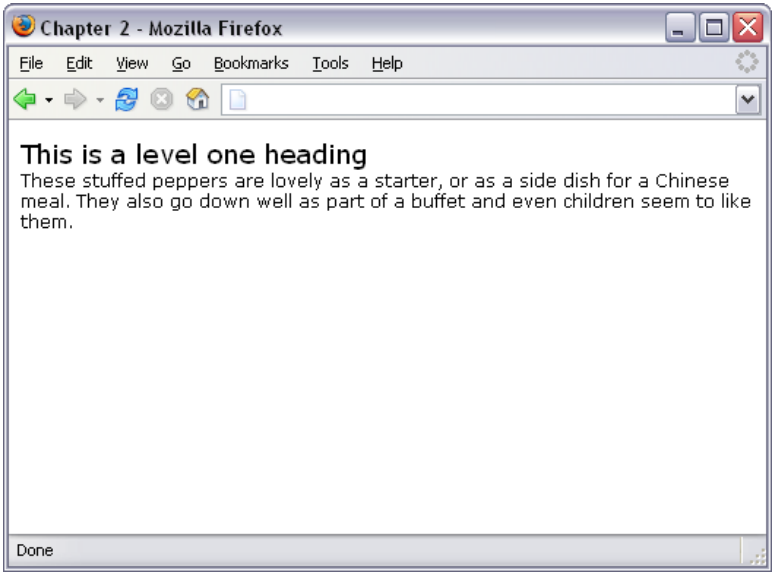


Figure 2.14 shows the same heading after the CSS below has been applied to the `<h1>` tag.

```
h1 {
  font: 18px Verdana, Geneva, Arial, Helvetica, sans-serif;
  margin-bottom: -12px;
}
```

Figure 2.14. The heading display changes when the `margin-bottom` is set to `-12px`.



How do I highlight text on the page without using font tags?

Before CSS, we might have used font tags to highlight an important term on the page, or to identify the search terms visitors had used to locate the document through a search engine.

Solution

CSS allows you to create a class for the highlighting style and apply it by wrapping the highlighted text with `<span>` tags that apply the class. For example, in the following paragraph, we've wrapped a phrase in span tags that apply the class, `hilite`.

File: **hilite.html (excerpt)**

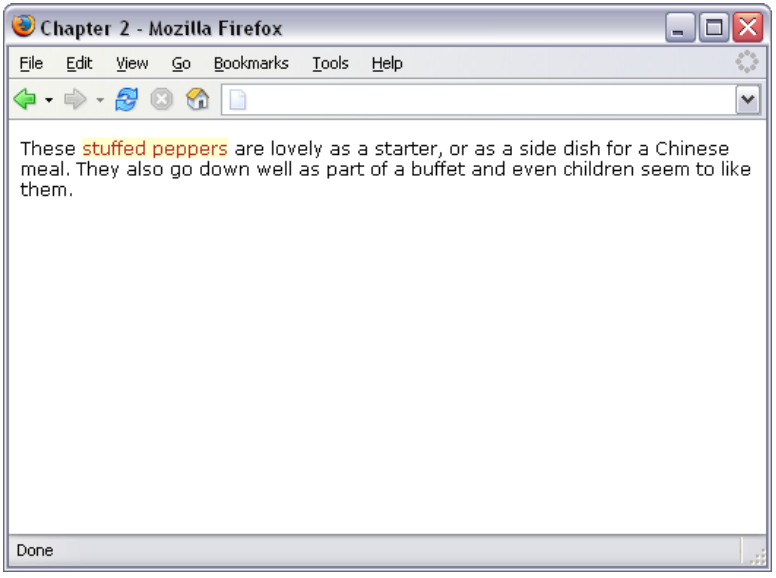
```
<p>These <span class="hilite">stuffed peppers</span> are lovely as
  a starter, or as a side dish for a Chinese meal. They also go
  down well as part of a buffet and even children seem to like
  them.</p>
```

The `hilite` class is shown below; the section will display as in Figure 2.15:

File: **hilite.css (excerpt)**

```
.hilite {
  background-color: #FFFFCC;
  color: #B22222;
}
```

Figure 2.15. Highlight text with CSS.



How do I alter the line-height (leading) on my text?

One of the great advantages of using CSS instead of font tags is that you have far more control over the look of the text on the page. In this solution, we'll see how to alter the leading of text in your document.

Solution

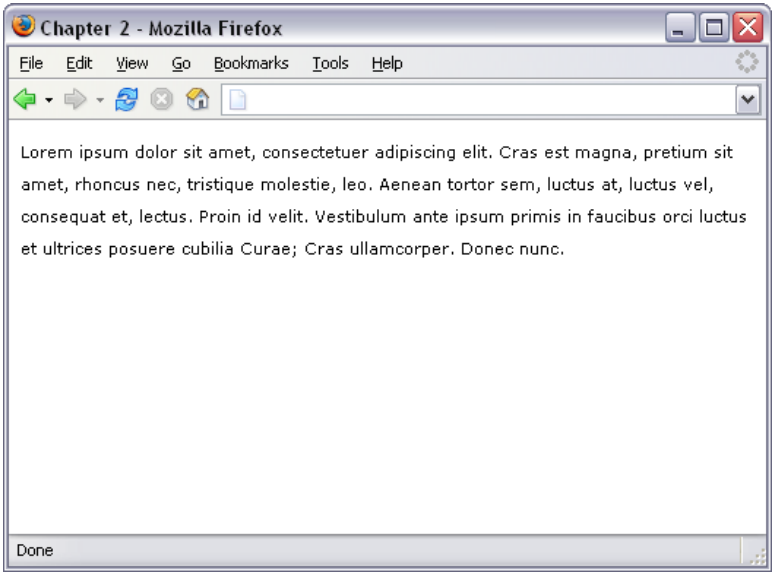
If the default spacing between lines of text looks a little narrow, change it with the `line-height` property.

File: **leading.css**

```
p {
  font: 11px Verdana, Geneva, Arial, Helvetica, sans-serif;
  line-height: 2.0;
}
```

The result is shown in Figure 2.16.

Figure 2.16. Alter the leading using the `line-height` property.



Be careful not to space the text out so much that it becomes difficult to read.

No Units?

You can also specify the `line-height` using standard CSS units of measurement, such as ems or even pixels. But doing so breaks the link between the line height and the font size for child elements.

For example, if the example above contained a `<span>` that set a large `font-size`, then the line height would scale up proportionally to maintain the same ratio because `line-height` of the paragraph was set to the numerical value `2.0`. If, however, `line-height` was set to `2em` or `200%`, the `<span>` would inherit the actual line height, not the ratio, and the large font size would not affect the line height of the span. Depending on the effect you're going for, this may actually be desirable.

How do I justify text?

When you justify text, you alter the spacing between the words so that both the right and left margins are straight, as with the text in this book. You can create this effect using CSS.

Solution

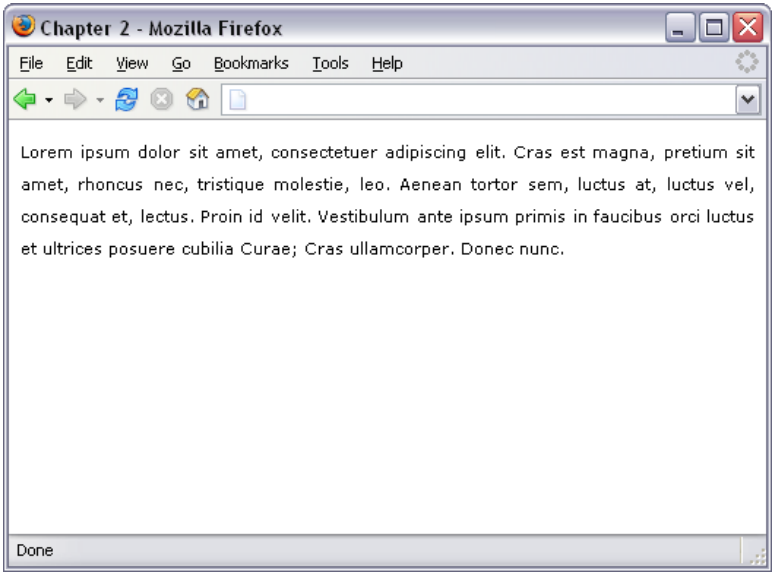
You can justify paragraph text with the help of the `text-align` property.

File: **justify.css**

```
p {
  text-align: justify;
  font: 11px Verdana, Geneva, Arial, Helvetica, sans-serif;
  line-height: 2.0;
}
```

Figure 2.17 shows the effect of this code.

Figure 2.17. Justify text using `text-align`.



Discussion

Other values for `text-align` are:

`right` aligns the text to the right of the container

`left` aligns the text to the left of the container

`center` aligns the text center

How do I style a horizontal rule?

Solution

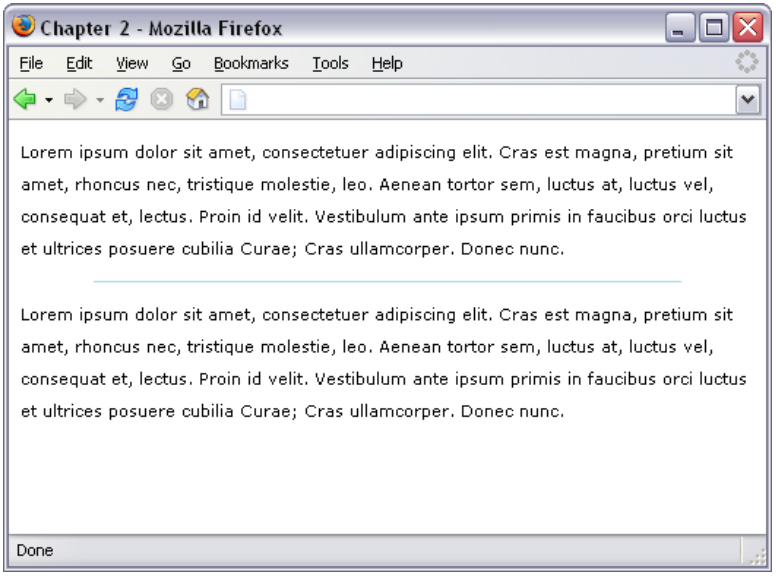
You can change the color, height and width of a horizontal rule. Note that, in the CSS below, I've used the same values for `color` and `background-color` because Mozilla-based browsers color the rule using `background-color`, while IE uses `color`.

File: **hrstyle.css (excerpt)**

```
hr {
  border: none;
  background-color: #ADD8E6;
  color: #ADD8E6;
  height: 1px;
  width: 80%;
}
```

The result can be seen in Figure 2.18.

Figure 2.18. Change the color, height and width of a horizontal rule.



How do I indent text?

Solution

You can indent text by applying a rule to the container that sets a `padding-left` value.

File: **indent.html (excerpt)**

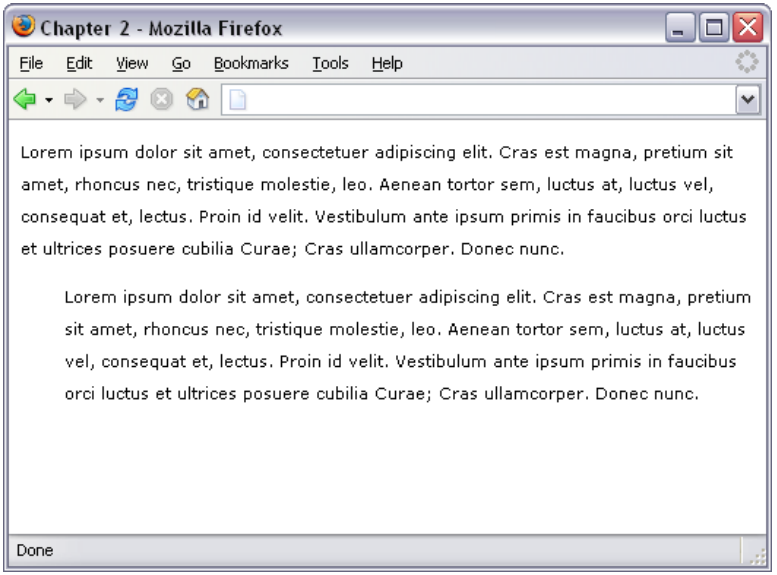
```
<p class="indent">Lorem ipsum dolor sit amet, consectetuer
  adipiscing elit. Vivamus mollis. Etiam aliquet. Ut ultrices
  justo ut arcu. Proin a purus. Fusce pharetra ultrices nibh.
  Nam erat lectus, dapibus id, congue vel, cursus a, tellus.
  Sed turpis ante, condimentum at, accumsan eget, pulvinar
  vitae, nibh.</p>
```

File: **indent.css (excerpt)**

```
.indent {
  padding-left: 30px;
}
```

You can see the indented paragraph in Figure 2.19.

Figure 2.19. Indent text using CSS.



Discussion

You should not use the HTML tag `<blockquote>` to indent text, unless the text is actually a quote. Although visual editing environments such as Macromedia Dreamweaver frequently refer to `<blockquote>` as "indent text," resist the temptation to use it for this purpose; instead, set up a CSS rule to indent the

appropriate blocks. `<blockquote>` is designed to mark up a quote, and devices such a screen readers for the visually impaired will read this text in a way that helps users understand that what they're hearing is a piece of quoted text. If you use `<blockquote>` to indent regular paragraphs, it will be very confusing for users to whom the content is read as a quote.

A One-Liner

A related technique enables the indentation of just the first line of each paragraph. To do this, use the CSS property `text-indent` applied either to the paragraph, or to a class that's applied to paragraphs you wish to display in this way.

```
p {
  text-indent: 20px;
}
```

How do I center text?

Solution

You can center text, or any other element, using the `text-align` property with a value of `center`.

File: **center.html (excerpt)**

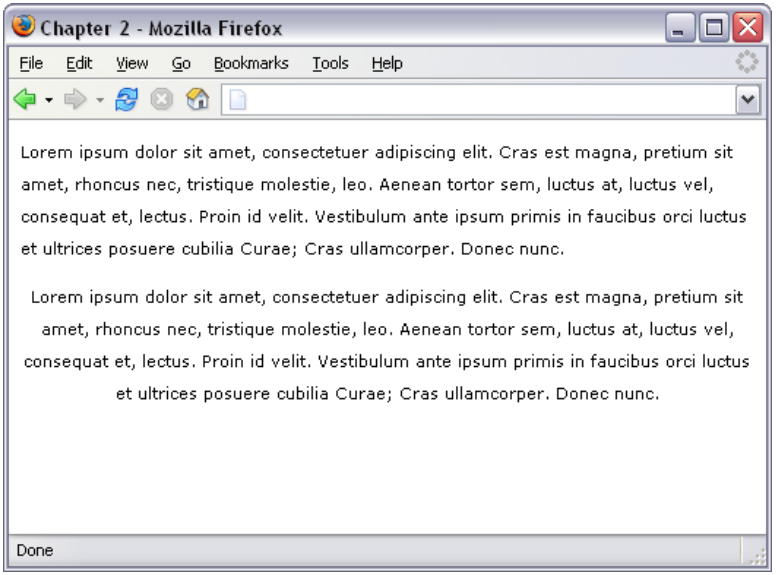
```
<p class="centered">Lorem ipsum dolor sit amet, consectetuer
  adipiscing elit. Vivamus mollis. Etiam aliquet. Ut ultrices
  justo ut arcu. Proin a purus. Fusce pharetra ultrices nibh. Nam
  erat lectus, dapibus id, congue vel, cursus a, tellus. Sed
  turpis ante, condimentum at, accumsan eget, pulvinar vitae,
  nibh.</p>
```

File: **center.css (excerpt)**

```
.centered {
  text-align: center;
}
```

The result can be seen in Figure 2.20.

Figure 2.20. Center text using `text-align`.



How do I change text to all-capitals using CSS?

Solution

You can change text to all-capitals and perform other transformations using the `text-transform` property.

File: **`uppercase.html` (excerpt)**

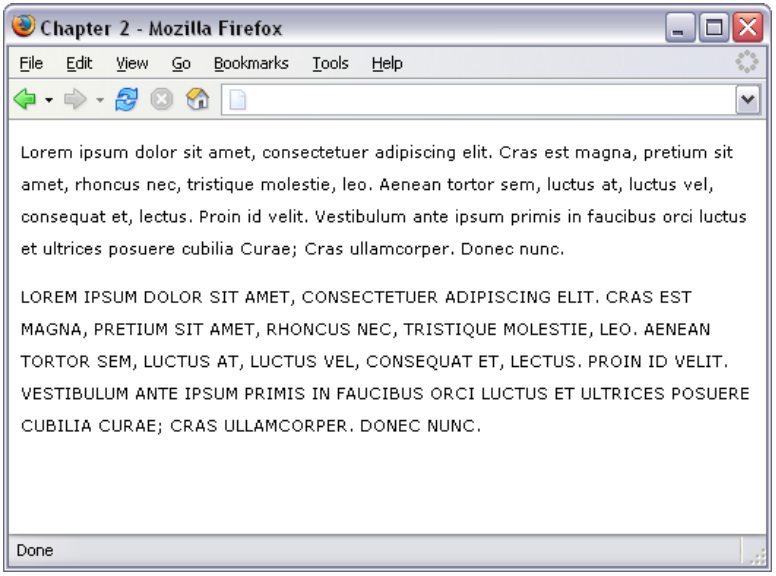
```
<p class="transform">Lorem ipsum dolor sit amet, consectetuer
  adipiscing elit. Vivamus mollis. Etiam aliquet. Ut ultrices
  justo ut arcu. Proin a purus. Fusce pharetra ultrices nibh. Nam
  erat lectus, dapibus id, congue vel, cursus a, tellus. Sed
  turpis ante, condimentum at, accumsan eget, pulvinar vitae,
  nibh.</p>
```

File: **uppercase.css (excerpt)**

```
.transform {
  text-transform: uppercase;
}
```

Note the uppercase text in Figure 2.21.

Figure 2.21. This `text-transform` makes all the letters uppercase.



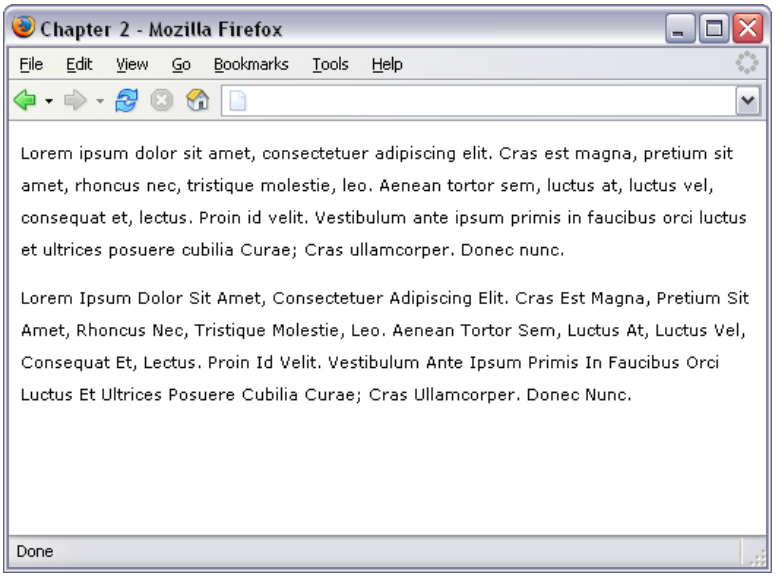
Discussion

There are other useful values for the `text-transform` property. The value `capitalize` will capitalize the first letter of each word, as shown in Figure 2.22:

File: **capitalize.css (excerpt)**

```
.transform {
  text-transform: capitalize;
}
```

Figure 2.22. The application of `text-transform` changes the appearance of the words.



Other values of the `text-transform` property are:

- `lowercase`
- `none` (the default)

How do I change or remove the bullets on list items?

Solution

You can change the style of bullets used in your lists by altering the `list-style-type` property.

File: **`listtype.html` (excerpt)**

```
<ul>
  <li>list item one</li>
  <li>list item two</li>
```

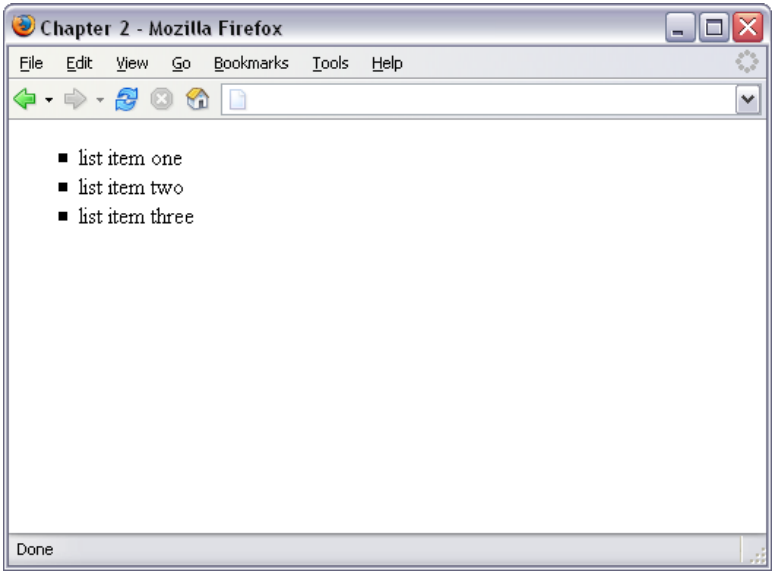
```
  <li>list item three</li>
</ul>
```

To use square bullets, as shown in Figure 2.23, set the property to `square`:

File: **listtype.css**

```
ul {
  list-style-type: square;
}
```

Figure 2.23. This list uses square bullets.



Discussion

Other values for the `list-style-type` property are:

- `disc`
- `circle`
- `decimal-leading-zero`
- `decimal`

- `lower-roman`
- `upper-roman`
- `lower-greek`
- `lower-alpha`
- `lower-latin`
- `upper-alpha`
- `upper-latin`
- `Hebrew`
- `Armenian`
- `Georgian`
- `none`

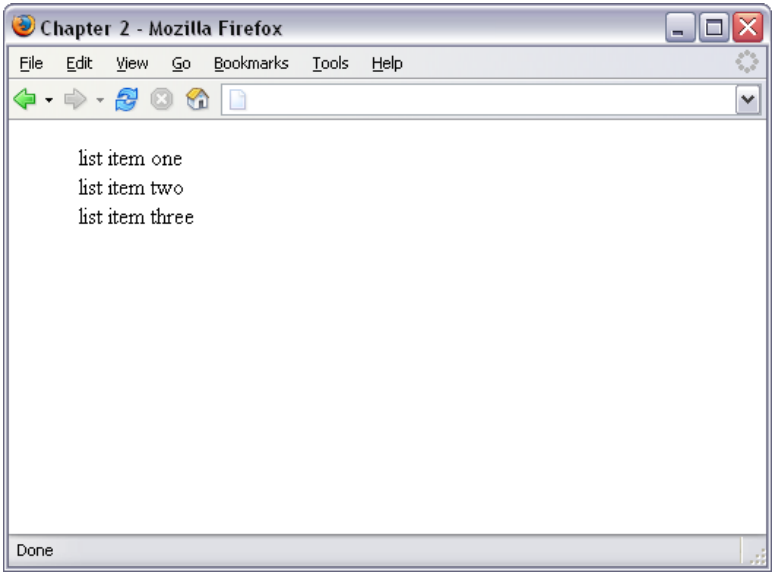
Not all of these values are supported by all browsers; those that don't provide support for a particular type will display the default bullet type instead. You can see the different types and check your browser's support for them at the CSS Test Suite for list style type.[1] Setting `list-style-type` to `none` will result in no bullets being displayed, although the list will still be indented as if the bullets were there. You can see this in Figure 2.24.

File: **listtype2.css**

```
ul {
  list-style-type: none;
}
```

[1] http://www.meyerweb.com/eric/css/tests/css2/sec12-06-02a.htm

Figure 2.24. This list displays without bullets.



How do I use an image for a list item bullet?

Solution

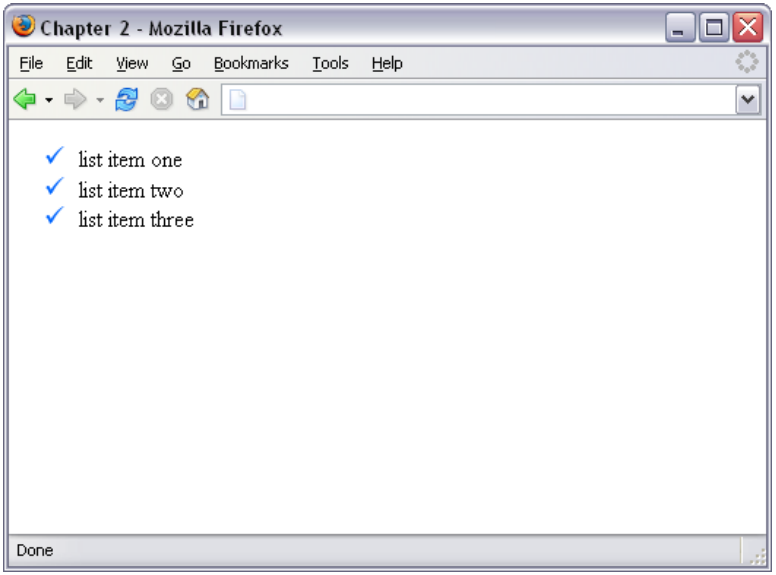
To use an image for a bullet, create your image, then use the `list-style-image` property, instead of `list-style-type`, for your bullets. This property accepts a URL, which incorporates the path to your image file as a value.

File: **listimage.css**

```
ul {
  list-style-image: url(bullet.gif);
}
```

You can see an example of an image used as a bullet in Figure 2.25.

Figure 2.25. Use an image as a list bullet.



Per Item Bullets

The `list-style-image` property actually applies to the list item (`<li>`) tags in the list, but by applying it to the list as a whole the individual items will inherit it. You do, however, have the option of setting the property on individual list items, giving individual items their own bullet images.

How do I remove the indented left margin from a list?

If you have set `list-style-type` to `none`, you may also wish to remove or decrease the default left margin that the browser sets on a list.

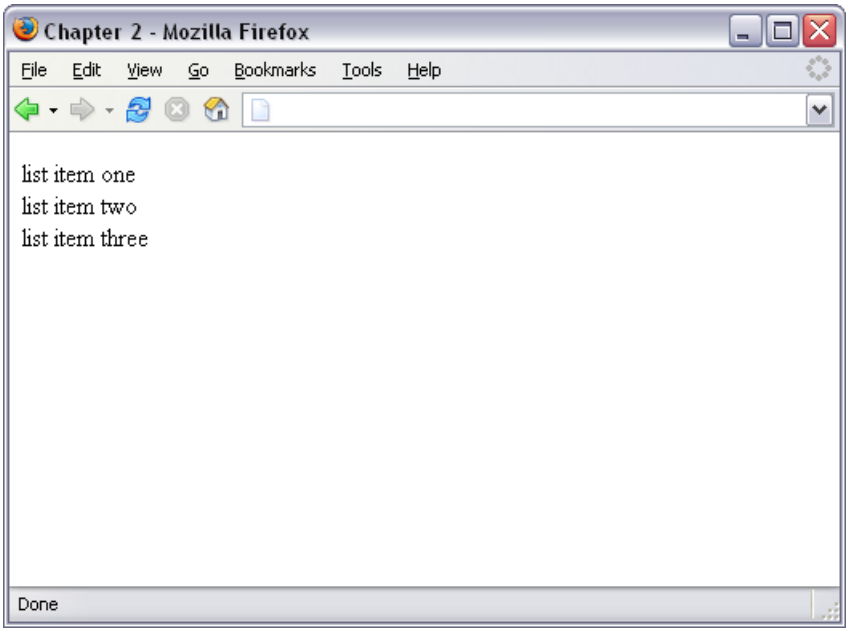
Solution

To remove the indentation entirely so that the list will align left with any other content, as shown in Figure 2.26, use the following code:

File: **listnomargin.css**

```
ul {
  list-style-type: none;
  padding-left: 0;
  margin-left: 0;
}
```

Figure 2.26. The list has no indentation or bullets.



Discussion

You can tweak the indentation after doing this. To indent the content by a few pixels, try this:

```
ul {
  list-style-type: none;
  padding-left: 5px;
  margin-left: 0;
}
```

How do I display a list horizontally?

By default, list items display as block elements; therefore, each new item will display on a new line. However, there may be times when some content on your page is, structurally speaking, a list, even though you mightn't want to display it as such. What happens if, for instance, you want to display the "list items" horizontally?

Solution

You can make a list display horizontally by altering the `display` property of the `<li>` tag to `inline`.

File: **`listinline.html` (excerpt)**

```
<ul class="horiz">
  <li>list item one</li>
  <li>list item two</li>
  <li>list item three</li>
</ul>
```

File: **`listinline.css`**

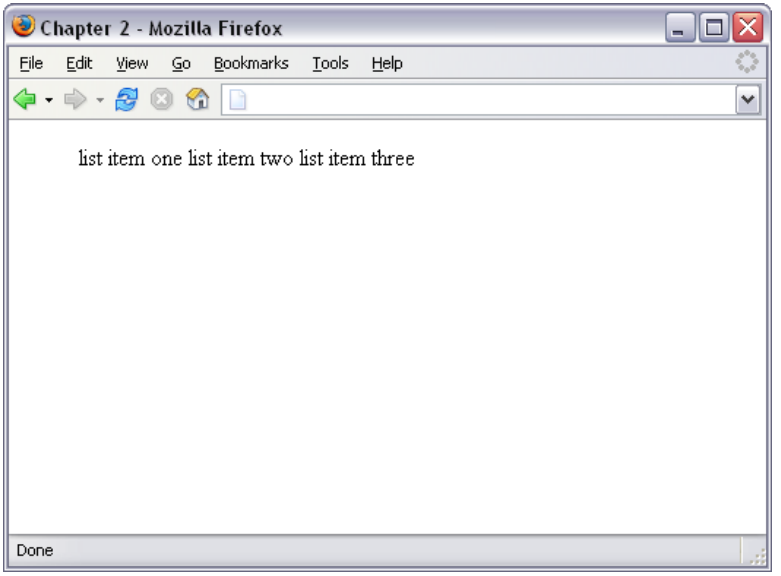
```
ul.horiz li {
  display: inline;
}
```

The result of the above code can be seen in Figure 2.27.

How do I add comments to my CSS file?

You can, and should, add comments to your CSS file. CSS files that are very simple, with just a few rules for text styling, for instance, may not require commenting. However, once you start to use large CSS files and multiple style sheets for a site, comments come in very handy! Without them, you can spend a lot of time hunting around for the right classes, pondering which class does what and in which style sheet it lives.

Figure 2.27. A list can be displayed horizontally.



Solution

CSS supports multiline C-style comments, just like JavaScript. So, to comment out an area, use the following:

```
/*
  …
*/
```

At the very least, you should add a comment at the top of each style sheet to explain what's in that style sheet.

```
/* This is the default style sheet for all text on the site */
```

How do I get rid of the page margins without adding attributes to the body tag?

Before CSS support, we would remove the default gutter between the document and the browser window by adding attributes to the `<body>` tag:

```
<body topmargin="0" leftmargin="0" marginheight="0"
    marginwidth="0">
```

Solution

The above attributes of the body tag are now deprecated. They should be replaced by CSS defined for the `<body>` tag.

```
body {
  margin: 0;
  padding: 0;
}
```

Opera Sings its Own Tune

Some versions of Opera apply `margin` and `padding` to the `<html>` element instead of the `<body>` tag. Therefore, to support Opera, you'll need to use this code:

```
html, body {
  margin: 0;
  padding: 0;
}
```

Summary

In this chapter, we've covered some of the more common questions asked by those who are relatively new to CSS: questions that relate to styling and manipulating text on the page. By combining these techniques, you can create attractive effects that will degrade appropriately for those who aren't using a browser that supports CSS.

3

CSS and Images

Given many of the designs favored by the CSS purists, you'd be forgiven for thinking that the image is soon to be a thing of the past, eschewed in favor of clean, standards-compliant, CSS-formatted, text-based design. However, while sites that rely entirely on sliced-up images are beginning to look a little dated in comparison with the clean simplicity of the CSS layout "style," well-placed images can bring an otherwise commonplace design to life. And, as designers begin to push the boundaries of what can be achieved with standards-compliant semantic markup, sites that have managed to combine semantics and beauty are becoming much more commonplace.

Working with images in CSS requires a few simple skills, which can then be combined to create interesting effects. The solutions in this chapter demonstrate the basic concepts of working with images while answering some common questions. However, as with most of the solutions in this book, don't be afraid to experiment and see what unique effects you can create.

How do I add a border to images?

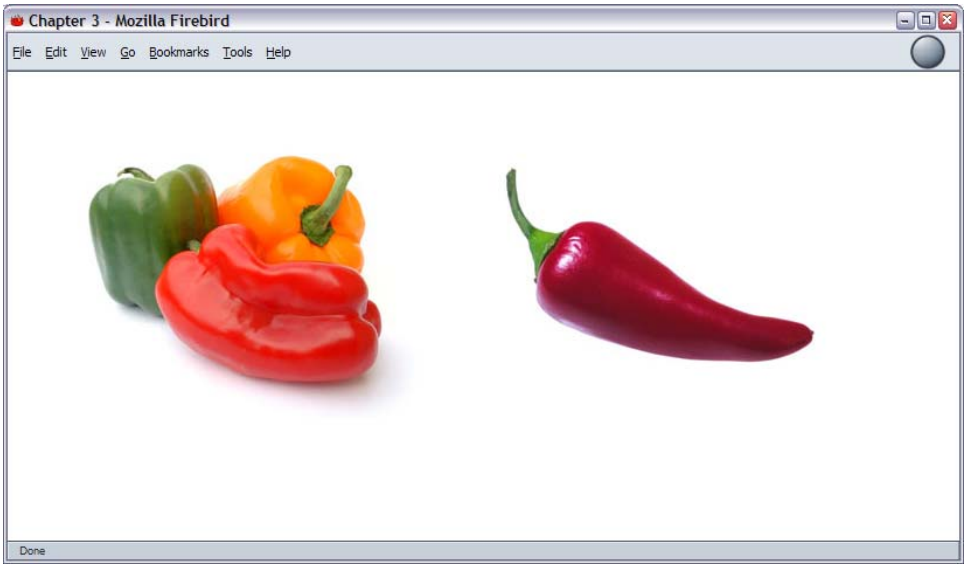
Photographic images, perhaps used to illustrate an article, or as a display in a photo album, look neat when bordered with a thin line. However, opening each shot in a graphics program to add the border is a time consuming process and,

if you ever need to change that border's color or thickness, you'll need to go through the same arduous process all over again to create the desired images.

Solution

Adding a border to an image is a simple procedure using CSS. There are two images in the document displayed in Figure 3.1.

Figure 3.1. Images are displayed in a Web browser.



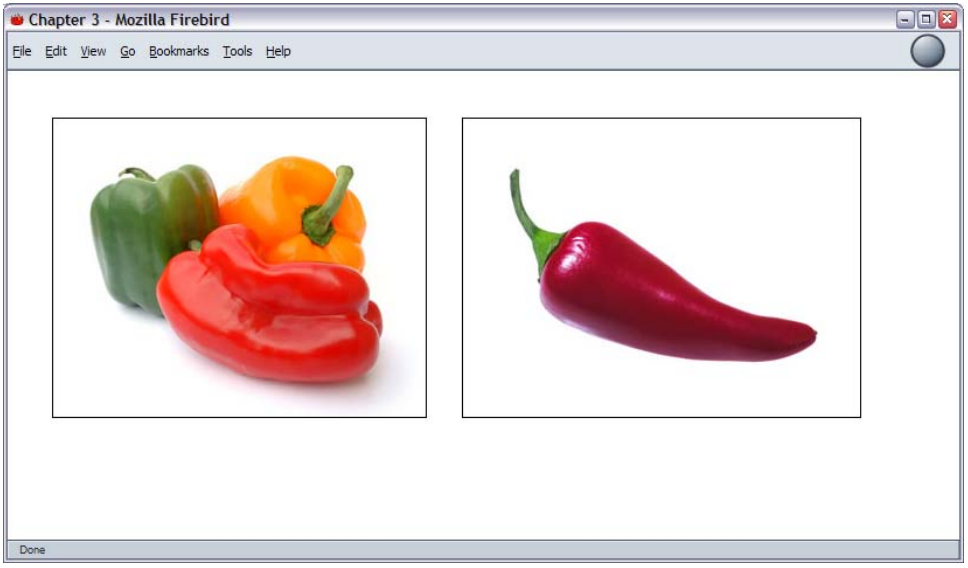

```
img {
  border-width: 1px;
  border-style: solid;
  border-color: #000000;
}
```

This code could also be written as follows:

```
img {
  border: 1px solid #000000;
}
```

In Figure 3.2, you can see the effect this has on the images.

Figure 3.2. The images look neater once the CSS border is applied.



Your layout probably contains images to which you don't really want to apply a permanent black border. You can create a CSS class for the border and apply it to selected images as required.

```
.imgborder {
  border: 1px solid #000000;
}
```

```
<img src="myfish.jpg" alt="My Fish" class="imgborder" />
```

If you have a whole selection of images—such as a photograph album—on the page, you could set borders on all the images within a container.

File: **imageborders.css (excerpt)**

```
#album img {
  border: 1px solid #000000;
}
```

This approach will save you having to add the class to each individual image within the container.

How do I use CSS to replace the deprecated HTML `border` attribute on images?

If you're anything like me, adding `border="0"` to images that are links to other documents is probably something you do almost automatically. Using the `border` attribute of the `<img>` tag is the way in which we all learned to ensure that an ugly blue border didn't appear around our navigation buttons and so on. However, `border` has been deprecated in the current versions of HTML and XHTML.

Solution

Just as you can create a border, so you can remove one. Setting an image's `border` property to `none` will remove those ugly borders.

```
img {
  border: none;
}
```

How do I set a background image for my page with CSS?

Before CSS, backgrounds were added using the `background` attribute of the `<body>` tag. This attribute is now deprecated and replaced by CSS properties.

Solution

File: **backgrounds.css**

```
body {
  background-color: #ffffff;
  background-image: url(peppers_bg.jpg);
  background-repeat: no-repeat;
}
```

The above rules add the image `peppers_bg.jpg` as a background to any page to which this style sheet is attached, as shown in Figure 3.3.

Figure 3.3. A standard image is displayed as a background image.

Discussion

The CSS property `background-image` enables you to specify as a URL the location of a background image. By default, the background will tile as shown in Figure 3.4.

As we don't want multiple peppers in this example, we need to prevent this image from tiling. To do so, we set the property `background-repeat` to `no-repeat`. Other values for `background-repeat` are:

`repeat` This default value causes the image to tile across and down the page, as shown in Figure 3.4.

`repeat-x` The image tiles across the page in a single row of images, as shown in Figure 3.5.

`repeat-y` The image tiles down the page in a single row, as shown in Figure 3.6.

Figure 3.4. A background image is set to repeat by default.

Figure 3.5. The background image is set to `repeat-x`.

Figure 3.6. The background image is set to `repeat-y`.

How do I position my background image?

By default, if you add a single, non-repeating background image to the page, it will appear in the top left corner of the viewport. If you have set the background to tile in any direction, the first image will appear at that location, and will tile from that point. However, it is also possible for the image to be displayed anywhere else on the page.

Solution

To position the image, we need to use the CSS property `background-position`.

File: **`backgroundposition.css` (excerpt)**

```
body {
  background-color: #FFFFFF;
  background-image: url(peppers_bg.jpg);
  background-repeat: no-repeat;
  background-position: center center;
}
```

The above CSS will center the image on the page as shown in Figure 3.7.

Figure 3.7. The `background-position` property can be used to center the image.

Discussion

The `background-position` property can take as values keywords, percentage values, or values in units, such as pixels.

Keywords

In the example above, we used keywords to specify that the background image should be displayed in the center of the page.

File: **backgroundposition.css (excerpt)**

```
  background-position: center center;
```

Keyword combinations that you can use are:

- `top left`
- `top center`
- `top right`
- `center left`
- `center center`
- `center right`
- `bottom left`
- `bottom center`
- `bottom right`

If you only specify one of the values, the other will default to center.

```
background-position: top;
```

The above code is the same as the following:

```
background-position: top center;
```

Percentage Values

To ensure more accurate image placement, you can specify the values as percentages. This is particularly useful in a liquid layout where other page elements are specified in percentages so that they resize in accordance with the user's screen resolution and size.

```
background-position: 30% 80%;
```

The first of the two percentages included here refers to the horizontal position; the second is the vertical position. Percentages are taken from the top left corner, with `0% 0%` placing the top left corner of the image against the top left corner of the browser window, and `100% 100%` placing the bottom right corner of the image against the bottom right corner of the window.

As with keywords, a default value comes into play if you only specify one value. That default is 50%.

```
background-position: 30%;
```

The above code is, therefore, the same as that shown below:

```
background-position: 30% 50%;
```

Unit Values

You can set the values using any CSS units, such as pixels or ems.

```
background-position: 20px 20px;
```

Just as with percentages, the first value dictates the horizontal position, while the second dictates the vertical. But unlike percentages, the measurements directly control the position of the top left corner of the background image.

You can mix units with percentages and, if you only specify one value, the second will default to 50%.

How do I make a background image that stays still while the text moves when the page is scrolled?

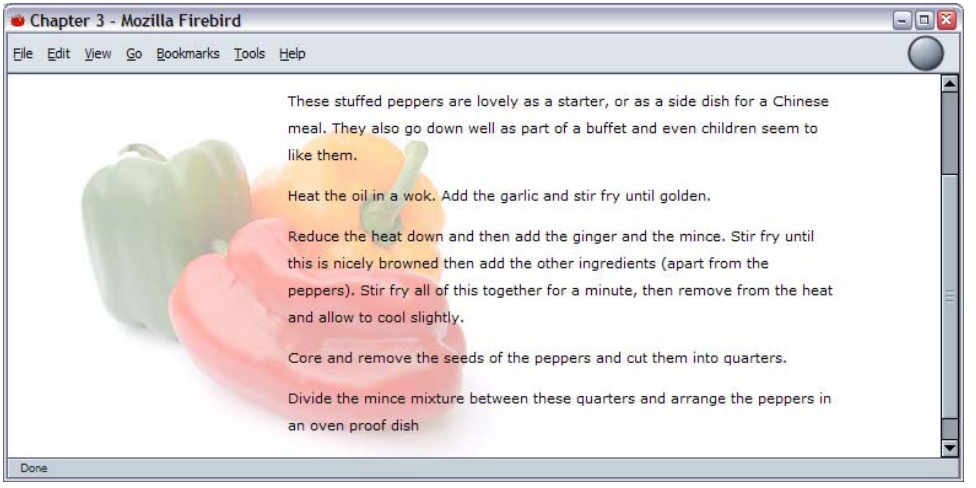
You may have seen sites where the background image remains static while the content scrolls over it. This effect is achieved using CSS.

Solution

File: **backgroundfixed.html (excerpt)**

```
body {
  background-color: #FFFFFF;
  background-image: url(peppers_bg.jpg);
  background-repeat: no-repeat;
  background-position: 20px 20px;
  background-attachment: fixed;
}
```

Figure 3.8. The background image is fixed and doesn't scroll off the page with the content.



We can use the property `background-attachment` with a value of `fixed` to fix the background so that it doesn't move with the content, as shown in Figure 3.8.

Discussion

In this solution, we're using several CSS properties to add our image to the background, position it, and detail how it behaves when the document is scrolled. It is also possible to use a shorthand method to write out this information, applying the CSS property, `background`. This property allows you to declare `background-color`, `background-image`, `background-repeat`, `background-attachment` and background-position in a single property declaration.

Take, for example, the CSS declarations shown below:

File: **backgroundfixed.css (excerpt)**

```
body {
  background-color: #FFFFFF;
  background-image: url(peppers_bg.jpg);
  background-repeat: no-repeat;
  background-attachment: fixed;
  background-position: 20px 20px;
}
```

These could also be written as follows:

```
body {
  background: #ffffff url(peppers_bg.jpg) no-repeat fixed 20px
    20px;
}
```

How do I set background images for other elements?

In this chapter, we've already looked at setting background images for the document. However, background images can be used on other elements, too.

Solution

File: **backgrounds2.css (excerpt)**

```
#smallbox {
  background-image: url(mini_chilli.jpg);
```

```
  background-repeat: repeat-y;
  float: left;
  padding-left: 60px;
  margin-right: 20px;
  width: 220px;
  border: 1px solid #F5C9C9;
}
```

The red chillies appearing down the left-hand side of the box in Figure 3.9 comprise a background that's created by tiling the background image on the y-axis. The background image shown here is applied to a `<div>`, along with other rules, to create the box.

Figure 3.9. The chilli image is a tiled background image.

Discussion

Background images can be used on any page element. You can apply a background to a heading, as shown in Figure 3.10:

File: **backgrounds2.html (excerpt)**

```
<h1 class="header">Chinese style stuffed peppers</h1>
```

File: **backgrounds2.css (excerpt)**

```
.header {
  background-image: url(dotty.gif);
  width: 400px;
  height: 30px;
  padding: 6px 6px 4px 8px;
  color: #B22222;
  background-color: transparent;
}
```

Figure 3.10. The heading has a background image.

You can even apply a background to links, which can assist you in making some interesting effects, as in Figure 3.11.

File: **backgrounds2.css (excerpt)**

```
a:link, a:visited {
  color: #B22222;
  background-color: transparent;
  font-weight: bold;
}
a:hover {
  background-image: url(dotty.gif);
  text-decoration: none;
}
```

Figure 3.11. A background image is applied to the link on hover.

How do I place text on top of an image?

In the bad old pre-CSS days, the only way to add text to an image was to add the text in your graphics program! CSS provides far better means to achieve this effect.

Solution

The easiest way to layer text on top of an image is to make the image a background image. The chilli image beneath the list of ingredients shown in Figure 3.12 is added using the following properties:

File: **backgrounds3.css (excerpt)**

```
background-image: url(chilli2.jpg);
background-repeat: no-repeat;
```

Figure 3.12. The chilli image is a background image.

Discussion

Using CSS to lay text over images has many advantages over the process of simply adding the text to the image in a graphics program.

First, it's more difficult to change text that is part of a graphic. To do so, you need to find the original graphic, edit it in a graphics program, and upload it every time you need to change the text.

Second, text is far more accessible as text content than as part of an image. Browsers that don't support images will still be able to read text that has been added as CSS. This also benefits your search engine ranking—search engines can't index text that's part of an image, but will see text laid over an image as regular text, and index it accordingly.

How do I add more than one background image to my document?

It isn't possible to add more than one background image to your document. However, it is possible to give the effect of multiple background images by applying a background to elements that are nested, such as the `<html>` tag and the `<body>` tag.

File: **backgrounds4.css (excerpt)**

```
html {
  background-image: url(mini_chilli.jpg);
  background-repeat: repeat-y;
  background-position: 2px 2px;
  background-attachment: fixed;
}
body {
  background-image: url(peppers_bg.jpg);
  background-repeat: no-repeat;
  background-position: 80px 20px;
}
```

This effect can be seen in Figure 3.13.

Figure 3.13. Background images are applied to the `<html>` and `<body>` elements.

Discussion

This simple example can form the basis of more complex effects using multiple background images. As you've seen through the examples in this chapter, a background image can be applied to any element on the page. The careful and creative use of images in this way can create many interesting visual effects while maintaining the accessibility of the document (as the background images will not interfere with the document's structure).

Many of the entries in the CSS Zen garden[1] rely on such careful use of background images to achieve their layouts.

Summary

This chapter has explained the answers to some common image-related questions. There are, of course, going to be image-related questions all through this book.

[1] http://www.csszengarden.com/

In particular, the chapters that deal with positioning will explore the positioning of images along with other elements on the page.

4

Navigation

Unless you limit yourself to one-page Web sites, you will need to design navigation. In fact, navigation is among the most important parts of any Web design, and requires a great deal of thought if visitors are to get around your site easily.

Making site navigation easy is one area in which CSS really comes into its own. Older methods of creating navigation tended to rely on lots of images, nested tables and JavaScript—all of which can seriously affect the usability and accessibility of a site. If your site cannot be navigated using a device that doesn't support JavaScript, for example, not only are you blocking users who have turned JavaScript off, you're also locking out text-only devices such as screen readers, and search engine robots—they'll never get past your home page to index the content of your site. If your design clients don't care about accessibility, tell them their clunky menu is stopping them from achieving a decent search engine ranking!

CSS allows you to create attractive navigation that is, in reality, no more than text—text that can be marked up in such a way as to ensure that it's both accessible and understandable by all those who can't physically see your design, but still want to get to your content. In this chapter, we'll look at a variety of solutions. Some are suited to implementation on an existing site, to make it load more quickly, and boost its accessibility by replacing an old-fashioned, image-based navigation. Others are more suited to being incorporated within a pure CSS layout.

How do I replace image-based navigation with CSS?

Creating an image as a navigation "button" is still a common way to design the navigation for a site. The images usually contain text to show where each navigation item leads. There are a variety of problems associated with using images for navigation buttons:

- To add a new item, a new image must be created. If, at this point, you discover that you've lost the original Photoshop file, you must recreate the whole navigation from scratch!
- Imagine your navigation is created dynamically, perhaps using database content. While it's possible to create new images on the fly, someone will have to write a whole lot more code to integrate them onto every page!
- Users who have turned images off, or use devices such as screen readers, will be unable to read the text within the button.
- Additional images slow page load times.

Solution

Using CSS, you can replace much image-based navigation with text styled using CSS. The following CSS and HTML creates a simple navigation menu by styling the cells of a table and the links within them.

File: **replaceimages.html**

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
    "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
<title>Replace images</title>
<meta http-equiv="content-type"
    content="text/html; charset=iso-8859-1" />
<link rel="stylesheet" type="text/css" href="replaceimages.css" />
</head>
<body>
<table id="navigation">
  <tr>
    <td>
```



```
      <a href="#">Recipes</a>
    </td>
  </tr>
  <tr>
    <td>
      <a href="#">Contact Us</a>
    </td>
  </tr>
  <tr>
    <td>
      <a href="#">Articles</a>
    </td>
  </tr>
  <tr>
    <td>
      <a href="#">Buy Online</a>
    </td>
  </tr>
</table>
</body>
</html>
```

File: **replaceimages.css**

```
#navigation {
  width: 180px;
  padding: 0;
  margin: 0;
  border-collapse: collapse;
}
#navigation td {
  height: 26px;
  border-bottom: 2px solid #460016;
  background-color: #FFDFEA;
  color: #460016;
}
#navigation a:link, #navigation a:visited {
  margin-left: 12px;
  color: #460016;
  background-color: transparent;
  font-size: 12px;
  font-family: Arial, Helvetica, sans-serif;
  text-decoration: none;
  font-weight: bold;
}
```

This technique could be used to ease the maintenance of an existing site by allowing the addition of new menu items without the need to create new images, and by reducing load times.

Discussion

CSS can be used to create attractive navigation systems through the simple styling of plain text. Figure 4.1 shows a menu created by inserting images into table cells—a common way to create a site menu.

Figure 4.1. Images are commonly used to create navigation.

The markup for this table is as follows.

```
<table width="180" cellpadding="0" cellspacing="0">
  <tr>
    <td>
      <a href="#"><img src="images/nav1.gif" width="180"
            height="28" alt="Recipes" border="0" /></a>
    </td>
  </tr>
  <tr>
    <td>
      <a href="#"><img src="images/nav2.gif" width="180"
            height="28" alt="Contact Us" border="0" /></a>
    </td>
  </tr>
  <tr>
    <td>
```

```
      <a href="#"><img src="images/nav3.gif" width="180"
            height="28" alt="Articles" border="0" /></a>
    </td>
  </tr>
  <tr>
    <td>
      <a href="#"><img src="images/nav4.gif" width="180"
            height="28" alt="Buy Online" border="0" /></a>
    </td>
  </tr>
</table>
```

By removing the images and replacing them with text for each navigation item, we immediately make our code smaller and the page more accessible. However, simply using plain text doesn't do much for the appearance of the page, as you can see in Figure 4.2.

Figure 4.2. The navigation system looks bland when images are removed.

We can use CSS to recreate the look of this menu without the images. First, let's give the navigation table an ID. This will enable us to identify it within the document, and create CSS selectors for the elements within that table.

We can create CSS for the ID `navigation`, which means that we can also lose the attributes from the `<table>` tag.

File: **`replaceimages.html`** **(excerpt)**

```
<table id="navigation">
```

Here's the CSS that controls how the table as a whole looks:

File: **replaceimages.css (excerpt)**

```
#navigation {
  width: 180px;
  padding: 0;
  margin: 0;
  border-collapse: collapse;
}
```

Setting the `border-collapse` property to `collapse` causes the cells of the table to stick together, with only a single border between cells. The default behavior of table cells is for each cell to have its own border, with additional margins between cells.

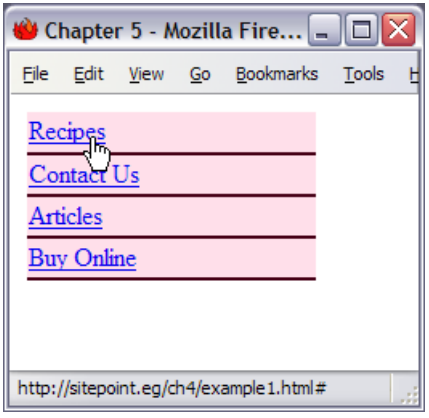
We now need to style the `<td>` tags in the table. We want to give the cells the desired background color, and to add a bottom border to each:

File: **replaceimages.css (excerpt)**

```
#navigation td {
  height: 26px;
  border-bottom: 2px solid #460016;
  background-color: #FFDFEA;
  color: #460016;
}
```

It's already looking pretty good, as you can see in Figure 4.3.

Figure 4.3. The new styles are applied to the navigation.



Now we must create CSS for the links within the table cells. We need to set a left margin to move the text away from the edge of the cell; to define a color, size, font family and weight; and to remove the underline from the link.

File: **replaceimages.css (excerpt)**

```
#navigation a:link, #navigation a:visited {
  margin-left: 12px;
  color: #460016;
  background-color: transparent;
  font-size: 12px;
  font-family: Arial, Helvetica, sans-serif;
  text-decoration: none;
  font-weight: bold;
}
```

Figure 4.4 shows the finished effect.

Figure 4.4. The navigation is created using CSS instead of images.

How do I style a structural list as a navigation menu?

The previous example illustrated the use of CSS to create navigation elements within a table. That method proves very useful where you are retrofitting an existing site to improve its accessibility and load time but, for new sites, you're likely to be trying to avoid using tables for layout, or using them only where it's absolutely necessary. Therefore, a navigation solution that doesn't require the

use of tables is useful; by eradicating the table tags, you'll find that your page contains far less markup as well.

Solution

Navigation is simply a list of places that users can visit on the site. Therefore, a list is the ideal way to mark up your navigation. The navigation in Figure 4.5 is created using CSS to style a list.

Figure 4.5. Navigation can be created by styling a list.

File: **listnav1.html**

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
    "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
<title>Lists as navigation</title>
<meta http-equiv="content-type"
    content="text/html; charset=iso-8859-1" />
<link rel="stylesheet" type="text/css" href="listnav1.css" />
</head>
<body>
<div id="navigation">
  <ul>
    <li><a href="#">Recipes</a></li>
    <li><a href="#">Contact Us</a></li>
    <li><a href="#">Articles</a></li>
    <li><a href="#">Buy Online</a></li>
  </ul>
```

```
</div>
</body>
</html>
```

File: **listnav1.css**

```
#navigation {
  width: 200px;
  font-family: Arial, Helvetica, sans-serif;
}
#navigation ul {
  list-style: none;
  margin: 0;
  padding: 0;
}
#navigation li {
  border-bottom: 1px solid #ED9F9F;
}
#navigation li a {
  display: block;
  padding: 5px 5px 5px 0.5em;
  border-left: 12px solid #711515;
  border-right: 1px solid #711515;
  background-color: #B51032;
  color: #FFFFFF;
  text-decoration: none;
}
```

Discussion

To create navigation based on a list, first create your list, placing each navigation link inside an `<li>` tag.

File: **listnav1.html (excerpt)**

```
  <ul>
    <li><a href="#">Recipes</a></li>
    <li><a href="#">Contact Us</a></li>
    <li><a href="#">Articles</a></li>
    <li><a href="#">Buy Online</a></li>
  </ul>
```

Wrap this list in a `<div>` with an appropriate ID.

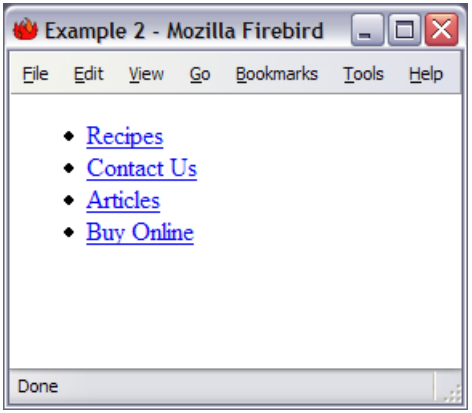
File: **listnav1.html (excerpt)**

```
<div id="navigation">
  <ul>
```

```
    <li><a href="#">Recipes</a></li>
    <li><a href="#">Contact Us</a></li>
    <li><a href="#">Articles</a></li>
    <li><a href="#">Buy Online</a></li>
  </ul>
</div>
```

As Figure 4.6 shows, this looks pretty ordinary with the browser's default styles applied.

Figure 4.6. An unstyled list is very basic.



The first thing we need to do is style the container in which the navigation sits, in this case, `#navigation`:

File: **listnav1.css (excerpt)**

```
#navigation {
  width: 200px;
  font-family: Arial, Helvetica, sans-serif;
}
```

I have given `#navigation` a width and a font family. If this were part of a CSS page layout, I'd probably add some positioning information to this ID as well.

Next, we style the list:

File: **listnav1.css (excerpt)**

```
#navigation ul {
  list-style: none;
  margin: 0;
```



```
  padding: 0;
}
```

As Figure 4.7 illustrates, the above rule removes list bullets and the indented margin that browsers apply, by default, when displaying a list.

Figure 4.7. The list's indentation and bullets have been removed.

The next step is to style the `<li>` tags within `#navigation`, to give them a bottom border:

File: **listnav1.css (excerpt)**

```
#navigation li {
 border-bottom: 1px solid #ED9F9F;
}
```

Finally, we style the link itself:

File: **listnav1.css (excerpt)**

```
#navigation li a:link, #navigation li a:visited  {
  display: block;
  padding: 5px 5px 5px 0.5em;
  border-left: 12px solid #711515;
  border-right: 1px solid #711515;
  background-color: #B51032;
  color: #FFFFFF;
  text-decoration: none;
}
```

Most of the work is done here, creating CSS rules to add left and right borders, removing the underline, and so on. The first property declaration in this rule sets the `display` property to `block`. This causes the link to display as a block element, meaning that the whole area of the navigation "button" is active when you hold your cursor over it—the same effect you'd see if you used an image for the navigation.

How do I use CSS to create rollover navigation without images or JavaScript?

Site navigation often features a rollover effect—when a user holds their cursor over the button, it displays a new image, creating a "highlight" effect. To achieve this effect using image-based navigation, you'd need to use two images and JavaScript.

Solution

Using CSS for your navigation makes the creation of attractive rollover effects far simpler than it would be if you used images. The CSS rollover is created using the `:hover` pseudo-class selector, just as you'd use to style a hover state for your links.

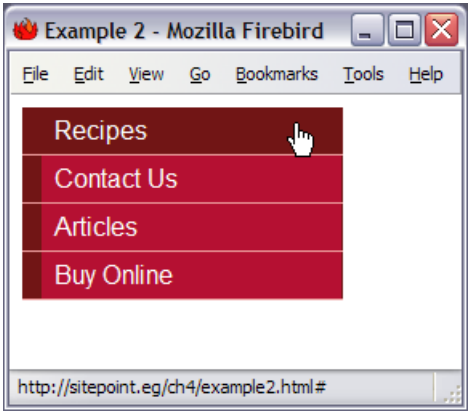
Using the above list navigation example, we could add the following markup to create a rollover effect:

File: **`listnav2.css` (excerpt)**

```
#navigation li a:hover {
  background-color: #711515;
  color: #FFFFFF;
}
```

Figure 4.8 shows what the menu looks like with the mouse cursor over the first menu item.

Figure 4.8. The CSS navigation shows a rollover effect.



Discussion

The CSS we used to create this effect is very simple. You can create hover states for heavily-styled links just as you can for standard links. In this example, I simply changed the background color to make it the same as the left hand border; however, you could alter the background, text and border color to create interesting effects for the navigation.

Hover Here? Hover There!

In Mozilla, you can apply the `:hover` pseudo-selector to any element you like, but in Internet Explorer it can only be applied to links.

Can I use CSS and lists to create a navigation system with sub-navigation?

The previous examples in this chapter have assumed that you only have one level of navigation to display. Sometimes more than one level is necessary—but is it possible using CSS styled lists?

Solution

A list with a sub-list is the perfect way to display navigation with sub-navigation. The two levels will be easy to understand when marked up in this way, even in browsers that don't support CSS.

To produce multilevel navigation, we can edit the above example single-level navigation to add a nested list and some more CSS:

File: **listnav_sub.html**

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
    "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
<title>Lists as navigation</title>
<meta http-equiv="content-type"
    content="text/html; charset=iso-8859-1" />
<link rel="stylesheet" type="text/css" href="listnav_sub.css" />
</head>
<body>
<div id="navigation">
  <ul>
    <li><a href="#">Recipes</a>
      <ul>
        <li><a href="#">Starters</a></li>
        <li><a href="#">Main Courses</a></li>
        <li><a href="#">Desserts</a></li>
      </ul>
    </li>
    <li><a href="#">Contact Us</a></li>
    <li><a href="#">Articles</a></li>
    <li><a href="#">Buy Online</a></li>
  </ul>
</div>
</body>
</html>
```

File: **listnav_sub.css**

```
#navigation {
  width: 200px;
  font-family: Arial, Helvetica, sans-serif;
}
#navigation ul {
  list-style: none;
```

```
  margin: 0;
  padding: 0;
}
#navigation li {
  border-bottom: 1px solid #ED9F9F;
}
#navigation li a:link, #navigation li a:visited  {
  display: block;
  padding: 5px 5px 5px 0.5em;
  border-left: 12px solid #711515;
  border-right: 1px solid #711515;
  background-color: #B51032;
  color: #FFFFFF;
  text-decoration: none;
}
#navigation li a:hover {
  background-color: #711515;
  color: #FFFFFF;
}
#navigation ul ul {
  margin-left: 12px;
}
#navigation ul ul li {
  border-bottom: 1px solid #711515;
  margin:0;
}
#navigation ul ul a:link, #navigation ul ul a:visited {
  background-color: #ED9F9F;
  color: #711515;
}
#navigation ul ul a:hover {
  background-color: #711515;
  color: #FFFFFF;
}
```

The result is shown in Figure 4.9.

Discussion

Nested lists are a perfect way to describe the navigation that we're using here. The first list contains the main sections of the site; the sub-list under "Recipes" shows the sub-sections within the Recipes category. Even without CSS, the structure of this list is still apparent and comprehensible, as you can see in Figure 4.10.

Figure 4.9. This CSS list navigation contains sub-navigation.

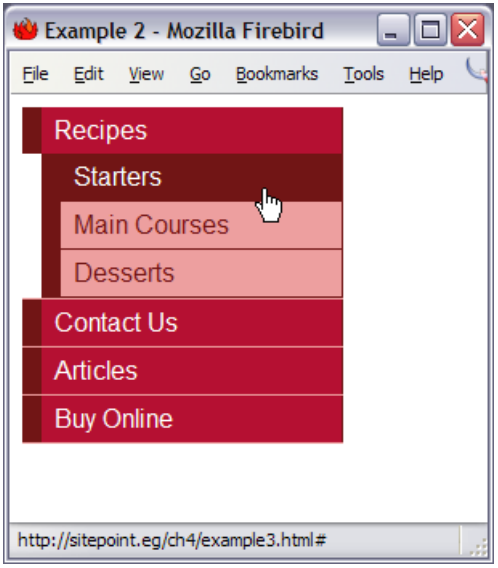
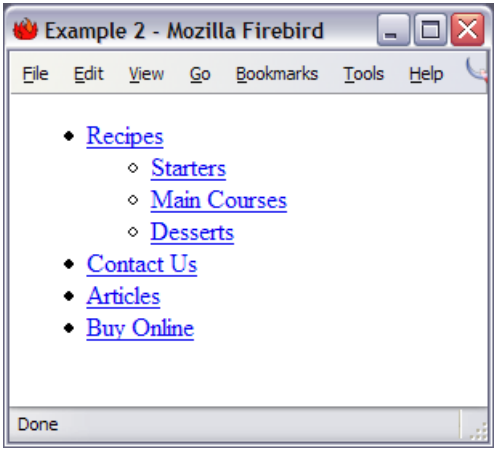


Figure 4.10. The navigation makes sense without the CSS.

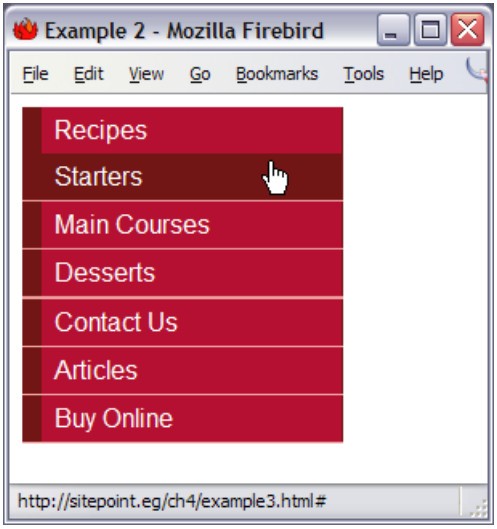


The HTML that marks up this list simply nests the sub-list inside the `<li>` tag of the appropriate main item.


```
<ul>
  <li><a href="#">Recipes</a>
    <ul>
      <li><a href="#">Starters</a></li>
      <li><a href="#">Main Courses</a></li>
      <li><a href="#">Desserts</a></li>
    </ul>
  </li>
  <li><a href="#">Contact Us</a></li>
  <li><a href="#">Articles</a></li>
  <li><a href="#">Buy Online</a></li>
</ul>
```

Without any changes to the CSS, the menu will now display as shown in Figure 4.11, where the `<li>` tags inherit the styles of the main menu.

Figure 4.11. The sub-list has taken on the styles of the main navigation.



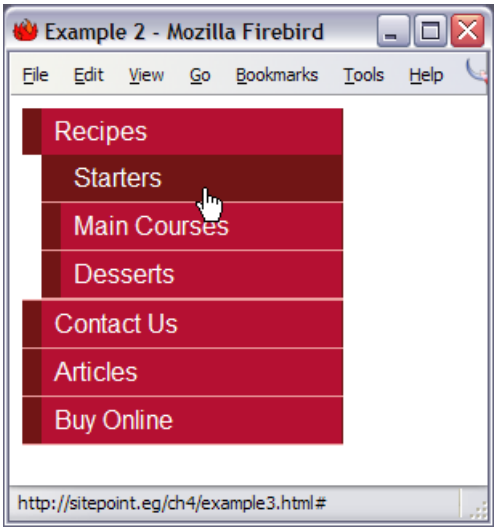
Let's add CSS for the nested list to show visually that it is a submenu, and not part of the main navigation.

File: **listnav_sub.css (excerpt)**

```
#navigation ul ul {
  margin-left: 12px;
}
```

The above will indent the `subnav` so that it's in line with the end of the border on the main menu, as you can see in Figure 4.12.

Figure 4.12. Indent the sub-navigation.



Add some simple styles to the `<li>` and `<a>` tags within the nested list to complete the effect:

File: **listnav_sub.css (excerpt)**

```
#navigation ul ul li {
  border-bottom: 1px solid #711515;
  margin:0;
}
#navigation ul ul a:link, #navigation ul ul a:visited {
  background-color: #ED9F9F;
  color: #711515;
}
#navigation ul ul a:hover {
  background-color: #711515;
  color: #FFFFFF;
}
```

How do I make a horizontal menu using CSS and lists?

All the examples we've seen in this chapter have dealt with vertical navigation—the kind of navigation that will most likely be found in a column to the left or right of a site's main content area. However, site navigation is also commonly found as a horizontal menu close to the top of the document.

Solution

As shown in Figure 4.13, this type of menu can be created using CSS-styled lists. The `<li>` tags must be set to display inline so that each list item does not move onto the next line.

Figure 4.13. Use CSS to create horizontal list navigation.

File: **listnav_horiz.html**

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
    "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
<title>Lists as navigation</title>
<meta http-equiv="content-type"
    content="text/html; charset=iso-8859-1" />
<link rel="stylesheet" type="text/css" href="listnav_horiz.css" />
</head>
<body>
<div id="navigation">
  <ul>
    <li><a href="#">Recipes</a></li>
    <li><a href="#">Contact Us</a></li>
```

```
    <li><a href="#">Articles</a></li>
    <li><a href="#">Buy Online</a></li>
  </ul>
</div>
</body>
</html>
```

File: **listnav_horiz.css**

```
#navigation {
  font-family: Arial, Helvetica, sans-serif;
  font-size: .9em;
}
#navigation ul {
  list-style: none;
  margin: 0;
  padding: 0;
  padding-top: 4px;
}
#navigation li {
  display: inline;
}
#navigation a:link, #navigation a:visited {
  padding: 3px 10px 2px 10px;
  color: #FFFFFF;
  background-color: #B51032;
  text-decoration: none;
  border: 1px solid #711515;
}
#navigation a:hover {
  color: #FFFFFF;
  background-color: #711515;
}
```

Discussion

To create the horizontal navigation, we start with a list that's identical to the one we created for our vertical list menu.

File: **listnav_horiz.html (excerpt)**

```
<div id="navigation">
  <ul>
    <li><a href="#">Recipes</a></li>
    <li><a href="#">Contact Us</a></li>
    <li><a href="#">Articles</a></li>
    <li><a href="#">Buy Online</a></li>
```

```
  </ul>
</div>
```

We style the ID `navigation` to give some basic font information, as we did with the vertical navigation. In a CSS layout, this ID would probably also contain some properties to determine the navigation's position on the page.

File: **listnav_horiz.css (excerpt)**

```
#navigation {
  font-family: Arial, Helvetica, sans-serif;
  font-size: .9em;
}
```

In styling the `<ul>`, we remove the list bullets and default indentation applied to the list by the browser.

File: **listnav_horiz.css (excerpt)**

```
#navigation ul {
  list-style: none;
  margin: 0;
  padding: 0;
  padding-top: 4px;
}
```

The property that transforms our list from vertical to horizontal is applied to the `<li>` tag. After setting the `display` property to `inline`, the list looks like Figure 4.14.

File: **listnav_horiz.css (excerpt)**

```
#navigation li {
  display: inline;
}
```

Figure 4.14. The list menu displays horizontally.

All that's left for us to do is to style the links for our navigation.

File: **listnav_horiz.css (excerpt)**

```
#navigation a:link, #navigation a:visited {
  padding: 3px 10px 2px 10px;
  color: #FFFFFF;
  background-color: #B51032;
  text-decoration: none;
  border: 1px solid #711515;
}
#navigation a:hover {
  color: #FFFFFF;
  background-color: #711515;
}
```

If you're creating boxes around each link, as I have here, remember that, in order to make more space between the text and the edge of its container, you'll need to add more left and right padding. To create more space between the navigation items, add left and right margins.

How do I create button-like navigation using CSS?

Navigation that looks to be composed of clickable buttons is a feature of many Websites. This kind of navigation is often created using images, with effects applied to make the edges look beveled and button-like. Often, an image swap JavaScript will be used so the button appears to depress when the cursor is held over it or the image is clicked.

Is it possible to create such button-like navigation systems using straight CSS? Absolutely!

Solution

Creating a button effect like that shown in Figure 4.15 is possible, and fairly straightforward, using CSS. The effect hinges on your use of the CSS border properties.

Figure 4.15. Button-style navigation can be built with CSS.

File: **listnav_button.html**

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
    "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
<title>Lists as navigation</title>
<meta http-equiv="content-type"
    content="text/html; charset=iso-8859-1" />
<link rel="stylesheet" type="text/css" href="listnav_button.css"
    />
</head>
<body>
<div id="navigation">
  <ul>
    <li><a href="#">Recipes</a></li>
    <li><a href="#">Contact Us</a></li>
    <li><a href="#">Articles</a></li>
    <li><a href="#">Buy Online</a></li>
  </ul>
</div>
</body>
</html>
```

File: **listnav_button.css**

```
#navigation {
  font-family: Arial, Helvetica, sans-serif;
  font-size: .9em;
}
#navigation ul {
  list-style: none;
  margin: 0;
  padding: 0;
  padding-top: 4px;
```

```
}
#navigation li {
  display: inline;
}
#navigation a:link, #navigation a:visited {
  margin-right: 2px;
  padding: 3px 10px 2px 10px;
  color: #A62020;
  background-color: #FCE6EA;
  text-decoration: none;
  border-top: 1px solid #FFFFFF;
  border-left: 1px solid #FFFFFF;
  border-bottom: 1px solid #717171;
  border-right: 1px solid #717171;
}
#navigation a:hover {
  border-top: 1px solid #717171;
  border-left: 1px solid #717171;
  border-bottom: 1px solid #FFFFFF;
  border-right: 1px solid #FFFFFF;
}
```

Discussion

To create this effect, we'll use the horizontal list navigation described in the previous example. However, to create the button look, we'll use different colored borders at the top and left than we do for the bottom and right sides of each button. By giving the top and left edges of the button a lighter colored border than we assign the button's bottom and right edges, we create a slightly beveled effect.

File: **listnav_button.css (excerpt)**

```
#navigation a:link, #navigation a:visited {
  margin-right: 2px;
  padding: 3px 10px 2px 10px;
  color: #A62020;
  background-color: #FCE6EA;
  text-decoration: none;
  border-top: 1px solid #FFFFFF;
  border-left: 1px solid #FFFFFF;
  border-bottom: 1px solid #717171;
  border-right: 1px solid #717171;
}
```

We reverse the border colors for the hover state, which gives the effect of the button being pressed.

File: **listnav_button.css (excerpt)**

```
#navigation a:hover {
  border-top: 1px solid #717171;
  border-left: 1px solid #717171;
  border-bottom: 1px solid #FFFFFF;
  border-right: 1px solid #FFFFFF;
}
```

Try using heavier borders, and changing the background images on the links, to create effects that fit in with your design.

How do I create tabbed navigation with CSS?

Navigation that appears as tabs across the top of the page is becoming increasingly popular. Commonly, this effect is built with images, but it's also possible to achieve using only CSS!

Solution

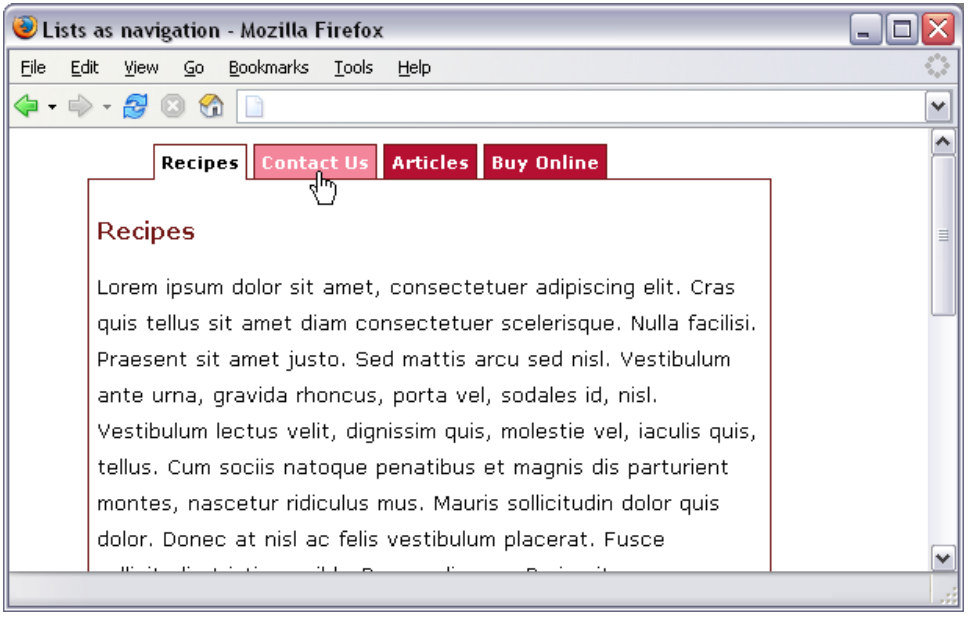
The tabbed navigation shown in Figure 4.16 can be created with an advanced version of the horizontal list.

File: **tabs.html**

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
    "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
<title>Lists as navigation</title>
<meta http-equiv="content-type"
    content="text/html; charset=iso-8859-1" />
<link rel="stylesheet" type="text/css" href="tabs.css" />
</head>
<body id="recipes">
<ul id="tabnav">
 <li class="recipes"><a href="#">Recipes</a></li>
 <li class="contact"><a href="#">Contact Us</a></li>
 <li class="articles"><a href="#">Articles</a></li>
 <li class="buy"><a href="#">Buy Online</a></li>
```

```
</ul>
<div id="content">
<h1>Recipes</h1>
<p>Lorem ipsum dolor sit amet, …</p>
</div>
</body>
</html>
```

Figure 4.16. CSS can be used to create tabbed navigation.



File: **tabs.css**

```
body {
  font: .8em/1.8em verdana, arial, sans-serif;
  background-color: #FFFFFF;
  margin-left: 50px;
  margin-right: 100px;
}
#content {
  border-left: 1px solid #711515;
  border-right: 1px solid #711515;
  border-bottom: 1px solid #711515;
  padding: 10px 5px 6px 5px;
}
```

```
#content h1 {
  font-size: 1.2em;
  color: #711515;
  background-color: transparent;
}
ul#tabnav {
  list-style-type: none;
  margin: 0;
  padding-left: 40px;
  padding-bottom: 24px;
  border-bottom: 1px solid #711515;
  font: bold 11px verdana, arial, sans-serif;
}
ul#tabnav li {
  float: left;
  height: 21px;
  background-color: #B51032;
  color: #FFFFFF;
  margin: 2px 2px 0 2px;
  border: 1px solid #711515;
}
ul#tabnav a:link, ul#tabnav a:visited {
  display: block;
  color: #FFFFFF;
  background-color: transparent;
  text-decoration: none;
  padding: 4px;
}
ul#tabnav a:hover {
  background-color: #F4869C;
  color: #FFFFFF;
}
body#recipes li.recipes, body#contact li.contact,
body#articles li.articles, body#buy li.buy {
  border-bottom: 1px solid #fff;
  color: #000000;
  background-color: #FFFFFF;
}
body#recipes li.recipes a:link, body#recipes li.recipes a:visited,
body#contact li.contact a:link, body#contact li.contact a:visited,
body#articles li.articles a:link,
body#articles li.articles a:visited, body#buy li.buy a:link,
body#buy li.buy a:visited {
  color: #000000;
  background-color: #FFFFFF;
}
```

Discussion

The tabbed navigation I've used here is adapted from a CSS Tabs example by Joshua Kaufman[1]. The basis for this example is the kind of simple ordered list we've seen throughout this chapter, except that each list item is assigned a `class` attribute that describes the link it contains:

File: **tabs.html (excerpt)**

```
<ul id="tabnav">
  <li class="recipes"><a href="#">Recipes</a></li>
  <li class="contact"><a href="#">Contact Us</a></li>
  <li class="articles"><a href="#">Articles</a></li>
  <li class="buy"><a href="#">Buy Online</a></li>
</ul>
```

To build the tabbed effect, first create the CSS for the `<ul>` with an ID of `tabnav`.

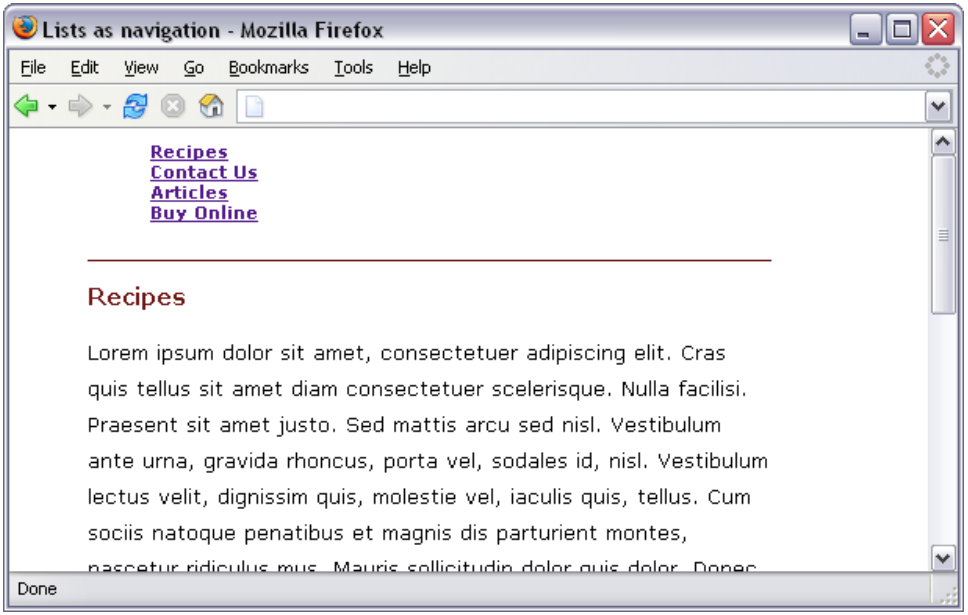
File: **tabs.css (excerpt)**

```
ul#tabnav {
  list-style-type: none;
  margin: 0;
  padding-left: 40px;
  padding-bottom: 24px;
  border-bottom: 1px solid #711515;
  font: bold 11px verdana, arial, sans-serif;
}
```

This CSS removes the bullets, controls the margin and padding, adds a bottom border to the list, and sets the desired font. Figure 4.17 shows the result so far.

[1] http://unraveled.com/projects/css_tabs/

Figure 4.17. The navigation displays after the <ul> is styled.



We now need to style the list items:

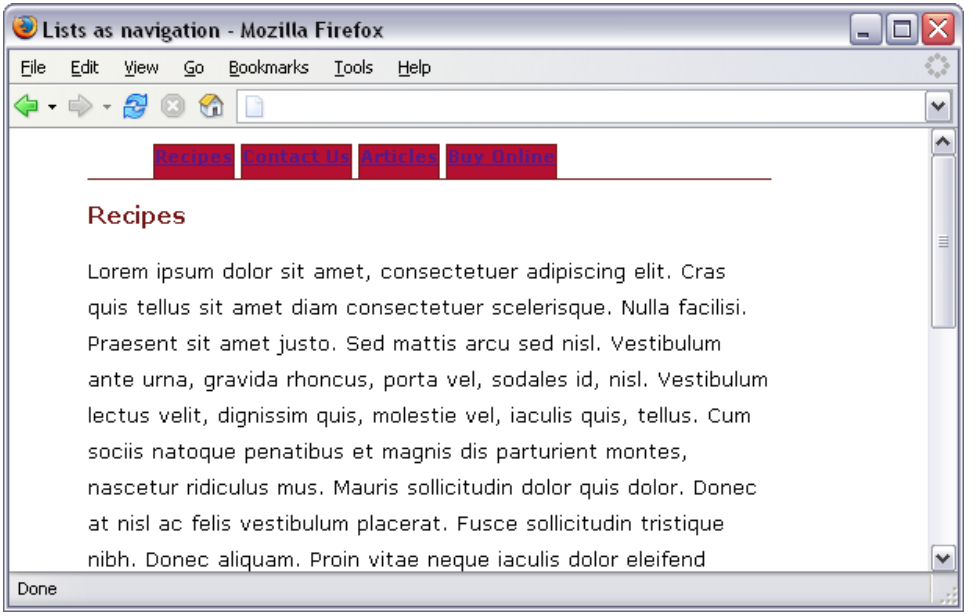
File: **tabs.css (excerpt)**

```
ul#tabnav li {
  float: left;
  height: 21px;
  background-color: #B51032;
  color: #FFFFFF;
  margin: 2px 2px 0 2px;
  border: 1px solid #711515;
}
```

This rule uses the `float` property to move each list item up onto the same line while maintaining its block-level status.

We also set `color` and `background-color`, and add a border, creating a visual style for the tabs. See what we have so far in Figure 4.18.

Figure 4.18. The navigation tabs reflect the new styles.

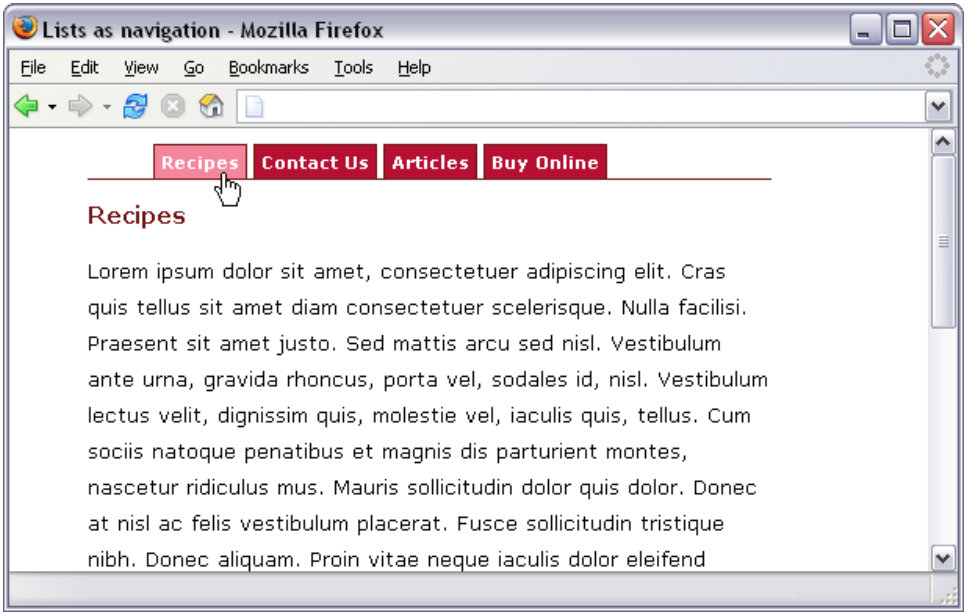


Next, we style the links, finishing the look of the tabs in their unselected state. The results are shown in Figure 4.19.

File: **tabs.css (excerpt)**

```
ul#tabnav a:link, ul#tabnav a:visited {
  display: block;
  color: #FFFFFF;
  background-color: transparent;
  text-decoration: none;
  padding: 4px;
}
ul#tabnav a:hover {
  background-color: #F4869C;
  color: #FFFFFF;
}
```


Figure 4.19. Style the navigation links.



Now, to complete the tab navigation we must highlight the tab which corresponds to the currently displayed page. You'll recall that each list item has been assigned a unique class name. If we assign an ID to the `<body>` tag that matches the tab class that we wish to highlight for that page, the CSS code can do the rest of the work:

File: **tabs.html (excerpt)**

```
<body id="recipes">
```

Though bulky in the selector department (since it targets both unvisited and visited links), the CSS code that styles the tab matching the `<body>` ID is relatively straightforward:

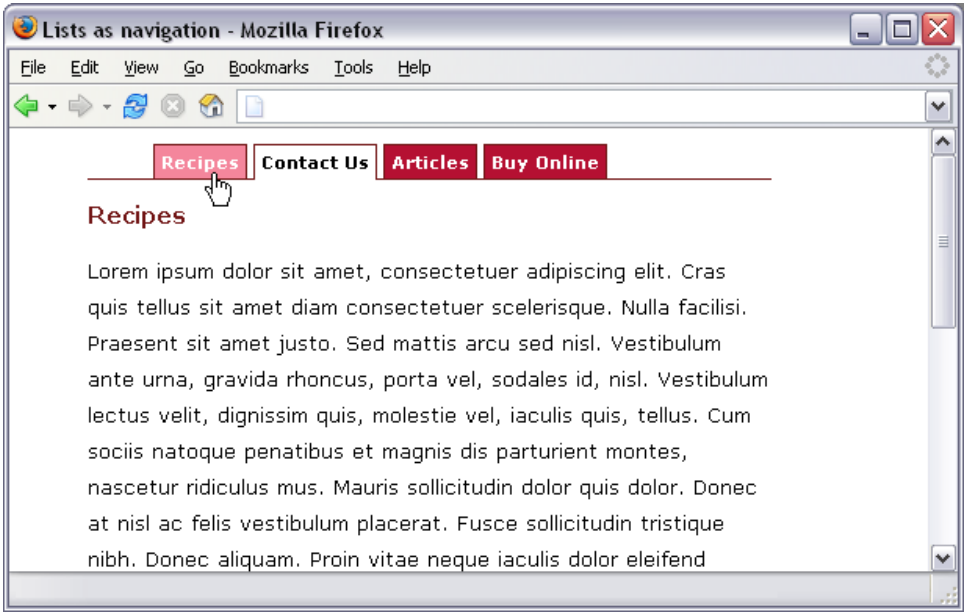
File: **tabs.css (excerpt)**

```
body#recipes li.recipes, body#contact li.contact,
body#articles li.articles, body#buy li.buy {
  border-bottom: 1px solid #fff;
  color: #000000;
  background-color: #FFFFFF;
}
body#recipes li.recipes a:link, body#recipes li.recipes a:visited,
```

```
body#contact li.contact a:link, body#contact li.contact a:visited,
body#articles li.articles a:link,
body#articles li.articles a:visited, body#buy li.buy a:link,
body#buy li.buy a:visited {
  color: #000000;
  background-color: #FFFFFF;
}
```

With this code in place, a `<body>` ID of `recipes` will highlight the Recipes tab, `contact` will highlight the Contact Us tab, and so on. The results are shown in Figure 4.20.

Figure 4.20. Highlight Contact Us by adding `contact` as the ID of the `<body>` tag.



Identifying a Useful Technique

This technique of adding an ID to the `<body>` tag can be very useful. For example, you may have different color schemes for different sections of your site, to help the user identify which section they're in. You can simply add the section name to the `<body>` tag and pick it up exactly as we did in this example, within the style sheet.

To finish off, we simply add a border to the content area to neatly enclose the navigation along its top.

File: **tabs.css (excerpt)**

```
#content {
  border: 1px solid #711515;
  border-top: none;
  padding: 10px 5px 6px 5px;
}
```

How do I change the cursor type?

It is usual for the browser to change the cursor to a hand icon when users move the mouse over a link on any part of the page. Occasionally—perhaps to fit in with a particular interface—you might want to change the cursor to represent something else.

Solution

Changing the cursor is achieved with the CSS property, `cursor`. Figure 4.21 shows this property in action.

Figure 4.21. `cursor: wait` causes an hourglass to display as the cursor.

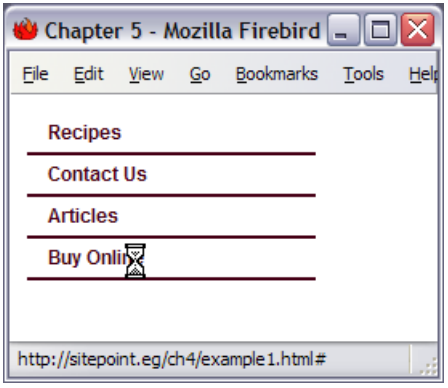


Table 4.1. CSS2 Standard Cursors

`cursor` value	Appearance (as in IE6)	IE (Win)	IE (Mac)	NS/Moz
`auto`	n/a	4	4	6/1
`crosshair`		4	4	6/1
`default`		4	4	6/1
`e-resize`		4	4	6/1
`help`		4	4	6/1
`move`		4	4	6/1
`n-resize`		4	4	6/1
`ne-resize`		4	4	6/1
`nw-resize`		4	4	6/1
`pointer`		4	4	6/1
`s-resize`		4	4	6/1
`se-resize`		4	4	6/1
`sw-resize`		4	4	6/1
`text`		4	4	6/1
`url(`*`url`*`)`	n/a	6	–	–
`w-resize`		4	4	6/1
`wait`		4	4	6/1

Discussion

The `cursor` property can take a range of values, which create cursors commonly seen in graphical user interfaces. Changing the cursor can be a useful way to provide user feedback in a Web application that tries to create a friendly interface. For example, you might decide to use a question mark cursor to indicate help text. However, you should use this effect with care, and keep in mind the fact

that people are generally used to standard browser behavior where, for instance, the cursor represents a hand icon when it's hovered over a link.

Table 4.1 lists the various properties that are available in the CSS standard, and the common browser versions that support them. Table 4.2 lists additional values that are only supported by Internet Explorer browsers.

Table 4.2. Internet Explorer-only Cursors

`cursor` value	Appearance (as in IE6)	IE (Win)	IE (Mac)
`all-scroll`		6	–
`col-resize`		6	–
`hand`		4	4
`no-drop`		6	–
`not-allowed`		6	–
`progress`		6	–
`row-resize`		6	–
`vertical-text`		6	–

How do I create rollovers in CSS without JavaScript?

CSS-based navigation can provide some really interesting effects, but there are still some things that require images. Is it possible to enjoy the advantages of text-based navigation, and still use images?

Solution

It is possible to combine images and CSS to create JavaScript-free rollovers. This solution is based on a technique described at wellstyled.com.[2]

[2] http://wellstyled.com/css-nopreload-rollovers.html

File: **images.html**

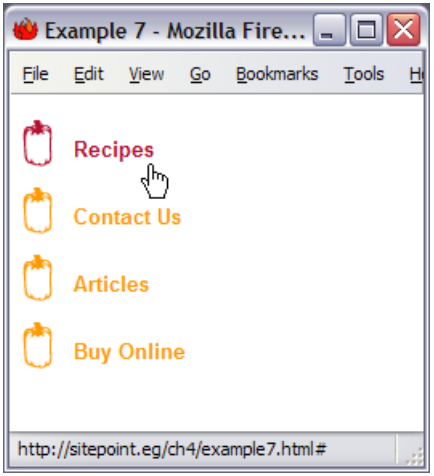
```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
    "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
<title>Lists as navigation</title>
<meta http-equiv="content-type"
    content="text/html; charset=iso-8859-1" />
<link rel="stylesheet" type="text/css" href="images.css" />
</head>
<body>
<ul id="nav">
  <li><a href="#">Recipes</a></li>
  <li><a href="#">Contact Us</a></li>
  <li><a href="#">Articles</a></li>
  <li><a href="#">Buy Online</a></li>
</ul>
</body>
</html>
```

File: **images.css**

```
ul#nav {
  list-style-type: none;
  padding: 0;
  margin: 0;
}
#nav a:link, #nav a:visited {
  display: block;
  width: 150px;
  padding: 10px 0 16px 32px;
  font: bold 80% Arial, Helvetica, sans-serif;
  color: #FF9900;
  background: url("peppers.gif") top left no-repeat;
  text-decoration: none;
}
#nav a:hover {
  background-position: 0 -69px;
  color: #B51032;
}
#nav a:active {
  background-position: 0 -138px;
  color: #006E01;
}
```

The results can be seen in Figure 4.22, but since the effect is color-based I suggest you try it for yourself. Don't forget to click on a link or two!

Figure 4.22. The completed menu uses images to advantage.



Discussion

This solution offers a means of using images in your navigation without having to resort to preloading lots of separate files.

The navigation has three states. These three states are not depicted using three separate images; rather, they use one large image that contains all three images, as seen in Figure 4.23.

Figure 4.23. The pepper image contains all three states.

The navigation is marked up as a simple list.

File: **images.html (excerpt)**

```
<ul id="nav">
  <li><a href="#">Recipes</a></li>
  <li><a href="#">Contact Us</a></li>
  <li><a href="#">Articles</a></li>
  <li><a href="#">Buy Online</a></li>
</ul>
```

In the CSS, the rule for the links displays the background image. Because the image is far bigger than the area required for this element, however, we only see the first, yellow, pepper at first:

File: **images.css (excerpt)**

```
#nav a:link, #nav a:visited {
  display: block;
  width: 150px;
  padding: 10px 0 16px 32px;
  font: bold 80% Arial, Helvetica, sans-serif;
  color: #FF9900;
  background: url("peppers.gif") top left no-repeat;
  text-decoration: none;
}
```

The `:hover` state simply shifts the background image up a number of pixels sufficient to reveal the red pepper. I had to move it 69 pixels, but the distance you'll need to move your image will depend on the graphic itself. You could probably work it out mathematically, or you could do as I do and simply increment the image a few pixels at a time, until it appears in the right location on hover.

File: **images.css (excerpt)**

```
#nav a:hover {
  background-position: 0 -69px;
  color: #B51032;
}
```

The active state again shifts the background image, this time to display the green pepper when the link is clicked.

File: **images.css (excerpt)**

```
#nav a:active {
  background-position: 0 -138px;
  color: #006E01;
}
```

That's all there is to it! The effect can fall apart if the text is resized to a larger font, allowing the edges of the hidden images to display. You can get around that issue, to some degree, by leaving quite large spaces between the images, but this is a point to be aware of.

The site on which I found the technique we've discussed here also offers a technique that counters a flickering effect you can unwittingly achieve in Internet Explorer. In my tests, that effect tends only to be a problem when the image is larger than the ones we've used here; however, if your navigation items flicker, check out the link given above.

Summary

This chapter has discussed a range of different ways in which we can create navigation using structurally sound markup. These examples can be used as starting points for your own experiments and ideas. On existing sites, where a full redesign is not possible, switching to a CSS-based navigation system can be a good way to improve the accessibility and load speed of the site without affecting its look and feel in a big way.

What's Next?

If you've enjoyed these chapters from *The CSS Anthology: 101 Essential Tips, Tricks & Hacks*, why not order yourself a copy?

In the rest of the book, you'll learn how to put the full power of CSS to use in a number of practical situations. In particular, you'll learn some of the well kept secrets of table and form styling with CSS, and using CSS to accommodate the many browsers and devices that may view your site.

Of course there is a massive chapter on CSS page layout techniques, including the very latest discoveries in this area. Finally, there's an entire chapter on experimental, or browser-specific effects that can add flair to your projects when viewed in particular browsers.

Here are just a few of the solutions in the rest of the book:

- Highlight table rows when the mouse hovers over them
- Display a monthly calendar data in a familiar format
- Work around tricky CSS bugs in Internet Explorer
- Create alternative style sheets without duplicating code
- Create two- and three-column layouts that work in most browsers without resorting to JavaScript.
- Create drop-down menus, sticky page footers, rounded corners, and more, all with pure CSS.

And as with all SitePoint books, you'll also gain access to the code archive download, so you can try out all the examples without retyping.

Index

C

D

E

F

G

H

I

J

M

N

O

P

Q

R

S

T

U

V

W

X