CHAPTER

37

Media Styles and Defining Documents for Printing

IN THIS CHAPTER

Understanding CSS Media Types

Setting Up Documents for Printing

Creating a Multimedia Document

The Web was originally designed to bring printed media to the computer screen. Paragraph elements, list elements, and tables were all designed to provide adequate vehicles for approximating documents normally found in print.

As times have changed, this situation has somewhat reversed. Now documents that originated on the Web are being formatted for the printed page. This phenomenon is especially true for such documents as e-commerce invoices, calendars and events, and documents of directions — whether they be do-it-yourself instructions or directions to a popular venue.

Of course, printed documents are not the only secondary destination for Web-originated documents; Web documents are also being made available to aural (speech only) devices, presentation and handheld devices, and low-resolution Web browsers (such as WebTV).

Thankfully, CSS has several mechanisms for formatting a document for these various media types. This chapter concentrates on the print media type, but many of the techniques discussed translate to other media types as well.

Understanding CSS Media Types

Table 37-1 lists the various media types supported in CSS.

Note

Because of the rapidly evolving deployment of Web documents to other media types, the preceding list (specified in CSS2) is not designed to be all-inclusive. The list is amended from time to time and will continue to grow as Web documents continue to be deployed in other formats. ■

TABLE 37-1

CSS Media Types

Media Type	Intended Destination/User Agent
`all`	Suitable for all devices
`aural`	Intended for speech-capable user agents and synthesizers
`Braille`	Intended for Braille tactile feedback devices
`embossed`	Intended for paged Braille printers
`handheld`	Intended for handheld devices (typically small screen, monochrome, limited bandwidth)
`print`	Intended for paged, opaque material and for documents viewed onscreen in print preview mode
`projection`	Intended for projected presentations, such as projectors or print to transparencies
`screen`	Intended primarily for color computer screens
`tty`	Intended for media using a fixed-pitch character grid, such as teletypes, terminals, or portable devices with limited display capabilities
`tv`	Intended for television-type devices (low resolution, color, limited-scrollability screens, with sound available)

Specifying media types

You have several ways to specify that a style, or group of styles, should be used only for a specific media type. The following sections detail each method.

Note

If no media type is specified in a document, the default of "all" is assumed and all user agents try to render the document according to its CSS code. If you intend your document to be displayed only on computer-type devices, ensure that you set all of your document's styles to the `screen` media type using the methods outlined in the next few sections. ■

Specifying one style's media type

The `@media` rule can be used to specify that a single style definition applies only to specific media type(s). This rule has the following syntax:

```
@media <media_type(s)> {  definition }
```

For example, to specify that a particular definition is to be used with print media only, you could use code similar to the following, which has been indented for legibility:

```
@media print {
               body  { background-color: white;
                       color: black;
                     }
              }
```

You might have noticed the mention of media types (plural) in the preceding definition of the @media rule. This is because you can specify several media types, each separated by a comma, in the @media definition. For example, you might choose to specify the same style for both print and handheld media, using code similar to the following:

```
@media print, handheld {
               body  { background-color: white;
                       color: black;
        }
 }
```

Specifying a group of styles' media type

The @media rule is not limited to specifying one style's media type; you can place as many styles as you like within the @media rule's brackets. For example, consider the following code, which specifies that two styles are meant for print media:

```
@media print {
               body  { background-color: white;
                       color: black;
        }
               .highlight { background-color: black;
                            color: white;
             }
 }
```

The style element (style) supports a media attribute that can be used to specify that all the styles encapsulated within the element are of a particular media type, or types. The style element has the following format:

```
<style type="text/css" media="<media_type(s)>">
... style definitions ...
</style>
```

Note that the media attribute can be used to specify a single media type or a comma-separated list of multiple types.

You can use either of these methods to specify the media for entire blocks of styles, and create multiple media blocks within the same style block. However, it is generally easier to create groups of media-specific styles in external files and use the link element to link each into your document. This technique is outlined in the next section.

Specifying an external style sheet's media type

If you choose to place your media-specific styles in an external file, you can use the link element (link) to attach one or more of the files into your document. The link element has the following syntax:

```
<link rel="stylesheet" href="<css_file>" type="text/css"
   media="<media_type>" />
```

For example, if you have a style sheet named printed.css that you intend to use as print media instructions for a document, you could use a link element similar to the following:

```
<link rel="stylesheet" href="printed.css" type="text/css"
   media="print" />
```

Note

The link element's href argument requires a full path to the specified CSS file. For example, if your CSS files are stored in a directory named "styles," you should specify something similar to styles/printed.css in the preceding example. ■

You can also use the CSS @import rule to specify an external style sheet. The @import rule has the following syntax:

```
@import url("<stylesheet>") <media_type>;
```

For example, the following code imports an external style sheet named printstyles.css and designates its media type as print:

```
@import url("printstyles.css") print;
```

Keep in mind that the @import rule is a CSS command and must appear within an appropriate style section within your document.

Setting Up Documents for Printing

Each media type has additional properties to help define aspects of its medium and how documents using that type should be handled and rendered; and covering all the media types is beyond the scope of this book. In addition to the screen media type, however, print media is the most frequently used media type for Web documents.

This section covers the properties and methods available for print media.

The page box formatting model

If you've ever worked with a desktop publishing platform using software such as Quark XPress, InDesign, or PageMaker, you're probably familiar with the concept of a page box, within which fits everything that must appear on a page. Even if you haven't worked with desktop publishing software, you've probably seen precursors to the HTML/CSS box formatting model in word-processing packages you've used.

When you work in a word-processing or desktop publishing environment, you work with finite page sizes and page margins. The CSS page box formatting model is an attempt to replicate this for browser-based media. The page box model is based on the CSS box model, as shown in Figure 37-1.

Figure 37-1 simply extends the familiar CSS box model to reveal two major areas:

- The **page area** contains all of a page's elements.
- The **margin area** surrounds the page area. When a page area size is specified, the margins, if any, are subtracted.

On top of the page box, the model is expanded still further to account for the difference between *continuous* media, as represented by a browser, and *paged* media, which consists of discrete and specific page entities. This expansion is represented by the visual formatting model, which allows transfer of the continuous media as seen in a Web browser to an actual sheet of paper or transparency.

Defining the page size with the @page rule

In CSS, you define your desired page size using the `@page` rule. The `@page` rule defines which pages should be bound to the definitions within the rule. You then use a `page` property within a style element or attribute to indicate to which page a specific element belongs.

Note

Unfortunately, browser support has still not caught up to this particular CSS rule, and support is largely nonexistent at this point. Microsoft actually does provide support for this rule, but only through the MSHTML component, which application developers use as a browser widget within their applications. Internet Explorer itself does not include support for this rule in its printing templates, which are used for print previewing and printing Web documents from the browser. ■

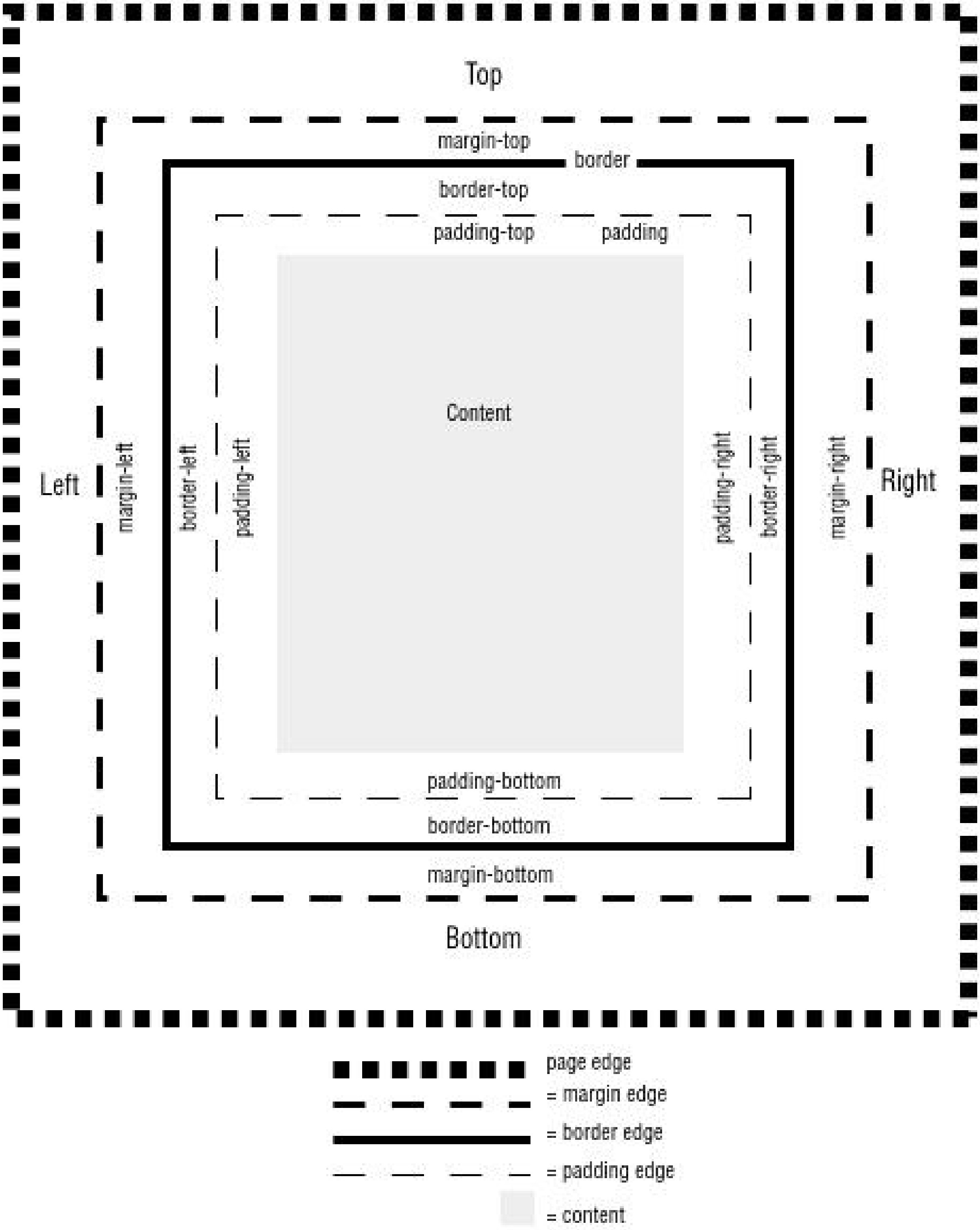
The following listing shows an example of `@page` use:

```
@page printed {
         size: 3in 3in;
 }
body, p {
         page: printed;
}
```

In this listing, a page named `printed` is defined. Then the body and paragraph elements are defined using `printed` as the value of the `page` property and should print according to the specifications outlined in the `@page` rule.

FIGURE 37-1

The CSS box model



Note

The page name `printed` in the previous example is optional. If your document will use only one type of page, you can omit the page name, as in the following:

```
@page {
      size: 3in 3in;
}
```

Thereafter, any element defined as being print media will use the one, unnamed page definition. ■

In CSS2, you can use page selectors to name the first page, all left pages, all right pages, or a page with a specific name to which the rules apply. In the case of the previous listing, a named page called `printed` was used.

Setting up the page size with the size property

The actual dimensions of the page are defined using the `size` property, which consists of two values: one for the width and the other for the height. For example, the following definition sets up an 8½ by 11 inches page:

```
@page {
         size: 8.5in 11in;
      }
```

You can also use the following relative size values with the `size` property:

- `auto` — The default value set to whatever the target paper size is in the printer's settings.
- `landscape` — Rotates the page and swaps the two measures. In the previous example, the printed sheet would print at 11 inches wide by 8½ inches long.
- `portrait` — Overrides the targeted media's default settings to correspond with the dimensions you set in the `size` property.

Setting margins with the margin property

You should be careful when using margins because they are the outermost layer of a page. If you set the margins of a `body` element to three inches on either side, for example, be sure to adjust the width of the `body` element as well, or your page will not format properly.

However, in theory, you should be able to set margins for the printed page without worrying about the body text running off to one side of the browser when you neglect to set the width of the page's other elements. This is because margins can be set within the `@page` rule using the margin properties in CSS. You can set the page margins using the `margin` property in the same manner as you use it anywhere else, as shown here:

```
@page {
       size: 8.5in 11in;
       margin: .5in;
{
```

Cross-Ref

The CSS margin properties are covered in Chapter 32. ■

When they are set within the @page rule (currently supported by Opera only), the margin settings should be ignored when being viewed on the Web. However, as mentioned earlier, at the time of this writing, browser support for this feature is weak and results can be unpredictable.

Including crop and cross marks

If your document is destined for publication, it might be useful to include *crop marks* or *cross marks*. Crop marks look like plus signs (+) and are printed on the edge of page margins, showing a printer where to cut the page. Cross marks are other symbols printed near the edge of pages that enable printers to align numerous pages.

The marks property controls the printing of crop and cross marks and has the following syntax:

```
marks: <crop> | <cross> | <none> | <inherit> ;
```

Note

User agent support for the marks property is virtually nonexistent. ■

Controlling page breaks

In the event that your users may want to print your Web pages, you may want to avoid inappropriate page breaks. The three page-break properties can help better control page break behavior:

- page-break-before — Specifies how a page should break before a specific element, and on what side of the page the flow should resume
- page-break-after — Specifies how a page should break after a specific element, and on what side of the page the flow should resume
- page-break-inside — Tells the browser how to break a page from within a box element. Actual support for this property is limited to Opera. Neither Internet Explorer nor Netscape-based browsers support this property.

Using the page-break-before and page-break-after properties

The page-break-before and page-break-after properties specify how a page should break before or after a specific element, depending on which of the two properties you use, and on what side of the page the flow should resume. The CSS2 documentation provides some guidelines for page breaks:

- Page breaking should be avoided inside table elements, floating elements, and block elements with borders.
- Page breaking should occur as few times as possible. In other words, it's not a good idea to break a page with every paragraph.
- Pages that don't have explicit breaks should be approximately the same height.

Opera offers the best support for these properties. The Gecko browsers provide partial support, and support is all but missing from Internet Explorer. Table 37-2 lists the values that can be used with either the `page-break-before` or `page-break-after` property.

TABLE 37-2

Page-Break-Before/After Property Values

Value	Description
`inherit`	Specifies that the value should be inherited from the parent
`auto`	Allows the user agent (browser) to insert page breaks on an as-needed basis
`avoid`	Tells the user agent to avoid inserting page breaks before or after the current element
`left`	Forces one or two page breaks to create a blank left page
`right`	Forces one or two page breaks to create a blank right page
`always`	Tells the browser or user agent to always force a page break before or after the current element
`'""'`	This is not a value found in the spec but a value that can be used in Internet Explorer; it specifies that no property value should be used and, therefore, that no page break should be inserted before the current element.

Figure 37-2 shows the effects of a badly formatted page break. Notice how the heading remains on one page while its text flows to the next.

You can set the `page-break-before` or `page-break-after` property in a `p` element, for example, to force a page break before or after all `p` elements. You probably wouldn't want to actually do that, but you can create a class selector rule and apply the rule to the first or last paragraph of a page, depending on your needs, like this:

```
.pagebreak {
                page-break-before: always;
}
```

You can then use this class with elements, such as with specific `h2` elements as shown in the following code, which fixes the odd page break shown earlier in Figure 37-2:

```
<h2 class="pagebreak">Spa Services</h2>
```

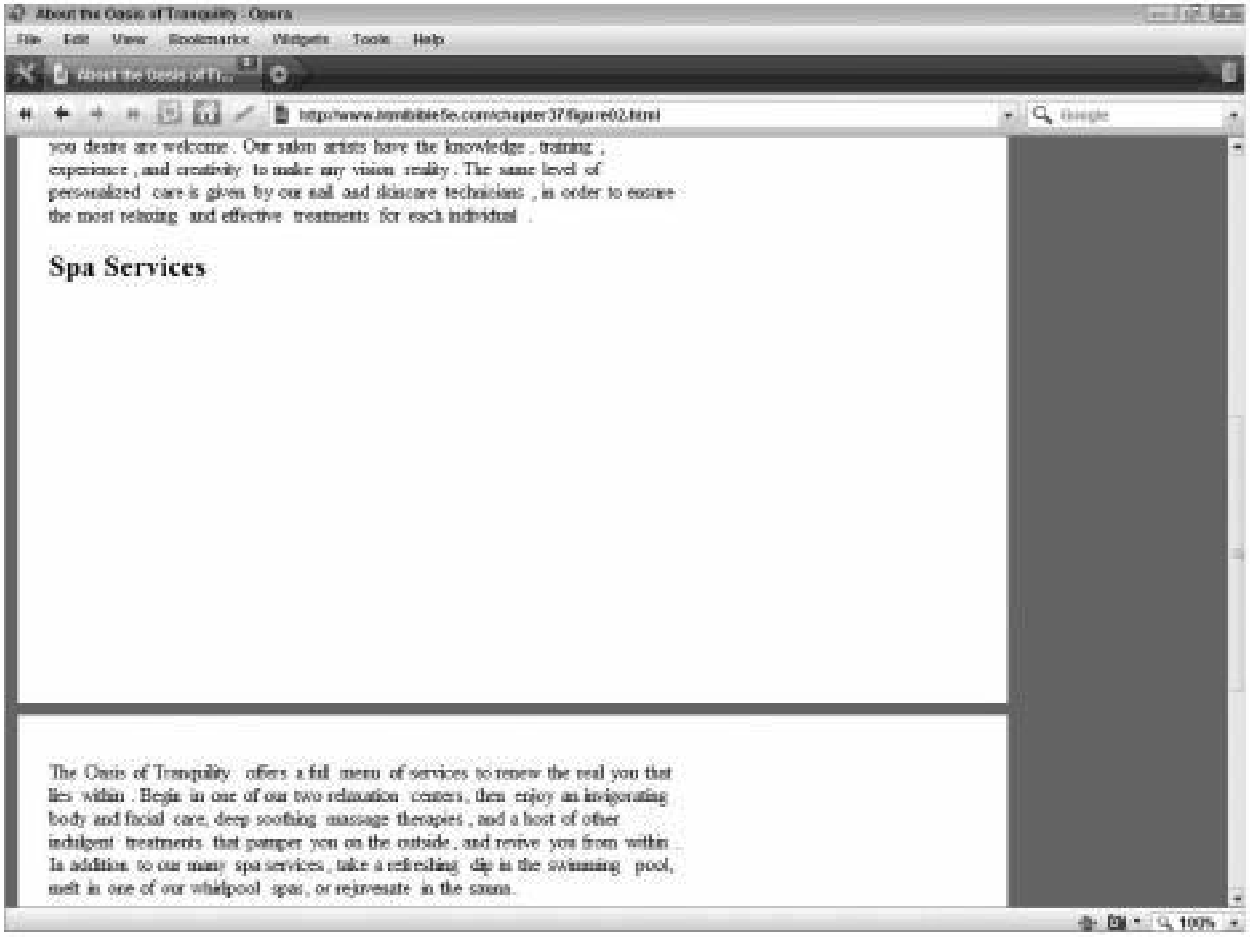
The fixed break is shown in Figure 37-3.

Note

Page breaks cannot be used within positioned elements. For example, if you have an absolutely positioned `div` element with a child `p` element, you cannot assign a page break to the `p` element. ■

FIGURE 37-2

A bad page break separates text from its heading.


About the Oasis of Tranquility - Opera
File Edit View Bookmarks Widgets Tools Help
About the Oasis of Tr...
http://www.htmlbible5e.com/chapter37/figure02.html
Google
100%

you desire are welcome . Our salon artists have the knowledge , training , experience , and creativity to make any vision reality . The same level of personalized care is given by our nail and skincare technicians , in order to ensure the most relaxing and effective treatments for each individual .

Spa Services

The Oasis of Tranquility offers a full menu of services to renew the real you that lies within . Begin in one of our two relaxation centers, then enjoy an invigorating body and facial care, deep soothing massage therapies , and a host of other indulgent treatments that pamper you on the outside , and revive you from within . In addition to our many spa services , take a refreshing dip in the swimming pool, melt in one of our whirlpool spas , or rejuvenate in the sauna.

Using the page-break-inside property

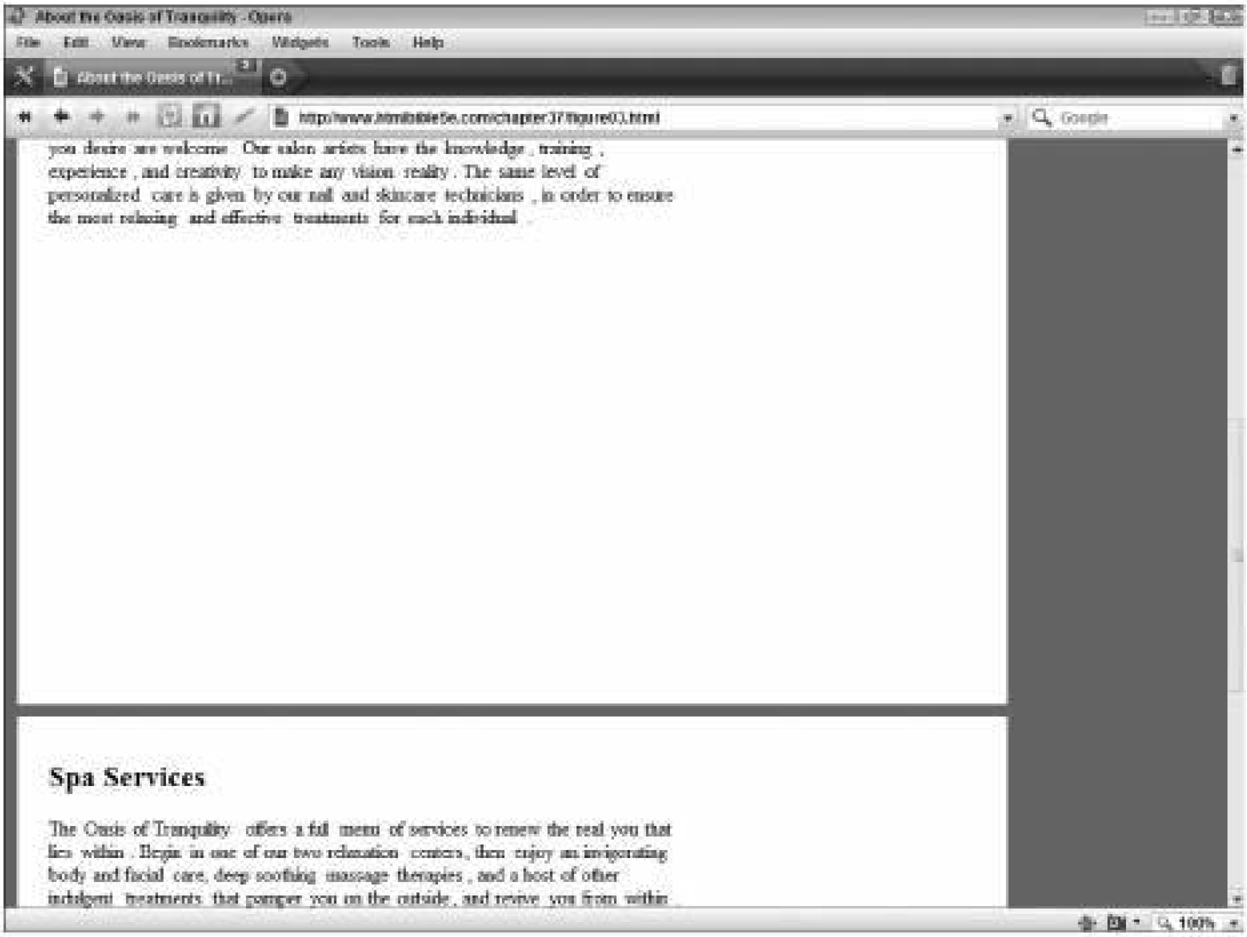
You can also use a `page-break-inside` property to handle page breaks within elements (for example, if you have a very long `div` element). However, in practice, most browsers have little to no support for this property.

Handling widows and orphans

Widows and orphans are undesirable in typography, but CSS has provided an opportunity to reduce their impact. Widows and orphans refer to broken text, such as when a page break severs a paragraph, leaving one line from the paragraph on either side of the break. A *widow* is a spurious line at the top of a page, left over from the paragraph on the previous page. An *orphan* is similar, except it consists of a spurious line at the bottom of a page, where the remainder of the paragraph occurs on the next page. Again, it can be unsightly if a section or paragraph starts at the very end of a page and the page break results in a single line being left at the end of the previous page.

FIGURE 37-3

Adding a page break directive can indicate where a page should break, avoiding problems.


About the Oasis of Tranquility - Opera
File Edit View Bookmarks Widgets Tools Help
About the Oasis of Tr...
http://www.htmlbible5e.com/chapter37/figure03.html
Google
100%

you desire are welcome . Our salon artists have the knowledge , training , experience , and creativity to make any vision reality . The same level of personalized care is given by our nail and skincare technicians , in order to ensure the most relaxing and effective treatments for each individual .

Spa Services

The Oasis of Tranquility offers a full menu of services to renew the real you that lies within . Begin in one of our two relaxation centers, then enjoy an invigorating body and facial care, deep soothing massage therapies , and a host of other indulgent treatments that pamper you on the outside , and revive you from within .

The two CSS properties relevant to widows and orphans, are conveniently named, respectively, `widow` and `orphan`.

Note

These two properties have virtually no support beyond the Opera browser. ■

Both of these properties have similar syntax:

```
widow: <integer or inherit>;
orphan: <integer or inherit>;
```

You name the property, and then supply the value, which can be either an integer or the explicit value `inherit`, the latter of which means that the element named in the style rule inherits the properties of its parent. The following sets a `p` element's widow to a minimum of three lines. This means that the bottom of the page must have a minimum of three lines when printing:

```
p { widow: 3; }
```

An integer value for either property specifies that the top (widow) or bottom (orphan) should have the given number of lines — if that is not possible, the entire element moves to the next page.

Preparing documents for double-sided printing

To set up pages for printing double-sided documents, you must account for different margins on each side of a page. One method to handle this would be to manually set margins for each element, anticipating on which side of the page each would appear. Thankfully, there is an easier way. You can use the pseudo-classes `:left` and `:right` to set the margins of each differently. For example, consider the following:

```
@page :left {
  margin-left: .5in;
  margin-right: .25in;
}
@page :right {
  margin-left: .25in;
  margin-right: .5in;
}
```

Note

Of course, like many advanced CSS features, user agent support for the `:left` and `:right` pseudo-classes is almost nonexistent. ■

You can also use the `:first` pseudo-class to specify different values for the first page of a document with code similar to the following:

```
@page {
        margin: 1in
}
@page :first {
        margin-top: 3in
}
```

The preceding code sets all the page margins to one inch, but the `:first` definition resets the top margin of the first page to three inches.

Creating a Multimedia Document

No doubt you have run across several documents on the Web that have corresponding "print friendly" versions. These documents tend to be informational in nature — recipes, maps and directions, do-it-yourself instructions, invoices, and so on. The screen media representations of the documents typically include all the usual elements — graphical headings, several advertisements, and colorful layouts — not entirely suitable for printing the key information embedded in the document.

As such, many of these documents include a print friendly link when displayed on the Web. Clicking that link displays a new page with the important information, but devoid of the superfluous elements and information. Sometimes the print friendly page includes additional information necessary, or information just simply nice to have.

Using CSS media descriptors and relying on the CSS cascade and inheritance properties, you can use one document for both the screen and print versions. An example of this concept follows.

The online (screen media) document

The following listing and Figure 37-4 show a typical online page with information (driving directions) that would be advantageous to print:

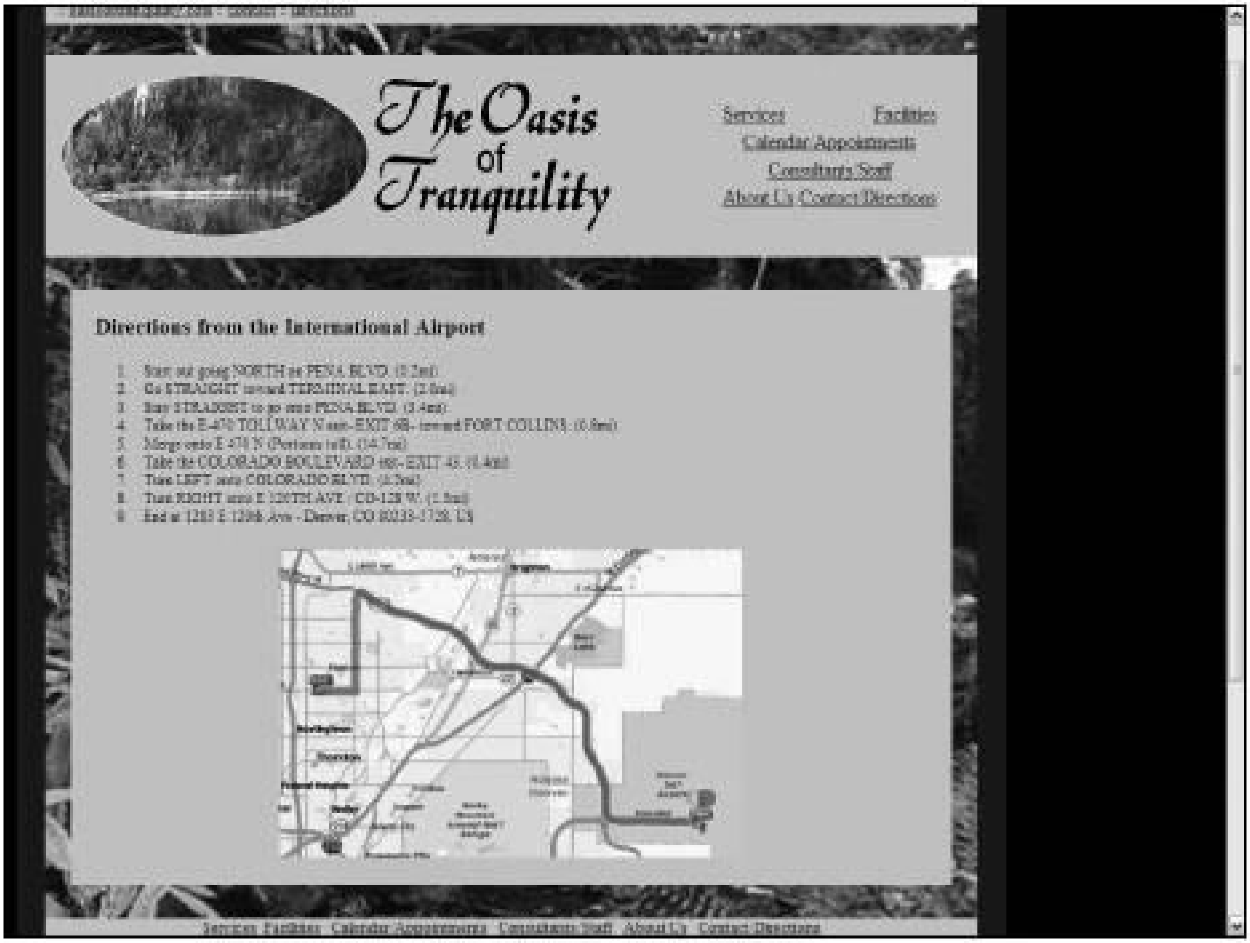
```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01//EN"
    "http://www.w3.org/TR/html4/strict.dtd">
    <html>
    <head>
    <!-- Breadcrumb and utility scripts -->
    <script type="text/JavaScript" src="/functions.js">
                     </script>
                     <style type="text/css">
   body { background-color: black; }
 .main { width: 768px;
 height: 1024px;
 padding: 20px;
      text-align: center;
      background-color: #222222;
      background-image:url(/images/junglepool.jpg);
      background-position: center center;
      background-repeat: no-repeat;
 border: 3pt #222222 solid; }
 .breadcrumbs { font-size: small;
 text-align: left;
 padding-left: 10px;
 background-color: #CCCCCC;
 height: 1.5em; }
 .heading { background-color: #CCCCCC;
 text-align: left;
 margin: 25px 0px; }
   /* Heading for printing - hide it for now */
 .printheading { text-align: center;
 margin-bottom: 50px;
 display: none;
 visibility: hidden; }
 .content { text-align: left;
 font-size: 9pt;
 background-color: #CCCCCC;
 margin: 15px;
     margin-bottom: 0px;
 padding: 20px;  }
```

```
.center { text-align: center; }
.mleft { text-align: left; }
.mright { text-align: right; }
.mcenter { text-align: center; }
.footer {  margin-top: 25px;
font-size: small;
text-align: center;
background-color: #CCCCCC;
height: 1.5em; }
</style>
   <!-- Print friendly (print media) style sheet -->
   <link rel="stylesheet" href="/printstyles.css"
    type="text/css" media="print" />
  <title>Oasis of Tranquility - Directions</title>
  </head>
  <body>
  <div class="main">
         <div class="breadcrumbs">
              <script type="text/JavaScript">
                     breadcrumbs();
</script>
   </div>
   <div class="printheading">
        <h1>The Oasis of Tranquility</h1>
        </div>
        <div class="heading">
             <table border="0" width="100%">
  <tr>
  <td><img src="/images/OasisHeader.gif" width="475px"
  height="161" alt="The Oasis of Tranquility Header" /></td>
           <td>
           <table border="0">
                   <tr>
                   <td class="mleft"><a
                    href="services">Services</a></td>
  <td class="mright">
        <a href="facilities">Facilities</a>
            </td>
            </tr><tr>
            <td colspan="2" class="mcenter">
                      <a href="calendar-appointments">
  Calendar/Appointments</a>
  </td>
  </tr><tr>
  <td colspan="2" class="mcenter">
              <a href="consultants-staff">
                  Consultants/Staff</a>
                  </td>
                  </tr><tr>
                  <td class="mleft"><a
```

```
                   href="aboutus">About Us</a></td>
  <td class="mright">
        <a href="contact-directions">
            Contact/Directions</a>
            </td>
            </tr>
            </table>
            </td>
            </tr>
            </table>
            </div>
            <table border="0" width="95%">
                        <tr><td class="content">
  <h3>Directions from the International Airport</h3>
  <p>
  <ol>
  <li>Start out going NORTH on PENA BLVD. (0.2mi)</li>
  <li>Go STRAIGHT toward TERMINAL EAST. (2.0mi)</li>
  <li>Stay STRAIGHT to go onto PENA BLVD. (3.4mi)</li>
  <li>Take the E-470 TOLLWAY N exit- EXIT 6B-
  toward FORT COLLINS. (0.8mi)</li>
  <li>Merge onto E 470 N (Portions toll). (14.7mi)</li>
  <li>Take the COLORADO BOULEVARD exit- EXIT 43. (0.4mi)</li>
  <li>Turn LEFT onto COLORADO BLVD. (4.2mi)</li>
  <li>Turn RIGHT onto E 120TH AVE / CO-128 W. (1.5mi)</li>
  <li>End at 1283 E 120th Ave - Denver, CO 80233-5728, US</li>
  </ol>
  </p>
  <p class="center" >
       <img src="/images/AirportMap.jpg" width="381" height="255"
  alt="Directions from airport" />
    </p>
    </td></tr>
    </table>
    <div class="footer">
           <p>
           <a href="services">Services</a>
<a href="facilities">Facilities</a>

<a href="calendar-appointments">Calendar/Appointments</a>

<a href="consultants-staff">Consultants/Staff</a>

<a href="aboutus">About Us</a>

<a href="contact-directions">Contact/Directions</a>
    </p>
    </div>
    </div> <!-- Main -->
    </body>
    </html>
```

FIGURE 37-4

Most Web pages contain many elements that are disadvantageous to print, such as headers, graphics, backgrounds, and special features.



Directions from the International Airport

1. Start out going NORTH on PENA BLVD. (0.2mi)
2. Go STRAIGHT toward TERMINAL EAST. (2.0mi)
3. Stay STRAIGHT to go onto PENA BLVD. (3.4mi)
4. Take the E-470 TOLLWAY N exit- EXIT 6B- toward FORT COLLINS. (0.8mi)
5. Merge onto E-470 N (Portions toll). (14.7mi)
6. Take the COLORADO BOULEVARD exit- EXIT 45. (0.4mi)
7. Turn LEFT onto COLORADO BLVD. (1.3mi)
8. Turn RIGHT onto E 120TH AVE / CO-128 W. (1.8mi)
9. End at 1283 E 120th Ave - Denver, CO 80233-5728, US

Services Facilities Calendar/Appointments Consultants/Staff About Us Contact/Directions

As you can see, a handful of superfluous elements on the page aren't only unnecessary to print, but are downright undesirable. Such elements include the following:

- Navigational aids
- Page background colors and graphics
- Graphical headers — image and stylistic text
- Footer navigation links

It is best to reformat or remove these elements from the printed page.

Reformatting the page

Because most of the elements are attached to styles, reformatting the page is fairly trivial. Using the CSS cascade and inheritance properties, you can create additional styles for print media. If the styles are added to the end of the style list, they will take precedence over the previous styles. However, when designated for print media, they will only come into play when printed.

For example, consider the following styles, created for all elements in the document that need different styles when printed. The code shows the styles created to modify elements for printing:

```
@page { size: 8.5in 11in;}
body { background-color: white;}
.main { border: none;
     background-color: white;
     background-image: none;}
.breadcrumbs { display: none;}
.heading { display: none;
           visibility: hidden;}
.heading-content { display: none;}
.printheading { display: block;
                visibility: visible;}
.content { background-color: white;}
.footer { display: none;
          visibility: hidden;}
```

As you can see, there isn't a one-to-one correlation between the styles in the original document and the print media styles, nor the properties of either. The print styles only need to include styles and individual properties needing changes. For example, the print styles concentrate on doing the following:

- Removing images
- Removing borders
- Removing backgrounds
- Hiding irrelevant elements (headers, footers, navigation elements)
- Making desirable print elements previously hidden visible

The last bullet in this list is important — you may have elements that you don't want to print, but which contain information you don't want to lose. In the example, I don't want to print the graphical header but would like to print the name of the website ("The Oasis of Tranquility"). To achieve this effect, I embed a special `div` in the document, give it a class of `printheader`, and use corresponding styles to hide it by default. The print styles then reverse the visibility, hiding the graphical header and revealing the plain text header.

The print styles are placed in a separate style document and linked into the main HTML document and to the end of the embedded styles using a `link` element, similar to the following:

```
<link rel="stylesheet" href="/printstyles.css"
type="text/css" media="print" />
```

Note

In this example, the styles are included in an external style sheet for manageability. They could also be embedded within the document using an `@media` rule. ■

It's important that the print media styles be included at the end of the style list to ensure that the style inheritance is correctly followed. You might have noticed that the styles embedded within the document are not coded for any particular media; this means they are valid for print media as well. If the print-specific styles did not appear at the end of the style list, the general styles could override them.

Tip

When creating media-specific styles for documents, it is generally a good idea to mark every group of styles for the media they should support. ■

Summary

This chapter covered how to use media styles to format a document for printing. You learned how to best define the page, adjust page breaks, and combine media types in one document. The next chapter wraps up Part III of the book, the core CSS coverage. The following part (Chapters 39 through 41) covers more niche CSS topics.

CHAPTER

38

The Future of CSS: CSS3

IN THIS CHAPTER

Modularity

Using CSS3 Properties Today

More Control over Selections

Revisiting the Brass Ring of CSS: Rounded Corners

For the development of Web content, Cascading Style Sheets (CSS) have become the inseparable sibling of HTML. Without CSS, designers cannot create elaborately designed documents; without HTML, designers cannot create any platform for documents.

Introduced in late 1996, CSS level 1 brought a whole new design perspective to the Web. Later, in mid-1998, CSS level 2 was responsible for a new dimension of Web content, positional element, a third dimension of elements, media types, and more. In 2005, CSS level 2.1 was released to fix existing bugs in CSS but did not provide any evolutionary changes to Web design.

CSS level 3, currently in development, promises another leap forward on the Web, introducing new properties and values that bring the current realm of the Web that much closer to printed medium. Although full coverage of the new version could fill a book of its own, this chapter provides an overview of some of the more globally anticipated features of CSS level 3.

Note

As mentioned, CSS3 is currently still in development. Because this chapter was written based on the current *draft* specification, it is likely that some of the material presented here will change. Visit the W3C working site to follow the development of this new version: `www.w3.org/Style/CSS/current-work#table`. ■

Just Better

Much of the work going into this new version of CSS is to make the specification more exacting, to better specify how dynamic elements should be dynamic and how static elements are static. Although CSS is thought of as being fairly mature, the fact that the CSS specification is only entering its fourth revision should be kept in mind — there is still plenty of room in the specification for alternate interpretations. A big part of the CSS2.1 specification was to close various holes and idiosyncrasies in CSS2. Look for CSS3 to continue this effort.

Note

In theory, a more exacting specification should mean more consistent support for the specification. After all, the specification should leave little doubt as to how the various properties should affect document elements. Unfortunately, the actual implementation of the CSS3 specification remains in the user agent vendor's hands — hands that have proven to take a few liberties with specification interpretation. ■

Modularity

CSS3 is being developed in a modular fashion, broken down into 26 distinct modules instead of the monolithic structure of CSS past. The modules and their descriptions are provided in the following table.

Module	Content/Purpose
Syntax/Grammar	A means to attach properties to a structured document
Selectors	Describes the selectors used to marry properties to document elements
Values and Units	Specifies how values and units are described and used
Value Assignment/ Cascade/Inheritance	Describes how various properties interact and how the core of CSS (cascade, inheritance, etc.) operates
Box Model/Vertical	Describes how the box model of CSS operates, causing element flow, floating elements, and providing a third dimension for elements
Positioning	Describes how elements are positioned differently from how they would normally appear in a document
Color/Gamma/Color Profiles	Describes the handling of color in CSS
Colors and Backgrounds	Describes how element foreground and background colors are handled
Line Box Model	Describes the format of the line element box model
Text	Describes the method of text handling in user agents
Fonts	Describes the method of font handling in user agents
Ruby	A draft describing methods of handling new styles of typographic traditions

Module	Content/Purpose
Generated Content/Markers	Describes of how content is generated and markers are displayed
Replaced Content	Describes the definition and handling of replaced content
Paged Media	Describes how documents should handle paged media — headers, footers, etc.
User Interface	Describes how the user interface should handle various states and elements
Web Fonts	Describes the use of fonts and better control over them
ACSS (Aural CSS)	Describes methods of making stylized content more accessible
Synchronized Multimedia Integration Language (SMIL)	Describes how to make CSS and SMIL work together to produce better multimedia
Tables	Describes how table content should be implemented via CSS
Columns	Describes how to use CSS to create flexible column layouts
Scalable Vector Graphics (SVG)	Describes methods that can be used to produce dynamic, scalable vector graphics
Math	Describes how to apply stylistic conventions to mathematic expressions
Behavioral Extensions to CSS (BECSS)	Describes methods to attach behaviors (rather than styles) to elements
Media Queries	Describes methods of dynamically determining what styles, from what source, should be applied

The reasoning behind breaking CSS into a modular structure is simple: It makes the project much easier to manage. As the popularity of CSS increases, so does the user's desire to extend the capabilities of the standard. Using a modular structure provides a more efficient way of extending features and capabilities while still being able to adequately manage the whole.

Using CSS3 Properties Today

For various reasons, some of the major user agents have implemented CSS3 properties, even though the specification isn't yet final. These user agents include the Firefox browser and browsers that rely upon the Webkit framework, such as Mac Safari.

To maintain compatibility with other browsers and older versions of CSS, these browsers implement CSS3 features by using unique property names. These property names use prefixes such as `-moz` and `-webkit`, making the property definitions unrecognizable to other browsers.

Unfortunately, support of CSS3 properties is sketchy at best, and the browsers are apt to use property names that only resemble the actual CSS properties.

For example, all modern user agents support the `border-<side>-color` property to CSS2.1 levels, whereas Firefox 3.1+ also supports the CSS3 level of this property, via the `-moz-border-<side>-colors` property. The CSS3 specification enables the Web author to use a range of colors in the `-moz-border-<side>-colors` definition to achieve unique results.

Cross-Ref

See the section "Revisiting the Brass Ring of CSS: Rounded Corners," later in this chapter for more information on border colors and CSS3. ■

To ensure that use of the browser-specific properties does not adversely affect documents, the properties should appear in the following order:

1. Standard CSS2.1 format
2. Browser-specific CSS3 format

This enables all browsers to pick up the CSS2.1 property, ensuring that the non-CSS3 browsers receive their property value(s), and then allows the CSS3-enabled browsers to receive their values, overwriting the CSS2.1 value.

For example, consider the border color example. If you wanted to infuse your documents with that CSS3 property for bottom borders, you should use definitions similar to the following:

```
div  { border-bottom-color: black; }
div  { -moz-border-bottom-colors: #F9F9F9 #AFAFAF
                            #BFBFBF #CFCFCF
                            #DFDFDF #EFEFEF
                            #FFFFFF; }
```

These two definitions cover non-CSS3-enabled user agents as well as Firefox 3.1+ browsers that support the extended property. As a result, non-CSS3 user agents will use black borders for `div` elements while Firefox will use the extended definition resulting in the gradient.

Note

The website `www.css3.info/` is a good source of information on CSS level 3, including current browser support of its features. ■

More Control over Selections

Not surprisingly, CSS3's selector interface is much more robust than that of CSS2.1. The new selector formats provide the capability to select almost any element in a document. The new CSS3 selectors are summarized in the following table.

Selector Format	Selects . . .
`E[foo^="bar"]`	an E element whose “foo” attribute value begins exactly with the string “bar"
`E[foo$="bar"]`	an E element whose “foo” attribute value ends exactly with the string “bar"
`E[foo*="bar"]`	an E element whose “foo” attribute value contains the substring “bar"
`E:root`	an E element, root of the document
`E:nth-child(n)`	an E element, the n^{th} child of its parent
`E:nth-last-child(n)`	an E element, the n^{th} child of its parent, counting from the last one
`E:nth-of-type(n)`	an E element, the n^{th} sibling of its type
`E:nth-last-of-type(n)`	an E element, the n^{th} sibling of its type, counting from the last one
`E:last-child`	an E element, last child of its parent
`E:first-of-type`	an E element, first sibling of its type
`E:last-of-type`	an E element, last sibling of its type
`E:only-child`	an E element, only child of its parent
`E:only-of-type`	an E element, only sibling of its type
`E:empty`	an E element that has no children (including text nodes)
`E:target`	an E element being the target of the referring URI
`E:checked`	a user interface element E which is checked (for instance a radio-button or checkbox)
`E:not(s)`	an E element that does not match simple selector s
`E ~ F`	an F element preceded by an E element

Revisiting the Brass Ring of CSS: Rounded Corners

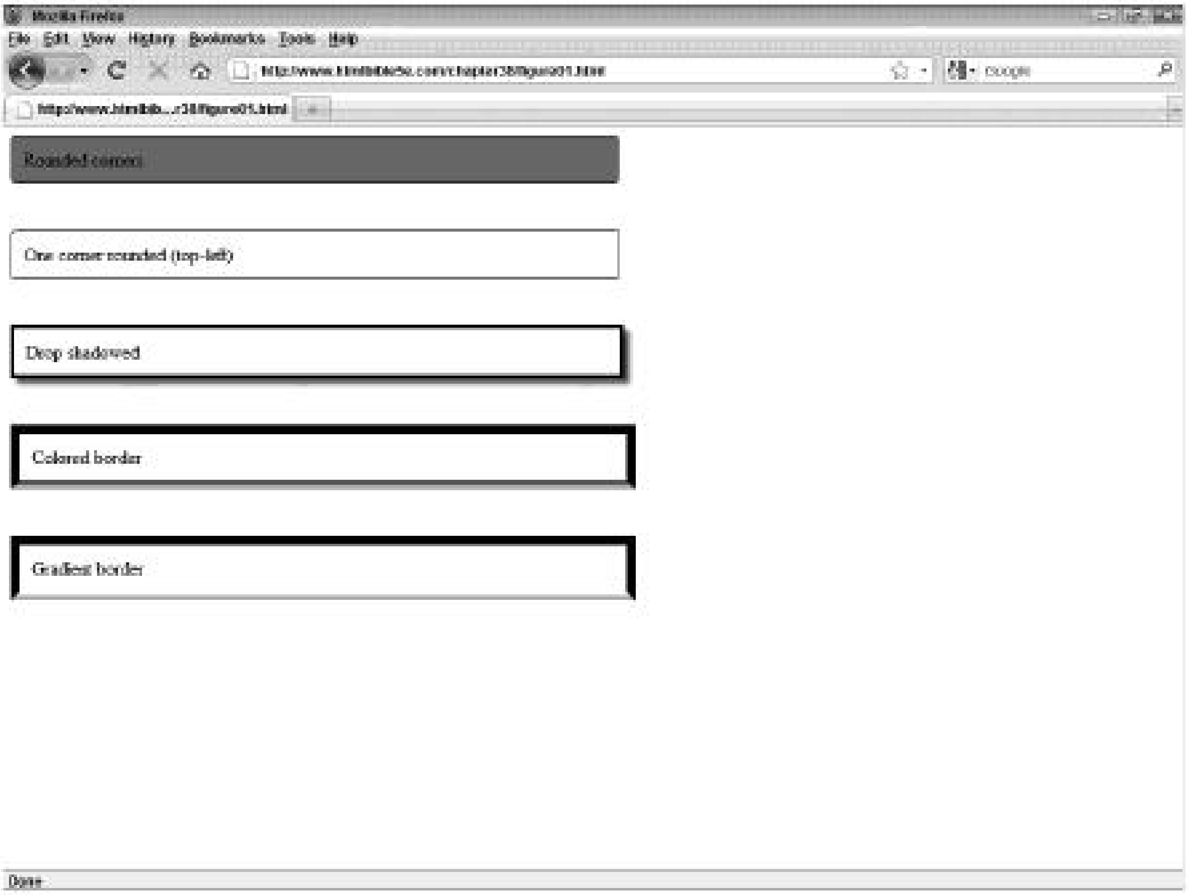
As previously mentioned in this book, designers have been clamoring for the ability to provide rounded corners to elements. As of this writing, the number of ways to do this without pure CSS is numbered in the 50s. These methods use uniquely placed elements, special graphics, and other optical tricks to simulate rounded corners. CSS3 simplifies the matter by providing rounded corner properties for block elements. As of this writing, Firefox 3.1+ supports rounded corners. Besides rounded corners, Firefox also supports multiple-color borders, enabling designers to make document elements have a unique and 3D appearance. For example, consider the following code, and Figure 38-1, showing how the document renders in Firefox 3.5, thanks to the -moz properties:

```
<html>

<head>
<style type="text/css">

div { clear: both;
      width: 50%;
      padding: 10px;
      margin-bottom: 40px; }


#rounded { background-color: gray;
           border: 1px solid black;
           -moz-border-radius: 5px; }

#shadowed { background-color: white;
            border: 3px solid black;
            -moz-box-shadow: 5px 5px 5px #666;}


#colored { border-width: 8px;
           border-style: solid;
           -moz-border-bottom-colors: red yellow blue
                                      yellow green;}


#gradient { border-width: 8px;
            border-style: solid;
            -moz-border-bottom-colors: #F9F9F9 #AFAFAF
                                       #BFBFBF #CFCFCF
                                       #DFDFDF #EFEFEF
                                       #FFFFFF; }

#partial { border: 1px solid black;
           -moz-border-radius-topleft: 5px; }


</style>
</head>

<body>
<div id="rounded">Rounded corners</div>
<div id="partial">One corner rounded (top-left)</div>
<div id="shadowed">Drop shadow</div>
<div id="colored">Colored border</div>
```

```
<div id="gradient">Gradient border</div>
</body>
</html>
```

FIGURE 38-1

CSS level 3 enables much more control over the components of elements, such as borders.


Mozilla Firefox
File Edit View History Bookmarks Tools Help
http://www.htmlbible5e.com/chapter38figure01.html
Google
http://www.htmlbib...r38figure01.html
Rounded corners
One corner rounded (top-left)
Drop shadowed
Colored border
Gradient border
Done

Note that the border color properties enable multiple colors to be defined. This allows borders to take on a rainbow effect or a 3D appearance if a smooth gradient of colors is specified. In addition, each side or corner of an element can be styled individually.

Summary

CSS level 3 has been in development since 1998. That has been plenty of time for the work group behind it to witness how the Web has matured and what is necessary to help revolutionize Web content.

Unfortunately, the CSS3 standard is still several years away from release; and even after the release, designers will have to wait for user agent adoption of the standards, and then user adoption of those new user agents. Nonetheless, it will be a welcome addition to the Web, providing a whole new generation of design possibilities.

Part IV

Additional CSS Tools

IN THIS PART

CHAPTER

39

User Interface Styles

IN THIS CHAPTER

Changing the Cursor

User Interface Colors

User Interface Fonts

In Part III of this book you learned how to use CSS to style and format almost every part of an HTML document. However, there are several additional, user agent-related elements you can affect with CSS. This chapter shows you how to use styles on user interface elements — the mouse pointer, system colors, and system fonts.

Changing the Cursor

The CSS `cursor` property enables you to specify what cursor type should be displayed when the cursor is over a specific element. This property is used like any other property, with a familiar format:

```
cursor: value;
```

The `cursor` property supports the values listed in Table 39-1.

The `uri` value takes the following familiar form:

```
url("uri_path_to_resource")
```

This value is unique in that it supports several values and can be followed by a default value if none of the `uri` resources can be found or used. For example, the following property definition defines two external pointer files that should be used, and a fallback default of `crosshair` if those two resources cannot be used for some reason:

```
cursor: url("angle.cur"), url("simple.cur"), crosshair;
```

Note

Many general graphic editing programs and several specific programs can be used to create custom cursors. Try searching Google for "create cursor file." ■

TABLE 39-1

Values for the Cursor Property

Value	Description
`auto`	The user agent displays an appropriate cursor for the current context.
`crosshair`	The cursor is set to the shape of a simple crosshair (resembling a narrow + sign).
`default`	The cursor is set to the platform's default cursor (typically an arrow).
`pointer`	The cursor is set to a shape that typically indicates a link (typically a pointing hand).
`move`	The cursor is set to a shape indicating that something can be moved (typically a four-pointed arrow).
`e-resize`, `ne-resize`, `nw-resize`, `n-resize`, `se-resize`, `sw-resize`, `s-resize`, or `w-resize`	The cursor is set to a shape indicating that something can be resized (typically a multi-headed arrow showing the direction(s) that an object can be sized). The leading letters refer to the edge of the sizable element that can be sized — `ne-resize`, for example, refers to an element's northeast (top-left corner) edge.
`text`	The cursor is set to a text edit/insert cursor (typically an I-beam cursor).
`wait`	The cursor is set to a shape that indicates the user should wait (typically an hourglass or clock image).
`progress`	The cursor is set to a shape that indicates the computer is in the process of doing an operation and the user might have to wait (typically an hourglass or clock).
`help`	The cursor is set to a shape indicating help is available, usually by clicking the object under the cursor. (The shape is typically a pointer with a question mark or a text balloon.)
`<uri>`	The cursor is set to a shape stored in an external cursor resource. This value supports multiple values in the form of a comma-separated list — if the first entry is unavailable, the second is used, and so forth.

Although CSS provides resize cursors for every side and corner of an element, they are not dynamic — as the cursor moves around the border of an element, the appropriate cursor will not automatically appear. You must assign the appropriate cursor to elements that can use the appropriate resize arrow.

Figure 39-1 shows how the default arrow cursor can be changed to a hand cursor (`pointer`) for the button, using the following code:

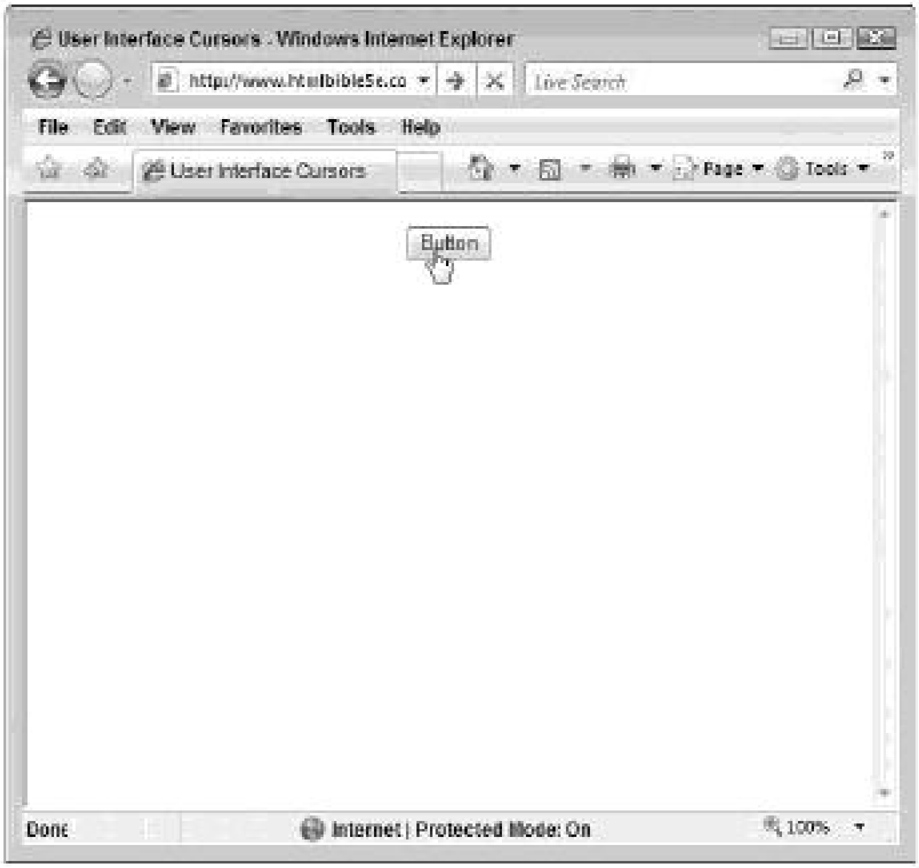
```
<image type="button" value="Button" style="cursor:pointer;" />
```

Tip

Just because you can change the cursor doesn't mean you should. Graphical user environments rely on consistency to increase user familiarity and comfort with their systems. If the cursor changes randomly in your pages, you risk confusing and alienating users. ■

FIGURE 39-1

A custom cursor


User Interface Cursors - Windows Internet Explorer
http://www.htmlbible5e.co
Live Search
File
Edit
View
Favorites
Tools
Help
User Interface Cursors
Page
Tools
Button
Done
Internet | Protected Mode: On
100%

User Interface Colors

In addition to changing the cursor to system default cursors, you can also change your document's colors to match system colors. Table 39-2 lists the keyword values you can use with the `color` and `background-color` properties to match colors in your document to the current system colors.

TABLE 39-2

System Color Keywords

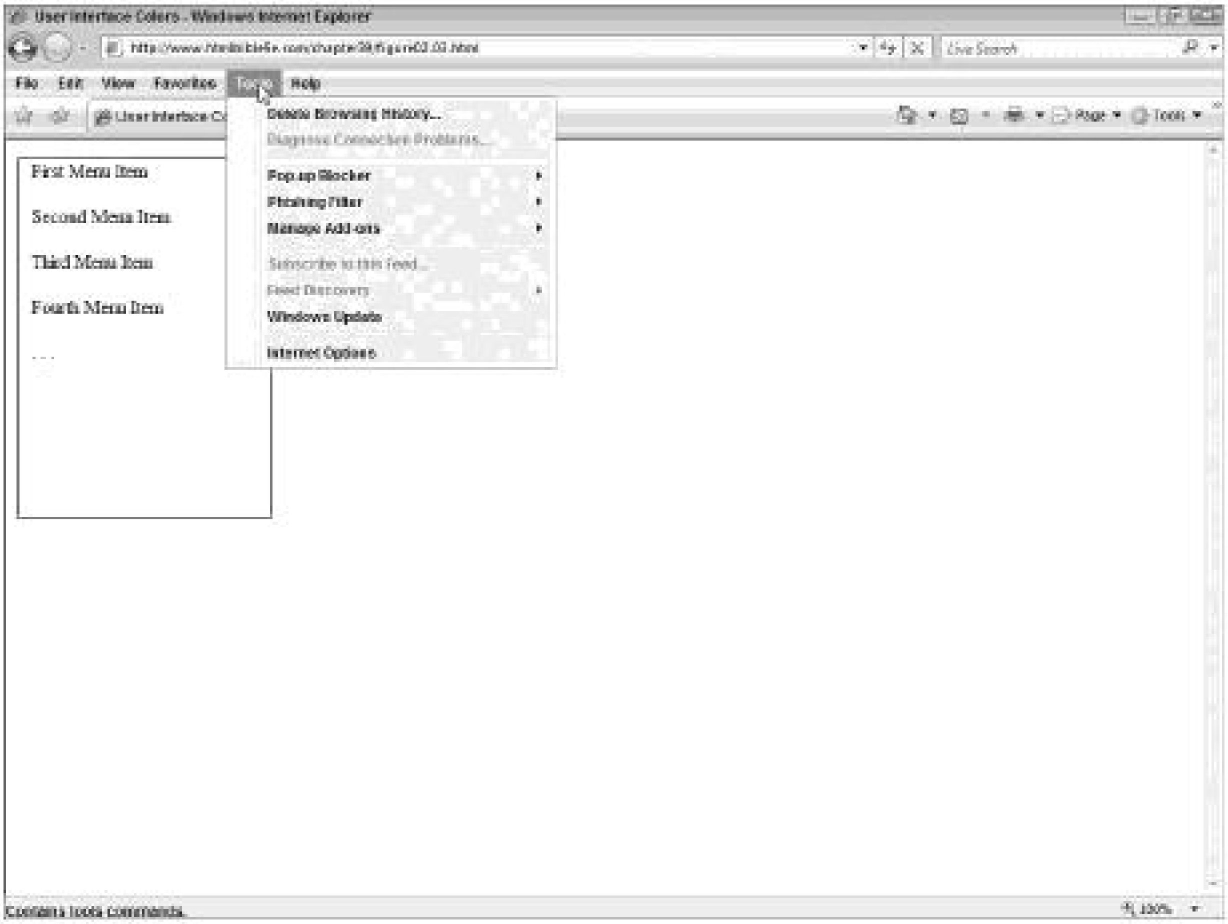
Keyword	Matches This System Color
`ActiveBorder`	The border of the current (active) window
`ActiveCaption`	The caption text of the current (active) window
`AppWorkspace`	The background color of a multiple-document interface (usually the background color of a document window)
`Background`	The background color (not image) of the desktop
`ButtonFace`	The face of button elements
`ButtonHighlight`	The dark shadow area on the edge of 3D button elements
`ButtonShadow`	The shadow area on the edge of 3D button elements
`ButtonText`	The text of button elements
`CaptionText`	The text in captions, size boxes, and scrollbar arrow boxes
`GrayText`	Gray (disabled) text
`Highlight`	Selected item(s)
`HighlightText`	The selected text
`InactiveBorder`	The border of an inactive window
`InactiveCaption`	The caption of an inactive window
`InactiveCaptionText`	The caption text of an inactive window
`InfoBackground`	The background for tooltips
`InfoText`	The text of tooltips
`Menu`	The background of menus
`MenuText`	The text of menus
`Scrollbar`	The "gray" area of scrollbars
`ThreeDDarkShadow`	The dark shadow for 3D elements
`ThreeDFace`	The face of 3D elements
`ThreeDHighlight`	The highlight color for 3D elements
`ThreeDLightShadow`	The light color shadow for 3D elements
`ThreeDShadow`	The dark color shadow for 3D elements
`Window`	The window background (for nondocument windows)
`WindowFrame`	The window frame
`WindowText`	The text in windows

These keywords can be used as you would any other color keyword, such as `blue`, `red`, or `green`. For example, to set up a paragraph to mimic system menus — background and text — you could use the following tag with its embedded styles:

```
<p style="background-color: Menu; color: MenuText">
```

FIGURE 39-2

A document with one system color scheme



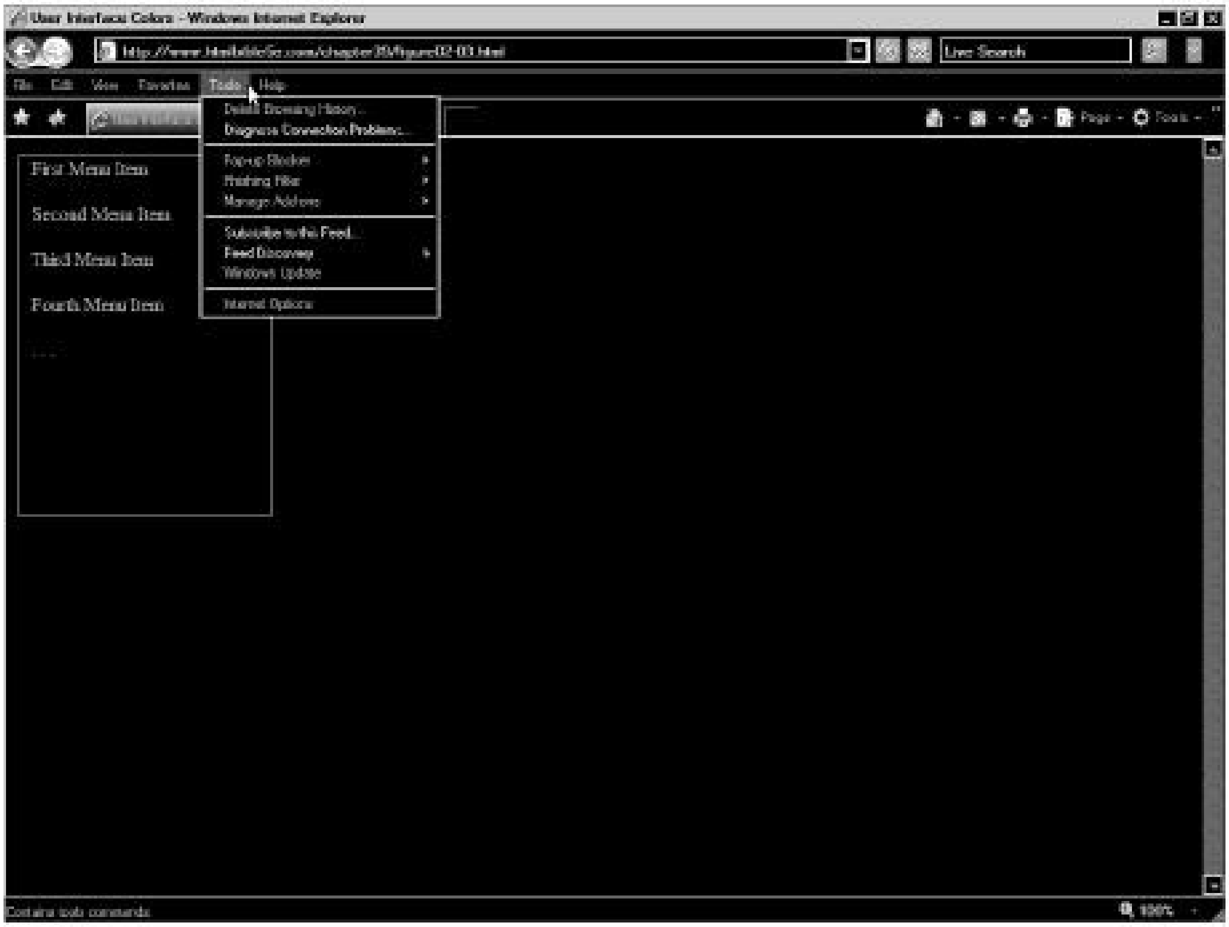
A big advantage to these keywords is that they are dynamic. As system colors change, so do the colors these keywords represent, so your document colors will match end users' colors even if they change. For an example of this effect, consider the following code and Figures 39-2 and 39-3, where the system color scheme changes between the figures (as referenced by the extended menu):

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01//EN"
  "http://www.w3.org/TR/html4/strict.dtd">
<html>
<head>
  <title>User Interface Colors</title>
  <style type="text/css">
```

```
    div { padding: 0px 0px 0px 10px;
          border: 1pt solid black;
          width: 200px;
          height: 300px;
          background-color: Menu;}
    div p { color: MenuText;}
  </style>
</head>
<body>
<div>
   <p>First Menu Item</p>
   <p>Second Menu Item</p>
   <p>Third Menu Item</p>
   <p>Fourth Menu Item</p>
   <p>. . .</p>
</div>
</body>
</html>
```

FIGURE 39-3

The same document with another system color



User Interface Fonts

CSS has a handful of interface font keywords that can be used similarly to the user interface color keywords to create documents that mesh with users' system interfaces. The interface font keywords set values of the `font` property. Table 39-3 lists the interface font keywords.

TABLE 39-3

Interface Font Keywords

Keyword	Matches This System Font
`caption`	Text on user interface elements
`icon`	Labels on icons
`menu`	Text in menus
`message-box`	Text in dialog boxes
`small-caption`	Text in small controls
`status-bar`	Text in window status bars

Like the color keywords, the font keywords are dynamic — the font they represent will change as the system fonts change.

Tip

Using the right combinations of user interface styles can give your document a look that's virtually indistinguishable from a user's system interface. Keep in mind that your document must still behave like a Web document if your audience approaches your document from a Web viewpoint. However, if your audience views your document as an application, infusing it with more of a computer interface design would be the better choice. In short, create a document and interface that will be familiar to your audience. ■

Summary

In this chapter you learned about the CSS user interface styles and how they can be used to customize a user agent's pointer, colors, and fonts. Using these styles you can easily change what the user interface of your document resembles. In the next chapter, you'll learn how to test and validate your CSS code.

CHAPTER

40

Testing and Validating CSS

IN THIS CHAPTER

As you have seen in this book, HTML and CSS provide a yin and yang approach to the content and formatting of Web documents. As such, each depends on the other being complete, sturdy, and robust.

In Chapter 23 you saw the benefits of well-formed and validated HTML. This chapter explores the other piece of the equation — well-formed and validated CSS.

Testing Syntax As You Create Styles

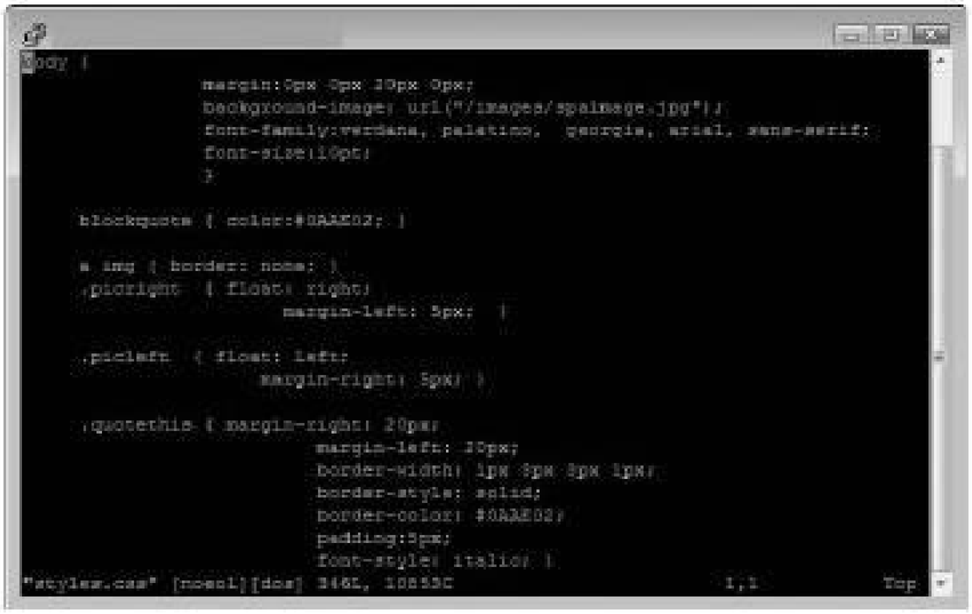
The best and easiest method to ensure that your CSS is valid is to check its syntax as you create it. This means using a syntax highlighting, syntax checking, and text auto-insertion CSS editor for even the first draft of your styles.

Syntax highlighting involves using different colors for different portions of code. When editing CSS, for example, an editor might display the selector portion of a style in green, the braces in white, the properties in red, and the values in yellow. If a piece of the style is missing, the color of the previous element usually bleeds into the next section of the file. Using these visual cues, you can easily see the various pieces of the whole and recognize missing portions (such as a closing brace). These rudimentary editors also include features like auto-indenting of code to help keep your code tidy and easy to read. Figure 40-1 shows the popular Linux editor vim.

Syntax checking involves actively checking code as you type it and sometimes offering to auto-insert the next required piece for you. When editing CSS, for example, an editor might pop up a completion dialog box after you type a selector and its trailing space. The dialog box would contain an open brace ({) and perhaps a handful of selector symbols, such as a

FIGURE 40-1

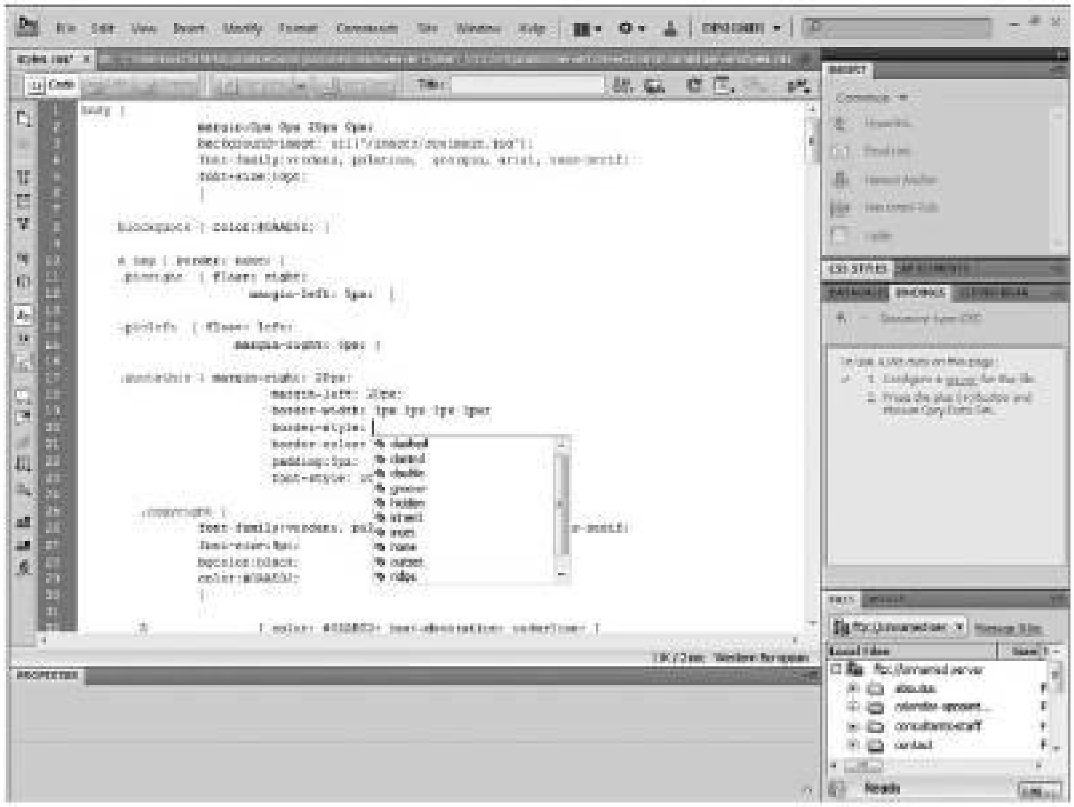
Many text editors, like vim (shown here), offer syntax highlighting, automatic code indentation, and more.



```
body {
                margin:0px 0px 20px 0px;
                background-image: url("/images/spaimage.jpg");
                font-family:verdana, palatino,  georgia, arial, sans-serif;
                font-size:10pt;
                }

     blockquote { color:#0AAE02; }

     a img { border: none; }
     .picright  { float: right;
                       margin-left: 5px;  }

     .picleft  { float: left;
                     margin-right: 5px; }

     .quotethis { margin-right: 20px;
                          margin-left: 20px;
                          border-width: 1px 5px 5px 1px;
                          border-style: solid;
                          border-color: #0AAE02;
                          padding:5px;
                          font-style: italic; }
"styles.css" [noeol][dos] 346L, 10853C                    1,1          Top
```

FIGURE 40-2

A few specialty Web editors, such as Dreamweaver (shown here), offer syntax checking and text auto-completion.



```
body {
               margin:0px 0px 20px 0px;
               background-image: url("/images/spaimage.jpg");
               font-family:verdana, palatino,  georgia, arial, sans-serif;
               font-size:10pt;
               }

     blockquote { color:#0AAE02; }

     a img { border: none; }
     .picright  { float: right;
                     margin-left: 5px;  }

     .picleft  { float: left;
                   margin-right: 5px; }

     .quotethis { margin-right: 20px;
                       margin-left: 20px;
                       border-width: 1px 5px 5px 1px;
                       border-style: |
                       border-color:  dashed
                       padding:5px;   dotted
                       font-style: it double
                                      groove
                                      hidden
      .copyright {                    inherit
            font-family:verdana, pal  inset                  s-serif;
            font-size:8px;            none
            bgcolor:black;            outset
            color:#0AAE02;            ridge
            }

     a              { color: #0AAE02; text-decoration: underline; }
```

1K / 1 sec Western European

class indicator (.), a child separator (>), or an adjacent sibling separator (+). One keyboard stroke or mouse click can insert any of the suggested characters; and when a second character is necessary to complete a pair (as with braces), the second character is inserted as well. Savvy code editors know CSS properties and can offer to auto-insert them as well, reducing the chance of typing errors. Figure 40-2 shows a CSS file being edited in Dreamweaver, with Dreamweaver offering a dialog box of style properties that can be auto-inserted.

Using even the simplest editor with these features enables you to retain better control over your CSS documents as you edit them, helping to ensure that they begin, and stay, error-free.

A Word About Formatting

As you have probably noticed from examples in this book and other sources, you can format your CSS definitions in many ways. For example, consider the following definition, which adheres to CSS standards but is shown formatted three different ways:

```
h1 { font-size: 16px; margin: 0px; padding: 0px;
color: #0AAE02;}
h1 { font-size: 16px;
     margin: 0px;
     padding: 0px;
     color: #0AAE02;}
h1 {
  font-size: 16px;
  margin: 0px;
  padding: 0px;
  color: #0AAE02;
}
```

Each of the preceding examples is completely valid, although the last example seems to be preferred by experts such as the W3C. (It's the format the Jigsaw validation tool uses.)

When creating your styles, it is a good idea to pick a formatting style that you are comfortable with and use it exclusively. The routine will help you spot common errors, enabling you to write valid CSS out of the gate.

Validating CSS

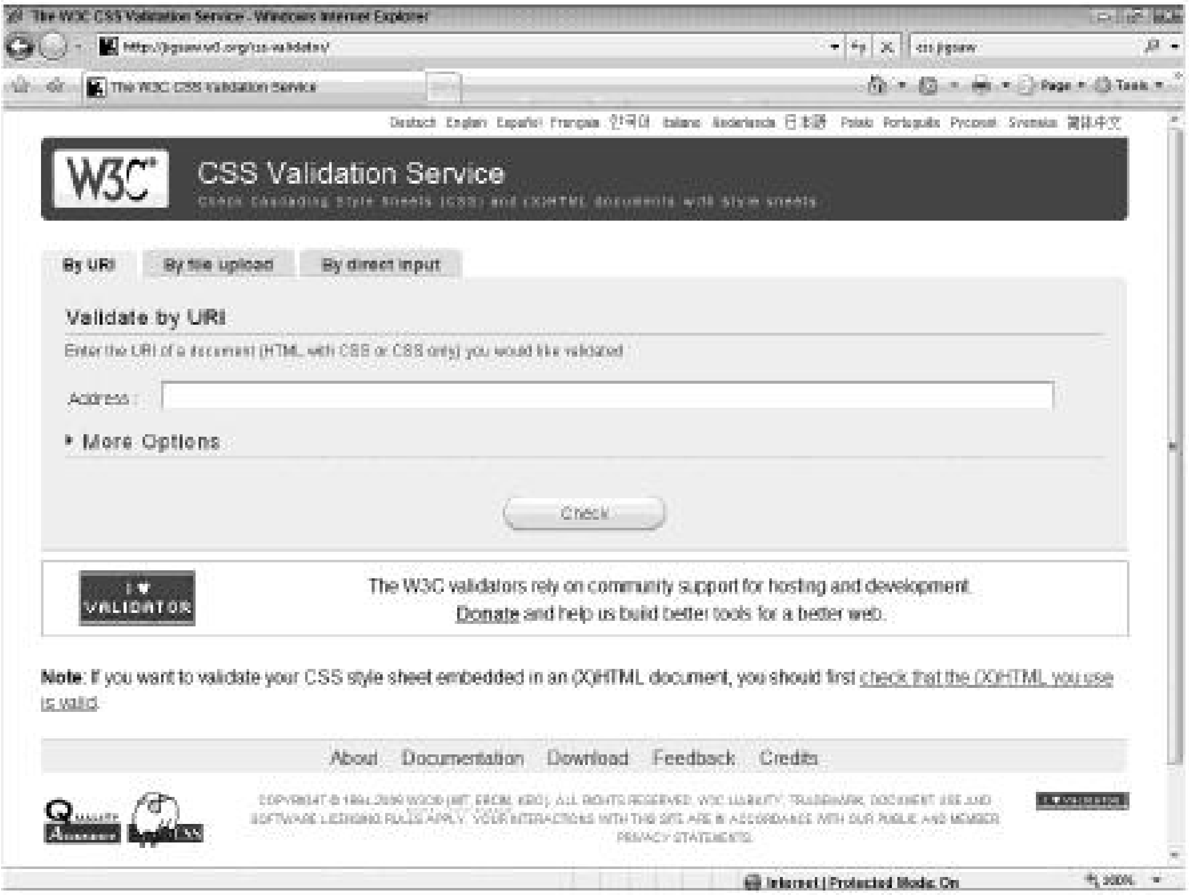
Several tools are available for validating your CSS code — both online tools and applications. One of the best tools available is the W3C tool, code-named Jigsaw, which is available at `http://jigsaw.w3.org/css-validator/` and shown in Figure 40-3.

The tool enables you to provide your CSS via a URL or file upload, or by directly inputting your code into the tool. You can also tailor the types of warnings you receive, the CSS profile that should be used to check your code, and the intended medium for your styles (aural, print, screen, and so on).

One advantage of this tool is that it was developed and is maintained by the W3C, arguably the absolute authority for CSS.

FIGURE 40-3

The W3C Jigsaw tool provides in-depth validation for CSS files


The W3C CSS Validation Service - Windows Internet Explorer
http://jigsaw.w3.org/css-validator/
css jigsaw
The W3C CSS Validation Service
Page
Tools
Deutsch English Español Français 한국어 Italiano Nederlands 日本語 Polski Português Русский Svenska 简体中文
W3C
By URI
By file upload
By direct input
Check
I ♥ VALIDATOR
About
Documentation
Download
Feedback
Credits
Internet | Protected Mode: On
100%

CSS Validation Service

Check Cascading Style Sheets (CSS) and (X)HTML documents with style sheets

Validate by URI

Enter the URI of a document (HTML with CSS or CSS only) you would like validated

Address :

▸ More Options

The W3C validators rely on community support for hosting and development.
Donate and help us build better tools for a better web.

Note: If you want to validate your CSS style sheet embedded in an (X)HTML document, you should first check that the (X)HTML you use is valid.

COPYRIGHT © 1994-2009 W3C® (MIT, ERCIM, KEIO), ALL RIGHTS RESERVED. W3C LIABILITY, TRADEMARK, DOCUMENT USE AND SOFTWARE LICENSING RULES APPLY. YOUR INTERACTIONS WITH THIS SITE ARE IN ACCORDANCE WITH OUR PUBLIC AND MEMBER PRIVACY STATEMENTS.

When your code is run through the tool, it will indicate any errors it finds and display the rest of your code in a particular format, as shown in Figure 40-4.

Firefox Add-ons for CSS Editing

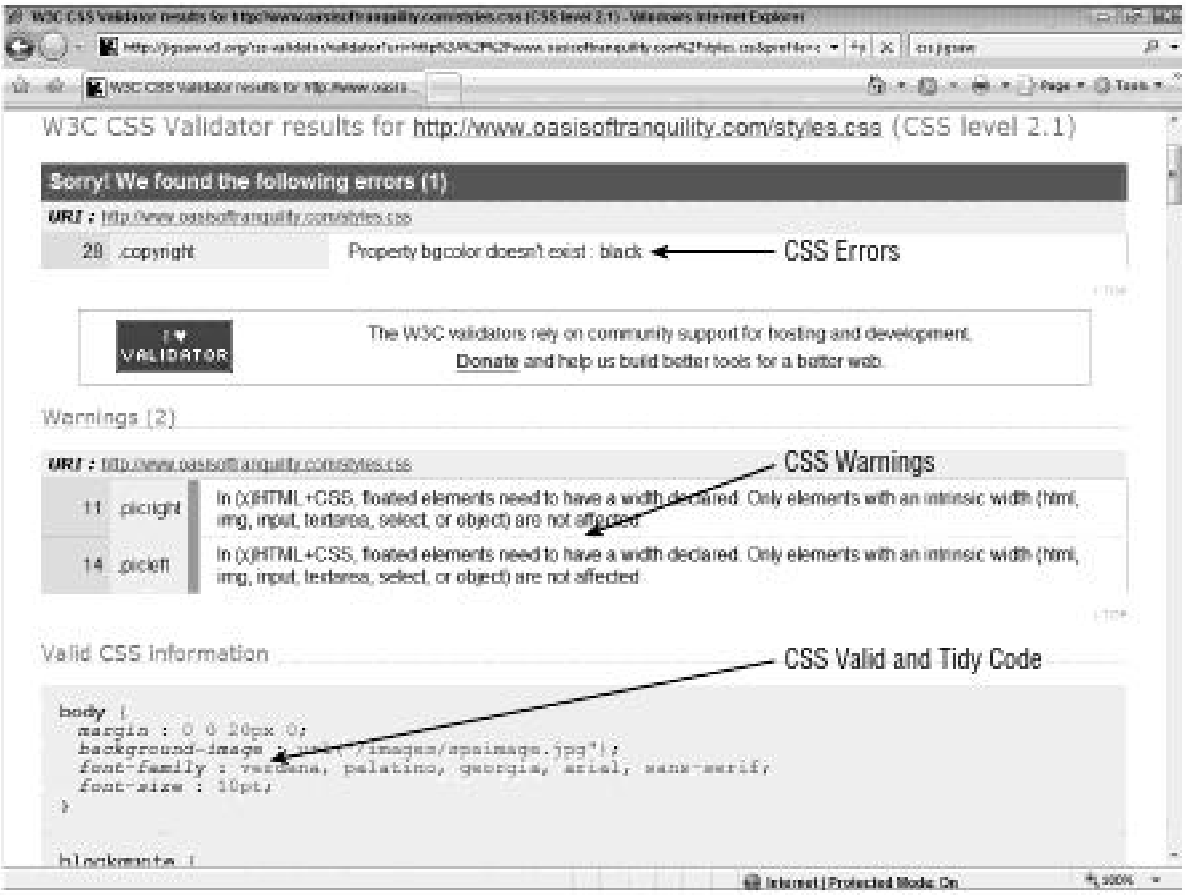
As much as nature abhors a vacuum, Web developers abhor a lack of tools to edit Web code. To that end, many Firefox add-ons have been developed to aid in the creation and maintenance of CSS code. Piggybacking on the functionality of the Firefox browser, these tools provide very capable (and usually free) means of editing CSS.

One such tool is EditCSS, currently in pre-release version 0.3.7 (`https://addons.mozilla.org/en-US/firefox/addon/179`). Using a handful of editing features, EditCSS enables you to edit a currently loaded stylesheet.

Another tool is CSS Validator, currently in its third version (`https://addons.mozilla.org/en-US/firefox/addon/2289`). CSS Validator gives developers access to a handful of tools that can be used to correct validation errors it finds using its validation mode.

FIGURE 40-4

The Jigsaw tool both displays errors and formats the rest of your code.



W3C CSS Validator results for http://www.oasisoftranquility.com/styles.css (CSS level 2.1) - Windows Internet Explorer

W3C CSS Validator results for http://www.oasisoftranquility.com/styles.css (CSS level 2.1)

Sorry! We found the following errors (1)

URI : http://www.oasisoftranquility.com/styles.css

28	.copyright	Property bgcolor doesn't exist : black

The W3C validators rely on community support for hosting and development.
Donate and help us build better tools for a better web.

Warnings (2)

URI : http://www.oasisoftranquility.com/styles.css

11	.picright	In (x)HTML+CSS, floated elements need to have a width declared. Only elements with an intrinsic width (html, img, input, textarea, select, or object) are not affected
14	.picleft	In (x)HTML+CSS, floated elements need to have a width declared. Only elements with an intrinsic width (html, img, input, textarea, select, or object) are not affected

Valid CSS information

```
body {
  margin : 0 0 20px 0;
  background-image : url("/images/spaimage.jpg");
  font-family : verdana, palatino, georgia, arial, sans-serif;
  font-size : 10pt;
}

blockquote {
```

Other tools like Firebug, in version 1.4.2 (`https://addons.mozilla.org/en-US/firefox/addon/1843`), are geared more toward development than they are validation, but their ability to edit and manipulate styles provide some valuable insight during development.

Note

More plug-ins for Firefox are being developed every day. You are encouraged to visit the Firefox add-on site (`https://addons.mozilla.org/en-US/firefox/`) frequently to browse for additional tools. ■

Summary

Although CSS follows a very simple and strict format structure, it is still easy to incorrectly format a selector or improperly use a value keyword. This chapter presented a few ways you can double-check the formatting and validity of your CSS code, helping keep such errors to a minimum. The next chapter covers a handful of CSS tips and tricks you can use in your documents.

CHAPTER

41

CSS Tips and Tricks

IN THIS CHAPTER

Throughout this part, I've covered CSS from an elementary, technical standpoint. This chapter provides a handful of practical examples demonstrating how CSS can be used in unique yet useful ways.

Hanging Indents

The hanging indent, where all but the first line of a paragraph is indented, is a staple of the publishing world. As documentation on the Web becomes more reflective of the traditional printed page, the need for publishing conventions continues to grow.

For clarity, the following is an example of a hanging indent:

```
The Oasis of Tranquility offers a full menu of services
     to renew the real you that lies within. Begin
     in one of our two relaxation centers, then enjoy
     an invigorating body and facial care, deep soothing
     massage therapies, and a host of other indulgent
     treatments that pamper you on the outside, and
     revive you from within. In addition to our many
     spa services, take a refreshing dip in the swimming
     pool, melt in one of our whirlpool spas, or rejuvenate
     in the sauna.
```

Examining the list of indent, padding, margin, and alignment properties in CSS might give you quite a few ideas about how to accomplish this formatting. There is also the `:first-line` pseudo-element; despite its sketchy adoption in user agents, it might be the perfect solution.

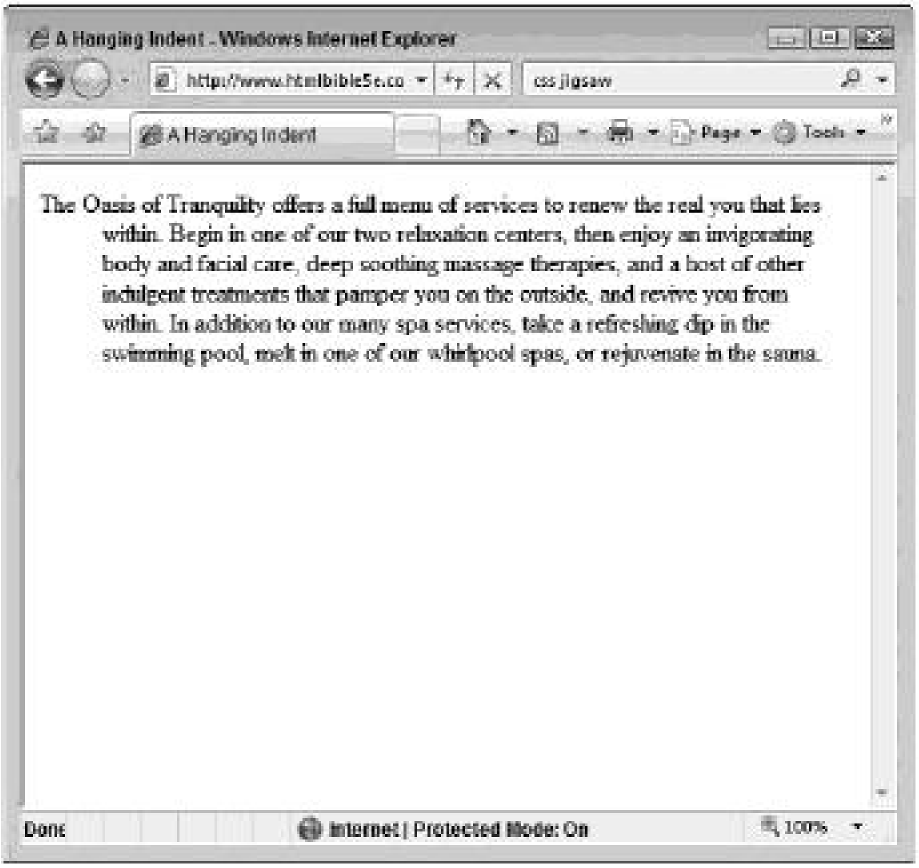
However, the solution is quite simple: utilize the `text-indent` and `margin-left` properties, as shown in the following code and Figure 41-1:

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

```
<html>
<head>
<title>A Hanging Indent</title>
<style type="text/css">
  p.hang { text-indent: -40px;
           margin-left: 40px;}
</style>
</head>
<body>
<p class="hang">The Oasis of Tranquility offers a full menu of
services to renew the real you that lies within. Begin in one of
our two relaxation centers, then enjoy an invigorating body and
facial care, deep soothing massage therapies, and a host of
other indulgent treatments that pamper you on the outside, and
revive you from within. In addition to our many spa services,
take a refreshing dip in the swimming pool, melt in one of our
whirlpool spas, or rejuvenate in the sauna.</p>
</body>
</html>
```

FIGURE 41-1

A hanging indent is relatively easy to achieve.


A Hanging Indent - Windows Internet Explorer
http://www.htmlbible5e.co
css jigsaw
A Hanging Indent
Page
Tools
Done
Internet | Protected Mode: On
100%

The Oasis of Tranquility offers a full menu of services to renew the real you that lies within. Begin in one of our two relaxation centers, then enjoy an invigorating body and facial care, deep soothing massage therapies, and a host of other indulgent treatments that pamper you on the outside, and revive you from within. In addition to our many spa services, take a refreshing dip in the swimming pool, melt in one of our whirlpool spas, or rejuvenate in the sauna.

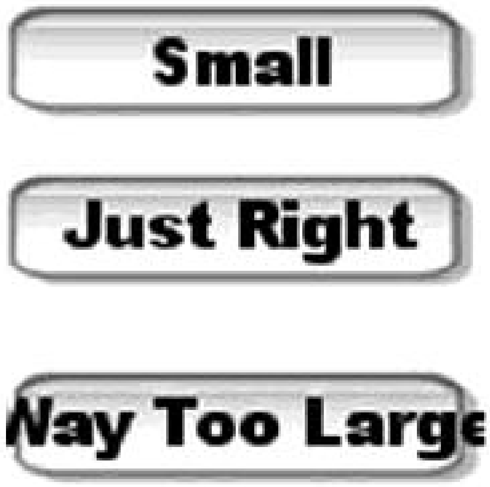
The trick is to set the paragraph's `text-indent` property to the exact negative of the `left-margin` property. The `text-indent` property controls the indentation of the first line of a block — our paragraph in this example. Setting this property to a negative value results in the first line escaping the overall margin.

Expanding Buttons

For many years, graphic images have been used as the background for buttons. However, for as many years these images suffered from a particular flaw: They didn't grow or shrink with the size of the text used for the button. An example of this is shown in Figure 41-2, where different-size captions are placed over the same button.

FIGURE 41-2

Small buttons with large type or large buttons with small type can look unprofessional.



Thankfully, several industrious souls have created methods for creating "shrink-wrapped" buttons, whereby the graphic acts like the plastic of the same name and wraps itself around the text it is given. The key to these buttons is a layered graphic approach, where one part of the button code contains the left half of the image and another part of the code contains the right. The result is a type of sliding door — like a set of glass patio doors — that slide together or apart, as necessary, to span the required gap.

Figure 41-3 shows a visual representation of how this concept works.

The key is that the right side of the button slides on top of the left, and the left side stops where the right side is placed. This keeps the extra length of the left side of the button from jutting out when the whole comes together.

Another image technique that has become popular for buttons is to place several images together in one background image and use the `background-position` property to slide the required image into place when it is needed. For example, it is a popular practice to place the inactive and active button graphics together in one image. The combined image is then placed as the button background such that only the inactive image is visible. When the button is activated, CSS rules cause the background graphic to shift and display the active image. Figure 41-4 illustrates this concept.

FIGURE 41-3

One approach to dynamically sized buttons is the "sliding door" approach–using two pieces of the button that can slide together or apart as necessary.

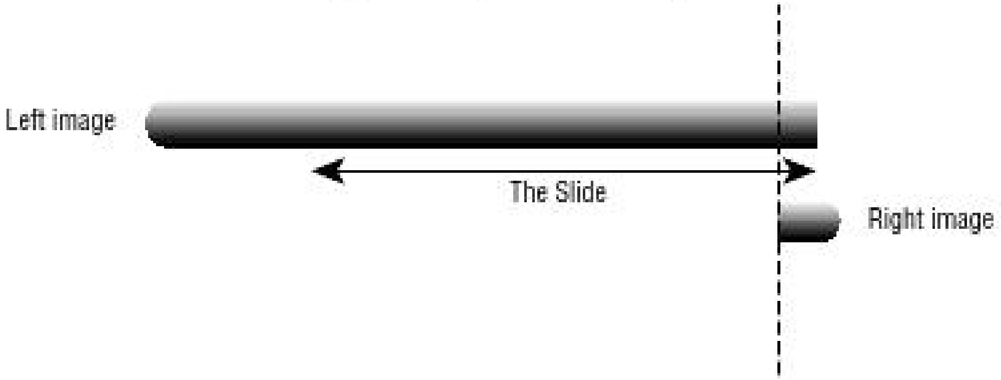
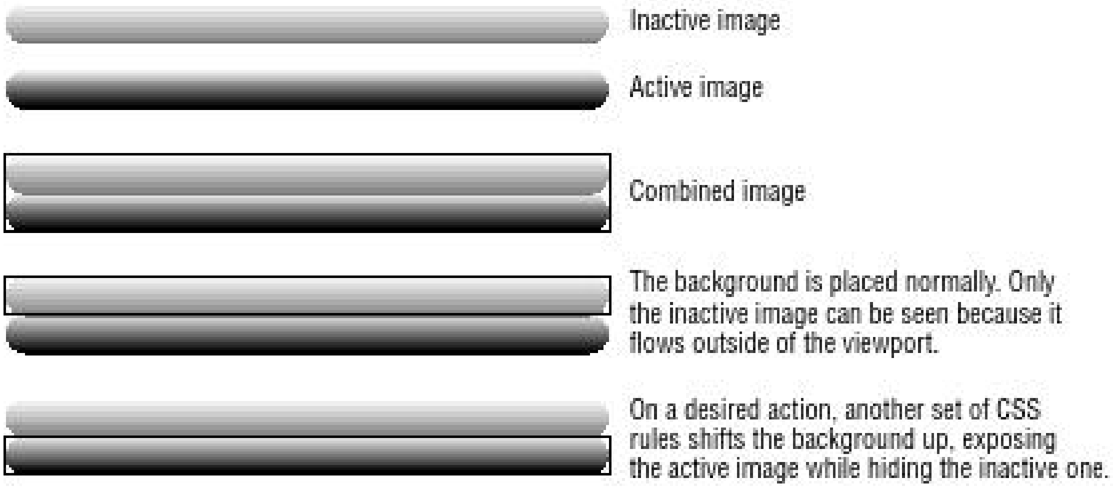


FIGURE 41-4

A popular technique to produce dynamic buttons is to place all the button graphics on a larger graphic and to use CSS positioning to expose the necessary graphic, as demonstrated in the figure below.


Inactive image
Active image
Combined image
The background is placed normally. Only the inactive image can be seen because it flows outside of the viewport.
On a desired action, another set of CSS rules shifts the background up, exposing the active image while hiding the inactive one.

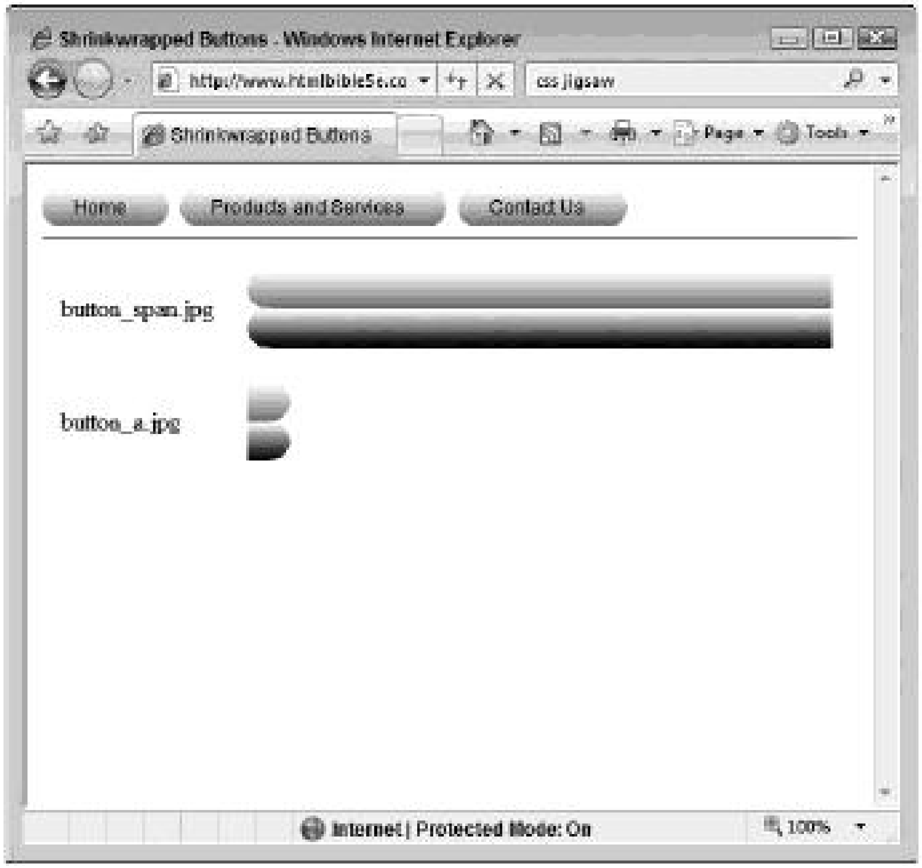
The following code illustrates the entire process by creating a handful of buttons, as shown in Figure 41-5. Note that the button graphics are displayed under the buttons to better illustrate how they look.

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

```
<html>
<head>
<title>Shrinkwrapped Buttons</title>
<style type="text/css">
  a.button {
    background: transparent url('images/button_a.jpg')
          no-repeat scroll top right;
    color: black;
    display: block;
    float: left;
    font: 12px arial, sans-serif;
    height: 25x;
    margin-right: 6px;
    padding-right: 28px;    /* padding for the right image, */
    text-decoration: none;  /* should match the image width */
}
  a.button span {
    background: transparent url('images/button_span.jpg') no-repeat;
    display: block;
    line-height: 14px;
    padding: 5px 0 5px 20px;
}
  a.button:active {
    background-position: bottom right;
    color: #aaaaaa;
    outline: none; /* hide outline displayed by some user agents */
}
  a.button:active span {
    padding: 6px 0 4px 20px; /* move text a bit for effect */
    background-position: bottom left;
}
</style>
</head>
<body>
<div>
<a class="button" href="#" onclick="this.blur();">
<span>Home</span></a>
<a class="button" href="#" onclick="this.blur();">
<span>Products and Services</span></a>
<a class="button" href="#" onclick="this.blur();">
<span>Contact Us</span></a>
</div>
<hr style="clear: both;">
<table border="0" cellpadding="10px">
<tr>
  <td>button_span.jpg</td>
  <td><img src="images/button_span.jpg" width="372" height="50" />
  </td>
</tr>
<tr>
  <td>button_a.jpg</td>
  <td><img src="images/button_a.jpg" width="28" height="50" /></td>
</tr>
</table>
</body>
</html>
```

FIGURE 41-5

The finished buttons with two visual states and shrink-wrap capability



In this particular case, the `:active` link pseudo-class is used to trigger the button transformation. Unfortunately, this state is sticky in Internet Explorer, requiring the bit of JavaScript with each link to instantly set it back to inactive when clicked.

Pull Quotes

Although more prevalent in magazine publishing, pull quotes are another publishing convention that is now popular on the Web. Pull quotes are generally an edited excerpt of an article placed in larger type and outside the article's text. They are meant to draw attention to the article by offering little tidbits of content.

Although you are likely familiar with pull quotes, an example of a pull quote might resemble the following:

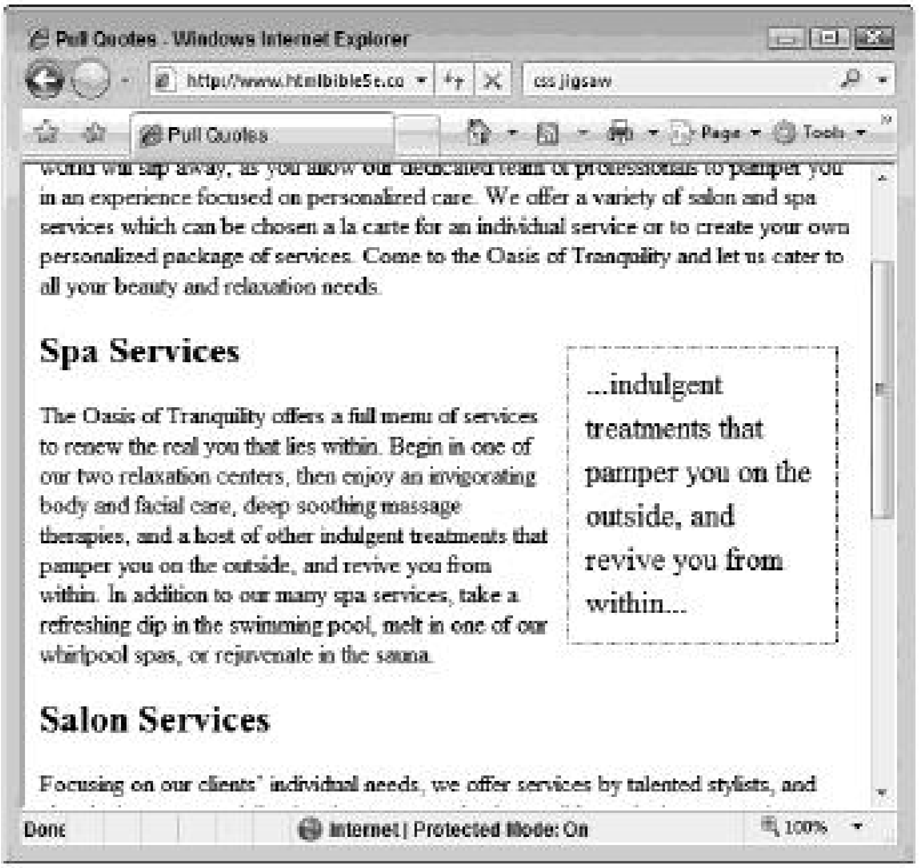
> " . . . his car and jacket were at the scene . . . "

Implementing a pull quote is simple: Place the appropriate text in a `div element`, make the text a bit larger than normal, and float the `div` to a margin. For example, consider the following code, whose results are shown in Figure 41-6:

```
 <!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Frameset//EN"
   "http://www.w3.org/TR/html4/frameset.dtd">
 <html>
 <head>
   <title>Pull Quotes</title>
     <style type="text/css">
         .pullquote {
             width: 150px;
             font-size:125%;
             line-height:140%;
             margin:10px;
             padding:10px;
             border: 1pt dotted black;
             float:right;
 }
     </style>
 <body>
 <div id="intro"><div><h2>Introducing the Oasis</h2>
 <p>The Oasis of Tranquility is a premier spa with an environment de-
signed to embrace you in an air of calming relaxation. When you walk
through our doors the outside world will slip away, as you allow our
dedicated team of professionals to pamper you in an experience
focused on personalized care. We offer a variety of salon and spa
services, which can be chosen a la carte for an individual service or
to create your own personalized package of services. Come to the
Oasis of Tranquility and let us cater to all your beauty and relaxa-
tion needs.</p></div></div>
 <div class="pullquote"><p>...indulgent treatments that pamper you on
the outside, and revive you from within...</p></div>
 <div id="spa"><h2>Spa Services</h2>
 <p>The Oasis of Tranquility offers a full menu of services to renew
the real you that lies within. Begin in one of our two relaxation
centers, then enjoy an invigorating body and facial care, deep sooth-
ing massage therapies, and a host of other indulgent treatments that
pamper you on the outside, and revive you from within. In addition to
our many spa services, take a refreshing dip in the swimming pool,
melt in one of our whirlpool spas, or rejuvenate in the sauna.</p>
</div>
 <div id="salon"><h2>Salon Services</h2>
 <p>Focusing on our clients' individual needs, we offer services by
talented stylists, and chemical treatments delivering the most stun-
ning hues, all in a relaxing spa environment. Focusing on our guests'
needs begins with an open dialogue between you and your stylist. This
communication allows our hair salon guests to get to know their styl-
ist, while educating the guest on how we will achieve and maintain
that specific look they desire. Photographs and descriptions of the
looks you desire are welcome. Our salon artists have the knowledge,
training, experience, and creativity to make any vision reality. The
same level of personalized care is given by our nail and skincare
technicians, in order to ensure the most relaxing and effective
treatments for each individual.</p></div>
 <div id="gift"><h2>Give the Gift of Tranquility</h2>
 <p>All services at the Oasis of Tranquility can be experienced indi-
vidually, or selected a la carte to create your own personalized day
of pampering.  Gift certificates are excellent for surprising your
loved ones with an hour or a day of pampering and rejuvenation.</p>
</div>
 <div id="summary"><h2>In Summary...</h2>
 <p>So when you are looking for an experience that will relax, reju-
venate, and free you from the weight and stress of everyday life and
leave you looking and feeling like the person you really are, come to
the Oasis of Tranquility.</p></div>
 </body>
 </html>
```

FIGURE 41-6

The pull quote block can be as simple or as ornate as you wish.


Pull Quotes - Windows Internet Explorer
http://www.htmlbible5e.co
css jigsaw
Pull Quotes
Page
Tools
Done
Internet | Protected Mode: On
100%

world will slip away, as you allow our dedicated team of professionals to pamper you in an experience focused on personalized care. We offer a variety of salon and spa services which can be chosen a la carte for an individual service or to create your own personalized package of services. Come to the Oasis of Tranquility and let us cater to all your beauty and relaxation needs.

Spa Services

...indulgent treatments that pamper you on the outside, and revive you from within...

The Oasis of Tranquility offers a full menu of services to renew the real you that lies within. Begin in one of our two relaxation centers, then enjoy an invigorating body and facial care, deep soothing massage therapies, and a host of other indulgent treatments that pamper you on the outside, and revive you from within. In addition to our many spa services, take a refreshing dip in the swimming pool, melt in one of our whirlpool spas, or rejuvenate in the sauna.

Salon Services

Focusing on our clients' individual needs, we offer services by talented stylists, and

Of course, you can use additional CSS rules to style the pull quote block to your liking — inserting backgrounds, images, creative font effects, and so on.

Tabbed Menus

One trend in menu design is to create menus in tabular form — that is, menu items that look like the tabs at the top of paper files. This method of creating menus is popular for a couple of reasons:

- The structure more closely resembles the menus found in most computer applications.
- The structure is created from components that don't lose their visual meaning if the ornate styling is not present.

The importance of the first reason should be apparent — after years of cramming Web navigation into every nook and cranny on the page, we can start to emulate the rest of the computer world by putting distinct blocks across the top of a page.

The second reason makes more sense when you consider what most tabular menus are constructed of: unnumbered list elements. Because menus (and submenus) are just lists at heart, using list elements in their construction makes a lot of sense; and, of course, if a list is broken down into its base components due to the absence of fancy styles, it retains its meaning and hierarchy.

Of course, a list is meant to be vertical in orientation, so how do you get it to display horizontally? Using the `display` property and setting it to `inline` effectively takes a block element and makes it an inline one.

Figure 41-7 shows an example of a simple menu created from list elements.

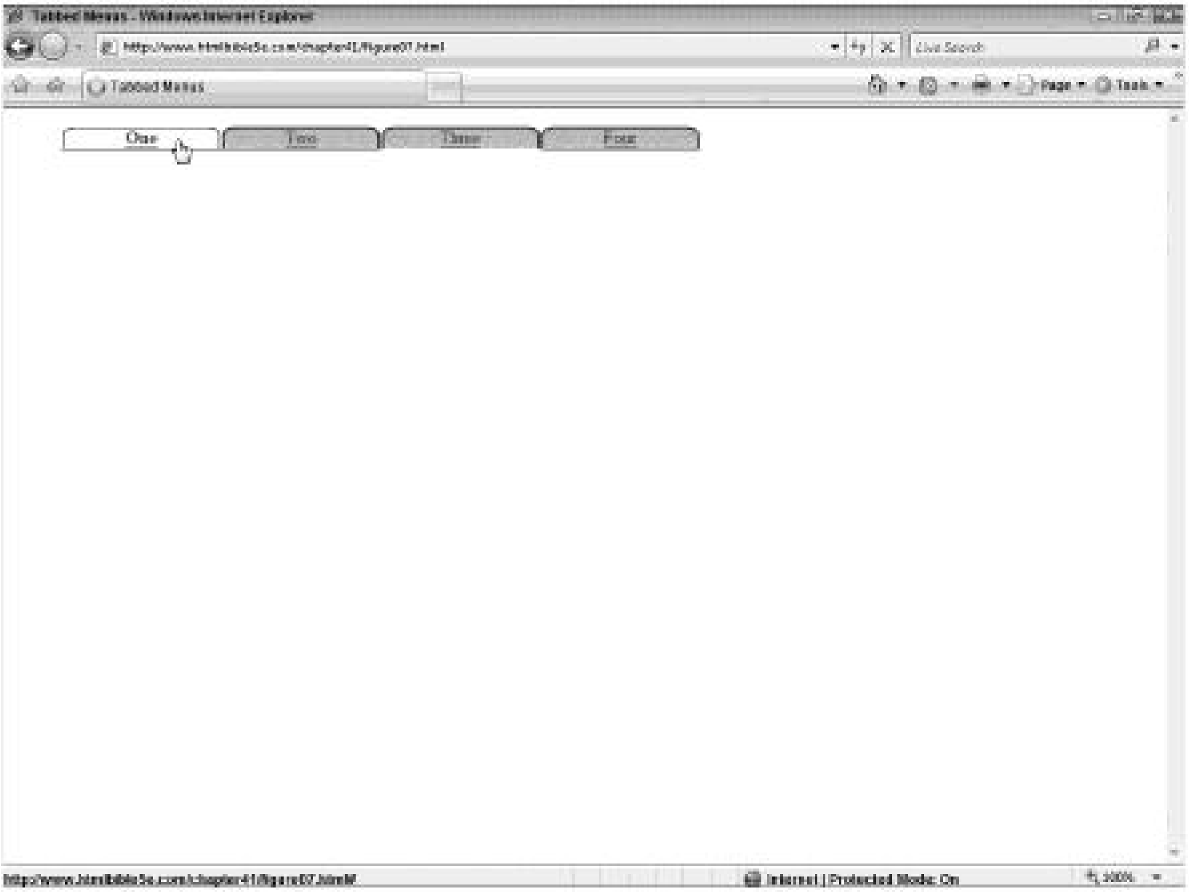
The code that generates this menu is as follows:

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Frameset//EN"
  "http://www.w3.org/TR/html4/frameset.dtd">
<html>
<head>
<title>Tabbed Menus</title>
<style type="text/css">
  li.menu   {
       display: inline;}
  li.menu a {
       background-image: url('images/tab.gif');
       width: 138px;
       text-align: center;
       border-bottom: 1pt black solid;
       float: left;}
  li.menu a:hover {
       background-image: url('images/tabhover.gif');
       font-weight: bold;}
  li.menu a.active {
       border-bottom: none;}
</style>
</head>
```

```
<body>
<p>
<ul>
  <li class="menu"><a href="#">One</a></li>
  <li class="menu"><a href="#">Two</a></li>
  <li class="menu"><a href="#" class="active">Three</a></li>
  <li class="menu"><a href="#">Four</a></li>
</ul>
</p>
</body>
</html>
```

FIGURE 41-7

List elements make excellent tabbed menus. Tab Three is active (note the line under the tab isn't visible) and tab One has changed color due to the mouse hovering over it.


Tabbed Menus - Windows Internet Explorer
http://www.htmlbible5e.com/chapter41/figure07.html
Live Search
Tabbed Menus
Page
Tools
One
Two
Three
Four
http://www.htmlbible5e.com/chapter41/figure07.html#
Internet | Protected Mode: On
100%

While the CSS is a bit complex, the HTML itself is very simple. The list elements handle the layout and some of the formatting, while their embedded anchors handle the actual navigation. A breakdown and explanation of the styles follow:

- All the `li` elements used in the menu have a class of `menu` to match the selectors that enable them.
- The first style simply sets each `li` element to be displayed inline so they display horizontally.

- The next style, selecting the anchor element under each list element, does most of the heavy lifting:
  - It sets a background image to that of a tab.
  - It sets the element's width to match the width of the background image.
  - It aligns the text to the center of the element.
  - It applies a border to the bottom of the element to give it the appearance of being behind the other elements.
  - Perhaps most important, it floats the element to the left to cause all the list elements to stack up against the element, or margin, to its left.
  - The hover style changes the tab's appearance when the mouse hovers over it. This is accomplished by making the font bold and changing the background image so the tab visually changes.
- The last style is applied only to a tab that has a class of `active`. This style removes the bottom border, making the tab appear as if it were on the top of the stack (hence, active).

Rounded Boxes

If there was ever a "brass ring" of CSS, it was elements with rounded corners. Text treatments of this type are fairly routine in the print world, but not in the boxy world of HTML.

Note

Several different techniques to achieve this result have been created over the years. The following code shows only one of the methods you can use. ■

To accomplish this effect, you create the four corners as images, placed using `div` elements, and fill the space in between with a matching background color. For example, consider the following code, and the annotated results shown in Figure 41-8:

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01//EN"
   "http://www.w3.org/TR/html4/strict.dtd">
<html>
<head>
<title>Rounded Corner Content</title>
<style type="text/css">
  /* Main content div */
  .rounddiv {
    width: 250px;
    background-color: #E22000;
    color: #FFFFFF;
  }
  /* Paragraphs in rounded div get
     a default margin */
  .rounddiv p {
    margin: 0 10px;
  }
  /* Div for top corners */
  .roundtop {
    background: url('images/tr.jpg') no-repeat top right;
  }
  /* Div for bottom corners */
  .roundbottom {
    background: url('images/br.jpg') no-repeat top right;
  }
  /* Default settings for rounded corner images */
  img.corner {
     width: 17px;
     height: 17px;
     border: none;
     display: block !important;
  }
</style>
</head>
<body>
<div class="rounddiv">
   <div class="roundtop">
   <img src="images/tl.jpg" alt=""
   width="17" height="17" class="corner"
   style="display: none" />
   </div>
   <p>Lorem ipsum dolor sit amet, consectetur adipisicing
   elit, sed do eiusmod tempor incididunt ut labore et
   dolore magna aliqua. Ut enim ad minim veniam, quis
   nostrud exercitation ullamco laboris nisi ut aliquip
   ex ea commodo consequat. Duis aute irure dolor in
   reprehenderit in voluptate velit esse cillum dolore eu
   fugiat nulla pariatur. Excepteur sint occaecat cupidatat
   non proident, sunt in culpa qui officia deserunt mollit
   anim id est laborum.</p>
   <div class="roundbottom">
   <img src="images/bl.jpg" alt=""
   width="17" height="17" class="corner"
   style="display: none" />
   </div>
</div>
</body>
</html>
```

Note

Although `div` elements provide the most flexible and intuitive element to achieve rounded corners, it is possible to construct alternatives using other XHTML elements. ■

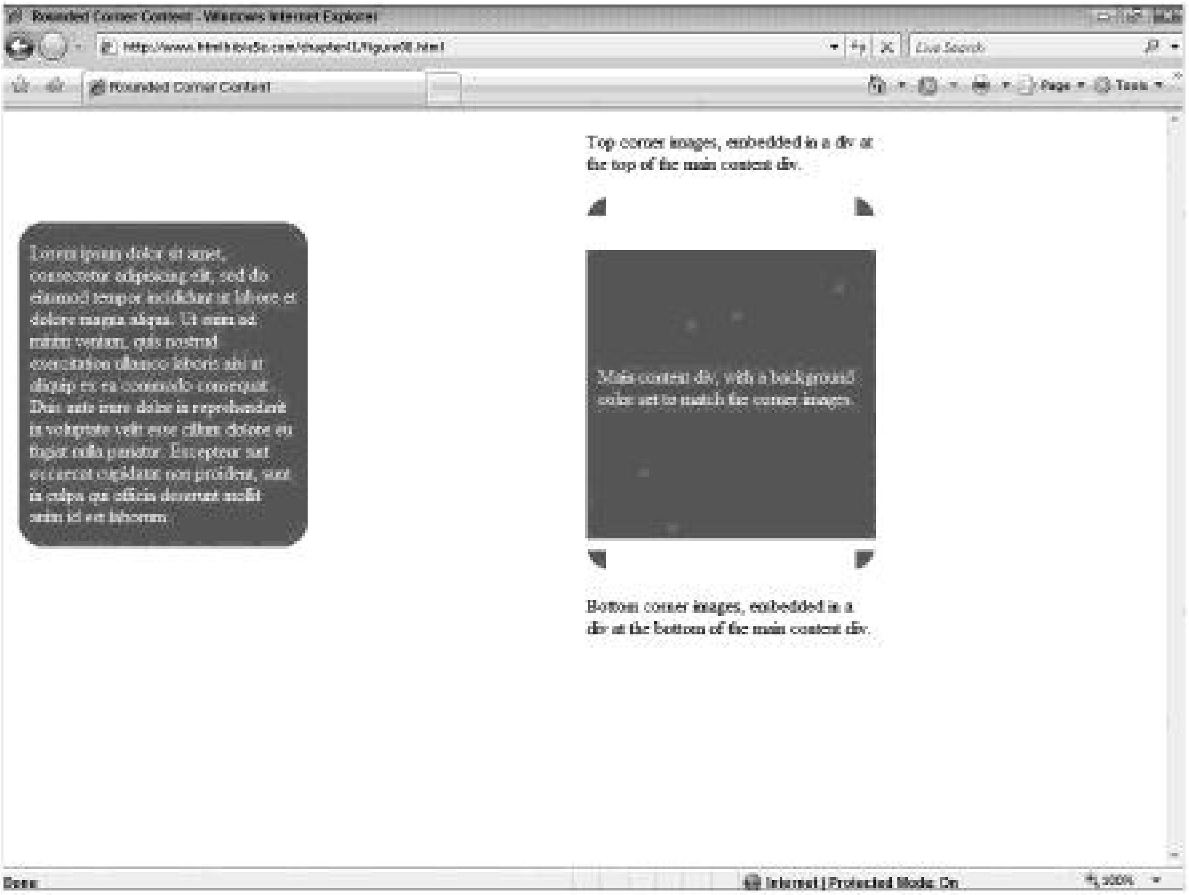
Note that the preceding code renders only the completed rounded-corner box shown on the left in Figure 41-8, not the exploded, annotated version on the right.

There are several caveats to this approach:

- The rounded image and the background of the `div` elements must match. You cannot use the same corner images with different color `div` elements.
- The rounded corners use a white background. While this is appropriate for documents that have a white background, it doesn't work well with documents that use a different color for their background, as shown in Figure 41-9.

FIGURE 41-8

The rounded-corner text box was once the brass ring of HTML formatting.


Rounded Corner Content - Windows Internet Explorer
http://www.htmlbible5e.com/chapter41/figure08.html
Live Search
Rounded Corner Content
Page
Tools
Done
Internet | Protected Mode: On
100%

Lorem ipsum dolor sit amet, consectetur adipisicing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat. Duis aute irure dolor in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur. Excepteur sint occaecat cupidatat non proident, sunt in culpa qui officia deserunt mollit anim id est laborum.

Top corner images, embedded in a div at the top of the main content div.

Main content div, with a background color set to match the corner images.

Bottom corner images, embedded in a div at the bottom of the main content div.

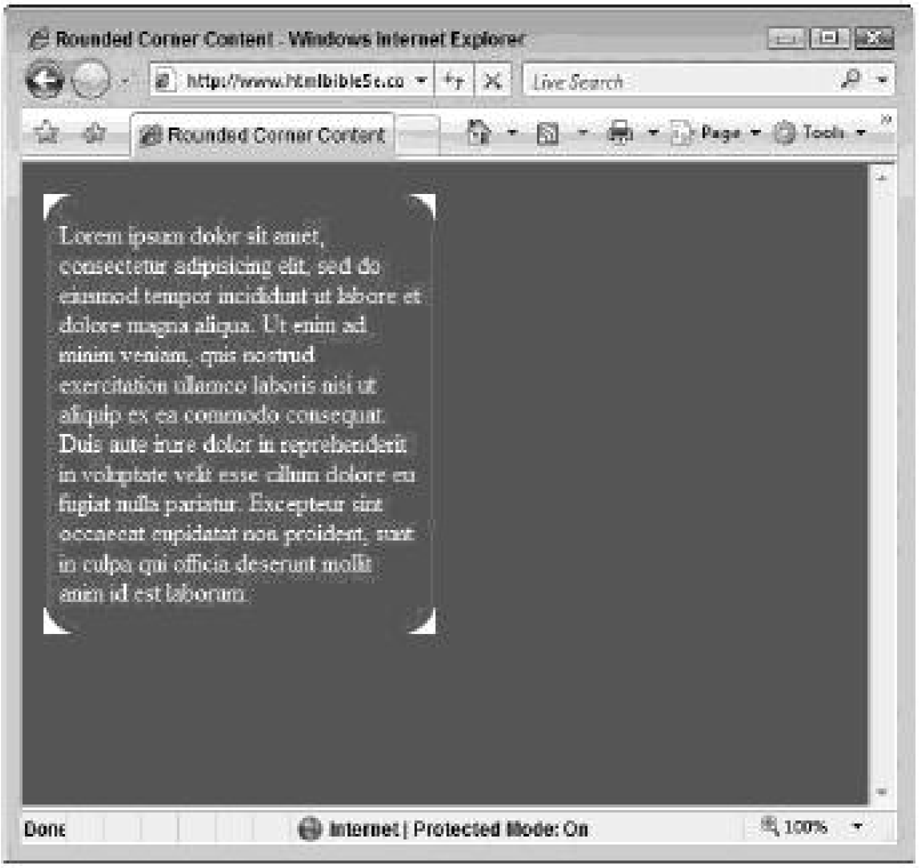
- The rounded corner element does not scale well — it is set for a fixed width of 20em, which is capable of handling most textual elements, but larger elements (such as tables) might cause issues within the confines of the rounded corner element.
- Most of these caveats have been overcome by other methods to achieve rounded corners. Of course, each method has its own set of caveats — the trick is to pick a method appropriate for your document.
- Currently, there are more methods to achieve rounded corners than can be reasonably counted. The following two sites provide many rounded corner methods:
  - Smileycat Web Design Blog — "CSS Rounded Corners 'Roundup'" at `www.smileycat.com/miaow/archives/000044.php`
  - CSS Juice — "25 Rounded Corners Techniques with CSS" at `www.cssjuice.com/25-rounded-corners-techniques-with-css/`

Tip

In the event that none of the techniques listed on these sites meet your requirements, use your favorite search engine to find more "CSS rounded corner" solutions. ■

FIGURE 41-9

Different background colors give the rounded corner trick away.


Rounded Corner Content - Windows Internet Explorer
http://www.htmlbible5e.co
Live Search
Rounded Corner Content
Page
Tools
Done
Internet | Protected Mode: On
100%

Lorem ipsum dolor sit amet, consectetur adipisicing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat. Duis aute irure dolor in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur. Excepteur sint occaecat cupidatat non proident, sunt in culpa qui officia deserunt mollit anim id est laborum.

Flowing Elements

Some document designs can benefit from allowing certain elements on the page to "float." For example, compare Figure 41-10 and Figure 41-11. They show the same document in the same user agent, but the user agent's window has been narrowed in Figure 41-11.

Notice how the small boxes flow into rows that fit the document's width. As the document narrows, fewer elements can fit on each line. The last element(s) on the line are forced to the next line, accordingly. If the document widens, allowing more elements to fit on each line, element(s) from the next line move up a line to fill the gap. These types of document designs are used primarily in catalog or item lookup directories where the floating `div` size should remain constant but flow according to the document's width.

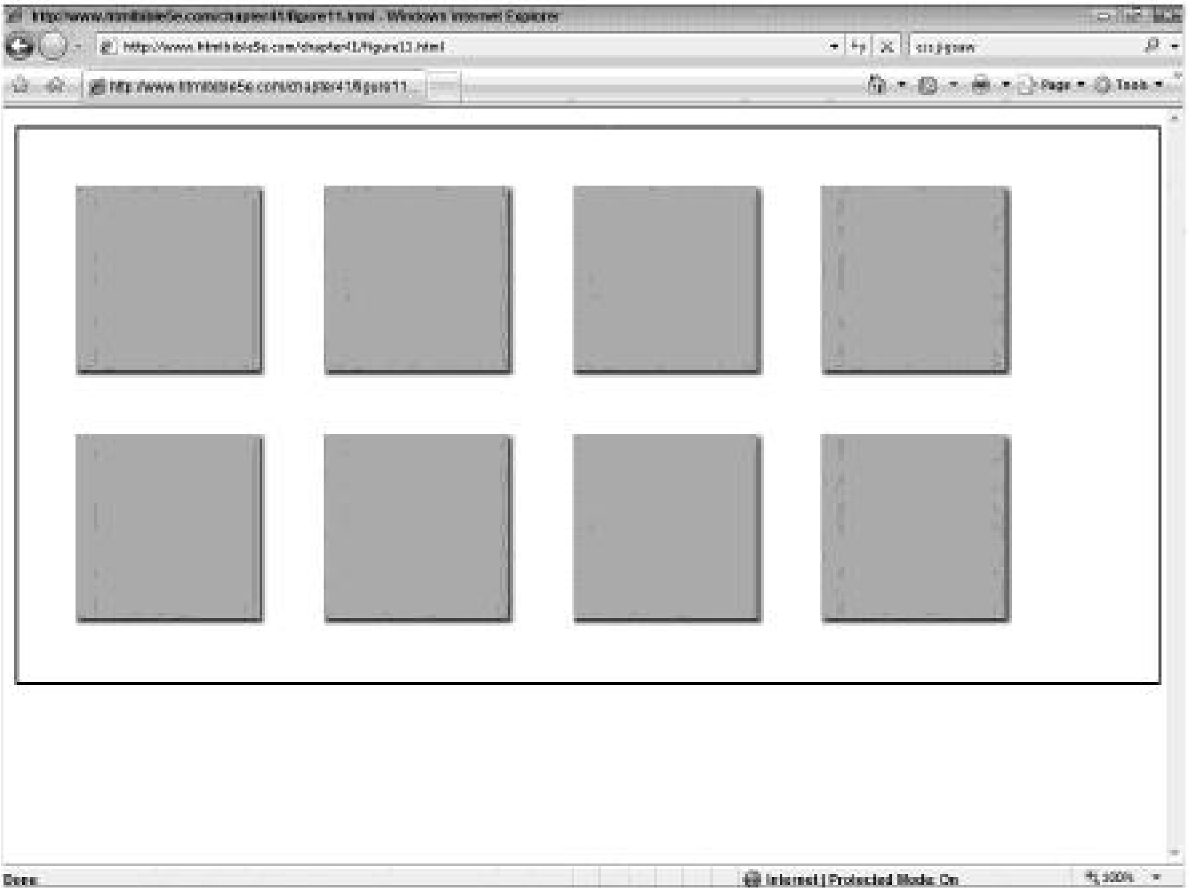
The design is remarkably easy to achieve. The general steps are listed here, followed by example code:

1. Create a container `div` element that is roughly the width of the user agent screen (width: 95%).
2. Set the top and right padding of the container to a suitable value — one that will provide the appropriate amount of space between the top and right edges of the container

and the interior `div` elements. For the best results, set both padding values to the same amount.

FIGURE 41-10

A directory or catalog of items can be represented by individual cells (*div* elements).


http://www.htmlbible5e.com/chapter41/figure11.html - Windows Internet Explorer
http://www.htmlbible5e.com/chapter41/figure11.html
css jigsaw
http://www.htmlbible5e.com/chapter41/figure11...
Page
Tools
Done
Internet | Protected Mode: On
100%

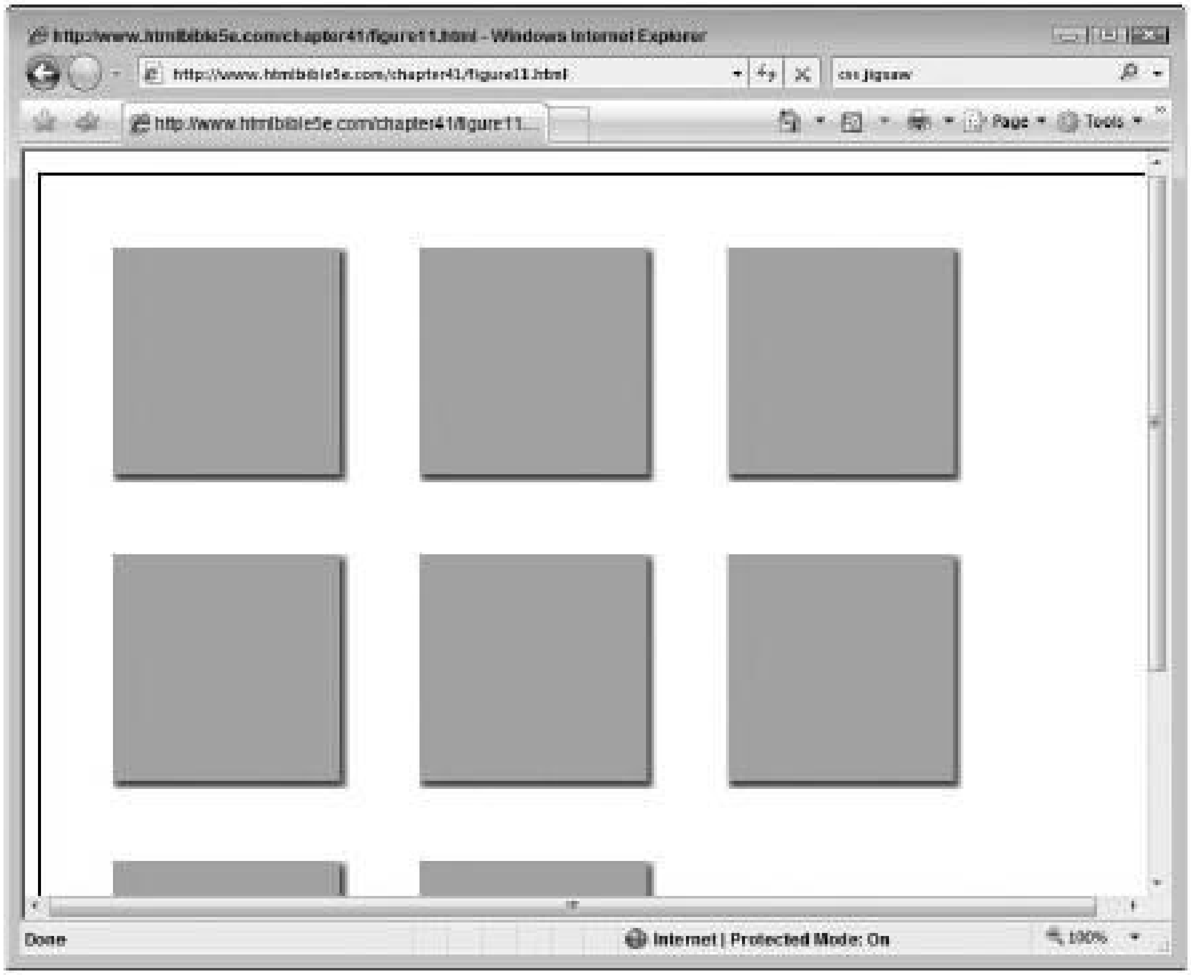
3. Place the required number of `div` elements (floaters) inside the container element.
4. Style the floating `div` elements any way you like (e.g., the drop shadow effect in Figure 41-11).
5. Set an explicit width and height for each floating element.
6. Set the right and bottom margins of the floating elements to the same value you used for the container's padding (step 2). The container's padding provides the space between the top row and rightmost column of floating elements. The floating element margins provide the space between floating element columns and rows.
7. Set all elements (container and float) to float to the left.

Note

All of the preceding formatting should be accomplished only via appropriate CSS styles. ■

FIGURE 41-11

If the container and elements are styled correctly, the elements will flow within the container and change their flow if the container changes shape.


http://www.htmlbible5e.com/chapter41/figure11.html - Windows Internet Explorer
http://www.htmlbible5e.com/chapter41/figure11.html
css jigsaw
http://www.htmlbible5e.com/chapter41/figure11...
Page
Tools
Done
Internet | Protected Mode: On
100%

```
<!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>
<style type="text/css">
div.container {
    width: 90%;
    padding: 50px;
    float: left;
    border: 2px solid black;
    }

div.floating {
    background-image: url("images/thumbback.jpg");
    background-position: center;
    background-repeat: no-repeat;
    width: 165px;
    height: 165px;
```

```
        margin-right: 50px;
        margin-bottom: 50px;
        float: left;
        }

</style>

<body>

<div class="container">

<div class="floating">&nbsp;</div>
<div class="floating">&nbsp;</div>
<div class="floating">&nbsp;</div>
<div class="floating">&nbsp;</div>
<div class="floating">&nbsp;</div>
<div class="floating">&nbsp;</div>
<div class="floating">&nbsp;</div>
<div class="floating">&nbsp;</div>

</div>

</body>
</html>
```

Flowing Text

Another often sought after CSS formatting effect is the ability to flow text seamlessly around other elements. For example, Figure 41-12 shows text that flows around the curved image in the document.

This effect is achieved by placing several spacing `span` elements along the curve, forcing the text to flow accordingly. This is best illustrated by turning on the borders of the spacing `span` elements, as shown in Figure 41-13.

The following code can be used to accomplish the formatting shown in Figure 41-13:

```
<!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>
<style type="text/css">

   #wrapper {
            text-align: left;
            width: 600px;
            margin: 30px auto;
            border: 1px solid blue;
            padding: 15px 0px 5px 15px;
            background: url('images/circleRight.gif') no-repeat;
            background-position: right top;
            }
   .spacer {
           float: right;
           display: block;
           height: 15px;
           clear: right;
           margin-left: 10px;
           }

    #vspacer {
             width: 1px;
             height: 15px
             }


    #spacer01 { width: 70px; }
    #spacer02 { width: 85px; }
    #spacer03 { width: 100px; }
    #spacer04 { width: 115px; }
    #spacer05 { width: 130px; }
    #spacer06 { width: 140px; }
    #spacer07 { width: 140px; }
    #spacer08 { width: 130px; }
    #spacer09 { width: 115px; }
    #spacer10 { width: 100px; }
    #spacer11 { width: 85px; }
    #spacer12 { width: 70px; }
    #spacer13 { width: 60px; }
    #spacer14 { width: 60px; }

</style>

<body>

<div id="wrapper">
  <span id="spacer01" class="spacer"></span>
  <span id="spacer02" class="spacer"></span>
  <span id="spacer03" class="spacer"></span>
  <span id="spacer04" class="spacer"></span>
  <span id="spacer05" class="spacer"></span>
  <span id="spacer06" class="spacer"></span>
  <span id="spacer07" class="spacer"></span>
  <span id="spacer08" class="spacer"></span>
  <span id="spacer09" class="spacer"></span>
  <span id="spacer10" class="spacer"></span>
  <span id="spacer11" class="spacer"></span>
  <span id="spacer12" class="spacer"></span>
  <span id="spacer13" class="spacer"></span>
  <span id="spacer13" class="spacer"></span>
  <p>Lorem ipsum dolor sit amet, consectetuer adipiscing elit.
Nam venenatis  facilisis risus. Vestibulum lacus neque, scelerisque
ut, gravida sit amet, laoreet sed, sapien. Nunc tincidunt convallis
mauris. Proin  aliquam tristique ipsum. Pellentesque libero orci,
```