Chapter 5. Cascading Style Sheets

- 5.1 Specifying Style Rules
- 5.2 Using External and Local Style Sheets
- 5.3 Selectors
- 5.4 Cascading: Style Sheet Precedence Rules
- 5.5 Font Properties
- 5.6 Foreground and Background Properties
- 5.7 Text Properties
- 5.8 Properties of the Bounding Box
- 5.9 Images and Floating Elements
- 5.10 List Properties
- 5.11 Standard Property Units
- 5.12 Layers
- 5.13 Summary

Topics in This Chapter

- Defining style sheet rules
- Associating style rules with an HTML document
- Determining which elements support style rules
- Understanding the precedence rules of conflicting styles
- Controlling font characteristics
- Applying color and image characteristics
- Formatting text with styles
- Controlling the bounding box around HTML elements
- Adding image and floating text properties
- Improving lists through styles
- Using size units and color properties
- Adding multiple layers to a document

Cascading style sheets are a powerful and flexible way of specifying formatting information for Web pages. They let you define the font, size, background and foreground color, background image, margin, and other characteristics for each of the standard HTML elements. In addition to specifying how standard elements should be displayed, style sheets let you define your own classes, effectively letting authors define new HTML elements, albeit with some constraints on the characteristics these new elements can have. Formatting rules are applied in a hierarchical or "cascading" manner, letting default rules from the browser combine with explicit rules from both the reader and the author. Style sheets can be loaded from external sites, permitting sharing of style sheets and letting authors change the look and feel of entire Web sites by changing only a single file. The current standard for style sheets is "Cascading Style Sheets, Level 1," known as "CSS1." Style sheets are supported by Netscape and Internet Explorer, versions 4.0x and above. Internet Explorer, version 3.0, supports most, but not all, of CSS1.

In addition to CSS1, the World Wide Web Consortium introduced CSS2, which adds support for media-specific style sheets so that authors can tailor the presentation of their documents to visual browsers, aural devices, printers, braille devices, and handheld devices. In addition, CSS2 supports content positioning, downloadable fonts, table layout, features for internationalization, automatic counters, and numbering. Extensive work is underway to extend style sheets in numerous areas. See the following links for additional information on the upcoming standards.

Cascading Style Sheets, Level 1

http://www.w3.org/TR/REC-CSS1

Cascading Style Sheets, Level 2

http://www.w3.org/TR/REC-CSS2

Extensible Stylesheet Language

http://www.w3.org/Style/XSL/

Style Sheet News and Proposed Extensions

http://www.w3.org/Style/

Only Cascading Style Sheets, Level 1 are covered in this chapter.

5.1 Specifying Style Rules

HTML elements are customized by the use of *style rules.* The most common practice is to place the style rules in a separate text file and refer to the text file through a `LINK` element located in the `HEAD` section of the document. However, style rules can also be placed directly in the `HEAD` section of an HTML document or in the body of the document. Style rules are of the form

```
selector { property: value }
```

or

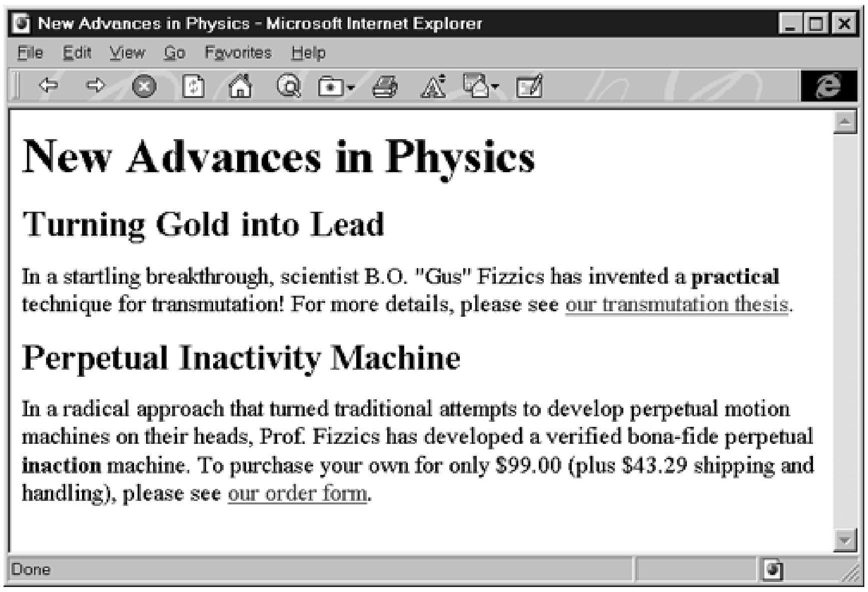
```
selector { property1: value1;
           property2: value2;
           ...
           propertyN: valueN }
```

Note that multiple properties for a single selector should end in a semicolon; the last property of a selector does not require a semicolon.

The types of selectors that can be used are discussed in Section 5.3, but the most basic type is simply the name of an HTML element, signifying that the properties listed inside the braces should apply to all occurrences of that element in the document. The available properties and their possible values are described in Sections 5.5 through 5.10. For instance, consider the small HTML document shown in Listing 5.1.

When displayed in Internet Explorer, the result looks like Figure 5-1.

Figure 5-1. `Fizzics1.html` without style information.


New Advances in Physics - Microsoft Internet Explorer
File Edit View Go Favorites Help
Done

New Advances in Physics

Turning Gold into Lead

In a startling breakthrough, scientist B.O. "Gus" Fizzics has invented a **practical** technique for transmutation! For more details, please see our transmutation thesis.

Perpetual Inactivity Machine

In a radical approach that turned traditional attempts to develop perpetual motion machines on their heads, Prof. Fizzics has developed a verified bona-fide perpetual **inaction** machine. To purchase your own for only $99.00 (plus $43.29 shipping and handling), please see our order form.

Listing 5.1 `Fizzics1.html`

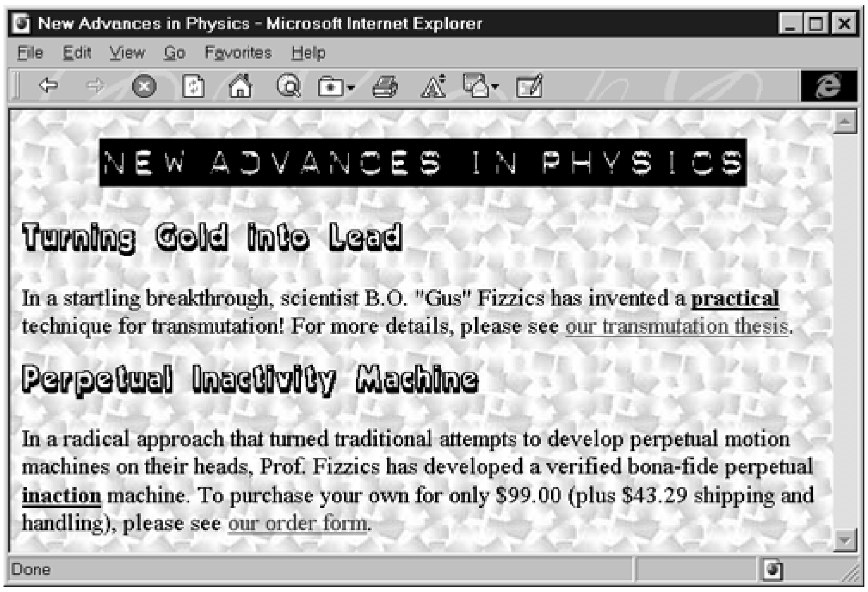
```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0 Transitional//EN">
<HTML>
<HEAD>
  <TITLE>New Advances in Physics</TITLE>
</HEAD>
<BODY>
<H1>New&nbsp;Advances&nbsp;in&nbsp;Physics</H1>

<H2>Turning Gold into Lead</H2>
In a startling breakthrough, scientist B.O. "Gus" Fizzics
has invented a <STRONG>practical</STRONG> technique for
transmutation! For more details, please see
<A HREF="give-us-your-gold.html">our transmutation thesis</A>.

<H2>Perpetual Inactivity Machine</H2>
In a radical approach that turned traditional attempts to
develop perpetual motion machines on their heads, Prof.
Fizzics has developed a verified, bona-fide perpetual
<STRONG>inaction</STRONG> machine. To purchase your own for
only $99.00 (plus $43.29 shipping and handling), please see
<A HREF="rock.html">our order form</A>.
</BODY>
</HTML>
```

However, when the style information shown in the following snippet is added to the document `HEAD`, the result changes to that of Figure 5-2. Note the use of HTML comments to prevent non-CSS-capable browsers from seeing the text between `<STYLE>` and `</STYLE>`.

Figure 5-2. `Fizzics1.html` after style information is added.


New Advances in Physics - Microsoft Internet Explorer
File Edit View Go Favorites Help
Done

NEW ADVANCES IN PHYSICS

Turning Gold into Lead

In a startling breakthrough, scientist B.O. "Gus" Fizzics has invented a **practical** technique for transmutation! For more details, please see our transmutation thesis.

Perpetual Inactivity Machine

In a radical approach that turned traditional attempts to develop perpetual motion machines on their heads, Prof. Fizzics has developed a verified bona-fide perpetual **inaction** machine. To purchase your own for only $99.00 (plus $43.29 shipping and handling), please see our order form.

```
<STYLE TYPE="text/css">
<!--
BODY { background: URL(images/confetti-background.jpg) }
H1 { text-align: center;
     font-family: Blackout }
H2 { font-family: MeppDisplayShadow }
STRONG { text-decoration: underline }
-->
</STYLE>
```

5.2 Using External and Local Style Sheets

Style rules are most commonly placed in a separate text file on the Web site and referred to through a link in the HTML document. By creating an external file, you greatly simplify the application of the same set of style rules to multiple HTML files on the Web site. Alternatively, you can place the style rules locally in the document HEAD or directly in the body of the document.

External Style Sheets

If you plan to use the same formatting styles in several documents, you can create a separate style sheet in a WWW-accessible file. You can then link this file to HTML documents by using a LINK element in the document HEAD with a REL value of STYLESHEET, an HREF giving the location, and a TYPE of text/css. For instance, if the external style sheet is called Site-style.css and is located in the css directory, the file could be associated with the current document by a LINK element that looked like:

```
<LINK REL=STYLESHEET
      HREF="http://www.oursite.com/css/Site-style.css"
      TYPE="text/css">
```

The style sheet file should include the style rules for each selector, as shown in Listing 5.2. Including the STYLE tags in linked external file is optional—the style sheet is implied through the TYPE attribute in the LINK element. Many authors still include the STYLE tags and comment deliminators in the external style sheet, even though doing so is not required.

Core Approach

Simplify maintenance of your Web site and HTML files by placing your style sheets in external text files.

Listing 5.2 `Sitestyle.css`

```
/* Example of an external style sheet */

H1 { text-align: center;
     font-family: Arial
}
H2 { color: #440000;
     text-align: center;
     font-family: Arial Black, Arial, Helvetica, sans-serif
}

...
```

Styles rules placed in the HEAD section of the HTML document must be enclosed in the STYLE element.

HTML Element:	**`<STYLE TYPE="..." ...> ... </STYLE>`**
Attributes:	TYPE (required), MEDIA

The STYLE element defines a container for style sheet rules and can only appear within the HEAD section of a document. The standard syntax is:

```
<STYLE TYPE="text/css">
<!--
/* optional comment */
Style Rules
-->
</STYLE>
```

Typically, the style rules are enclosed in HTML comments to accommodate older browsers that do not support style sheets and would normally see the rules in between the `<STYLE>` and `</STYLE>` tags.

Core Approach

Enclose your STYLE rules in HTML comments.

TYPE The TYPE attribute is required and defines the format of the style sheet rules. For a cascading style sheet the type is `"text/css"`. The type for a JavaScript style sheet is `"text/javascript"`.

MEDIA The MEDIA attribute is currently unsupported by Netscape and Internet Explorer but is intended to indicate the required media device before the rules are applied, the idea being that a document could have multiple style sheets, depend on a particular media type. Legal values include ALL, AURAL, BRAILLE, HANDHELD, HELD, PRINT, PROJECTION, SCREEN (default), SPEECH, TTY, and TV.

The STYLE Element and JavaScript Style Sheets

In addition to a style TYPE of `"text/css"`, Netscape 4.x also supports a value of `"text/javascript"`. JavaScript style sheets use the `tags` object, use normal JavaScript syntax, and change property names of the form `property-name` to `propertyName`. For instance,

```
<STYLE TYPE="text/css">
<!--
```

```
H1 { text-align: center;
     font-family: Arial }
-->
</STYLE>
```

is equivalent to

```
<STYLE TYPE="text/javascript">
<!--
tags.H1.textAlign="center";
tags.H1.fontFamily="Arial";
//-->
</STYLE>
```

The standard syntax is more portable, but for Netscape-specific applications, JavaScript allows you to *calculate* values, rather than just specify them statically. JavaScript is discussed in more detail in Chapter 24.

Inline Style Specification

The CSS1 specification allows you to specify formatting information directly in an HTML element by use of the STYLE *attribute* added to all elements in the HTML 4.0 specification. Inline rules look just like normal rules except that the braces and HTML element name are omitted. For instance:

```
<H1>New Advances in Physics</H1>
<P STYLE="margin-left: 0.5in;
          margin-right: 0.5in;
          font-style: italic">
This paper gives the solution to three
previously unsolved problems: turning lead into gold,
antigravity, and a practical perpetual motion machine.
```

Inline rules override separate styles declared in the HEAD section. Separate style rules are easier to extend and maintain than inline styles and should generally be used when you are defining complex rules.

5.3 Selectors

You usually define style rules by placing an entry of the following form in the STYLE element:

```
selector { property1: value1; ... ; propertyN: valueN }
```

Up to this point, we have only given examples where the selector was an HTML element name, indicating that the rule should be applied to all elements of that type unless overridden in the element declaration itself. For instance, to specify that strongly emphasized text be rendered in bold with a 50% increased font size, you could use:

```
STRONG { font-weight: bold; font-size: 150% }
```

Although HTML elements are a common selector type, they are not the only option. The cascading style sheet standard allows a variety of selector types to define formatting rules that apply only in certain situations. The four most common categories of selectors are:

- HTML elements
- User-defined classes
- User-defined IDs
- Anchor pseudoclasses

To support style rules that apply only in certain conditions, use the attributes CLASS, ID, and STYLE, which are allowed for *all* HTML elements except for BASE, BASEFONT (ID allowed), HEAD, HTML, META, PARAM (ID allowed), SCRIPT, STYLE, and TITLE. In addition, the HTML 4.0 specification supports a SPAN element to apply a style to any arbitrary section of a document.

HTML Elements

Any HTML element can be used as a selector, although BR has no associated text, so rules would be meaningless. Property settings are inherited, for example, background colors for BODY apply to paragraphs within the body by default, font sizes for P apply to CODE sections within the paragraph unless overridden, and so forth. For instance, to make blue the default color for all text except first-level headings, with top-level headings in red, you could use:

```
BODY { color: blue }
H1 { color: red }
```

Elements can be grouped in comma-separated lists to allow common styles to be set for multiple HTML elements. For instance, you could use

```
H1, H2, H3, H4, H5, H6 { text-align: center;
                         font-family: sans-serif }
```

rather than setting each of H1 through H6 separately.

Most property settings are inherited. Elements contained within stylized elements receive the styles of the outer elements. Consider the following styles:

```
BODY { color: blue }
H1 { color: red }
EM { color: red }
```

In this example, setting the style of color: blue at the BODY element is also inherited by all elements in the HTML document, for example, P, UL, OL, and TABLE elements; basically, all text in the document will be blue, except for any overriding styles, as is the case with the H1 and EM style declarations.

One problem with the preceding example is that emphasized text *inside* a level-one heading would not be distinguishable from the rest of the heading. So, you could add a rule specifying that emphasized text be green *only* when inside a main heading, as follows:

```
H1 EM { color: green }
```

Contexts can be arbitrarily nested. The CSS specification clarifies which styles are inherited by child containers. However, be advised that Netscape 4.x suffers from numerous style sheet bugs. As a result, styles are not inherited by containers as expected. A workaround is to explicitly state the style for every nested container or to apply a user-defined class to problem containers.

Core Warning

In Netscape 4.x, not all styles are properly inherited by child containers, so thoroughly test your HTML document before posting on a Web site.

User-Defined Classes

You can also define your own classes of selectors, separated by a period from the associated HTML element. For instance, to define an "abstract" paragraph type, <P>, with indented left and right margins and italic text, you could use the following:

```
P.abstract { margin-left: 0.5in;
             margin-right: 0.5in;
             font-style: italic }
```

To use this class, you would supply the name of the class inside the `CLASS` attribute of the HTML element in the body of the document. For example, given the preceding `abstract` class, you could use the defined style as follows:

```
<H1>New Advances in Physics</H1>
<P CLASS="abstract">
This paper gives the solution to three previously unsolved
problems: turning lead into gold, antigravity, and a
practical perpetual motion machine.
```

You can also define classes that apply to any HTML element by omitting the HTML element that normally precedes the class name in the definition. For instance, the following defines a class that sets the foreground color to blue and uses a bold font:

```
.blue { color: blue; font-weight: bold }
```

This style could be used in an existing paragraph:

```
This text is in the default color, but
<SPAN CLASS="blue">this text is blue.</SPAN>
```

or could be applied to an entire block:

```
<H2 CLASS="blue">A Blue Heading</H2>
```

Be aware that Netscape does not recognize user-defined class names that contain an underscore. For example, Netscape would not recognize `.blue_font` as a style.

Core Warning

Class names that contain an underscore (_) are not recognized by Netscape.

User-Defined IDs

An `ID` is like a class but can be applied only once in a document. You define an `ID` by preceding the name with a `#`, then you reference the definition with the `ID` attribute, as follows:

```
<HEAD>
<TITLE>...</TITLE>
<STYLE TYPE="text/css">
<!--
#foo { color: red }
-->
</STYLE>
</HEAD>
<BODY>
...
<P id="foo">
...
</BODY>
```

In most cases, classes are better choices than `ID`s.

Anchor Pseudoclasses

Although HTML has a single element to indicate a hypertext link (the anchor element `A`), browsers typically treat links in one of four different ways, depending on whether the links are new, visited, or active. The CSS1 standard lets you specify separate properties for each link type. To indicate the type of link, use one of the following selectors:

A:link or :link This selector matches anchor elements only if they have *not* been visited, as determined from the browser's history log.

A:visited or :visited This selector matches anchor elements only if they *have* been visited, as determined from the browser's history log.

A:active or :active This selector indicates how links should be displayed as the user clicks on them (but before releasing the mouse).

A:hover or :hover This selector, supported only in Internet Explorer, applies when the mouse is over the link.

These pseudoclasses can be combined with other selectors. For instance, in the following, the first style rule applies only to visited links inside text sections in a particular class, and the second style rule applies only to images that are inside hypertext links that have not been visited.

```
.bizarre :active { font-size: 300% }
A:link IMG { border: solid green }
```

5.4 Cascading: Style Sheet Precedence Rules

Multiple style rules may apply to a particular section of text; thus, the browser needs to know the order in which to apply the rules. The highest-precedence rules are applied last so that they replace conflicting values from lower-priority rules. The rules for determining the precedence (or "cascading") order are as outlined below.

1. **Rules marked "important" have the highest priority.**

   A style rule can have the tag `!important` appended. For instance, in the following example, the foreground color property is marked as important.

   ```
   H1 { color: black !important;
        font-family: sans-serif }
   ```

   These declarations are normally used sparingly, if at all.

2. **Author's rules have precedence over reader's rules.**

   Browsers may permit readers to create style sheets to override the system defaults. In such a case, explicit settings by the Web page author have higher priority over the browser settings.

3. **More specific rules have precedence over less specific rules.**

   In specificity determination, `ID` attributes in the selector have the highest priority. Ties based on ID selectors are broken by counting the number of class attributes in the selector. Finally, if a rule is still tied, the number of HTML element (tag) names determines specificity. For instance, the following rules are sorted in order of specificity. The first is most specific because it has an `ID` selector. The other three are tied according to this measure, but the second rule has a class selector (`big`) and so has precedence over rules three and four, which don't. Finally, the third style rule is more specific than the fourth style rule because of the presence of two tags rather than one.

```
#foo { ... }
P.big H1 { ... }
P STRONG { ... }
STRONG { ... }
```

4. **In case of a tie, the last rule specified has priority.**

   If two or more rules have the same priority after the previous three rules are applied, then later rules are given priority over earlier rules.

5.5 Font Properties

CSS1 gives the author control over several aspects of the font: whether the font is bold, italic, or normal, what size to use, what font families are preferred, and whether a small-cap variation should be used. Netscape also supports *dynamic fonts,* which let you attach a font definition file to a document rather than depending on certain fonts already being on the client system. Underlined text, subscripts, and superscripts are not set by these font properties but rather by the `text-decoration` and `vertical-align` properties, covered in Section 5.7 (Text Properties). In Sections 5.5 through 5.10, the style values are listed immediately after the style declaration, with the default value indicated in bold.

font-weight

normal | lighter | bold | bolder | 100 | 200 | … | 900

This style specifies the weight of the font and has legal values of 100 (lightest) through 900 (heaviest) in units of 100, plus the relative values `normal`, `lighter`, `bold`, and `bolder`. For instance,

```
H1 { font-weight : 200 }
H2 { font-weight : bolder }
```

Netscape does not support the `bolder` font weight.

font-style

normal | italic | oblique

This property selects the font face type within a family. Legal values are `normal`, `italic`, and `oblique`.

```
P  { font-style : normal }
TH { font-sytle : italic }
```

Netscape does not support `oblique`.

font-size

pt, pc, in, cm, mm | em, ex, px, % |

xx-large | x-large | large | **medium** | small | x-small | xx-small |

smaller | larger

This style specifies the font size. A value can be in standard length units (see Table 5.1 in Section 5.11), a symbolic value, or a percentage. Symbolic values can be absolute (`xx-large`, `x-large`, `large`, `medium`, `small`, `x-small`, and `xx-small`) or relative (`smaller` or `larger`). A percentage is interpreted with respect to the font size of the container. For instance,

```
STRONG { font-size: 150% }
```

should mean that text in a STRONG element should be 50% larger than the current font size. Additional examples:

```
P { font-size: 14pt }
P { font-size: 1cm }
P { font-size: xx-large }
```

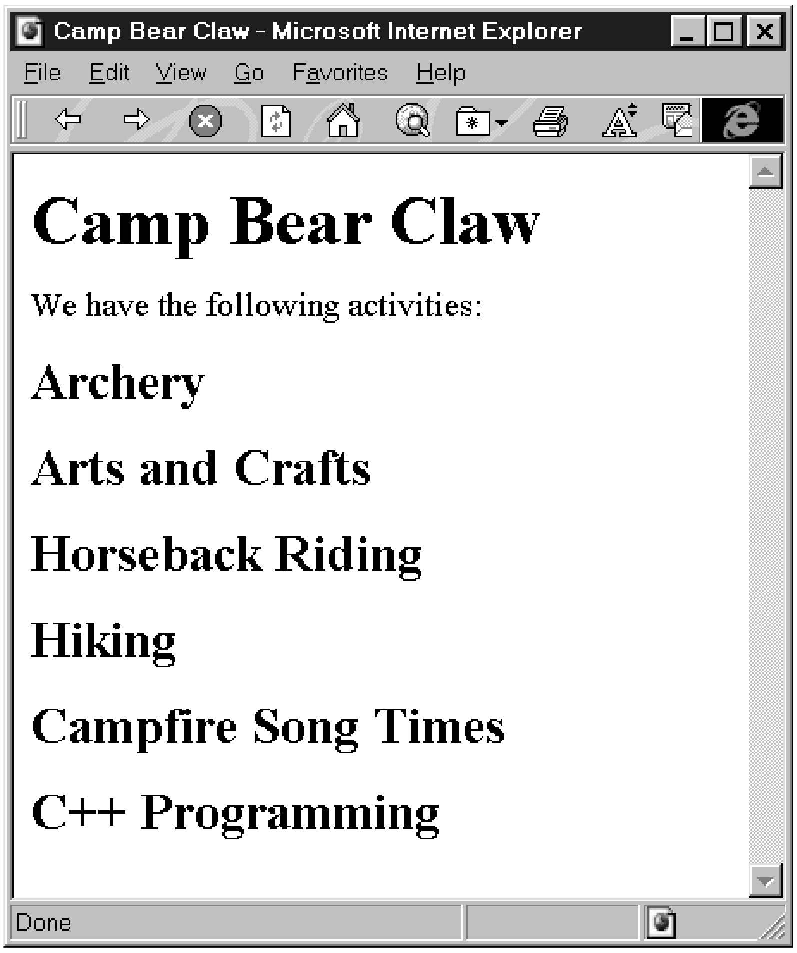
Note that Netscape will not recognize the style if you place a space between the numerical declaration and unit declaration, as in "14 pt" versus "14pt".

font-family

family name

This style specifies the typeface. For instance, Listing 5.3 shows a rather dry page describing "Camp Bear Claw," with the standard look in Internet Explorer shown in Figure 5-3.

Figure 5-3. Bear Claw page with standard fonts.


Camp Bear Claw - Microsoft Internet Explorer
File Edit View Go Favorites Help
Done

Camp Bear Claw

We have the following activities:

Archery

Arts and Crafts

Horseback Riding

Hiking

Campfire Song Times

C++ Programming

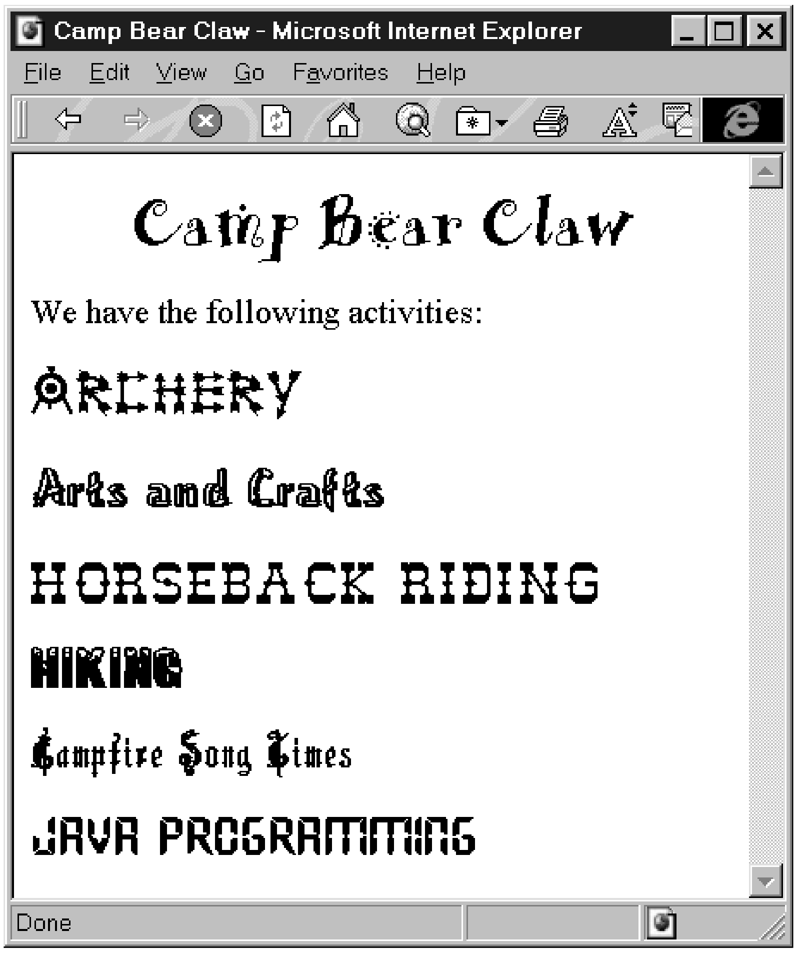
Listing 5.3 A boring summer camp

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0 Transitional//EN">
<HTML>
<HEAD>
  <TITLE>Camp Bear Claw</TITLE>
</HEAD>
```

```
<BODY>
<H1>Camp Bear Claw</H1>
We have the following activities:
<H2>Archery</H2>
<H2>Arts and Crafts</H2>
<H2>Horseback Riding</H2>
<H2>Hiking</H2>
<H2>Campfire Song Times</H2>
<H2>C++ Programming</H2>
</BODY>
</HTML>
```

However, by adding `font-family` entries (Listing 5.4 and Listing 5.5), we obtain a much more pleasing result (Figure 5-4).

Figure 5-4. Bear Claw page with custom fonts.


Camp Bear Claw - Microsoft Internet Explorer
File Edit View Go Favorites Help
Camp Bear Claw
We have the following activities:
ARCHERY
Arts and Crafts
HORSEBACK RIDING
HIKING
Campfire Song Times
JAVA PROGRAMMING
Done

Listing 5.4 An exciting summer camp

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0 Transitional//EN">
<HTML>
<HEAD>
  <TITLE>Camp Bear Claw</TITLE>
  <LINK REL=STYLESHEET HREF="CampBearClaw.css" TYPE="text/css">
</HEAD>
<BODY>
<H1>Camp Bear Claw</H1>
```

```
We have the following activities:
<H2 CLASS="archery">Archery</H2>
<H2 CLASS="arts">Arts and Crafts</H2>
<H2 CLASS="horseback">Horseback Riding</H2>
<H2 CLASS="hiking">Hiking</H2>
<H2 CLASS="campfire">Campfire Song Times</H2>
<H2 CLASS="java">Java Programming</H2>
</BODY>
</HTML>
```

Listing 5.5 `CampBearClaw.css`

```
H1 { text-align: center;
     font-family: Funstuff }
H2.archery { font-family: ArcheryDisplay }
H2.arts { font-family: ClampettsDisplay }
H2.horseback { font-family: Rodeo }
H2.hiking { font-family: SnowtopCaps }
H2.campfire { font-family: Music Hall }
H2.java { font-family: Digiface }
```

Note that if `font-family` is used inside a `STYLE` attribute and enclosed in double quotes (e.g., `<P STYLE="font-family: SomeFont">`), the multiword font names can be enclosed in single quotes as per the CSS1 specification. In addition, instead of a single font name, you can use a comma-separated list indicating the preferred order. The system will choose the first font available. You can put a generic typeface name such as "sansserif" at the end, or let the system choose a default font if none of the preferred choices are available.

Choosing font faces appropriately can make pages significantly more attractive, especially for intranet applications where a standard set of fonts can be assumed. On the Web, however, finding a font that is likely to be installed on all of the different systems that access your page is not easy. If you use a graphical or foreign-language font to present icons or cyrillic text in your page and that particular font is not installed on the user's machine, your page will appear as a jumbled mess. Your best bet is to create GIF files in such a case, but they can result in much longer download times and prevent users and Web indexing robots from correctly searching your page. Cascading Style Sheets, Level 2, offers an extremely useful extension: dynamic fonts. See `fontdef` later in this section for implementing dynamic fonts in Netscape, and see `font-face` for implementing dynamic fonts in Internet Explorer.

font-variant

normal | small-caps

This property is intended to be applied to text to make a variation in small caps. Legal values are `normal` and `small-caps`. Internet Explorer displays `small-caps` in capital letters, not *small* capital letters. Netscape does not support this font property.

font

The `font` property lets you group `font-weight`, `font-variant`, `font-style`, `font-size`, `line-height`, and `font-family` in a single entry. Items can be omitted, but if included, they should appear in that order, with spaces, between all except `font-size` and `line-height`, which should be separated by a slash (/). For instance,

```
P { font-weight: demi-bold;
    font-style: italic;
    font-size: 14pt;
    line-height: 150%;
    font-family: Times, serif }
```

could be replaced by

```
P { font: demi-bold italic 14pt/150% Times, serif }
```

fontdef

Dynamic fonts, an extension to CSS1, let you supply a font definition file for use in your HTML document. Netscape supports *dynamic fonts both through the* `fontdef` *style,*

```
<STYLE TYPE="text/css">
<!--
@fontdef URL(http://.../font-file.pfr);
...
-->
</STYLE>
```

and through the `LINK` *element placed in the* `HEAD` *section of the document,*

```
<LINK REL="fontdef"
      SRC="http://.../font-file.pfr">
```

After providing the link to the `pfr` file, you simply use the dynamic font as any normal font. Internet Explorer does not support the `pfr` file format directly. However, Bitstream, Inc., provides an ActiveX control to enable `pfr` font files in Internet Explorer. For more information on dynamic fonts and for several free font-definition files, see the following sites:

http://www.truedoc.com/webpages/intro/

http://www.bitstream.com/

font-face

Internet Explorer supports dynamic fonts through the `font-face` style defined in Cascading Style Sheets, Level 2 (see http://www.w3.org/TR/REC-CSS2/fonts.html#font-descriptions). To specify a dynamic font in Internet Explorer, use

```
<STYLE TYPE="text/css">
<!--
@font-face {
  font-family: fontname;
  font-style: normal;
  font-weight: normal;
  src: url(http://.../font-file.eot)
}
</STYLE>
```

Note that the font file format differs between Internet Explorer (`eot`) and Netscape (`pfr`). Microsoft provides Web Embedding Fonts Tools (WEFT) for creating `eot` object files. For examples of Microsoft dynamic fonts see,

http://www.microsoft.com/typography/

The site also provides a free download of WEFT Version 2.

5.6 Foreground and Background Properties

Style sheets support a powerful and convenient way of changing foreground colors, background colors, and background images for regions of text. Style sheets don't suffer from the problem of a color becoming invisible if the user overrides the author's settings on the `BODY`'s `BGCOLOR`. Setting the `BGCOLOR` through a style sheet overrides the browser setting.

color

color-name | #RRGGBB | #RGB | rgb(rrr, ggg, bbb) | rgb(rrr%, ggg%, bbb%) This property specifies the color of text or the foreground color of the associated section, using any of the standard color designators (Table 5.2 in Section 5.11). For example,

```
P  { color : blue }
H1 { color : #00AABB }
H2 { color : #0AB }
H3 { color : rgb(255, 0, 0 ) } /* red */
H4 { color : rgb(0, 0, 255 ) } /* blue */
```

background-color

transparent |

color-name | #RRGGBB | #RGB | rgb(rrr, ggg, bbb) | rgb(rrr%, ggg%, bbb%) This property specifies the background color of the associated section by using any of the standard color designators. Alternatively, the keyword `transparent` can be used to let an inherited color show through.

background-image

none | url(*filename*)

This property specifies an image to be used as the background of the specified region only. Authors should supply a background color to use if the image is unavailable or if the user has disabled image loading. For example,

```
H2 { background-image: url(Bluedrop.gif);}
```

background-repeat

repeat | repeat-x | repeat-y | norepeat

This property takes values of `repeat`, `repeat-x`, `repeat-y`, or `no-repeat` and means, respectively, that the image should be tiled in both directions, just in the x direction, just in the y direction, or displayed once in the background but not tiled. For instance,

```
BODY {
  background-image: url(Bluedot.gif);
  background-repeat: repeat-x;
}
```

background-attachment

scroll | fixed

This property determines whether the background image scrolls with the content (value: `scroll`) or is fixed (value: `fixed`). Netscape does not support this background property.

background-position

[top | center | bottom] [left | center | right] |

[pt, pc, in, cm, mm][pt, pc, in, cm, mm] |

[em, ex, px, %][em, ex, px, %]

The `background-position` property specifies the position of the background image with respect to the upper-left corner of the region You normally specify a pair of values (separated by a space), specified with the keywords `left`/`center`/`right`, `top`/`middle`/`bottom`, percentages, or distances in the standard units (see Table 5.1 in Section 5.11). For instance, a value of `50%` means to put the center of the image at the center of the region. Similarly, a horizontal value of `25px` means to position the left side of the image 25 pixels from the left side of the region. If you supply a single value instead of a pair, the value applies just to the horizontal position; the vertical position is set to 50%. The default position is `0% 0%`. Negative positions are permitted, allowing images to hang into margins or previous text sections. For example,

```
BODY { background-image: url(Marty.jpg);
       background-position: 10% 10%; }
H1 { background-image: Bluedrop.gif;
     background-position: center; } /* 50% 50% */
```

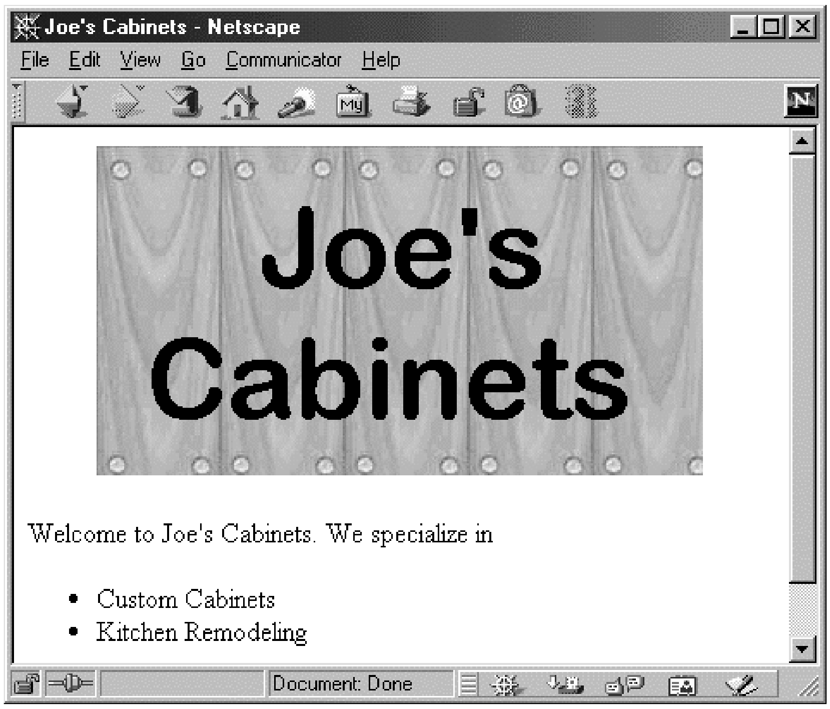
Netscape does not support the `background-position` property.

background

The `background` property lets you combine `background-color`, `background-image`, `background-repeat`, `background-attachment`, and `background-position` in a single entry.

As an example, consider Listing 5.6 and Listing 5.7, which defines a page for Joe's Carpenter Shop and uses "wooden" boards repeated horizontally as the background image of the title banner. Figure 5-5 shows the result in Netscape Communicator 4.7.

Figure 5-5. Background images can be used for individual text sections.


Joe's Cabinets - Netscape
File Edit View Go Communicator Help
Document: Done

Joe's Cabinets

Welcome to Joe's Cabinets. We specialize in

- Custom Cabinets
- Kitchen Remodeling

Listing 5.6 `Cabinets.html`

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0 Transitional//EN">
<HTML>
```

```
<HEAD>
  <TITLE>Joe's Cabinets</TITLE>
  <LINK REL=STYLESHEET HREF="Cabinets.css" TYPE="text/css">
</HEAD>
<BODY>
<CENTER>
<TABLE WIDTH=360 HEIGHT=199>
  <TR><TD ALIGN="CENTER" CLASS="banner">Joe's Cabinets
</TABLE>
</CENTER>
<P>
Welcome to Joe's Cabinets. We specialize in
<UL>
  <LI>Custom Cabinets
  <LI>Kitchen Remodeling
  <!--  Etc  -->
</UL>
<!--  Etc  -->
</BODY>
</HTML>
```

Listing 5.7 `Cabinets.css`

```
.banner { background: url(images/boards.jpg) repeat-x;
          font-size: 50pt;
          font-family: Arial Rounded MT Bold }
```

5.7 Text Properties

The text properties control the way text in a paragraph is laid out. Options let the user customize characteristics like interword spacing, paragraph justification, and indentation of leading lines in paragraphs.

word-spacing, letter-spacing

normal | +/– pt, pc, in, cm, mm | +/– em, ex, px

These properties specify a change to the default spacing between words or characters. Values are expressed in the standard length units (see Table 5.1 in Section 5.11) or by the keyword `normal`. Numeric values can be positive (add the space to the default word spacing) or negative (subtract the space from the default). Neither Netscape nor Internet Explorer supports `word-spacing`. Netscape does not support `letter-spacing`.

text-decoration

none | underline | overline | line-through | blink

The `text-decoration` property describes text additions, or "decorations" that are added to the text of an element. Legal values are `none`, `underline`, `overline`, `line-through`, and `blink`. For instance, to make hypertext links blue but not underlined and to underline text in paragraphs, use:

```
A:link { color:blue; text-decoration: none }
P { text-decoration: underline }
```

Note that Internet Explorer does not support the `blink` value. Netscape does not support the `overline` value.

vertical-align

top | bottom | **baseline** | middle | sub | super | text-top | text-bottom | %

This property determines how elements are positioned vertically. The value can be a percentage (positive or negative), indicating how far to raise the baseline of the element above the baseline of the parent element, or can be a symbolic value. Legal symbolic values are `top` (align the top with the tallest element in the line), `bottom` (align the bottom with the lowest element in the line), `baseline` (align the baseline of the element with the baseline of the parent element), `middle` (align the middle of the element with a point halfway up from the parent element's baseline), `sub` (make the element a subscript), `super` (make the element a superscript), `text-top` (align the top with the top of the parent element's font), and `text-bottom` (align the bottom of the element with the bottom of the parent element's font).

text-transform

none | uppercase | lowercase | capitalize

This property determines whether the text should be changed to all uppercase (`uppercase`), changed to all lowercase (`lowercase`), have the first letter of each word uppercase (`capitalize`), or have inherited text transformations suppressed (`none`).

text-align

left | right | center | justify

This property creates left-aligned, center-aligned, and right-aligned paragraphs or paragraphs aligned on both sides (i.e., justified).

text-indent

+/– pt, pc, in, cm, mm | +/– em, ex, px, %

This property specifies the indentation of the *first* line of the paragraph and is calculated with respect to the existing left margin as specified by `margin- left`. Values can be in standard length units or can be a percentage interpreted with respect to the width of the parent element. The default value is 0. Negative values indicate that the first line should hang out into the left margin. For example,

```
P { text-indent: -25px } /* Hanging indent */
```

line-height

normal | *number* | pt, pc, in, cm, mm | em, ex, px, %

This property specifies the height of each line—the distance between two consecutive baselines in a paragraph (sometimes known as *leading,* pronounced "ledding"). In addition to the standard length units (see Table 5.1 in Section 5.11), a percent value can be supplied, interpreted with respect to the font size. For instance,

```
.double { line-height: 200% }
.triple { line-height: 3 } /* 3x the font size */
DIV { line-height: 1.5em }
```

white-space

normal | pre | nowrap

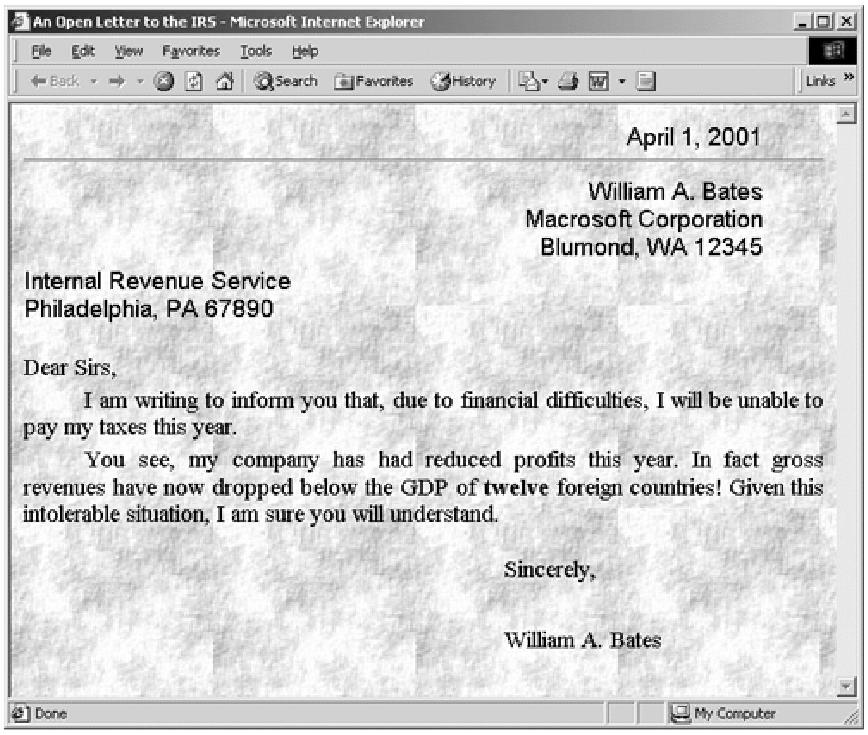
The `white-space` property specifies how spaces, tabs, carriage returns, and new lines should be treated within the element. Legal values are `normal` (collapse white space to a single space), `pre` (maintain whites space as in the `PRE` element), and `nowrap` (only wrap at `BR` elements). Internet Explorer does not support the `white-space` style. Netscape does not recognize the `nowrap` option.

By way of example, consider a Web page that is intended to look like a business letter. Listing 5.8 gives the HTML source. First, the default spacing between all paragraphs is reduced by

```
P { margin-top: 5px }
```

Next, right-aligned (`rhead`) and left-aligned (`lhead`) paragraph classes are created for the date, return address, and receiver's address. The main body of the letter (`body`) uses indented lines and justified text, and the footer used for the signature (`foot`) is indented 60% and has a large interline spacing. Figure 5-6 shows the result in Internet Explorer 5.0 on Windows 2000.

Figure 5-6. Using text properties enables customized text formatting.


An Open Letter to the IRS - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Search Favorites History Links
Done
My Computer

April 1, 2001

William A. Bates
Macrosoft Corporation
Blumond, WA 12345

Internal Revenue Service
Philadelphia, PA 67890

Dear Sirs,

I am writing to inform you that, due to financial difficulties, I will be unable to pay my taxes this year.

You see, my company has had reduced profits this year. In fact gross revenues have now dropped below the GDP of **twelve** foreign countries! Given this intolerable situation, I am sure you will understand.

Sincerely,

William A. Bates

Listing 5.8 `Bates.html`

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0 Transitional//EN">
<HTML>
<HEAD>
  <TITLE>An Open Letter to the IRS</TITLE>
  <LINK REL=STYLESHEET HREF="Bates.css" TYPE="text/css">
</HEAD>
<BODY BACKGROUND="images/bond-paper.jpg">
<P CLASS="rhead">
April 1, 2001
<HR>
<P CLASS="rhead">
William A. Bates<BR>
Macrosoft Corporation<BR>
Blumond, WA 12345
<P CLASS="lhead">
Internal Revenue Service<BR>
```

```
Philadelphia, PA 67890
<P>
<BR>
Dear Sirs,
<P CLASS="body">
I am writing to inform you that, due to financial difficulties,
I will be unable to pay my taxes this year.
<P CLASS="body">
You see, my company has had reduced profits this year. In fact
gross revenues have now dropped below the GDP of <B>twelve</B>
foreign countries! Given this intolerable situation, I am sure
you will understand.
<P CLASS="foot">
Sincerely,<BR>
William A. Bates
</BODY>
</HTML>
```

Listing 5.9 `Bates.css`

```
P { margin-top: 5px }
P.rhead { text-align: right;
          margin-right: 0.5in;
          font-family: sans-serif }
P.lhead { font-family: sans-serif }
P.body { text-align: justify;
         text-indent: 0.5in }
P.foot { margin-left: 60%;
         line-height: 300% }
```

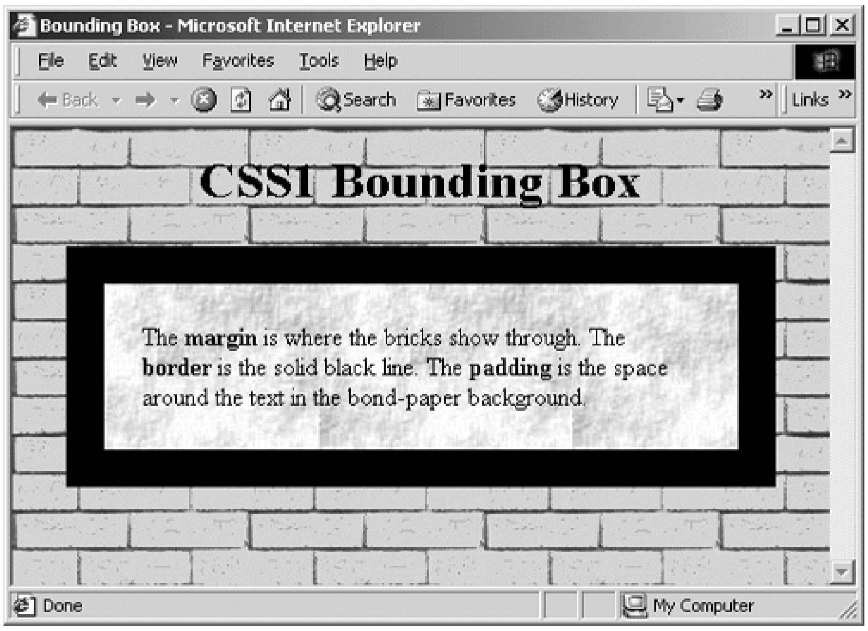
5.8 Properties of the Bounding Box

Cascading style sheets assume that all elements will result in one or more *rectangular* regions. Such a region is known as the "bounding box" (or simply "box") and contains a margin, border, padding area, and main element, each nested within the other. The width and height of the total box is the sum of the width and height of the main element, the padding area that surrounds the main element, the border surrounding the padding, and the margins surrounding the border. Margins are always transparent, letting the color or image underneath show through. The padding always takes on the background color or image of the main element. The border, which is in between, can have a separate background. For instance, you could set margins, border, and padding to a quarter of an inch each (but using different colors/patterns) by the following:

```
P { margin: 0.25in;
    border: 0.25in solid black;
    padding: 0.25in;
    background: URL(images/bond-paper.jpg) }
BODY { background: URL(images/bricks.jpg) }
```

Figure 5-7 shows a page created with these settings. The details of the property specifications are explained later, but for now the important point is that the margin is outside the border, which is outside the padding, which is outside the main element area.

Figure 5-7. The differences between margin, border, and padding.


Bounding Box - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Search Favorites History Links
Done
My Computer

CSS1 Bounding Box

The **margin** is where the bricks show through. The **border** is the solid black line. The **padding** is the space around the text in the bond-paper background.

Margins

An element's margins are the reserved areas around the element where the background colors or images show through. Margins can be negative to allow paragraphs to overlap.

margin-left, margin-right, margin-top, margin-bottom

auto | +/– pt, pc, in, cm, mm | +/– em, ex, px, %

These properties set the left, right, top, and bottom margins, using normal length units (see Table 5.1 in Section 5.11), percentages, or the keyword `auto`. The default value is 0. Negative values are permitted; they allow text or graphics to hang into the left margin or overlap previous paragraphs. For instance,

```
P  { margin-right: 5ex }
H1 { margin-top: 200% }
```

margin

This property lets you set the top, right, bottom, and left margins (in that order) in one property. If only one value is supplied, then the value applies to all four margins. If two or three values are supplied, then values for any unspecified margin are taken from the opposite margin. Negative values are permitted and are sometimes used to implement layered text effects.

Borders

An element's borders are the reserved areas around the element where a specific color or background image is displayed. The borders are inside the margins. Borders can have zero thicknesses but cannot be negative.

border-left-width, border-right-width, border-top-width, border-bottom-width

none | thin | **medium** | thick **|**

pt, pc, in, cm, mm | em, ex, px

These properties set the left, right, top, and bottom border sizes, using normal length units (see Table 5.1 in Section 5.11) or the symbolic names `thin`, `medium`, `thick`, or `none`. Negative values are prohibited.

border-width

none |.3 thin | **medium** | thick |

pt, pc, in, cm, mm | em, ex, px

This property is a shorthand method for setting `border-width-top`, `border-width-right`, `border-width-bottom`, and `border-width- left` (in that order) in one fell swoop. If only one value is supplied, the value applies to all four borders. If two or three values are supplied, values for any missing border are taken from the opposite border. For example,

```
DIV { border-width: medium thin }
```

produces a division with a medium border for the top and bottom sides, and a thin border for the left and right sides.

border-color

color-name | #RRGGBB | #RGB | rgb(rrr, ggg, bbb) | rgb(rrr%, ggg%, bbb%)

This property sets the border colors. One to four values can be supplied, specifying characteristics for the top, right, bottom, and left borders in the same manner as with `border-width`. Each value can be a color specified in the normal way (see Table 5.2 in Section 5.11). For example,

```
P { border-style: solid;
    border-color: black gray gray black;
}
```

Netscape does not properly support specification of more than one color value.

border-style

none | dotted | dashed | solid | double | groove | ridge | inset | outset

This property specifies the way in which the borders will be drawn. One to four values can be supplied, specifying characteristics for the top, right, bottom, and left borders in the same manner as with `border-width`. Each value can be one of `none`, `dotted`, `dashed`, `solid`, `double`, `groove`, `ridge`, `inset`, or `outset`. The main element's background shows through for the nonforeground part of `double`. Neither Netscape nor Internet Explorer supports the `dashed` or `dotted` values. In addition, Netscape does not supports the values for this style directly. These values can be stated in the `border` style. For example,

```
P { border-sytle: ridge }
```

is not recognized by Netscape, but the following is properly recognized,

```
P { border: ridge }
```

Internet Explorer properly supports both approaches.

border-left, border-right, border-top, border-bottom

This property lets you set the width, style, and color for each of the four borders. For instance, to display major headings in red with a solid blue line above and below, you could use:

```
H1 { color: red;
     border-top: 10px solid blue;
     border-bottom: 10px solid blue }
```

border

This property sets the width, style, and color for all four borders at once. For instance, the quarter-inch solid black border of Figure 5-7 was specified by:

```
border: 0.25in solid black
```

Padding

An element's padding area is the reserved space around an element and inside its borders where the background color or image of the element itself shows through. Padding area sizes cannot be negative.

padding-left, padding-right, padding-top, padding-bottom

pt, pc, in, cm, mm | em, ex, px, %

These properties let you set the left, right, top, and bottom sizes of the padding area. Recall that the padding area is inside the margin and the border. The margin lets the background of the parent element (often the `BODY`) show through, the border can have an independent background, and the padding area has the same background as the main element it is associated with. Values can be lengths (see Table 5.1 in Section 5.11) or percentages, where percentages are interpreted with respect to the parent element's width and height. The default value is 0. Negative values are not allowed.

padding

This property lets you set the sizes of the top, right, bottom, and left sides of the padding area in one location. If only one value is supplied, the value applies to all four sides. Otherwise, if fewer than four values are supplied, then the value for any missing side is taken from the opposite side. For instance, the quarter-inch padding area used in Figure 5-7 was specified with

```
padding: 0.25in
```

Bounding Box Display Types

Most HTML elements can have margins, borders, and padding areas. However, the way these box components are interpreted depends on whether an element is embedded in another paragraph (i.e., an inline, text-level element), is a separate block-level element, or is part of a list. The `display` property can be used to change how the box is interpreted.

display

block | inline | list-item | none

This property determines whether the element should be considered to have a separate bounding box, as would a paragraph such as `P` or `PRE`, or have inline bounding boxes on each line inside an existing box, as would the various character-style elements (`B`, `I`, `CODE`, and so forth). Legal values are `block`, `inline`, `list-item`, and `none`. The `list-item` value is treated just like `block` except that a list item marker is added (see Section 5.10).

5.9 Images and Floating Elements

Most style rules apply to elements at fixed locations. Images and text items that "float" to the margins are in a special category, however.

width, height

auto | pt, pc, in, cm, mm | em, ex, px

These properties specify a fixed size for the element and are usually applied to images. Values can be in normal length units (see Table 5.1 in Section 5.11) or can be the keyword `auto`. For instance, a "bullet" type might be created as follows:

```
IMG.bullet { width: 50px; height: 50px }
```

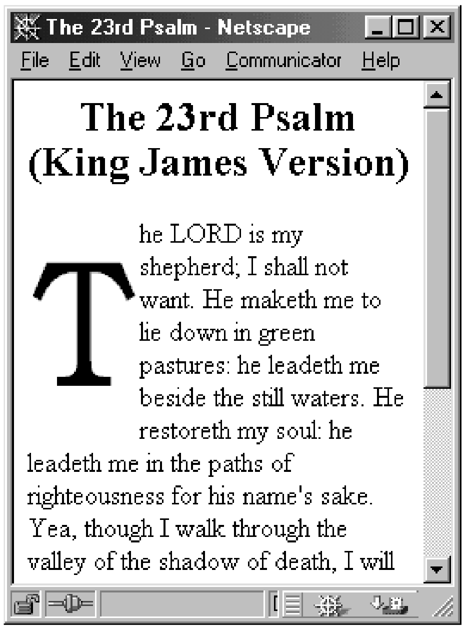
The `auto` keyword applies to images where only one of width or height is specified as a length; it means that the image should be scaled, maintaining the original aspect ratio.

float

none | left | right

This property lets elements float into the left or right margins with text wrapping around. The legal values are `left`, `right`, and `none`. This style property can be used to implement drop caps, floating images, and the like. For instance, Listing 5.10 implements a 75 point drop capital to lead off Psalm 23. The result is shown in Figure 5-8.

Figure 5-8. The `float` property can be used to implement "drop caps."


The 23rd Psalm - Netscape
File Edit View Go Communicator Help

The 23rd Psalm
(King James Version)

The LORD is my shepherd; I shall not want. He maketh me to lie down in green pastures: he leadeth me beside the still waters. He restoreth my soul: he leadeth me in the paths of righteousness for his name's sake. Yea, though I walk through the valley of the shadow of death, I will

Listing 5.10 `Psalm23.html`

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0 Transitional//EN">
<HTML>
<HEAD>
  <TITLE>The 23rd Psalm</TITLE>
<STYLE>
<!--
SPAN { float: left;
       font-family: "Cushing Book";
       font-size: 75pt }
-->
</STYLE>
</HEAD>
<BODY>
<H2 ALIGN="CENTER">
The 23rd Psalm (King James Version)</H2>
<SPAN>T</SPAN>he LORD is my shepherd; I shall not want.
```

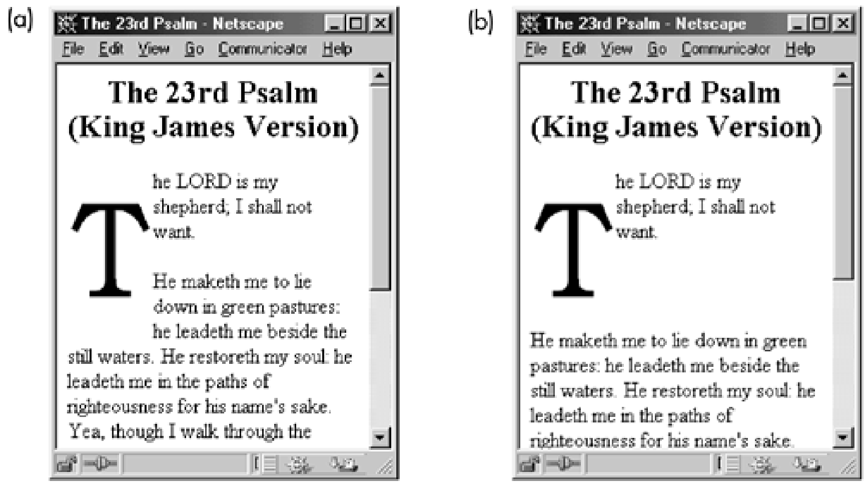
```
He maketh me to lie down in green pastures: he leadeth me
beside the still waters. He restoreth my soul: he leadeth me
in the paths of righteousness for his name's sake. Yea,
though I walk through the valley of the shadow of death, I
will fear no evil: for thou art with me; thy rod and thy
staff they comfort me. Thou preparest a table before me in
the presence of mine enemies: thou anointest my head with oil;
my cup runneth over. Surely goodness and mercy shall follow me
all the days of my life: and I will dwell in the house of the
LORD forever.
</BODY>
</HTML>
```

clear

none | left | right | both

This property specifies whether the element permits floating elements on its sides. Legal values are `left` (skip below any floating elements on the left), `right` (skip past floating elements on the right), `both` (skip past all floating elements), and `none` (permit floating elements). For instance, in the 23rd Psalm shown in Listing 5.10 and Figure 5-8, suppose a new paragraph was inserted after the first sentence. If the paragraph break were simply `<P>`, the result would be as shown in Figure 5-9 (a), with the second sentence beginning before the bottom of the drop cap T.

Figure 5-9. (a) When a paragraph break is added, the default behavior is to permit the floating element and continue the flow, instead of skipping past the floating element. (b) Paragraphs can skip past floating elements from previous paragraphs.


(a)
The 23rd Psalm - Netscape
File Edit View Go Communicator Help
(b)
The 23rd Psalm - Netscape
File Edit View Go Communicator Help

(a)

The 23rd Psalm (King James Version)

The LORD is my shepherd; I shall not want.

He maketh me to lie down in green pastures: he leadeth me beside the still waters. He restoreth my soul: he leadeth me in the paths of righteousness for his name's sake. Yea, though I walk through the

(b)

The 23rd Psalm (King James Version)

The LORD is my shepherd; I shall not want.

He maketh me to lie down in green pastures: he leadeth me beside the still waters. He restoreth my soul: he leadeth me in the paths of righteousness for his name's sake.

However, if the paragraph was `<P STYLE="clear: left">`, then the result would be as shown in Figure 5-9 (b), with the second sentence beginning below the bottom of the initial T.

5.10 List Properties

Cascading style sheets allow you to customize the way list items are formatted in ordered lists (`OL`), unordered lists (`UL`), and definition lists (`DL`). These properties are not supported by Internet Explorer version 3, and Netscape 4 supports only the `list-style-type` property.

list-style-image

none | url(*filename*)

This property allows you to set your own "bullets" for lists. However, this property is only supported by Internet Explorer. The value should be a URL or the keyword `none`. For instance, the following would set the default bullet for unordered lists to be a diamond, plus set up a "star" class that can be used in unordered lists to get a star as a bullet.

```
UL { list-style-image: url(diamond.gif) }
UL.star { list-style-image: url(star.gif) }
```

If a browser supports animated GIFs, they should be permitted as bullets. Netscape does not support the `list-style-image` style.

list-style-type

none | **disc** | circle | square | decimal | upper-alpha | lower-alpha | upper-roman | lower-roman

This property sets the list item marker in the cases when the list-style image is `none` (the default for `OL` and `DL`). Legal values are `disc` (solid circle), `circle` (hollow circle), `square`, `decimal` (1, 2, 3, and so forth), `upper-alpha` (A, B, C, and so forth), `lower-alpha` (a, b, c, and so forth), `upper-roman` (I, II, III, and so forth), `lower-roman` (i, ii, iii, and so forth), and `none`.

list-style-position

outside | inside

This property, with legal values of `outside` (default) and `inside`, determines whether the list item marker runs into the paragraph (`inside`) or hangs out to the left (`outside`). Neither Netscape nor Internet Explorer supports `list-style-position`.

list-style

The `list-style` property permits you to set the list's image, style type, and position in a single property.

5.11 Standard Property Units

Cascading style sheets allow you to specify sizes and colors in a variety of different formats. You can specify lengths in either absolute or relative units, using either integers or decimal floating-point numbers. Some properties allow negative lengths, which are indicated by a leading minus sign ("–") before a length otherwise specified in the normal manner.

Lengths

Cascading style sheets permit authors to use any of the following formats in Table 5.1 for properties that describe le5.1 ngths (sizes).

Table 5.1. Absolute and Relative Length Units

cm	Centimeters (absolute unit)
em	The height of the current font (relative unit)
ex	The height of the letter "x" in the current font (relative unit)
in	Inches (absolute unit)
mm	Millimeters (absolute unit)
pc	Picas; 6 picas per inch; 12 points per pica (absolute unit)
pt	Points; 72 points per inch (absolute unit)
px	Pixels (relative unit)

Colors

Cascading style sheets allow you to specify colors in any of the ways listed.

Table 5.2. Color Formats in CSS1

color-name	This color should be one of the standard HTML colors listed in Table 1.1 (Section 1.6). Netscape and Internet Explorer also support the X11 window system color names.
`#RRGGBB`	Each of `RR`, `GG`, and `BB` should be hexadecimal numbers ranging from `00` to `FF`, as with standard HTML colors.
#RGB	This format is a shorthand notation for `#RRGGBB`. For instance, `#0AF` is equivalent to `#00AAFF`.
rgb(rrr, ggg, bbb)	In this format, each of `rrr`, `ggg`, and `bbb` should be decimal numbers ranging from 0 to 255.
rgb(rrr%, ggg%, bbb%)	In this format, each of `rrr`, `ggg`, and `bbb` should be decimal numbers ranging from 0 to 100.

5.12 Layers

Netscape 4 supports a capability known as *layers* which allows you to place HTML markup in separate rectangular regions, then to position the regions at particular absolute or relative positions on the page. Regions can overlap, and upper regions can be transparent to let lower regions show through. Layers let you create overlapping banners and sidebars, make multicolumn text, annotate diagrams and other pictures, and create composite images by placing transparent GIFs on top of other images. Furthermore, you can use JavaScript to dynamically make regions visible or invisible, to change the stacking order, or to shrink, expand, or move regions on the screen. The specifics of how to change layers dynamically is covered in Chapter 24 (JavaScript: Adding Dynamic Content to Web Pages), but it is worthwhile to keep this capability in mind when evaluating the benefits of layers. Layers can be defined in the `BODY` of the document with the `LAYER` and `ILAYER` elements. To be compliant with the HTML 4.0 specification, Netscape 6 no longer supports the `LAYER` and `ILAYER` elements. Internet Explorer only supports layers through cascading style sheets.

Core Note

Internet Explorer and Netscape 6 do not support the `LAYER` and `ILAYER` element, but do support layers through the use of style sheets.

Specifying Layers with the LAYER and ILAYER Elements

HTML Element:	**`<LAYER ...> ... </LAYER>` `<ILAYER ...> ... </ILAYER>`**
Attributes:	`ABOVE`, `BACKGROUND`, `BELOW`, `BGCOLOR`, `CLIP`, `HEIGHT`, `ID`, `LEFT`, `ONBLUR`, `ONFOCUS`, `ONLOAD`, `ONMOUSEOVER`, `ONMOUSEOUT`, `PAGEX`, `PAGEY`, `SRC`, `TOP`, `VISIBILITY`, `WIDTH`, `Z-INDEX`

The `LAYER` element creates regions that have an absolute position with respect to the window or parent layer. `ILAYER` creates inline layers: regions that are embedded in the flow of the text. Alternate text for browsers that do not support layers can be placed in a `NOLAYER` element. Contents of `NOLAYER` are ignored by browsers that support `LAYER`.

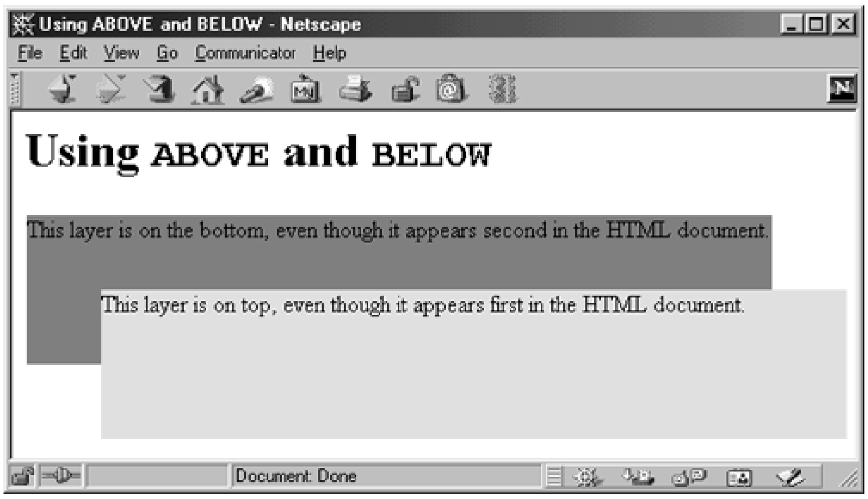
ABOVE, BELOW, Z-INDEX Normally, layers are stacked in the order in which they appear in the document, with the first being on the bottom and later ones stacking above. You can use these attributes to override this behavior. Only one of `ABOVE`, `BELOW`, or `Z-INDEX` should be used for a given layer. `Z-INDEX` takes a positive integer, where layers with higher numbers are stacked on lower-numbered layers. `ABOVE` and `BELOW` give the `ID` of a layer that should be immediately above or below the current layer. This definition is a bit counterintuitive to some people; `<LAYER id="Foo" ABOVE="Bar">` means that `Bar` is above `Foo`, not that `Foo` is above `Bar`.

Core Warning

*ABOVE and BELOW specify whether the referenced layer is above or below the current layer, not whether the current layer is above or below the referenced one. Thus, `<LAYER id="currentLayer" ABOVE="referencedLayer">` means that `currentLayer` is **below** `referencedLayer`.*

Listing 5.11 gives a simple example, with the result shown in Figure 5-10.

Figure 5-10. ABOVE, BELOW, and Z-INDEX can override layer stacking order.


Using ABOVE and BELOW - Netscape
File Edit View Go Communicator Help
Document: Done

Using ABOVE and BELOW

This layer is on the bottom, even though it appears second in the HTML document.

This layer is on top, even though it appears first in the HTML document.

Listing 5.11 Using ABOVE and BELOW

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0 Transitional//EN">
<HTML>
<HEAD>
  <TITLE>Using ABOVE and BELOW</TITLE>
</HEAD>
<BODY>
<H1>Using <CODE>ABOVE</CODE> and <CODE>BELOW</CODE></H1>

<LAYER id="Top" LEFT=60 TOP=120
       WIDTH=500 HEIGHT=100 BGCOLOR="#F5DEB3">
This layer is on top, even though it appears
first in the HTML document.
</LAYER>

<LAYER id="Bottom" ABOVE="Top" LEFT=10 TOP=70
       WIDTH=500 HEIGHT=100 BGCOLOR="gray">
This layer is on the bottom, even though it appears
second in the HTML document.

</LAYER>
</BODY>
</HTML>
```

BACKGROUND, BGCOLOR By default, layers are transparent. However, you can use a background image or color to make the layer opaque. BACKGROUND and BGCOLOR are used for this purpose. For instance, the two layers shown in the ABOVE and BELOW example (Listing 5.11, Figure 5-10) use BGCOLOR to assign colors to each layer. Although BGCOLOR always creates an opaque background, BACKGROUND can make a partially transparent background if a transparent GIF is specified.

CLIP This attribute takes comma-separated integers as a value, specifying the boundaries of the visible area of the layer. The HTML is rendered in the full size of the region, but part of the layer may be chopped off when displayed. Boundaries are specified either by `"left, top, right, bottom"` or `"right, bottom"`. The latter form is equivalent to `"0, 0, right, bottom"`.

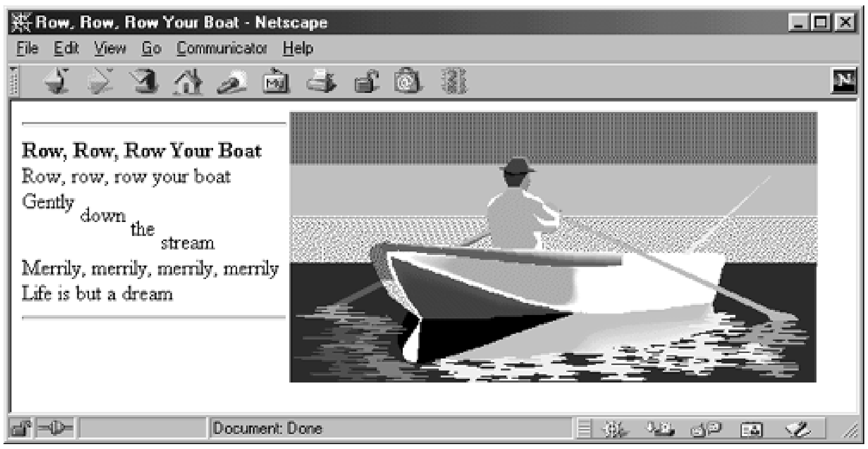
These attributes specify a minimum width and height for the layer. Otherwise, the browser uses the smallest possible width and height that encloses the layer's content. These values give minimums, not maximums; the width and height will be expanded if necessary to fit the content of layer. For example, the two layers shown in the ABOVE and BELOW example (Listing 5.11, Figure 5-10) use WIDTH and HEIGHT to specify sizes for the layers.

ID This attribute gives a name to the layer. The name can be used by the ABOVE or BELOW attributes or by JavaScript code.

LEFT, TOP, PAGEX, PAGEY For positioned layers (LAYER), these attributes specify the position of the layer in pixels relative to the position of the enclosing layer (LEFT, TOP) or with respect to the entire page (PAGEX, PAGEY). In the absence of these attributes, the layer starts at the current location in the Web page. For instance, the two layers shown in the ABOVE and BELOW example (Listing 5.11, Figure 5-10) use LEFT and TOP to specify locations for the layers.

For in-line layers (ILAYER), the LEFT and TOP attributes are interpreted with respect to the current location in the Web page. For instance, Listing 5.12 uses TOP to move each of the words in the line "Gently down the stream" 10 pixels down from the previous word. The result is shown in Figure 5-11.

Figure 5-11. Using TOP in ILAYER can move text up or down in the current paragraph.


Row, Row, Row Your Boat - Netscape
File Edit View Go Communicator Help
Document: Done

Row, Row, Row Your Boat
Row, row, row your boat
Gently down the stream
Merrily, merrily, merrily, merrily
Life is but a dream

Listing 5.12 Using TOP in ILAYER

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0 Transitional//EN">
<HTML>
<HEAD>
  <TITLE>Row, Row, Row Your Boat</TITLE>
</HEAD>
<BODY>
<IMG SRC="images/Rowboat.gif" ALIGN="RIGHT">
<HR>
<B>Row, Row, Row Your Boat</B><BR>
```

```
Row, row, row your boat<BR>
Gently
<ILAYER TOP=10>down</ILAYER>
<ILAYER TOP=20>the</ILAYER>
<ILAYER TOP=30>stream<BR>
Merrily, merrily, merrily, merrily<BR>
Life is but a dream<BR>
<HR>
</ILAYER>
</BODY>
</HTML>
```

SRC This attribute gives the URL of an HTML document to be placed inside the specified layer. Any layers in the designated file are treated as child layers of the current layer. This behavior is particularly useful when you include components that you want to put in multiple documents or when you have a frequently changing part of an otherwise static page. For instance, you could include contact information at the bottom of every Web page in a site by specifying `<LAYER SRC="Contact-Info.html></LAYER>` just before the `</BODY>` tag of each document. Alternatively, you could specify a dynamic piece of an otherwise fixed document as follows:

```
<H1>Menu for Joe's Diner</H1>
<ILAYER SRC="Blue-Plate-Special.html"></ILAYER>
<H2>Appetizers</H2> ...
<H2>Main Dishes</H2> ...
<H2>Vegetables</H2> ...
```

VISIBILITY This attribute determines whether the layer will be displayed or not. Legal values are `SHOW`, `INHERIT`, and `HIDDEN`, specifying that the layer should be shown, should inherit the parent layer's visibility, or be hidden, respectively. Hidden frames are not particularly useful for static pages but can be used with JavaScript to hide and display regions interactively.

ONBLUR, ONFOCUS, ONLOAD, ONMOUSEOVER, ONMOUSEOUT These attributes can be used to supply JavaScript code to be executed in various situations. For details, see Chapter 24 (JavaScript: Adding Dynamic Content to Web Pages).

Specifying Layers with Style Sheets

Style sheets let you do most, but not all, of the things that can be done with the `LAYER` and `ILAYER` elements. Style sheets don't let you specify an external file for the content of a layer through the `SRC` attribute, and there are no attributes equivalent to `PAGEX` and `PAGEY` for positioning nested layers independently of the parent layer's location. Furthermore, Netscape 4's support for layers is more complete and reliable when the `LAYER` or `ILAYER` element is used. For instance, when style sheets are used to specify a background color for a layer, the background of the underlying layer or page shows through in the margins between paragraphs. Background images work as expected. However, Netscape 6 does not support the `LAYER` and `ILAYER` element. Thus, use of style sheets for layers fits more cleanly with the general way in which style sheets are used and supports positioning of content using standard CSS length units (rather than just pixels). More importantly, both Internet Explorer and Netscape support layers through style sheets and are an excellent approach for creating layers that run on both browsers.

Layer declarations should use either the `ID` tag format (`#tag`) in the header, or be declared inline through `<DIV STYLE="...">` (block-level) or `<SPAN STYLE="...">` (text-level). Layer declarations should contain a `position` property. For instance, combining the style rule

```
#layer1 { position: absolute;
          left: 50px; top: 75px;
          ... }
```

with

```
<SPAN id="layer1">
...
</SPAN>
```

is roughly equivalent to

```
<LAYER id="layer1" LEFT=50 TOP=75 ...>
...
</LAYER>
```

Similarly, combining

```
#layer2 { position: relative;
          top: 10px;
          ... }
```

with

```
<SPAN id="layer2">
...
</SPAN>
```

is roughly equivalent to

```
<ILAYER id="layer2" TOP=10 ...>
...
</ILAYER>
```

In addition to the standard attributes available in CSS1, layers support the following attributes:

clip Clipping is specified by `rect(top right bottom left)` or by the keyword `auto` (the default). The `clip` property is equivalent to the `CLIP` attribute of `LAYER` and `ILAYER`. Note that the clipping region, like each of the other layer positions, can be specified by normal CSS length units (Section 5.11). These included pixels, points, inches, and centimeters, not just pixels like `LAYER` and `ILAYER`.

left, top These properties specify the left and top sides of the layer, in normal CSS length units. They are equivalent to the `LEFT` and `TOP` attributes of `LAYER` and `ILAYER`.

overflow This property determines what happens when an element's contents exceed the height or width of the layer. A value of `none` (the default) means the contents should be drawn normally, obscuring any underlying layers. A value of `clip` indicates that clipping should occur; `scroll` means the browser should scroll to accommodate the overflow.

position This property can have the value `absolute`, `relative`, or `static`. These correspond to `LAYER` elements, `ILAYER` elements, and normal, unlayered CSS elements, respectively. The default is `static`.

visibility This property determines whether a layer is visible or hidden. Legal values are `visible`, `hidden`, or `inherit`, signifying that the layer should be shown normally, hidden, or that the parent layer's visibility be used, respectively. This property is useful when you use JavaScript to change visibility values dynamically. JavaScript is discussed at more length in Chapter 24 (JavaScript: Adding Dynamic Content to Web Pages), but Listing 5.13 gives a simple example. A two-button input form is created with two invisible layers sharing the region below the buttons. The result of Listing 5.13 is shown in Figure 5-12. Clicking on the first button displays the first hidden layer (Figure 5-13), hiding the second if necessary. The document model is different in Internet Explorer and Netscape, so a helper JavaScript function, `display`, is added to provide cross-platform capability for dynamically changing the layers.

Figure 5-12. This page shows two JavaScript buttons and has two hidden layers.

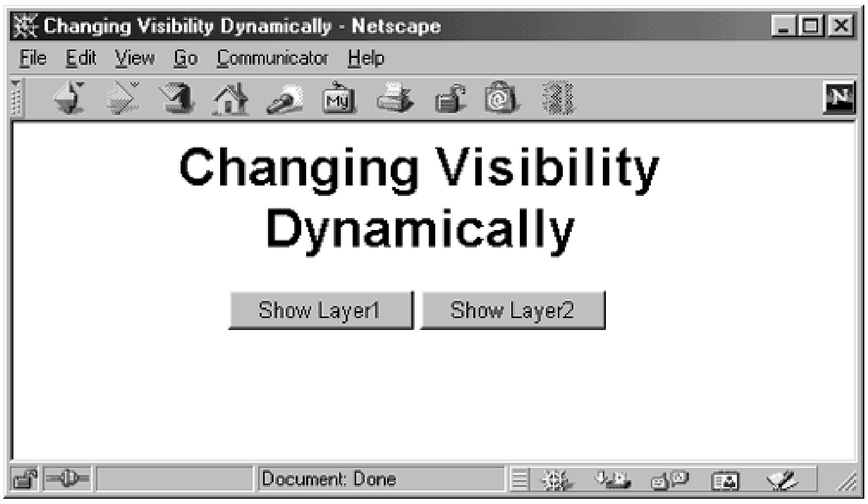
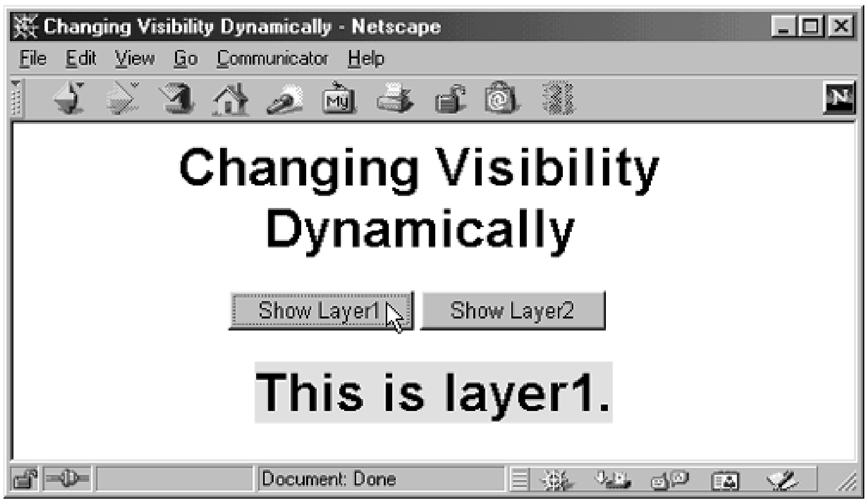

Changing Visibility Dynamically - Netscape
File Edit View Go Communicator Help
Changing Visibility Dynamically
Show Layer1
Show Layer2
Document: Done

Figure 5-13. Clicking the first button displays the first hidden layer.


Changing Visibility Dynamically - Netscape
File Edit View Go Communicator Help
Changing Visibility Dynamically
Show Layer1
Show Layer2
This is layer1.
Document: Done

Listing 5.13 Dynamically changing a layer's visibility

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0 Transitional//EN">
<HTML>
<HEAD>
  <TITLE>Changing Visibility Dynamically</TITLE>
<STYLE>
<!--
#layer1 { position: absolute; left: 0.25in; top: 1.5in;
          color: black; background-color: #F5DEB3;
          visibility: hidden }
#layer2 { position: absolute; left: 0.25in; top: 1.5in;
          color: #F5DEB3; background-color: black;
          visibility: hidden }
H1 { text-align: center;
     font-family: Arial }
FORM { text-align: center }
-->
```

```
</STYLE>
<SCRIPT TYPE="text/javascript">
<!--
function display(value1,value2){
  if(document.layers) { //Test for Netscape.
    document.layers.layer1.visibility = value1;
    document.layers.layer2.visibility = value2;
  } else {
    document.all.layer1.style.visibility = value1;
    document.all.layer2.style.visibility = value2;
  }
}
//-->
</SCRIPT>
</HEAD>
<BODY BGCOLOR="WHITE">
<H1>Changing Visibility Dynamically</H1>
<FORM>
  <INPUT TYPE="BUTTON" VALUE="Show Layer1"
         onClick="display('visible','hidden')">
  <INPUT TYPE="BUTTON" VALUE="Show Layer2"
         onClick="display('hidden','visible')">
</FORM>
<DIV id="layer1">
<H1>This is layer1.</H1>
</DIV>
<DIV id="layer2">
<H1>This is layer2.</H1>
</DIV>
</BODY>
</HTML>
```

width, height These properties specify the size of the layer; they are equivalent to the `WIDTH` and `HEIGHT` attributes of `LAYER` and `ILAYER`.

z-index Normally, layers are stacked from the bottom to the top according to their order in the HTML source. The `z-index` property overrides this behavior. Values are integers; lower numbers are stacked below higher numbers. This property is equivalent to the `Z-INDEX` attribute of `LAYER` and `ILAYER`.

5.13 Summary

Cascading style sheets provide a powerful capability for customizing the look of Web pages and are now the preferred approach for specifying page formatting in HTML 4.0 documents. Style sheets let you specify fonts, background colors, and images for individual sections of text, floating elements, margins and indentation, and list styles. In addition to the new effects that style sheets enable, they also allow you to give pages a much more consistent look across browsers. Through style sheets, layers are supported on both Netscape and Internet Explorer.

Even with this capability, however, HTML is still a markup language, not a programming language, so the types of applications you can create are limited. The Java programming language, however, lets you create general programs that can be attached to Web pages and run in the browser when the page is loaded. Java is also widely used as a general-purpose programming language, independent of any association with an HTML document or Web browser. Onward! Java technology is the topic of Part 2.

CONTENTS