6

Cascading Style Sheets™ (CSS)

Objectives

- To control the appearance of a Web site by creating style sheets.
- To use a style sheet to give all the pages of a Web site the same look and feel.
- To use the `class` attribute to apply styles.
- To specify the precise font, size, color and other properties of displayed text.
- To specify element backgrounds and colors.
- To understand the box model and how to control the margins, borders and padding.
- To use style sheets to separate presentation from content.

Fashions fade, style is eternal.
Yves Saint Laurent

A style does not go out of style as long as it adapts itself to its period. When there is an incompatibility between the style and a certain state of mind, it is never the style that triumphs.
Coco Chanel

How liberating to work in the margins, outside a central perception.
Don DeLillo

I've gradually risen from lower-class background to lower-class foreground.
Marvin Cohen

Outline

6.1 Introduction

In Chapters 4 and 5, we introduced the Extensible HyperText Markup Language (XHTML) for marking up information. In this chapter, we shift our focus to formatting and presenting information. To do this, we use a W3C technology called **Cascading Style Sheets** (**CSS**) that allows document authors to specify the presentation of elements on a Web page (e.g., fonts, spacing, margins, etc.) separately from the structure of the document (section headers, body text, links, etc.). This **separation of structure from presentation** simplifies maintaining and modifying a document's layout.

6.2 Inline Styles

A Web developer can declare document styles in many ways. This section presents **inline styles** that declare an individual element's format using the XHTML attribute **`style`**. Inline styles override any other styles applied using the techniques we discuss later in the chapter. Figure 6.1 applies inline styles to `p` elements to alter their font size and color.

```
1  <?xml version = "1.0"?>
2  <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.1//EN"
3     "http://www.w3.org/TR/xhtml11/DTD/xhtml11.dtd">
4
5  <!-- Fig. 6.1: inline.html -->
6  <!-- Using inline styles   -->
7
8  <html xmlns = "http://www.w3.org/1999/xhtml">
9     <head>
10       <title>Inline Styles</title>
```

Fig. 6.1 Inline styles. (Part 1 of 2.)

```
11      </head>
12
13      <body>
14
15         <p>This text does not have any style applied to it.</p>
16
17         <!-- The style attribute allows you to declare -->
18         <!-- inline styles. Separate multiple styles   -->
19         <!-- with a semicolon.                         -->
20         <p style = "font-size: 20pt">This text has the
21         <em>font-size</em> style applied to it, making it 20pt.
22         </p>
23
24         <p style = "font-size: 20pt; color: #0000ff">
25         This text has the <em>font-size</em> and
26         <em>color</em> styles applied to it, making it
27         20pt. and blue.</p>
28
29      </body>
30   </html>
```

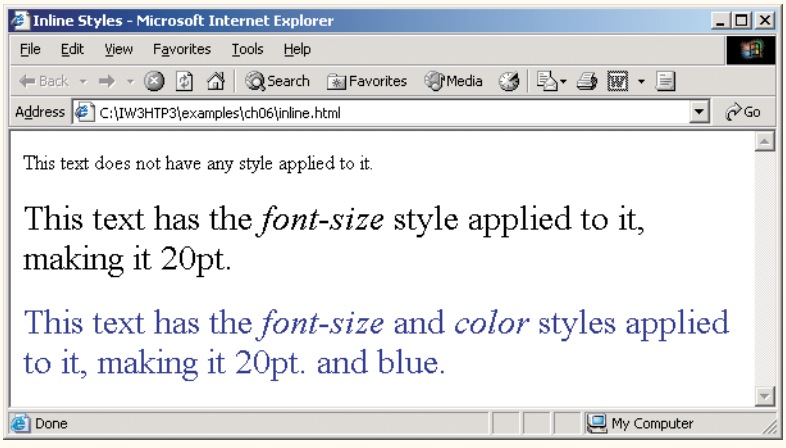

Fig. 6.1 Inline styles. (Part 2 of 2.)

The first inline style declaration appears in line 20. Attribute `style` specifies the style for an element. Each **CSS property** (the **`font-size`** property in this case) is followed by a colon and a value. In line 20, we declare this particular `p` element to use 20-point font size. Line 21 uses element **`em`** to "emphasize" text, which most browsers do by making the font italic.

Line 24 specifies the two properties, `font-size` and **`color`**, separated by a semicolon. In this line, we set the given paragraph's `color` to blue, using the hexadecimal code `#0000ff`. Color names may be used in place of hexadecimal codes, as we demonstrate in the next example. We provide a list of hexadecimal color codes and color names in Appendix B.

6.3 Embedded Style Sheets

A second technique for using style sheets is **embedded style sheets**. Embedded style sheets enable a Web-page author to embed an entire CSS document in an XHTML document's `head` section. Figure 6.2 creates an embedded style sheet containing four styles.

```
<?xml version = "1.0"?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.1//EN"
   "http://www.w3.org/TR/xhtml11/DTD/xhtml11.dtd">

<!-- Fig. 6.2: declared.html                          -->
<!-- Declaring a style sheet in the header section. -->

<html xmlns = "http://www.w3.org/1999/xhtml">
   <head>
      <title>Style Sheets</title>

      <!-- this begins the style sheet section -->
      <style type = "text/css">

         em       { background-color: #8000ff;
                    color: white }

         h1       { font-family: arial, sans-serif }

         p        { font-size: 14pt }

         .special { color: blue }

      </style>
   </head>

   <body>

      <!-- this class attribute applies the .special style -->
      <h1 class = "special">Deitel & Associates, Inc.</h1>

      <p>Deitel &amp; Associates, Inc. is an internationally
      recognized corporate training and publishing organization
      specializing in programming languages, Internet/World
      Wide Web technology and object technology education.
      Deitel &amp; Associates, Inc. is a member of the World Wide
      Web Consortium. The company provides courses on Java,
      C++, Visual Basic, C, Internet and World Wide Web
      programming, and Object Technology.</p>

      <h1>Clients</h1>
      <p class = "special"> The company's clients include many
      <em>Fortune 1000 companies</em>, government agencies,
      branches of the military and business organizations.
      Through its publishing partnership with Prentice Hall,
      Deitel &amp; Associates, Inc. publishes leading-edge
      programming textbooks, professional books, interactive
```

Fig. 6.2 Embedded style sheets. (Part 1 of 2.)

```
48          CD-ROM-based multimedia Cyber Classrooms, satellite
49          courses and World Wide Web courses.</p>
50
51       </body>
52    </html>
```

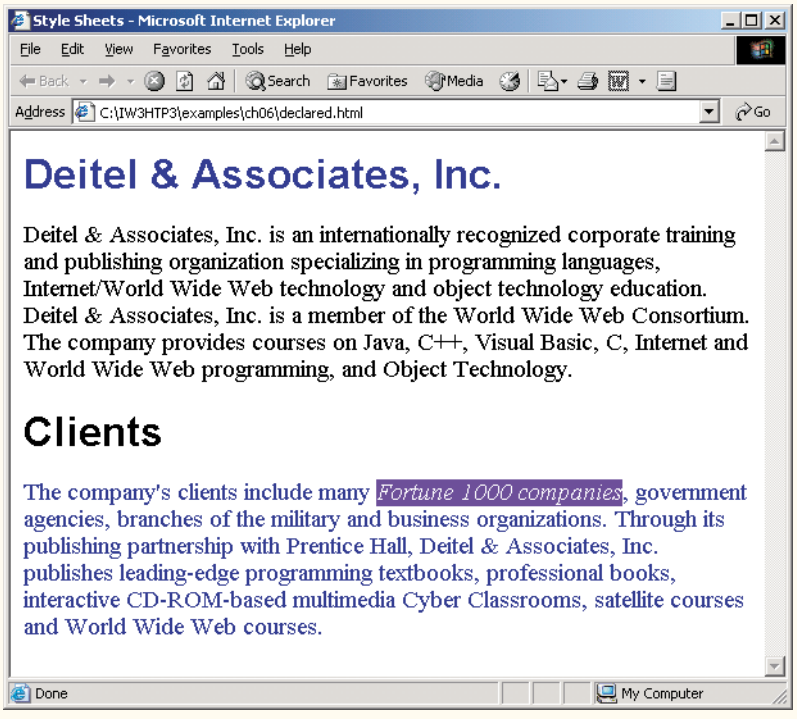

Fig. 6.2 Embedded style sheets. (Part 2 of 2.)

The `style` element (lines 13–24) defines the embedded style sheet. Styles placed in the `head` apply to matching elements wherever they appear in the entire document. The `style` element's `type` attribute specifies the **Multipurpose Internet Mail Extensions** (**MIME**) **type** that describes a file's content. CSS documents use the MIME type `text/css`. Other MIME types include `image/gif` (for GIF images) and `text/javascript` (for the JavaScript scripting language, which we discuss in Chapters 7–12).

The body of the style sheet (lines 15–22) declares the **CSS rules** for the style sheet. We declare rules for `em` (lines 15–16), `h1` (line 18) and `p` (line 20) elements. When the browser renders this document, it applies the properties defined in these rules to every element to which the rule applies. For example, the rule in lines 15–16 will be applied to all `em` elements (in this example, there is one in line 43). The body of each rule is enclosed in curly braces (`{` and `}`).

Line 22 declares a **style class** named `special`. Style classes define styles that can be applied to any type of element. In this example, we declare class `special`, which sets `color` to `blue`. We can apply this style to elements of any type, whereas the other rules in

this style sheet apply only to specific element types (i.e., `em`, `h1` or `p`). Style class declarations are preceded by a period. We will discuss how to apply a style class momentarily.

CSS rules in embedded style sheets use the same syntax as inline styles; the property name is followed by a colon (`:`) and the value of the property. Multiple properties are separated by semicolons (`;`). In the rule for `em` elements, the `color` property specifies the color of the text, and property **`background-color`** specifies the background color of the element.

The **`font-family`** property (line 18) specifies the name of the font to use. In this case, we use the `arial` font. The second value, `sans-serif`, is a **generic font family**. Not all users have the same fonts installed on their computers, so Web-page authors often specify a comma-separated list of fonts to use for a particular style. The browser attempts to use the fonts in the order they appear in the list. Many Web-page authors end a font list with a generic font family name in case the other fonts are not installed on the user's computer. In this example, if the `arial` font is not found on the system, the browser instead will display a generic `sans-serif` font, such as `helvetica` or `verdana`. Other generic font families include `serif` (e.g., `times new roman`, `Georgia`), `cursive` (e.g., `script`), `fantasy` (e.g., `critter`) and `monospace` (e.g., `courier`, `fixedsys`).

The **`font-size`** property (line 20) specifies a 14-point font. Other possible measurements in addition to **`pt`** (point) are introduced later in the chapter. Relative values— **`xx-small`**, **`x-small`**, **`small`**, **`smaller`**, **`medium`**, **`large`**, **`larger`**, **`x-large`** and **`xx-large`**— also can be used. Generally, relative values for `font-size` are preferred over point sizes because an author does not know the specific measurements of the display for each client. Relative `font-size` values permit more flexible viewing of Web pages. For example, a user may wish to view a Web page on a handheld device with a small screen. Specifying an 18-point font size in a style sheet will prevent such a user from seeing more than one or two characters at a time. However, if a relative font size is specified, such as `large` or `larger`, the actual size is determined by the browser that displays the font. Using relative sizes also makes pages more accessible to users with disabilities. Users with impaired vision, for example, may configure their browser to use a larger default font, upon which all relative sizes are based. Text that the author specifies to be `smaller` than the main text still displays in a smaller size font, yet it is clearly visible to each user.

Line 30 uses attribute **`class`** in an `h1` element to apply a style class—in this case class `special` (declared as `.special` in the style sheet). When the browser renders the `h1` element, note that the text appears on screen with the properties of both an `h1` element (`arial` or `sans-serif` font defined in line 18) and the `.special` style class applied (the color `blue` defined in line 22).

The formatting for the `p` element and the `.special` class are applied to the text in lines 42–49. All the styles applied to an element (the **parent** or **ancestor element**) also apply to the element's nested elements (**child** or **descendant elements**). The `em` element nested in the `p` element in line 43 **inherits** the style from the `p` element (namely, the 14-point font size in line 20), but retains its italic style. The `em` element has its own `color` property, so it overrides the `color` property of the `special` class. We discuss the rules for resolving these conflicts in the next section.

6.4 Conflicting Styles

Cascading style sheets are "cascading" because styles may be defined by a user, an author or a **user agent** (e.g., a Web browser). Styles "cascade," or flow together, such that the ul-

timate appearance of elements on a page results from combining styles defined in several ways. Styles defined by the user take precedence over styles defined by the user agent, and styles defined by authors take precedence over styles defined by the user. Styles defined for parent elements are also inherited by child (nested) elements. In this section, we discuss the rules for resolving conflicts between styles defined for elements and styles inherited from parent and ancestor elements.

Figure 6.2 presented an example of **inheritance** in which a child `em` element inherited the `font-size` property from its parent `p` element. However, in Fig. 6.2, the child `em` element had a `color` property that conflicted with (i.e., had a different value than) the `color` property of its parent `p` element. Properties defined for child and descendant elements have a greater **specificity** than properties defined for parent and ancestor elements. According to the W3C CSS Recommendation, conflicts are resolved in favor of properties with a higher specificity. In other words, the styles explicitly defined for a child element are more specific than the styles defined for the child's parent element; therefore, the child's styles take precedence. Figure 6.3 illustrates examples of inheritance and specificity.

```
1  <?xml version = "1.0"?>
2  <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.1//EN"
3     "http://www.w3.org/TR/xhtml11/DTD/xhtml11.dtd">
4
5  <!-- Fig 6.3: advanced.html     -->
6  <!-- More advanced style sheets -->
7
8  <html xmlns = "http://www.w3.org/1999/xhtml">
9     <head>
10       <title>More Styles</title>
11
12       <style type = "text/css">
13
14          a.nodec  { text-decoration: none }
15
16          a:hover  { text-decoration: underline;
17                     color: red;
18                     background-color: #ccffcc }
19
20          li em    { color: red;
21                     font-weight: bold }
22
23          ul       { margin-left: 75px }
24
25          ul ul    { text-decoration: underline;
26                     margin-left: 15px }
27
28       </style>
29    </head>
30
31    <body>
32
33       <h1>Shopping list for <em>Monday</em>:</h1>
34
```

Fig. 6.3 Inheritance in style sheets. (Part 1 of 2.)

```
35          <ul>
36             <li>Milk</li>
37             <li>Bread
38                <ul>
39                   <li>White bread</li>
40                   <li>Rye bread</li>
41                   <li>Whole wheat bread</li>
42                </ul>
43             </li>
44             <li>Rice</li>
45             <li>Potatoes</li>
46             <li>Pizza <em>with mushrooms</em></li>
47          </ul>
48
49          <p><a class = "nodec" href = "http://www.food.com">
50          Go to the Grocery store</a></p>
51
52       </body>
53    </html>
```

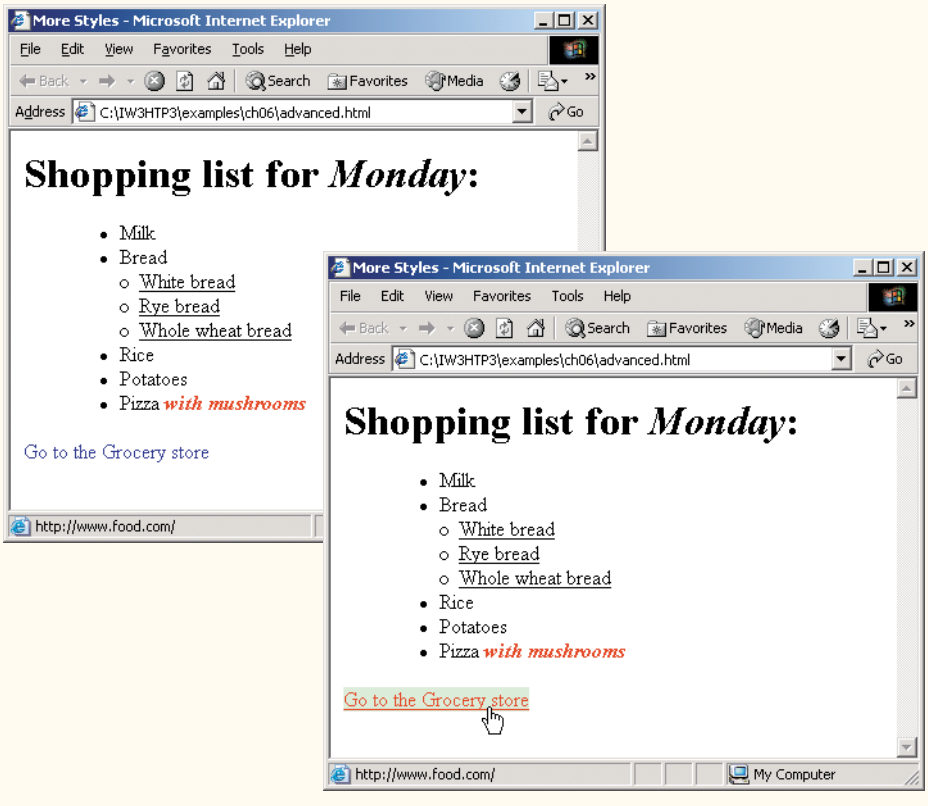

Fig. 6.3 Inheritance in style sheets. (Part 2 of 2.)

Line 14 applies property `text-decoration` to all `a` elements whose `class` attribute is set to `nodec`. The `text-decoration` property applies **decorations** to text within an

element. By default, browsers underline the text of an `a` (anchor) element. Here, we set the `text-decoration` property to `none` to indicate that the browser should not underline hyperlinks. Other possible values for `text-decoration` include **`overline`**, **`line-through`**, **`underline`** and **`blink`**. [*Note*: `blink` is not supported by Internet Explorer.] The `.nodec` appended to `a` is an extension of class styles; this style will apply only to `a` elements that specify `nodec` in their `class` attribute.

Portability Tip 6.1

To ensure that your style sheets work in various Web browsers, test them on all the client Web browsers that will render documents using your styles.

Lines 16–18 specify a style for `hover`, which is a **pseudoclass**. Pseudoclasses give the author access to content not specifically declared in the document. The `hover` pseudoclass is activated dynamically when the user moves the mouse cursor over an element. Note that pseudoclasses are separated by a colon (with no surrounding spaces) from the name of the element to which they are applied.

Common Programming Error 6.1

Including a space before or after the colon separating a pseudoclass from the name of the element to which it is applied is an error that prevents the pseudoclass from being applied properly.

Lines 20–21 declare a style for all `em` elements that are children of `li` elements. In the screen output of Fig. 6.3, note that **Monday** (which line 33 contains in an `em` element) does not appear in bold red, because the `em` element is not in an `li` element. However, the `em` element containing **with mushrooms** (line 46) is nested in an `li` element; therefore, it is formatted in bold red.

The syntax for applying rules to multiple elements is similar. For example, to apply the rule in lines 20–21 to all `li` and `em` elements, you would separate the elements with commas, as follows:

```
li, em  { color: red; font-weight: bold }
```

Lines 25–26 specify that all nested lists (`ul` elements that are descendants of `ul` elements) are to be underlined and have a left-hand margin of 15 pixels. A pixel is a **relative-length measurement**—it varies in size, based on screen resolution. Other relative lengths are **`em`** (the so-called *M*-height of the font, which is usually set to the height of an uppercase *M*), **`ex`** (the so-called *x*-height of the font, which is usually set to the height of a lowercase *x*) and percentages (e.g., `margin-left: 10%`). To set an element to display text at 150% of its default text size, the author could use the syntax

```
font-size: 1.5em
```

Other units of measurement available in CSS are **absolute-length measurements**—i.e., units that do not vary in size based on the system. These units are **`in`** (inches), **`cm`** (centimeters), **`mm`** (millimeters), `pt` (points; 1 `pt`=1/72 `in`) and **`pc`** (picas—1 `pc` = 12 `pt`).

Good Programming Practice 6.1

Whenever possible, use relative-length measurements. If you use absolute-length measurements, your document may not be readable on some client browsers (e.g., wireless phones).

In Fig. 6.3, the entire list is indented because of the 75-pixel left-hand margin for top-level `ul` elements. However, the nested list is indented only 15 pixels more (not another 75 pixels) because the child `ul` element's `margin-left` property (in the `ul ul` rule in line 25) overrides the parent `ul` element's `margin-left` property.

6.5 Linking External Style Sheets

Style sheets are a convenient way to create a document with a uniform theme. With **external style sheets** (i.e., separate documents that contain only CSS rules), Web-page authors can provide a uniform look and feel to an entire Web site. Different pages on a site can all use the same style sheet. When changes to the styles are required, the Web-page author needs to modify only a single CSS file to make style changes across the entire Web site. Figure 6.4 presents an external style sheet. Lines 1–2 are **CSS comments**. Like XHTML comments, CSS comments describe the content of a CSS document. Comments may be placed in any type of CSS code (i.e., inline styles, embedded style sheets and external style sheets) and always start with `/*` and end with `*/`. Text between these delimiters is ignored by the browser.

```
1  /* Fig. 6.4: styles.css   */
2  /* An external stylesheet */
3
4  a       { text-decoration: none }
5
6  a:hover { text-decoration: underline;
7            color: red;
8            background-color: #ccffcc }
9
10 li em   { color: red;
11           font-weight: bold;
12           background-color: #ffffff }
13
14 ul       { margin-left: 2cm }
15
16 ul ul    { text-decoration: underline;
17           margin-left: .5cm }
```

Fig. 6.4 External style sheet (`styles.css`).

Figure 6.5 contains an XHTML document that references the external style sheet in Fig. 6.4. Lines 11–12 (Fig. 6.5) show a **`link`** element that uses the **`rel`** attribute to specify a **relationship** between the current document and another document. In this case, we declare

```
1  <?xml version = "1.0"?>
2  <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.1//EN"
3     "http://www.w3.org/TR/xhtml11/DTD/xhtml11.dtd">
4
5  <!-- Fig. 6.5: external.html        -->
6  <!-- Linking external style sheets  -->
```

Fig. 6.5 Linking an external style sheet. (Part 1 of 3.)

```
7
8  <html xmlns = "http://www.w3.org/1999/xhtml">
9     <head>
10       <title>Linking External Style Sheets</title>
11       <link rel = "stylesheet" type = "text/css"
12          href = "styles.css" />
13    </head>
14
15    <body>
16
17       <h1>Shopping list for <em>Monday</em>:</h1>
18       <ul>
19          <li>Milk</li>
20          <li>Bread
21             <ul>
22                <li>White bread</li>
23                <li>Rye bread</li>
24                <li>Whole wheat bread</li>
25             </ul>
26          </li>
27          <li>Rice</li>
28          <li>Potatoes</li>
29          <li>Pizza <em>with mushrooms</em></li>
30       </ul>
31
32       <p>
33       <a href = "http://www.food.com">Go to the Grocery store</a>
34       </p>
35
36    </body>
37 </html>
```

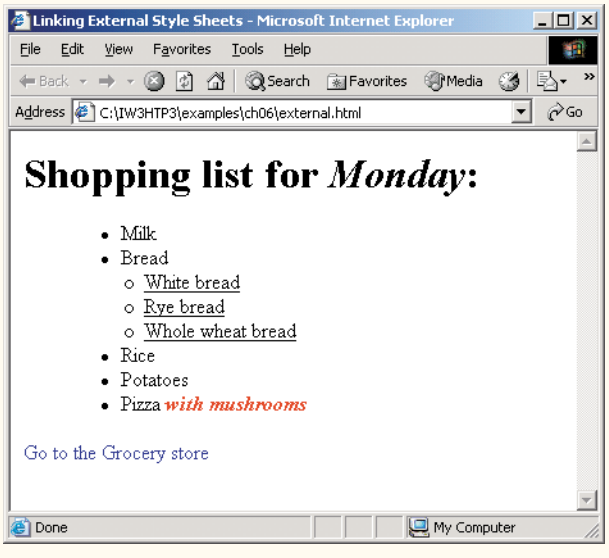

Fig. 6.5 Linking an external style sheet. (Part 2 of 3.)

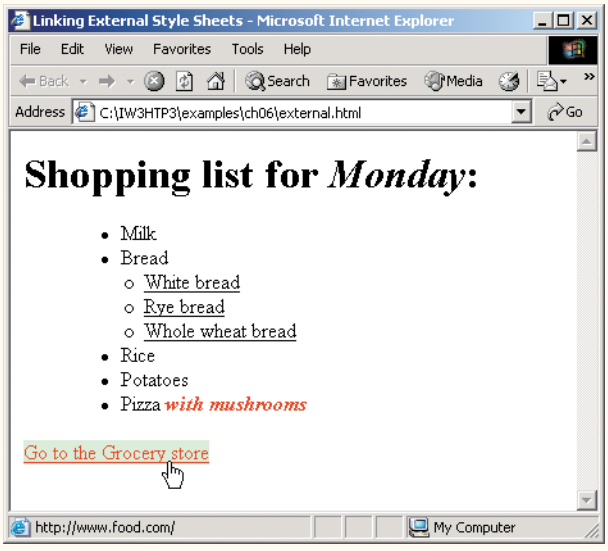

Fig. 6.5 Linking an external style sheet. (Part 3 of 3.)

the linked document to be a **`stylesheet`** for this document. The `type` attribute specifies the MIME type as `text/css`. The `href` attribute provides the URL for the document containing the style sheet. In this case, `styles.css` is in the same directory as `external.html`.

Software Engineering Observation 6.1

External style sheets are reusable. Creating them once and reusing them reduces programming effort.

Software Engineering Observation 6.2

The `link` element can be placed only in the `head` element. The user can specify `next` and `previous` as values of the `rel` attribute, which allow the user to link a whole series of documents. This feature allows browsers to print a large collection of related documents at once. (In Internet Explorer, select **Print all linked documents** *in the* **Print...** *submenu of the* **File** *menu.)*

6.6 W3C CSS Validation Service

The W3C provides a validation service (`jigsaw.w3.org/css-validator`) that validates external CSS documents to ensure that they conform to the W3C CSS Recommendation. Like XHTML validation, CSS validation ensures that style sheets are syntactically correct. The validator provides the option of either entering the CSS document's URL, pasting the CSS document's contents into a text area or uploading a CSS document.

Figure 6.6 illustrates uploading a CSS document, using the file upload feature available at `jigsaw.w3.org/css-validator/validator-upload.html`.

To validate the document, click the **Browse...** button to locate the file on your computer. Like many W3C Recommendations, the CSS Recommendation is being developed in stages (or **versions**). The current version under development is Version 3, so select **CSS version 3** in the **Profile** drop-down list. This field indicates to the validator the CSS Recommendation against which the uploaded file should be validated. Click **Submit this CSS**

file for validation to upload the file for validation. Figure 6.7 shows the results of validating `styles.css` (Fig. 6.4).

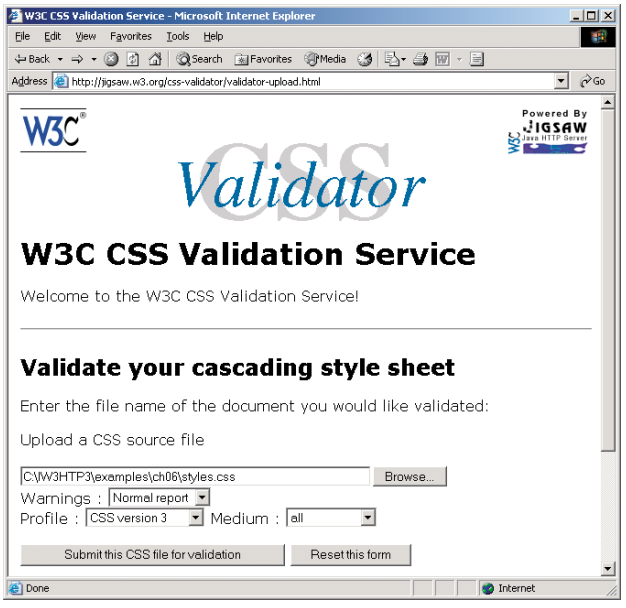

Fig. 6.6 Validating a CSS document. (Courtesy of World Wide Web Consortium (W3C).)

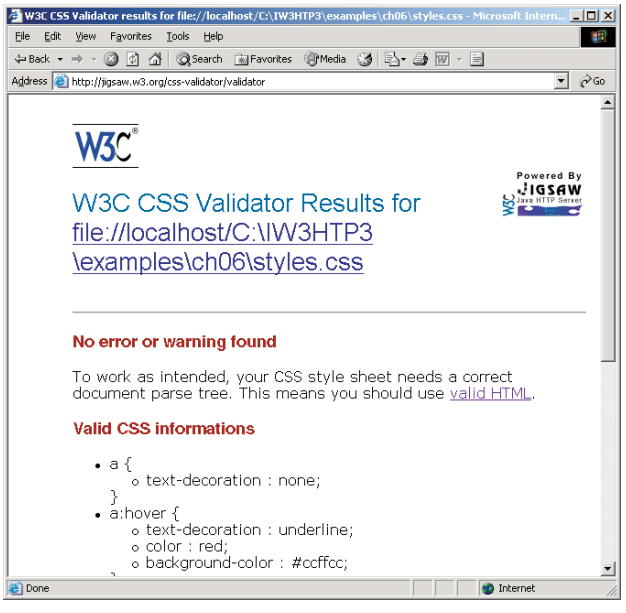

Fig. 6.7 CSS validation results. (Courtesy of World Wide Web Consortium (W3C).)

6.7 Positioning Elements

Before CSS, controlling the positioning of elements in an XHTML document was difficult—the browser determined positioning. CSS introduced the **position** property and a capability called **absolute positioning**, which gives authors greater control over how document elements are displayed. Figure 6.8 demonstrates absolute positioning.

```
1  <?xml version = "1.0"?>
2  <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.1//EN"
3     "http://www.w3.org/TR/xhtml11/DTD/xhtml11.dtd">
4
5  <!-- Fig 6.8: positioning.html        -->
6  <!-- Absolute positioning of elements -->
7
8  <html xmlns = "http://www.w3.org/1999/xhtml">
9     <head>
10       <title>Absolute Positioning</title>
11    </head>
12
13    <body>
14
15       <p><img src = "i.gif" style = "position: absolute;
16          top: 0px; left: 0px; z-index: 1"
17          alt = "First positioned image" /></p>
18       <p style = "position: absolute; top: 50px; left: 50px;
19          z-index: 3; font-size: 20pt">Positioned Text</p>
20       <p><img src = "circle.gif" style = "position: absolute;
21          top: 25px; left: 100px; z-index: 2" alt =
22          "Second positioned image" /></p>
23
24    </body>
25 </html>
```

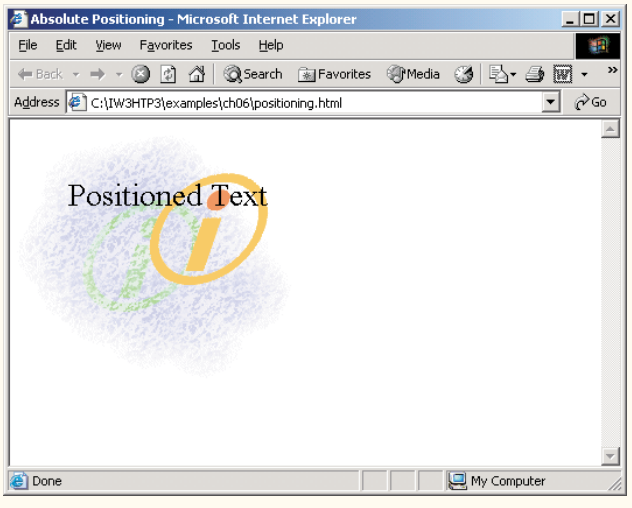

Fig. 6.8 Absolute positioning of elements with CSS.

Lines 15–17 position the first `img` element (`i.gif`) on the page. Specifying an element's `position` as **`absolute`** removes the element from the normal flow of elements on the page, instead positioning it according to the distance from the `top`, `left`, `right` or `bottom` margins of its **containing block-level element** (i.e., an element such as `body` or `p`). Here, we position the element to be `0` pixels away from both the `top` and `left` margins of the `p` element (lines 15–17).

The **`z-index`** attribute allows you to layer overlapping elements properly. Elements that have higher `z-index` values are displayed in front of elements with lower `z-index` values. In this example, `i.gif` has the lowest `z-index` (`1`), so it displays in the background. The `img` element in lines 20–22 (`circle.gif`) has a `z-index` of `2`, so it displays in front of `i.gif`. The `p` element in lines 18–19 (`Positioned Text`) has a `z-index` of `3`, so it displays in front of the other two. If you do not specify a `z-index` or if elements have the same `z-index` value, the elements are placed from background to foreground in the order they are encountered in the document.

Absolute positioning is not the only way to specify page layout. Figure 6.9 demonstrates **relative positioning**, in which elements are positioned relative to other elements.

```
1  <?xml version = "1.0"?>
2  <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.1//EN"
3     "http://www.w3.org/TR/xhtml11/DTD/xhtml11.dtd">
4
5  <!-- Fig. 6.9: positioning2.html       -->
6  <!-- Relative positioning of elements  -->
7
8  <html xmlns = "http://www.w3.org/1999/xhtml">
9     <head>
10       <title>Relative Positioning</title>
11
12       <style type = "text/css">
13
14          p           { font-size: 1.3em;
15                        font-family: verdana, arial, sans-serif }
16
17          span        { color: red;
18                        font-size: .6em;
19                        height: 1em }
20
21          .super      { position: relative;
22                        top: -1ex }
23
24          .sub        { position: relative;
25                        bottom: -1ex }
26
27          .shiftleft  { position: relative;
28                        left: -1ex }
29
30          .shiftright { position: relative;
31                        right: -1ex }
32
33       </style>
```

Fig. 6.9 Relative positioning of elements with CSS. (Part 1 of 2.)

```
34      </head>
35
36      <body>
37
38         <p>The text at the end of this sentence
39         <span class = "super">is in superscript</span>.</p>
40
41         <p>The text at the end of this sentence
42         <span class = "sub">is in subscript</span>.</p>
43
44         <p>The text at the end of this sentence
45         <span class = "shiftleft">is shifted left</span>.</p>
46
47         <p>The text at the end of this sentence
48         <span class = "shiftright">is shifted right</span>.</p>
49
50      </body>
51   </html>
```

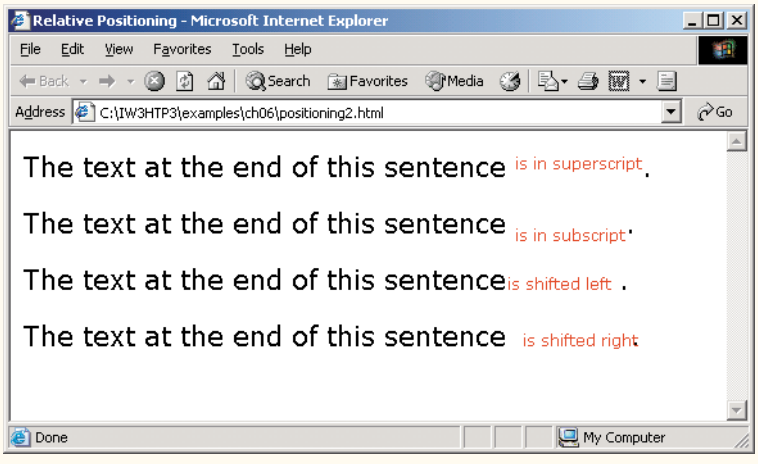

Fig. 6.9 Relative positioning of elements with CSS. (Part 2 of 2.)

Setting the `position` property to `relative`, as in class `super` (lines 21–22), lays out the element on the page and offsets it by the specified `top`, `bottom`, `left` or `right` value. Unlike absolute positioning, relative positioning keeps elements in the general flow of elements on the page, so positioning is relative to other elements in the flow. Recall that `ex` (line 22) is the *x*-height of a font, a relative length measurement typically equal to the height of a lowercase *x*.

We introduce the **`span`** element in line 39. Element `span` is a **grouping element**—it does not apply any inherent formatting to its contents. Its primary purpose is to apply CSS rules or `id` attributes to a block of text. Element `span` is an **inline-level element**—it is displayed inline with other text and with no line breaks. Lines 17–19 define the CSS rule for `span`. A similar element is the **`div`** element, which also applies no inherent styles but is displayed on its own line, with margins above and below (a block-level element).

Common Programming Error 6.2

Because relative positioning keeps elements in the flow of text in your documents, be careful to avoid unintentionally overlapping text.

6.8 Backgrounds

CSS provides control over the element backgrounds. In previous examples, we introduced the `background-color` property. CSS also can add background images to documents. Figure 6.10 adds a corporate logo to the bottom-right corner of the document. This logo stays fixed in the corner even when the user scrolls up or down the screen.

```
<?xml version = "1.0"?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.1//EN"
   "http://www.w3.org/TR/xhtml11/DTD/xhtml11.dtd">

<!-- Fig. 6.10: background.html                -->
<!-- Adding background images and indentation -->

<html xmlns = "http://www.w3 .org/1999/xhtml">
   <head>
      <title>Background Images</title>

      <style type = "text/css">

         body  { background-image: url(logo.gif);
                 background-position: bottom right;
                 background-repeat: no-repeat;
                 background-attachment: fixed; }

         p     { font-size: 18pt;
                 color: #aa5588;
                 text-indent: 1em;
                 font-family: arial, sans-serif; }

         .dark { font-weight: bold }

      </style>
   </head>

   <body>

      <p>
      This example uses the background-image,
      background-position and background-attachment
      styles to place the <span class = "dark">Deitel
      &amp;amp; Associates, Inc.</span> logo in the bottom,
      right corner of the page. Notice how the logo
      stays in the proper position when you resize the
      browser window.
      </p>

```

Fig. 6.10 Background image added with CSS. (Part 1 of 2.)

```
41      </body>
42  </html>
```

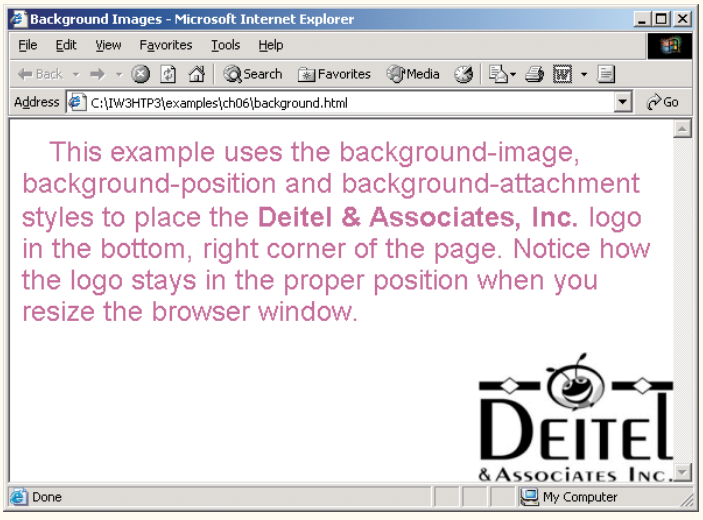

Fig. 6.10 Background image added with CSS. (Part 2 of 2.)

The **background-image** property (line 14) specifies the image URL for the image `logo.gif` in the format `url(` *fileLocation* `)`. The Web-page author also can set the `background-color` property in case the image is not found.

The **background-position** property (line 15) places the image on the page. The keywords **top**, **bottom**, **center**, **left** and **right** are used individually or in combination for vertical and horizontal positioning. An image can be positioned using lengths by specifying the horizontal length followed by the vertical length. For example, to position the image as horizontally centered (positioned at 50% of the distance across the screen) and 30 pixels from the top, use

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

The **background-repeat** property (line 16) controls the **tiling** of the background image. Tiling places multiple copies of the image next to each other to fill the background. Here, we set the tiling to **no-repeat** to display only one copy of the background image. The `background-repeat` property can be set to **repeat** (the default) to tile the image vertically and horizontally, **repeat-x** to tile the image only horizontally or **repeat-y** to tile the image only vertically.

The final property setting, **background-attachment: fixed** (line 17), fixes the image in the position specified by `background-position`. Scrolling the browser window will not move the image from its position. The default value, **scroll**, moves the image as the user scrolls through the document.

Line 21 indents the first line of text in the element by the specified amount, in this case `1em`. An author might use this property to create a Web page that reads more like a novel, in which the first line of every paragraph is indented.

Line 24 uses the **font-weight** property to specify the "boldness" of text. Possible values are **bold**, **normal** (the default), **bolder** (bolder than bold text) and **lighter** (lighter than normal text). Boldness also can be specified with multiples of 100, from 100 to 900 (e.g., 100, 200, …, 900). Text specified as normal is equivalent to 400, and bold text is equivalent to 700. However, many systems do not have fonts that can scale with this level of precision, so using the values from 100 to 900 might not display the desired effect.

Another CSS property that formats text is the **font-style** property, which allows the developer to set text to **none**, **italic** or **oblique** (oblique will default to italic if the system does not support oblique text).

6.9 Element Dimensions

In addition to positioning elements, CSS rules can specify the actual dimensions of each page element. Figure 6.11 demonstrates how to set the dimensions of elements.

```
1  <?xml version = "1.0"?>
2  <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.1//EN"
3     "http://www.w3.org/TR/xhtml11/DTD/xhtml11.dtd">
4
5  <!-- Fig. 6.11: width.html                      -->
6  <!-- Setting box dimensions and aligning text  -->
7
8  <html xmlns = "http://www.w3.org/1999/xhtml">
9     <head>
10       <title>Box Dimensions</title>
11
12       <style type = "text/css">
13
14          div { background-color: #ffccff;
15                margin-bottom: .5em }
16       </style>
17
18    </head>
19
20    <body>
21
22       <div style = "width: 20%">Here is some
23       text that goes in a box which is
24       set to stretch across twenty percent
25       of the width of the screen.</div>
26
27       <div style = "width: 80%; text-align: center">
28       Here is some CENTERED text that goes in a box
29       which is set to stretch across eighty percent of
30       the width of the screen.</div>
31
32       <div style = "width: 20%; height: 30%; overflow: scroll">
33       This box is only twenty percent of
34       the width and thirty percent of the height.
35       What do we do if it overflows? Set the
```

Fig. 6.11 Element dimensions and text alignment. (Part 1 of 2.)

```
36          overflow property to scroll!</div>
37
38       </body>
39    </html>
```

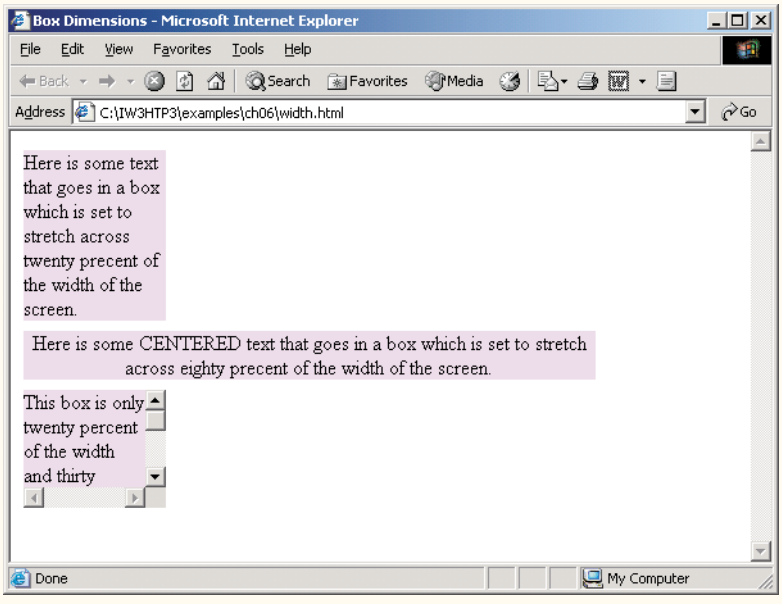

Fig. 6.11 Element dimensions and text alignment. (Part 2 of 2.)

The inline style in line 22 illustrates how to set the **`width`** of an element on screen; here, we indicate that the `div` element should occupy 20% of the screen width. Most elements are left-aligned by default; however, this alignment can be altered to position the element elsewhere. The height of an element can be set similarly, using the **`height`** property. The `height` and `width` values also can be specified as relative or absolute lengths. For example

```
width: 10em
```

sets the element's width to be equal to 10 times the font size. Line 27 sets text in the element to be **`center`** aligned; other values for the `text-align` property include **`left`** and **`right`**.

One problem with setting both dimensions of an element is that the content inside the element can exceed the set boundaries, in which case the element is simply made large enough for all the content to fit. However, in line 32, we set the **`overflow`** property to **`scroll`**, a setting that adds scrollbars if the text overflows the boundaries.

6.10 Text Flow and the Box Model

A browser normally places text and elements on screen in the order in which they appear in the XHTML document. However, as we have seen with absolute positioning, it is possible to remove elements from the normal flow of text. **Floating** allows you to move an element to one side of the screen; other content in the document then flows around the floated

element. In addition, each block-level element has a virtual box drawn around it, based on what is known as the **box model**. The properties of this box can be adjusted to control the amount of padding inside the element and the margins outside the element (Fig. 6.12).

```
<?xml version = "1.0"?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.1//EN"
   "http://www.w3.org/TR/xhtml11/DTD/xhtml11.dtd">

<!-- Fig. 6.12: floating.html            -->
<!-- Floating elements and element boxes -->

<html xmlns = "http://www.w3.org/1999/xhtml">
   <head>
      <title>Flowing Text Around Floating Elements</title>

      <style type = "text/css">

         div { background-color: #ffccff;
               margin-bottom: .5em;
               font-size: 1.5em;
               width: 50% }

         p   { text-align: justify }

      </style>

   </head>

   <body>

      <div style = "text-align: center">
         Deitel &amp; Associates, Inc.</div>

      <div style = "float: right; margin: .5em;
         text-align: right">
         Corporate Training and Publishing</div>

      <p>Deitel &amp; Associates, Inc. is an internationally
      recognized corporate training and publishing organization
      specializing in programming languages, Internet/World
      Wide Web technology and object technology education.
      The company provides courses on Java, C++, Visual Basic, C,
      Internet and World Wide Web programming, and Object Technology.</p>

      <div style = "float: right; padding: .5em;
         text-align: right">
         Leading-Edge Programming Textbooks</div>

      <p>The company's clients include many Fortune 1000
      companies, government agencies, branches of the military
      and business organizations.</p>

```

Fig. 6.12 Floating elements, aligning text and setting box dimensions. (Part 1 of 2.)

```
49          <p style = "clear: right">Through its publishing
50          partnership with Prentice Hall, Deitel &amp;amp; Associates,
51          Inc. publishes leading-edge programming textbooks,
52          professional books, interactive CD-ROM-based multimedia
53          Cyber Classrooms, satellite courses and World Wide Web
54          courses.</p>
55
56       </body>
57    </html>
```

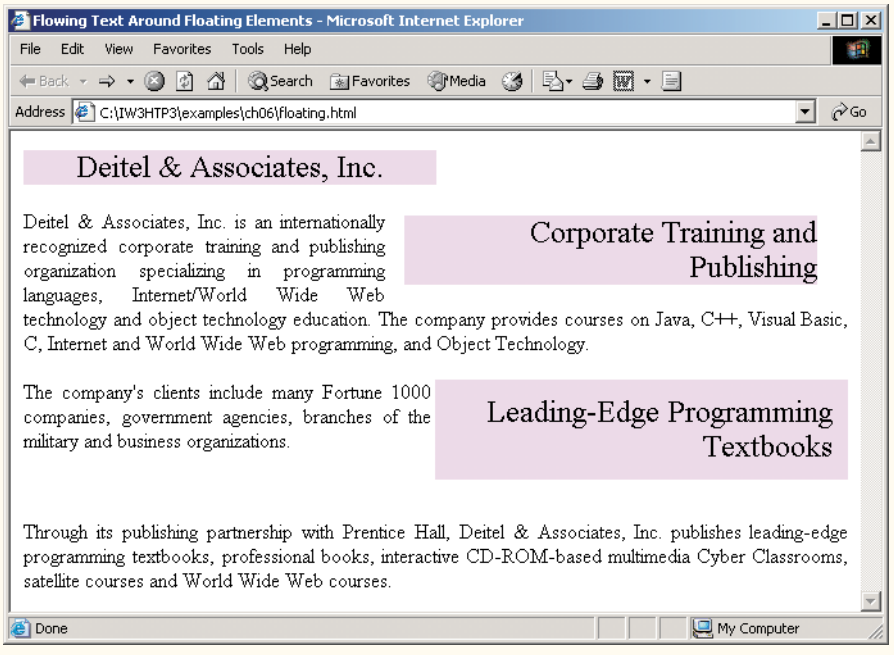

Fig. 6.12 Floating elements, aligning text and setting box dimensions. (Part 2 of 2.)

In addition to text, whole elements can be **floated** to the left or right of content. This means that any nearby text will wrap around the floated element. For example, in lines 30–32 we float a `div` element to the `right` side of the screen. As you can see from the sample screen capture, the text from lines 34–39 flows cleanly to the left and underneath the `div` element.

The second property in line 30, `margin`, specifies the distance between the edge of the element and any other element on the page. When the browser renders elements using the box model, the content of each element is surrounded by **padding**, a **border** and a **margin** (Fig. 6.13).

Margins for individual sides of an element can be specified by using the properties **`margin-top`**, **`margin-right`**, **`margin-left`** and **`margin-bottom`**.

Lines 41–43 specify a `div` element that floats at the right side of the content. Property **`padding`** for the `div` element is set to `.5em` (half a line's height). **Padding** is the distance between the content inside an element and the element's border. Like the `margin`, the `padding` can be set for each side of the box, with **`padding-top`**, **`padding-right`**, **`padding-left`** and **`padding-bottom`**.

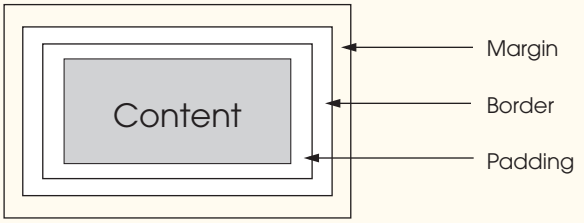

Fig. 6.13 Box model for block-level elements.

Lines 49–54 show that you can interrupt the flow of text around a `floated` element by setting the `clear` property to the same direction as that in which the element is `floated`—`right` or `left`. Notice in the screen capture that the text from lines 45–47 flows to the left of the `floated` `div` element (lines 43–45), but the text from lines 49–54 does not. Setting the **`clear`** property to **`all`** interrupts the flow on both sides of the document.

Another property of every block-level element on screen is the border, which lies between the padding space and the margin space, and has numerous properties for adjusting its appearance, as shown in Fig. 6.14.

```
1  <?xml version = "1.0"?>
2  <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.1//EN"
3     "http://www.w3.org/TR/xhtml11/DTD/xhtml11.dtd">
4
5  <!-- Fig. 6.14: borders.html       -->
6  <!-- Setting borders of an element -->
7
8  <html xmlns = "http://www.w3.org/1999/xhtml">
9     <head>
10       <title>Borders</title>
11
12       <style type = "text/css">
13
14          body    { background-color: #ccffcc }
15
16          div     { text-align: center;
17                    margin-bottom: 1em;
18                    padding: .5em }
19
20          .thick  { border-width: thick }
21
22          .medium { border-width: medium }
23
24          .thin   { border-width: thin }
25
26          .groove { border-style: groove }
27
28          .inset  { border-style: inset }
29
```

Fig. 6.14 Borders of block-level elements. (Part 1 of 2.)

```
30            .outset { border-style: outset }
31
32            .red    { border-color: red }
33
34            .blue   { border-color: blue }
35
36         </style>
37      </head>
38
39      <body>
40
41         <div class = "thick groove">This text has a border</div>
42         <div class = "medium groove">This text has a border</div>
43         <div class = "thin groove">This text has a border</div>
44
45         <p class = "thin red inset">A thin red line...</p>
46         <p class = "medium blue outset">
47            And a thicker blue line</p>
48
49      </body>
50   </html>
```

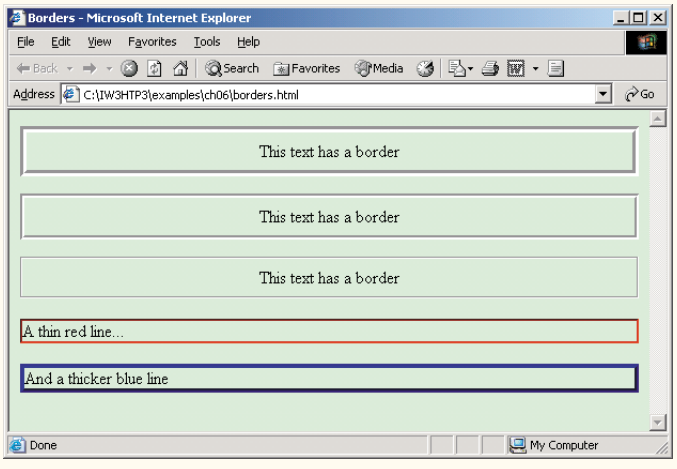

Fig. 6.14 Borders of block-level elements. (Part 2 of 2.)

In this example, we set three properties—**`border-width`**, **`border-color`** and **`border-style`**. The `border-width` property may be set to any valid CSS length or to the predefined value of **`thin`**, **`medium`** or **`thick`**. The `border-color` property sets the color. [*Note*: This property has different meanings for different style borders.]

As with `padding` and `margin`s, each of the border properties may be set for an individual side of the box (e.g., `border-top-style` or `border-left-color`). A developer can assign more than one class to an XHTML element by using the `class` attribute, as shown in line 41.

The `border-style`s are **`none`**, **`hidden`**, **`dotted`**, **`dashed`**, **`solid`**, **`double`**, **`groove`**, **`ridge`**, **`inset`** and **`outset`**. Borders `groove` and `ridge` have opposite effects, as do `inset` and `outset`. Figure 6.15 illustrates these border styles.

```
<?xml version = "1.0"?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.1//EN"
   "http://www.w3.org/TR/xhtml11/DTD/xhtml11.dtd">

<!-- Fig. 6.15: borders2.html    -->
<!-- Various border-styles       -->

<html xmlns = "http://www.w3.org/1999/xhtml">
   <head>
      <title>Borders</title>

      <style type = "text/css">

         body     { background-color: #ccffcc }

         div      { text-align: center;
                    margin-bottom: .3em;
                    width: 50%;
                    position: relative;
                    left: 25%;
                    padding: .3em }
      </style>
   </head>

   <body>

      <div style = "border-style: solid">Solid border</div>
      <div style = "border-style: double">Double border</div>
      <div style = "border-style: groove">Groove border</div>
      <div style = "border-style: ridge">Ridge border</div>
      <div style = "border-style: inset">Inset border</div>
      <div style = "border-style: outset">Outset border</div>

   </body>
</html>
```

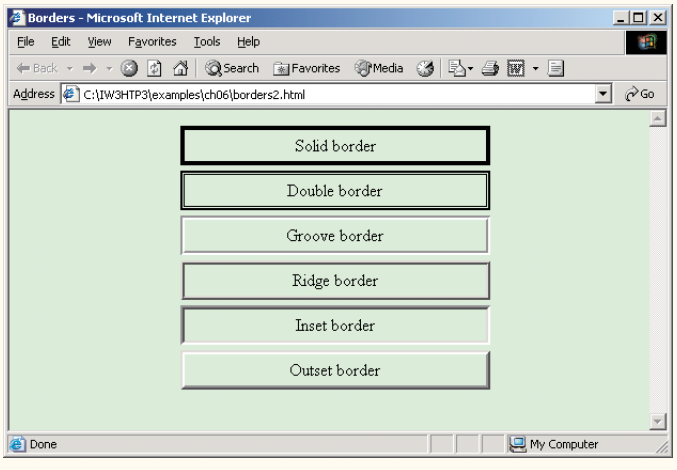

Fig. 6.15 `border-style` property of the box model.

6.11 User Style Sheets

Users can define their own **user style sheets** to format pages based on their preferences. For example, people with visual impairments may want to increase the page's text size. Web-page authors need to be careful not to inadvertently override user preferences with defined styles. This section discusses possible conflicts between **author styles** and **user styles**.

Figure 6.16 contains an author style. The `font-size` is set to `9pt` for all <p> tags that have class `note` applied to them.

```
<?xml version = "1.0"?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.1//EN"
   "http://www.w3.org/TR/xhtml11/DTD/xhtml11.dtd">

<!-- Fig. 6.16: user_absolute.html    -->
<!-- User styles                      -->

<html xmlns = "http://www.w3.org/1999/xhtml">
   <head>
      <title>User Styles</title>

      <style type = "text/css">

         .note { font-size: 9pt }

      </style>
   </head>

   <body>

      <p>Thanks for visiting my Web site. I hope you enjoy it.
      </p><p class = "note">Please Note: This site will be
      moving soon. Please check periodically for updates.</p>

   </body>
</html>
```

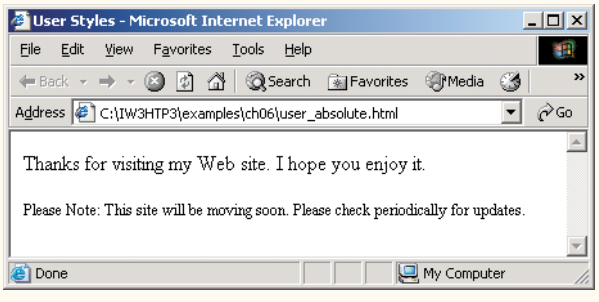

Fig. 6.16 `pt` measurement for text size.

User style sheets are external style sheets. Figure 6.17 shows a user style sheet that sets the `body`'s `font-size` to `20pt`, `color` to `yellow` and `background-color` to `#000080`.

```
/* Fig. 6.17: userstyles.css */
/* A user stylesheet         */

body     { font-size: 20pt;
           color: yellow;
           background-color: #000080 }
```

Fig. 6.17 User style sheet.

User style sheets are not `link`ed to a document; rather, they are set in the browser's options. To add a user style sheet in Internet Explorer 6, select **Internet Options...**, located in the **Tools** menu. In the **Internet Options** dialog (Fig. 6.18) that appears, click **Accessibility...**, check the **Format documents using my style sheet** checkbox, and type the location of the user style sheet. Internet Explorer 6 applies the user style sheet to any document it loads.

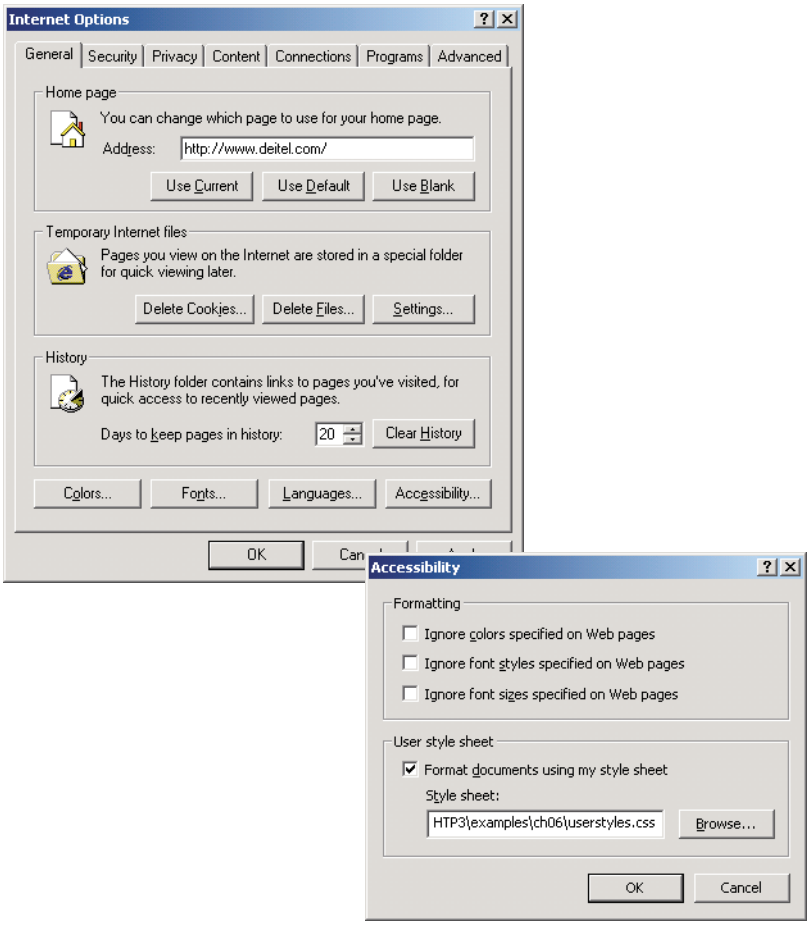

Fig. 6.18 User style sheet in Internet Explorer 6.

The Web page from Fig. 6.16 is displayed in Fig. 6.19, with the user style sheet from Fig. 6.17 applied.

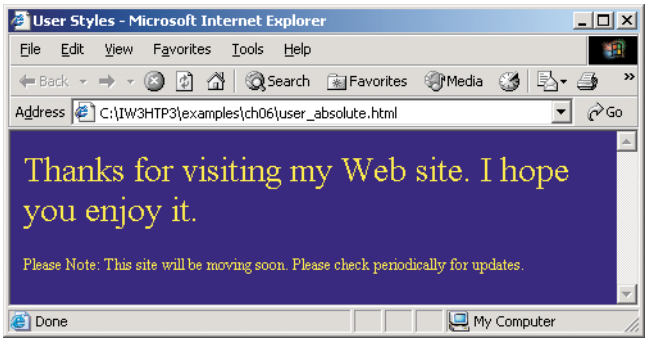

Fig. 6.19 User style sheet applied with `pt` measurement.

In this example, if users define their own `font-size` in a user style sheet, the author style has a higher precedence and overrides the user style. The `9pt` font specified in the author style sheet overrides the `20pt` font specified in the user style sheet. This small font may make pages difficult to read, especially for individuals with visual impairments. A developer can avoid this problem by using relative measurements (e.g., `em` or `ex`) instead of absolute measurements, such as `pt`. Figure 6.20 changes the `font-size` property to use a relative measurement (line 14) that does not override the user style set in Fig. 6.17. Instead, the font size displayed is relative to the one specified in the user style sheet. In this case, text enclosed in the `<p>` tag displays as `20pt` ,and `<p>` tags that have class `note` applied to them are displayed in `15pt` (`.75` times `20pt`).

```
1  <?xml version = "1.0"?>
2  <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.1//EN"
3     "http://www.w3.org/TR/xhtml11/DTD/xhtml11.dtd">
4
5  <!-- Fig. 6.20: user_relative.html    -->
6  <!-- User styles                      -->
7
8  <html xmlns = "http://www.w3.org/1999/xhtml">
9     <head>
10       <title>User Styles</title>
11
12       <style type = "text/css">
13
14          .note { font-size: .75em }
15
16       </style>
17    </head>
18
19    <body>
20
```

Fig. 6.20 `em` measurement for text size. (Part 1 of 2.)

```
21          <p>Thanks for visiting my Web site. I hope you enjoy it.
22          </p><p class = "note">Please Note: This site will be
23          moving soon. Please check periodically for updates.</p>
24
25       </body>
26    </html>
```

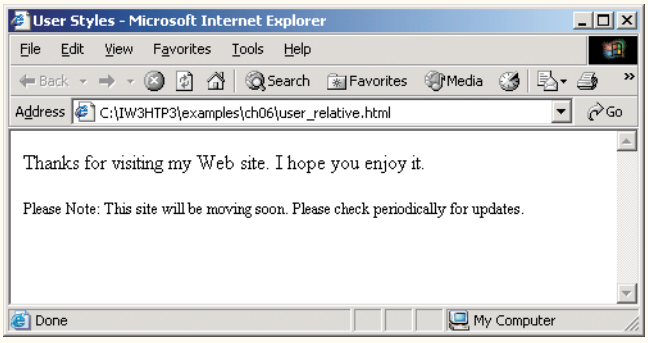

Fig. 6.20 `em` measurement for text size. (Part 2 of 2.)

Figure 6.21 displays the Web page from Fig. 6.20 with the user style sheet from Fig. 6.16 applied. Note that the second line of text displayed is larger than the same line of text in Fig. 6.19.

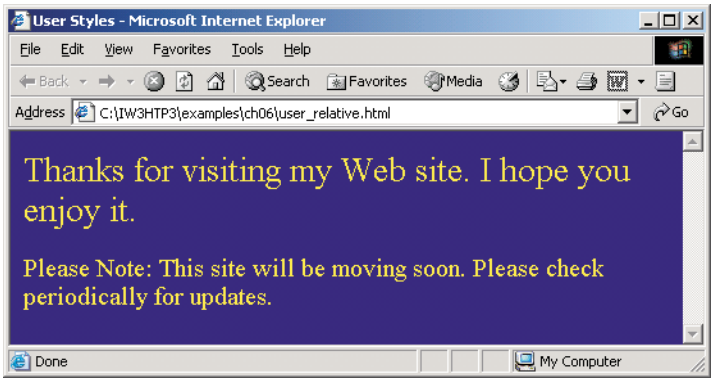

Fig. 6.21 User style sheet applied with `em` measurement.

6.12 Web Resources

`www.w3.org/TR/css3-roadmap`
The W3C *Cascading Style Sheets, Level 3* specification contains a list of all the CSS properties. The specification also provides helpful examples detailing the use of many of the properties.

`www.ddj.com/webreview/style`
This site has several charts of CSS properties, including a list stating which browsers support what attributes and to what extent.

tech.irt.org/articles/css.htm
This site contains articles dealing with CSS.

SUMMARY

- The inline style allows a developer to declare a style for an individual element by using the `style` attribute in the element's start tag.
- Each CSS property is followed by a colon and the value of the attribute.
- The `color` property sets text color. Color names and hexadecimal codes may be used as the value.
- Styles that are placed in a `style` element apply to the entire document.
- `style` element attribute `type` specifies the MIME type (the specific encoding format) of the style sheet. Style sheets use `text/css`.
- Each rule body in a style sheet begins and ends with a curly brace (`{` and `}`).
- Style class declarations are preceded by a period and are applied to elements of the specific class.
- The CSS rules in a style sheet use the same format as inline styles: The property is followed by a colon (`:`) and the value of that property. Multiple properties are separated by semicolons (`;`).
- The `background-color` attribute specifies the background color of the element.
- The `font-family` attribute names a specific font that should be displayed. Generic font families allow authors to specify a type of font instead of a specific font, in case a browser does not support a specific font. The `font-size` property specifies the size used to render the font.
- The `class` attribute applies a style class to an element.
- Pseudoclasses give the author access to content not specifically declared in the document. The `hover` pseudoclass is activated when the user moves the mouse cursor over an element.
- The `text-decoration` property applies decorations to text within an element, such as `underline`, `overline`, `line-through` and `blink`.
- To apply rules to multiple elements, separate the elements with commas in the style sheet.
- A pixel is a relative-length measurement: It varies in size based on screen resolution. Other relative lengths are `em`, `ex` and percentages.
- The other units of measurement available in CSS are absolute-length measurements—that is, units that do not vary in size. These units can be `in` (inches), `cm` (centimeters), `mm` (millimeters), `pt` (points; 1 `pt`=1/72 `in`) or `pc` (picas; 1 `pc` = 12 `pt`).
- External linking of style sheets can create a uniform look for a Web site; separate pages can all use the same styles. Modifying a single style sheet file makes changes to styles across an entire Web site.
- `link`'s `rel` attribute specifies a relationship between two documents.
- The CSS `position` property allows absolute positioning, which provides greater control over where on a page elements reside. Specifying an element's `position` as `absolute` removes it from the normal flow of elements on the page and positions it according to distance from the `top`, `left`, `right` or `bottom` margin of its parent element.
- The `z-index` property allows a developer to layer overlapping elements. Elements that have higher `z-index` values are displayed in front of elements with lower `z-index` values.
- Unlike absolute positioning, relative positioning keeps elements in the general flow on the page and offsets them by the specified `top`, `left`, `right` or `bottom` value.
- Property `background-image` specifies the URL of the image, in the format `url(`*fileLocation*`)`. The property `background-position` places the image on the page using the values `top`, `bottom`,

`center`, `left` and `right` individually or in combination for vertical and horizontal positioning. You can also position by using lengths.

- The `background-repeat` property controls the tiling of the background image. Setting the tiling to `no-repeat` displays one copy of the background image on screen. The `background-repeat` property can be set to `repeat` (the default) to tile the image vertically and horizontally, to `repeat-x` to tile the image only horizontally or to `repeat-y` to tile the image only vertically.
- The property setting `background-attachment: fixed` fixes the image in the position specified by `background-position`. Scrolling the browser window will not move the image from its set position. The default value, `scroll`, moves the image as the user scrolls the window.
- The `text-indent` property indents the first line of text in the element by the specified amount.
- The `font-weight` property specifies the "boldness" of text. Values besides `bold` and `normal` (the default) are `bolder` (bolder than `bold` text) and `lighter` (lighter than `normal` text). The value also may be justified using multiples of 100, from 100 to 900 (i.e., `100`, `200`, …, `900`). Text specified as `normal` is equivalent to `400`, and `bold` text is equivalent to `700`.
- The `font-style` property allows the developer to set text to `none`, `italic` or `oblique` (`oblique` will default to `italic` if the system does not have a separate font file for oblique text, which is normally the case).
- `span` is a generic grouping element; it does not apply any inherent formatting to its contents. Its main use is to apply styles or `id` attributes to a block of text. Element `span` is displayed inline (an inline element) with other text and with no line breaks. A similar element is the `div` element, which also applies no inherent styles, but is displayed on a separate line, with margins above and below (a block-level element).
- The dimensions of elements on a page can be set with CSS by using properties `height` and `width`.
- Text within an element can be `centered` using `text-align`; other values for the `text-align` property are `left` and `right`.
- One problem with setting both vertical and horizontal dimensions of an element is that the content inside the element might sometimes exceed the set boundaries, in which case the element must be made large enough for all the content to fit. However, a developer can set the `overflow` property to `scroll`; this setting adds scroll bars if the text overflows the boundaries set for it.
- Browsers normally place text and elements on screen in the order in which they appear in the XHTML file. Elements can be removed from the normal flow of text. Floating allows you to move an element to one side of the screen; other content in the document will then flow around the floated element.
- CSS uses a box model to render elements on screen. The content of each element is surrounded by padding, a border and margins. The properties of this box are easily adjusted.
- The `margin` property determines the distance between the element's edge and any outside text.
- Margins for individual sides of an element can be specified by using `margin-top`, `margin-right`, `margin-left` and `margin-bottom`.
- The `padding` property determines the distance between the content inside an element and the edge of the element. Padding also can be set for each side of the box by using `padding-top`, `padding-right`, `padding-left` and `padding-bottom`.
- A developer can interrupt the flow of text around a `floated` element by setting the `clear` property to the same direction in which the element is `floated`—`right` or `left`. Setting the `clear` property to `all` interrupts the flow on both sides of the document.
- The border of a block-level element lies between the padding space and the margin space and has numerous properties with which to adjust its appearance.

- The `border-width` property may be set to any of the CSS lengths or to the predefined value of `thin`, `medium` or `thick`.
- The `border-style`s available are `none`, `hidden`, `dotted`, `dashed`, `solid`, `double`, `groove`, `ridge`, `inset` and `outset`.
- The `border-color` property sets the color used for the border.
- The `class` attribute allows more than one class to be assigned to an XHTML element by separating each class name from the next with a space.

TERMINOLOGY

absolute positioning
absolute-length measurement
`arial` font
`background` property
`background-attachment` property
`background-color` property
`background-image` property
`background-position` property
`background-repeat` property
`blink` text decoration
block-level element
border
`border-color` property
`border-style` property
`border-width` property
box model
Cascading Style Sheets (CSS)
`class` attribute
`clear` property value
`cm` (centimeter)
colon (`:`)
`color` property
CSS rule
`cursive` generic font family
`dashed` border-style
`dotted` border-style
`double` border-style
`em` (size of font)
embedded style sheet
`ex` (*x*-height of font)
floated element
`font-style` property
generic font family
`groove` border style
`hidden` border style
`href` attribute
`in` (inch)
inline style
inline-level element
`inset` border-style
`large` relative font size
`larger` relative font size
`left` property value
`line-through` text decoration
`link` element
linking to an external style sheet
`margin` property
`margin-bottom` property
`margin-left` property
`margin-right` property
`margin-top` property
`medium` relative border width
`medium` relative font size
MIME (Multipurpose Internet Mail Extensions) type
`mm` (millimeter)
`monospace` font
`none` border-style
`outset` border-style
`overflow` property
`overline` text decoration
`padding` property
parent element
`pc` (pica)
pseudoclass
`pt` (point)
`rel` attribute (`link`)
relative positioning
relative-length measurement
`repeat` property value
`ridge` border-style
`right` property value
`sans-serif` generic font family
`scroll` property value
separation of structure from content
`serif` generic font family
`small` relative font size
`smaller` relative font size
`solid` border-style
`span` element

style
`style` attribute
style class
style in header section of document
text flow
`text/css` MIME type
`text-align` property
`text-decoration` property
`text-indent` property
`thick` border width
`thin` border width
user style sheet
`x-large` relative font size
`x-small` relative font size
`xx-large` relative font size
`xx-small` relative font size
`z-index` property-

SELF-REVIEW EXERCISES

6.1 Assume that the size of the base font on a system is 12 points.

a) How big is a 36-point font in `em`s?
b) How big is a 9-point font in `em`s?
c) How big is a 24-point font in picas?
d) How big is a 12-point font in inches?
e) How big is a 1-inch font in picas?

6.2 Fill in the blanks in the following statements:

a) Using the _____ element allows authors to use external style sheets in their pages.
b) To apply a CSS rule to more than one element at a time, separate the element names with a(n) _____.
c) Pixels are a(n) _____ -length measurement unit.
d) The _____ pseudoclass is activated when the user moves the mouse cursor over the specified element.
e) Setting the `overflow` property to _____ provides a mechanism for containing inner content without compromising specified box dimensions.
f) While _____ is a generic inline element that applies no inherent formatting and _____ is a generic block-level element that applies no inherent formatting.
g) Setting property `background-repeat` to _____ tiles the specified `background-image` vertically.
h) If you `float` an element, you can stop the flowing of text by using property _____.
i) The _____ property allows you to indent the first line of text in an element.
j) Three components of the box model are the _____, _____ and _____.

ANSWERS TO SELF-REVIEW EXERCISES

6.1 a) 3 `em`s. b) 0.75 `em`s. c) 2 picas. d) 1/6 inch. e) 6 picas.

6.2 a) `link`. b) comma. c) relative. d) `hover`. e) `scroll`. f) `span`, `div`. g) `y-repeat`. h) `clear`. i) `text-indent`. j) padding, border, margin.

EXERCISES

6.3 Write a CSS rule that makes all text 1.5 times larger than the base font of the system and colors the text red.

6.4 Write a CSS rule that removes the underlines from all links inside list items (`li`) and shifts all list items left by 3 `em`s.

6.5 Write a CSS rule that places a background image halfway down the page, tiling it horizontally. The image should remain in place when the user scrolls up or down.

6.6 Write a CSS rule that gives all `h1` and `h2` elements a padding of 0.5 `em`s, a `groove`d border style and a margin of 0.5 `em`s.

6.7 Write a CSS rule that changes the color of all elements containing attribute `class = "greenMove"` to green and shifts them down 25 pixels and right 15 pixels.

6.8 Write an XHTML document that shows the results of a color survey. The document should contain a form with radio buttons that allows users to vote for their favorite color. One of the colors should be selected as a default. The document should also contain a table showing various colors and the corresponding percentage of votes for each color. (Each row should be displayed in the color to which it is referring.) Use attributes to format width, border and cell spacing for the table.

6.9 Add an embedded style sheet to the XHTML document in Fig. 4.5. The style sheet should contain a rule that displays `h1` elements in blue. In addition, create a rule that displays all links in blue without underlining them. When the mouse hovers over a link, change the link's background color to yellow.

6.10 Modify the style sheet in Fig. 6.4 by changing `a:hover` to `a:hver` and `margin-left` to `margin left`. Validate the style sheet using the CSS Validator. What happens?