Creating Special Effects with CSS

Adding Advanced Styles to a Web Site

Case | Online Scrapbooks

Scrapbooking is the popular hobby of creating albums containing photos, memorabilia, writing, and other embellishments. This hobby has become a multimillion-dollar industry with companies that specialize in scrapbooking supplies and support. One of these companies is Online Scrapbooks.

Kathy Pridham, who leads the Web development team at Online Scrapbooks, has hired you to work on the style for the new company’s Web site. The Web site’s home page will have information on how to get started in scrapbooking and links to other pages that contain a wide variety of information. Because the Web site will have so many pages, Kathy is using Cascading Style Sheets to manage the layout, design, and function of the pages. She has a style sheet providing the site’s basic layout and design. She would like you to add features such as graphical bullets, rollover effects, and drop caps. To make those enhancements, you’ll need to use some of the special features supported by CSS.

Kathy knows that many users want to access the Web site from mobile devices, while others want to be able to print some of the site’s contents. She wants the site to work with any kind of output, including mobile devices and printed output.
Session 4.1

Working with Selector Patterns

Kathy has already created a basic Web page describing how to get started in scrapbooking. She’s written an article and created the basic Web page layout using an external style sheet. She’s provided you with her HTML document, her graphic files, and her style sheet to study. Kathy suggests that this Web page would be a good place to start in your task of enhancing her basic design.

To view Kathy’s data files:

1. In your text editor, open the starttxt.htm and scrapstxt.css files, located in the tutorial.04\tutorial folder included with your Data Files. Within the comment section at the top of each file, add your name and the date in the space provided. Save the files as start.htm and scraps.css, respectively, in the same folder.

2. Take some time to review the code in both the external style sheet file and the HTML document. Note how the CSS styles are applied to specific elements in the start.htm file to create an interesting layout and design.

3. Open start.htm in your Web browser. Figure 4-1 shows the current appearance of the start.htm file.

Figure 4-1
Initial design for the Getting Started page
As you can see from Figure 4-1, Kathy has applied a layout in which the list of links floats on the left page margin and a box describing basic scrapbooking materials floats on the right margin. There is one main heading marked as an h1 element displayed at the top of page, providing a logo with the company name.

The three other headings—Getting Started, Preserving Your Memories, and Basic Materials—are marked as h2 elements. Kathy wants to apply a slightly different format to the Basic Materials h2 heading than the one applied to the Getting Started and Preserving Your Memories h2 headings. One way of applying a specific format to this heading is through the use of an id attribute. However, Kathy doesn’t want to maintain a list of id values for all the various elements on her Web page. Instead, she would like to create styles for elements based on their location or their use in the document. She asks if this can be done with CSS.

**Contextual Selectors**

So far, the only styles you’ve worked with are ones in which the style selector references either an element (or a group of elements) or an element identified by an id. For example, the style

```
b {color: blue}
```

displays all boldface text in a blue font. What would you do, however, if you didn’t want every example of boldface text to be displayed in a blue font? What if you wanted this style applied only to boldface text located within an ordered or unordered list?

Recall that on a Web page, elements are nested within other elements, forming a hierarchical tree structure. The top element on the Web page is the body element because it contains all of the content appearing in the page. From this top element, other elements descend. Figure 4-2 shows an example of such a tree structure for a Web page consisting of a few headings, a couple of paragraphs, some boldface elements, and a span element nested within a paragraph.

![A sample tree hierarchy of page elements](image)

To take advantage of this tree structure, CSS allows you to create **contextual selectors** that express the location of an element within the hierarchy of elements. The general syntax of a contextual selector has the form

```
parent descendant {styles}
```

where *parent* is the parent element, *descendant* is a descendant of the parent, and *styles* are the styles to be applied to the descendant element. To apply a blue color only to boldface text found in lists, you would use the style:

```
li b {color: blue}
```
In this case, \texttt{li} is the parent element and \texttt{b} is the descendant element (because it is contained within the list item). Any bold element not nested within a list element is not affected by this style. Note that the descendant element does not have to be a direct child of the parent element; it can appear several levels below the parent element in hierarchy. For example in the code

```html
<li>
  <span><b>Special</b>Orders</span> this month!
</li>
```

the bold element is a descendant of the list item, but it is a direct child only of the span element. So the word “Special” would appear in a bold font if the above style is applied to the document. Contextual selectors can be grouped with other selectors. The following style applies a blue font to h2 headings and to boldface list items, but nowhere else:

```css
li b, h2 {color: blue}
```

Contextual selectors can also be applied with elements marked with a specific id. The style

```css
#notes b {color: blue}
```

displays bold text in a blue font if it is nested within an element with an id of notes.

The parent/descendant form is only one example of a contextual selector. Figure 4-3 describes some of the other contextual forms supported by CSS.

**Figure 4-3**

<table>
<thead>
<tr>
<th>Selector</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>*</td>
<td>Matches any element in the hierarchy</td>
</tr>
<tr>
<td>e</td>
<td>Matches any element, ( e ), in the hierarchy</td>
</tr>
<tr>
<td>( e_1, e_2, e_3, \ldots )</td>
<td>Matches the group of elements: ( e_1, e_2, e_3, \ldots )</td>
</tr>
<tr>
<td>( e f )</td>
<td>Matches any element, ( f ), that is a descendant of an element, ( e )</td>
</tr>
<tr>
<td>( e &gt; f )</td>
<td>Matches any element, ( f ), that is a direct child of an element, ( e )</td>
</tr>
<tr>
<td>( e + f )</td>
<td>Matches any element, ( f ), that is immediately preceded by a sibling element, ( e )</td>
</tr>
</tbody>
</table>

For example, the style

```css
* {color: blue}
```

causes all of the elements in the document to appear in a blue font. On the other hand, the style

```css
p > b {color: blue}
```

applies the blue font only to boldface text that is contained within a paragraph element as a child of that element and not any descendant. Figure 4-4 provides additional examples of how to select different elements of the Web page document based on the expression in the contextual selector. Selected elements are highlighted in red for each pattern. Remember that because of style inheritance, any style applied to an element is passed down the document tree. So a style applied to a paragraph element is automatically passed down to elements contained within that paragraph unless it conflicts with a more specific style.
Simple and contextual selectors

Although the contextual selectors listed in Figure 4-3 are part of the specifications for CSS2, they are not well supported by earlier versions of the Internet Explorer browser. In particular, the \( e > f \) and \( e + f \) contextual selectors should be used with caution if you need to support Internet Explorer. Other browsers, including Firefox, Opera, and Safari, do support all of the contextual selectors described in Figure 4-3.

Attribute Selectors

On occasion you might also need to select elements based on their attribute values. For example, if you want to display link text in a blue font, you might use the following declaration:

```css
a {color: blue}
```

However, this declaration makes no distinction between `<a>` tags used to mark links and `<a>` tags used to mark document anchors (for a discussion of anchors, see Tutorial 2). HTML makes this distinction based on the presence or absence of the `href` attribute. To select an element based on the element's attributes, you can create an attribute selector that has the form

```css
element[att] {styles}
```

where `element` is a page element, `att` is the name of an attribute associated with the element, and `styles` are the styles applied to the element. The declaration

```css
a[href] {color: blue}
```

applies the blue font color style only to link elements that contain an `href` attribute. Any `<a>` tag used to mark anchors would not contain the `href` attribute, and therefore would not be affected by this style. Figure 4-5 describes some of the other attribute selectors supported by CSS.
<table>
<thead>
<tr>
<th>Selector</th>
<th>Description</th>
<th>Example</th>
<th>Interpretation</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>elem[att]</code></td>
<td>The element contains the <code>att</code> attribute</td>
<td><code>a[href]</code></td>
<td>Matches hypertext elements containing the href attribute</td>
</tr>
<tr>
<td><code>elem[att=&quot;val&quot;]</code></td>
<td>The element's <code>att</code> attribute equals <code>val</code></td>
<td><code>a[href=&quot;gloss.htm&quot;]</code></td>
<td>Matches hypertext elements whose href attribute equals &quot;gloss.htm&quot;</td>
</tr>
<tr>
<td><code>elem[att~=&quot;val&quot;]</code></td>
<td>The element's <code>att</code> attribute value is a space-separated list of words, one of which is exactly <code>val</code></td>
<td><code>a[rel=&quot;glossary&quot;]</code></td>
<td>Matches hypertext elements whose rel attribute contains the word &quot;glossary&quot;</td>
</tr>
<tr>
<td><code>elem[att=&quot;val&quot;]</code></td>
<td>The element's <code>att</code> attribute value is a hyphen-separated list of words beginning with <code>val</code></td>
<td><code>p[id=&quot;first&quot;]</code></td>
<td>Matches paragraphs whose id attribute starts with the word &quot;first&quot; in a hyphen-separated list of words</td>
</tr>
<tr>
<td>`elem[att</td>
<td>^=&quot;val&quot;]`</td>
<td>The element's <code>att</code> attribute begins with <code>val</code> (CSS3)</td>
<td><code>a[rel=&quot;prev&quot;]</code></td>
</tr>
<tr>
<td><code>elem[att$=&quot;val&quot;]</code></td>
<td>The element's <code>att</code> attribute ends with <code>val</code> (CSS3)</td>
<td><code>a[href$=&quot;org&quot;]</code></td>
<td>Matches hypertext elements whose href attribute ends with &quot;org&quot;</td>
</tr>
<tr>
<td><code>elem[att*=&quot;val&quot;]</code></td>
<td>The element's <code>att</code> attribute contains the value <code>val</code> (CSS3)</td>
<td><code>a[href*=&quot;faq&quot;]</code></td>
<td>Matches hypertext elements whose href attribute contains the text string &quot;faq&quot;</td>
</tr>
</tbody>
</table>

Browser support for attribute selectors is mixed. For this reason, you should use attribute selectors with caution. Note that some of the attribute selectors listed in Figure 4-5 are part of the proposed specifications for CSS3 and have scattered browser support at the present time. As with contextual selectors, attribute selectors enjoy good support from Firefox, Opera, and Safari, but poor support from Internet Explorer. IE does support attribute and contextual selectors if you write your HTML code to put Internet Explorer into standards mode (for a discussion of standards mode, see Tutorial 3).
Using Selector Patterns

- To apply a style to all elements in the document, use the * selector.
- To apply a style to a single element, use the e selector, where e is the name of the element.
- To apply a selector to a descendant element, f, use the e f selector, where e is the name of the parent element and f is an element nested within the parent.
- To apply a selector to a child element, f, use the e > f selector, where e is the name of a parent element and f is an element that is a direct child of the parent.
- To apply a selector to a sibling element, use the e + f selector, where e and f are siblings and f immediately follows e in the document tree.

Applying a Selector Pattern

After discussing how to use selector patterns, you and Kathy decide to apply them to her Getting Started document. You decide to create a style for the h2 heading in the Basic Materials Scrapbooking box so that you can use the style in similar boxes on other pages in the Online Scrapbooking site. You’ll center this heading, change the background color to white, reduce the top margin to 0 pixels, and add a solid orange border to the bottom of the element. Because this heading appears within a div element that is identified with an id value of pullout, you’ll add the following style to the style sheet:

```
#pullout h2 {text-align: center; background-color: white; margin-top: 0px; border-bottom: 2px solid orange}
```

Add this style declaration to the scraps.css file.

To add a contextual selector to the style sheet:

1. Go to the scraps.css file in your text editor.
2. Directly below the style for the #pullout selector, insert the following style, as shown in Figure 4-6:

```
#pullout h2 {text-align: center; background-color: white; margin-top: 0px; border-bottom: 2px solid orange}
```

Using a contextual selector

Copyright 2009 Cengage Learning, Inc. All Rights Reserved. May not be copied, scanned, or duplicated, in whole or in part.
3. Save your changes to the file and then reload `start.htm` in your Web browser. Figure 4-7 shows the revised appearance of the document.

As shown in Figure 4-7, the appearance of the h2 element nested within the Basic Materials box has been modified under the new style, but h2 elements located elsewhere in the document retain their original appearance.
Conditional Comments and Internet Explorer

Several versions of the Internet Explorer browser don't follow all of the CSS specifications for selectors and styles. You can correct many of the incompatibilities by converting your HTML code to XHTML and putting IE into standards mode rather than quirks mode. However, this might not work for older versions of Internet Explorer, such as IE5 and even IE6. For those browser versions, you can use *conditional comments* that allow you to apply different HTML code for different versions of Internet Explorer. The general syntax of a conditional comment is

```
<!--[if condition IE version]>-->

 HTML code

 <![endif]-->
```

where *condition* is a condition that is either true or false, *version* is the version number of an IE browser, and *HTML code* is code that will be run if *condition* is true. For example, the code

```
<!--[if lt IE 6]>-->

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

 <![endif]-->
```

links the Web page to the old.css style sheet file, but only if the browser version in use is older than Internet Explorer 6. In this case, the *condition* value is lt for “less than.” Other *condition* values include lte (less than or equal to), gt (greater than), gte (greater than or equal to), and ! (not equal to). If you specify no *condition* value, the *HTML code* will be run only for the specified version of Internet Explorer. You can also leave off the version number to apply the HTML code to Internet Explorer but not to other browsers. So the code

```
<!--[if IE]>-->

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

 <![endif]-->
```

links the file to the ie_styles.css style sheet file, but only if Internet Explorer is being used. Conditional comments are one of the best ways you can tailor your HTML code to match the capabilities of different versions of Internet Explorer and other browsers.

Applying Styles to Lists

Kathy has her Web page links in an unordered list that is displayed in a box floated on the left page margin. Like all unordered lists, the browser displays the items in this list with bullet markers. Kathy would like to remove the bullet markers from this list. To remove the markers you can apply one of the many CSS list styles.

Choosing a List Style Type

To specify the list marker displayed by the browser, you can apply the style

```
list-style-type: type
```

where *type* is one of the markers shown in Figure 4-8.
List style types

<table>
<thead>
<tr>
<th>list-style-type</th>
<th>Marker (s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>disc</td>
<td>●</td>
</tr>
<tr>
<td>circle</td>
<td>○</td>
</tr>
<tr>
<td>square</td>
<td>□</td>
</tr>
<tr>
<td>decimal</td>
<td>1, 2, 3, 4, ...</td>
</tr>
<tr>
<td>decimal-leading-zero</td>
<td>01, 02, 03, 04, ...</td>
</tr>
<tr>
<td>lower-roman</td>
<td>i, ii, iii, iv, ...</td>
</tr>
<tr>
<td>upper-roman</td>
<td>I, II, III, IV, ...</td>
</tr>
<tr>
<td>lower-alpha</td>
<td>a, b, c, d, ...</td>
</tr>
<tr>
<td>upper-alpha</td>
<td>A, B, C, D, ...</td>
</tr>
<tr>
<td>none</td>
<td>no marker displayed</td>
</tr>
</tbody>
</table>

For example, to create a list with alphabetical markers such as

A. Home
B. Getting Started
C. Scrapbooking Tips
D. Supply List

you would apply the following list style to the ol list element:

```css
ol {list-style-type: upper-alpha}
```

List style types can be used with contextual selectors to create an outline style for several levels of nested lists. Figure 4-9 shows an example in which several levels of list style markers are used in formatting an outline. Note that each marker style is determined by the location of each ordered list within the levels of the outline. The top level is displayed with uppercase Roman numerals; the bottom level, nested within three other ordered lists, uses lowercase letters for markers.
Creating an outline style

If you don’t find the marker you want from the list-style-type style, you can supply your own in a graphic image file. To use a graphic image for the list marker, use the style

```
list-style-image: url(url)
```

where (url) is the URL of the graphic image file. The style

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

displays items in an unordered list marked with the graphic image in the redball.gif file.

Kathy wants her list of links to appear without any bullet marker, but she wants the list of basic materials to appear with a bullet marker based on one of her graphic image files. She suggests that you use both the list-style-type and list-style-image attributes to modify the appearance of the two lists. To differentiate between the two lists, you’ll use contextual selectors. The list of links is an unordered list nested within a div container with the id named links, while the list of basic materials is nested within the pullout div box.

**To apply a list style to Kathy’s list of links:**

1. Return to the `scraps.css` file in your text editor.
2. Directly below the style for the `#links` selector, enter:
   ```
   #links ul  {list-style-type: none}
   ```
3. Directly below the style for the `#pullout h2` selector, enter:
   ```
   #pullout ul {list-style-image: url(bullet.jpg)}
   ```

Figure 4-10 shows the revised code in the style sheet.
4. Save your changes to the file, and then refresh **start.htm** in your Web browser. Figure 4-11 shows the revised appearance of the two lists in the document.

The bullet markers have been removed from the list of links and have been replaced by blue arrows in the Basic Materials list.
Defining the List Position and Layout

Kathy likes the revised markers, but she thinks there’s too much empty space to the left of the lists. She would like you to modify the layout to remove the extra space. As you learned in Tutorial 1, each list is treated as a block-level element. By default, most browsers place the list marker to the left of this block, lining up the markers with each list item. You can change this default behavior by using the style

```
list-style-position: position
```

where position is either “outside” (the default) or “inside.” Placing the marker inside of the block causes the list text to flow around the marker. Figure 4-12 shows how the list-style-position affects the appearance of a bulleted list.

By specifying “inside” for the list-style-position value, you force both the list text and the list marker to be displayed inside of the block. With the addition of the list marker, you will have less space available for the list text.

When a browser renders a list, it offsets the list text a certain distance from the bullet marker. At this time there is no style for specifying the space between the list marker and the list text. The browser also indents the entire list a certain distance from other elements on the page. There is no commonly accepted value for the length that the entire list is indented. Browsers indent lists by setting a value for either the list’s left margin or left padding. Firefox indents the list by setting the left padding value, while Opera and Internet Explorer set the size of the left margin. So to have a consistent layout across all browsers, you need to set a value for both the left padding and the left margin.

The internal style sheets for Explorer and Opera set the left margin size to 40 pixels or about 2.5 em and set the left padding space to 0 pixels. Firefox’s internal style sheet does the opposite, setting the size of the left margin to 0 pixels and the left padding space to 40 pixels or 2.5 em. So if you want to reduce the indent applied by the browser, you should choose a style that reduces the sum of the left margin and left padding spaces to less than 40 pixels or 2.5 em. Finding the right combination of left padding and left margin values is often a matter of trial and error; you’ll have to test your choices under different browsers and different resolutions.
Applying List Styles

- To define appearance of the list marker, use the style
  \[\text{list-style-type}: \text{type}\]
  where \text{type} is disc, circle, square, decimal, decimal-leading-zero, lower-roman, upper-roman, lower-alpha, upper-alpha, or none.
- To insert a graphic image as a list marker, use the style
  \[\text{list-style-image}: \text{url(url)}\]
  where \text{url} is the URL of the graphic image file.
- To set the position of the list marker, use the style
  \[\text{list-style-position}: \text{position}\]
  where \text{position} is inside or outside.
- To define all of the list style properties in a single style, use
  \[\text{list-style}: \text{type} \text{url(url)} \text{position}\]

After some work, you decide to indent Kathy’s list of links by 15 pixels and the Basic Materials list by 25 pixels. You’ll add these styles to the scraps.css external style sheet.

To change the margins and padding for the two lists:

1. Return to the \textit{scraps.css} file in your text editor.
2. Add the following style to the style declaration for the \#links ul selector:
   \[\text{margin-left}: 15\text{px}; \text{padding-left}: 0\text{px}\]
3. Add the following to the style declaration for the \#pullout ul selector:
   \[\text{margin-left}: 25\text{px}; \text{padding-left}: 0\text{px}\]
   Figure 4-13 shows the revised style code.

You can combine all of the CSS styles for lists into a single style attribute. The syntax of this combined style is

\[\text{list-style}: \text{type} \text{url(url)} \text{position}\]

where \text{type} is one of the CSS marker types, \text{url} is the location of a graphic file containing a marker image, and \text{position} is the position of the list markers relative to the containing box.
Nongraphical browsers use the marker defined by the type value, while graphical browsers use the image from the graphic file. For example, the style

ul {list-style: circle url(dot.gif) inside}

displays unordered lists using the marker stored in the dot.gif file; unless a nongraphical browser is displaying the page, in which case the circle marker is applied. In both cases, the marker will be displayed on the inside of the box surrounding the list.

Working with Classes

The list of links on the Getting Started page covers three main areas: pages that teach scrapbooking, pages that sell products, and pages that provide information about the company. Although Kathy has ordered the links by area, the sections are not separated visually on the rendered page. Kathy suggests that you increase the space between the three groups so it's clear where one group ends and another starts. One method for doing this is to mark the first link in each group, and then to increase the size of the margin above those links. You can mark those links using the class attribute. The class attribute is used when you want to identify elements that share a common characteristic. It has the syntax

<elem class="class"> ... </elem>

where elem is an element in the body of the Web page and class is a name that identifies the class of objects to which the element belongs. The HTML code

<h2 class="subtitle">Getting Started</h2>
<h2 class="subtitle">Preserving Your Memories</h2>

marks both of the h2 headings—Getting Started and Preserving Your Memories—as belonging to the subtitle class. Note that unlike the id attribute, several elements can share the same class value. The class values need not be assigned to the same type of element. You can, for example, also mark h3 headings and address elements as belonging to the subtitle class if it suits your purpose. Also, unlike the id attribute, you can place several class values in a space-separated list in the class attribute. The h2 element

<h2 class="subtitle mainpage">Preserving Your Memories</h2>

belongs to both the subtitle and the mainpage classes.

The advantage of the class attribute is that you can use it to assign the same style to multiple elements sharing the same class value. The selector for the class attribute is

.class {styles}

where class is the name of the class and styles are the styles applied to that class of element. So to display all elements belonging to the subtitle class in a blue font, you could apply the following style:

.subtitle {color: blue}

Because the same class name can be used with elements of different types, you might need to specify exactly which elements of a particular class receive a defined style. This is done using the selector

elem.class {styles}

where elem is the element and class is the class. The style

h2.subtitle {color: blue}
applies a blue font to elements of the subtitle class, but only if they are h2 headings. You can also use class selectors with other selectors in more complicated expressions. The style

```css
blockquote h2.subtitle {color: blue}
```

applies the blue font color only to h2 headings of the subtitle class nested within a blockquote element.

Reference Window

**Applying a Style to an Element Class**

- To assign an element to a class, add the attribute
  ```html
class="class"
  
  ```
to the element's markup tag, where `class` is the name of the class.
- To apply a style to a class of elements, use the selector
  ```html
  .class
  
  ```
  where `class` is the name of the class.
- To apply a style to an element of a particular class, use the selector
  ```html
  elem.class
  
  ```
  where `elem` is the name of the element and `class` is the name of the class.

Now that you’ve seen how to create and apply a style to an element class, you can create a style for the list of links on the Getting Started page. The three links that indicate the start of a new link group are named Home, Online Store, and About Us. You need to mark these as belonging to the newGroup class and then apply a style that increases the top margin of these elements.

To create a style for a class of elements:

1. Go to the `start.htm` file in your text editor.
2. Locate the div element containing the list of links, and then insert the attribute
   ```html
   class="newGroup"
   
   ```
in the opening `<li>` tag for the Home, Online Store, and About Us links. See Figure 4-14.

3. Save your changes to the file.

Next you’ll go to the style sheet and create a style for the class of newGroup elements.
4. Return to the `scraps.css` file in your text editor. Directly below the style for the #links ul selector, insert the following style:

```css
#links li.newGroup {margin-top: 15px}
```

Note that the selector includes both the element name and the class name to make it clear that elements receive the margin-top style. Figure 4-15 shows the revised style code.

5. Save your changes to the file and then refresh `start.htm` in your Web browser. As shown in Figure 4-16, the list of links is now divided into three topical areas.

Kathy likes the layout of the list of links. Now she wants to focus on some design elements to enhance the user's interaction with those links.

### Using Pseudo-Classes and Pseudo-Elements

Although she realizes that most browsers underline linked text by default, Kathy thinks that a large block of underlined text is difficult to read. She's seen sites in which links are underlined only when the mouse pointer hovers over the linked text. This type of effect is called a **rollover effect** because it is applied only when a user “rolls” the mouse pointer over an element. She would like you to make underlining a rollover effect for the list of links.
Creating a Link Rollover

Rollover effects for links can be created using pseudo-classes. A **pseudo-class** is a classification of an element based on its current status, position, or use in the document. For example, one pseudo-class indicates whether a link has been previously visited by the user. Another pseudo-class indicates whether a link is currently being activated or clicked. To create a style for a pseudo-class, use the style

```
selector:pseudo-class {styles}
```

where selector is an element or group of elements within a document, pseudo-class is the name of a pseudo-class, and styles are the styles you want to apply. Figure 4-17 lists some of the pseudo-classes supported by CSS.

### Figure 4-17  Pseudo-classes

<table>
<thead>
<tr>
<th>Pseudo-class</th>
<th>Description</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>link</td>
<td>The link has not yet been visited by the user</td>
<td>a:link {color: red}</td>
</tr>
<tr>
<td>visited</td>
<td>The link has been visited by the user</td>
<td>a:visited {color: green}</td>
</tr>
<tr>
<td>active</td>
<td>The link is in the process of being activated by the user</td>
<td>a:active {color: yellow}</td>
</tr>
<tr>
<td>hover</td>
<td>The mouse pointer is hovering over the link</td>
<td>a:hover {color: blue}</td>
</tr>
<tr>
<td>focus</td>
<td>The element has received the focus of the keyboard or mouse pointer</td>
<td>input.focus {background-color: yellow}</td>
</tr>
<tr>
<td>first-child</td>
<td>The element is the first child of its parent</td>
<td>p:first-child {text-indent: 0}</td>
</tr>
<tr>
<td>lang</td>
<td>Specifies the language to be used with the element</td>
<td>q:lang(FR) {quotes: '&lt;&lt;' '&gt;&gt;'}</td>
</tr>
</tbody>
</table>

If you want the font color of your links to change to red after they’ve been visited, you could use the following style declaration:

```
a:visited {color: red}
```

In some cases, two or more pseudo-classes can apply to the same element—for example, a link can be both previously visited and hovered over. In such situations, the standard cascading rules apply: the pseudo-class that is listed last in the style sheet will be applied to the element. For this reason, you should enter the hypertext pseudo-classes in an order that reflects how users interact with hypertext links. The link pseudo-class should come first, followed by the visited class, the hover class, and finally the active class. The link pseudo-class comes first because it represents a hypertext link that has not yet been visited or even clicked by the user. The visited pseudo-class comes next, for the link that has been previously visited or clicked. The hover pseudo-class comes next, for the situation in which the user has once again moved the mouse pointer over the hypertext link before clicking the link. The active pseudo-class is last, representing the exact instant in which the link is clicked by the user.

*Tip*

You can achieve interesting rollover effects by having the browser change the background image or background color of the hypertext link.
Creating a Hypertext Rollover

To create a rollover for a hypertext link, apply these styles to the link element

```css
a:link {styles}
a:visited {styles}
a:hover {styles}
a:active {styles}
```

where *styles* are the CSS styles applied to hypertext links that have not been visited (link), already visited (visited), have the mouse pointer over them (hover), or are actively being clicked (active).

Kathy wants to remove the underlining from all of the links on her Getting Started page. If the mouse pointer is hovering over a link, however, she wants the link text to appear in a black font and underlined. The style declarations to remove the underlining and to create this rollover effect are:

```css
#links a:link {text-decoration: none}
#links a:visited {text-decoration: none}
#links a:hover {color: black; text-decoration: underline}
#links a:active {text-decoration: none}
```

Add these styles now to the scraps.css style sheet.

**To create a rollover effect for hypertext links:**

1. Return to the `scraps.css` file in your text editor.
2. Directly below the style for the `#links li.newGroup` selector, insert the following style declarations, as shown in Figure 4-18:

```css
#links a:link {text-decoration: none}
#links a:visited {text-decoration: none}
#links a:hover {color: black; text-decoration: underline}
#links a:active {text-decoration: none}
```

3. Save your changes to the file, and then refresh the `start.htm` file in your Web browser.
4. Verify that the links in the list of links are no longer underlined (because you have set the text-decoration style to have a value of none).
5. Hover your mouse pointer over a link in the list and verify that when the mouse pointer hovers over the link, it appears in a black font and is underlined. See Figure 4-19.
Earlier versions of HTML did not include support for the link, visited, and active pseudo-classes. If a Web page author wanted to change the color of a hypertext link, he or she would have to add to the page’s <body> tag the attributes

```html
<body link="color" vlink="color" alink="color">
```

where the link attribute specifies the color of unvisited links, the vlink attribute specifies the color of visited links, and the alink attribute specifies the color of active links. Colors had to be entered either as a supported color name or as a hexadecimal color value. There is no HTML attribute for creating a rollover effect, so for older browsers you would have to use CSS (if it was supported) or a programming language such as JavaScript to display rollovers.

The link, vlink, and alink attributes have been deprecated and are not supported by strictly compliant XHTML code, but you might still see them used in the code of older Web pages.

## Creating a Drop Cap

Kathy has a few more formatting changes she would like you to make to the Getting Started page. She wants you to add the following effects to the first paragraph on the page:

- The first line should be displayed in a small caps style.
- The first letter should be increased in size and displayed as a drop cap.
So far all of our selectors have been based on elements that exist somewhere in the
document hierarchy. We can also define selectors based on pseudo-elements that are not
part of the document tree, but instead are abstracted from what we know of an element’s
content, use, or position in the document. For example, a paragraph element is part of
the document tree and is marked with the <p> tag, but the first line of that paragraph is
not—there is no “first line” element even though people intuitively know what page con-
tent corresponds to the paragraph’s first line. CSS’s support for pseudo-elements enables
you to create styles for objects such as a paragraph’s first line.

The selector for a pseudo-element is similar to what we use for a pseudo-class. The
syntax of the pseudo-element selector is

```
selector: pseudo-element { styles }
```

where selector is an element or group of elements within the document, pseudo-element is
an abstract element based on the selector, and styles are the styles that you want to apply
to the pseudo-element. Figure 4-20 lists some of the pseudo-elements supported by CSS.

<table>
<thead>
<tr>
<th>Pseudo-element</th>
<th>Description</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>first-letter</td>
<td>The first letter of the element text</td>
<td>p:first-letter {font-size:14pt}</td>
</tr>
<tr>
<td>first-line</td>
<td>The first line of the element text</td>
<td>p:first-line {text-transform: uppercase}</td>
</tr>
<tr>
<td>before</td>
<td>Content inserted directly before the element</td>
<td>p:before {content:&quot;Special!&quot;}</td>
</tr>
<tr>
<td>after</td>
<td>Content appended to the element</td>
<td>p:after {content:&quot;eof&quot;}</td>
</tr>
</tbody>
</table>

For example, to display the first letter of every paragraph in a gold fantasy font, you
could apply the following style:

```
p:first-letter {font-family: fantasy; color: gold}
```

The advantage of this pseudo-element is that you don’t have to mark the first letter in the
HTML document; its position is inferred by the browser when it applies the style.

A pseudo-element is also useful for a design element such as a drop cap. To create a
drop cap, you increase the font size of an element’s first letter and float it on the left
margin. Drop caps also generally look better if you decrease the line height of the first
letter, enabling the surrounding content to better wrap around the letter. Finding the best
combination of font size and line height is a matter of trial and error; and unfortunately
what looks best in one browser might not look as good in another. After trying out several
combinations for the Getting Started page, you settle on a drop cap that is 400% the size
of the surrounding text, with a line height of 0.8. The following style will create this
effect:

```
p:first-letter {float: left; font-size: 400%; line-height: 0.8}
```

However, Kathy only wants to apply this style to the first paragraph on each page. The
first paragraph on the Getting Started page has already been given the id value firstp, so
the style declaration becomes

```
Tip
Older browsers might not
support the first-letter
pseudo-element. If you still
want to create a drop cap
for those browsers, mark
the first letter with a span
element and apply your
style to that element.
```
Because Kathy also wants the first line of that paragraph to be displayed in small caps, you will also use the first-line pseudo-element in the following style:

```
#firstp:first-line {font-variant: small-caps}
```

Add both of these styles to the scraps.css style sheet.

---

**To create the drop cap effect:**

1. Return to the `scraps.css` file in your text editor.
2. Directly above the style for the `#article` selector, insert the following two styles involving the first-letter and first-line pseudo-elements. See Figure 4-21:

```
#firstp:first-line {font-variant: small-caps}
#firstp:first-letter {float: left; font-size: 400%; line-height: 0.8}
```

3. Close the `scraps.css` file, saving your changes, and then refresh the `start.htm` file in your Web browser. Figure 4-22 shows the final layout of the Getting Started page.

---

4. If you want to take a break before starting the next session, close any open files or programs now.
Working with Pseudo-Elements

- To apply a style to the first line of an element, use the pseudo-element selector
  
  \texttt{selector:first-line}

  where \texttt{selector} is the name of the element or elements in the document.

- To apply a style to the first letter of an element, use the pseudo-element selector
  
  \texttt{selector:first-letter}

- To insert a text string before an element, use the style
  
  \texttt{selector:before \{content: "text"\}}

  where \texttt{text} is the content of the text string.

- To insert a text string after an element, use the style
  
  \texttt{selector:after \{content: "text"\}}

Generating Text with Pseudo-Elements

You can use CSS to insert text into your Web page using the before and after pseudo-elements. The before pseudo-element places text directly before the element, while the after pseudo-element placed the text directly after the element. The syntax of both pseudo-elements is

\texttt{selector:before \{content: "text"\}}

\texttt{selector:after \{content: "text"\}}

where \texttt{selector} is an element to which you want to add the \texttt{text} string. For example, the style

\texttt{em:after \{content: "!"\}}

appends an exclamation point to the end of every element marked with a \texttt{<em>} tag. You can use the before and after pseudo-elements in conjunction with other pseudo-elements and pseudo-classes. The code

\texttt{a:hover:before \{content: "<"\}}

\texttt{a:hover:after \{content: ">"\}}

creates a rollover effect in which the \texttt{<} and \texttt{>} characters are placed around a hypertext link when a mouse pointer hovers over the link.

The content value must be entered as a text string, and you cannot use the content property to insert HTML code. The browser displays the HTML code rather than the element the code represents. For example, if you apply the style

\texttt{em:after \{content: "<b>!</b>"\}}

the browser displays the text of opening and closing \texttt{<b>} tags in addition to the exclamation point. Although you cannot insert an HTML element, you can insert an HTML attribute. This is useful because attribute values are usually not displayed on the Web page, but you can automatically insert an attribute value using the \texttt{attr} property

\texttt{content: attr(attribute)\}}

where \texttt{attribute} is an attribute of the element. For example, the following style appends every hypertext link with the link's URL (as stored in the \texttt{href} attribute):

\texttt{a:after \{content attr(" [" attr(href) " ] "\)}}

Note that in this example, the \texttt{href} attribute will be enclosed within a set of opening and closing square brackets \texttt{[ ]}. This makes your text easy to read by using spaces or brackets to offset the generated content from its surrounding text.
Using the before and after pseudo-elements, you can create truly dynamic Web pages whose content can change based on the styles stored in different style sheets. Internet Explorer does not support the before and after pseudo-elements unless your code puts the IE browser in standards mode. You will not need to use the before and after pseudo-elements in Kathy’s Web site.

Kathy is pleased with the work you’ve done adding special effects to the Getting Started page. She feels that the use of the first-letter and first-line pseudo-elements to create the drop cap effect added a great deal to the appearance of the page. She’s also pleased with your work on the rollover effect in the list of links and the graphic image used in the Basic Materials list. In the next session, you’ll expand your understanding of CSS by using the styles to directly position elements on the rendered Web page.

Review | Session 4.1 Quick Check

1. Specify the style to italicize the content of all span elements nested within paragraphs.
2. Specify the style to italicize the content of all span elements that are direct children of paragraph elements.
3. Specify the style to italicize all h2 headings that directly follow h1 headings.
4. Specify a style to display all elements that belong to the newsAlert class in boldface text.
5. Specify a style to display only span elements belonging to the newsAlert class in boldface text.
6. Specify a style in which every hypertext link is displayed with a yellow background when the mouse pointer hovers over the link.
7. Specify a style in which hovering over a hypertext link causes the Web browser to change the link’s background image to the graphic file hover.jpg.
8. Specify a style that displays the first letter of every block quote in a red font.
9. Specify a style that displays the first line of every block quote in a red font.

Session 4.2

Positioning Objects with CSS

One purpose of the Online Scrapbooks Web site is to teach new scrapbookers how to create beautiful and interesting pages. Every month Kathy wants to highlight a scrapbook page that displays some noteworthy features. Figure 4-23 shows the current Samples page. (Note that because of the scraps.css style sheet, this page uses the same layout as the other pages in the Web site.) The scrapbooking sample is displayed in the main section of the document.
Kathy wants to augment the page by inserting callouts that highlight certain portions of the scrapbooking sample for the reader. She wants each callout to be placed close to the feature that it highlights. Kathy has drawn in the locations of the three callouts that she wants to add in the sketch shown in Figure 4-24.
Figure 4-25 shows the text of the three callout notes.

<table>
<thead>
<tr>
<th>Note</th>
<th>Text</th>
</tr>
</thead>
<tbody>
<tr>
<td>note 1</td>
<td>Paste cut-out letters and words in your scrapbook to create a 3D effect. Online Scrapbooks sells professionally designed cut-out letters, words, and phrases for all occasions.</td>
</tr>
<tr>
<td>note 2</td>
<td>Clippings, flyers, programs, and other memorabilia are valuable sources of information that can enhance your scrapbook pages. Make sure that any material is copied to acid-free paper. Newspaper clippings are especially susceptible to deterioration.</td>
</tr>
<tr>
<td>note 3</td>
<td>Photographic cut-outs and textured backgrounds can add visual interest to your pages. See the online store for our wide variety of textured and embossed papers.</td>
</tr>
</tbody>
</table>

You’ll insert each of these notes in div containers placed within the main section of the page. You’ll set the id values of the three elements to note1, note2, and note3, respectively, and you’ll add the class value notes to each element so that you can apply a common set of styles to all of the notes.
To insert the three notes:

1. Use your text editor to open the `samptxt.htm` from the tutorial.04\tutorial folder included with your Data Files. Enter your name and the date in the comment section of the file and save it as `samples.htm` in the same folder.

2. Directly below the h2 heading, Samples from Online Scrapbooks, insert the following div container elements, as shown in Figure 4-26:

   ```html
   <div id="note1" class="notes">
     <p>Paste cut-out letters and words in your scrapbook to create a 3D effect. Online Scrapbooks sells professionally designed cut-out letters, words, and phrases for all occasions.</p>
   </div>
   <div id="note2" class="notes">
     <p>Clippings, flyers, programs, and other memorabilia are valuable sources of information that can enhance your scrapbook pages. Make sure that any material is copied to acid-free paper. Newspaper clippings are especially susceptible to deterioration.</p>
   </div>
   <div id="note3" class="notes">
     <p>Photographic cut-outs and textured backgrounds can add visual interest to your pages. See the online store for our wide variety of textured and embossed papers.</p>
   </div>
   ``

Because the styles in this task will apply only to this page and no others in Kathy's Web site, you'll add an embedded style sheet to the samples.htm file to format the appearance of the three notes. Kathy wants the text to appear in a brown 8-point sans-serif font on an ivory background. She wants the note boxes to be displayed with a 3-pixel light gray inset border. The notes should be 130 pixels wide with a margin space of 5 pixels around the paragraphs.
To define a style for the three notes:

1. Scroll to the top of the samples.htm file.
2. Directly below the link element, insert the following embedded style sheet as shown in Figure 4-27:

   ```html
   <style type="text/css">
   .notes {font-family: sans-serif; font-size: 8pt; color: brown;
           background-color: ivory;
           border: 3px inset rgb(212, 212, 212); width: 130px}
   .notes p {margin: 5px}
   </style>
   ```

3. Save your changes to the file.
4. Open the samples.htm file in your Web browser. Figure 4-28 shows the formatted appearance of the three note boxes. Note that although the boxes are placed side-by-side in this figure to make them easier to read, they should be stacked one on top of the other at the top of your Web page.

Trouble? Depending on your browser, your note boxes may look slightly different than those shown in Figure 4-28.

Now that you've entered the text and the formatting styles for the three callout notes, your next task is to position them at appropriate locations on the Samples page.

The Position Style

Positioning was one of the first enhancements to the original CSS1 specifications. Collectively, the various positioning styles were known as CSS-Positioning, or more commonly, CSS-P. CSS-P became part of the specification for CSS2, and positioning styles were some of the first CSS2 styles to be adopted by browsers.
To place an element at a specific position on the page, use the styles

```css
position: type; top: value; right: value; bottom: value; left: value;
```

where `type` indicates the type of positioning applied to the element, and the top, right, bottom, and left styles indicate the coordinates of the top, right, bottom, and left edges of the element. In practice, usually only the left and top coordinates are used because the right and bottom coordinates can be inferred given the element’s height and width. Coordinates can be expressed in any of the CSS measuring units.

The position style has five possible values: static, absolute, relative, fixed, and inherit. The default position is static, which enables browsers to place an element based on where it flows in the document. This is essentially the same as not using any CSS positioning at all. Any values specified for the left or top styles with a static position are ignored by the browser. You’ll explore each of the other values (absolute, relative, fixed, and inherit) so that you can use them to position the notes on Kathy’s Sample Pages page.

### Absolute Positioning

**Absolute positioning** enables you to place an element at specific coordinates either on a page or within a containing element. For example, the declaration

```css
position: absolute; left: 100px; top: 50px
```

places an element at the coordinates (100, 50), or 100 pixels to the right and 50 pixels down from upper-left corner of the page or the containing element. Once an element has been placed using absolute positioning, it affects the placement of other objects on the Web page. To explore how absolute positioning affects page layout, you’ll use a demo containing objects that can be positioned on the Web page.

---

**To explore absolute positioning:**

1. Use your Web browser to open the `demo_positioning.htm` file from the `tutorial.04\demo` folder included with your Data Files.

   The demo page contains two colored boxes that you can move by changing the values in the Positioning Styles box. The boxes are initially set to their default position, which is within the flow of the other elements on the demo page. To make it easier to place the boxes at specific positions, a grid marked in pixels has been added to the page background.

2. Select `absolute` from the list box for the outer box, and then press the **Tab** key.

3. Enter **275** in the left box, and then press the **Tab** key. Enter **350** in the top box, and then press the **Tab** key again. As shown in Figure 4-29, the red outer box is placed at the page coordinates (275, 350).
Absolute positioning takes an element out of the normal flow of a document, so that any subsequent content flows into the space previously occupied by the element. Note that on the demo page, the sample paragraph moves up into the space that was previously occupied by the red outer box.

When elements are nested within one another, the position of the element is based on the coordinates within the parent object if the object is itself placed on the page using a CSS positioning style. If the parent object is not positioned using a CSS style, then the position of the nested object is set within the next object higher up in the hierarchy of elements positioned on the page. If no other objects are positioned on the page, the top and left coordinates are based on the browser window. To see this effect, return to the demo page.

**To view absolute positioning with a nested object:**

1. Within the demo page, select `absolute` from the list box for the inner element.
2. Enter **90** in the left box for the inner object and **75** for the top box. As shown in Figure 4-30, the inner yellow box is placed at the (90, 75) coordinate within the outer box, not within the Web page.
Positioning a nested object

3. Select static from the list box for the outer element.

As shown in Figure 4-31, the red outer box is returned to its default position on the Web page. The yellow inner box is now placed at the coordinate (90, 75), but within the Web page.
4. Continue experimenting with the demo page by entering values for the top and left coordinates and observing the effect on the placement of the boxes.

Relative Positioning

Relative positioning is used to move an element relative to its default position on the page. An element's default position is where the browser would have placed it if no positioning style was applied to it. For example, the style

```
position: relative; left: 100px; top: 50px
```

places an element 100 pixels to the right and 50 pixels down from its normal placement in a browser window. Relative positioning does not affect the position of other elements on a page, which retain their original positions as if the element had never been moved. You'll use the demo page to experiment with this.

To explore relative positioning:

1. Click the reset button within the demo page to return both boxes to their default locations on the Web page.
2. Select relative from the list box for the outer element, and then enter 275 for the left value and 50 for the top value. As shown in Figure 4-32, the outer box moves 275 pixels to the right and 50 pixels down from its default location.

Also note that the sample paragraph does not flow into the space previously occupied by the colored boxes. The layout of the rest of the page is unaffected when relative positioning is applied.
3. Explore other combinations of absolute and relative positioning to see their effect on the layout of the demo page.

In many Web page layouts, you might want to position any object nested within a div container, but you don’t need to move the container itself. In those cases, use relative positioning to place the div container with the top and left values set to 0 pixels. The position you apply to the nested object will then be based on coordinates within the div container object.

**Fixed and Inherited Positioning**

An element placed with absolute or relative positioning scrolls with the rest of the document. Alternately, you can fix an element at a specific spot in the document window while the rest of the page scrolls by setting the value of the position style to fixed. Note that not all browsers support the fixed position, so you should use it with some caution if it is a crucial part of your Web page layout.

You can also assign the inherit position style to an element so that it inherits the position value of its parent element. You’ll explore both positioning styles on the demo page.

**To explore fixed and inherited positioning:**

1. Click the *reset* button within the demo page to return both boxes to their default locations on the Web page.

2. Select *fixed* from the list box for the outer element, and then enter 300 for the left and top values.

   The red box is placed at the window coordinates (300, 300). The sample paragraph moves up into the space previously occupied by the red box.

   **Trouble?** If you are running an older browser, you might not see any change in the position of the red box.

3. Select *inherit* from the list box for the inner element, and then enter 600 for the left value and 300 for the top value.

   The yellow box inherits the position style of its parent. In this case it uses fixed positioning and is placed to the right of the outer red box. See Figure 4-33.
Fixed and inherited positioning

Figure 4-33

Trouble? Internet Explorer does not support the inherit position style at the time of this writing. To fix the position of the inner box, you have to choose fixed drop in the list box.

4. Resize the browser window so it’s small enough to force the browser to display the vertical and horizontal scroll bars. Scroll through the document and verify that the two color boxes remain fixed at the same location within the window.

5. Continue to experiment with different positioning combinations. Close the demo page when you’re finished.

Reference Window | Positioning an Object with CSS

• To position an object at a specific coordinate, use the style

```
position: type; top: value; right: value; bottom: value; left: value;
```

where `type` indicates the type of positioning applied to the object (absolute, relative, static, fixed, or inherit) and the top, right, bottom, and left styles indicate the coordinates of the object.

Now that you’ve seen how to work with the different positioning styles of CSS, you can apply your knowledge to position the three callout notes. After trying different values, you and Kathy settle on the following coordinates using absolute positioning:

- `note1`: (600, 120)
- `note2`: (170, 400)
- `note3`: (570, 550)

You’ll add styles for these positions to the embedded style sheet in the `samples.htm` file.
To position the three notes for the Samples page:

1. Return to the `samples.htm` file in your text editor.

2. Add the following styles to the embedded style sheet, as shown in Figure 4-34:

   ```css
   #note1 {position: absolute; left: 600px; top: 120px}
   #note2 {position: absolute; left: 170px; top: 400px}
   #note3 {position: absolute; left: 570px; top: 550px}
   ```

3. Save your changes, and then reload the `samples.htm` file in your Web browser. Figure 4-35 shows the placement of the three sample notes.
You show Kathy the revised page. She likes the position of the notes, but she points out that they are pretty big and they hide too much of the scrapbooking sample. Kathy would like you to investigate ways of making the notes less intrusive.

Working with Overflow and Clipping

Reducing the height of each note by lowering the value of its height attribute might seem like an easy solution to Kathy’s first request. Unfortunately, though, this would not meet her needs because the height of each note expands to accommodate its content. If you want to force an element into a specified height and width, you have to define how the browser should handle a situation where content overflows the space allotted to the object. The syntax of the overflow style is

\[ \text{overflow: type} \]

where \textit{type} is visible (the default), hidden, scroll, or auto. A value of visible instructs browsers to increase the height of an element to fit the overflow content. The hidden value keeps an element at the specified height and width, but cuts off excess text. The scroll value keeps an element at the specified dimensions, but adds horizontal and vertical scroll bars to allow users to scroll through the overflow. Finally, the auto value keeps an element at the specified size, adding scroll bars only as they are needed. Figure 4-36 shows examples of the effect of each overflow value.

You decide to limit the height of each callout note to 90 pixels and have the browser display scroll bars as needed by setting the value of the overflow style to auto.

To define the overflow style for the callout notes:

1. Return to the samples.htm file in your text editor.
2. Add the following styles to the .notes selector, as shown in Figure 4-37:
   \[ \text{height: 90px; overflow: auto} \]

Figure 4-36  Values of the overflow style

<table>
<thead>
<tr>
<th>visible</th>
<th>hidden</th>
<th>scroll</th>
<th>auto</th>
</tr>
</thead>
<tbody>
<tr>
<td>Clippings, flyers, programs, and other memorabilia are valuable sources of information that can enhance your scrapbook pages. Make sure that any material is copied to acid-free paper. Newspaper clippings are especially susceptible to deterioration.</td>
<td>Clippings, flyers, programs, and other memorabilia are valuable sources of information that can enhance your scrapbook pages. Make sure that any material is copied to acid-free paper. Newspaper clippings are especially susceptible to deterioration.</td>
<td>Clippings, flyers, programs, and other memorabilia are valuable sources of information that can enhance your scrapbook pages. Make sure that any material is copied to acid-free paper. Newspaper clippings are especially susceptible to deterioration.</td>
<td>Clippings, flyers, programs, and other memorabilia are valuable sources of information that can enhance your scrapbook pages. Make sure that any material is copied to acid-free paper. Newspaper clippings are especially susceptible to deterioration.</td>
</tr>
</tbody>
</table>

Figure 4-37  Setting the overflow style for the notes

\[
\text{<style type="text/css">}
\text{\ .notes { \text{font-family: \ sans-serif; \ font-size: \ 0pt; \ color: \ brown; \ background-color: \ ivory; \ border: \ 0px \ inset \ rgb(252, \ 212, \ 212); \ width: \ 120px; \ height: \ 90px; \ overflow: \ auto} \n\text{\ .notes p { \text{margin: \ 5px} \n\text{\ #note1 { \text{position: \ absolute; \ left: \ 600px; \ top: \ 120px} \n\text{\ #note2 { \text{position: \ absolute; \ left: \ 480px; \ top: \ 120px} \n\text{\ #note3 { \text{position: \ absolute; \ left: \ 570px; \ top: \ 350px} \n\text{</style>}
\]
3. Save your changes, and then refresh the `samples.htm` file in your Web browser. Figure 4-38 shows the appearance of the three callout notes with heights limited to 90 pixels and scroll bars added.

4. Use the scroll bars to verify that the entire content of each note is still available to the user.

   **Trouble?** Depending on your browser, you might not see scrollbars around each note box.

5. If you want to take a break before starting the next session, you can close any open files or programs now.

---

**Clipping an Element**

Closely related to the overflow style is the clip style. The clip style allows you to define a rectangular region through which the element's content can be viewed. Anything that lies outside the boundary of the rectangle is hidden. The syntax of the clip style is

```
clip: rect(top, right, bottom, left)
```
where top, right, bottom, and left define the coordinates of the clipping rectangle. For example, a clip value of rect(10, 175, 125, 75) defines a clip region whose top and bottom edges are 10 and 125 pixels from the top of the element, and whose right and left edges are 175 and 75 pixels from the left side of the element. See Figure 4-39.

The top, right, bottom, and left values can also be set to auto, which matches the specified edge of the clipping region to the edge of the parent element. For example, a clip value of rect(10, auto, 125, 75) creates a clipping rectangle whose right edge matches the right edge of the parent element.

### Reference Window

**Working with Content Overflow and Clipping**

- To specify how the browser should handle content that overflows an element's boundary, use the style
  ```html
  overflow: type
  ```
  where type is visible (to expand the element height to match the content), hidden (to hide the excess content), scroll (to always display horizontal and vertical scroll bars), or auto (to display scroll bars if needed).
- To clip an element's content, use the style
  ```html
  clip: rect(top, right, bottom, left)
  ```
  where top, right, bottom, and left define the boundaries of the clipping rectangle.

### Limiting Width and Height

In some page layouts, you might want to limit an element's height or width. This is often desirable when you’ve specified the element's size using relative units that can expand or contract depending on the size of the browser window. If the browser window is very wide, the element might become too wide to be easily readable. If the browser window is too narrow, the element might be reduced to a size that is also difficult to view. Rather
than allowing these problems to occur, you can specify an element’s minimum or maximum height or width using the styles

\[
\text{min-width: value} \\
\text{min-height: value} \\
\text{max-width: value} \\
\text{max-height: value}
\]

where \text{value} is the width or height value in one of the CSS units of measure. The min and max values are usually used alongside the height and width styles to set a possible range of values for an element. For example, the style declaration

\[
\text{div {width: 80%; min-width: 200px; max-width: 700px}}
\]

sets the width of the div element to 80% of the Web browser window. If the browser window is 800 pixels wide, the div element will be 640 pixels wide. However, browser windows can vary in size and many users will resize their browser windows to free up desktop space. In that case, the size of the div element will vary accordingly, but it will never be allowed to get smaller than 200 pixels or larger than 700 pixels. Using the min and max styles enables the Web page designer to have some control over the page layout and avoid problems caused by either very large or very small windows.

Max-Width and Internet Explorer

As mentioned in Tutorial 3, usability studies have shown that most users are comfortable reading text that extends no more than 60 to 70 characters per line or about 30 em. Beyond this length, reading comprehension goes down rapidly and eye fatigue increases. To deal with this problem, Web page designers often use the max-width style to ensure that their Web pages are not too wide on large monitors or screens set to high resolutions.

Internet Explorer did not fully support maximum widths until IE 7. For browser versions earlier than IE 7, Web page authors have had to adopt workarounds to approximate the effect of the max-width style. One popular approach, offered by Svend Tofte, is to use a CSS command introduced and supported by Internet Explorer to automatically size the width of an element based on the width of the browser window. For example, the following set of styles defines a maximum width of 800 pixels for an object:

\[
\text{max-width: 800px;}
\text{width: expression(document.body.clientWidth > 800? "800px": "auto" )};
\]

In this code, browsers that support maximum widths use the max-width style in the first line to set the maximum width of the object to 800 pixels. Those browsers then ignore the next line and continue on to the rest of the style sheet. Internet Explorer on the other hand, ignores the max-width style in the first line and goes directly to the second line. The second line contains a command that tests whether the browser window is wider than 800 pixels. If it is, it sets the width of the object to 800 pixels. If the browser window is not wider than 800 pixels, the object will be automatically sized by the browser to fit into whatever space is available. The result is that the object will have a maximum width under both IE and browsers that support the max-width style.

This particular workaround can be adapted for different widths and different units of measure. For more information, you can view Svend Tofte’s work at www.svendtofte.com/code/max_width_in_ie/ or do a Web search for IE workarounds to the max-width problem. As always, you should test any code to ensure that it works with a variety of browsers and operating systems.
Stacking Elements

Positioning elements can sometimes lead to objects that overlap each other. By default, elements that are formatted later in an HTML or XHTML document are stacked on top of earlier elements. In addition, elements placed using CSS positioning are stacked on top of elements that are not. To specify a different stacking order, use the style

\[
\text{z-index: value}
\]

where \text{value} is a positive or negative integer or the keyword “auto.” As shown in Figure 4-40, objects are stacked based on their z-index values, with the highest z-index values placed on top. A value of auto allows the browser to determine stacking order using the default rules.

The z-index style only works for elements that are placed with absolute positioning. Also, an element’s z-index value determines its position relative only to other elements that share a common parent; the style has no impact when applied to elements with different parents. Figure 4-41 shows a diagram in which the object with a high z-index value of 4 is still covered because it is nested within another object that has a low z-index value of 1.
Kathy is pleased with how the notes are positioned over the scrapbooking sample, so you don’t need to use the clip or z-index styles. The page looks good on computer monitors. In the next session, you’ll explore styles to make your Web pages ready for print media and portable devices.

Session 4.2 Quick Check

1. Specify the style to place an element with the id named logo at the coordinates (150, 75) using absolute positioning.
2. Specify a style to place the logo element 25% down from the top of the page and 10% to the right.
3. What is the style to move span elements belonging to the class highlight up 10 pixels?
4. Specify a style that moves all link elements 5 pixels down when the mouse pointer hovers over them.
5. What is the style to fix an element with the links id at the browser window coordinates (10, 50)?
6. Specify a style to set the width of all block quotes to 70% of the browser window width with a minimum width of 250 pixels and a maximum width of 650 pixels.
7. Specify a style to set the height of all block quotes to 25% of the browser window height. If the content of the block quote cannot fit within this space, include a style to add scroll bars to the block quote as needed.
8. The #title element has a z-index of 1. The #subtitle has a z-index of 5. Will the #subtitle element always be displayed on top of the #title element? Explain why or why not.

Session 4.3

Working with Different Media

Many users of the Online Scrapbooks Web site have reported to Kathy that they enjoy the monthly Samples page so much that they print the samples and store them for future reference. However, these users often find that the pages don’t print well. Most users would prefer to print only the scrapbook sample, without the Online Scrapbooks header, links list, and footer. Also, they enjoy the notes that Kathy adds to the sample page, but they would like those notes to be printed on a separate page from the scrapbook sample. In Figure 4-42, Kathy has sketched the design she envisions for the printed version of the Samples page.
One solution to Kathy’s problem would be to create two versions of the Samples page: one for computer screens and the other for printouts. However, Kathy would like to avoid having multiple versions of the same file on her Web site. She would much prefer having a separate style sheet: one that is designed for printed output. She would like you to examine how to create style sheets that are designed for specific devices such as a printer.

**Media Types**

By default, a style sheet is applied to all devices, and each device must determine how best to match the styles to its own requirements. For example, when you print a Web page, the Web browser and its built-in styles prepare the document for the printer. The user also has some control over that process—for example, determining the size of the page margins or the content of the printout’s header or footer. However, beyond that, the user cannot control how the page is printed.

CSS2 and subsequent versions have given more control to Web page authors to specify output styles for particular devices. To do that, you use the media attribute to specify an output device in either the style element (for embedded style sheets) or in the link element (for external style sheets). The syntax of the media attribute is

```html
<style type="text/css" media="type">
  ...
</style>
```

or

```html
<link href="url" type="text/css" media="type" ... />
```

where *type* is the type of media used by the style sheet. Figure 4-43 lists the different values of the media attribute.
Values of the media attribute

<table>
<thead>
<tr>
<th>Value</th>
<th>Used for</th>
</tr>
</thead>
<tbody>
<tr>
<td>all</td>
<td>All output devices (the default)</td>
</tr>
<tr>
<td>aural</td>
<td>Speech and sound synthesizers</td>
</tr>
<tr>
<td>braille</td>
<td>Braille tactile feedback devices</td>
</tr>
<tr>
<td>embossed</td>
<td>Paged Braille printers</td>
</tr>
<tr>
<td>handheld</td>
<td>Small or handheld devices with small screens, monochrome graphics, and limited bandwidth</td>
</tr>
<tr>
<td>print</td>
<td>Printers</td>
</tr>
<tr>
<td>projection</td>
<td>Projectors</td>
</tr>
<tr>
<td>screen</td>
<td>Computer screens</td>
</tr>
<tr>
<td>tty</td>
<td>Fixed-width devices like teletype machines and terminals</td>
</tr>
<tr>
<td>tv</td>
<td>Television-type devices with low resolution, color, and limited scrollability</td>
</tr>
</tbody>
</table>

For example, to specify that aural browsers should render your Web page using the sounds.css style sheet, you would enter the following link element in the HTML file:

```html
<link href="sounds.css" type="text/css" media="aural" />
```

In the same way, you would use the following media attribute in an embedded style sheet to indicate that its styles are intended for aural devices:

```html
<style type="text/css" media="aural">
...
</style>
```

The media attribute can also contain a comma-separated list of media types. The following link element points to a style sheet designed for both print and screen media:

```html
<link href="output.css" type="text/css" media="print, screen" />
```

Style sheets cascade through the media types in the same way they cascade through a document tree. A style sheet in which the output device is not specified is applied to all devices, unless it is superseded by a style designed for a particular device. In the following set of embedded style sheets, h1 headings are displayed in a sans-serif font for all devices; however, the text color is red for computer screens and black for printed pages:

```html
<style type="text/css">
  h1 {font-family: sans-serif}
</style>
<style type="text/css" media="screen">
  h1 {color: red}
</style>
<style type="text/css" media="print">
  h1 {color: black}
</style>
```

When no value is given to the media attribute, any style defined in the embedded or external style sheet is used for all media, where applicable.
The @media Rule

It's not always convenient to maintain several different style sheets for the same document. In place of several style sheets, you can use a single style sheet broken down into different sections for each media type. This is done using the rule

```
@media type {
  styles declarations
}
```

where `type` is one of the supported media types and `styles declarations` are style declarations associated with that media type. For example, the following style sheet is broken into four sections with a different collection of styles for screen, print, handheld, and television media:

```
@media screen { body {font-size: 1em} h1 {font-size: 2em} }
@media print { body {font-size: 12pt} h1 {font-size: 16pt} }
@media handheld { body {font-size: 8pt} h1 {font-size: 12pt} }
@media tv { body {font-size: 16pt} h1 {font-size: 24pt} }
```

In this style sheet, the font size is smallest for a handheld device (which presumably has a limited screen area), and largest for a television (which is usually viewed from a greater distance). Similar to the media attribute, the @media rule also allows you to place media types in a comma-separated list, as in the following declaration:

```
@media screen, print, handheld, tv {
  h1 {font-family: sans-serif}
}
```

Both the media attribute and the @media rule come with their own benefits and disadvantages. The @media rule enables you to consolidate all of your styles within a single style sheet; however, this consolidation can result in larger and complicated files. The alternative—placing media styles in different sheets—can make those sheets easier to maintain; however, if you change the design of your site, you might have to duplicate your changes across several style sheets.

Media Groups

The distinction among the different media types is not always immediately clear. For example, how is projection media different from screen media? The difference lies in what kind of output can be sent to the media. All output media can be described based on some common properties. CSS uses media groups to describe how different media devices render content. There are four media groups based on the following characteristics:

- continuous or paged
- visual, aural, or tactile
- grid (for character grid devices) or bitmap
- interactive (for devices that allow user interaction) or static (for devices that allow no interaction)

Figure 4-44 shows how all output media are categorized based on the four media groups. For example, a printout is paged (because the output comes in discrete units or pages), visual, bitmap, and static (you can't interact with it). A computer screen, on the other hand, is continuous, visual, bitmap, and can be either static or interactive.
Media groups are important because the CSS2 specifications indicate which media group a particular style belongs to, rather than the specific media device. For example, the font-size style belongs to the visual media group because it describes the visual appearance of the document content; and as indicated in Figure 4-44, this means you can use the font-size style with handheld, print, projection, screen, tty, and tv media. However, it would have no meaning to—and will in fact be ignored by—devices whose output consists of Braille or aural communication. On the other hand, the pitch style, used to define the pitch or frequency of a speaking voice, belongs to the aural media group and is supported by aural and tv devices. By studying the media groups, you can choose the styles that apply to a given output device.

### Creating Styles for Different Media

- To create a style sheet for specific media, add the attribute `media = "type"` to either the link element or the style element, where `type` is one or more of the following: aural, braille, embossed, handheld, print, projection, screen, tty, or tv, or all. If you don’t specify a media type, the style sheet applies to all media. Multiple media types should be entered in a comma-separated list.
- To create a style for specific media from within a style sheet, add to the sheet the rule

```
@media type {style declarations}
```

where `type` is the media type and `style declarations` are the styles that are applied to the different page elements within that media.

Now that you’ve seen how to define the style sheet for a particular media device, you decide to create one for printers.

### To create a style sheet for print media:

1. Use your text editor to open the `printtxt.css` style sheet from the tutorial.04\tutorial folder included with your Data Files. Enter your name and the date in the comment section of the file.
2. Save the file as `print.css` in the same folder.
Kathy wants you to use the print.css style sheet for any paged visual media, which includes both printed media and projected media. You’ll use the scraps.css style sheet for continuous visual media, which includes computer screens, television monitors, and ttys. In the samples.htm file, add a link to the print.css style sheet and insert the media attribute to indicate which style sheets to use for which output devices.

To link Kathy’s Samples page to the print.css style sheet:

1. Return to the samples.htm file in your text editor.
2. Directly above the link element in the document head, insert the following link element for the print.css style sheet:
   ```html
   <link href="print.css" rel="stylesheet" type="text/css" media="print, projection" />
   ```
3. Add the following media attribute to the link element for the scraps.css file to indicate that it should be used for screen, tv, and tty media:
   ```html
   media="screen, tv, tty"
   ```

Figure 4-45 highlights the new code in the samples.htm file.

The samples.htm file also includes an embedded style sheet. Like the external style sheet, you need to create two embedded sheets: one for printers and projection devices, and the other for screens, ttvs, and ttys.

To create an embedded style sheet for print media:

1. Within the samples.htm file, directly above the embedded style sheet, insert the following HTML code:
   ```html
   <style type="text/css" media="print, projection">
   </style>
   ```
2. Add the following media attribute to the opening <style> tag for the first embedded style sheet. See Figure 4-46.
   ```html
   media="screen, tv, tty"
   ```

Figure 4-46 Embedded style sheets for different media

To link Kathy’s Samples page to the print.css style sheet:

1. Return to the samples.htm file in your text editor.
2. Directly above the link element in the document head, insert the following link element for the print.css style sheet:
   ```html
   <link href="print.css" rel="stylesheet" type="text/css" media="print, projection" />
   ```
3. Add the following media attribute to the link element for the scraps.css file to indicate that it should be used for screen, tv, and tty media:
   ```html
   media="screen, tv, tty"
   ```

Figure 4-45 highlights the new code in the samples.htm file.

The samples.htm file also includes an embedded style sheet. Like the external style sheet, you need to create two embedded sheets: one for printers and projection devices, and the other for screens, ttvs, and ttys.

To create an embedded style sheet for print media:

1. Within the samples.htm file, directly above the embedded style sheet, insert the following HTML code:
   ```html
   <style type="text/css" media="print, projection">
   </style>
   ```
2. Add the following media attribute to the opening <style> tag for the first embedded style sheet. See Figure 4-46.
   ```html
   media="screen, tv, tty"
   ```

Figure 4-46 Embedded style sheets for different media
3. Save your changes to the file, and then reload the `samples.htm` file in your Web browser. Confirm that the appearance of the page has not changed. (It should not change because your Web browser is treated as screen media and you haven’t changed the style sheet for that media type.)

With two sets of style sheets for the different media types, you are ready to start defining the styles for printed output.

**Hiding Elements**

The first thing you notice when examining Kathy’s sketch of the printed version of the Samples page is that many elements from the Web page—such as the list of links on the left and the address at the bottom—are missing. CSS has two styles that you can use to keep an element from being displayed in the output: the display style and the visibility style. As you’ve already seen in Tutorial 3, the display style supports the value “none,” which causes the element to not be rendered by the output device. Alternately, you can use the visibility style, which has the syntax

```
visibility: type
```

where `type` is visible, hidden, collapse, or inherit (the default). A value of “visible” makes an element visible; the “hidden” value hides the element; a value of “collapse” is used with the tables to prevent a row or column from being displayed; and the “inherit” value causes an element to inherit the visibility style from its parent. Unlike the display style, the visibility style hides an element, but does not remove it from the flow of elements on the page. As shown in Figure 4-47, setting the display style to none not only hides an element, but also removes it from the page flow.

---

**Comparing the visibility and display styles**

- **visibility: hidden**
  - object is hidden but still is part of the page flow

- **display: none**
  - object is hidden and is removed from the page flow
The display: none style is more appropriate for hiding elements in most cases. Use of the visibility: hidden style is usually reserved for scripts in which an element is alternatively hidden and made visible to create an animated effect. You’ll use the display: none style to hide the #head, #links, and address selectors in the printed output.

**To apply the display: none style:**

1. Return to the **print.css** file in your text editor.
2. Add the following style declaration below the comment section:
   ```css
   #head, #links, address {display: none}
   ```
3. Kathy still wants all headings to appear in a sans-serif font in the printed version of the page. Add the following style to the sheet:
   ```css
   h1, h2, h3, h4, h5, h6 {font-family: sans-serif}
   ```
   Figure 4-48 shows the code from the print.css style sheet.

4. Save your changes to the file.

Next, you need to modify the style for the callout notes. Kathy wants the notes to be displayed as items in a bulleted list. You can change the style of the notes to list items by applying the following display style:

```css
display: list-item
```

Once the display style has been set to list-item, you can apply the same list styles you would use with elements marked with HTML’s `<li>` tag. You decide to display each note with the bullet.jpg graphic image you used earlier in Session 1. You’ll also set the text style to a 12-point sans-serif font with a margin of 20 pixels.

**To set the print style of the callout notes:**

1. Return to the **samples.htm** file in your text editor.
2. Add the following style to the embedded style sheet for printed output. See Figure 4-49.
   ```css
   .notes {display: list-item; list-style-image: url(bullet.jpg); font-family: sans-serif; font-size: 12pt; margin: 20px}
   ```

3. Save your changes to the file.

Now test whether the styles you’ve defined have been applied to the printed version of the page.
4. Reload the `samples.htm` file in your Web browser. Verify that the appearance of the page within the browser window has not changed.

5. Either print the Web page from within your browser or use your browser’s Print Preview command to preview the printed version of the page. Figure 4-50 shows how the page appears when printed.

Kathy likes the printout you created; however, she still wants the notes to appear on a separate sheet. To do this, you’ll have to place a page break in the middle of the document. Although page breaks are not supported by media types such as computer screens, they are supported in printed output and for projection devices.
Using Print Styles

CSS defines printed pages by extending the box model described in Tutorial 3 to incorporate the entire page in a page box. As shown in Figure 4-51, the page box is composed of two areas: the page area, containing the content of the document, and the margin area, containing the space between the printed content and the edges of the page.

As with the box model, you can specify the size of a page box, the page margins, the internal padding, and other features. The general rule to create and define a page box is

```
@page {styles}
```

where styles are the styles you want applied to the page. For example, the following @page rule sets the page margin for the printed output to 5 inches:

```
@page {margin: 5in}
```

A page box does not support all of the measurement units you’ve used with the other elements. For example, pages do not support the em or ex measurement units. In general, you should use measurement units that are appropriate to the dimensions of your page, such as inches or centimeters.

Page Pseudo-Classes and Named Pages

If your Web pages will require several pages when printed, you might want to define different styles for different pages. You can do this with pseudo-classes that reference specific pages. The syntax to apply a pseudo-class to a page uses the following rule
@page:pseudo-class {styles}

where pseudo-class is first (for the first page of the printout), left (for the pages that appear on the left in double-sided printouts), and right (for pages that appear on the right in double-sided printouts).

For example, if you are doing two-sided printing, you might want to mirror the margins of the left and right pages of the printout. The following styles result in pages in which the inner margin is set to 5 centimeters and outer margin is set to 2 centimeters:

@page:left {margin: 3cm 5cm 3cm 2cm}
@page:right {margin: 3cm 2cm 3cm 5cm}

To format specific pages other than the first, left, or right pages, you first must create a page name that contains a set of styles for the page. The syntax to create a page name is

@page name {styles}

where name is the label assigned to the page style. The following code creates the large_margins page name that defines a page box with 10-centimeter margins:

@page large_margins {margin: 10cm}

Once you define a page name, you can apply it to any block-level element in your document. The content of the block-level element will appear on its own page, with the browser automatically inserting page breaks before and after the element if required. To assign a page name to a block-level element, use the style

selector {page: name}

where selector is a CSS selector that points to a block-level element and name is the name of a defined page. For example, the style

blockquote {page: large_margins}

causes all block quotes to be displayed on their own separate pages using the styles defined for the large_margins page.

Setting the Page Size

Because printed media can vary in size and orientation, one of the styles supported by the page box is the size style that allows the Web author to define the default dimensions of the printed page as well as whether the pages should be printed in portrait or landscape orientation. The syntax of the size style is

size: width height orientation

where width and height are the width and height of the page, and orientation is the orientation of the page (portrait or landscape). If you don’t specify the orientation, browsers assume a portrait orientation. To format a page as a standard-size page in landscape orientation with a 1-inch margin, you would apply the following style:

@page {size: 8.5in 11in landscape; margin: 1in}

If you remove the orientation value, as in the style

@page {size: 8.5in 11in; margin: 1in}

browsers print the output in portrait by default. Note that the page sizes and orientations chosen by the Web page author can still be overridden by the user, who may choose different settings when actually printing the page.
You can also replace the width, height, and orientation values with the keyword "auto" (to let the browser determine the page dimensions) or "inherit" (to inherit the page size from the parent element). If a page does not fit into the dimensions specified by the style, browsers will either rotate the page box 90 degrees or scale the page box to fit the sheet size.

Use the @page rule to define the print layout of the Samples page. Kathy suggests that you set the page size to 8.5 × 11 inches, in portrait orientation, with 0.5-inch margins.

To set the style of the printed page:

1. Return to the `print.css` file in your text editor.
2. As shown in Figure 4-52, add the following rule to the top of the list of style declarations:

   ```css
   @page {size: 8.5in 11in portrait; margin: 0.5in}
   ```

3. Save your changes to the file.

Working with Page Breaks

When a document is sent to the printer, the printer decides the location of the page breaks unless that information is included as part of the print style. To specify a page break that occurs either before or after a page element, you apply the following styles:

```css
page-break-before: type
page-break-after: type
```

The `type` style attribute has the following values:

- `always`, to always place a page break before or after the element
- `avoid`, to never place a page break
- `left`, to place a page break where the next page will be a left page
- `right`, to place a page break where the next page will be a right page
- `auto`, to allow the printer to determine whether or not to insert a page break
- `inherit`, to insert the page break style from the parent element

For example, if you want h1 headings to always be placed at the start of a new page, you would apply the following style in your style sheet:

```css
h1 {page-break-before: always}
```

Or, if you want block quotes to always appear on their own page, you could place a page break before and after the block quote using the style:

```css
blockquote {page-break-before: always; page-break-after: always}
```
Preventing a Page Break

Sometimes you want to keep the printer from inserting a page break inside of an element. This usually occurs when you have a long string of text that you don’t want broken into two pages. You can prevent the printer from inserting a page break by using the style

```css
page-break-inside: type
```

where `type` is auto, inherit, or avoid. To prevent a block quote from appearing on two separate pages, you could apply the following style:

```css
blockquote {page-break-inside: avoid}
```

Note that the avoid type does not guarantee that there will not be a page break within the element. If the content of an element exceeds the dimensions of the sheet, the browser will be forced to insert a page break.

Working with Widows and Orphans

Even with the three page break styles, there will be situations where a printer will have to divide the contents of an element across two pages. Although this situation is largely unavoidable, designers can control the occurrence of widows and orphans in their printed output. A **widow** occurs when only a few ending lines of an element appear at the top of a page. An **orphan** is just the opposite: it occurs when only a few beginning lines of an element appear at the bottom of a page. Leaving one or two lines “stranded” on a page either as a widow or an orphan makes the material more difficult to read and is considered poor page design. The styles to control the appearance of widows and orphans in the printout are

```css
widow: value
orphan: value
```

where `value` is the number of lines that must appear within the element before a page break can be inserted by the printer. The default value is 2, which means the widow or orphan must contain at least two lines of text. If you want to increase the size of widows and orphans to three lines for the paragraphs of your document, you could use the style declaration

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

and the printer will not insert a page break if less than three lines of a paragraph will be stranded at either the top or the bottom of a page. It’s important to note that the widow and orphan values might not always be followed. Browsers attempt to use page breaks that obey the following guidelines:

- Insert all of the manual page breaks as indicated by the `page-break-before`, `page-break-after` styles, and `page-break-inside` styles.
- Avoid inserting page breaks where indicated in the style sheet.
- Break the pages as few times as possible.
- Make all pages that don’t have a forced page break appear to have the same height.
- Avoid page breaking inside a block-level element that has a border.
- Avoid breaking inside a table.
- Avoid breaking inside of a floating element.

Only after attempting to satisfy these constraints are the Web page designer’s recommendations for the widow and orphan styles applied.
You can combine all of the various page styles described above to provide the greatest control over the appearance of your printed document. The following set of styles shows how to create a style for the blockquote element that places each block quote on a separate 8.5 \times 11 sheet of paper in landscape orientation:

@page quote_page {8.5in 11in landscape}
blockquote {page: quote_page; page-break-before: always;
page-break-inside: avoid;
page-break-after: always}

Browser support for the various CSS print styles is very uneven, so you should always test your print styles on a wide variety of browsers and operating systems.

### Reference Window | Working with Print Styles

- To define a page box for a printout that indicates the page size, margins, and orientation, use the declaration

  @page \{styles\}

  where styles are the styles that define the page.

- To set the page size and orientation, use the style

  size: width height \(orientation\)

  where width and height are the width and height of the page, and orientation is the orientation of the page (portrait or landscape).

- To insert a page break before an element, use the style

  page-break-before: type

  where type is always (to always place a page break), avoid (to never place a page break), left (to force a page break where the succeeding page will be a left page), right (to force a page break where the succeeding page will be a right page), auto (to allow the browser to determine whether or not to insert a page break), or inherit (to inherit the page break style of the parent element).

- To insert a page break after an element, use the style

  page-break-after: type

  where type has the same values as the page-break-before style.

- To apply a page break inside an element, use the style

  page-break-inside: type

  where type is auto, inherit, or avoid.

Now that you’ve seen how to insert page breaks into printed output, you are ready to insert a break into the printed version of the Samples page. Recall that Kathy wants the list of notes to appear on one page and the scrapbooking sample to appear on another. To do this, you can either place a page break after the third callout note or place a page break before the inline image of the scrapbooking sample. You decide to place a page break before the image. This will enable you to insert additional callout notes later without having to revise the page break structure. The sample image has been placed within a div container element with the id sample_image. To ensure that this container will start on a new page, you’ll add the style below to the print.css style sheet. Kathy also wants the image centered horizontally on the page. The complete style for the #sample_image selector is:

#sample_image {page-break-before: always;
    text-align: center}
Kathy also wants the sample image itself resized to better fit the size of the page. She suggests you increase the size of the printed image to 7 inches wide by 9.1 inches tall. Because the img element for the sample image is nested within the #sample_image div container, you can set the size using the following style:

```
#sample_image img {width: 7in; height: 9.1in}
```

Add both of these styles to the print.css style sheet.

---

**To complete the print.css style sheet:**

1. As shown in Figure 4-53, add the following styles to the bottom of the `print.css` style sheet:

   ```
   #sample_image {page-break-before: always;
   text-align: center}
   #sample_image img {width: 7in; height: 9.1in}
   ```

   ![Final print.css style sheet](image)

2. Close the file, saving your changes.

3. Reload the `samples.htm` file in your Web browser.

4. Either print the Web page or use the Print Preview feature of your Web browser to view the layout and design of the printed version of the document. As shown in Figure 4-54, the printed version covers 2 pages, with the list of notes on one page and the sample image resized and centered on the second page.
5. Close your Web browser and any other programs and files.
Although CSS allows you to create styles for handheld devices such as cell phones, PDAs, and MP3 players, effectively translating a large Web page into a smaller space is not easy. Some handheld devices support screens only up to 120 pixels wide; so you might quickly find your graphics-intensive Web page does not translate well into a portable world.

In general, if you want your Web page to be accessible to handheld devices, you should avoid using decorative images, and you should always specify alternative text for your graphic images. Also avoid floating elements. In the small, confined space of a handheld device, a floating element can behave unpredictably and ruin your page layout. Instead try to limit your page layout to a single column.

Use relative units such as the em unit and percent values to set the size of your fonts and block-level elements. If you must use pixels to specify a margin or padding size, try to keep your sizes within 5 pixels. Using larger pixel values such as 15 or 20 pixel widths can have an unpredictable effect on your page.

Finally, you have to pick and choose the features that are the most crucial to your Web page. A long list of links, while useful on a computer screen, can be distracting and difficult to navigate in a portable browser. Use the display:none style to control which elements will be sent to handheld devices.

Support for handheld browsing is still in its infancy, so don’t be surprised to find a great deal of variation in the support for your HTML and CSS code among the various portable devices.

You’ve completed your work on the Samples page for the Online Scrapbooks Web site, and you’ll be able to apply what you’ve learned about print styles to the other pages in the site. At the moment, most browsers support few of the page styles other than page breaking. This is sure to change in the future, however, as Web pages expand beyond the limitations of the computer screen into new media. Kathy finds this an exciting prospect, providing the opportunity to advertise the company to a whole new set of potential customers.

Session 4.3 Quick Check

1. What attribute would you add to an embedded or external style sheet link to apply a style sheet to a mobile phone?
2. Which media types belong to the continuous/visual group?
3. Which media types would be most appropriate for Web browsers designed for the visually impaired?
4. What is the difference between the display:none and visibility:hidden styles?
5. Specify a style to set the page size of the printed document to 11 inches wide by 14 inches high in landscape orientation with a 1.5-inch margin.
6. Specify the style to insert a page break before every h1 heading in your document.
7. In page design, what is a widow? What is an orphan?
In this tutorial, you learned how to use Cascading Style Sheets to create interesting and flexible layouts and designs. The first session examined different types of CSS selectors, providing the Web author flexibility in creating and applying specific design styles. The first session also explored how to apply styles to unordered lists. It concluded by examining how to use CSS to create rollover effects and drop caps. The second session focused on CSS positioning styles and explored how to use CSS positioning to place elements in absolute and relative coordinates. The final session looked at applying styles to media other than computer screens, focusing on creating styles for printed output. In the session you learned how to work with page flow within printed materials by controlling the placement of page breaks before, after, and within page elements.

**Key Terms**

<table>
<thead>
<tr>
<th>absolute positioning</th>
<th>media group</th>
<th>pseudo-element</th>
</tr>
</thead>
<tbody>
<tr>
<td>attribute selector</td>
<td>orphan</td>
<td>relative positioning</td>
</tr>
<tr>
<td>contextual selector</td>
<td>page box</td>
<td>rollover effect</td>
</tr>
<tr>
<td></td>
<td>pseudo-class</td>
<td>widow</td>
</tr>
</tbody>
</table>
Practice Assignments


Kathy stopped by to ask for your help in designing a new Web page to display scrapbooking samples sent in by different visitors to the Web site. The screen version of the Web page will show four new sample scrapbook pages each month, laid out on the page in a 2 × 2 grid. The print version of the same page will display enlarged versions of the four samples, printed on separate pages. Kathy also has some changes she wants you to make to the size navigation links. A preview of the design you’ll apply to the Scrapbook Gallery page is shown in Figure 4-55.

Figure 4-55
Complete the following:

1. Use your text editor to open the gallerytxt.htm, printertxt.css, and screentxt.css files from the tutorial.04\review folder included with your Data Files. Enter your name and the date in the comment section of each file. Save the files as gallery.htm, print.css, and screen.css, respectively, in the same folder. Take some time to study the content and layout of the Gallery Web page and observe how the styles in the screen.css style sheet file affects the layout and appearance of the page as it appears in your Web browser.

2. Return to the gallery.htm file in your text editor. Kathy wants you to format the appearance of the list of links by indenting links belonging to a particular group or class. To define the class of links, do the following:
   - Add a class attribute to the li elements in the list of links, placing the Home, Online Store, and About Us links in the newgroup class.
   - Place the other li elements in the list of links in the subgroup class.

3. The Gallery page contains four images of scrapbook page samples chosen for the May gallery. Scroll down to the four div container elements (marked with ids sample1 through sample4) and place each of the div containers in the samples class.

4. Save your changes to the gallery.htm file and then go to the screen.css style sheet file in your text editor. This style sheet will be used to design the layout of the Gallery page as it appears on computer screens.

5. Kathy wants to remove the bullet markers from the list of links. She also wants to change the layout of the links, moving them farther to the left and indenting links belonging to the subgroup class. To apply these styles, do the following:
   - For ul elements nested within the #links selector, change the marker style to none.
   - To move the list of links to the left on the Web page, set the size of the left margin to 15 pixels and the size of the left padding to 0 pixels.

6. The Gallery page also includes links to galleries from the months of February, March, and April. These links also appear in a list nested within the content div container. Kathy would like you to replace the bullet marker on this list with a graphical marker. To apply this style, set all the ul elements nested within the #content selector to use the marker.gif file as their bullet marker.

7. Kathy would like you to create a rollover effect for the list of links displayed on the left margin of the Web page. To create the rollover effect, add the following styles to the style sheet:
   - For links within the #links selector, remove any underlining by setting the text-decoration style to none. Do this for the link, visited, and active pseudo-classes.
   - When the mouse pointer is hovering over those links, change the font color to black and change the background color to white, and use the text-decoration style to add an underline and an overline to the link text.

8. Kathy wants the four scrapbook samples to be reduced in size and placed in a 2 x 2 grid on the Web page. To create this effect, add the following styles to the style sheet:
   - Apply absolute positioning to all elements belonging to the samples class.
   - For all img elements nested within the samples class, set the width of the image to 150 pixels, the height to 193 pixels, and the border width to 0 pixels.
All four scrapbook samples are nested within a div container with the id `samples_container`. Place this div container on the Web page using relative positioning. Set the top and left coordinates of the element to 0 pixels. Set the height of `samples_container` to 450 pixels.

Place the `#sample1` selector at the page coordinates (0, 0). Place the `#sample2` selector at the coordinates (170, 0). Place the `#sample3` selector at the coordinates (0, 220). Place the `#sample4` selector at the coordinates (170, 220).

9. Save your changes to the `screen.css` file. Load the `gallery.htm` file in your Web browser and verify that its layout resembles that shown in Figure 4-55. Confirm that the Web browser displays the correct rollover effect when you hover your mouse pointer over any of the links in the list on the left page margin.

10. Kathy also wants you to create a style sheet for printed versions of the Gallery page in which only the four scrapbook samples and their headings are shown, each on its own page. To create this style, return to the `printer.css` file in your text editor.

11. Add the following styles to the style sheet:
   - Set the page size to 8.5 × 11 inches, in portrait orientation, with a margin of 0.5 inches.
   - Prevent the display of the `#head` and `#links` selectors as well as the address, ul, and h2 elements and paragraphs nested within the `#content` selector.
   - Horizontally center all elements belonging to the `samples` class, and add a page break after every occurrence of this class of element.
   - Set the font size of h3 headings nested within the `samples` class to 18 points and the font family to sans-serif.
   - Set the size of img elements nested within the `samples` class to 6.5 inches wide by 8.35 inches tall. Set the border width to 0 pixels.

12. Save your changes to the `printer.css` file and return to the `gallery.htm` file.

13. Edit the link element pointing to the `screen.css` style sheet, adding an attribute that indicates that this style sheet should only be used for screen output.

14. Add a link pointing to the `printer.css` style sheet with an attribute indicating that this style sheet is used for printed output.

15. Save your changes to the `gallery.htm` file and refresh the page in your Web browser. Confirm that the appearance of the page within the browser window is unchanged. Print the Web page or use your browser’s Print Preview command to confirm that the printed version of the page displays only the four scrapbook samples and their headings.

16. Submit your completed files to your instructor.
Case Problem 1

Data Files needed for this Case Problem: h01txt.htm – h18txt.htm, hole01.jpg – hole18.jpg, next.jpg, prev.jpg, printtxt.css, willet.jpg, and willettxt.css

Willet Creek Golf Course  Willet Creek is a popular public golf course in central Idaho. You’ve been asked to work on the design of the course’s Web site by Michael Carpenter, the head of promotion for the course. Part of the Web site contains a preview of each of the course’s 18 holes, complete with yardages and shot recommendations. Each hole has been given its own Web page with a set of links to navigate from one page to another. Figure 4-56 shows a preview of the design you’ll use to show one of the pages on the golf course Web site.

Figure 4-56

Michael also wants another style sheet designed for printed output. Figure 4-57 shows a preview of the print style used with the golf course pages.
Michael has already done a lot of work in setting up the page content and has even applied a few CSS styles to the page elements. He needs you to complete the task, getting the Web pages ready for the next round of golfers.
Complete the following:

1. In your text editor, open the files h01txt.htm through h18txt.htm from the tutorial.04\case1 folder included with your Data Files. Enter your name and the date in the comment section of each file and save the files as h01.htm through h18.htm in the same folder.

2. Use your text editor to open the printtxt.css and willettxt.css files, also from the tutorial.04\case1 folder included with your Data Files. Enter your name and the date in each file and save them as print.css and willet.css, respectively, in the same folder. Take some time to review the contents and layout of the Web pages, paying particular attention to the use of div container tags and id attributes to mark off different sections of the document. Also take some time to review the contents of the willet.css style sheet to examine what styles Michael has already created for the Web site.

3. Return to the willet.css file in your text editor. The list of links to each page in the Web site is contained within a div container with the id hole_list. Apply the following styles to the list and the links it contains:
   - Display each link in a bold, white font, with a font size of 1.25 em.
   - When a mouse pointer hovers over a link, change the background color to yellow and the font color to black.
   - Display each li element within the #hole_list selector as an inline element with a margin of 0 pixels, a padding space of 0 pixels above and below the element, and padding space of 5 pixels to the left and right.

4. The hole_stats div container stores statistical information about each hole, providing the par score, distance, and handicap value for men and women. Apply the following styles to this element:
   - Use absolute positioning to place the div container at the page coordinates (0, 115).
   - For h2 elements nested within the container, set the font size to 1 em and the left margin to 5 pixels. Set the size of the other margins to 0 pixels.
   - Remove the list markers from the unordered list within the container. Also, set the left margin of the list to 10 pixels, the left padding to 0 pixels, and the top margin to 0 pixels. Display the unordered list in a 0.7 em size font.
   - Display the contents of the strong element within the container in a yellow font.

5. The hole_summary div container contains a text summary of the hole and how to play it. Add a style to place the container at the coordinates (120, 115). Use absolute positioning.

6. The hole_image div container stores an image of the hole. Use absolute positioning to place this element at the coordinates (10, 5).

7. The hole_description div container contains a text description of the hole. Use absolute positioning to place this element at the coordinates (100, 5).

8. Within the hole-description container is a span element that contains the hole number. Michael would like this number to appear as a drop cap. Create this effect by floating the span element on the left margin with a line height of 0.75 with 2 pixels of padding on the right. Set the font color to green and the font size to 300%.

9. At the bottom of each page are a pair of flag images that can be clicked to move to the next hole on the course. The images are nested within a div container with the id flags. Apply the following styles to the div container and the images it contains:
   - Set the border width of img elements within the flags container to 0 pixels.
10. Save your changes to the `willet.css` file. Go to the `h01.htm` through `h18.htm` files in your text editor and link each file to the `willet.css` style sheet, specifying that the sheet is to be used with screen and tv media. Save your changes to each file, and then view the Web site in your browser. Verify that the layout resembles that shown in Figure 4-57 and that the list of links at the top of the page has a rollover effect.

11. Return to the `print.css` file in your text editor. Add the following styles to the style sheet:
   - Set the page size to 8.5 × 11 inches in portrait orientation.
   - Set the font family of the h1, h2, and h3 headings to sans-serif.
   - Horizontally align the contents of the `#head` selector.
   - Prevent the display of the `#hole_list` and `#flags` selectors.
   - Float the `#hole_stats` selector on the left margin with a 0.2-inch right margin and 0.1 inches of padding. Add a 0.1-inch double black border to the right and bottom of the element.
   - Set the font size of h2 elements nested within the `#hole_stats` selector to 12 points.
   - Use absolute positioning to place the `#hole_image` selector 3 inches from the top of the page and 3 inches from the page's left margin.
   - Set the width of img elements within the `#hole_image` selector to 3.54 inches wide by 5 inches tall.
   - Display the span element nested within the `#hole_description` selector as a block-level element. Display the text in a 20-point bold sans-serif font.
   - Use the before pseudo-element to place the text “Hole” directly before the content of the span element.

12. Save your changes to the `print.css` file. Return to the `h01.htm` through `h18.htm` files in your text editor. Link each file to the `print.css` style sheet, indicating that this style sheet is used for print media.

13. Print `h01.htm` (or use the Print Preview feature on your Web browser) to verify that the layout of the printed Web page resembles that shown in Figure 4-57. (Note: Internet Explorer does not support the before pseudo-element, so in this browser you will not see the word “Hole” next to the hole number in the printed version of the page.)

14. Submit your completed files to your instructor.

---

**Challenge**

Test your knowledge of Web site design by completing a Civil War history page.

**Case Problem 2**

Data Files needed for this Case Problem: cwlogo.gif, cwpagetxt.htm, cwtxt.css, and tan.jpg

**Civil War Studies**  Adanya Lynne is a professor of military history at Ridgeview State College in Bartlett, Tennessee. She has been working on a Web site for a course she is preparing in Civil War studies. Professor Lynne has already created some sample pages and done work on the design and layout, but she needs your help in completing the project. She would like to create a list of links in an outline format by nesting one ordered list inside of another. She’s also interested in using CSS to create a drop-shadow effect on the main topic headings on her pages. To test your design, you’ll create a style for a page containing the text of Lincoln’s second inaugural address. A preview of the page you’ll create for Professor Lynne is shown in Figure 4-58.
Complete the following:

1. Use your text editor to open the cwpagetxt.htm and cwtxt.css files from the tutorial.04\case2 folder included with your Data Files. Enter your name and the date in the comment section of each file. Save the files as cwpage.htm and cw.css, respectively, in the same folder. Take some time to examine the contents and structure of the HTML file and the external style sheet.

2. Return to the cwpage.htm file in your text editor. Create a link to the cw.css style sheet. You do not have to specify a media attribute.

3. Scroll down the file and locate the h2 heading “Lincoln’s Second Inaugural.” Directly below this heading, insert another h2 heading containing exactly the same text but with the class name shadow.

4. Go to the paragraphs within the article div container. Give the first paragraph the class name first_para and give the remaining paragraphs the class name following_para.

5. Save your changes to the cwpage.htm file and then return to the cw.css file in your text editor.

6. The page_content div container contains the entire contents of the Web page. Add the following styles for the container:
   - Use relative positioning to place the container with top and left coordinates of 0 pixels.
   - Set the width of the container to 95% of the width of the document window.
   - Insert styles to set the minimum width of the container to 800 pixels and the maximum width to 1000 pixels.

7. The linkList div container contains the list of links in Adanya’s Web site. Use absolute positioning to place this element at the coordinates (5, 140). Set the width to 280 pixels. Add a 1-pixel-wide solid black border to the right edge of the element.

8. Remove the underlining from the links nested within the linkList container, and change the font color to black. If a mouse pointer hovers over any of the links, have the browser underline the link text.
9. The links appear in a set of nested ordered lists. Adanya wants these links to appear in an outline format. To create the outline, do the following:
   - Display the ol element using uppercase Roman numerals as the bullet marker. Set the font size to 0.9 em.
   - Display ol elements nested within another ordered list using uppercase letters as bullet markers.
   - Display ol elements nested within two levels of ordered lists using decimal numbers as bullet markers.

10. Use absolute positioning to place the article div container at the coordinates (320, 190).

11. You can create a drop-shadow around a heading by duplicating the heading and then offsetting one heading from another. You’ve already duplicated the heading in the cwpage.htm file; complete the drop-shadow by applying the following styles:
   - Use absolute positioning to place the h2 element at the coordinates (320, 125). Set the font color of the heading to the RGB value (237, 227, 178) and the z-index value to 2.
   - Directly below the h2 style, insert a style to place the element belonging to the shadow class at the coordinates (321, 126), once again using absolute positioning. Set the font color to black and the z-index value to 1.

12. Adanya wants you to create drop caps and a special first line style for the paragraphs that contain the text of Lincoln’s second inaugural address. Add the following styles to the style sheet:
   - Display the first line of the paragraph belonging to the first_para class in small capital letters.
   - Float the first letter of the “first_para” paragraph on the left margin of the paragraph with top and left margin values of 0 pixels and right bottom margins of 5 pixels. Set the font size to 300% and the line height to 0.75.
   - Use the text-indent style to indent the paragraphs belonging to the following_para class by 10 pixels each.

13. Save your changes to the style sheet.

14. Load the cwpage.htm file in your Web browser and verify that the layout matches that shown in Figure 4-58. Confirm that the Lincoln’s Second Inaugural heading appears with a drop-shadow. Verify that the list of links appears in outline form and that links within the outline display a rollover effect.

15. If you have a large screen monitor and access to Firefox, Opera, or another Web browser that supports minimum and maximum widths, resize your browser window and verify that the width of the page content does not exceed 1000 pixels or fall below 800 pixels even as you resize the browser window. (Note: Internet Explorer does not support minimum and maximum width styles.)

16. Submit your completed files to your instructor.
Challenge | Case Problem 3

Broaden your knowledge of CSS styles by creating an interactive map for a national park Web site.

Data Files needed for this Case Problem: image0.jpg – image9.jpg, longstxt.htm, and lpmap.jpg

**Longs Peak Interactive Map**  Longs Peak is one of the most popular attractions of Rocky Mountain National Park (RMNP). Each year during the months of July, August, and September, thousands of people climb Longs Peak by the Keyhole Route to reach the 14,255-foot summit. Ron Bartlett, the head of the RMNP Web site team, has asked for your help in creating an interactive map of the Keyhole Route. The map will be installed at electronic kiosks in the park’s visitor center. Ron envisions a map with 10 numbered waypoints along the Keyhole Route, displaying photos and text descriptions of each waypoint when a mouse pointer hovers over its corresponding numbered point. Figure 4-59 shows a preview of the online map with the first waypoint highlighted by the user.

**Longs Peak Online Trail Map**

At 14,255 feet, Longs Peak towers above all other summits in Rocky Mountain National Park. The summer is the only season in which the peak can be climbed by a non-technical route. Early mornings break calm, clouds build in the afternoon sky, often exploding in storms of brief, heavy rain, thunder and dangerous lightning. Begin your hike early, way before dawn, to be back below timberline before the weather turns for the worse.

The Keyhole Route, Longs Peak’s only non-technical hiking pathway, is a 16 mile round trip with an elevation gain of 4,850 feet. Though non-technical, the Keyhole Route is still challenging and is not recommended for those who are afraid of heights or exposed ledges. Hikers should be properly outfitted with clothing, food and water. Use caution when ascending or descending steep areas. Don’t be afraid to back down when bad weather threatens.

Move your mouse pointer over the numbered landmarks in the map to preview the hike.
Ron also wants to create a print version of the waypoint descriptions. Users can print the map elsewhere on the park’s Web site, so he wants the print style sheet to display just the list of waypoints in a bulleted list.

Complete the following:

1. Use your text editor to open the longstxt.htm file from the tutorial.04\case3 folder included with your Data Files. Enter your name and the date in the comment section at the top of the file. Save the file as longs.htm in the same folder. Take some time to study the contents of the file. Notice that the waypoint descriptions are nested within the online_map div container. Each waypoint description has been placed in a separate div container with the class name notes and id names ranging from point0 to point9.

2. To create a rollover effect for the nine waypoints, you need to mark the waypoint contents as hyperlinks. Within each of the nine div containers, enclose the img element and paragraph element within a single <a> tag. Point each link to the longs.htm file so that if a user clicks the link, it will simply refresh the current Web page.

3. Add a style to the embedded style sheet at the top of the file to remove underlining from all hypertext links.

4. Currently the online_map div container does not display the Longs Peak map. Add the following style to the embedded style sheet to display the map:
   - Set the width of the container to 600 pixels wide by 294 pixels high.
   - Add a 1-pixel-wide solid black border to the container.
   - Apply the lpmap.jpg graphic file as the background image.
   - Use relative positioning to place the container on the page. Set the top and left coordinates to 0 pixels.

5. In the embedded style sheet, add the following styles for all of the div containers belonging to the notes class:
   - For paragraphs nested within each note, set the font size to 8 points and the margin to 5 pixels.
   - Set the font color to yellow for each span element nested within a note.

6. For each link nested within a notes class element, apply the following styles:
   - Set the width and height of the link to 20 pixels.
   - Hide any content that overflows the boundary of the link.
   - Set the background color to blue and the font color to white.
   - Set the z-index value to 1.

7. When the mouse hovers over a link within a notes class element, have the browser apply the following style:
   - Change the width to 150 pixels and the height to 170 pixels.
   - Change the overflow property to visible.
   - Set the z-index value to 2.

8. Use absolute positioning to place the link nested within the #point0 selector at the coordinates (560, 60).

9. Repeat Step 8 for the nine remaining waypoints:
   - #point1 at (277, 90)
   - #point2 at (175, 0)
   - #point3 at (110, 115)
   - #point4 at (55, 165)
   - #point5 at (5, 180)
10. Go to the top of the file and add a media attribute to the embedded style sheet indicating that the sheet is designed for screen and tv media.

11. Save your changes to the file and then open longs.htm in your Web browser. Verify that the placement of the waypoints follows the locations shown in Figure 4-59. Confirm that when you hover your mouse over each of the nine waypoints, a description of the waypoint appears on the top of the trail map.

12. Return to the longs.htm file in your text editor. Create a new embedded style sheet designed for print media.

13. Scroll down to the “Longs Peak Online Trail Map” h2 heading and enclose the text “Online Trail Map” within a span element.

14. Add the following styles to the embedded print style sheet:
   - Set the font family for the page body to sans-serif.
   - Remove underlining from all hypertext links.
   - Do not display the #instructions selector, the span element nested within the h2 element, or the img element nested within elements belonging to the notes class.
   - Display the notes class of elements as list items with a disc marker. Set the margin to 20 pixels.
   - Change the display property of paragraphs nested within the notes class of elements to inline.
   - Display span elements nested within the notes class of elements in a bold font.

15. Add a style that inserts the text string “Trail Itinerary” after the h2 heading.

16. Save your changes to the file.

17. Refresh the longs.htm file in your Web browser. By either printing the page or viewing the page within the Print Preview window, confirm that the printed page only shows a bulleted list of the waypoint descriptions. If you are running Firefox, Opera, or Safari, confirm that the heading at the top of the page reads “Longs Peak Trail Itinerary.” (If you are using Internet Explorer, the title will simply read “Longs Peak.”)

18. Submit your completed files to your instructor.

**EXPLORE**

Test your knowledge of CSS and HTML by creating a Web page design for a children’s choir.

**Case Problem 4**

Data Files needed for this Case Problem: bizet.jpg, bizetbio.txt, bizellist.txt, mozart.jpg, mozartbio.txt, mozartlist.txt, puccini.jpg, puccinibio.txt, puccinilist.txt, verdi.jpg, verdibio.txt, verdilist.txt, wagner.jpg, wagnerbio.txt, and wagnerlist.txt

Gresham Children’s Choir  Faye Dawson is an instructor for Gresham Children’s Choir in Gresham, Oregon. The choir is a chance for talented youth to perform and to learn about music history. Faye is working on a Web site describing the history of opera. She’s asked for your help in creating a design. Faye has provided you with information on five different composers: Bizet, Mozart, Puccini, Verdi, and Wagner. For each composer, she’s given you an image file containing the composer’s picture, a text file listing the composer’s works, and a text file containing a biographical sketch. Use this information to design your Web site. You may supplement these files with any other material you think will enhance your site’s design.
Complete the following:

1. Use your text editor to create four HTML files named bizet.htm, mozart.htm, puccini.htm, verdi.htm, and wagner.htm, placing them in the tutorial.04\case4 folder included with your Data Files. Enter your name and the date in a comment section of each file. Include any other comments you think will aptly document the page’s purpose and history.

2. Use the provided text files and image files to create a Web page describing each composer’s life and accomplishments. Include hypertext links between the five composer Web pages.

3. Create an external style sheet named gresham.css for your Web site. Insert a comment section in the style sheet file that includes your name and the date as well as other comments that describe the style sheet.

4. The content of the gresham.css style sheet is up to you, but it must include the following features:
   - Styles that use contextual selectors
   - A style that uses a pseudo-element and a pseudo-class
   - Styles that use positioning styles (either absolute or relative)
   - A style that creates or modifies an ordered or unordered list
   - A style to create a rollover effect

5. Create another style sheet named printer.css containing styles for a printed version of the pages in your Web site. Add appropriate comments to the different parts of your style sheet.

6. Test your Web site on a variety of browsers to ensure your design works under different conditions.

7. Submit your completed files to your instructor.

---

### Review | Quick Check Answers

#### Session 4.1

1. p span {font-style: italic}
2. p > span {font-style: italic}
3. h1 + h2 {font-style: italic}
4. .newsAlert {font-weight: bold}
5. span.newsAlert {font-weight: bold}
6. a:hover {background-color: yellow}
7. a:hover {background-image:url(hover.jpg)}
8. blockquote:first-letter {color: red}
9. blockquote:first-line {color: red}

#### Session 4.2

1. #logo {position: absolute; top: 75px; left: 150px}
2. #logo {position: relative; top: 25%; left: 10%}
3. span.highlight {position: relative; top: -10px}
4. a:hover {position: relative; top: 5px}
5. #links {position: fixed; top: 50px; left: 10px}
6. blockquote {width: 70%; min-width: 250px; max-width: 650px}
7. blockquote {height: 25%; overflow: auto}
8. No. It will only be on top of other elements for which it shares a common parent.
Session 4.3

1. media = "handheld"
2. screen, tv, tty
3. aural, Braille, embossed
4. The display:none style hides the element and removes it from the document flow.
   The visibility:hidden style hides the element, but does not remove it from the document flow.
5. @page {11in 14in landscape ; margin 1.5 in}
6. h1 {page-break-before: always}
7. A widow occurs when a page break divides a block of text, leaving only one or two lines of text on the succeeding page. An orphan occurs when the page break occurs near the start of the block of text, leaving only one or two lines of text on the first page.

Ending Data Files

<table>
<thead>
<tr>
<th>Tutorial.04</th>
<th>Tutorial</th>
<th>Review</th>
<th>Case1</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>print.css samples.htm scraps.css start.htm + 4 graphic files</td>
<td>gallery.htm printer.css screen.css + 7 graphics files</td>
<td>h01.htm - h18.htm print.css willet.css + 21 graphic files</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Case2</th>
<th>Case3</th>
<th>Case4</th>
</tr>
</thead>
<tbody>
<tr>
<td>cw.css cwpage.htm + 2 graphic files</td>
<td>longs.htm + 11 graphic files</td>
<td>bizet.htm mozart.htm printer.css puccini.htm verdi.htm wagner.htm + 5 graphic files</td>
</tr>
</tbody>
</table>