TUTORIAL 3

Designing a Web Page with CSS

Creating a Web Site for a Rural Farm

OBJECTIVES

Session 3.1
- Explore the history and theory of CSS
- Define a style rule
- Study style precedence and inheritance
- Apply color using CSS
- Explore CSS3 color extensions

Session 3.2
- Use contextual selectors
- Work with attribute selectors
- Apply text and font styles
- Install a Web font

Session 3.3
- Define list styles
- Use pseudo-classes and pseudo-elements
- Create a rollover effect

Case | *Sunny Acres*

Tammy Nielsen and her husband, Brent, live and work at Sunny Acres, a 200-acre farm near Council Bluffs, Iowa. Over the past 25 years, the Nielsen family has expanded the farm's operations to include a farm shop, which sells fresh produce, baked goods, jams, jellies, and gifts; a pick-your-own garden, which operates from May through October and offers great produce at discounted prices; a petting barn with over 100 animals and the opportunity to bottle-feed the baby animals; a corn maze with over 4 miles of twisting trails through harvested corn fields; and a Halloween Festival featuring the corn maze haunted with dozens of spooky effects and tricks. The farm also hosts special holiday events during the winter.

Tammy created a Web site for Sunny Acres several years ago to make information about the farm easily accessible to her customers. The Web site has become outdated, so Tammy would like to enliven it with a new design based on the latest elements and styles from HTML and CSS. Tammy's knowledge of HTML and Web styles is limited, so she's come to you for help in creating a new look for the Sunny Acres Web site.

STARTING DATA FILES

tutorial

haunttxt.htm
hometxt.htm
mazetxt.htm
pettingtxt.htm
producetxt.htm
sa_stylestxt.css
+ 1 style sheet
+ 7 graphic files
+ 4 Web fonts
+ 1 text file
modernizr-1.5.js

review

holidaytxt.htm
hs_stylestxt.css
+ 1 style sheet
+ 3 graphic files
+ 4 Web fonts
+ 1 text file
modernizr-1.5.js

case1

crypttxt.htm
c_stylestxt.css
+ 5 HTML files
+ 1 style sheet
+ 3 graphic files
modernizr-1.5.js

case2

bmtourtxt.htm
mw_stylestxt.css
+ 1 style sheet
+ 2 graphic files
modernizr-1.5.js

case3

civilwartxt.htm
cw_stylestxt.css
+ 1 style sheet
+ 3 graphic files
modernizr-1.5.js

case4

choirtxt.htm
gcc_stylestxt.css
+ 1 style sheet
+ 2 graphic files
+ 4 Web fonts
+ 1 text file
modernizr-1.5.js

demo

demo_color_names.htm
demo_css.htm
+ 3 graphic files

SESSION 3.1 VISUAL OVERVIEW

Style comments provide information about the style sheet.

```
/*
   Sunny Acres Style Sheet

   Author: Tammy Nielsen
   Date:   3/1/2014
*/


/* Body styles */

body {
   background-color: white;
   font-family: Verdana, Geneva, sans-serif;
   line-height: 1.4em;
}
```

The appearance of the Web page is determined by the styles in a style sheet.

Sunny Acres

HOME

Autumn Fun

Scary Good

Meet the Animals

For your Tastebuds

The Farm Sh

The Sunny Acres Farm Shop aims to of highest quality fresh produce. You can your own or buy it in our shop. Set ami acres of outstanding natural beauty on beautiful rolling hills northeast of Coun Bluffs, the Farm Shop is easily reached Highway G, with easy access from Inte 80.

The Farm Shop was established over 25 ago with great success. Our products h won numerous awards at local festival fairs. We also cater to local supermarke the Council Bluffs/Omaha area. Look fo products every Saturday morning from to October at the Council Bluffs Farmer

Hours

- Monday - Friday: 9 am - 5 pm
- Saturday: 9 am - 3 pm
- Pick Your Own Produce is availabl
- The Farm Shop is open year-roun

Products

- Freshly baked breads and quiches
- High quality meats
- Cheese and other dairy products
- Freshly-picked fruits and vegetabl
- Canned goods and preserves

SUNNY ACRES ☀ TAMMY & BRENT NIELSEN ☀ 1977 HIGHWAY G ☀

```
h2 {
   background-color: rgb(0, 165, 0);
   color: rgba(255, 255, 255, 0.8);
}
```

Hou

The color is 80% opaque.

Color values using the rgba or hsla properties can include opacity to create semi-transparent colors.

STYLE SHEETS AND COLOR

Every style rule needs to be enclosed in curly braces, with the style property values separated by semicolons.

The **selector** defines what element or elements are affected by a rule.

```
h1 {
color: white;
background-color: rgb(50, 69, 99);
}
```

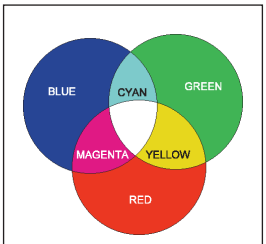
The background-color style property sets the background color.

The color style property sets the text color.

CSS supports **color names** for a select group of commonly used colors.

The **HSL model** selects color from a color wheel at varying levels of saturation and lightness.

CSS Color Models



rgb(*red*, *green*, *blue*)

red, *green*, and *blue* range from 0 (no intensity) up to 255 (highest intensity)

hsl(*hue*, *saturation*, *lightness*)

hue ranges from 0 to 360 degrees, *saturation* and *lightness* vary from 0% to 100%

pink
brown
orange
seagreen
powderblue

The **RGB model** combines red, blue, and green color of varying intensities.

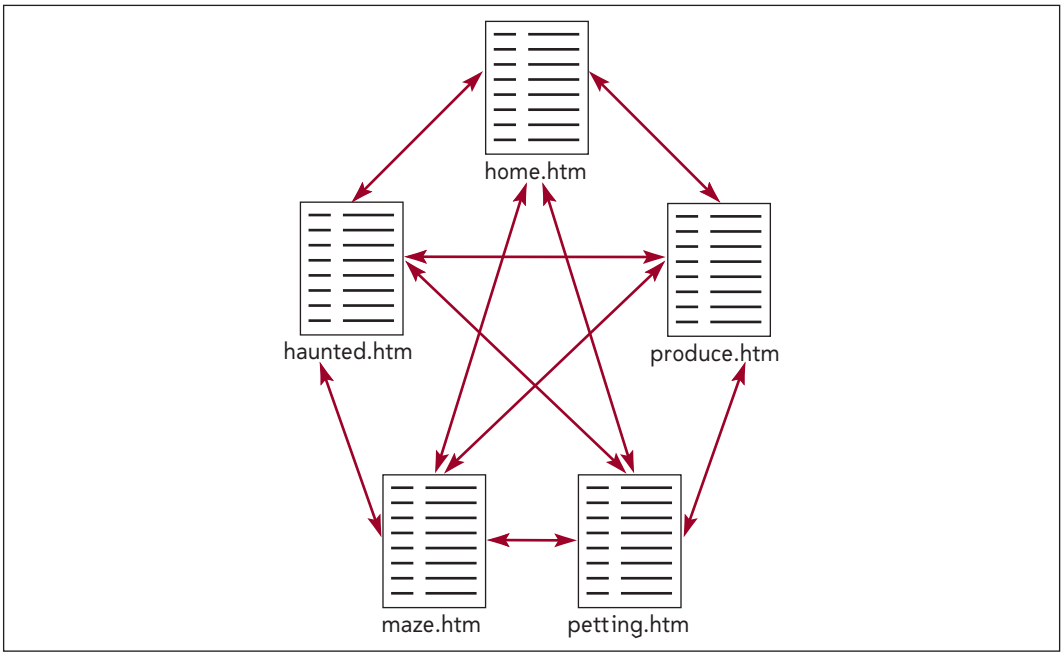
Introducing CSS

You and Tammy met to discuss her ideas for upgrading the design of the Sunny Acres Web site. She's created a few sample pages that she wants you to work with:

- *home.htm*—the home page, describing the operations and events sponsored by the farm
- *maze.htm*—a page describing the farm's corn maze
- *haunted.htm*—a page describing the farm's annual Halloween Festival and haunted maze
- *petting.htm*—a page describing the farm's petting barn
- *produce.htm*—a page describing the Sunny Acres farm shop and the pick-your-own produce garden

Figure 3-1 shows the links among these sites in the Sunny Acres storyboard.

Figure 3-1 Storyboard of the Sunny Acres Web site

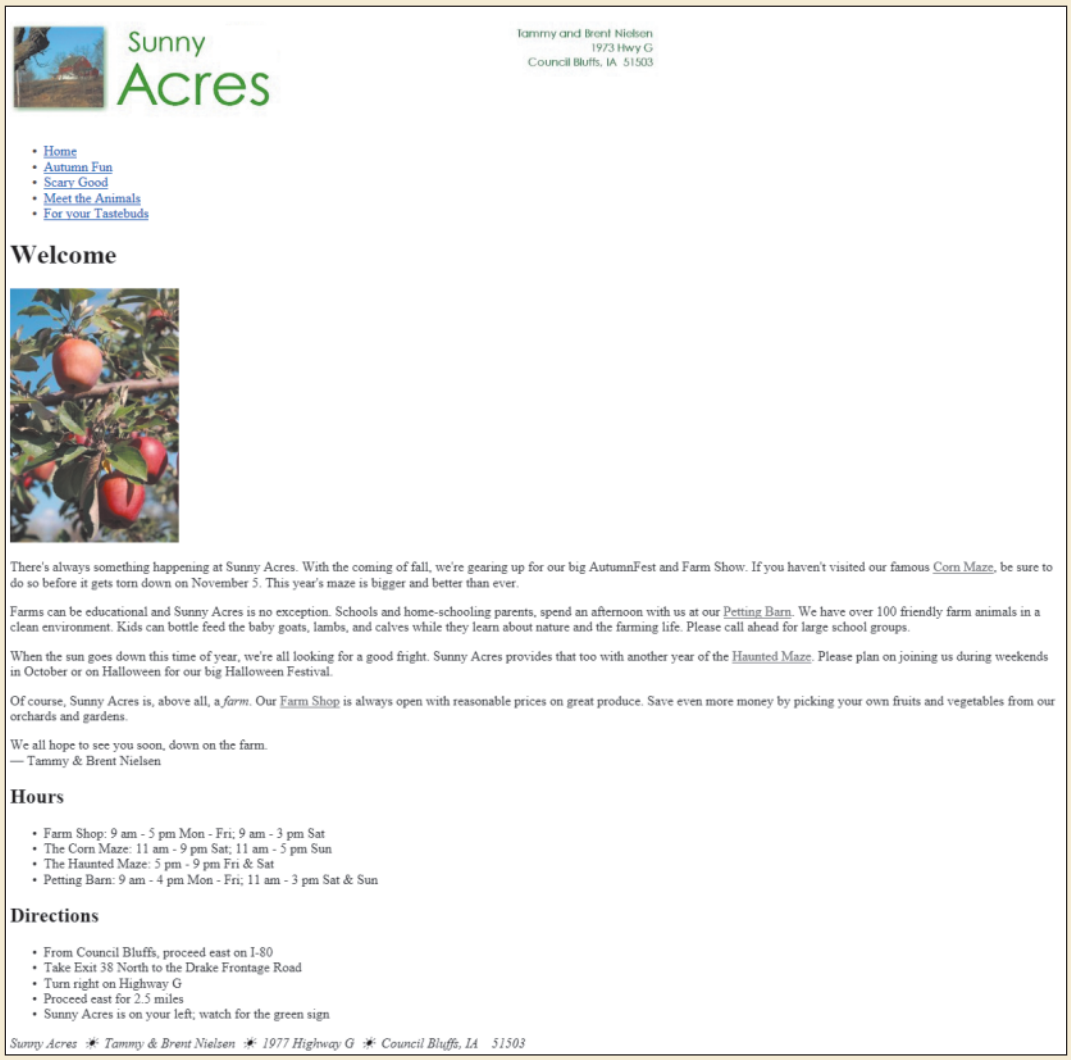


You'll start by opening these files in your text editor and browser.

To view the Sunny Acres Web pages:

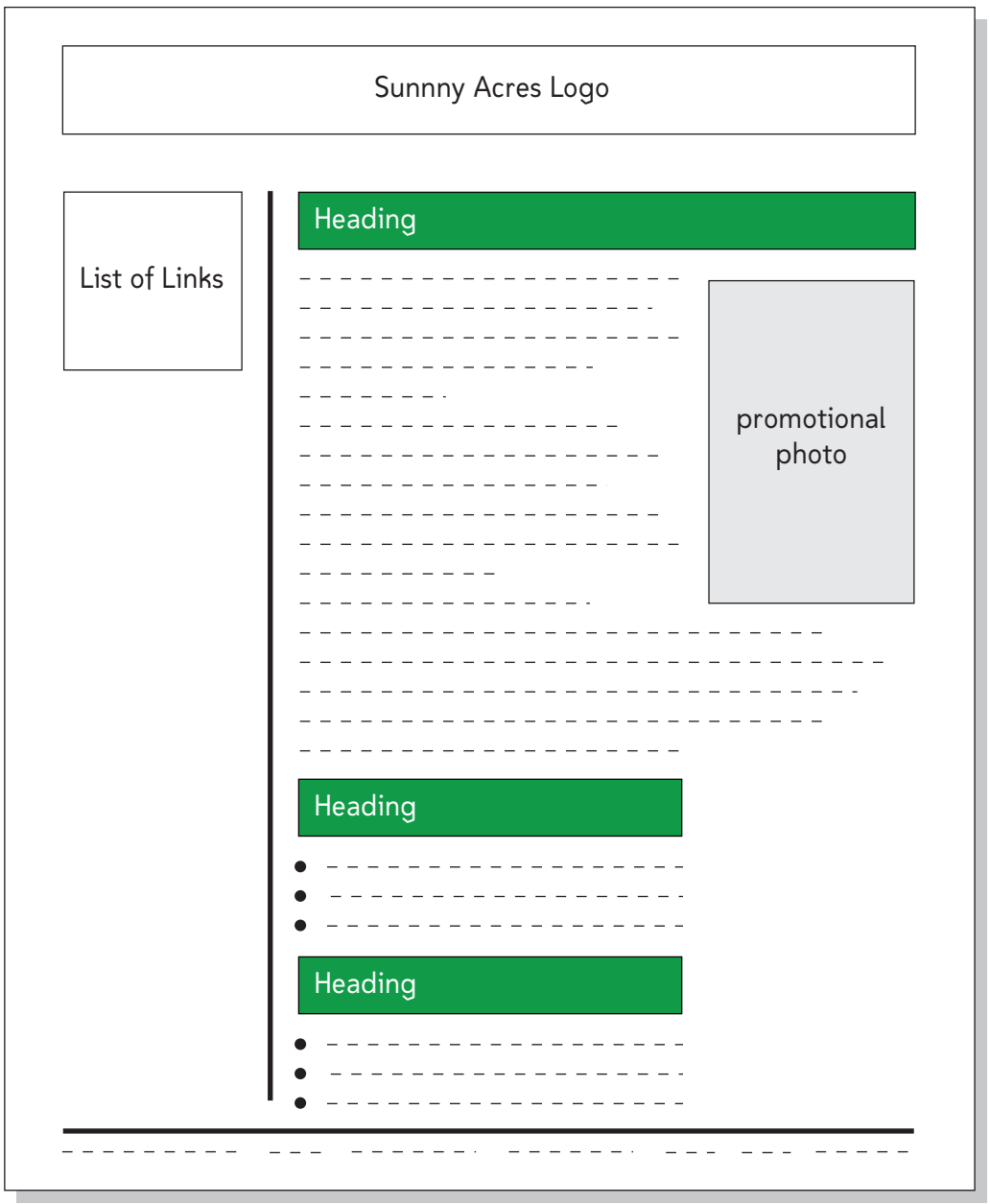
1. Use your text editor to open the **haunttxt.htm**, **hometxt.htm**, **mazetxt.htm**, **pettingtxt.htm**, and **producetxt.htm** files, located in the tutorial.03\tutorial folder included with your Data Files. Within each file, go to the comment section at the top of the file and add ***your name*** and ***the date*** in the space provided. Save the files as **haunted.htm**, **home.htm**, **maze.htm**, **petting.htm**, and **produce.htm**, respectively, in the same folder.
2. Take some time to review the HTML code within each document so that you understand the structure and content of the files.
3. Open the **home.htm** file in your Web browser, and then click the links at the top of the page to view the current appearance of the *haunted.htm*, *maze.htm*, *petting.htm*, and *produce.htm* files. Figure 3-2 shows the current layout and appearance of the Sunny Acres home page.

Figure 3-2 **Initial Sunny Acres home page**



Tammy already has created most of the content for the new and revised pages, but she has not upgraded the Web site design. She needs your help with that. In Figure 3-3, she sketches the basic design she has in mind for her pages.

Figure 3-3 **Proposed design for the Sunny Acres home page**



To apply this page format, Tammy wants you to use the design features available with CSS. Before starting, you'll review the history and concepts behind CSS.

The History of CSS

You learned in Tutorial 1 that HTML specifies a document's content and structure, but not how that document should be rendered. To render a document, the device displaying the page needs a style sheet that specifies the appearance of each page element. The style sheet language used on the Web is the Cascading Style Sheets language, also known as CSS.

The specifications for CSS are maintained by the World Wide Web Consortium (W3C); and as with HTML and XHTML, several versions of CSS exist with varying levels of browser support. The first version of CSS, called **CSS1**, was introduced in 1996 and enabled Web designers to create styles to:

- Set the font size, type, and other properties of Web page text
- Control text alignment and apply decorative elements such as underlining, italic, and capitalization
- Specify background and foreground colors of different page elements
- Apply a background image to any element
- Set the margins, internal space, and borders of grouping elements such as paragraphs and headings

CSS1 made it possible to create Web pages that had visually interesting and attractive designs and layouts. The second version of CSS, **CSS2**, was introduced in 1998, expanding the language to provide styles to:

- Position elements at specific locations on a page
- Clip and hide element content
- Design styles for different output devices, including printed media and aural devices
- Control the appearance and behavior of browser features such as scroll bars and mouse cursors

An update to CSS2, **CSS 2.1**, was introduced by the W3C in April 2002. Although the update did not add any new features to the language, it cleaned up minor errors that were introduced in the original specification. At the time of this writing, almost all aspects of CSS2 are supported by current browsers.

As browsers were implementing all of the features of CSS2, in December 2005 the W3C pressed forward to the next version, **CSS3**, which further expanded the design tools available to Web page authors. Currently still in a working draft, CSS3 adds styles for:

- Enhanced text effects, including drop shadows and Web fonts
- Semi-transparent colors and overlays
- Column-based layout
- Rounded borders, drop shadows, and box outlines
- Transformations of page elements, including scaling, skewing, and rotation

With CSS, as with HTML, Web page designers need to be aware of compatibility issues that arise not just among different versions of the language, but also among different versions of the same browser. Although it's tempting to always apply the latest and most exciting features of CSS, you should not create a situation where users of older browsers will not be able to view your Web pages.

Browser Extensions

Not content to wait for the W3C's final specifications, several browser manufacturers are creating their own extensions to the CSS language. Many of these extensions have been incorporated in the CSS3 specification. By putting forward their own extensions, these vendors are able to test and debug new styles that are still in the development stage with CSS3. You can use these browser extensions as long as you realize that they might not be supported by other browsers and you do not make their use crucial to your page's readability.

Defining a Style Rule

In every version of CSS, you apply a **style rule** containing a list of style properties to an element or a group of elements known as a selector. The general syntax of a CSS style rule is

```
selector {
   property1: value1;
   property2: value2;
   property3: value3;
...
}
```

where *selector* identifies an element or a group of elements within the document and the *property: value* pairs specify the style properties and their values. For example, to display the text of all `h1` headings in blue and centered horizontally on the page, you could use the following style rule:

```
h1 {
   color: blue;
   text-align: center;
}
```

To apply these style properties to more than one element, you specify the elements in a comma-separated list. The following style rule causes all `h1` through `h6` headings to be displayed in blue and centered on the page:

```
h1, h2, h3, h4, h5, h6 {
   color: blue;
   text-align: center;
}
```

Like HTML, CSS ignores the use of white space, so you can also enter your styles on a single line, as in the following example:

```
h1 {color: blue; text-align: center;}
```

Writing a style rule on a single line saves space, but entering each style property on a separate line often makes your code easier to read and edit. You will see both approaches used in the CSS files you encounter on the Web.

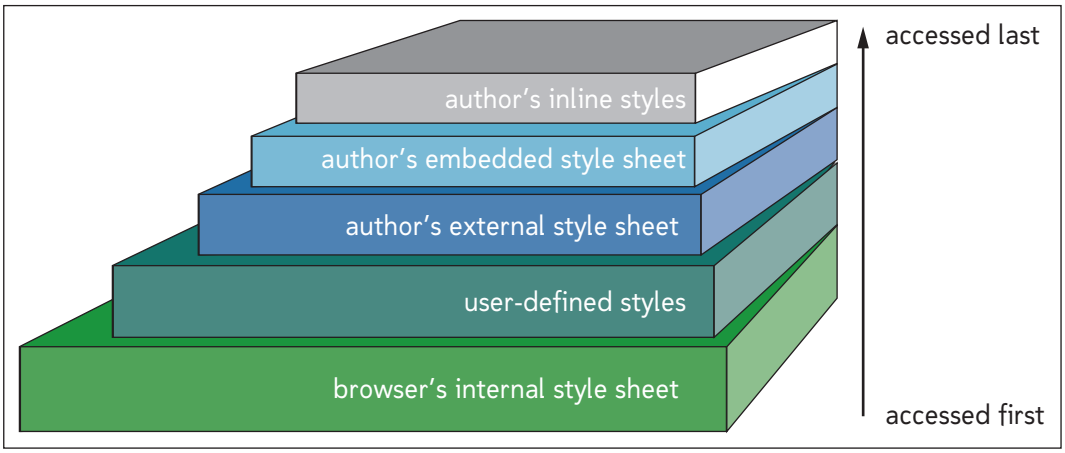
Applying a Style Sheet

TIP

You can make your style sheets easier to manage by entering the style names in alphabetical order.

The design you apply to a Web site is usually a combination of several style sheets. In general, the style sheet that is loaded last has precedence over style sheets loaded earlier. Figure 3-4 summarizes the different types of style sheets in the order they are usually installed and processed by browsers.

Figure 3-4 Order in which style sheets are interpreted

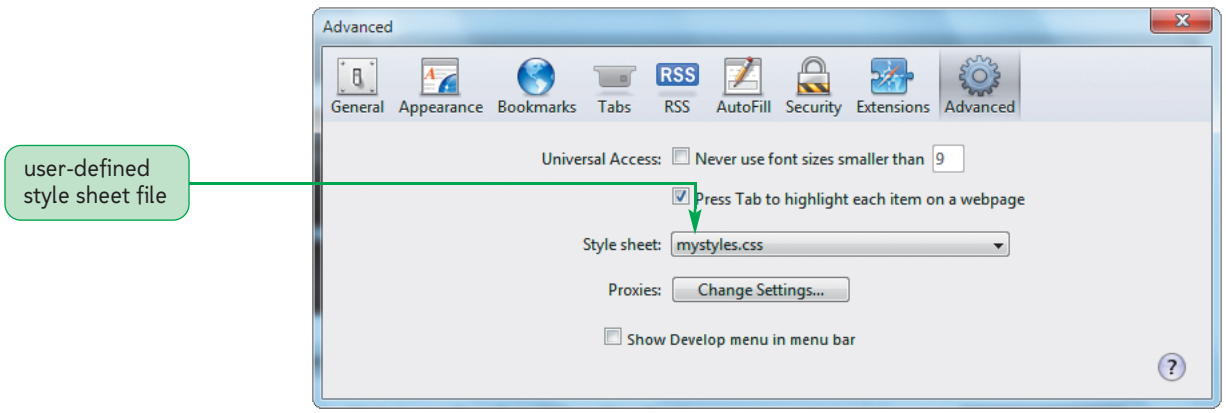


The first style sheet interpreted by the browser is the one built into the browser itself. The current appearance of the Sunny Acres Web page displayed in Figure 3-2 is based on styles applied by the browser itself that are contained within its own style rules about how headings, paragraphs, inline images and so forth, should be rendered. Browsers use similar default styles. So if you rely only on the browser's internal style sheet, your Web sites should appear alike across most browsers.

User-Defined Styles

Almost all browsers allow users to modify the default settings of the internal style sheet. For example, a user could change the font size assigned to paragraph and heading text, set foreground and background colors, and specify whether or not to display inline images. Browsers such as Internet Explorer and Safari also allow users to substitute their own style sheets for the browser's internal sheet, providing more control over how the browser renders the pages it encounters. One advantage of user-defined style sheets is that they make the Web more accessible to visually impaired users who may require larger fonts or the absence of clashing color schemes. Figure 3-5 shows the Advanced dialog box from Safari, in which users can replace the browser's internal style sheet with their own.

Figure 3-5 Choosing a user-defined style sheet in Safari



External Style Sheets

Styles set by the author of a Web page and stored in an external style sheet are loaded after internal and user-defined style sheets. You've already worked with external style sheets in the first two tutorials using CSS files created for you. Recall that an external style sheet is included by adding the `link` element

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

to the document head, where *url* is the URL of the external style sheet file. The style sheet rules in an external style sheet take precedence over any rules set in the browser's internal style sheet or in a user-defined style sheet.

Tammy already has created a style sheet for her sample pages to define how the elements should be laid out on the page. You'll apply the style rules from her *sa_layout.css* file to the Sunny Acres home page now.

To link to the layout style sheet:

1. Return to the **home.htm** file in your text editor.
2. Directly above the closing `</head>` tag, insert the following `link` element (see Figure 3-6):

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

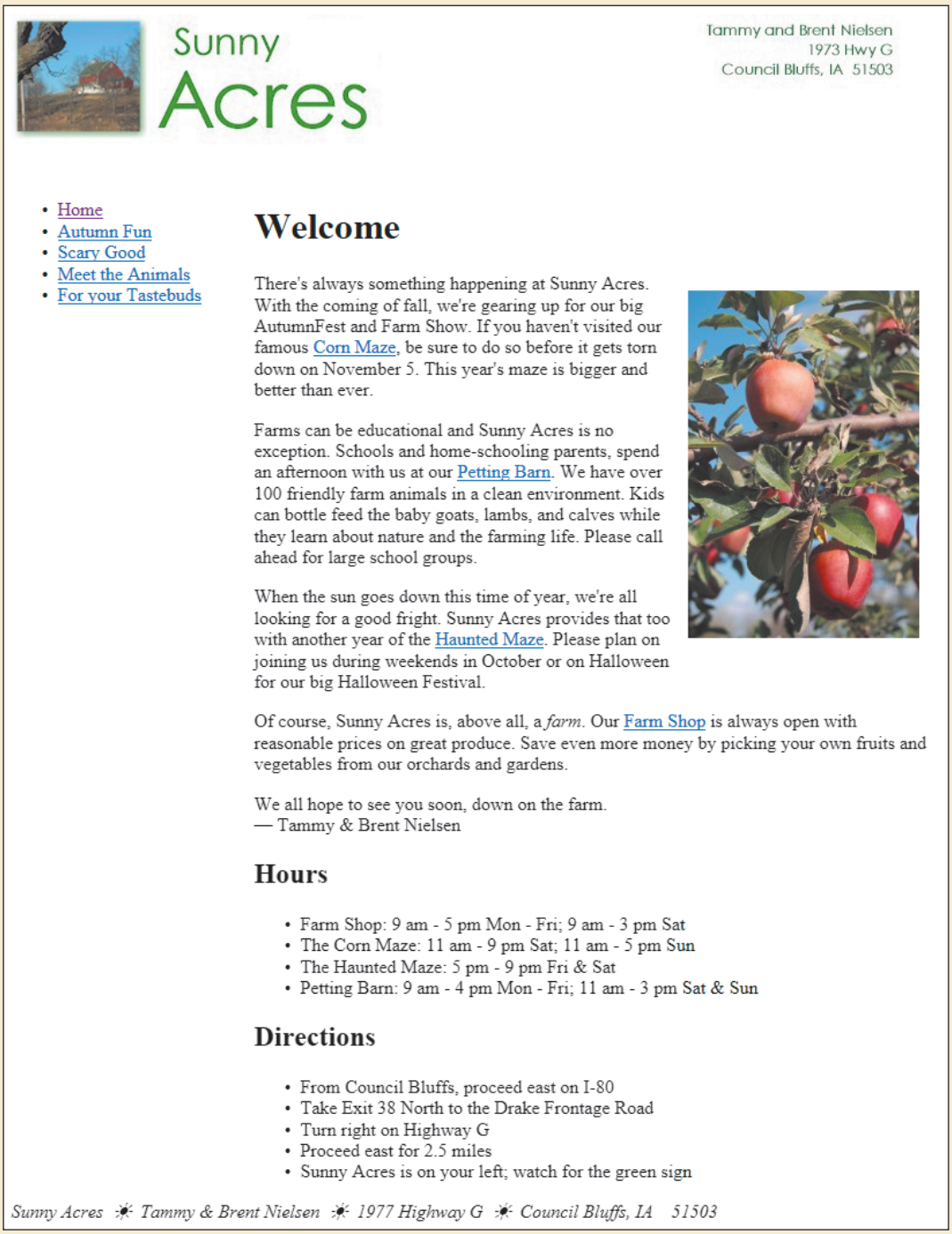
Figure 3-6 Linking to an external style sheet

style sheet file

```
    <meta charset="UTF-8" />
    <title>Sunny Acres</title>
    <script src="modernizr-1.5.js"></script>
    <link href="sa_layout.css" rel="stylesheet" type="text/css" />
</head>
```

3. Save your changes to the file.
4. Reopen the **home.htm** file in your browser. As shown in Figure 3-7, the layout of the page has been altered using the design styles present in the *sa_layout.css* file.

Figure 3-7 Sunny Acres home page using the sa_layout.css style sheet



The *sa_layout.css* style sheet changes the layout of the page, displaying the navigation links in a column on the left and the main page content in a column on the right. The width of the page also has been reduced to make the content easier to read.

INSIGHT

Importing Style Sheets

On large Web sites that involve hundreds of pages, you might decide to use different styles for different groups of pages to give a visual cue to users about where they are on the site. One way of organizing these different styles is to break them into smaller, more manageable units. The different style sheets then can be imported into a single sheet. To import a style sheet, add the command

```
@import url(url);
```

to the style sheet file, where *url* is the URL of an external style sheet file. For example, a company might have one style sheet named *company.css* that contains basic styles used in all Web pages, and another style sheet named *support.css* that only applies to Web pages containing technical support information. The following code added to a style sheet imports both files:

```
@import url(company.css);
@import url(support.css);
```

The `@import` statement must always come before any other style rules in the style sheet. When a browser encounters the `@import` statement, it imports the content of the style sheet file directly into the current style sheet, much as if you had typed the style declarations yourself.

The `@import` rule has the same impact as adding multiple `link` elements to the HTML file. An advantage of the `@import` rule is that it simplifies your HTML file (since you only need to access one style sheet file), and it places all style rules and decisions about which style sheets to include and exclude in an external file. This is an important distinction if you want to put all of your design choices in the external style sheet file, which you can then easily edit and modify without having to touch the HTML document.

Embedded Style Sheets

Another type of style sheet created by a Web page author is an embedded style sheet, in which the styles are inserted directly within the `head` element of an HTML document using the `style` element

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

where *styles* are the rules of the style sheet. For example, the following embedded style sheet applies a rule to display the text of all `h1` headings from the current document centered horizontally and in red:

```
<style type="text/css">
   h1 {
   color: red;
   text-align: center;
   }
</style>
```

The exact order in which external style sheets and embedded style sheets are processed by the browser depends on the order in which they are listed within the HTML file. The HTML code

```
<link href="sa_layout.css" rel="stylesheet" type="text/css" />
<style type="text/css">
   h1 {color: red; }
</style>
```

loads the external style sheet first and then the embedded sheet. However, if that order is switched as in the code

```
<style type="text/css">
   h1 {color: red; }
</style>
<link href="sa_layout.css" rel="stylesheet" type="text/css" />
```

then the external style sheet is processed after the embedded sheet.

TIP

Always place embedded styles after external style sheets to avoid confusion about which style sheet is loaded last.

Unlike an external style sheet, an embedded style sheet is applied only to the Web page in which it is placed. Thus, if you want to apply the same style to all of the headings on your Web site, it is more efficient and easier to manage if you define your styles only once within an external style sheet and link all of the pages to that file. If you later need to change the site design, you'll have to edit only one file, rather than dozens.

Inline Styles

The very last styles to be interpreted by the browser are inline styles, which are styles applied directly to specific elements using the `style` attribute

```
<element style="style rules"> … </element>
```

where *element* is the HTML element and *style rules* are CSS styles applied to that element. For example, the following `style` attribute is used to display the text of a specific `h1` heading in green and centered on the page:

```
<h1 style="color: green; text-align: center;">
   Sunny Acres
</h1>
```

The advantage in using inline styles is that it is clear exactly what page element is being formatted; however, inline styles are not recommended in most cases because they make changing styles tedious and inefficient. For example, if you wanted to use inline styles to format all of your headings, you would have to locate all of the `h1` through `h6` elements in all of the Web pages within the entire Web site and add `style` attributes to each tag. This would be no small task on a large Web site containing hundreds of headings spread out among dozens of Web pages, and it would be a nightmare if you had to modify the design of a large Web site that was created using inline styles.

TIP

View your Web site with and without your style sheet. It should be readable even if a user is limited to the default styles supplied by a Web browser.

However, the primary reason to not use inline styles is that you want to, as much as possible, separate document content from document design. Ideally, the HTML code and CSS styles should be so separate that one group of employees could define the page content using HTML and another group could define the page design using CSS. This isn't possible with inline styles because the code for the page design is intermingled with the code for the page content.

Exploring the Style Cascade

With the potential for many different style sheets to be applied to the same Web page, there has to be an orderly method by which conflicts between those different style sheets are resolved. CSS does this by assigning a level of importance to each style, with the most important style rule taking precedence over other competing rules.

Style Precedence and Specificity

Many factors determine how the importance of each style is calculated. But as a general rule of thumb, *all other things being equal, the more specific style is applied instead of the more general*. Thus, a style applied to a specific paragraph is given more importance than a general style applied to an entire Web page, and a style applied to a section of text within that paragraph has more importance than the style for the entire paragraph. For example, the following set of style rules would set the text color of the Web page to black, except for text within the `header` element:

```
body {color: black;}
header {color: red;}
```

Specificity is only a concern when two or more styles conflict. If the style rules involve different properties, there is no conflict and both rules are applied. If two style rules have equal specificity, and thus equal importance, then the one that is defined last in the style sheet is the one used.

Style Inheritance

An additional factor in applying a style sheet is that properties are passed from a parent element to its children in a process known as **style inheritance**. For example, to set the text color of a page to blue, you could apply the style

```
body {color: blue;}
```

and every element nested within the `body` element (which is every element on the Web page) would inherit this style. This means that the text of every heading, every paragraph, every numbered list, and so forth would be displayed in blue unless a different text color were defined for those specific elements. Thus, the style rules

```
body {color: blue;}
h1 {text-align: center;}
```

would result in the `h1` heading text appearing in blue and centered even though only the text alignment is specifically set within the style rule for the `h1` element.

Not all properties are inherited. For example, the style property above that defines the text color for the page body has no meaning if applied to an inline image.

The final rendering of any page element thus becomes the result of multiple style sheets and multiple style rules. You may have to track a set of styles as they are passed from one style sheet to another. For example, browsers typically display all `h1` headings in a large bold black font and left-aligned on the page. The following rule applied within an external style sheet modifies the color of `h1` headings on the Web site, but does nothing to change the default settings for size, weight, or alignment:

```
h1 {color: red;}
```

The combination of the internal and external style sheets results in all `h1` headings being displayed in a large bold red font and left-aligned on the page. However, if a particular heading is formatted with the inline style

```
<h1 style="text-align: center;">Sunny Acres</h1>
```

then that `h1` heading will be centered, displayed in red (defined in the external style sheet), and rendered in a large bold font (defined in the browser's default style sheet). The final appearance is thus a result of a combination of several styles drawn from multiple sources. Many Web browsers now include developer tools to allow page designers to track each style back to its source.

Defining Important Styles

If you need browsers to enforce a style, you can append the `!important` keyword to the style property, using the syntax

```
property: value !important;
```

where *property* is the style property and *value* is the property value. The following style rule sets the color of all `h1` headings to orange; and because this property is marked as important, it takes precedence over any other style that may be defined in the style sheet:

```
h1 {color: orange !important;}
```

TIP

Make sure that the `!important` keyword is placed between the style property value and the closing semicolon; it is invalid if placed elsewhere.

The `!important` keyword is often necessary for visually impaired users who require their pages rendered with large, clear text and highly contrasting colors. Such a user could set the text size in a user-defined style sheet and override any styles specified by the page author through the use of the `!important` keyword. In general, Web page authors should not use the `!important` keyword and should instead write style sheets that are based on the content of the Web documents. This practice makes the style sheets easier to edit and maintain if the content or design of the Web site changes.

Writing Style Comments

Now that you've reviewed some principles of style sheet design and application, you can begin creating your own style sheets. You'll start by creating an external style sheet that will be used to format the appearance of text on the Sunny Acres Web site. Because style sheets are text files, you can create your style sheets with the same text editor you used for creating and editing your HTML files.

To start creating the sa_styles.css style sheet:

1. Use your text editor to open the blank text file **sa_stylestxt.css** from the tutorial.03/tutorial folder.
2. Save the file as **sa_styles.css**.

TIP

Style comments can also be added to embedded style sheets as long as they are placed between the opening and closing `<style>` tags.

Style sheets can be as long and complicated as HTML files. To help others read your style sheet code, you should document the content and purpose of the style sheet using style sheet comments. Style sheet comments are entered as follows

```
/* comment */
```

where *comment* is the text of the comment. Like HTML, CSS ignores the presence of white space, so you can place style comments on several lines to make them easier to read. For example, the following style comment extends over four lines in the style sheet:

```
/*
   Sunny Acres
   Style Sheet
*/
```

Add style comments to the *sa_styles.css* file now to document the purpose and authorship of the style sheet.

To document the style sheet:

1. At the top of the file, insert the following style comments, as shown in Figure 3-8:

```
/*
   Sunny Acres Style Sheet

   Author: your name
   Date:   the date
*/
```

Figure 3-8 Entering style sheet comments

```
/*
   Sunny Acres Style Sheet

   Author: Tammy Nielsen
   Date:   3/1/2014
*/
```

2. Save your changes to the file.

Next, you'll link the Sunny Acres home page to this new style sheet.

To link to the sa_styles.css file:

1. Return to the **home.htm** file in your text editor.
2. Directly below the `link` element for the *sa_layout.css* file, insert the following:

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

3. Save your changes to the file.

Defining Color in CSS

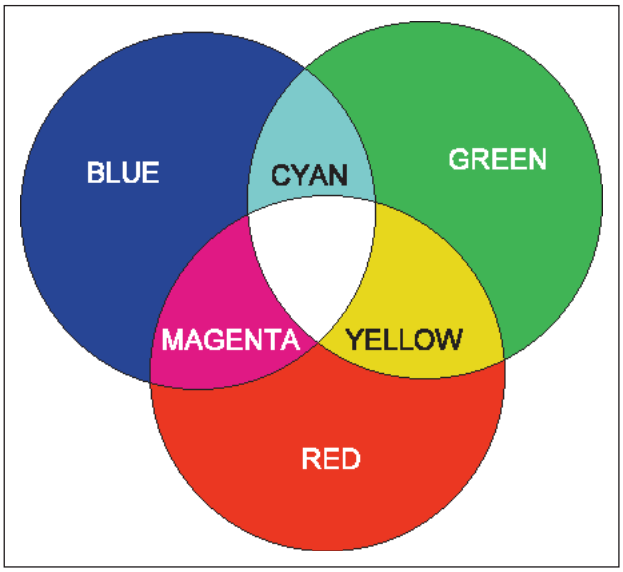
The first part of your style sheet will focus on color. If you've worked with graphics software, you've probably made your color selections using a graphical interface where you can see your color options. Specifying color with CSS is somewhat less intuitive because CSS is a text-based language and requires colors to be defined in textual terms. This is done through either a color value or a color name.

RGB Color Values

A **color value** is a numerical expression that describes the properties of a color. To better understand how numbers can represent colors, it can help to review some of the basic principles of color theory and how they relate to the way colors are rendered in a browser.

In classical color theory, all colors are based on adding three primary colors—red, green, and blue—at different levels of intensity. For example, adding all three primary colors at maximum intensity produces the color white, while adding any two of the three primary colors at maximum intensity produces the trio of complementary colors—yellow, magenta, and cyan (see Figure 3-9).

Figure 3-9 Color addition in the RGB color model



Varying the intensity of the three primary colors creates other colors. Orange, for example, is created from a high intensity of red, a moderate intensity of green, and a total absence of blue. CSS represents these intensities mathematically as a set of numbers called an **RGB triplet**, which has the format

```
rgb(red, green, blue)
```

where *red*, *green*, and *blue* are the intensities of the red, green, and blue components of the color. Intensity values range from 0 (absence of color) to 255 (maximum intensity); thus, the color white has the color value `rgb(255, 255, 255)`, indicating that red, green, and blue are mixed equally at the highest intensity, and orange is represented by `rgb(255, 165, 0)`. RGB triplets can describe 256^3 (16.7 million) possible colors, which is a greater number of colors than the human eye can distinguish. You can also enter each component value as a percentage, with 100% representing the highest intensity. In this form, you would specify the color orange with the following values:

```
rgb(100%, 65%, 0%)
```

The percentage form is less commonly used than RGB values, but many page designers find it easier to work with.

CSS also allows RGB values to be entered as hexadecimal numbers. A **hexadecimal** number is a number expressed in the base 16 numbering system rather than in the commonly used base 10 system. In base 10 counting, numeric values are expressed using combinations of 10 characters (0 through 9); hexadecimal numbering includes six extra characters: A (for 10), B (for 11), C (for 12), D (for 13), E (for 14), and F (for 15). For values above 15, you use a combination of those 16 characters. For example, 16 has a hexadecimal representation of 10, and a value of 255 is represented as FF in hexadecimal numbering. The style value for color represented as a hexadecimal number has the form

```
#redgreenblue
```

where *red*, *green*, and *blue* are the hexadecimal values of the red, green, and blue components. Therefore, the color yellow could be represented either by the RGB triplet

```
rgb(255,255,0)
```

or by the hexadecimal

```
#FFFF00
```

One advantage of using the compact hexadecimal format is that it results in smaller style sheet files; the disadvantage is that hexadecimals are more difficult to read and interpret. Most graphics programs provide color values in either a decimal or a hexadecimal format that you can easily copy into a style sheet. So a common practice is to use graphics software to choose your colors, and then copy the color values from your graphics package.

REFERENCE

Defining Color Values

- To define a color value using the RGB color model, use the property value

  `rgb(red, green, blue)`

  where *red*, *green*, and *blue* are the intensities of red, green, and blue ranging in value from 0 up to 255.
- To define a color value using the HSL color mode, use

  `hsl(hue, saturation, lightness)`

  where *hue* is the tint of the color on the color wheel measured in degrees, *saturation* is the intensity of the color in percent, and *lightness* is the brightness of the color in percent.
- To create a semi-transparent color, use either

  `rgba(red, green, blue, opacity)`

  or

  `hsla(hue, saturation, lightness, opacity)`

  where *opacity* ranges from 0 (transparent) up to 1 (opaque).

Using Color Names

If you don't want to use color values, you can also specify colors by name. CSS supports the 16 basic color names shown in Figure 3-10.

Figure 3-10 **The 16 basic CSS2 color names**

Color Name	RGB Triplet	Hexadecimal	Color Name	RGB Triplet	Hexadecimal
Aqua	(0, 255, 255)	00FFFF	Navy	(0, 0, 128)	000080
Black	(0, 0, 0)	000000	Olive	(128, 128, 0)	808000
Blue	(0, 0, 255)	0000FF	Purple	(128, 0, 128)	FF0000
Fuchsia	(255, 0, 255)	FF00FF	Red	(255, 0, 0)	C0C0C0
Gray	(128, 128, 128)	808080	Silver	(192, 192, 192)	008080
Green	(0, 128, 0)	008000	Teal	(0, 128, 128)	FFFFFF
Lime	(0, 255, 0)	00FF00	White	(255, 255, 255)	FFFF00
Maroon	(128, 0, 0)	800000	Yellow	(255, 255, 0)	

Sixteen colors are not a lot, so most browsers support an extended list of 140 color names, including such colors as orange, crimson, khaki, and brown. Although this extended color list was not part of the CSS specification until CSS3, most browsers support it. You can view these color names in the appendix and in a demo page.

To view the extended list of color names:

1. Use your browser to open the **demo_color_names.htm** file from the tutorial.03\demo folder included with your Data Files.
2. As shown in Figure 3-11, the demo page displays the list of 140 color names along with their color values expressed both as RGB triplets and in hexadecimal form.

Figure 3-11 A partial list of extended color names

Sample	Name	RGB	Hexadecimal
	aliceblue	(240,248,255)	#F0F8FF
	antiquewhite	(250,235,215)	#FAEBD7
	aqua	(0,255,255)	#00FFFF
	aquamarine	(127,255,212)	#7FFFD4
	azure	(240,255,255)	#F0FFFF
	beige	(245,245,220)	#F5F5DC
	bisque	(255,228,196)	#FFE4C4
	black	(0,0,0)	#000000
	blanchedalmond	(255,235,205)	#FFEBCD
	blue	(0,0,255)	#0000FF
	blueviolet	(138,43,226)	#8A2BE2
	brown	(165,42,42)	#A52A2A
	burlywood	(222,184,135)	#DEB887

3. Close the page when you are finished reviewing the extended color names list.

PROSKILLS

Written Communication: Communicating in Color

Humans are born to respond to color. Studies have shown that infants as young as two months prefer colorful objects to non-colored objects, and that memory is often associated with color. While marketing products such as clothes, companies rely on knowing what colors are "in" and what colors are passé. Your color choices can also impact the way your Web pages are received. You want to choose a color scheme that is tailored to the personality and interests of your target audience.

Color also evokes an emotional response, in which certain colors are associated with particular feelings or concepts, such as:

- *red*—assertive, powerful, sexy, dangerous
- *pink*—innocent, romantic, feminine
- *black*—strong, classic, stylish
- *gray*—business-like, detached
- *yellow*—warm, cheerful, optimistic
- *blue*—consoling, serene, quiet
- *orange*—friendly, vigorous, inviting
- *white*—clean, pure, straightforward, innocent

International businesses need to understand how cultural differences can affect people's responses to color. For instance, white, which is associated with innocence in Western cultures, is the color of mourning in China; yellow is considered a bright, cheerful color in the West, while in Buddhist countries it represents spirituality.

When you develop a Web site design, you should test it out before a group of potential customers. In addition to evaluating responses to the content of your Web site, pay attention to reactions to its presentation and appearance, including your color choices.

Defining Text and Background Colors

Now that you've studied how CSS works with colors, you can start applying color to some of the elements of the Sunny Acres Web site. CSS supports styles to define both the text and background color for each element on your page. You've already seen examples of how to set the text color of a page element using the `color` property, which has the form

```
color: color;
```

where *color* is either a color value or a color name. Background colors are defined using the property

```
background-color: color;
```

where once again *color* is either a color name or value. Tammy wants the body of each page on her Web site to have a white background. Although most browsers by default will apply a white background, it's a good idea to make this explicit. Also, she wants the text of the `h2` headings to be displayed in white on a green background. The style rules to apply these two design choices are:

```
body {
   background-color: white;
}

h2 {
   background-color: rgb(0, 154, 0);
   color: white;
}
```

Whether to use a color value in place of a color name is often a matter of personal preference. This code does both, with the RGB triplet (0, 154, 0) representing the color green, and the color name *white* used for the text color of the `h2` headings and the background color of the page body. You'll add these style rules to the *sa_styles.css* style sheet.

REFERENCE

Setting Foreground and Background Color

- To set the background color of an element, use the property

  ```
  background-color: color;
  ```

  where *color* is a color name or a color value.
- To set the foreground or text color of an element, use the following property:

  ```
  color: color;
  ```

To format the text and background colors:

1. Return to the **sa_styles.css** file in your text editor.
2. Directly below the style comments, insert the following style rules, as shown in Figure 3-12:

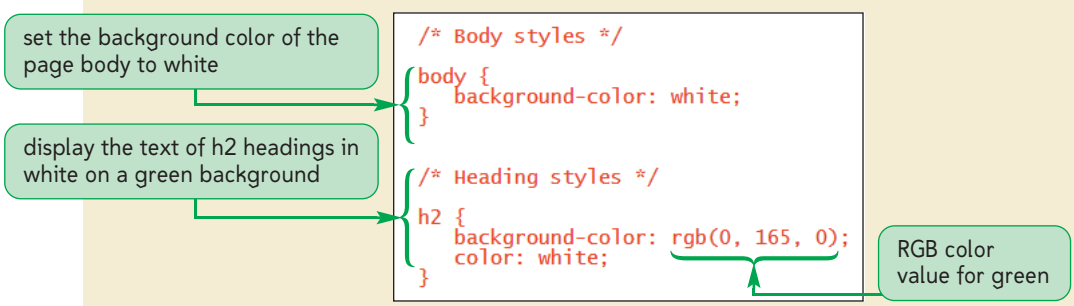
```
/* Body styles */

body {
   background-color: white;
}


/* Heading styles */

h2 {
   background-color: rgb(0, 165, 0);
   color: white;
}
```

Make sure you end every style property value with a semicolon to separate it from other style properties.

TIP

About 8% of all men and 0.5% of all women have some form of color blindness. Because red-green color blindness is the most common form of color impairment, you should avoid using red text on a green background or vice versa.

Figure 3-12 Setting the foreground and background colors



3. Save your changes to the file and then reload the **home.htm** file in your Web browser. As shown in Figure 3-13, the h2 heading text appears in white on a green background.

Figure 3-13 Formatted h2 headings

INSIGHT

Deprecated Approaches to Color

Because CSS was not part of the original HTML specifications, older HTML code used HTML attributes to define page colors. If you work with older Web pages, you may encounter some of these deprecated attributes. For example, the `bgcolor` attribute in the `<body>` tag was used to define the background color for an entire page. To define the text color for the entire page, the `text` attribute was used. Both attributes required the page author to enter either a hexadecimal color value or a recognized color name. Thus, the following code set the page background to yellow and the page text color to sky blue with the hexadecimal color value 99CCFF:

```
<body bgcolor="yellow" text="#99CCFF">
```

To color a section of text, page authors enclosed the text within a two-sided `<font>` tag, which supported several design attributes. One of these, `color`, defined the font color of the enclosed text. For example, the following deprecated code sets the text color of an `h1` heading to green:

```
<h1><font color="green">Sunny Acres</font></h1>
```

These attributes, as well as the `<font>` tag, have been deprecated due to the desire to completely separate page content from page design. Although you may still encounter them and browsers still support them, you should always use style sheets to set your page design.

Enhancements to Color in CSS3

RGB color values and color names have been part of Web page design since the introduction of CSS. However, graphic designers have long wanted additional options for creating and working with colors in CSS. For this reason, CSS3 introduced additional tools to allow Web page designers to create more interesting and flexible designs based on color.

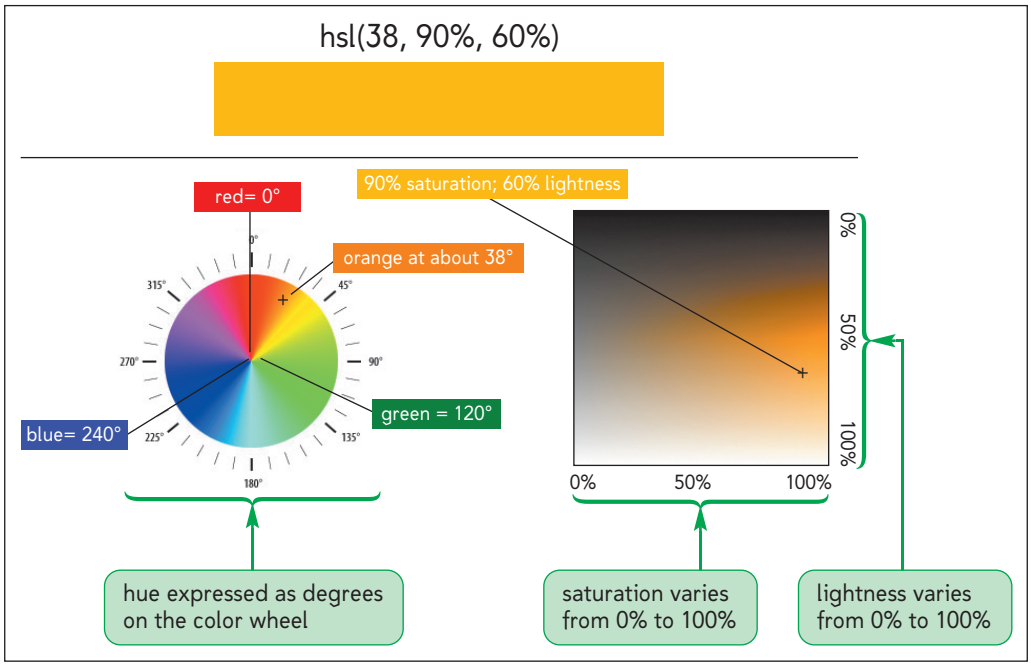
HSL Color Values

The RGB color model is only one way of describing colors. CSS3 also supports the Hue Saturation Lightness (HSL) model that describes colors based on hue, saturation, and lightness. **Hue** is the tint of the color and is based on the color's location on the color wheel. Hue values range from 0° up to 360°, where 0° matches the location of red on the color wheel, 120° matches green, and 240° matches blue. **Saturation** measures the intensity of the chosen color and ranges from 0% (no color) up to 100% (full color). Finally, **lightness** measures the brightness of the color and ranges from 0% (black) up to 100% (white). Color values using the HSL model are described in CSS3 using

```
hsl(hue, saturation, lightness)
```

where *hue* is the tint of the color in degrees, *saturation* is the intensity of the color in percent, and *lightness* is the brightness of the color in percent. Figure 3-14 shows how setting the hue to 38°, the saturation to 90%, and the lightness to 60% results in a medium shade of orange.

Figure 3-14 HSL color saturation model



Graphic designers consider HSL easier to use because it allows you to guess at an initial color based on hue and then tweak the saturation and lightness values to fine-tune the final color. This is more difficult in the RGB model because you have to balance three completely different colors to achieve the right mix. For example, the RGB equivalent to the color in Figure 3-14 would be the color value rgb(245, 177, 61); however, it's not immediately apparent why that mixture of red, green, and blue would result in that particular shade of orange.

Opacity Values in CSS3

CSS3 also allows page designers to augment RGB and HSL color values by specifying a color's opacity. Opacity defines how much of the colors below the surface of the current object show through to affect its appearance. The opacity of a color can be specified using either of the following `rgba` and `hsla` color values

```
rgba(red, green, blue, opacity)
hsla(hue, saturation, lightness, opacity)
```

where *opacity* sets the transparency of the color as a decimal ranging from 0 (completely transparent) up to 1.0 (completely opaque). For example, the following style displays the text of `h1` headings in a medium shade of orange at 70% opacity:

```
hsla(38, 90%, 60%, 0.7)
```

TIP

The *a* in rgba and hsla stands for *alpha* and refers to the alpha channel, a color concept developed in the 1970s to add transparency to the color model.

With semi-transparent colors, the final color rendered by a browser depends on the background color of the parent element. Displayed against a white background, this medium orange color would appear in a lighter shade of orange, while displayed against a black background it would appear as very dark orange. The advantage of using semi-transparent colors is that it makes it easier to create a color theme in which similarly tinted colors are used throughout the page.

Styles Using Progressive Enhancement

Tammy suggests that you modify the style of the `h2` headings to make the text appear as a semi-transparent white against a green background. To create this effect, you can employ the following style rule:

```
h2 {
   background-color: rgb(0, 154, 0);
   color: white;
   color: rgba(255, 255, 255, 0.8);
}
```

TIP

Apply progressive enhancement whenever you use newer code that would cause your page to be unreadable under older browsers.

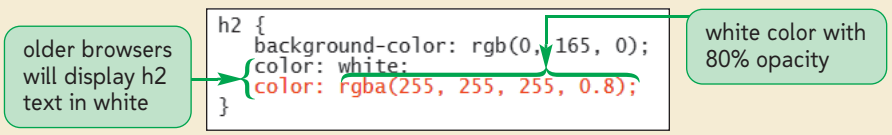
Notice that this code doesn't remove the initial `color` property that set the text color to white, but simply adds another `color` property to the style rule. This is an example of a technique known as **progressive enhancement**, which places code conforming to older standards before newer properties. Older browsers that do not support CSS3 will ignore the RGBA color value and display the text in white, while newer browsers that do support CSS3 will apply the RGBA color value because that color value, being declared last, has precedence. Thus, both older and newer browsers are served by this style rule.

To make the heading text semi-transparent:

1. Return to the **sa_styles.css** file in your text editor.
2. Within the style rule for the `h2` selector, insert the following `color` property, as shown in Figure 3-15:

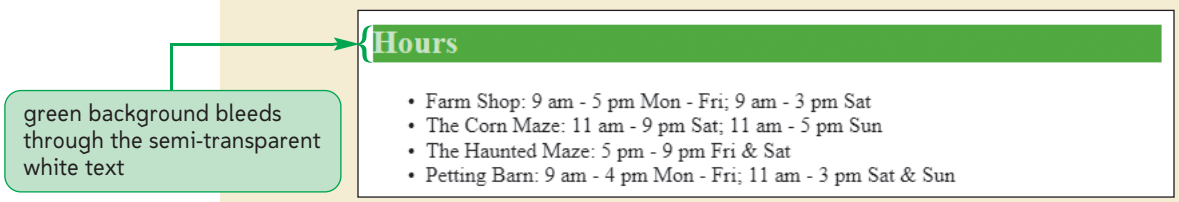
```
color: rgba(255, 255, 255, 0.8);
```

Figure 3-15 Setting a semi-transparent color



3. Save your changes and then reload the **home.htm** file in your Web browser. If you're using a current browser that supports CSS3, the `h2` heading text should appear as light green (see Figure 3-16). Under older browsers, the text should retain the white color shown earlier in Figure 3-13.

Figure 3-16 Heading text in semi-transparent white



PROSKILLS

Problem Solving: Choosing a Color Scheme

One of the worst things you can do to your Web site is to associate interesting and useful content with jarring and disagreeable color. Many designers prefer the HSL color system because it makes it easier to select visually pleasing color schemes. The following are some basic color schemes you may want to apply to your own Web sites:

- *monochrome*—a single hue with varying values for saturation and lightness; this color scheme is easy to manage but is not as vibrant as other designs
- *complementary*—two hues separated by 180° on the color wheel; this color scheme is the most vibrant and offers the highest contrast and visual interest, but can be misused and might distract users from the page content
- *triad*—three hues separated by 120° on the color wheel; this color scheme provides the same opportunity for pleasing color contrasts as a complementary design, but might not be visibly striking
- *tetrad*—four hues separated by 90° on the color wheel; perhaps the richest of all color schemes, it is also the hardest one in which to achieve color balance
- *analogic*—two hues close to one another on the color wheel in which one color is the dominant color and the other is a supporting color used only for highlights and nuance; this scheme lacks color contrasts and is not as vibrant as other color schemes

Once you have selected a color design and the main hues, you then vary those colors by altering the saturation and lightness. One of the great advantages of style sheets is that you can quickly modify your color design choices and view the impact of those changes on your Web page content.

You show Tammy the work you've done on colors. She's pleased with the ease of using CSS to modify the design and appearance of elements on the Sunny Acres Web site. In the next session, you'll continue to explore CSS styles, focusing on text and image styles.

REVIEW

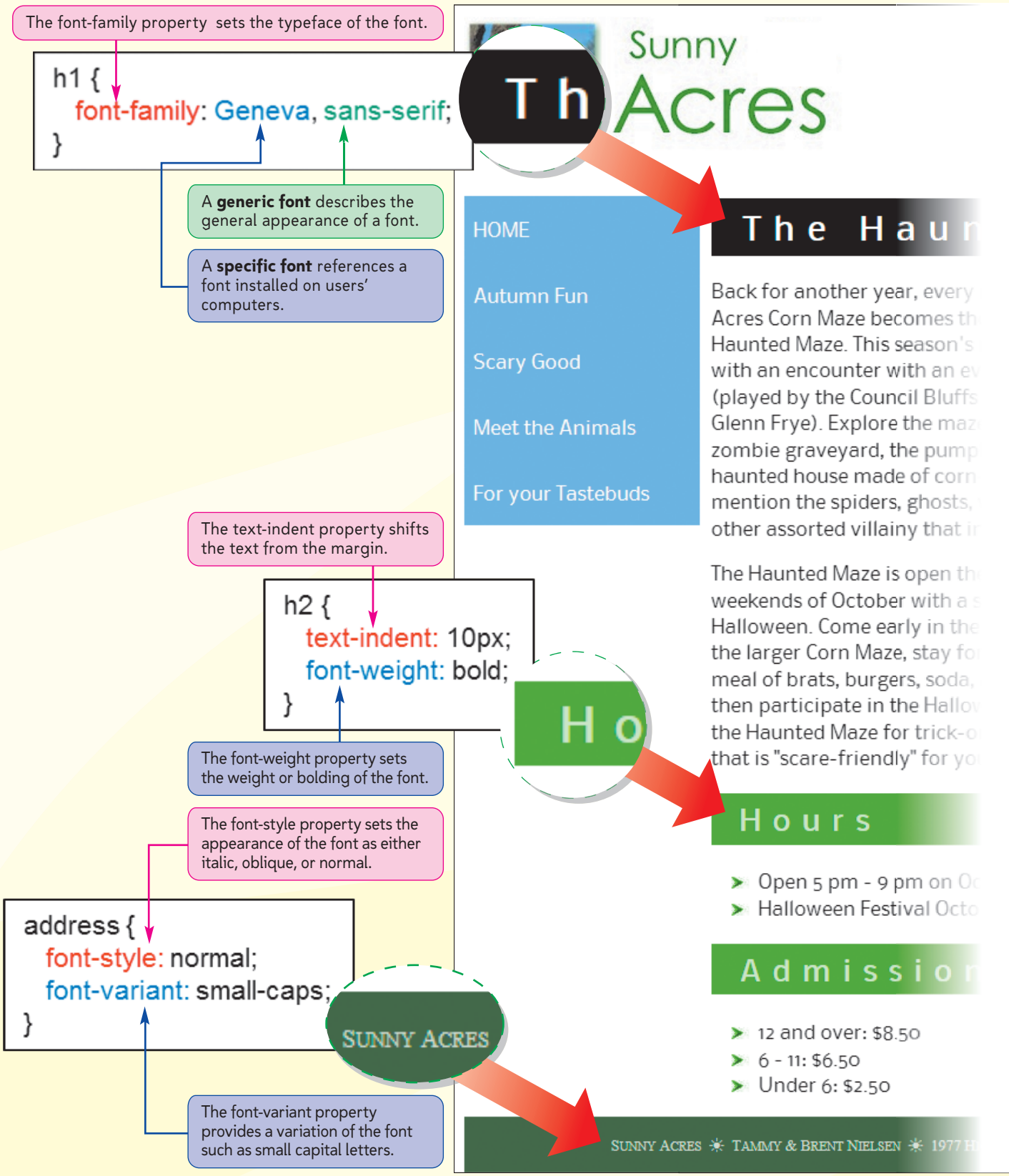
Session 3.1 Quick Check

1. What are inline styles, embedded styles, and external style sheets? Which would you use to define a design for an entire Web site?
2. What keyword do you add to a style property to override style precedence and style inheritance?
3. Specify the code to enter the comment *Sunny Acres Color Styles* in a style sheet.
4. Provide the style rule to display blockquote text in red using an RGB triplet.
5. The color chartreuse is located at 90° on the color wheel with 100% saturation and 50% lightness. Provide a style rule to display address text with chartreuse as the background color.
6. What is progressive enhancement?
7. Based on the following style rule for paragraph text, which style property will be used by an older browser that supports only CSS2?

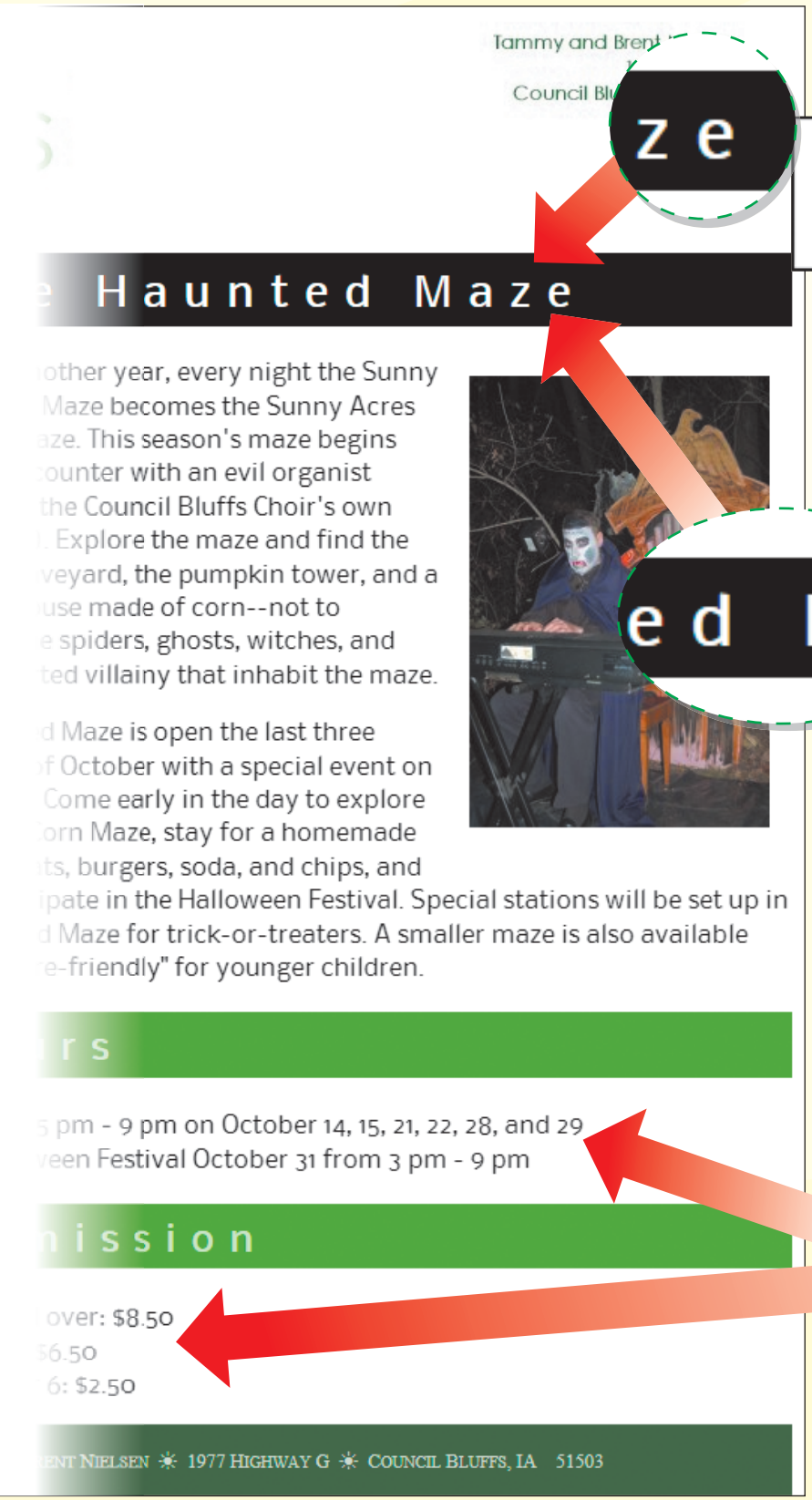
```
p {
   color: rgb(232, 121, 50);
   color: hsla(23, 80%, 55%, 0.75);
}
```

8. Provide a style rule to display `h1` and `h2` headings with a background color of yellow (an equal mixture of red and green at highest intensity with no blue) at 70% opacity.

SESSION 3.2 VISUAL OVERVIEW



SELECTORS AND TEXT STYLES



The font-size property sets the size of the font.

```
h1 {
   font-size: 22px;
}
```

The abbreviation px stands for **pixel**, which is a single dot on the screen.

The letter-spacing property sets the **kerning**, which is the space between letters.

```
h1 {
   letter-spacing: 10px;
   word-spacing: 12px;
   line-height: 18px;
}
```

The line-height property sets the **leading**, which is the height of the line.

The word-spacing property sets the **tracking**, which is the space between words.

Contextual selectors match elements based on their positions in the document hierarchy.

```
section ul li {
   styles;
}
```

Exploring Selector Patterns

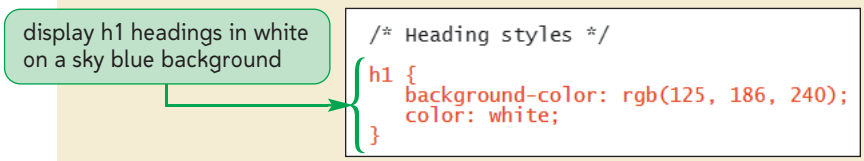
Tammy has examined your work on color styles from the last session and asks that you create another color style for `h1` headings. She suggests that you display the text of your `h1` headings in white on a sky blue background.

To format `h1` headings:

1. Return to the **sa_styles.css** file in your text editor.
2. Directly above the style rule for `h2` headings, insert the following style rule, as shown in Figure 3-17:

```
h1 {
   background-color: rgb(125, 186, 240);
   color: white;
}
```

Figure 3-17 Creating a style for h1 headings



3. Save your changes to the file and then reload the **home.htm** file in your Web browser. Figure 3-18 shows the revised appearance of the page.

Figure 3-18 Effect of the h1 style rule

Tammy notices that the sky blue background has been added to the Welcome text in the `h1` heading, but it also has been added to the company logo. A quick investigation of the HTML code reveals that the logo itself is also within an `h1` heading:

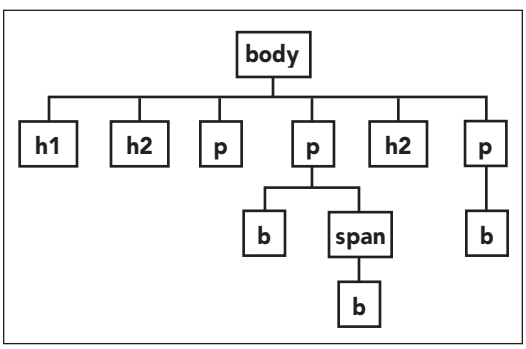
```
<header>
   <h1>
      <img src="salogo.png" alt="Sunny Acres" />
   </h1>
</header>
```

Tammy doesn't want all `h1` headings formatted the same way. To specify which `h1` headings receive a sky blue background and which ones don't, you have to modify the selector in the style rule you just entered.

Contextual Selectors

So far, the only selectors you've studied involve either single elements or groups of elements in a comma-separated list. However, this approach doesn't take into account that Web pages are structured documents in which elements are nested within other elements, forming a hierarchy of elements. Figure 3-19 shows an example of such a tree structure for a Web page consisting of a few headings, a couple of paragraphs, and a few text-level elements, all descending from the `body` element.

Figure 3-19 A sample hierarchy of page elements



To create styles that take advantage of this tree structure, CSS allows you to create contextual selectors whose values represent the locations of elements within the hierarchy. As with the folder structure discussed in Tutorial 2, elements in a Web page are often referenced using their familial relationships. A **parent element** is an element that contains one or more other elements, which are **child elements** of the parent. Two child elements that share the same parent are referred to as **sibling elements**. Each child element may contain children of its own and so forth down the hierarchy, creating a set of **descendant elements** that are all descended from a common parent. The ultimate parent element for the HTML file is the `html` element itself, and the parent element for all elements within the page body is the `body` element.

One commonly used selector that takes advantage of these familial relationships has the form

```
parent descendant {styles}
```

where *`parent`* is the parent element, *`descendant`* is a descendant of the parent, and *`styles`* are the style properties applied to the descendant element. For example, to display the text of all `h1` headings found within a page header in blue, you could apply the following style rule:

```
header h1 {color: blue;}
```

In this case, `header` is the parent element and `h1` is the descendant element (because it is contained within the `header` element). Any `h1` heading that is not placed within a page header is not affected by this style. Note that the descendant element does not have to be a direct child of the parent; it can appear several levels below the parent in the hierarchy. This style applies equally to the following HTML code:

```
<header>
   <hgroup>
      <h1>Sunny Acres</h1>
   </hgroup>
</header>
```

Here, the `h1` element is a direct child only of the `hgroup` element; but because it is still a descendant of the `header` element, it would still appear in blue.

Contextual selectors take advantage of the general rule that the more specific style is applied in preference to the more general. For instance, the styles

```
section h1 {color: red;}
h1         {color: blue;}
```

would result in any `h1` heading text nested within a `section` element appearing in red, even though the last style sets the text color to blue. The more specific style using the contextual selector takes precedence over the general style in which no context has been given.

Contextual selectors also can be listed with other selectors. The following style rule is applied both to `strong` elements nested within list items and to `h2` headings:

```
li strong, h2 {color: blue;}
```

The parent/descendant form is only one example of a contextual selector. Figure 3-20 describes some of the other contextual forms supported by CSS.

Figure 3-20 Contextual selectors

Selector	Description
*	Matches any element in the hierarchy
e	Matches any element, *e*, in the hierarchy
e1, *e2*, *e3*, ...	Matches the group of elements *e1*, *e2*, *e3*, ...
e f	Matches any element, *f*, that is a descendant of an element, *e*
e>*f*	Matches any element, *f*, that is a direct child of an element, *e*
e+*f*	Matches any element, *f*, that is immediately preceded by a sibling element, *e*
e~*f*	Matches any element, *f*, that is a sibling to an element, *e*

For example, the style rule

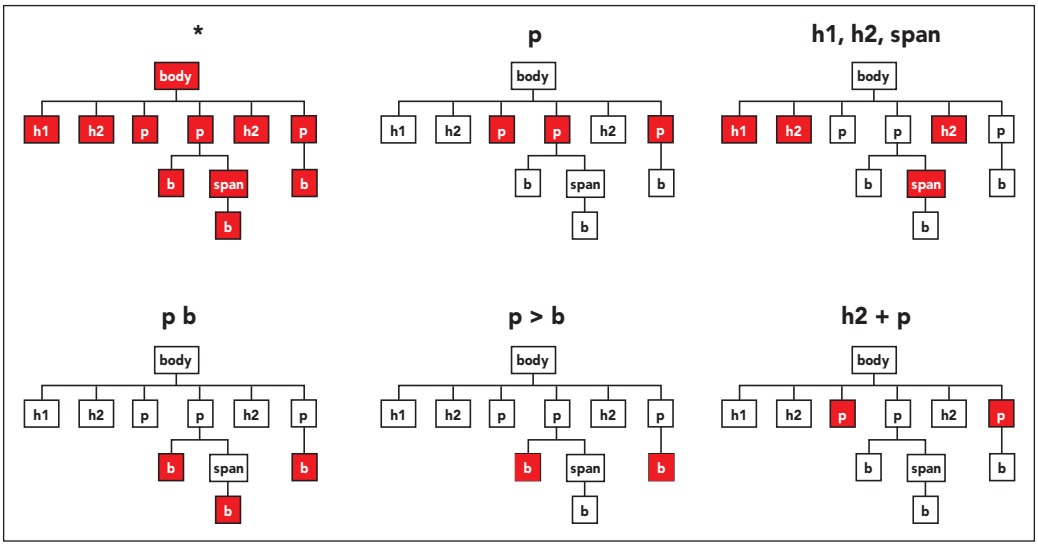
```
* {color: blue;}
```

uses the asterisk (*) selector—also known as the **wildcard selector**—to select all elements in the document. The result is that the text of all elements in the document appears in blue. On the other hand, the rule

```
p > em {color: blue;}
```

applies the blue text color only to emphasized text placed as a direct child of a paragraph. Figure 3-21 provides additional examples of selectors applied to a document tree. 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. Thus, a style applied to a paragraph element is automatically passed down to elements contained within that paragraph unless that style conflicts with a more specific style.

Figure 3-21 Examples of contextual selectors



REFERENCE

Using Contextual Selectors

- To apply a style to all elements in a 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.
- To apply a selector to any sibling element, use the *e ~ f* selector, where *e* and *f* are siblings.

Now that you've seen how to create contextual selectors, you can fix the style rule you created earlier. Rather than applying white text and a sky blue background to every `h1` heading, you'll apply those properties only to `h1` headings that are descendants of the `section` element. The style rule thus becomes

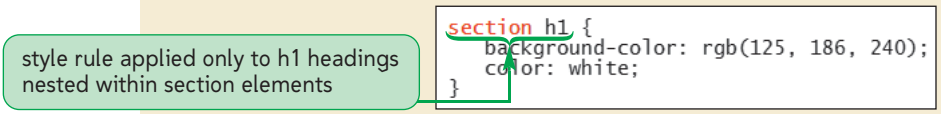
```
section h1 {
   background-color: rgb(125, 186, 240);
   color: white;
}
```

with the `section h1` selector replacing simply `h1`. You'll revise the style sheet accordingly.

To revise the style sheet:

1. Return to the **sa_styles.css** file in your text editor.
2. Change the selector for the `h1` heading rule to **`section h1`** (see Figure 3-22).

Figure 3-22 Applying a contextual selector



3. Save your changes to the style sheet and then reload the **home.htm** file in your Web browser. Verify that the sky blue background is applied only to the Welcome heading.

Attribute Selectors

Selectors also can be defined based on attributes and attribute values associated with elements. Two attributes, `id` and `class`, are often key in targeting styles to a specific element or group of elements. Recall that the `id` attribute is used to identify specific elements within the Web document. To apply a style to an element based on its id, you use the selector

```
#id
```

where *id* is the value of the `id` attribute. Thus, to format the text of the `h1` heading

```
<h1 id="main">Sunny Acres</h1>
```

to appear in red, you could apply the following style rule:

```
#main {color: red;}
```

TIP

A style rule involving an `id` selector has precedence over any rule except those defined within an inline style.

Because no two elements can share the same id value, HTML uses the `class` attribute to identify groups of elements that share a similar characteristic or property. The attribute has the syntax

```
<elem class="className"> … </elem>
```

where *className* is the name of the element class. For example, the following `h1` heading and paragraph both belong to the *intro* class of elements.

```
<h1 class="intro">Sunny Acres</h1>
<p class="intro">
   Welcome to Sunny Acres, where there's always
   something happening on the farm.
</p>
```

TIP

You can associate a single element with several classes by listing each class name, separated by a space, as part of the `class` attribute value.

One reason to use the `class` attribute is to assign the same style to multiple elements that belong to the same class. A selector based on a class has the form

```
.class {styles}
```

where *class* is the name of the class and *styles* are styles associated with that class of element. Thus, to display the text of all elements belonging to the *intro* class in blue, you could apply the following style:

```
.intro {color: blue;}
```

Because different types of elements can belong to the same class, you can also specify exactly which kinds of elements within that class receive the style rule by using the selector

```
elem.class {styles}
```

where *elem* is the name of the element. Thus, the style rule

```
h1.intro {color: blue;}
```

causes the text of all h1 headings that belong to the intro class to appear in blue.

While id and class are the most common attributes to use with selectors, any attribute or attribute value can be the basis for a selector. Figure 3-23 lists the selector patterns that are based on attributes and their values.

Figure 3-23 Attribute selectors

Selector	Description	Example	Matches
#id	The element with the id value, *id*	#intro	The element with the id *intro*
.class	All elements with the class value, *class*	.main	All elements belonging to the *main* class
elem.class	All *elem* elements with the class value *class*	p.main	All paragraphs belonging to the *main* class
elem[att]	All *elem* elements containing the *att* attribute	a[href]	All hypertext elements containing the href attribute
elem[att="*text*"]	All *elem* elements whose *att* attribute equals *text*	a[href="gloss.htm"]	All hypertext elements whose href attribute equals *gloss.htm*
elem[att~="*text*"]	All *elem* elements whose *att* attribute contains the word *text*	a[rel~="glossary"]	All hypertext elements whose rel attribute contains the word *glossary*
elem[att|="*text*"]	All *elem* elements whose *att* attribute value is a hyphen-separated list of words beginning with *text*	p[id|="first"]	All paragraphs whose id attribute starts with the word *first* in a hyphen-separated list of words
elem[att^="*text*"]	All *elem* elements whose *att* attribute begins with *text* (CSS3)	a[rel^="prev"]	All hypertext elements whose rel attribute begins with *prev*
elem[att$="*text*"]	All *elem* elements whose *att* attribute ends with *text* (CSS3)	a[href$="org"]	All hypertext elements whose href attribute ends with *org*
elem[att*="*text*"]	All *elem* elements whose *att* attribute contains the value *text* (CSS3)	a[href*="faq"]	All hypertext elements whose href attribute contains the text string *faq*

All current browsers support the attribute selector patterns listed in Figure 3-23; be aware, however, that some older browsers—primarily IE6—may not support these selectors in your style sheets.

REFERENCE

Using Attribute Selectors

- To apply a style based on the id value of an element, use the *#id* selector, where *id* is the value of the id attribute.
- To apply a style based on the class value of elements, use either the *.class* or the *elem.class* selectors, where *class* is the value of the class attribute and *elem* is the element name.
- To apply a style based on whether an element contains an attribute, use the *elem[att]* selector, where *class* is the class name and *att* is the attribute name.
- To apply a style based on whether the attribute value for elements equals a specified value, use the *elem[att="val"]* where *val* is the specified value.

In the Sunny Acres home page, Tammy has added the following class attribute to the last paragraph that introduces the Web site:

```
<p class="closing">We all hope to see you soon,
   down on the farm.<br />
   &mdash; <span>Tammy &amp; Brent Nielsen</span>
</p>
```

She suggests that you format this paragraph so that it appears in green and is right-aligned on the page. To do this, you'll add the following style to the *sa_styles.css* style sheet:

```
section p.closing {
   color: rgb(0, 165, 0);
   text-align: right;
}
```

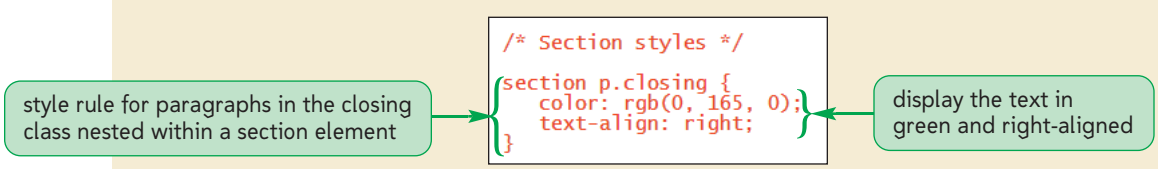
Note that this style rule applies to paragraphs belonging to the closing class and nested within the section element. You could have simply used the .closing selector, but then the style rule would apply to any element belonging to the *closing* class. Usually, you want to be as specific as possible in your style sheet so that you are always targeting exactly the elements that you want to target.

To create a style based on the class attribute:

1. Return to the **sa_styles.css** file in your text editor.
2. Add the following style rule at the bottom of style sheet (see Figure 3-24):

```
/* Section styles */

section p.closing {
   color: rgb(0, 165, 0);
   text-align: right;
}
```

Figure 3-24 Applying a selector based on class



3. Save your changes to the style sheet and then reload the **home.htm** file in your Web browser. Verify that the text of the last paragraph appears in green and is right-aligned on the page (see Figure 3-25).

Figure 3-25 Closing paragraph of the home page

Styling Web Page Text

The `text-align` property you used in your style rule to right-align the contents of the closing paragraph is an example of a text style. In this section, you'll explore other CSS styles used to format the appearance of Web page text.

Choosing the Text Font

Tammy has noticed that all of the text in her sample pages is displayed in the same typeface, or **font**. She'd like to see more variety in how the Web page text is rendered. The default font used by most browsers is Times New Roman, but you can specify a different font for any page element using the property

```
font-family: fonts;
```

where *fonts* is a comma-separated list of specific or generic font names. A specific font is a font that is identified by name, such as Times New Roman or Helvetica. When referenced by the `font-family` property, a specific font refers to a font definition that is stored on a user's computer. A generic font describes the general appearance of a typeface, but does not rely on a specific font definition. CSS supports the following generic font groups:

- *serif*—a typeface in which a small ornamentation appears at the tail end of each character
- *sans-serif*—a non-serif font without any ornamentation
- *monospace*—a typeface in which each character has the same width; often used to display programming code
- *cursive*—a typeface that mimics handwriting with highly stylized elements and flourishes; best used in small doses for decorative page elements
- *fantasy*—a highly ornamental typeface used for page decoration; should never be used with body text

When you use generic fonts, you have no control over which font a user's browser will choose for your Web page. Therefore, the common practice is to list specific fonts first, in order of preference, and end the list with a generic font. If a user's browser cannot find any of the specific fonts listed, it uses a generic font of its own choosing. For example, the style

```
font-family: 'Arial Black', Gadget, sans-serif;
```

tells a browser to use the Arial Black font if available; if not, to look for the Gadget font; and if neither are available, to use a generic sans-serif font of its own selection. Note that font names containing one or more blank spaces (such as Arial Black) must be enclosed within single or double quotes.

Because the available fonts vary with each user's operating system, the challenge is to choose a list of fonts known as **Web safe fonts**, which will be displayed in mostly the same way in all browsers and on all devices. Figure 3-26 shows several commonly used fonts.

Figure 3-26 Web safe fonts

A general rule for printing is to use sans-serif fonts for headlines and serif fonts for body text. For computer monitors, which have lower resolutions than printed material, the general rule is to use sans-serif fonts for headlines and body text, leaving serif fonts for special effects and large text.

REFERENCE

Setting Font Face and Sizes

- To define a font face, use the style property

```
font-family: fonts;
```

  where *fonts* is a comma-separated list of fonts that the browser can use with the element. List specific fonts first and complete the list with a generic font.
- To set a font size, use the style property

```
font-size: size;
```

  where *size* is a CSS unit of length in either relative or absolute units.
- To set kerning (the space between letters), use the following style property:

```
letter-spacing: size;
```

- To set tracking (the space between words), use the following style property:

```
word-spacing: size;
```

Tammy expects that her Web page will be viewed only on computer monitors, so you'll use a sans-serif font for all of the body text by adding the style

```
font-family: Verdana, Geneva, sans-serif;
```

to the body selector. Browsers will first try to load the Verdana font, followed by the Geneva font. If both are unavailable, browsers will load a generic sans-serif font.

To apply a sans-serif font to the body text:

1. Return to the **sa_styles.css** file in your text editor.
2. Add the following style to the body style rule at the top of the style sheet, as shown in Figure 3-27:

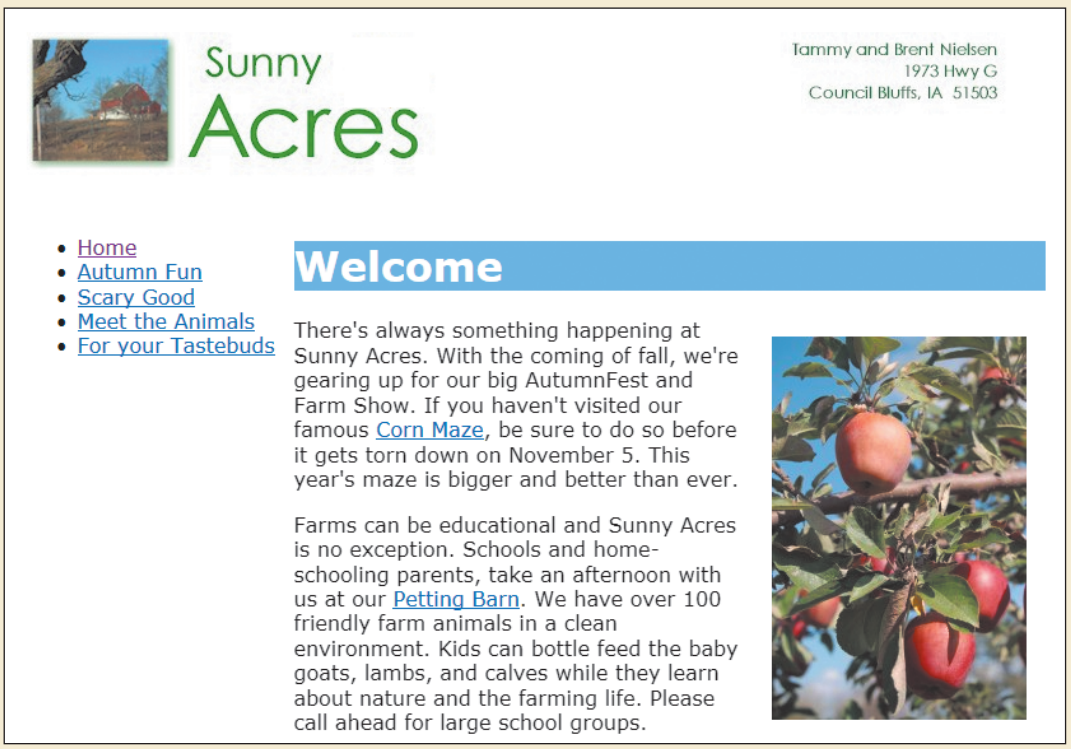
```
font-family: Verdana, Geneva, sans-serif;
```

Figure 3-27 Specify the default font for the Web page body

```
body {
   background-color: white;
   font-family: Verdana, Geneva, sans-serif;
}
```

3. Save your changes to the style sheet and then reload the **home.htm** file in your Web browser. As shown in Figure 3-28, the text of the entire page is displayed in a sans-serif font.

Figure 3-28 Displaying the page text in a sans-serif font



Setting the Font Size

TIP

Including too many fonts can make your page difficult to read. Don't use more than two or three typefaces within a single page.

Tammy would like the Welcome heading on her home page to be displayed in slightly smaller text than is generally set by a browser's internal style sheet. The style to change the font size is

```
font-size: size;
```

where *size* is a length measurement. Lengths can be specified in four different ways:

- with a unit of measurement
- as a percentage of the size of the containing element
- with a keyword description
- with a keyword expressing the size relative to the size of the containing element

TIP

Use absolute units only when you can predict or fix the size and dimensions of the output device.

If you choose to specify lengths using measurement units, you can use absolute units or relative units. **Absolute units** are units that are fixed in size regardless of the device rendering the Web page. They are specified in one of five standard units of measurement: mm (millimeters), cm (centimeters), in (inches), pt (points), and pc (picas). Points and picas might not be as familiar to you as inches, millimeters, and centimeters. For comparison, there are 72 points in an inch, 12 points in a pica, and 6 picas in an inch. Size values for any of these measurements can be whole numbers (0, 1, 2 ...) or decimals (0.5, 1.6, 3.9 ...). For example, if you want your text to be 1/2 inch in size, you can use any of the following styles:

```
font-size: 0.5in
font-size: 36pt
font-size: 3pc
```

Note that you should not insert a space between the size value and the unit abbreviation.

Absolute measurements are appropriate when you know the physical properties of an output device and want to fix a size to a specific value. Of course, this is not often the case with Web pages that can be displayed on a variety of devices and under several possible screen or page resolutions. To cope with the uncertainty about how their pages will be viewed, many Web page designers opt to use **relative units**, which are expressed relative to the size of other objects within the Web page. One commonly used relative unit is the **em unit**. The exact meaning of the em unit depends on its use in the style sheet. If the em unit is used for setting font size, it expresses the size relative to the font size of the parent element. For an `h1` heading, the parent element is the Web page body. Thus, the style rule

```
h1 {font-size: 2em;}
```

sets the font size of `h1` headings to twice the font size of body text. If body text is displayed in a 12-point font, this style will cause `h1` headings to be displayed in a 24-point font. On the other hand, if the `h1` heading is nested within another element, such as a `section` element, the size of the `h1` heading will be twice the size of text in that parent element. Context is important when interpreting the effect of the em unit.

One of the great advantages of relative units like the em unit is that they can make your page **scalable**, allowing the page to be rendered the same way no matter what font size is used by the browser. Setting the font size of `h1` headings to 1.5em ensures the heading will be 50% larger than the body text for all users.

Another way to create relative font sizes is to express the font size as a percentage. Like the em unit, percentages are based on the font size of the parent element. The style

```
h1 {font-size: 200%;}
```

sets the font size of `h1` headings to 200%, or twice the font size of body text.

Another unit of measurement widely used on the Web is the **pixel**, which represents a single dot on the output device. The size or **resolution** of most output devices is typically expressed in terms of pixels. Thus a 1280 × 720 screen resolution on a computer monitor is 1280 pixels wide by 720 pixels tall, for a total of 921,600 pixels or 0.92 megapixels. A pixel is a relative unit because the actual rendered size depends on the **density** of the output device. A Windows PC, for example, has a density of 96 dpi (dots per inch), while a Macintosh computer has a density of 72 dpi. Some mobile phones have densities as high as 200 or 300 dpi. The pixel measure is the most precise unit of measure and gives designers the most control over the appearance of a page; however, pixels are not scalable. This can pose a problem for visually impaired users who need larger fonts, or for users of mobile devices with very dense screens.

Finally, you also can express font sizes using one of the following keywords: `xx-small`, `x-small`, `small`, `medium`, `large`, `x-large`, `xx-large`, `larger`, or `smaller`. The size corresponding to each of these keywords is determined by the browser. Note that the `larger` and `smaller` keywords are relative sizes, making the font size of the element one size larger or smaller than the surrounding text. For example, the following set of styles causes the `body` text to be displayed in a small font, while `h2` text is displayed in a font one size larger (medium):

```
body {font-size: small;}
h2   {font-size: larger;}
```

Tammy suggests that you set the size of the `h1` headings to 1.7em, making the headings 70% larger than the default size of the body text in the document.

To set the font size of the `h1` headings:

1. Return to the **sa_styles.css** file in your text editor.
2. Add the following style to the style rule for `h1` headings in the `section` element (see Figure 3-29):

```
font-size: 1.7em;
```

Figure 3-29 Setting the font size of h1 headings

```
section h1 {
   background-color: rgb(125, 186, 240);
   color: white;
   font-size: 1.7em;
}
```

3. Save your changes to the file and then reload the **home.htm** file in your Web browser. Verify that the font size of the `h1` heading appears slightly smaller under the revised style sheet.

PROSKILLS

Decision Making: Selecting a Text Font

The challenge with designing Web text is that you don't have the same control over the output device as you do when choosing a font style for printed output. A user may not have that beautiful font you selected, and may have installed a font that will render your page unreadable. If you absolutely must have a section of text rendered in a specific font at a specific size, then your best choice may be to use an inline image in place of text.

Of course, you can't make your entire page an inline image, so you *always* should provide options for your customers in the form of extensive font lists. Other important things to consider when designing your text include the following:

- *Keep it plain*—Avoid large blocks of italicized text and boldfaced text. Those styles are designed for emphasis, not readability.
- *Sans-serif vs. serif*—Because they are more easily read on a computer monitor, use sans-serif fonts for your body text. Reserve the use of serif, cursive, and fantasy fonts for page headings and special decorative elements.
- *Relative vs. absolute*—Font sizes can be expressed in relative or absolute units. A relative unit like the em unit is more flexible and will be sized to match the screen resolution of the user's device; but you have more control over your page's appearance with an absolute unit. Generally, you want to use an absolute unit only when you know the configuration of the device the reader is using to view your page.
- *Size matters*—Almost all fonts are readable at a size of 14 pixels or greater; however, for smaller sizes you should choose fonts that were designed for screen display, such as Verdana and Georgia. On the other hand, Times and Arial often do not render well at smaller sizes. If you have to go really small (at a size of only a few pixels), you should either use a Web font that is specially designed for that purpose or replace the text with an inline image.
- *Avoid long lines*—With more users accessing the Web with widescreen monitors, you run the risk of presenting users with long lines of text. In general, try to keep the length of your lines to 60 characters or less. Anything longer is difficult to read.

When choosing any typeface and font style, the key is to test your selection on a variety of browsers, devices, screen resolutions, and densities. Don't assume that text that is readable and pleasing to the eye on your computer screen will work as well on another screen.

Controlling Spacing and Indentation

Tammy thinks that the text for the Welcome heading looks too crowded. She's wondering if you can further spread it out across the width of the page. She also would like to see more space between the first letter, W, and the left edge of the sky blue background.

CSS supports styles that allow you to control some basic typographic attributes, such as kerning and tracking. **Kerning** refers to the amount of space between characters, while **tracking** refers to the amount of space between words. The styles to control an element's kerning and tracking are

```
letter-spacing: value;
word-spacing:   value;
```

where *value* is the size of space between individual letters or words. You specify these sizes with the same units that you use for font sizing. The default value for both kerning and tracking is 0 pixels. A positive value increases the letter and word spacing, while a negative value reduces the space between letters and words. If you choose to make your pages scalable for a variety of devices and resolutions, you will want to express kerning and tracking values as percentages or in em units.

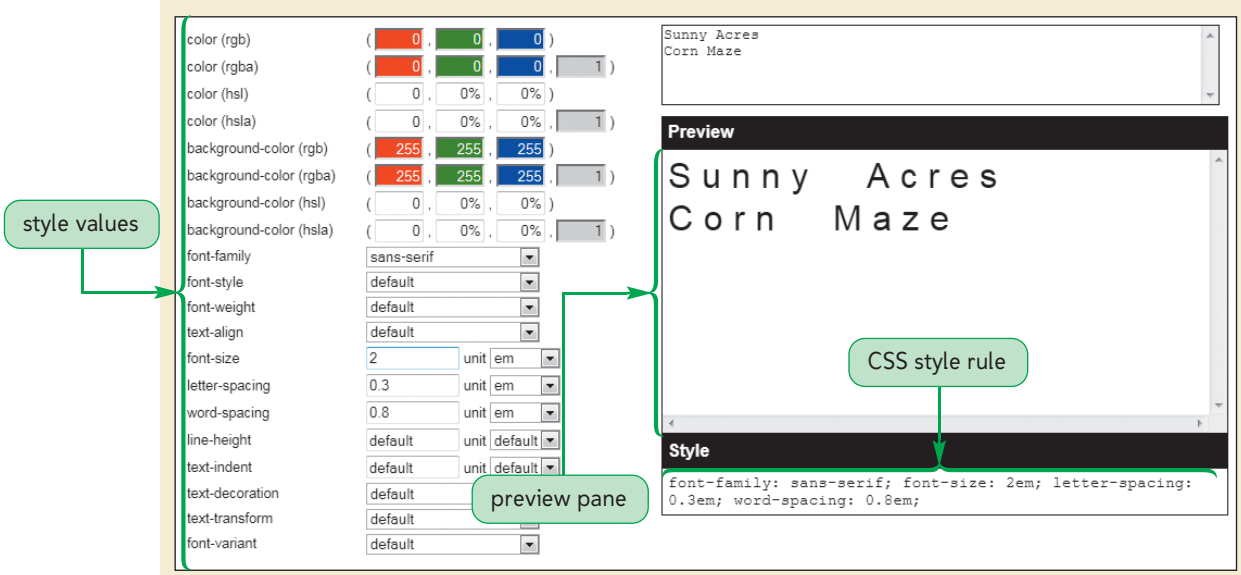
To see how modifying these values can affect the appearance of text, a demo page has been created for you.

To use the demo to explore kerning and tracking styles:

1. Open the **demo_css.htm** file from the tutorial.03/demo folder in your Web browser.

   The demo page contains a collection of CSS text styles. You can specify text style values using the boxes on the left side of the demo. In the top-right box, you can enter text to be displayed using the selected styles. The style as applied to the sample text appears in the middle box. The CSS code for the style appears in the bottom-right box. You press the Tab key to apply the style and view the results.
2. Click the top-right box, select and delete the text *Enter sample text here*, type **Sunny Acres**, press the **Enter** key, and then type **Corn Maze** on the second line. Press the **Tab** key to display this text in the Preview box.
3. Select **sans-serif** from the font-family box.
4. In the font-size box, replace the default text with **2**, and then select **em** in the corresponding unit box.
5. Enter **0.3** in the letter-spacing box, and then select **em** in the corresponding unit box. Press the **Tab** key.
6. Enter **0.8** in the word-spacing box and then select **em** in the corresponding unit box. Press the **Tab** key. Figure 3-30 shows the revised appearance of the text after applying the letter-spacing and word-spacing styles.

Figure 3-30 Using the demo page to explore letter-spacing and word-spacing



7. Experiment with other letter-spacing and word-spacing style values to see their effects on kerning and tracking.

Another typographic feature that you can set is **leading**, which is the space between lines of text. The style to set the leading value is

```
line-height: size;
```

where *size* is a specific length or a percentage of the font size of the text on the affected lines. If no unit is specified, most browsers interpret the number to represent the ratio of the line height to the font size. The standard ratio is 1.2:1, which means that the line height is usually 1.2 times the font size. By contrast, the style rule

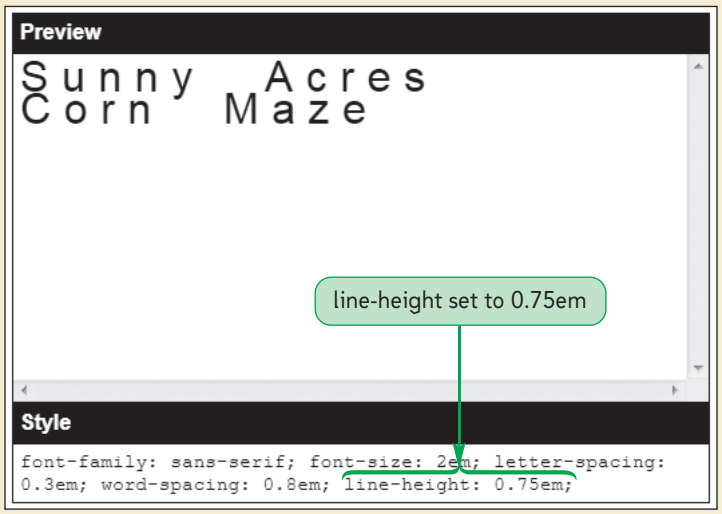
```
p {line-height: 2em;}
```

makes all paragraphs double-spaced. A common technique for multi-line titles is to give title text more impact using large fonts and small line heights. Use the demo page to see how this works.

To use the demo to explore leading styles:

1. If necessary, return to the **demo_css.htm** page in your Web browser.
2. Enter **0.75** in the line-height box, and then select **em** from the corresponding unit box.
3. Press the **Tab** key to apply the line-height style. Figure 3-31 shows the revised appearance of the text.

Figure 3-31 **Setting the line height**



An additional way to control text spacing is to set the indentation for the first line of a block of text. The style is

```
text-indent: size;
```

where *size* is a length expressed in absolute or relative units, or as a percentage of the width of the text block. For example, an indentation value of 5% indents the first line by 5% of the width of the block. The indentation value also can be negative, extending the first line to the left of the text block to create a **hanging indent**.

Now you can use what you've learned about spacing to make the changes that Tammy has suggested. To spread out her heading text, you'll set the kerning of the `h1` heading to 0.4em. You'll also set the indentation to 1em, moving the text of both `h1` and `h2` headings to the left.

To change the spacing of the headings on the Web site:

1. Return to the **sa_styles.css** file in your text editor.
2. Within the style rules for the `section h1` selector and the `h2` selector, insert the following style values (see Figure 3-32):

```
letter-spacing: 0.4em;
text-indent: 1em;
```

Figure 3-32 Defining letter-spacing and text-indent

```
section h1 {
   background-color: rgb(125, 186, 240);
   color: white;
   font-size: 1.7em;
   letter-spacing: 0.4em;
   text-indent: 1em;
}

h2 {
   background-color: rgb(0, 165, 0);
   color: white;
   color: rgba(255, 255, 255, 0.8);
   letter-spacing: 0.4em;
   text-indent: 1em;
}
```

3. Save your changes to the file and then reload the **home.htm** file in your browser. As shown in Figure 3-33, the indent and the spacing between the letters have increased.

Figure 3-33 Revised spacing in h1 and h2 headings

By increasing the kerning in the headings, you've made the text appear less crowded, making it easier to read.

Working with Font Styles

Browsers often apply default font styles to particular types of elements; for instance, `address` elements are usually displayed in italic. You also can specify a different font style using the style

```
font-style: type;
```

where *type* is `normal`, `italic`, or `oblique`. The italic and oblique styles are similar in appearance, but might differ subtly depending on the font in use.

You also have seen that browsers render certain elements in heavier fonts. For example, most browsers render headings in a boldfaced font. You can specify the font weight for any page element using the style

```
font-weight: weight;
```

where *weight* is the level of bold formatting applied to the text. The *weight* value ranges from 100 to 900 in increments of 100. In practice, however, most browsers cannot

TIP

To prevent your browser from displaying address text in italic, you can set the `font-style` property to normal.

distinguish between nine different font weights. For practical purposes, you can assume that 400 represents normal (not bold) text, 700 is bold text, and 900 represents heavy bold text. You also can use the keywords `normal` or `bold` in place of a weight value, or you can express the font weight relative to the text of the containing element, using the keywords `bolder` or `lighter`.

TIP

To prevent your browser from displaying page headings in bold, you can set the `font-weight` property to normal.

Another style you can use to change the appearance of your text is

```
text-decoration: type;
```

where the *type* values include `none` (for no decoration), `underline`, `overline`, and `line-through`. You can apply several decorative features to the same element by listing them as part of the text-decoration style. For example, the style

```
text-decoration: underline overline;
```

places a line under and over the text in the element. Note that the text-decoration style has no effect on nontextual elements, such as inline images.

TIP

To prevent your browser from underlining hypertext links, set the `text-decoration` property of the hypertext element to none.

To control the case of the text within an element, use the style

```
text-transform: type;
```

where *type* is `capitalize`, `uppercase`, `lowercase`, or `none` (to make no changes to the text case). For example, if you want to capitalize the first letter of each word in an element, you could use the following style:

```
text-transform: capitalize;
```

Finally, you can display text in uppercase letters and a small font using the style

```
font-variant: type;
```

where *type* is `normal` (the default) or `small-caps` (small capital letters). Small caps are often used in legal documents, such as software agreements, in which the capital letters indicate the importance of a phrase or point, but the text is made small so as to not detract from other elements in the document.

REFERENCE

Setting Font and Text Appearance

- To specify the font style, use

  ```
  font-style: type;
  ```

  where *type* is `normal`, `italic`, or `oblique`.
- To specify the font weight, use

  ```
  font-weight: type;
  ```

  where *type* is `normal`, `bold`, `bolder`, `light`, `lighter`, or a font weight value.
- To specify a text decoration, use

  ```
  text-decoration: type;
  ```

  where *type* is `none`, `underline`, `overline`, or `line-through`.
- To transform text, use

  ```
  text-transform: type;
  ```

  where *type* is `capitalize`, `uppercase`, `lowercase`, or `none`.
- To display a font variant of text, use

  ```
  font-variant: type;
  ```

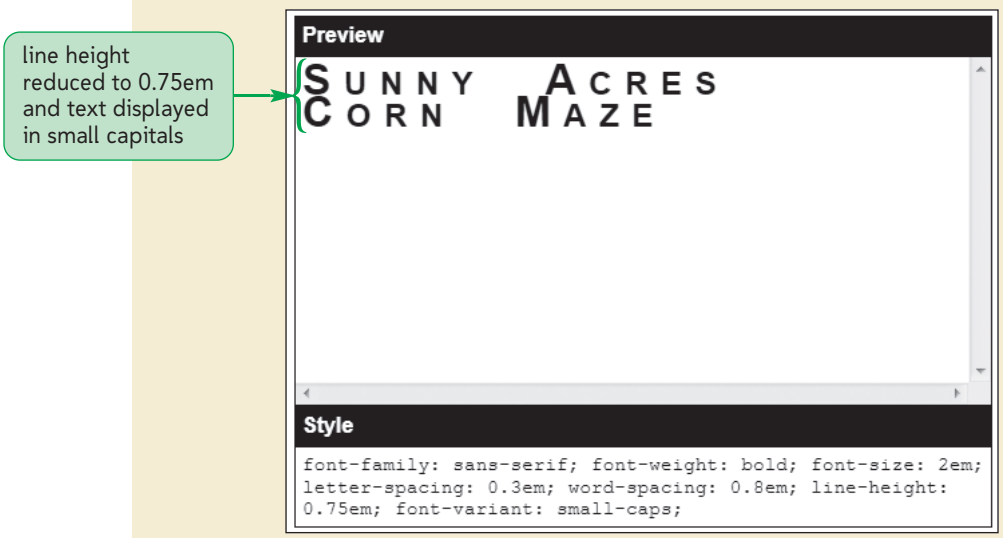
  where *type* is `normal` or `small-caps`.

To see the impact of these styles, you'll return to the demo page.

To use the demo to view the various font styles:

1. Return to the **demo_css.htm** page in your Web browser.
2. Select **bold** in the font-weight box.
3. Select **small-caps** in the font-variant box. Figure 3-34 shows the impact of applying the font-weight and font-variant styles.

Figure 3-34 Applying the font-weight and font-variant styles



4. You've completed your work with the CSS demo page. If you wish, continue to explore different CSS font and text styles. When you're finished, close the demo Web page.

Aligning Text Horizontally and Vertically

In earlier examples from this tutorial, you saw how the text-align style could be used to align text horizontally. The specific style is

```
text-align: alignment;
```

where *alignment* is `left`, `right`, `center`, or `justify` (align the text with both the left and the right margins). To vertically align the text, use the style

```
vertical-align: alignment;
```

where *alignment* is one of the keywords described in Figure 3-35.

Figure 3-35 Values of the vertical-align style

Value	Description
baseline	Aligns the element with the bottom of lowercase letters in surrounding text (the default)
bottom	Aligns the bottom of the element with the bottom of the lowest element in surrounding content
middle	Aligns the middle of the element with the middle of the surrounding content
sub	Subscripts the element
super	Superscripts the element
text-bottom	Aligns the bottom of the element with the bottom of the font of the surrounding content
text-top	Aligns the top of the element with the top of the font of the surrounding content
top	Aligns the top of the element with the top of the tallest object in the surrounding content

TIP

The subscript and superscript styles lower or raise text vertically, but do not resize it. To create true subscripts and superscripts, you also must reduce the font size.

Instead of using keywords, you can specify a length or a percentage for an element to be aligned relative to the surrounding content. A positive value moves the element up and a negative value lowers the element. For example, the style

```
vertical-align: 50%;
```

raises the element by half of the line height of the surrounding content, while the style

```
vertical-align: -100%;
```

drops the element an entire line height below the baseline of the current line.

Combining All Text Formatting in a Single Style

You've seen a lot of different text and font styles. You can combine most of them into a single property using the shortcut `font` property

```
font: font-style font-variant font-weight font-size/line-height
font-family;
```

where *font-style* is the font's style, *font-variant* is the font variant, *font-weight* is the weight of the font, *font-size* is the size of the font, *line-height* is the height of each line, and *font-family* is the font face. For example, the style

```
font: italic small-caps bold 18px/24px Arial, sans-serif;
```

displays the text of the element in italic, bold, and small capital letters in Arial or another sans-serif font, with a font size of 18 pixels and spacing between the lines of 24 pixels. You do not have to include all of the values in the `font` property; the only required values are *font-size* and *font-family*. A browser assumes the default value for any omitted property. However, you must place any properties that you do include in the order indicated above.

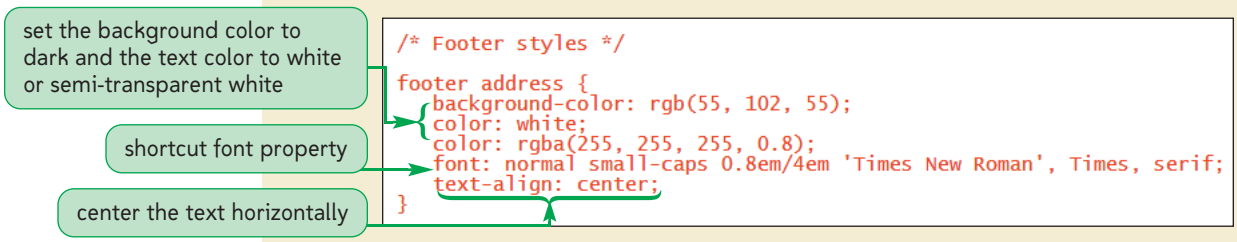
Tammy would like the address in the page footer formatted differently from the default style imposed by the browser's internal style sheet. She suggests that you display the text in a semi-transparent white Times New Roman font on a dark green background and centered on the page. She also suggests that you use the small-cap font variant to add visual interest, and increase the height of the address line to 4em. To make your CSS code more compact, you'll add all of the font values within a single line using the `font` property.

To change the style of the `address` element:

1. Return to the **sa_styles.css** file in your text editor.
2. At the bottom of the style sheet, add the following style rule for the `address` element nested within the `footer` element (see Figure 3-36):

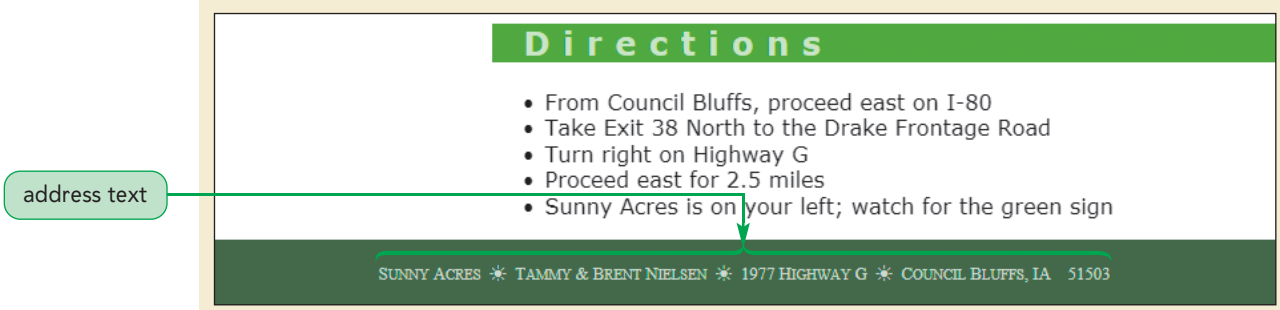
```
/* Footer styles */

footer address {
   background-color: rgb(55, 102, 55);
   color: white;
   color: rgba(255, 255, 255, 0.8);
   font: normal small-caps 0.8em/4em 'Times New Roman', Times, serif;
   text-align: center;
}
```

Figure 3-36 Designing the footer address



3. Save your changes to the file and then reload the **home.htm** file in your Web browser. Scroll to the bottom of the page and verify that the style of the `address` element has been changed as shown in Figure 3-37.

Figure 3-37 Reformatted address text



Tammy likes the way the fonts appear on her Web site. She especially likes the fact that because these changes were made in a CSS style sheet, the styles automatically apply to the other Web pages on the site, as well as any Web page she adds to the site in the future.

INSIGHT

Setting up Alternative Style Sheets

Many browsers recognize alternative style sheets. This is particularly useful in situations when you are supporting users who have special needs, such as a need for large text with highly contrasting colors. To support these users, you can make an alternative style sheet with the `link` element, using the code

```
<link href="url1" rel="alternate stylesheet"
      type="text/css" title="title1" />

<link href="url2" rel="alternate stylesheet"
      type="text/css" title="title2" />
```

where *url1*, *url2*, and so forth are the URLs of the style sheet files, and *title1*, *title2*, etc. are the titles of the alternate style sheets. For example, the following HTML code creates links to two style sheets, named *Large Text* and *Regular Text*:

```
<link href="large.css" rel="alternate stylesheet"
 type="text/css" title="Large Text" />

<link href="regular.css" rel="alternate stylesheet"
 type="text/css" title="Regular Text" />
```

Browsers that support alternative style sheets provide a menu option or toolbar that enables users to select which style sheet to apply. Among current browsers, Firefox, Opera, and Safari support alternate style sheets.

Working with Web Fonts

Text design on the Web largely has been limited to a few Web safe fonts that are supported by all major browsers. It would be better if a browser would automatically download whatever fonts are required for a Web page in the same way it downloads images. Specifications for downloadable fonts, or **Web fonts**, have been around for several years, but most browsers have begun to support this technology only in recent years. However, different browsers support different font file formats. Figure 3-38 describes these different formats and their current levels of browser support.

Figure 3-38 **Web font formats**

Format	Description	Browser
TrueType/OpenType	The most common font format, freely available on most computers; no support for licensing.	Chrome, Firefox, Opera, Safari
Embedded OpenType	Proprietary format developed by Microsoft for Internet Explorer; supports licensing and security against unauthorized use.	Internet Explorer
Scalable Vector Graphics	An XML vocabulary designed to describe resizable graphics and primarily supported by mobile browsers.	Chrome, Opera, Safari
Web Open Font Format	A new standard for Web fonts that is quickly gaining support; provides support for font licensing.	Firefox, other browsers in development

Web font files are available on several sites on the Web. In many cases, these are not free fonts and you must pay for their use. Other fonts are free but are licensed only for non-commercial use. You always should check the EULA (End User License Agreement) before downloading and using a Web font to make sure you are in compliance with the license. Finally, many Web fonts are available through **Web Font Service Bureaus**, which supply Web fonts on their servers that page designers can link to for a fee.

The great advantage of a Web font is that it gives a designer control over the typeface used in a document. The disadvantage is that it becomes another file for the browser to download and install, adding to the time required to render the page.

The `@font-face` Rule

To access and load a Web font, you add the rule

```
@font-face {
   font-family: name;
   src: url(url) format(text);
   descriptor:value;
   descriptor:value;
   …
}
```

to the style sheet, where *name* is the name assigned to the Web font, *url* is the location of the font definition file, *text* is an optional text description of the font format, and the *descriptor*:*value* pairs are optional style properties that describe how and when the font should be used. For example, the following `@font-face` rule defines a font face named *GentiumBold*:

```
@font-face {
   font-family: GentiumBold;
   src: url(GentiumB.ttf) format('truetype');
   font-weight: bold;
}
```

The GentiumBold font in this code is a TrueType font based on a description stored in the *GentiumB.ttf* file. The `font-weight` properties tell browsers to apply this font only for bold text. Note that at the time of this writing, you should avoid including the `font-weight` and `font-style` properties in the `@font-face` rule because those features are not well supported by most browsers and can produce unexpected results.

Once you've defined a font using the `@font-face` rule, you can use the font elsewhere in your style sheets by including the font name in your font lists. For example, the style

```
font-family: GentiumBold, 'Arial Black', Gadget, sans-serif;
```

attempts to load the GentiumBold font defined above, followed by Arial Black, Gadget, and then a sans-serif font of the browser's choosing.

Installing a Cross-Browser Web Font

TIP

The rule always should be placed at the top of the style sheet, before any styles that specify the use of a Web font.

To support all of the font formats described in Figure 3-38, you can add additional source files to the `@font-face` rule. A browser then will go through the list, loading the last font file format from the list that it supports. Part of the challenge is that at the time of this writing, Internet Explorer does not support all of the features of the `@font-face` rule and returns an error if it attempts to load another font file. Thus, to define a font that works across different browsers, you should design the `@font-face` rule as follows

```
@font-face {
   font-family: name;
   src: url(eot);
   src: local('☺'),
        url(woff) format(text),
        url(ttf) format(text),
        url(svg) format(text);
   descriptor:value;
   …
}
```

where *eot* is the font defined in an Embedded OpenType file (the only format supported by Internet Explorer at the time of this writing), *woff* is the font defined in a Web Open Font file, *ttf* is the font defined in a TrueType or an OpenType file, and *svg* is the font defined in a Scalable Vector Graphics file. The `local('☺')` part of this code is a programming hack developed by Paul Irish (*www.paulirish.com*) to prevent Internet Explorer from attempting to load the other font files, causing an error. Note that to enter this symbol, your text must be stored using Unicode encoding rather than ASCII or ANSI. The sa_styles.css file already has been created for you using that text encoding.

Tammy is not completely pleased with the Verdana font you've used for the home page and would like a typeface with thinner lines. She has located a Web font named *NobileRegular* that she is free to use under the End User License Agreement and thinks would work better. In acquiring this font, she also obtained the following `@font-face` rule that can be used to load the font into a CSS style sheet:

```
@font-face {
   font-family: 'NobileRegular';
   src: url('nobile-webfont.eot');
   src: local('☺'),
        url('nobile-webfont.woff') format('woff'),
        url('nobile-webfont.ttf') format('truetype'),
        url('nobile-webfont.svg#webfontsKo9tqe9') format('svg');
}
```

Rather than retype this code, you'll copy it from her text file and paste it into your style sheet. Add this font definition now.

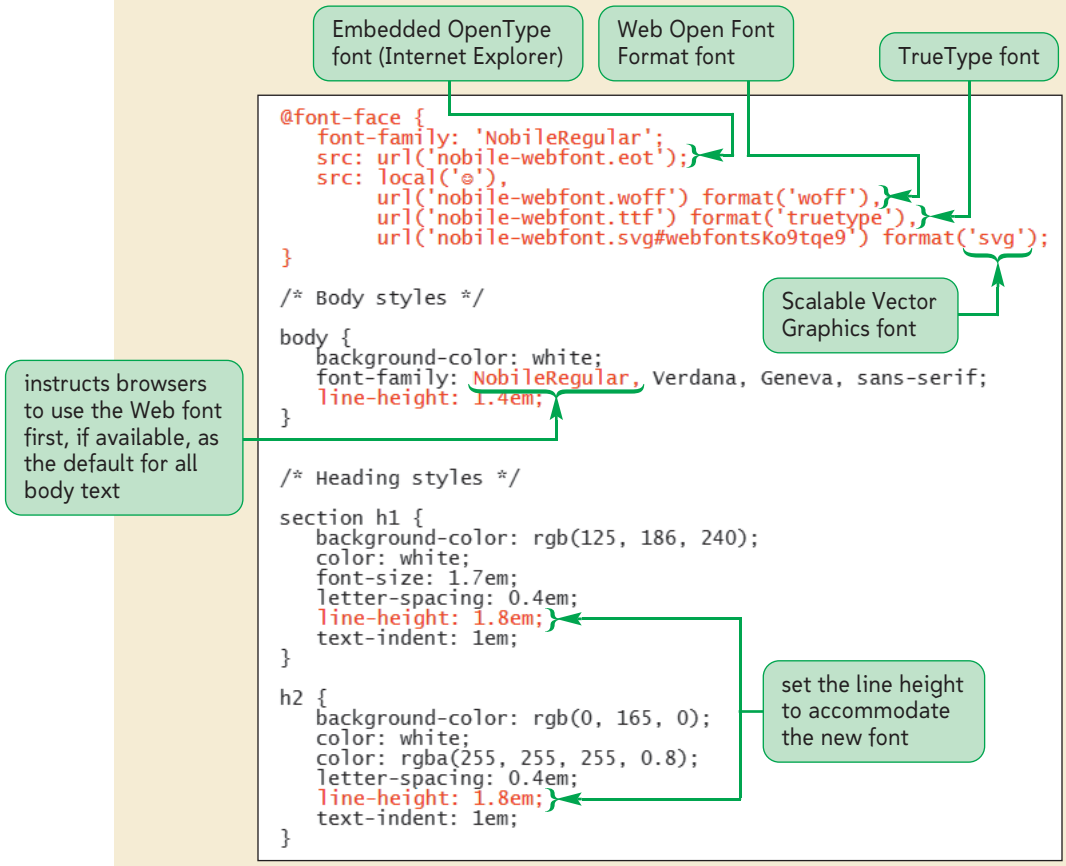
To insert and apply the NobileRegular font:

1. Using your text editor, open the **nobile.txt** text file located in the tutorial.03/tutorial folder.
2. Copy the `@font-face` rule located at the top of the file.
3. Return to the **sa_styles.css** file in your text editor.
4. Paste the copied text of the `@font-face` rule into your style sheet directly above the style rule for the `body` element.

   Next, you'll revise the style rule for the `body` element so that it uses the NobileRegular font as the first option, if available and supported by the browser. You'll also set the line height of body text to 1.4 em and the line height of the page headings to 1.8 em to accommodate the metrics of this new font.
5. Within the `font-family` property for the `body` element, insert **NobileRegular** followed by a comma and a space, at the beginning of the font list.
6. Add the property **`line-height: 1.4em;`** to the style rule for the `body` element and **`line-height: 1.8em;`** to the style rules for the `h1` and `h2` elements.

   Figure 3-39 highlights the newly inserted text in the style sheet.

Figure 3-39 Inserting a Web font



7. Save your changes to the file and then reload **home.htm** in your Web browser. As shown in Figure 3-40, the body text of the Web page has changed, using the NobileRegular font in place of the previous font.

Figure 3-40 **Text rendered in the NobileRegular font**

Understanding the CSS @rules

The `@font-face` rule is one example of a **CSS *@rule***, which specifies a command or directive that controls how browsers interpret and run the contents of a CSS style sheet. Figure 3-41 lists the *@rules* and describes how they are used.

Figure 3-41 **CSS @rules**

@rule	Description
`@charset "encoding";`	Defines the character encoding used in an external style sheet where *encoding* is the name of the character set
`@import url(url) media`	Imports an external style sheet file located at *url*. The optional *media* attribute provides a comma-separated list of media devices to be used with the style sheet
`@media media { styles }`	Targets the style rules in *styles* to devices that match the media types specified in *media*
`@page location { margins }`	Defines the page margins for printed output where *location* is either `left`, `right`, or `first` for left page, right page, or first page, and *margins* set the margin widths
`@font-face { font_description }`	Defines the properties of a custom Web font where *font_description* indicates the source and features of the font
`@namespace prefix uri`	Defines an XML namespace where *prefix* is the namespace prefix and *uri* is the location of the namespace

In general, all of the CSS *@rules* should be placed at the top of the style sheet before the style properties that may use them.

PROSKILLS

Problem Solving: Finding and Fixing Style Sheet Errors

As a style sheet increases in size and complexity, you will likely encounter a major headache: a Web page whose appearance in a browser does not match the style you planned for it. Once you recognize that a syntax error in your style sheet likely caused the browser to ignore some of your style rules, you need to figure out where the error is. The following are some errors that can cause your style sheets to fail:

- *Missing semicolons*—All style properties need to be separated by a semicolon. If you omit a semicolon, all subsequent properties within the affected style rule will be ignored.
- *Missing curly braces*—Every style rule must be enclosed within a set of curly braces. Failure to close off the style rule will cause that style rule and all subsequent rules to be ignored.
- *Missing closing quotes*—If you use quotes within typeface names or other text strings, you need to enclose the entire text within either double or single quotation marks. Omitting a closing quotation mark will cause the rest of the style sheet to fail.
- *Typos*—All selector names, style properties, and style values need to be typed correctly.
- *Improper application of a style*—Some styles are associated with all elements, but some are not. For instance, you can't change the color of an inline image using the background-color style, no matter how much you might want to.
- *Conflicting properties*—Remember the general principle that *the more specific style outweighs the less specific*. A style rule in one part of a style sheet might be superseded by a rule somewhere else in the file.
- *Inherited properties*—Properties are also inherited down the document tree. A style property applied to a `section` element affects all nested elements unless you override it with another rule.

If a problem persists after finding and fixing these common errors, try other techniques to locate the source of the trouble, such as removing style rules one by one until you locate the source of the trouble. Also consider using the `!important` keyword to temporarily force the browser to apply a rule regardless of specificity. Finally, many browsers now include debugging tools to assist you in writing your CSS code. Use these tools whenever you encounter an error that you can't easily find and fix.

You've completed your work on creating and editing the text styles used in the Sunny Acres home page. In the next session, you'll explore how to design styles for hypertext links and lists, and you'll learn how to use CSS to add special visual effects to your Web pages.

REVIEW

Session 3.2 Quick Check

1. Provide a style rule to display the text of all `address` elements nested within a `footer` element in red.
2. The initial `h1` heading in a document has the id *top*. Provide a style rule to display the text of this `h1` heading in Arial, Helvetica, or a sans-serif font.
3. Provide a style rule to display all `blockquote` elements belonging to the *reviews* class in italic and indented 3em.
4. Provide a style rule to center all `h1` through `h6` headings, and to display the text with normal weight.
5. What is the difference between an absolute unit and a relative unit?
6. Provide a style rule to double space the text of all paragraphs in the document.
7. Provide a style rule to remove underlining from the hypertext links for all elements marked with the `<a>` tag and nested within a navigation list.
8. If you want to use Web fonts with Internet Explorer, what font file format should you use?
9. Provide the `@font-face` rule to create a Web font named *Cantarell* based on the font file *cantarell.ttf*. Which browsers will be able to work with this font file?

SESSION 3.3 VISUAL OVERVIEW

The first-of-type structural pseudo-class matches the element in a collection.

```
nav ul li:first-of-type {
   text-transform: uppercase;
}
```

The text-transform property transforms the text in uppercase, lowercase, or sentence case.

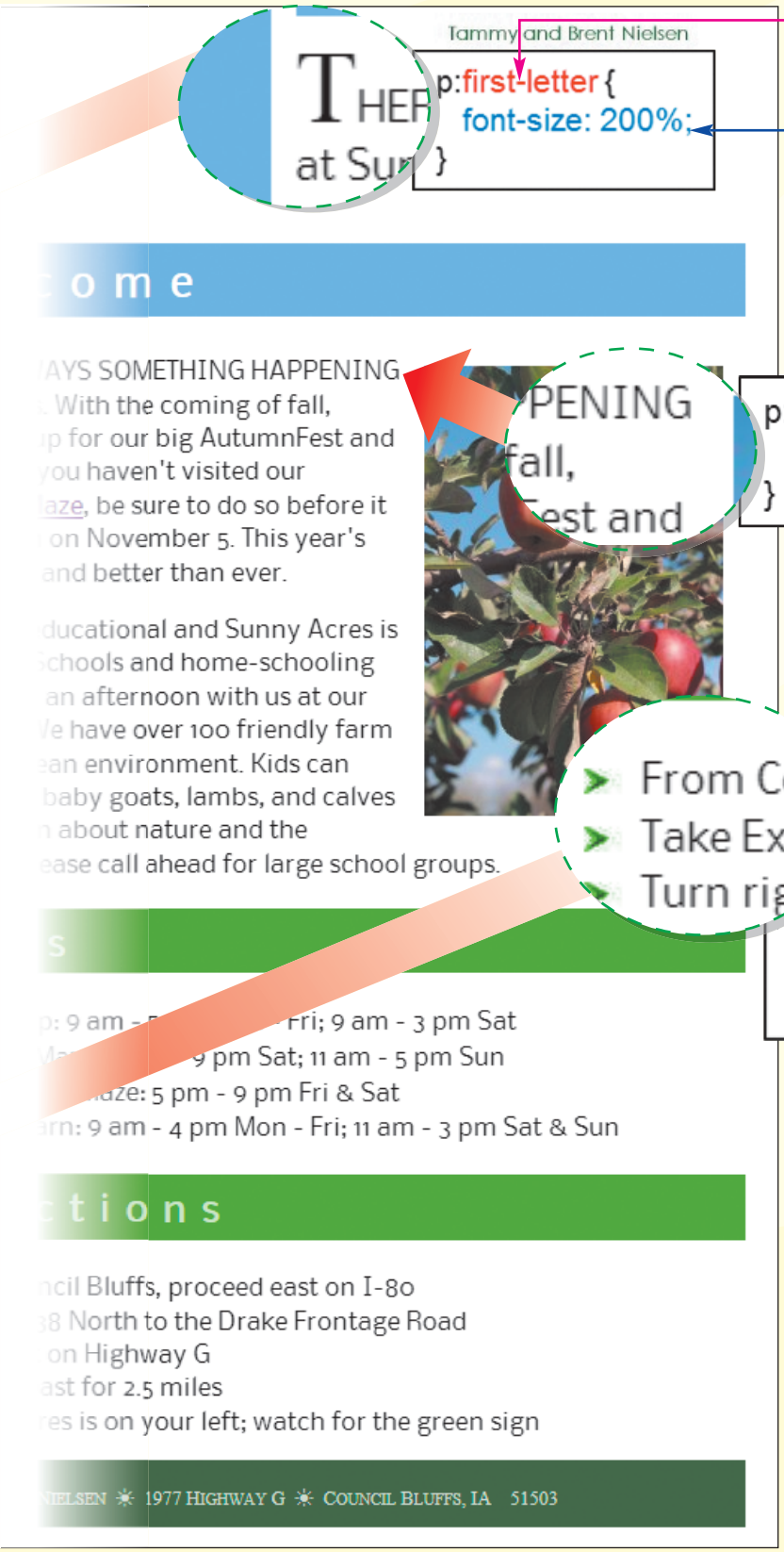
The hover dynamic pseudo-class matches the situation when a user is hovering the mouse pointer over the element.

```
nav ul li:hover {
   background-color: darkblue;
}
```

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

The list-style-type property specifies the marker that is displayed with the list.

LISTS AND PSEUDO-ITEMS



Designing Styles for Lists

Tammy has placed her navigation links in an unordered list set on the left page margin. As with all unordered lists, browsers display the list contents with bullet markers. Tammy would like the bullets removed. To alter the appearance of this navigation list, you change the default style applied by browsers.

Choosing a List Style Type

To change the marker displayed in ordered or unordered lists, you apply the style

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

where `type` is one of the markers discussed in Figure 3-42.

Figure 3-42 List style types

list-style-type	Marker (s)
disc	●
circle	○
square	■
decimal	1, 2, 3, 4, …
decimal-leading-zero	01, 02, 03, 04, …
lower-roman	i, ii, iii, iv, …
upper-roman	I, II, III, IV, …
lower-alpha	a, b, c, d, …
upper-alpha	A, B, C, D, …
lower-greek	α, β, γ, δ, …
upper-greek	A, B, Γ, Δ, …
none	no marker displayed

For example, to display an ordered 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` element:

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

REFERENCE

Designing a List

- To define the appearance of the list marker, use the style

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

where *type* is `disc`, `circle`, `square`, `decimal`, `decimal-leading-zero`, `lower-roman`, `upper-roman`, `lower-alpha`, `upper-alpha`, `lower-greek`, `upper-greek`, or `none`.

- To insert a graphic image as a list marker, use the style

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

where *url* is the URL of the graphic image file.

- To set the position of list markers, use the style

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

where *position* is `inside` or `outside`.

- To define all of the list style properties in a single style, use the following style:

```
list-style: type url(url) position;
```

- To set the indentation of a list, apply the style

```
padding-left: size;
```

where *size* is the length that the list should be indented.

List style types can be used with contextual selectors to create an outline style based on nested lists. Figure 3-43 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 the ordered list within the outline hierarchy. The top level is displayed with uppercase Roman numerals; the bottom level, which is nested within three other ordered lists, uses lowercase letters for markers.

Figure 3-43 Creating an outline style

```
ol          {list-style-type: upper-roman;}
ol ol       {list-style-type: upper-alpha;}
ol ol ol    {list-style-type: decimal;}
ol ol ol ol {list-style-type: lower-alpha;}
```

styles

```
<ol>
   <li>First Major Heading
      <ol>
         <li>Topic 1</li>
         <li>Topic 2
            <ol>
               <li>Item 1</li>
               <li>Item 2
                  <ol>
                     <li>Paragraph 1</li>
                     <li>Paragraph 2</li>
                  </ol>
               </li>
               <li>Item 3</li>
            </ol>
         </li>
         <li>Topic 3</li>
      </ol>
   </li>
   <li>Second Major Heading</li>
   <li>Third Major Heading</li>
</ol>
```

HTML code

I. First Major Heading
   A. Topic 1
   B. Topic 2
      1. Item 1
      2. Item 2
         a. Paragraph 1
         b. Paragraph 2
      3. Item 3
   C. Topic 3
II. Second Major Heading
III. Third Major Heading

text as rendered by the browser

Because Tammy wants to remove the markers from the navigation list, you'll set the `list-style-type` value of that list to `none`. Because you don't want to remove the bullet markers from all lists on the Web site, you'll use a contextual selector that targets only unordered lists nested within a `nav` element.

To remove the bullets from the navigation links:

1. Return to the **sa_styles.css** file in your text editor.
2. Directly below the style rule for the `h2` element, insert the following (see Figure 3-44):

```
/* Navigation list styles */

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

Figure 3-44 Removing bullet markers from navigation list items

```
   letter-spacing: 0.4em;
   line-height: 1.8em;
   text-indent: 1em;
}

/* Navigation list styles */

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

3. Save your changes to the file and then reload the **home.htm** file in your Web browser. Verify that the bullet markers have been removed from the items in the navigation list.

Using Images for List Markers

You can supply your own graphic image for the list marker using the style

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

where *url* is the URL of a graphic file containing the marker image. This is only done for unordered lists, in which the marker is the same for every list item. For example, the style rule

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

displays items from unordered lists marked with the graphic image in the *redball.gif* file. Tammy suggests that you display the list of hours and driving directions using a green arrow graphic she created. Both of these unordered lists are immediately preceded by an `h2` element and nested within a `section` element, so the style rule to change the list marker is:

```
section h2+ul {
   list-style-image: url(arrow.png);
}
```

You'll add this style rule to your style sheet.

To use an image for a list bullet:

1. Return to the **sa_styles.css** file in your text editor.
2. Directly below the style rule for the closing paragraph, insert the following as shown in Figure 3-45:

```
section h2+ul {
   list-style-image: url(arrow.png);
}
```

Figure 3-45 Displaying an image in place of a marker

```
section p.closing {
   color: rgb(0, 165, 0);
   text-align: right;
}

section h2+ul {
   list-style-image: url(arrow.png);
}
```

marker image

3. Save your changes to the file and then reload the **home.htm** file in your Web browser. As shown in Figure 3-46, the items for farm hours and driving directions are now displayed with green arrow markers.

Figure 3-46 Green arrow markers

Changing the List Layout

Tammy likes the revised markers, but she thinks there's too much empty space to the left of the entries in the navigation list. She would like you to modify the layout to remove the extra space. Each list item is treated as a group-level element and placed within its own box. By default, most browsers place the list marker to the left of this box, lining up each marker with its list item. You can change this default behavior using the property

```
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 3-47 shows how the `list-style-position` property affects the appearance of a bulleted list.

Figure 3-47 Marker positions

- Farm Shop: 9 am - 5 pm Mon - Fri; 9 am - 3 pm Sat
- The Corn Maze: 11 am - 9 pm Sat; 11 am - 5 pm Sun
- The Haunted Maze: 5 pm - 9 pm Fri & Sat
- Petting Barn: 9 am - 4 pm Mon - Fri; 11 am - 3 pm Sat & Sun

list-style-position: outside;

- Farm Shop: 9 am - 5 pm Mon - Fri; 9 am - 3 pm Sat
- The Corn Maze: 11 am - 9 pm Sat; 11 am - 5 pm Sun
- The Haunted Maze: 5 pm - 9 pm Fri & Sat
- Petting Barn: 9 am - 4 pm Mon - Fri; 11 am - 3 pm Sat & Sun

list-style-position: inside;

All three of the list styles can be combined within the property

```
list-style: type image position;
```

where *type* is the marker type, *image* is an image to be displayed in place of the marker, and *position* is the location of the marker. For example, the style rule

```
ul {list-style: circle url(bullet.jpg) inside;}
```

displays all ordered lists with the marker found in the *bullet.jpg* image placed inside the containing block. If a browser is unable to display the bullet image, it uses a circle marker instead. You do not need to include all three style properties with the list style. Browsers will set any property you omit to the default value.

TIP

Any page element can be turned into a list item by applying the `display: list-item` style.

By default, browsers offset ordered and unordered lists from the surrounding text. Tammy has noticed this and worries that the entries in the navigation list are crowding the text in the Welcome paragraph by being shifted too far to the right. She would like you to move the navigation list to the left, aligning it more toward the left edge of the Sunny Acres logo. To reduce the space that the browser inserts, you use the style property

```
padding-left: size;
```

where *size* is the length that the list should be shifted to the right. The `padding-left` style is one of the styles you'll study in more detail in the next tutorial on page layout. For now, you'll limit your use of this style to reduce the extra space inserted by the browser in the navigation list. After some experimenting, you settle on the following value for the style:

```
padding-left: 0.5em;
```

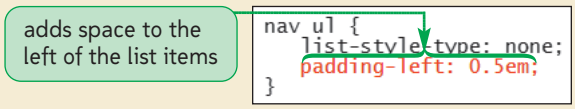
You'll add this style to the style rule for the navigation list.

To move the navigation list to the left:

1. Return to the **sa_styles.css** file in your text editor.
2. Within the style rule for the navigation list, insert the following style value, as shown in Figure 3-48:

```
padding-left: 0.5em;
```

Figure 3-48 Setting the padding space within a list



3. Save your changes to the file and then reload the **home.htm** file in your Web browser. Verify that the entries in the navigation list have been shifted to the left, aligned roughly with the left edge of the Sunny Acres logo.

To complete the design style for the navigation list, Tammy wants to change the background color of the list to sky blue, and to increase the space between the items by increasing the line height to 3.5em. Also, while browsers underline hypertext by default, the current Web design standard is to not underline the links in a navigation list; thus, Tammy also wants to change the text color of the navigation links to white and remove the underlining. You'll add these styles to the style sheet now.

To reformat the navigation list:

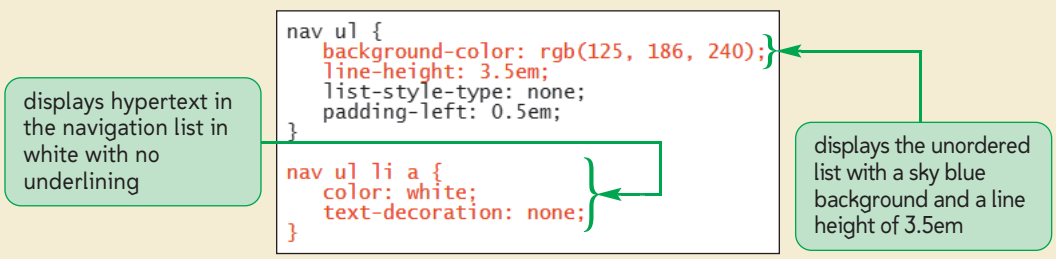
1. Return to the **sa_styles.css** file in your text editor.
2. Add the following style properties to the `nav ul` style rule

```
background-color: rgb(125, 186, 240);
line-height: 3.5em;
```

3. Directly below the `nav ul` style rule, add the following rule for navigation hypertext links (see Figure 3-49):

```
nav ul li a {
   color: white;
   text-decoration: none;
}
```

Figure 3-49 Formatting the navigation list



4. Save your changes to the file and then reload the **home.htm** file in your Web browser. Figure 3-50 shows the revised appearance of the navigation sidebar.

Figure 3-50 Revised navigation list

Using Pseudo-Classes and Pseudo-Elements

Without underlines, there is no visual clue that the links in the navigation list act as hypertext. Tammy has seen Web sites in which links are underlined or highlighted 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. Tammy would like you to add a rollover effect to the navigation list.

Pseudo-Classes

Rollover effects 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. Styles for pseudo-classes are created using the syntax

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

where *selector* is an element or a group of elements within a document, *pseudo-class* is the name of a pseudo-class, and *styles* are the style properties applied to that selector pseudo-class. Pseudo-classes are organized into dynamic and structural classes. A dynamic pseudo-class changes with a user's actions. For example, the `visited` pseudo-class indicates whether a hypertext link previously has been visited by the user. To display the text of all such links in red, you would apply the following style rule to the `a` element:

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

To change the text color to blue when users hover the mouse pointer over the link, you would add the following style rule:

```
a:hover {color: blue;}
```

Figure 3-51 lists the other dynamic pseudo-classes supported by CSS.

Figure 3-51 Dynamic pseudo-classes

Pseudo-Class	Description	Example
`link`	The link has not yet been visited by the user.	`a:link {color: red;}`
`visited`	The link has been visited by the user.	`a:visited {color: green;}`
`active`	The element is in the process of being activiated or clicked by the user.	`a:active {color: yellow;}`
`hover`	The mouse pointer is hovering over the element.	`a:hover {color: blue;}`
`focus`	The element has received the focus of the keyboard or mouse pointer.	`input:focus {background-color: yellow;}`

In some cases, two or more pseudo-classes can apply to the same element. For example, a hypertext link can be both previously visited and hovered over. In such situations, the standard cascading rules apply—the pseudo-class that is listed last is applied to the element. You should enter the hypertext pseudo-classes in the following order in your style sheets—`link`, `visited`, `hover`, and `active`. 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 links that previously have been visited or clicked. The `hover` pseudo-class follows, for the situation in which a user has once again moved the mouse pointer over a hypertext link before clicking the link. The `active` pseudo-class is last, representing the exact instant in which a link is clicked by a user. Any styles set with the `link` pseudo-class are inherited by the `visited`, `hover`, and `active` pseudo-classes.

REFERENCE

Creating a Hypertext Rollover

- To create a rollover for a hypertext link, apply the styles

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

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

The `hover`, `active`, and `focus` dynamic pseudo-classes also can be applied to non-hypertext elements. For example, the following style rule causes the background color of each entry in a navigation list to change to medium blue whenever a user hovers the mouse pointer over it:

```
nav ul li:hover {
   background-color: rgb(83, 142, 213);
}
```

Tammy suggests that you add this style rule to provide users with the visual feedback that is missing from the navigation links as a result of removing the underlining.

To apply the hover pseudo-class:

1. Return to the **sa_styles.css** file in your text editor.
2. Add the following style rule, as shown in Figure 3-52:

```
nav ul li:hover {
   background-color: rgb(83, 142, 213);
}
```

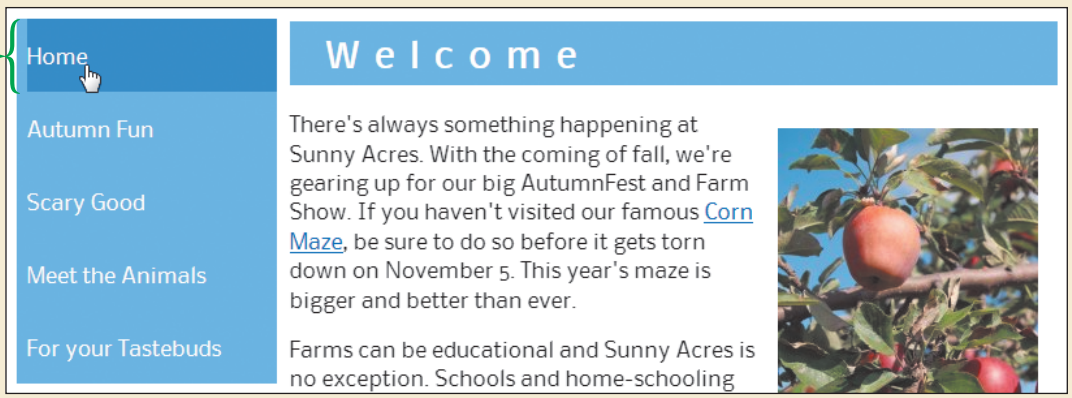
Figure 3-52 Applying the hover pseudo-class

change the background color to medium blue when the user hovers the mouse pointer over the list items

```
nav ul {
   background-color: rgb(125, 186, 240);
   line-height: 3.5em;
   list-style-type: none;
   padding-left: 0.5em;
}

nav ul li:hover {
   background-color: rgb(83, 142, 213);
}
```

3. Save your changes to the file and then reload **home.htm** in your Web browser. Move your mouse pointer over the navigation list items. As shown in Figure 3-53, the background color of the navigation list items changes to medium blue in response to the hover event.

Figure 3-53 Viewing the rollover effect



INSIGHT

Deprecated Attributes for Hypertext Links

Earlier versions of HTML did not support CSS and the `link`, `visited`, `hover`, 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 the attributes

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

to the `<body>` tag, 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. The `link`, `vlink`, and `alink` attributes have been deprecated and their use is discouraged, but you still might see them in the code of older Web pages.

Structural pseudo-classes are used to classify items based on their locations within the hierarchy of page elements. For example, the style rule

```
body:first-child {background-color: yellow;}
```

changes the background color to yellow in the first child of the `body` element within the document. Notice that the selector does not specify the element. It could be an `h1` heading, a `section` element, or anything else, as long as it is the first child of the `body` element. Figure 3-54 describes the structural pseudo-classes.

Figure 3-54 Structural pseudo-classes

Pseudo-Class	Matches
`root`	The top element in the document hierarchy (the `html` element)
`empty`	An element with no children
`only-child`	An element with no siblings
`first-child`	The first child of the parent element
`last-child`	The last child of the parent element
`first-of-type`	The first element of the parent that matches the specified type
`last-of-type`	The last element of the parent that matches the specified type
`nth-of-type(n)`	The n^{th} element of the parent of the specified type
`nth-last-of-type(n)`	The n^{th} from the last element of the parent of the specified type
`only-of-type`	An element that has no siblings of the same type
`lang(code)`	The element that has the specified language indicated by *code*
`not(s)`	An element not matching the specified selector, *s*

The first entry in a navigation list is often the most prominent or important. On the Sunny Acres Web site, the first entry is linked to the site's home page. Tammy would like this link to always be displayed in capital letters. Although you could edit the HTML code to make this happen, you decide to make this change to the style sheet using the following style rule because you want to separate content from design:

```
nav ul li:first-of-type {
   text-transform: uppercase;
}
```

The style rule uses the `first-of-type` pseudo-class to apply the uppercase transformation to the first `li` element found nested within the navigation list. You'll add this rule now to your style sheet.

To transform the text of the first navigation list element:

1. Return to the **sa_styles.css** file in your text editor.
2. Add the following style rule as shown in Figure 3-55:

```
nav ul li:first-of-type {
   text-transform: uppercase;
}
```

TIP

For browsers that don't support the `first-of-type` pseudo-class, you can use the `id` attribute to mark the first list item and write the style based on that id.

Figure 3-55 **Applying a structural pseudo-class**

displays the text of the first li element in the navigation list in uppercase

```
nav ul li:hover {
   background-color: rgb(83, 142, 213);
}

nav ul li:first-of-type {
   text-transform: uppercase;
}
```

3. Save your changes to the file and then reload **home.htm** in your Web browser. Verify that the first entry in the navigation list, Home, is displayed in uppercase letters.

   Trouble? If you are running a version of Internet Explorer earlier than 9, the first list item will not be displayed in uppercase letters.

Pseudo-Elements

Tammy has a few more changes she wants you to make to the Sunny Acres home page. In the first paragraph of the home page, she would like the following styling:

- The first line displayed in all uppercase letters
- The first letter increased in size and displayed as an initial cap

So far, all of your selectors have been based on elements that exist somewhere within the document hierarchy and are tagged in the HTML file. You also can define selectors based on pseudo-elements, which are not elements that exist in the document hierarchy but rather are based on objects that exist in the rendered Web page. For example, a paragraph is an element in the document hierarchy, but the first line of the paragraph is not. It only exists as an object once the paragraph has been rendered by the browser. Similarly, the first letter of that paragraph is also not a document element, but it certainly can be identified as an object in the Web page. Both the first line and the first letter are pseudo-elements, and you can create a style rule to format their appearance. A selector based on a pseudo-element is similar to one that is based on a pseudo-class, such as

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

where *pseudo-element* is an abstract element from the rendered Web page. Figure 3-56 lists some of the pseudo-elements supported by CSS.

Figure 3-56 **Pseudo-elements**

Pseudo-Element	Description	Example
first-letter	The first letter of the element text	p:first-letter {font-size:200%}
first-line	The first line of the element text	p:first-line {text-transform: uppercase}
before	Content inserted directly before the element	p:before {content:"Special! "}
after	Content inserted directly after the element	p:after {content:"eof"}

In order to differentiate between pseudo-elements and pseudo-classes, CSS3 changes the syntax of pseudo-elements by adding an extra colon to the selector:

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

To maintain backward compatibility with older browsers, however, you still should use the single colon syntax.

TIP

Only one pseudo-element may be used per selector.

The style rules to format the first line and first letter of the opening paragraph are:

```
section > p:first-of-type:first-letter {
   font-size: 250%;
   font-family: 'Times New Roman', Times, serif;
}
section > p:first-of-type:first-line {
   text-transform: uppercase;
}
```

Notice that the selector uses the `first-of-type` pseudo-class combined with the `first-letter` and `first-line` pseudo-elements so that the styles are applied only to the first paragraph of the `section` element and then to the first letter and first lines within that paragraph. After you apply the rule, the first letter will appear 250% larger than the surrounding text, and in a Times New Roman or other serif font. In addition, all of the text on the first line will appear in uppercase.

REFERENCE

Working with Pseudo-Elements

- To apply a style to the first line of an element, use the pseudo-element selector

  ```
  selector:first-line
  ```

  where *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 following pseudo-element selector:

  ```
  selector:first-letter
  ```

- To insert a text string before an element, use the style rule

  ```
  selector:before {content: text;}
  ```

  where *text* is the content of the text string.
- To insert a text string after an element, use the following style rule:

  ```
  selector:after {content: text;}
  ```

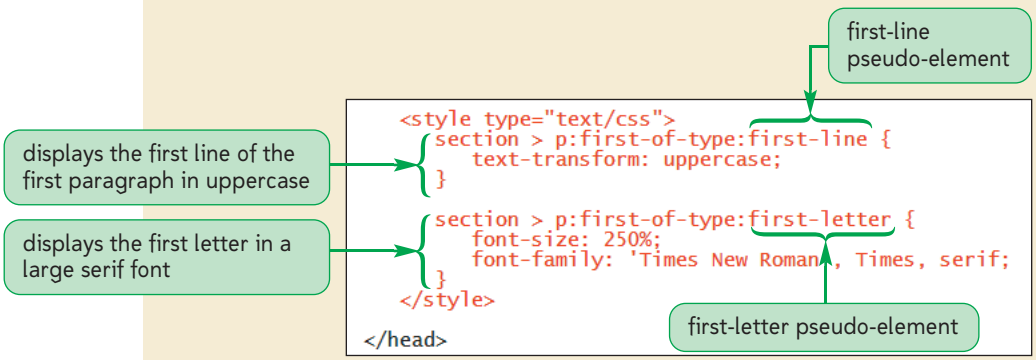
Since Tammy wants to apply this style rule only to the home page and not to every page on her Web site, you'll add the rule to an embedded style sheet placed within the *home.htm* file.

To create the initial cap and first line styles:

1. Go to the **home.htm** file in your text editor.
2. Directly above the closing `</head>` tag, insert the following code, as shown in Figure 3-57:

```
<style type="text/css">
   section > p:first-of-type:first-line {
      text-transform: uppercase;
   }

   section > p:first-of-type:first-letter {
      font-size: 250%;
      font-family: 'Times New Roman', Times, serif;
   }

</style>
```

Figure 3-57 Styling pseudo-elements



3. Save your changes to the file and then reload **home.htm** in your Web browser. As shown in Figure 3-58, the first line of the opening paragraph is displayed in uppercase letters, and the first letter of that line is larger than the surrounding text.

Figure 3-58 First letter and first line styles

Trouble? At the time of this writing, neither Safari nor Chrome is able to apply the `text-transform` style property with the `first-line` pseudo-element.

INSIGHT

Generating Content with CSS

You can generate page content using the following `before` and `after` pseudo-elements along with the `content` property

```
selector:before {content: text;}
selector:after {content: text;}
```

where *text* is the text you want to add to the element. For example, the style rule

```
em:after {content: " !";}
```

appends a space and an exclamation point to the end of every `em` element. You also can use the `before` and `after` pseudo-elements in conjunction with pseudo-classes. The style rules

```
a:hover:before {content: "<";}
a:hover:after {content: ">";}
```

create a rollover effect in which the < and > characters are placed around a hypertext link when a mouse pointer hovers over the link.

Another way to insert content using CSS is to retrieve information from an element attribute using the `attr` property, which has the syntax

```
content: attr(attribute)
```

where *attribute* is an attribute of the element. For example, the following style appends every hypertext link with the text of the link's URL as stored in the `href` attribute:

```
a:after {
   content: attr(" [" attr(href) "] ");
}
```

When applied to the hypertext link

```
<a href="home.htm">Sunny Acres</a>
```

this style will be rendered by the browser as follows:

Sunny Acres [home.htm]

Finally, you can insert an image by specifying the URL of the image file for the value of the `content` property, as the following code demonstrates:

```
a[href^="http"]:after {
   content: url(uparrow.png);
}
```

In this example, the *uparrow.png* file is appended to any hypertext link that contains the text string *http* within its `href` attribute. This technique is sometimes used to visually identify hypertext links that point to external Web sites.

By generating content through your CSS style sheets, you can create interesting dynamic effects for your Web site that are easy to develop and maintain. Note, however, that the `content` pseudo-class is not supported in older browsers, and so you should not rely on it to create text critical to the understanding of your Web site.

Tammy is pleased with your work on the style sheet for the Sunny Acres home page. Your only remaining task is to apply your style sheet to the other pages to confirm that your design choices work properly for every page. Tammy also would like a different background color for the `h1` headings on each page. You'll set the background colors using embedded style sheets in the other pages on the Web site.

To apply the style sheet to the other pages on the site:

1. Go to the **maze.htm** file in your text editor.
2. Directly above the closing `</head>` tag, insert the following `link` elements and embedded style sheet (see Figure 3-59):

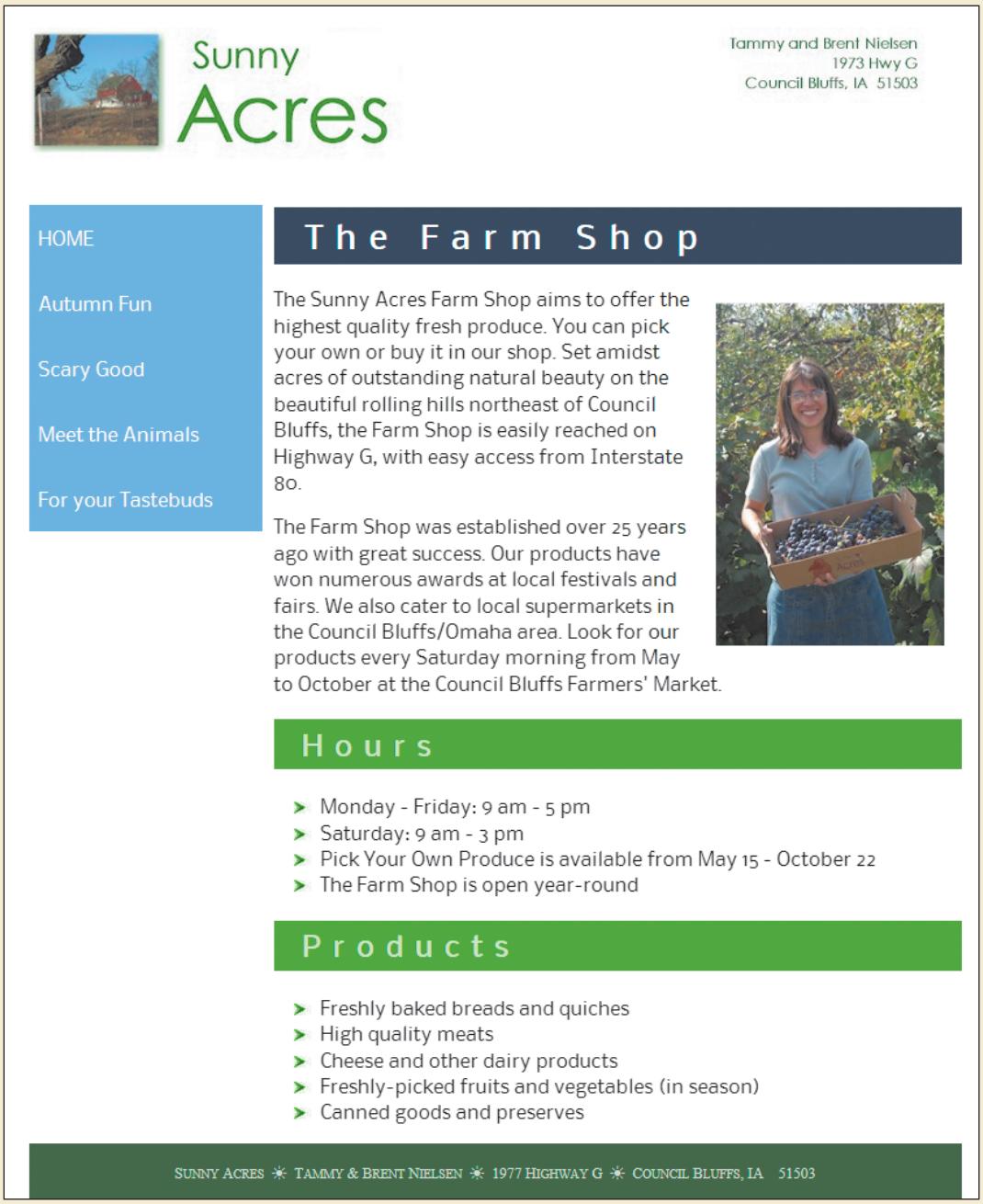
```
<link href="sa_layout.css" rel="stylesheet" type="text/css" />
<link href="sa_styles.css" rel="stylesheet" type="text/css" />
<style type="text/css">
   section h1 {
      background-color: rgb(191, 141, 101);
   }
</style>
```

Figure 3-59 Style sheets for the maze.htm file

```
   <link href="sa_layout.css" rel="stylesheet" type="text/css" />
   <link href="sa_styles.css" rel="stylesheet" type="text/css" />
   <style type="text/css">
      section h1 {
         background-color: rgb(191, 141, 101);
      }
   </style>

</head>
```

3. Save your changes to the file.
4. Repeat Steps 2 and 3 for the **haunted.htm**, **petting.htm**, and **produce.htm** files. Set the `h1` background colors for these three files to **rgb(0, 0, 0)**, **rgb(133, 109, 85)**, and **rgb(50, 69, 99)**, respectively.
5. Reopen the **home.htm** file in your Web browser and navigate through Tammy's Web site. Verify that the layout and color scheme have been applied to every sample page that Tammy has developed. Figure 3-60 shows the completed Web page for the Farm Shop.
6. Submit your completed files to your instructor, in either printed or electronic form, as requested.

Figure 3-60 Farm Shop Web page



PROSKILLS

Teamwork: Managing a Style Sheet

Your style sheets often will be as long and as complex as your Web site content. As the size of a style sheet increases, you might find yourself overwhelmed by multiple style rules and definitions. This can be an especially critical problem in a workplace where several people need to interpret and sometimes edit the same style sheet. Good management skills are as crucial to good Web design as a well-chosen color or typeface. As you create your own style sheets, here are some techniques to help you manage your creations:

- Use style comments throughout, especially at the top of the file. Clearly describe the purpose of the style sheet, where it's used, and who created it and when.
- Because color values are not always immediately obvious, include comments that describe your colors. For example, annotate a color value with a comment such as "body text is tan".
- Divide your style sheet into sections, with comments marking the section headings.
- Choose an organizing scheme and stick with it. You may want to organize style rules by the order in which they appear in your documents, or you may want to insert them alphabetically. Whichever you choose, be consistent and document the organizing scheme in your style comments.
- Keep your style sheets as small as possible, and break them into separate files if necessary. Use one style sheet for layout, another for text design, and perhaps another for color and graphics. Combine the style sheets using the `@import` rule, or combine them within each Web page. Also consider creating one style sheet for basic pages on your Web site, and another for pages that deal with special content. For example, an online store could use one style sheet (or set of sheets) for product information and another for customer information.

By following some of these basic techniques, you'll find your style sheets easier to manage and develop, and it will be easier for your colleagues to collaborate with you to create an eye-catching Web site.

You've completed your work on designing a style sheet to format the text and backgrounds on the Sunny Acres Web site. Tammy will continue to examine your work and get back to you with future projects or design changes.

REVIEW

Session 3.3 Quick Check

1. Provide a style rule to display all ordered lists with lowercase letters as the marker.
2. Provide a style rule to display all unordered lists using the *star.png* image file, placed inside of the containing box.
3. By default, most browsers indent lists from the surrounding text. Provide a style rule to remove the indentation from every unordered list nested within a `section` element.
4. Provide a style rule to display the text of all previously visited hypertext links in gray.
5. Describe the item selected by the following selector:

```
#top > p:first-of-type:first-line
```

6. Describe the items selected by the following selector:

```
div.links img[usemap]
```

7. Provide a style rule to insert the text string "*** " before every paragraph belonging to the *review* class.

Practice the skills you learned in the tutorial using the same case scenario.

PRACTICE

Review Assignments

Data Files needed for the Review Assignments: CloisterBlack.eot, CloisterBlack.svg, CloisterBlack.ttf, CloisterBlack.txt, CloisterBlack.woff, flake.png, holiday.jpg, holidaytxt.htm, hs_layout.css, hs_stylestxt.css, modernizr-1.5.js, salogo.png

Tammy has been working with the Web site you designed. She's returned to you for help with another Web page. The Sunny Acres farm is planning a festival called *Holidays on the Farm* to bring people to Sunny Acres during the months of November and December. The farm is planning to offer sleigh rides, sledding (weather permitting), and visits with Santa Claus. Tammy already has created the content for this page and located a few graphics she wants you to use. She needs you to complete the Web page by creating a style sheet for the text and colors in the page. A preview of the page you'll create for Tammy is shown in Figure 3-61.

Figure 3-61 Holidays on the Farm page

Complete the following:

1. Use your text editor to open the **holidaytxt.htm** and **hs_stylestxt.css** files from the tutorial.03\review folder included with your Data Files. Enter ***your name*** and ***the date*** within the comment section of each file, and then save them as **holiday.htm** and **hs_styles.css**, respectively.
2. Go to the **holiday.htm** file in your text editor and take some time to review the contents and structure of the document. Link the file to the **hs_layout.css** style sheet file. Open the Web page in your browser to study its current layout.

3. Tammy wants to use a Web font for the heading on the page. She has stored the CSS code for the font definition in the **CloisterBlack.txt** file. Open this file, copy the CSS code, and then go to the **hs_styles.css** style sheet file. Paste the copied CSS code into the file below the comment header.
4. Create a style rule for the page body, set the background color to **white**, and set the default font list to **Trebuchet MS**, **Helvetica**, and **sans-serif**.
5. Create a style rule for unordered lists within the `nav` element that: a) sets the background color to the value (**248, 175, 175**); b) sets the line height to **3.5em**; c) removes the bullet marker; d) sets the width of the `padding-left` property to **0** pixels; and e) indents the text **5** pixels. For the first list item, create a style rule to: a) increase the font size to **150%**; b) display the text in small caps; and c) display the text in **bold**. For every hypertext link within the navigation list, create a style that removes the underlining and sets the font color to **white**.
6. When the user hovers the mouse pointer over list items in the navigation list, change the background color to the value (**148, 51, 62**). When the user hovers the mouse pointer over a hypertext link in the navigation list, change the font color to **yellow**.
7. Tammy has placed a promotional photo in a figure box. Set the background color of figure boxes to the color value (**248, 175, 175**) and **center** the contents of figure boxes. For the caption within the figure box, add styles to: a) set the font size to **14** pixels; b) display the text in **italic**; and c) **center** the caption text.
8. For `h1` headings nested within the `section` element, create a style rule to: a) set the background color to the value (**148, 51, 62**); b) set the font color to **white**; c) use the font list containing the **CloisterBlack** and **fantasy** fonts; d) set the font size to **2.5em**; e) remove boldface from the text; f) set the kerning to **0.3em**; g) set the line height to **2em**; and h) indent the text **5** pixels.
9. For `h2` headings nested within the `section` element, create a style rule to: a) set the background color to the value (**182, 134, 52**); b) set the font color to **white** for older browsers and to the color value (**255, 255, 255**) with an opacity of **0.8** for newer browsers; c) set the font weight to **normal**; d) set the kerning to **0.7em**; and e) indent the text **1em**.
10. For unordered lists directly after `h2` headings nested within the `section` element, create a style rule that displays the image file **flake.png** as the bullet marker.
11. For `address` elements nested within the `footer` element, create a style rule to: a) set the background to the color (**148, 51, 62**); b) set the font color to **white** for older browsers and to the value (**255, 255, 255**) with 80% opacity for newer browsers; c) change the font style to **normal** weight in small caps, **0.8em** in size with a line height of **4em**, and use fonts from the list **Palatino Linotype**, **Book Antiqua**, **Palatino**, and **serif**; and d) **center** the address text.
12. Add comments to the style sheet to document what you've done.
13. Save your changes to the **hs_styles.css** style sheet.
14. Return to the **holiday.htm** file in your text editor and link to the **hs_styles.css** file.
15. Open the **holiday.htm** file in your Web browser and verify that the page matches the page shown in Figure 3-61.
16. Submit your completed files to your instructor, in either printed or electronic form, as requested.

Apply your knowledge of CSS text and color styles to format a Web page for a cryptographic institution.

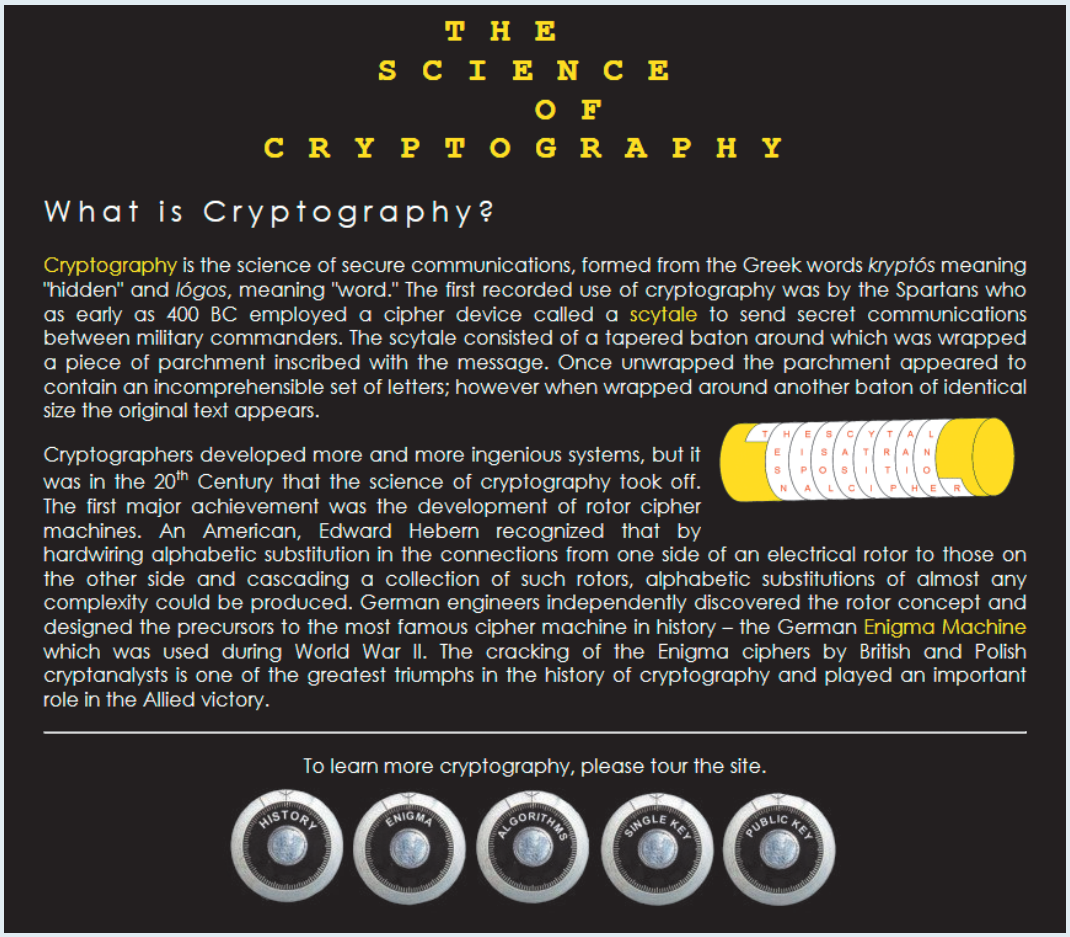
APPLY

Case Problem 1

Data Files needed for this Case Problem: algo.htm, c_layout.css, c_stylestxt.css, crypttxt.htm, enigma.htm, history.htm, locks.jpg, logo.gif, modernizr-1.5.js, public.htm, scytale.gif, single.htm

International Cryptographic Institute Sela Dawes is the media representative for the ICI, the International Cryptographic Institute. The ICI is an organization of cryptographers who study the science and mathematics of secret codes, encrypted messages, and code breaking. Part of the ICI's mission is to inform the public about cryptography and data security. Sela has asked you to work on a Web site containing information about cryptography for use by high school science and math teachers. She wants the design to be visually interesting in order to help draw students into the material. Figure 3-62 shows a preview of your design.

Figure 3-62 **Cryptography home page**



Complete the following:

1. In your text editor, open the **crypttxt.htm** and **c_stylestxt.css** files from the tutorial.03\case1 folder included with your Data Files. Enter ***your name*** and ***the date*** in the comment section of each file. Save the files as **crypt.htm** and **c_styles.css** respectively.
2. Go to the **crypt.htm** file in your text editor, and review the contents and structure of the document. Link the file to the **c_layout.css** style sheet.

3. Locate the three `strong` elements in the two paragraphs and add the `class` attribute to each element with the class value set to **keyword**.
4. Locate the *locks.jpg* inline image, and below the image create an image map with the name **locks**. Add the following hotspots to the image map:
   a. A circular hotspot linked to the **history.htm** file, centered at the coordinates (**52, 52**) with a radius of **43** pixels; the alternate text should be **History**.
   b. A circular hotspot with a radius of **43** pixels located at the coordinates (**155, 52**); link the hotspot to the **enigma.htm** file, and with the alternate text set to **Enigma**.
   c. A circular hotspot with a radius of **43** pixels located at the coordinates (**255, 52**); link the hotspot to the **algo.htm** file and set the alternate text to **Algorithms**.
   d. A circular hotspot with a radius of **43** pixels located at the coordinates (**355, 52**); link the hotspot to the **single.htm** file and set the alternate text to **Single Key**.
   e. A circular hotspot with a radius of **43** pixels located at the coordinates (**455, 52**); link the hotspot to the **public.htm** file and set the alternate text to **Public Key**.
5. Apply the **locks** image map to the *locks.jpg* inline image.
6. Save your changes to the **crypt.htm** file.
7. Go to the **c_styles.css** style sheet file in your text editor. Set the color of the page body background and text to **black** and **white**, respectively. Set the default font to a list consisting of **Century Gothic** followed by a generic **sans-serif** font.
8. Add a style rule for `h1` headings nested within a `header` element to: a) display the text in **yellow**; b) use **Courier New**, **Courier**, or another **monospace** font; c) set the font size to **28** pixels with a kerning of **20** pixels; and d) **center** the text.
9. Add a style rule for `h2` headings nested within an `article` element to: a) set the font size to **24** pixels; b) display the text without boldface; and c) set the kerning to **5** pixels.
10. Align paragraph text within the `article` element using full justification.
11. For `strong` elements belonging to the `keyword` class, create a style rule that displays the text in a **yellow**, non-bold font.
12. Center the contents of paragraphs nested within the `footer` element.

EXPLORE

13. You don't want image maps to appear with a colored border. To remove the border, create a style rule for inline images that contain the `usemap` attribute and set the border width to **0** pixels.
14. Document your work with style comments and then save your changes to the file.
15. Return to the **crypt.htm** file in your text editor and link the file to the **c_styles.css** style sheet.
16. Open **crypt.htm** in your Web browser and confirm that it matches the design shown in Figure 3-62.
17. Submit your completed files to your instructor, in either printed or electronic form, as requested.

Apply your knowledge of CSS to design a Web page for a bike touring company.

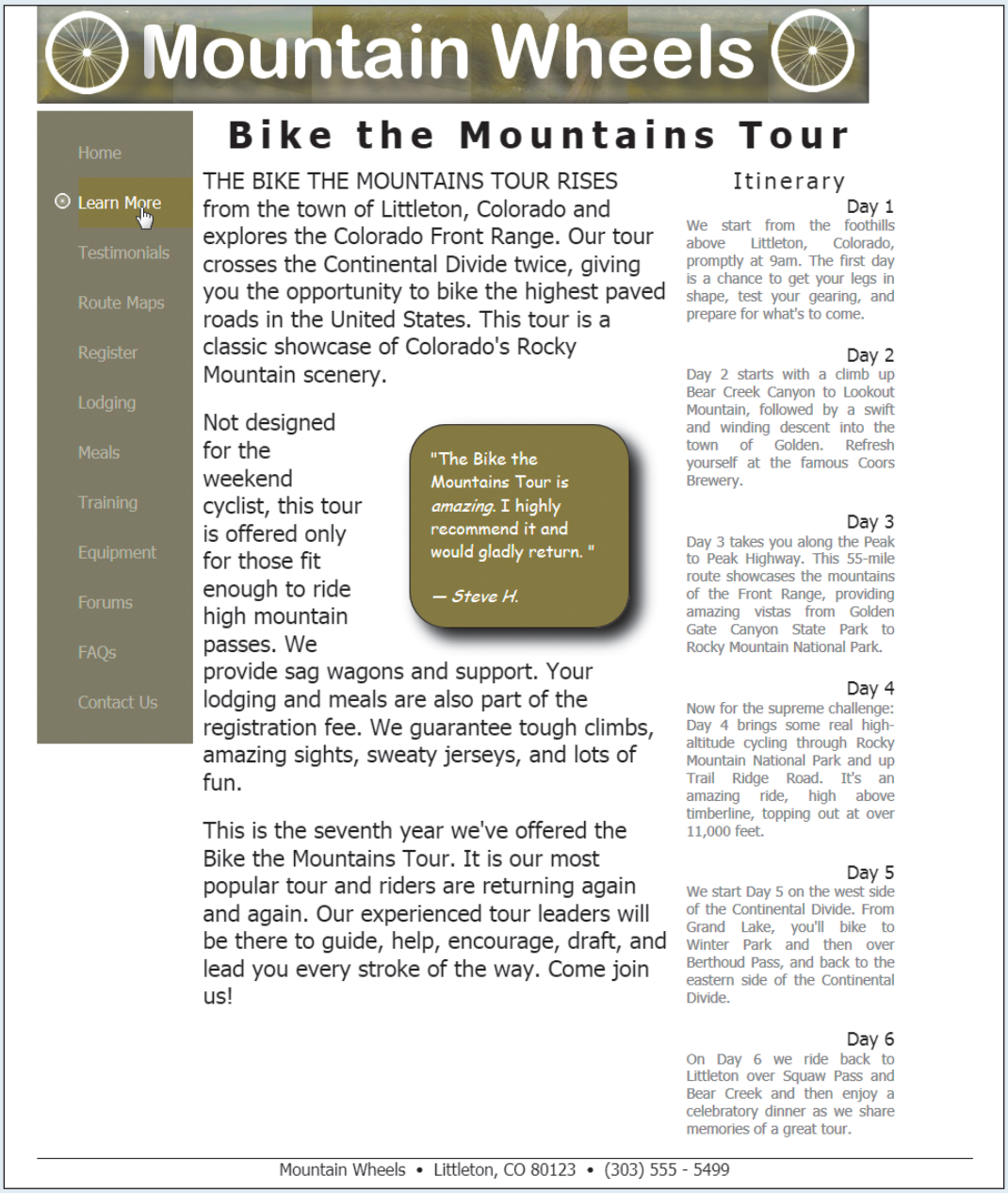
APPLY

Case Problem 2

Data Files needed for this Case Problem: bmtourtxt.htm, modernizr-1.5.js, mw_layout.css, mw_stylestxt.css, mwlogo.png, wheelmarker.png

Mountain Wheels Adriana and Ivan Turchenko are the co-owners of Mountain Wheels, a bike shop and touring agency in Littleton, Colorado. One of their most popular tours is the Bike the Mountains Tour, a six-day excursion over some of the highest roads in Colorado. Adriana wants to update the company's Web site to provide more information about the Bike the Mountains Tour. She already has had a colleague design a three-column layout with a list of links in the first column and descriptive text in the second and third columns. She has asked for your help in completing the design by formatting the text and colors in the page. Figure 3-63 shows a preview of the Web page you'll create.

Figure 3-63 Bike the Mountains page



Complete the following:

1. Open the **bmtourtxt.htm** and **mw_stylestxt.css** files from the tutorial.03\case2 folder. Enter ***your name*** and ***the date*** in the comment section of each file. Save the files as **bmtour.htm** and **mw_styles.css** in the same folder.
2. Return to the **bmtour.htm** file in your text editor. Link the file to the **mw_layout.css** style sheet. Take some time to review the contents and structure of the document.
3. You need to name different elements within the document using the `id` attribute. Add the following ids to the document: a) name the first `header` element **pageheader** and the second `header` element **articleheader**; and b) name the first `section` element **leftsection** and the second `section` element **rightsection**.
4. Save your changes to the file and then go to the **mw_styles.css** style sheet in your text editor.
5. Set the default font for the page body to a font list containing **Tahoma**, **Geneva**, and **sans-serif**.
6. For the `articleheader` id, apply a style rule that: a) sets the font size to **18** pixels and removes boldface; b) sets the kerning to **7** pixels; and c) centers the text.
7. Set the background color of the navigation list to the value (**125, 120, 89**) and set the line height to **3em**. Remove the bullet markers from the navigation list.
8. For hypertext links within the navigation list, create a style rule that: a) sets the font color to **white** for older browsers, and to **white** with 50% opacity for newer browsers; and b) removes the underlining from the link text.

EXPLORE

9. When a user hovers the mouse pointer over list items in the navigation list, change the background color to the value (**131, 121, 36**) and display the image file *wheelmarker.png* as the bullet image. When a user hovers the mouse pointer over a hypertext link in the navigation list, change the font color to **yellow** for older browsers, and to **white** with 100% opacity for newer browsers.
10. For paragraphs that are direct children of the `leftsection` id, set the font size to **22** pixels. Also, for the first paragraph that is also a direct child of the `leftsection` id, display the first line of the paragraph in uppercase.
11. For the `blockquote` element, create a style rule to set: a) the background color to (**131, 121, 36**); b) the font color to **white**; c) the font size to **16** pixels; and d) the font family to **Comic Sans MS**, **Times**, and cursive.

EXPLORE

12. For paragraphs within the `blockquote` element, create styles to insert a **double quotation mark** directly before and after the text of the paragraph.
13. For `h1` headings within the `rightsection` id, create a style rule to: a) set the font size to **22** pixels and the kerning to **3** pixels; b) remove the boldface from the text; and c) **center** the text. For `h2` headings within the `rightsection` id, create a style rule to: a) set the font size to **18** pixels; b) right-align the text; and c) remove the boldface from the text. Finally, for paragraphs within the `rightsection` id, create a style rule to: a) set the font color to **gray**; b) set the font size to **14** pixels; and c) **justify** the text.
14. For `address` elements within the page footer, create a style rule to: a) set the font size to **16** pixels; b) remove the italic style from the address; and c) **center** the text.
15. Add style comments to document your work and then save your changes to the file.
16. Return to the **bmtour.htm** file in your text editor and link that file to the **mw_styles.css** style sheet.
17. Open the **bmtour.htm** file in your Web browser and confirm that it matches the design and layout shown in Figure 3-63.
18. Submit your completed files to your instructor, in either printed or electronic form, as requested.

Explore how to use CSS to design a Web page for an online Civil War History course.

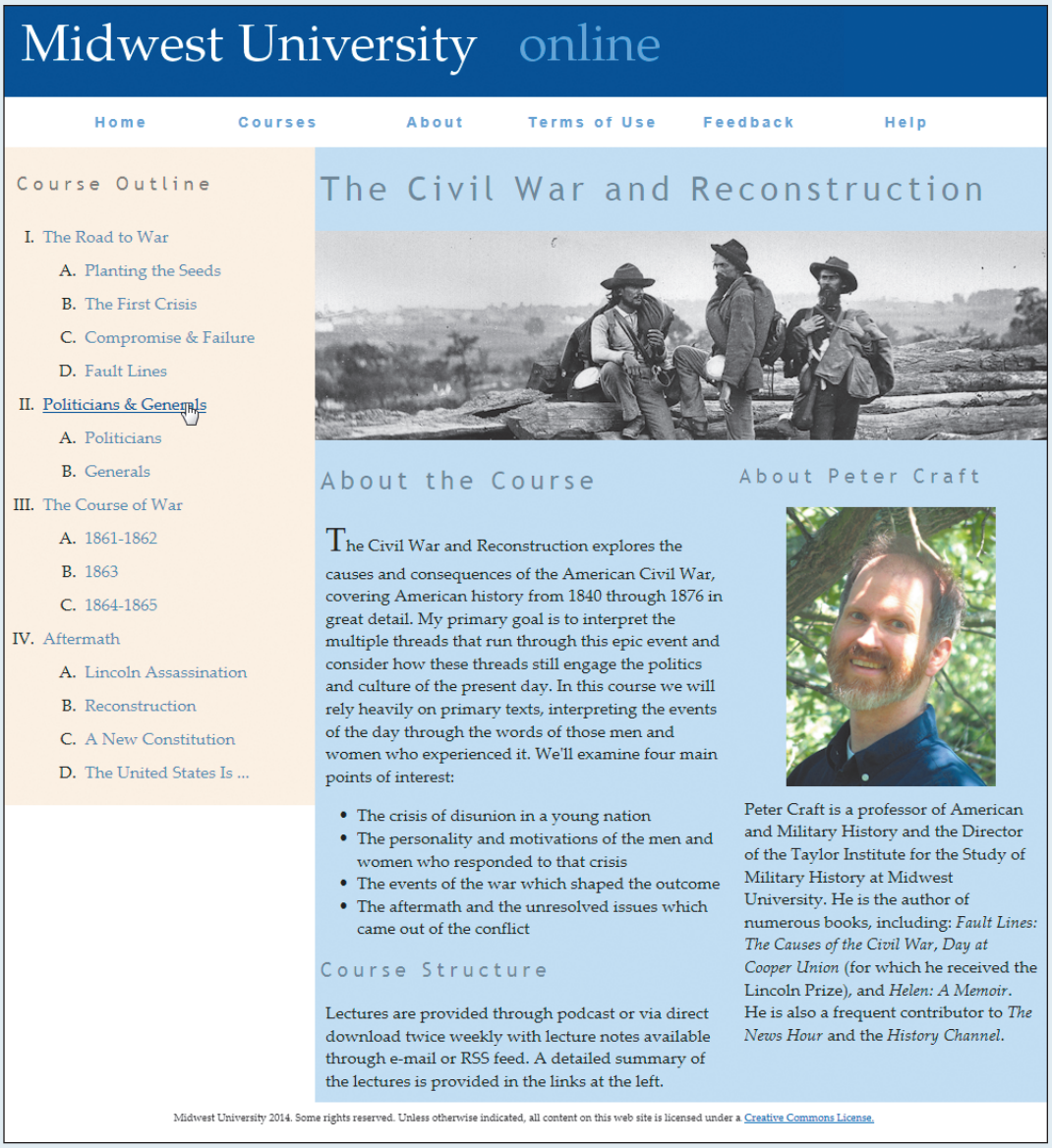
CHALLENGE

Case Problem 3

Data Files needed for this Case Problem: civilwartxt.htm, cw_layout.css, cw_stylestxt.css, cwphoto.png, modernizr-1.5.js, mwulogo.png, pcphoto.png

The Civil War and Reconstruction Peter Craft is a professor of military history at Midwest University. The college is offering a series of online courses, one of which is *The Civil War and Reconstruction* taught by Professor Craft. He has developed the online content and has had a colleague help with the page layout. You've been asked to complete the project by creating text and color styles. A preview of the sample page is shown in Figure 3-64.

Figure 3-64 Civil War and Reconstruction page



Complete the following:

1. Open the **civilwartxt.htm** and **cw_stylestxt.css** files in your text editor. Add ***your name*** and ***the date*** to the comment section of each file, and then save the files as **civilwar.htm** and **cw_styles.css**, respectively.

2. Go to the **civilwar.htm** file in your text editor. Take some time to review the content and structure of the Web page. There are two navigation lists in the document. Peter wants the first navigation list to be displayed horizontally and the second navigation list to be displayed vertically. Add the `class` attribute to each `nav` element, setting the class values to **horizontal** and **vertical** respectively.
3. Save your changes to the file, and then go to the **cw_styles.css** file in your text editor.

EXPLORE

4. Peter already has stored the layout styles in the *cw_layout.css* file. Directly after the opening comments, use the `@import` rule to import this style sheet. Add a comment describing the purpose of the `@import` rule.
5. Set the typeface of the page body to the font list **Palatino Linotype**, **Book Antiqua**, **Palatino**, and **serif**.

EXPLORE

6. For every `h1` through `h6` heading, apply styles to: a) set the color to the HSL value of (**212, 0%, 0%**) with an opacity of **0.4**; b) set the font family to **Trebuchet MS**, **Helvetica**, and **sans-serif**; c) remove the boldface; and d) set the kerning and text indent to **5** pixels.
7. For h1 headings that are direct children of a `header` element that is a direct child of the `body` element, set the background color to the HSL value (**212, 100%, 29%**).
8. For an unordered list within the horizontal navigation list, apply styles to: a) display the text in **Century Gothic MS** or another **sans-serif** font; b) set the font size to **14** pixels; c) display the text in **bold**; d) set the kerning to **3** pixels and the line height to **20** pixels; and e) remove the markers from the list.
9. For hypertext links within the horizontal navigation list, set the text to the HSL value (**212, 100%, 70%**) and remove the underlining. When the mouse hovers over a hypertext link in this list, change the font color to the HSL value (**212, 100%, 29%**).
10. Set the background color of the vertical navigation list to the HSL value (**32, 100%, 95%**).
11. For the `h4` element within the vertical navigation list, create a style rule to: a) set the color to the HSL value (**32, 0%, 0%**) with an opacity of **0.5**; and b) set the font size to **18** pixels and the text indent to **15** pixels.
12. Display the ordered list items in the vertical navigation list in an outline style with uppercase Roman numerals for the top level, and uppercase alphabetic letters for the second level. Set the line height of the lists to **2em**.

EXPLORE

13. Set the color of the hypertext links in the vertical navigation list to the HSL value (**212, 100%, 29%**) with an opacity of **0.6**. Remove the underlining from the hypertext links. If the user hovers the mouse pointer over a link in the list, increase the opacity to **1.0** and display the underline.
14. Set the background color of the `section` element to the HSL value (**212, 95%, 90%**).
15. For the first paragraph after the `h2` heading within the `article` element, create a style rule to display the first letter with a font size of **32** pixels.
16. For the paragraph within the page footer, create a style rule that sets the font size to **10** pixels and the line height to **30** pixels and centers the text.
17. Save your changes to the style sheet.
18. Return to the **civilwar.htm** file in your text editor and link the file to the **cw_styles.css** style sheet.
19. Add style comments to document your work and then save your changes to the style sheet.
20. Open the **civilwar.htm** file in your Web browser. Verify that the style layout and design match that of the Web page shown in Figure 3-64. Verify that when you hover the mouse pointer over the links in the horizontal navigation list, the text changes color. Verify that when you hover the mouse pointer over the links in the vertical navigation list, the text color changes and the text is underlined.
21. Submit your completed project to your instructor, in either printed or electronic form, as requested.

Test your knowledge of CSS to design text and color styles for a children's choir Web site.

APPLY

Case Problem 4

Data Files needed for this Case Problem: chen.png, choirtxt.htm, gcc_layout.css, gcc_stylestxt.css, gcclogo.png, modernizr-1.5.js, nobile.txt, nobile-webfont.eot, nobile-webfont.svg, nobile-webfont.ttf, nobile-webfont.woff

Gresham Children's Choir Faye Dawson is the program director for Gresham Children's Choir in Mentor, Ohio. The choir offers a chance for talented youth to perform and to learn about music history. Faye is working on redesigning the choir's Web site and has asked for your help. She already has created a sample Web page for you to work on and has developed a page layout. She wants you to complete the design by creating styles for the text and colors in the page.

Complete the following:

1. Use your text editor to open the **choirtxt.htm** and **gcc_stylestxt.css** files from the tutorial.03\case4 folder. Add ***your name*** and ***the date*** in the comment section of each file. Save the files as **choir.htm** and **gcc_styles.css**, respectively.
2. Go to the **choir.htm** file in your text editor and link the file to the **gcc_layout.css** file. Take some time to study the content and structure of the document. Save your changes to the file and then view the current layout in your Web browser.
3. Go to the **gcc_styles.css** file and create a style sheet with style rules that:
   - modify the typeface, font weight, font size, kerning, and line height
   - employ a Web font
   - modify the appearance of list items
   - change the text and background color, including at least one example of a semi-transparent color
   - employ contextual selectors, pseudo-elements, and pseudo-classes
   - create rollover effects
   - employ progressive enhancement

   Document all of your work with informative style comments.
4. Add an embedded style sheet to the **choir.htm** file to apply a style rule of your own choosing to that page only.
5. Test your Web site in a variety of browsers to ensure your design works under different conditions.
6. Submit your completed files to your instructor, in either printed or electronic form, as requested.

ENDING DATA FILES