




Applying Advanced CSS Styling

Files You Will Need:

To view a list of files needed for this unit, see the Data Files Grid in the back of the book.


Recent versions of CSS have increased the aspects of Web pages that CSS can style. While using the newest CSS features requires careful planning, many of these effects can significantly decrease the time you spend developing a Web site.  In this unit, you'll implement new CSS styles to enhance the layout of the Lakeland Reeds B&B Web site.

OBJECTIVES

- Assess advanced CSS styles
- Implement pseudo-elements
- Add generated content
- Integrate opacity
- Create rounded corners
- Create text shadows
- Add box shadows
- Test browser capabilities with Modernizr



Assessing Advanced CSS Styles

In addition to basic effects like text formatting and element positioning, CSS can style many different aspects of your Web pages. Many of the CSS properties proposed as part of the development of CSS3 enable browsers to create effects that until recently were only available by integrating images. As with the newest HTML features, the specifications from the W3C are only one aspect of using the most recently developed CSS properties. In fact one of the most important factors in deciding what code to include in a Web site is considering which features are supported by different browsers and how they are supported.  You've collaborated with the design team and planned a few tweaks to the design of the Lakeland Reeds B&B Web site. As you prepare to incorporate the new properties, you review a few guidelines for working with more recently developed CSS properties.

DETAILS

Best practices for using newer features consistently and predictably include

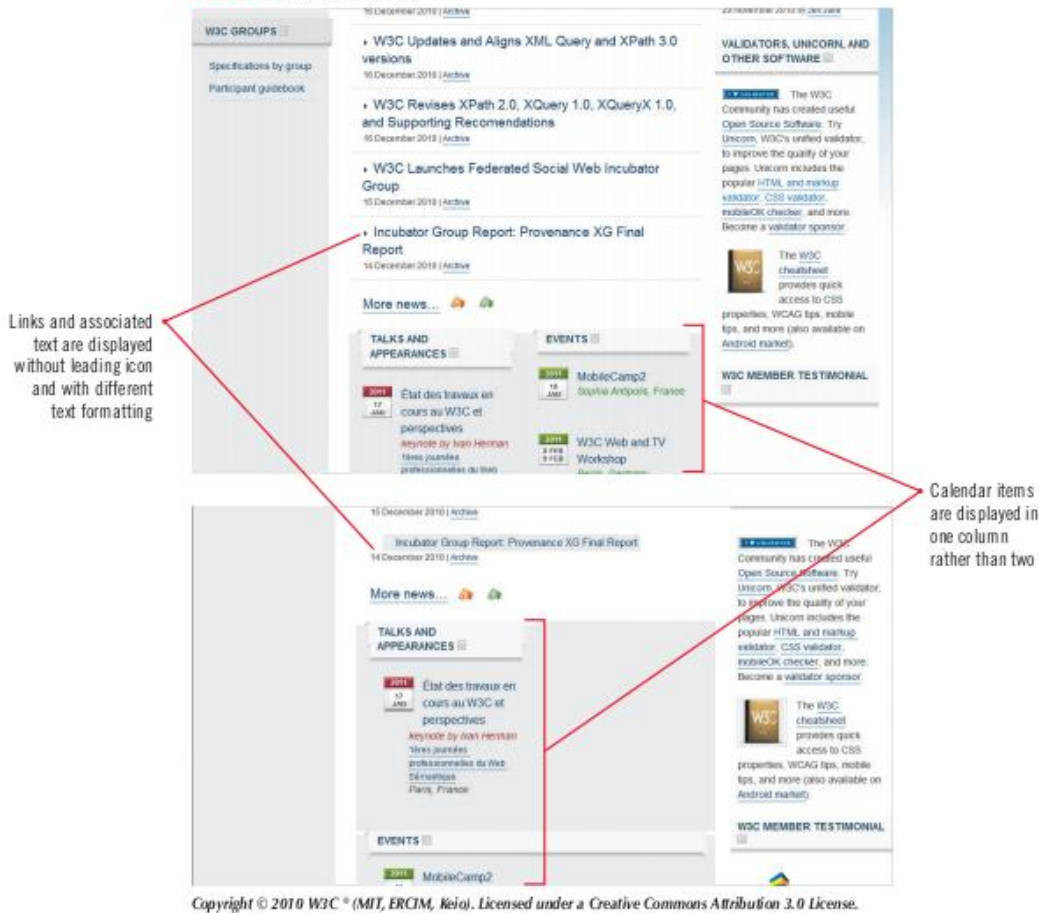
- **Progressive enhancement and graceful degradation**

Different browsers handle a given CSS property differently depending on a number of factors. An older browser generally won't support features developed after the browser was released. In addition, the specifications for some CSS features are modified over time, and older browsers support only earlier versions of CSS features. Even though a few main browsers are used for most Web viewing, many versions of each are still in use. Web developers use two main practices to plan Web sites that are usable across this spectrum of capabilities. The main content and capabilities of a Web site should be available for users of the least-featured browsers employed by a significant share of your users. Rather than making advanced features crucial to the layout of a Web site, you add additional features as enhancements only for browsers that can render them, a practice known as **progressive enhancement**. A complementary practice, **graceful degradation**, involves ensuring that the appearance and usability of a Web site doesn't depend on any advanced features; the site should degrade gracefully, meaning that when viewed in browsers that don't support some features, Web page elements that use those features should nevertheless be displayed in a usable way. Figure I-1 shows the same Web page code in browsers with different levels of feature support. You've already implemented both of these concepts in your Web development work for more widely supported features; for instance, when specifying an image as a Web page background, you also specified a color as a fallback. However, these practices are especially important when working with new features that aren't necessarily widely supported.

- **Shims and patches**

In addition to building backward-compatible sites by specifying alternative code, another useful tool is a script written specifically to bridge the gap between browsers with reduced feature sets and more fully featured browsers. Many of these scripts, known as **shims** or **patches**, are available for use by anyone free of charge. Some address only a single issue; the script that makes HTML5 semantic elements available in recent versions of Internet Explorer, which you've included in your Web pages in previous units, is one such shim. Other tools package many shims together into a library of scripts; later in this unit, you'll implement one such tool, known as Modernizr.

FIGURE I-1: Graceful degradation




Working with browser-specific styles

Sometimes the creators of browser rendering engines include support for CSS properties whose functions and syntax haven't yet reached consensus in the W3C development process. To support these properties while ensuring that the browsers won't misinterpret future standards, a few groups incorporate browser-specific prefixes into the property names that they implement. For instance, early implementations of the border-radius property used the property name `-webkit-border-radius` for browsers that use the WebKit rendering engine (including Chrome and Safari) and `-moz-border-radius` for browsers using the Mozilla rendering engine (including Firefox). Some properties also have variants with an `-o-` prefix for the Opera browser.

When you specify code for one of these properties, you should always specify name-value pairs for each of the browser-specific variants first and conclude the code for the style with the code for the generic style; in this example, you'd finish with the border-radius property. Because a later name-value pair in a style rule supersedes an earlier one that duplicates it, newer browsers that support the final specification for a property and its generic name-value pair will use the final line of code, while older browsers will use the relevant browser-specific code and ignore the generic property name.

Implementing Pseudo-elements

Beginning in version 2 of the language, CSS has included **pseudo-elements**, which are selectors that enable you to isolate a piece of a larger element for styling. Like a pseudo-class, a pseudo-element is preceded by a colon (:). In recent browsers, you can precede a pseudo-element with two colons (::) to differentiate it visually from a pseudo-class; however, because older versions of Internet Explorer don't recognize this format, it's best to use double colons only for CSS3 pseudo-elements, which these browsers don't recognize anyway. Table I-1 describes five commonly used CSS pseudo-elements.  You explore how the first-letter and first-line pseudo-elements affect the presentation of text on the home page.

STEPS

1. In your text editor, open **HTM I-1.html** from the **Unit I/Unit** folder where you store your Data Files, insert a blank line before the closing body tag, insert a paragraph element containing your first and last name and the text **HTML5 Unit I**, save it as **index.html**, repeat to save **HTM I-2.html** as **aboutus.html**, **HTM I-3.html** as **rooms.html**, and **HTM I-4.html** as **reserve.html**, then use a CSS comment to add the same information to **HTM I-5.css** and save it as **lakeland.css**
2. Return to **aboutus.html** in your text editor and examine the code
In addition to the conditional comment you added to include a script in an earlier unit, another comment adds an embedded style sheet for certain browsers. In the code [if lt IE 7], *lt* stands for *less than*. The code specifies that installations of Internet Explorer with version numbers less than 7 should use the embedded styles; all other browsers ignore it. The styles correct for rendering errors in IE6 and earlier versions.
3. Open **index.html** in your browser
The first two words of the main paragraph of text are displayed in bold as a result of styles applied to a span element.
4. Return to **lakeland.css** in your text editor, create a style rule with the **#maintext:first-line selector**, add a name-value pair to style text in **bold**, save your work, then reload **index.html** in your browser
As Figure I-2 shows, the entire first line is displayed in bold.
5. Return to **lakeland.css** in your text editor, delete the **#maintext:first-line** style rule, create a style rule with the **#maintext:first-letter selector**, then add name-value pairs to set font size to **3em**, float the element on the **left**, and set **line-height** to **0.8em**
The **:first-letter** selector allows you to create a **drop cap**, a common visual effect in print media in which the first letter of a paragraph or section is enlarged and drops below the first line of text. You use the float property to make the text that follows the enlarged letter flow around it. Setting the line height for a drop cap is commonly necessary to integrate the letter optimally with the remaining paragraph text.
6. Save your work, then reload **index.html** in your browser
As Figure I-3 shows, the first letter of Lakeland Reeds is enlarged, and multiple lines of text run to its right.

QUICK TIP

Selecting the most appropriate line-height value for a drop cap often requires trial and error.

FIGURE I-2: :first-line pseudo-element applied to paragraph



Unholy Vault Designs/Shutterstock.com
 Faith Wempen/sycamoreknoll.com
 Photo/Sasha Vodnik

FIGURE I-3: :first-letter pseudo-element applied to paragraph




Unholy Vault Designs/Shutterstock.com
 Faith Wempen/sycamoreknoll.com
 Photo/Sasha Vodnik

TABLE I-1: CSS pseudo-elements

pseudo-element	effect	CSS version
:first-line	styles the first line of text in the current element	2.1
:first-letter	styles the first letter of text in the current element	2.1
:before	inserts specified content before the current element	2.1
:after	inserts specified content after the current element	2.1
:selection	styles Web page content selected by user	3

Adding Generated Content

Unlike the other pseudo-elements, which simply select parts of existing Web page elements, the `:before` and `:after` pseudo-elements enable you to insert content into Web pages using style rules. You can use these selectors to add repeating text or icons to all elements of a given class, for instance, or to specify beginning or ending indicators for each page in a Web site.  Each description on the Rooms page includes a section listing the number and size of beds. You use the `:before` pseudo-element to add an icon before the bed details for each room.

STEPS

1. Open `rooms.html` in your browser, then scroll down to view the room descriptions
The code for each bed detail paragraph is part of the `beds` class.
2. Return to `lakeland.css` in your text editor
3. Below the `.beds` style rule, create a style rule based on the `.beds:before` selector
This selector creates and selects an element before each element in the class `beds`.
4. Within the style rule you created, add the following name-value pair:
`content: url("images/bedicon.png");`
Figure I-4 shows the completed rule. Style rules based on the `:before` or `:after` pseudo-elements must specify a value for the `content` property, which can be either text or the path and name for an image file. Table I-2 shows the syntax for the most common values of the `content` property.
5. Save your work, reload `rooms.html` in your browser, then scroll down to view the room descriptions
As Figure I-5 shows, an icon depicting a bed is displayed at the start of each paragraph that describes beds.

FIGURE I-4: Style rule using the :before pseudo-element

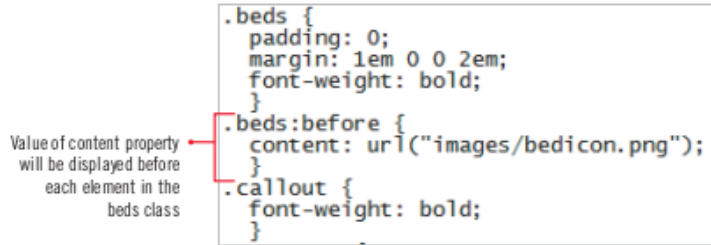


FIGURE I-5: :before pseudo-element applied to beds class

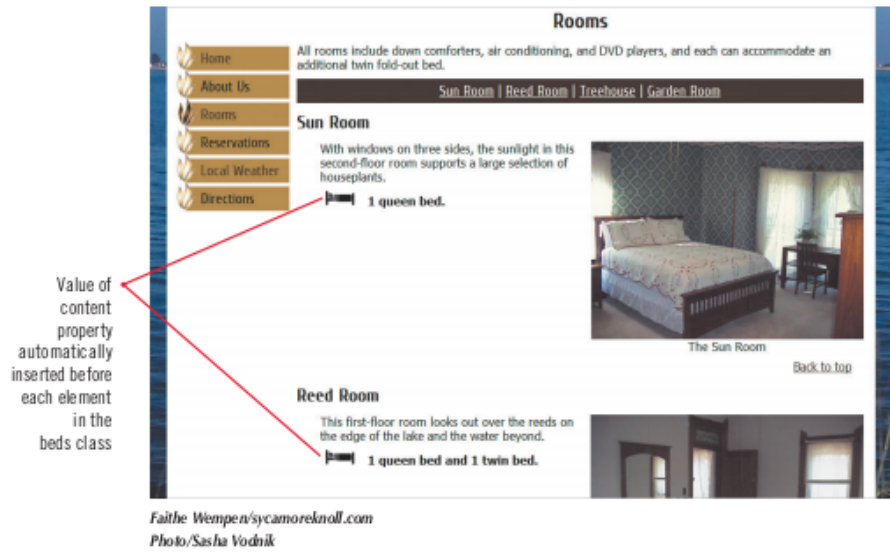



TABLE I-2: Common values for the content property

content type	syntax	example
text	"text"	content: "Chapter: ";
image	url("path/image")	content: url("images/bed.png");

Integrating Opacity

CSS enables you to specify the color numerically for a Web page element using the hexadecimal, rgb (red green blue), and hsl (hue saturation light) systems. While all three systems provide you access to a similar range of colors, CSS3 made both rgb and hsl more flexible by adding an alpha channel. The resulting systems are known as rgba and hsla; in each system, the earlier triplet of values becomes a set of four, with the final value representing the level of opacity as a decimal value from 0 (fully transparent) to 1 (totally opaque). Because many browsers in current use don't support the rgba property, it's important to use it along with a corresponding rgb name-value pair as a backup. Older browsers do not support either the hsl or hsla properties, so adding an rgb triplet is a useful tool for graceful degradation for these properties as well.  On the main page and the rooms page, you want to position the figure captions over the corresponding images with a background to make them more noticeable. You specify a semi-transparent white background with a solid white background as a backup.

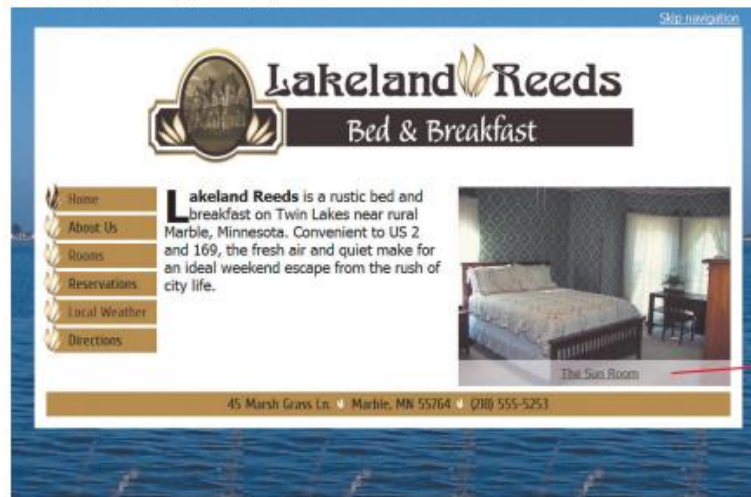
STEPS

1. Return to [lakeland.css](#) in your text editor
2. In the [figcaption](#) style rule, add name-value pairs to set the [position](#) to [absolute](#), [left](#) and [bottom](#) to [0](#), [padding](#) to [0.5em 0](#), [margin](#) to [0](#), and [width](#) to [100%](#)
3. Add a name-value pair that sets the [background](#) property to [rgb\(255, 255, 255\)](#)
This code provides a backup rgb value for browsers that don't support rgba, allowing your Web pages to degrade gracefully.
4. Beneath the name-value pair you inserted in Step 3, add a name-value pair that sets the [background](#) property to [rgba\(255, 255, 255, 0.6\)](#)
The value for this property includes an alpha value that sets the background to 60% opaque. Figure I-6 shows the completed code for the [figcaption](#) style rule.
5. Save your work, then reload [index.html](#) in your browser
As Figure I-7 shows, the caption is overlaid at the bottom of the image, with a semi-transparent background in newer browsers and a solid white background in browsers that don't support rgba.
6. Reload [rooms.html](#) in your browser, then scroll down to verify that the captions are overlaid on the room images with semi-transparent backgrounds

FIGURE I-6: Code to create translucent figure captions

```
figcaption {
  display: block;
  position: absolute;
  bottom: 0;
  left: 0;
  padding: 0.5em 0;
  margin: 0;
  width: 100%;
  background: rgb(255, 255, 255);
  background: rgba(255, 255, 255, 0.6);
}
```


FIGURE I-7: rgba value applied to figcaption element



Alpha value of 0.6 makes white background transparent

Unholy Vault Designs/Shutterslock.com
Faithe Wempen/sycamoreknoll.com
Photo/Sasha Vodnik

Creating Rounded Corners

CSS3 enables a number of functions using style rules that previously required a good deal more work to implement. One of these, transforming the corners of an element from a square to a rounded appearance, required creating and positioning image files simulating a rounded edge in one or more of the corners of an element. To create this effect in recent browsers, you instead use the CSS3 `border-radius` property along with `-moz` and `-webkit` styles for versions of some browsers created during the development of this property. Values for this property are expressed in pixels, with 0 creating a standard square border and larger numbers increasing the curve of the rounded area. The `border-radius` property sets values for all four corners of an element at once, while you can use specific properties for individual corners. Table I-3 lists properties for specific corners for different rendering engines.  The layout you created with the design team includes rounded corners on the right corners of each button, the bottom corners of the footer, and all four corners of the container element for the page content. You add these rounded corners using CSS.

STEPS

QUICK TIP

You can specify two values for any `border-radius` property to create more oblong corners; the first value is the horizontal radius and the second is the vertical radius.

1. Return to `lakeland.css` in your text editor, then after the last name-value pair in the `#mainnav li` style rule, add the six lines of code shown in Figure I-8

The first two name-value pairs set the upper-right and lower-right radii using WebKit-specific properties. The next two lines repeat these settings using Mozilla-specific code. The final two lines again repeat the settings using generic properties; any browser that supports this standard code will ignore the previous browser-specific settings.

2. Save your work, then reload `index.html` in your browser

As Figure I-8 shows, the upper- and lower-right corners of each button are rounded in most browsers. In this case, the original square corners that are displayed in older browsers don't affect the layout, so the feature degrades gracefully without any additional code.

3. Return to `lakeland.css` in your text editor, then after the last name-value pair in the `footer p` style rule, add the six lines of code shown in Figure I-9

Similar to the code for the nav bar buttons, these properties set radii for the lower-left and lower-right corners of the footer using the WebKit, Mozilla, and generic properties.

4. In the `#box` style rule, add the following code after the final name-value pair:

```
-webkit-border-radius: 10px;
-moz-border-radius: 10px;
border-radius: 10px;
```

These name-value pairs set the same border radius for all four corners of the `#box` element using the WebKit, Mozilla, and generic properties.

5. Save your work, then reload `index.html` in your browser

As Figure I-9 shows, the two lower corners of the footer, as well as all four corners of the white box containing the page content, are rounded.

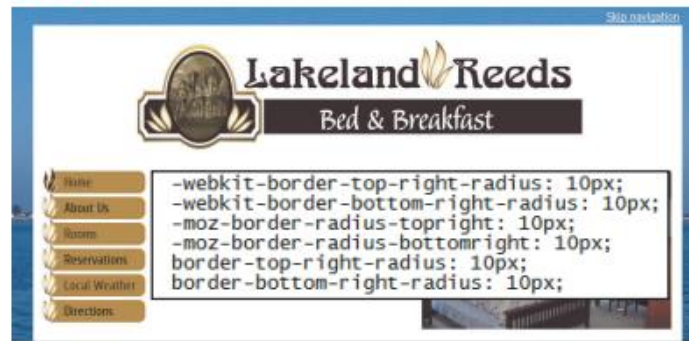
6. Use the nav bar links to open the other pages of the Web site

Because you applied these properties to existing elements and ids, the new styles are applied to all the pages on your site.

QUICK TIP

You can also use the `border-radius` property as a shorthand property, specifying 2, 3, or 4 values to set different border radii for different corners with a single name-value pair.

FIGURE I-8: Code for rounded corners and resulting nav bar appearance



Unholy Vault Designs/Shutterstock.com
 Faith Wempen/sycamoreknoll.com
 Photo/Sasha Vodnik

FIGURE I-9: Code for rounded corners and resulting footer and box appearance




Unholy Vault Designs/Shutterstock.com
 Faith Wempen/sycamoreknoll.com
 Photo/Sasha Vodnik

Rounded corners applied to lower-left and lower-right corners of footer, and to all four corners of the box element

TABLE I-3: Browser-specific border radius properties

rendering engine	corner-specific properties	apply to
WebKit	-webkit-border-top-left-radius -webkit-border-top-right-radius -webkit-border-bottom-left-radius -webkit-border-bottom-right-radius	Safari 4 and earlier
Mozilla	-moz-border-radius-topleft -moz-border-radius-topright -moz-border-radius-bottomleft -moz-border-radius-bottomright	Firefox 3.6 and earlier
generic	border-top-left-radius border-top-right-radius border-bottom-left-radius border-bottom-right-radius	Chrome Firefox 4 and later Safari 5 and later Internet Explorer 9 and later Opera 10.5 and later

Creating Text Shadows

Another effect used in some print layouts is a text shadow, which creates the appearance of a shadow on a surface behind each letter. Traditionally, text shadows have been brought to Web pages by creating the effects in image manipulation software and then linking to a graphic showing the text in the HTML code. CSS3 enables you to add shadows to text using the text-shadow property. The property takes four values: horizontal offset, vertical offset, blur, and shadow color. Table I-4 and Figure I-10 detail how each property affects the appearance of a text shadow.  You enhance the rollover effect of the nav bar text by adding a text shadow to the a:hover rule for main nav bar text.

STEPS

QUICK TIP

Because you have other hover effects in place for nav bar text, this feature already degrades gracefully in your Web pages.

1. Return to `lakeland.css` in your text editor
2. At the end of the `#mainnav a:hover` style rule, add the code `text-shadow: 1px 1px 1px black;`
This code creates a text shadow with a horizontal offset, vertical offset, and blur of 1px, and a shadow color of black.
3. Save your work, then reload `index.html` in your browser
The appearance of the nav bar text is unchanged.
4. Move the mouse pointer over the links in the nav bar
As Figure I-11 shows, most browsers add a shadow to nav bar text when the mouse pointer is over it.

FIGURE I-10: Values for the text-shadow property

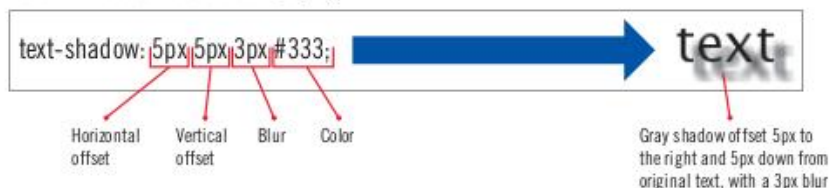



FIGURE I-11: Text shadow on a Web page



TABLE I-4: Values for the text-shadow property

value	affects	notes
horizontal offset	location of shadow horizontally behind text	Required value; may be negative; must be first number in list
vertical offset	location of shadow vertically behind text	Required value; may be negative; must be second number in list
blur	width and lightness of shadow	Optional value; must be positive; must be third number in list
color	color of shadow behind text	Optional value; may appear before or after numerical settings

Adding Box Shadows

Design in print media doesn't limit shadows to text; shadows are also used on entire units of page content. The CSS3 box-shadow property allows you to apply a shadow to many Web page elements as well. The box-shadow property uses the same values as the text-shadow property: horizontal offset, vertical offset, blur, and shadow color. Unlike text-shadow, however, box-shadow also has -webkit and -moz variants.  You add a box shadow to each link button in the nav bar.

STEPS

1. Return to [lakeland.css](#) in your text editor
2. In the `#mainnav li` style rule, add the following code at the bottom:

```
-webkit-box-shadow: 1px 1px 4px black;  
-moz-box-shadow: 1px 1px 4px black;  
box-shadow: 1px 1px 4px black;
```
3. Save your changes, then reload [index.html](#) in your browser
As Figure I-12 shows, browsers that support box shadows display a shadow around the white box containing each nav bar link.
4. Return to [lakeland.css](#) in your text editor, then change each 1px value in the code you just entered to 2px
Increasing the horizontal and vertical offsets moves the shadow farther away from the main box. Figure I-13 shows the completed code.
5. Save your changes, then reload [index.html](#) in your browser
Figure I-14 shows the changes in the appearance of the box shadows.

FIGURE I-12: Box shadows applied to nav bar buttons




FIGURE I-13: Code for bigger box shadow

```
-webkit-box-shadow: 2px 2px 4px black;  
-moz-box-shadow: 2px 2px 4px black;  
box-shadow: 2px 2px 4px black;
```

FIGURE I-14: Box shadows with increased horizontal and vertical offsets



Testing Browser Capabilities with Modernizr

Some styles can neatly fall back to other CSS properties, such as an `rgb` color value substituting when an `rgba` value isn't recognized. Others, however, require you to provide an entirely different set of styles for graceful degradation. One popular tool for enabling such alternative code is Modernizr, a free script library created by Faruk Ateş and Paul Irish. Once Modernizr is linked to a Web page, it tests each user's browser to detect which properties are supported. Based on the results, Modernizr generates a set of CSS classes. You can write alternative code for different browsers and use these classes to ensure that only the code relevant to a specific browser is rendered.  For browsers that don't support any box-shadow properties, you want to simulate a box shadow by creating borders in shades of gray on two sides of an element. You link Modernizr to the main Web page and then specify the alternative code.

STEPS

QUICK TIP

The Modernizr library includes the script you've been using to make HTML5 elements available for styling in Internet Explorer.

QUICK TIP

The class names that correspond to each feature are listed in the online documentation for Modernizr.

1. Return to `aboutus.html` in your text editor, delete the three-line comment just before the closing `</head>` tag, insert a blank line in the same location, then indent and enter the following element:

```
<script src="scripts/modernizr-1.6.min.js"></script>
```

This script element links to an external script file containing the Modernizr library.

2. In the opening `<html>` tag, add the attribute-value pair `class="no-js"`
This class is required for Modernizr to work. Figure I-15 shows the completed HTML code.
3. Return to `lakeland.css` in your text editor, create a new style rule based on the `.boxshadow #mainnav li` selector, then use the cut-and-paste functions of your text editor to remove the three box shadow name-value pairs from the `#mainnav li` style rule and insert them into the new rule you created

For each feature for which Modernizr finds support in a user's browser, it adds a class value to the html element. For the box-shadow properties, the class name created after a successful test is `.boxshadow`. If this class exists, the browser supports the style, and this rule is implemented; if box shadow support isn't present, the class isn't created, and the browser ignores this style rule.

4. At the bottom of the `#mainnav li` style rule, add the following code:
`border-bottom: 2px solid #666;`
`border-right: 2px solid #777;`
This code adds bottom and right borders in different shades of gray to the nav bar buttons.
5. Above the first name-value pair in the `.boxshadow #mainnav li` style rule, add the code `border: none`

This code removes the gray borders only in browsers that can create the CSS box shadow effect.

6. Save your work, then reload `aboutus.html` in your browser
Figure I-16 shows the shadow effect in a browser that doesn't support CSS box shadows.
7. Convert `lakeland.css` to a print style sheet with the name `lprint.css`, validate the code for your Web pages and your style sheets, then make changes as necessary to fix any errors
8. If you have space on a Web server, publish your Web site, then open your Web pages in your browser from the published location

TROUBLE

Because they are not part of official CSS specifications, all browser-specific properties trigger validation errors, and you may also receive an erroneous error for your `rgba` value. You can safely ignore all these errors.

FIGURE I-15: Web page code incorporating Modernizr script

```
<!DOCTYPE html>
<html class="no-js">
  <head>
    <meta charset="utf-8" />
    <title>Lakeland Reeds Bed & Breakfast - About Us</title>
    <link rel="stylesheet" type="text/css" media="screen" href="lakeland.css" />
    <link rel="stylesheet" type="text/css" media="print" href="llprint.css" />
    <!--[if lt IE 7]>
      <style type="text/css">
        #mainnav {left: 0px;}
        li img {left: -42px;}
      </style>
    <![endif]>
    <script src="scripts/modernizr-1.6.min.js"></script>
  </head>
```

Previous script element replaced with reference to external file containing Modernizr script

Class added to apply results of Modernizr scripts to Web page

FIGURE I-16: Alternative box shadow code in browser without box-shadow property support



Unholy Vault Designs/Shutterstock.com

Jason Bucy

Photo/Sasha Vodnik

Modernizr applies shadow-like gray borders only in browsers that don't support the box-shadow property

Browser sniffing vs. feature detection

Ensuring cross-browser compatibility has been an important task in Web development since early in the Web's existence. For much of that time, Web developers tried to identify the brand and version of each user's browser by using a script to ask the browser to identify itself, a technique known as **browser sniffing**. While you may encounter browser sniffing scripts in your work on existing Web sites, the technique has an important drawback; browsers may not

always identify themselves accurately. Modernizr instead approaches the problem by running a series of tests and using the results to decide which features a user's browser supports, a process known as **feature detection**. Feature detection is more reliable than browser sniffing and also provides a more detailed picture of the capabilities of a user's browser.

Practice

Concepts Review



For current SAM information, including versions and content details, visit SAM Central (<http://www.cengage.com/samcentral>). If you have a SAM user profile, you may have access to hands-on instruction, practice, and assessment of the skills covered in this unit. Since various versions of SAM are supported throughout the life of this text, check with your instructor for the correct instructions and URL/Web site for accessing assignments.

Refer to Figure I-17.

FIGURE I-17



1. Which feature is created with an `rgba` color value?
2. Which feature is created with the `text-shadow` property?
3. Which feature is created with the `box-shadow` property?
4. Which feature is created with the `:first-letter` pseudo-element?
5. Which feature is created with the `border-radius` property?

Match each term with the statement that best describes it.

- | | |
|----------------------------|--|
| 6. progressive enhancement | a. a script written specifically to bridge the gap between browsers with reduced feature sets and more fully featured browsers |
| 7. graceful degradation | b. a selector that enables you to isolate a piece of a larger element for styling |
| 8. shim | c. adding additional features to a Web page as enhancements only for browsers that can render them |
| 9. pseudo-element | d. a common visual effect in print media in which the first letter of a paragraph or section is enlarged |
| 10. drop cap | e. ensuring that the appearance and usability of a Web site doesn't depend on any advanced features |

Select the best answer from the list of choices.

11. Specifying a fallback color for a background image is an example of _____.
 - a. a shim
 - b. a pseudo-element
 - c. a patch
 - d. graceful degradation
12. Which is an example of a shim or patch?
 - a. Internet Explorer
 - b. Modernizr
 - c. CSS
 - d. the meta element
13. If you were using the `-moz-border-radius` property in a style rule, which additional property should you include as the last property in the code for that feature?
 - a. `-webkit-border-radius`
 - b. `-o-border-radius`
 - c. `border-radius`
 - d. `background`
14. To support users of older versions of Internet Explorer, you should precede non-CSS3 pseudo-elements with _____.
 - a. `:`
 - b. `::`
 - c. `-ie`
 - d. a space
15. Which is a CSS3 pseudo-element?
 - a. `first-line`
 - b. `first-letter`
 - c. `before`
 - d. `selection`
16. Which pseudo-element enables you to insert content into Web pages using style rules?
 - a. `first-line`
 - b. `first-letter`
 - c. `before`
 - d. `selection`
17. Which of the values the `rgba` set (255,100,178,0.5) represents opacity?
 - a. 255
 - b. 100
 - c. 178
 - d. 0.5
18. The browser prefix `-moz` targets which rendering engine?
 - a. WebKit
 - b. Mozilla
 - c. Opera
 - d. all of them

Skills Review

1. Implement pseudo-elements.

- a. In your text editor, open `HTM I-6.html` from the Unit I/Review folder where you store your Data Files, insert a blank line before the closing body tag, insert a paragraph element containing your first and last name and the text **HTML5 Unit I, Skills Review**, save it as **history.html**, then use a CSS comment to add the same information to `HTM I-7.css` and save it as **bigj.css**.
- b. In `bigj.css`, create a style rule with the `.content .maintext:first-letter` selector, then add name-value pairs to set font size to **3em**, float the element on the **left**, and set line height to **0.8em**.
- c. Save your work, open `history.html` in your browser, then verify that each of the three paragraphs telling the history of the restaurant begins with a drop cap.

2. Add generated content.

- a. Return to `bigj.css` in your text editor.
- b. Below the footer style rule, create a style rule based on the `footer .callout:before` selector.
- c. Within the style rule you created, add a name-value pair that specifies the file **phone.png** located in the **images** directory as the content of the specified pseudo-element.
- d. Save your work, reload `history.html` in your browser, scroll down to the footer, then verify that a phone icon is displayed before each phone number.

Skills Review (continued)

3. Integrate opacity.

- Return to `bigj.css` in your text editor, then locate the `figcaption` style rule.
- Add name-value pairs to set the position to **absolute**, left and bottom to **0**, padding to **0.5em 0**, and width to **100%**.
- Add a name-value pair that sets the background property to an rgb value of **255, 255, 255**.
- Beneath the name-value pair you inserted in Step c, add a name-value pair that sets the background to an rgba value of **255, 255, 255, 0.6**.
- Save your work, reload `history.html` in your browser, then verify that the figure caption has a translucent background and is overlaid on the image of the pizza sign.

4. Create rounded corners.

- Return to `bigj.css` in your text editor, then locate the `#words` style rule.
- Add name-value pairs to set the width to **40%** and margin to **0 auto**.
- Add name-value pairs for the `-webkit` and `-moz` browser-specific properties to create rounded corners, setting the values to **15px**, then add the generic property set to the same value.
- Save your work, reload `history.html` in your browser, then verify that the black box containing the list of words is narrower than the rest of the page content and is displayed with rounded corners.

5. Create text shadows.

- Return to `bigj.css` in your text editor.
- At the end of the `#mainnav a:hover` style rule, add code to create a text shadow using the values **1px 1px 1px black**.
- Save your work, then reload `history.html` in your browser.
- Move the mouse pointer over the links in the nav bar and verify that a text shadow is displayed when the mouse pointer is over each link.

6. Add box shadows.

- Return to `bigj.css` in your text editor
- In the figure style rule, add name-value pairs for the `-webkit` and `-moz` browser-specific properties to create box shadows, setting the values to **2px 2px 4px red**, then add the generic property set to the same value.
- Save your changes, reload `history.html` in your browser, then verify that a red box shadow is displayed on the right and bottom edges of the image of the pizza sign, as shown in Figure I-18.

7. Test browser capabilities.

- Return to `history.html` in your text editor, delete the three-line comment just before the closing `</head>` tag, insert a blank line in the same location, then indent and enter the element `<script src="scripts/modernizr-1.6.min.js"></script>`.
- Add the `no-js` class to the opening `<html>` tag, then save your work.
- Return to `bigj.css` in your text editor, create a new style rule based on the `.boxshadow figure` selector, then cut and paste the three box shadow name-value pairs from the figure style rule to the new rule you created.

FIGURE I-18



Photo/Sasha Vodnik

Skills Review (continued)

- d. At the bottom of the figure style rule, add code to set the bottom and right borders to **2px solid red**.
- e. Above the first name-value pair in the **.boxshadow figure** style rule, add code to set all borders to **none**.
- f. Save your work, reload history.html in your browser. Verify that recent browser versions continue to display a box shadow on the image of the pizza sign, while older browsers display red lines along the right and bottom of the image to approximate a box shadow.
- g. Convert bigj.css to a print style sheet with the name **bjprint.css**, validate the code for your Web pages and your style sheets, then make changes as necessary to fix any errors.
- h. If you have space on a Web server, publish your Web site, then open history.html in your browser from the published location.

Independent Challenge 1

As you continue your work on the Web site of the Spotted Wren Garden Center, you continue adjusting the layout by incorporating advanced CSS styles.

- a. Open HTM I-8.html from the Unit I/IC1 folder in your text editor. Insert a blank line before the closing body tag, insert a paragraph element containing your first and last name and the text **HTML5 Unit 1, Independent Challenge 1**, save it as **resource.html**, then use a CSS comment to add the same information to HTM I-9.css, saving the file as **spotwren.css**.
- b. In spotwren.css, create a style rule that applies to the first line of li elements within article elements. Set the font weight to **bold**. Save your work, then open resource.html in your browser and verify that only the first line of each list item is displayed in bold.
- c. Add CSS code to overlay figure captions over the associated figure content by positioning them absolutely with zero offset from the bottom left. Add a background rgb color of **(221, 221, 221)** followed by the same value as an rgba value. Determine an appropriate alpha value through trial and error. Save your work, then reload resource.html in your browser and verify the "Prairie Coneflower" figure caption is legible.
- d. Create a purple text shadow for figure captions using the values **1px 1px 1px #909** for the text-shadow property. Save your work and preview resource.html in your browser.
- e. Add a green text shadow to footer text using the values text-shadow: **1px 1px 2px green** for the text-shadow property. Save your work, preview resource.html in your browser, and compare the page to Figure I-19.

FIGURE I-19

The screenshot shows the Spotted Wren Garden Center website. It features a navigation bar with links for Home, Hours, Location, and Gardening Resources. The main content area includes a 'Resources' section with links to a PDF tip sheet, a weather forecast, and plant hardiness zone information. A table provides average annual minimum temperatures in Fahrenheit and Celsius, along with last and first frost dates for different zones. Below the table is a list of recommended drought-tolerant perennials. The footer contains contact information and social media links.

Zone	Average annual minimum temp		Last frost	First frost
	Fahrenheit	Celsius		
4b	-25°F to -20°F	-28.9°C to -31.6°C		
5a	-20°F to -15°F	-26.2°C to -28.8°C	May 1	Oct 15
5b	-15°F to -10°F	-23.4°C to -26.1°C		

Recommended drought-tolerant perennials:
 Black-eyed Susan
 Rudbeckia hirta
 Prairie coneflower, Echinacea
 Ratibida columnaris
 Perennial sunflower, Jerusalem artichoke
 Helianthus tuberosus
 Mexican Hat coneflower, Gray-Headed coneflower
 Ratibida pinnata
 May night salvia, Caradonna salvia
 Salvia nemorosa
 Prairie aster
 Aster tataricus
 Blanket flower
 Gaillardia aristata
 Golden tickseed
 Coreopsis tinctoria
 Poppy mallow
 Callirhoe involucrata
 Cranesbill geranium
 Geranium maculatum
 Spike gayfeather
 Liatris spicata
 Butterfly milkweed
 Asclepias tuberosa

Independent Challenge 1 (continued)

- Return to `spotwren.css` in your text editor, change the border color for the aside element to black, remove the name-value pairs that specify the background image and background color for the body element, remove the second background name-value pair for `figcaption`, and remove all non-border name-value pairs that specify color for other elements. Remove text shadows and in all other rules based on pseudo-classes, change all colors to black. Save a copy of the file as **`swprint.css`**.
- Validate your HTML and CSS documents.
- If you have space on a Web server, publish your Web site, then open `resource.html` in your browser from the published location.
- Close your browser, then close your text editor.

Independent Challenge 2

As you continue your work on the Murfreesboro Recreational Soccer League Web site, you integrate rounded corners into the design to evoke the center circle on a soccer field. You also incorporate some advanced CSS styles to enhance existing elements.

- Open `HTM I-10.html` from the Unit I/IC2 folder in your text editor. Insert a blank line before the closing body tag, insert a paragraph element containing your first and last name and the text **HTML5 Unit I, Independent Challenge 2**, save the file as **`schedule.html`**, then use a CSS comment to add the same information to `HTM I-11.css` and save it as **`mrs1.css`**.
- In `mrs1.css`, locate the `h1 #affiliation` style rule. Add a second name-value pair for background after the existing background property. Specify an `rgba` value for the new property, copying the existing `rgb` value and experimenting to find an alpha value that enables you to see the texture of the grass but keeps the text highly legible. Save your work and preview `schedule.html` in your browser. Your main heading and subheading should match those shown in Figure I-20.
- Add code to the style rule for list items in the `#mainnav` element that sets the border radius to **`20px`**. Specify all necessary browser-specific properties as well as the generic property. Save your work, then reload `schedule.html` in your browser. Your nav bar links should match those shown in Figure I-20.
- Add code to the style rule for the `#skiplink` element that sets the bottom-left border radius to **`20px`**. Specify all necessary browser-specific properties as well as the generic property. Save your work, then reload `schedule.html` in your browser. Your skip link should match the one shown in Figure I-20.

FIGURE I-20



Independent Challenge 2 (continued)

Advanced Challenge Exercise



- Using a pseudo-element, create a style rule that inserts a space followed by the word "Team" after each h3 element within an article element. (*Hint:* To create the space, insert a space after the opening quote for the value but before the first letter of the text to insert.)
 - Create a box shadow around all four sides of every img element within a figure element. Set both the x and y offsets to **0px**, and use a high value, such as **20px**, for the blur. Use **#ffd700** for the shadow color. Specify all necessary browser-specific properties as well as the generic property. Save your work, then reload schedule.html in your browser. Your headings and figure should match those shown in Figure I-21.
- e. Return to mrs1.css in your text editor, remove all attribute-value pairs that reference colors as well as the pairs that specify background images, then change all colors in rules based on pseudo-classes to black. Save a copy of the file as **mrsprint.css**.
 - f. Validate your HTML and CSS documents.
 - g. If you have space on a Web server, publish your Web site, then open schedule.html in your browser from the published location.
 - h. Close your browser, then close your text editor.

Independent Challenge 3

As you continue to work with Diego Merckx to refine the design of the Hotel Natoma Web site, you implement a few advanced CSS features.

- a. Open HTM I-12.html from the Unit I/IC3 folder in your text editor. Insert a blank line before the closing body tag, then insert a paragraph element containing your first and last name and the text **HTML5 Unit I, Independent Challenge 3**. Save the file as **nearby.html**. Open HTM I-13.css, below the existing comment at the top of the document, add another comment containing your first and last name and the text **HTML5 Unit I, Independent Challenge 3**. Save the file as **natoma.css**.
- b. In natoma.css, add CSS code to overlay figure captions over the associated figure content by positioning them absolutely with 0 offset from the bottom left. Add an appropriate rgba value for the background and precede it with an appropriate rgb color as a backup. Save your work, then reload nearby.html in your browser and verify that the figure captions are clearly legible.

FIGURE I-21



9507848116/Shutterstock.com
Xtreme/Shutterstock.com

Independent Challenge 3 (continued)

- c. Add rounded corners to the top left and bottom right of the Web page. Start by adding properties to round the top left and bottom right corners of the #box element with a **40px** radius. Next add properties for the h1 element to round the top left corner with a radius of **35px**, and for the footer element to round the bottom right corner with a radius of **35px**. Ensure that you specify both browser-specific and generic properties for each corner. Save your work, then reload nearby.html in your browser and verify that both corners are rounded, as shown in Figure I-22.
- d. Add rounded corners to the top left and bottom right corner of each list item in the main nav bar, with a radius of **20px**. Ensure that you specify both browser-specific and generic properties for each corner. Save your work, then reload nearby.html in your browser and verify that both corners are rounded, as shown in Figure I-22.

FIGURE I-22



Photos/Sasha Vodnik

Advanced Challenge Exercise



- Replace the existing script reference in nearby.html with a reference to the Modernizr script in the scripts folder. If necessary, refer to the "Testing Browser Capabilities with Modernizr" lesson in this unit for the script element code. In addition, add a class attribute to the html element with a value of **"no-js"**.
 - Create a new style rule based on the ".borderradius #mainnav li" selector, then move all the border-radius properties from the "#mainnav li" style rule to the new rule.
 - In the "#mainnav li" style rule, set the background to the file **buttonbg.png**, which is located in the images folder. Add another name-value pair to set the width to **102px**.
 - Copy the name-value pair in the "#mainnav li" style rule that specifies a background color and add it to the ".borderradius #mainnav li" rule. Add a name-value pair to the ".borderradius #mainnav li" rule that sets width to **auto**.
 - Save your work, then if you have access to Internet Explorer version 8 or earlier, or an older version of another browser, open nearby.html in your older browser. The nav bar buttons should be displayed with rounded corners, but each button should be the same width, which indicates that the backup rule based on the Modernizr testing is applying an image for the button backgrounds to simulate rounded corners.
- e. In natoma.css, change all border colors to **black**, change all colors in rules based on pseudo-classes to **black**, remove the second background attribute for the figcaption element, then remove all other attribute-value pairs that reference colors. Save a copy of the file as **hnprint.css**.
 - f. Validate all your HTML and CSS documents.
 - g. If you have space on a Web server, publish your Web site, then open nearby.html in your browser from the published location.
 - h. Close your browser, then close your text editor.

Visual Workshop

In your text editor, open the file HTM I-14.html from the Unit I/VW directory on the drive and folder where you store your Data Files, add a paragraph before the closing body tag that contains your first and last name and the text **HTML5 Unit I, Visual Workshop**, then save the file as **upcoming.html**. Open HTM I-15.css, add a comment containing the same information, then save it as **revision.css**. Open upcoming.html in your browser, then use your text editor to add advanced features to make your Web page match Figure I-23. Add code to test browser capabilities for the box shadow, then add backup styling for older browsers. Save your work, save a copy of your style sheet with the name **revprint.css**, then remove any styles for text shadow and make any other changes necessary for print formatting, leaving the styles that create box shadows unchanged. When you are finished, validate your HTML and CSS code. If you have space on a Web server, publish your files, then open the Web page in your browser from the published location. Close your browser and text editor.

FIGURE I-23

[Skip navigation](#)

[Home](#) [Store History](#) [Upcoming Events](#) [Location](#)



Custom brewed coffee.
Hand-selected books.
Special orders are our specialty.

Upcoming Events

Date	Time	Description
Oct. 31	6:00pm-8:00pm	Spooky stories read aloud (for kids 2-12)
Nov. 2	12:30p-1:30p	Sam Miller reads from <i>Lunchtime Chronicles</i>
	7:00p-8:00p	
Nov. 3	5:30p-7:00p	Cookbook Reading Club monthly discussion
Nov. 4	5:30am-6:45am	"Fitness in the Stacks" Pilates class

412 N. 25th St.
Richmond, VA 23223
(804) 555-2565