CHAPTER 8

Digging Deeper into CSS

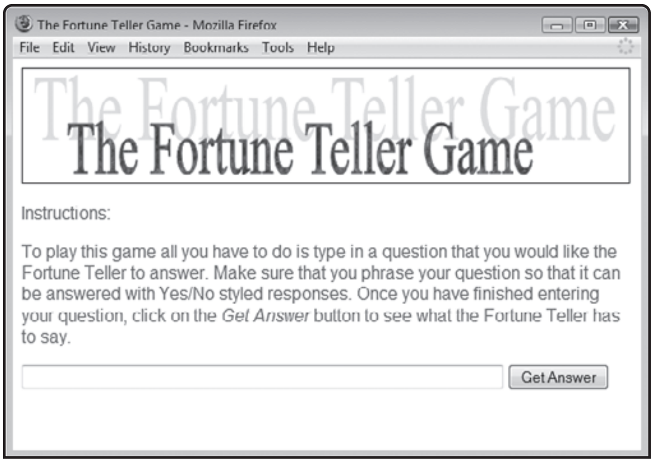
In the previous chapter, you learned different ways of integrating CSS into your web documents and the basic syntax rules used to formulate style rules. In this chapter, you will expand your understanding of CSS by learning how to use it to take control of the placement of content, customize the display of lists, and to style both text and graphic links and buttons. You will also learn how to use CSS to style the flow of graphics and text, to display background images, to style your tables in numerous ways, and to influence the presentation of text in (X)HTML forms.

Specifically, you will learn:

- How to work with and configure containers
- Different options for specifying the presentation of content
- How to modify the display of markers used to identify ordered and unordered list items
- Create rollover links, controls, and buttons
- A number of ways that you can improve the presentation of graphics, tables, and forms

Project Preview: The Fortune Teller Game

This chapter's web project is the development of the Fortune Teller game. When first started, this game displays the screen shown in Figure 8.1.


The Fortune Teller Game - Mozilla Firefox
File Edit View History Bookmarks Tools Help
The Fortune Teller Game
The Fortune Teller Game

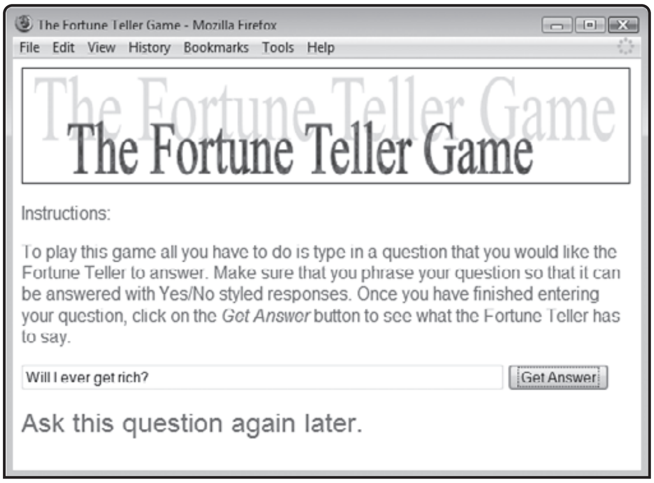
Instructions:

To play this game all you have to do is type in a question that you would like the Fortune Teller to answer. Make sure that you phrase your question so that it can be answered with Yes/No styled responses. Once you have finished entering your question, click on the *Get Answer* button to see what the Fortune Teller has to say.

Figure 8.1

The game's presentation is controlled using an external style sheet.

To play, type in a question in the text control located in a form at the bottom of the browser window and then click on the Get Answer button. For the game to work properly, the player must submit all questions in a form that allows for Yes/No responses. Once the Get Answer button is clicked, the game will display one of ten randomly selected responses, as demonstrated in Figure 8.2.


The Fortune Teller Game - Mozilla Firefox
File Edit View History Bookmarks Tools Help
The Fortune Teller Game
The Fortune Teller Game

Instructions:

To play this game all you have to do is type in a question that you would like the Fortune Teller to answer. Make sure that you phrase your question so that it can be answered with Yes/No styled responses. Once you have finished entering your question, click on the *Get Answer* button to see what the Fortune Teller has to say.

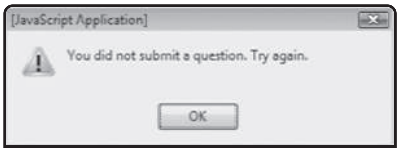
Will I ever get rich?

Ask this question again later.

Figure 8.2

The game's answer to the player's question is displayed at the bottom of the browser window.

After a short three-second pause, the game will automatically clear out both the player's question and the game's answer, readying the game to accept a new question while also providing the player with plenty of time to view and consider the game's answer. In the event that the player clicks on the Get Answer button without entering a question, the game will display the following popup dialog window, notifying the player of his error.



You did not submit a question. Try again.

FIGURE 8.3

Game play resumes as soon as the player clicks on the OK button.

Working with Containers

One of the most useful features of CSS is its ability to control the position at which elements are displayed on web pages. Key to understanding how this works is the concept of a container. A *container* is an entity to which you can apply CSS. In addition, many elements can serve as containers for other elements. To help illustrate this point, look at the following example.

```
<body>
  <p>
    Who did <em>Roger Rabbit</em> marry?
  </p>
</body>
```

Here, the paragraph is used to display a little text. In addition, an `em` element is embedded within the paragraph. Both the paragraph and the `em` elements are containers. Therefore, both can be styled.

HINT In most cases, any style you apply to an element is automatically inherited by any embedded elements. So, if you add a style rule that displays all text within the paragraph in red, the embedded `em` element inherits that color styling as well.

A container has three common presentation attributes that affect how its content is blended in with other content. These container attributes are margins, padding, and borders.

Setting Container Margins

A margin is the space that encloses the container. By changing container margins, you can make things feel less cluttered by adding a little extra white space. Setting a container's margin is easy. All you have to do is use the `margin` property, as demonstrated here:

```
p {margin: 10cm 10cm 5cm 5cm;}
```

Here, the left and right margins have been set to 10 centimeters and the top and bottom margins have been set to 5 centimeters.

HINT

If you prefer to be more explicit, you can specify each margin separately, as demonstrated here:

```
p {margin-left: 10cm; margin-right: 10cm; margin-top: 5cm;
    margin-bottom: 5cm;}
```

Padding Space Between the Container and Its Border

All containers have a border, whether it is visible or not. The border represents the outside edge of the container. If you want, you can add additional padding between a container and its border using the `padding` attribute. The `padding` attribute works just like the `margin` element, as demonstrated here.

```
p {padding: 10cm 10cm 5cm 5cm;}
```

Configuring a Container's Border

By default, container borders are invisible. However, you can display and configure them using the `border-width` and `border-style` properties. The `border-style` property lets you enable and disable the display of the container's border (disabled by default). Take, for example, the following rule.

```
p {border-style: solid;}
```

This rule will display a solid border around every paragraph in the document. The following list outlines the possible range of values supported by the `border-style` property. The last value (`none`) is the default.

- hidden
- dashed
- dotted
- double
- groove
- inset
- outset
- ridge
- solid
- none

Once you have used the `border-style` property to select and display a container's border, you can use the `border-width` property to specify the thickness of that border. You have four choices, as outlined here:

- **thin.** Displays the container using a thin border.
- **medium.** Displays the container using a medium border.
- **thick.** Displays the container using a thick border.
- *value.* Displays the container with a border whose thickness depends on what value you assign (example: 2 cm).

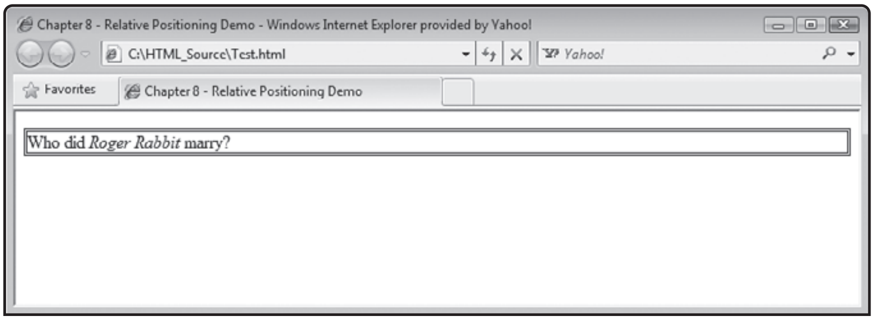
The following example demonstrates how to display a border around all paragraphs in a document using an inset border that is medium thick.

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
  "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">

<html xmlns="http://www.w3.org/1999/xhtml" lang="en" xml:lang="en">

  <head>
    <meta http-equiv="Content-type" content="text/xhtml; charset=UTF-8" />
    <title>Chapter 8 - Working with Borders</title>
    <style type = "text/css">
      p {border-style: double; border-width: medium;}
    </style>
  </head>

  <body>
    <p>
      Who did <em>Roger Rabbit</em> marry?
    </p>
  </body>

</html>
```

Figure 8.4 shows how the paragraph element's border looks when rendered by the browser.

HINT

Note that in this example and in the rest of the examples that you will see in this chapter, embedded style sheets are used to simplify examples and make things easier to present. However, use of external style sheets is still recommended for most real-life web documents.


Chapter 8 - Relative Positioning Demo - Windows Internet Explorer provided by Yahoo!
C:\HTML_Source\Test.html
Yahoo!
Favorites
Chapter 8 - Relative Positioning Demo

Who did *Roger Rabbit* marry?

FIGURE 8.4

An example of how to display and configure an element's content in a container.

TAKING CONTROL OF ELEMENT PLACEMENT

CSS provides you a number of different properties that you can use to take control over the placement of document content within the browser window. CSS supports a coordinate system along with properties that you can use to interact with that system. Figure 8.5 provides a visual depiction of CSS's coordinate system. As you can see, coordinate 0,0 is located in the upper-left corner of the browser window. 0,0 represents vertical and horizontal values. As the display of an element is moved down the browser window, the value of its y (vertical) coordinate increases. Likewise, as an element moves from left to right, the value of its x (horizontal) coordinate increases.

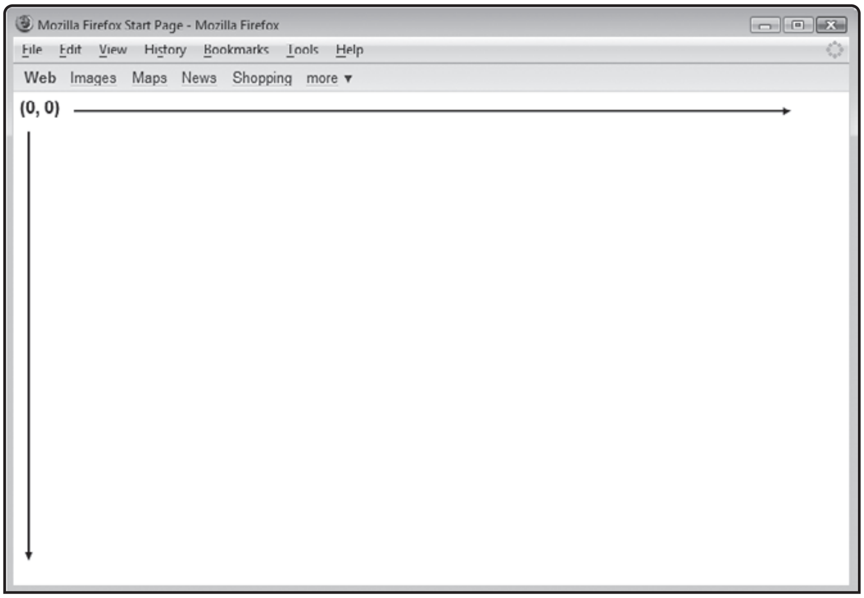

Mozilla Firefox Start Page - Mozilla Firefox
File Edit View History Bookmarks Tools Help
Web Images Maps News Shopping more ▼
(0, 0)

FIGURE 8.5

A depiction of CSS's coordinate system.

Table 8.1 provides a list of CSS properties that you can use to affect element positioning.

Table 8.1 CSS Properties that Affect Element Positioning

Property	Property	Description
top	pixel value	Offset from the top of the browser's display area (absolute) or from its default location as determined by the browser.
bottom	pixel value	Offset from the bottom of the browser's display area (absolute) or from its default location as determined by the browser.
left	pixel value	Offset from the top-left side of the browser's display area (absolute) or from its default location as determined by the browser.
right	pixel value	Offset from the top-right side of the browser's display area (absolute) or from its default location as determined by the browser.
position	static, absolute, relative, fixed, float	Determines how to position an element.
z-order	numeric value	A value that determines the order in which elements appear when they overlap one another.

Static Positioning

Static is the default positioning option, and it is the option you have been using throughout this book. With static positioning, elements are displayed in the order they are laid out in your documents, one after another, as demonstrated in the following example.

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
  "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">

<html xmlns="http://www.w3.org/1999/xhtml" lang="en" xml:lang="en">

  <head>
    <meta http-equiv="Content-type" content="text/xhtml; charset=UTF-8" />
    <title>Chapter 8 - Static Positioning Demo</title>
    <style type = "text/css">
      p {border-width: thin; border-style: groove; height: 200px;
      width: 200px;}
    #yellowbox {background-color: yellow;}
    #bluebox {background-color: blue;}
  </style>
</head>

<body>
  <p id = "yellowbox">
    Yellow Box
  </p>
  <p id = "bluebox">
    Blue Box
  </p>
</body>

</html>
```

Figure 8.6 shows how the resulting web page is laid out when this example is rendered by the browser.

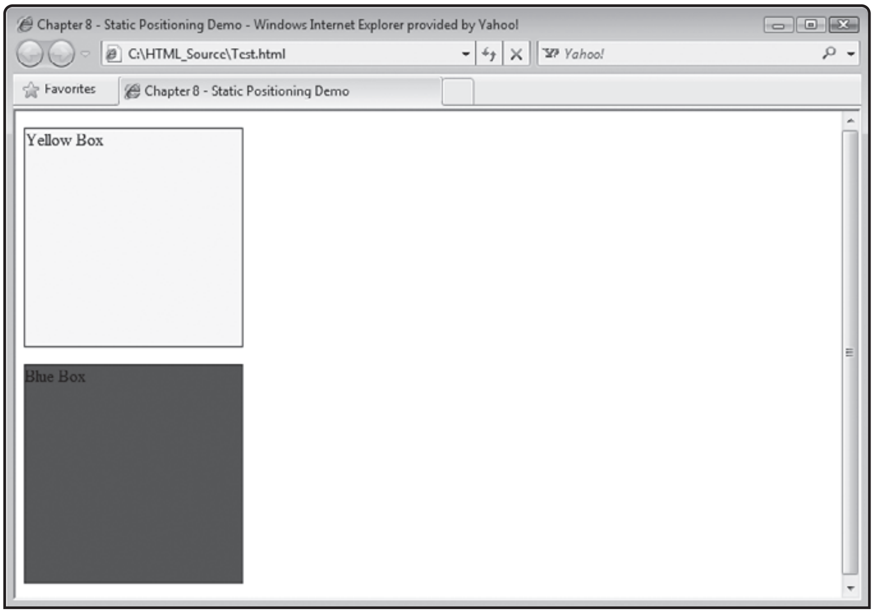

Chapter 8 - Static Positioning Demo - Windows Internet Explorer provided by Yahoo!
C:\HTML_Source\Test.html
Yahoo!
Favorites
Chapter 8 - Static Positioning Demo
Yellow Box
Blue Box

FIGURE 8.6

An example of how to control container placement using static positioning.

Absolute Positioning

Absolute positioning provides precise control over the placement of your content. You simply specify the location where you want an element's content to appear using the `top`, `bottom`, `left`, and `right` properties, as demonstrated by the following example.

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
  "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">

<html xmlns="http://www.w3.org/1999/xhtml" lang="en" xml:lang="en">

  <head>
    <meta http-equiv="Content-type" content="text/xhtml; charset=UTF-8" />
    <title>Chapter 8 - Static Positioning Demo</title>
  </head>

  <body>

    <p style = "position: absolute; left: 50px; top: 25px; z-index: 20;" >
      <img src = "cats.jpg" width = "270" height = "200"
        alt = "Picture of two cats" />
    </p>

    <p style = "position: absolute; left: 200px; top: 150px; z-index: 10;">
      <img src = "river.jpg" width = "270" height = "200"
        alt = "Picture of a river" />
    </p>

  </body>

</html>
```

As you can see, in this example two images are displayed on the browser window. The assigned coordinates cause the images to overlap. As a result, the image with the highest specified `z-index` value is displayed on top of the other figure. Figure 8.7 shows how this example's context looks when rendered by the browser.

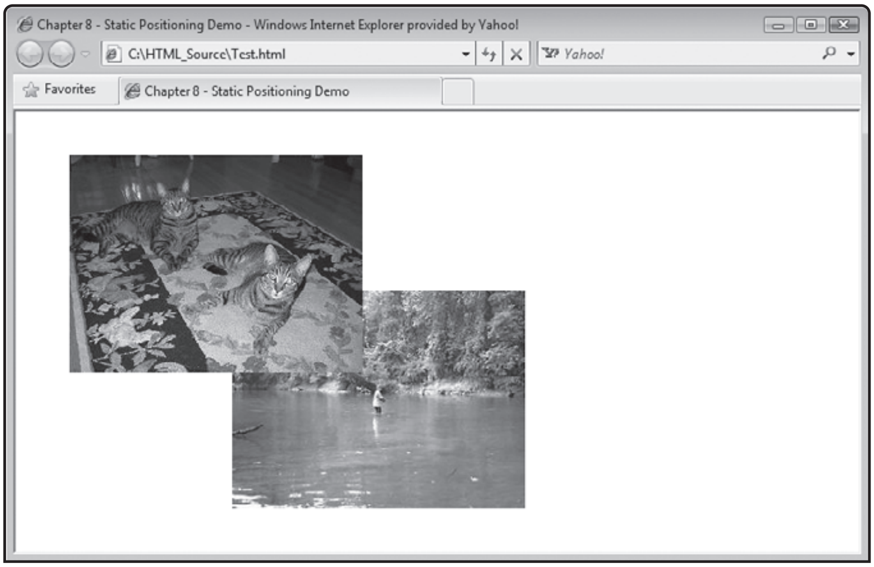

Chapter 8 - Static Positioning Demo - Windows Internet Explorer provided by Yahoo!
C:\HTML_Source\Test.html
Yahoo!
Favorites
Chapter 8 - Static Positioning Demo

FIGURE 8.7

Using absolute position to exercise precise control over the display of two images.

Relative Positioning

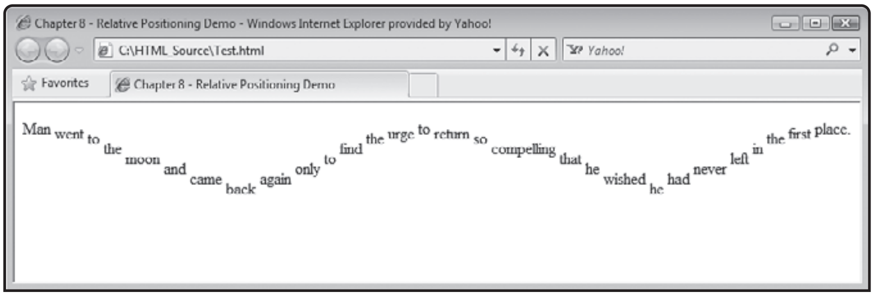
A problem with absolute positioning is that not all computers are set up with the same screen resolution. Therefore, the coordinates system changes from user to user. As a result, it can be difficult to ensure web pages look the way you want them to at different resolutions. Relative positioning sets an element's position relative to other elements.

Using relative positioning, you can create an application that automatically repositions its elements based on window resolution. This helps keep things from overlapping or from being pushed out of view. As an example of how relative positioning works, look at the following example.

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
  "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">

<html xmlns="http://www.w3.org/1999/xhtml" lang="en" xml:lang="en">

  <head>
    <meta http-equiv="Content-type" content="text/xhtml; charset=UTF-8" />
    <title>Chapter 8 - Relative Positioning Demo</title>
  </head>

  <body>
```

```
    <p>
      Man
      <span style = "position: relative; top: 5px;">went</span>
      <span style = "position: relative; top: 10px;">to</span>
      <span style = "position: relative; top: 20px;">the</span>
      <span style = "position: relative; top: 30px;">moon</span>
      <span style = "position: relative; top: 40px;">and</span>
      <span style = "position: relative; top: 50px;">came</span>
      <span style = "position: relative; top: 60px;">back</span>
      <span style = "position: relative; top: 50px;">again</span>
      <span style = "position: relative; top: 40px;">only</span>
      <span style = "position: relative; top: 30px;">to</span>
      <span style = "position: relative; top: 20px;">find</span>
      <span style = "position: relative; top: 10px;">the</span>
      <span style = "position: relative; top: 5px;">urge</span>
      <span style = "position: relative; top: 0px;">to</span>
      <span style = "position: relative; top: 5px;">return</span>
      <span style = "position: relative; top: 10px;">so</span>
      <span style = "position: relative; top: 20px;">compelling</span>
      <span style = "position: relative; top: 30px;">that</span>
      <span style = "position: relative; top: 40px;">he</span>
      <span style = "position: relative; top: 50px;">wished</span>
      <span style = "position: relative; top: 60px;">he</span>
      <span style = "position: relative; top: 50px;">had</span>
      <span style = "position: relative; top: 40px;">never</span>
      <span style = "position: relative; top: 30px;">left</span>
      <span style = "position: relative; top: 20px;">in</span>
      <span style = "position: relative; top: 10px;">the</span>
      <span style = "position: relative; top: 5px;">first</span>
      <span style = "position: relative; top: 0px;">place.</span>
    </p>
  </body>

</html>
```

Figure 8.8 demonstrates how this page will look when loaded into the browser.


Chapter 8 - Relative Positioning Demo - Windows Internet Explorer provided by Yahoo!
C:\HTML Source\Test.html
Yahoo!
Favorites
Chapter 8 - Relative Positioning Demo

Man went to the moon and came back again only to find the urge to return so compelling that he wished he had never left in the first place.

FIGURE 8.8

Using relative positioning to control text presentation.

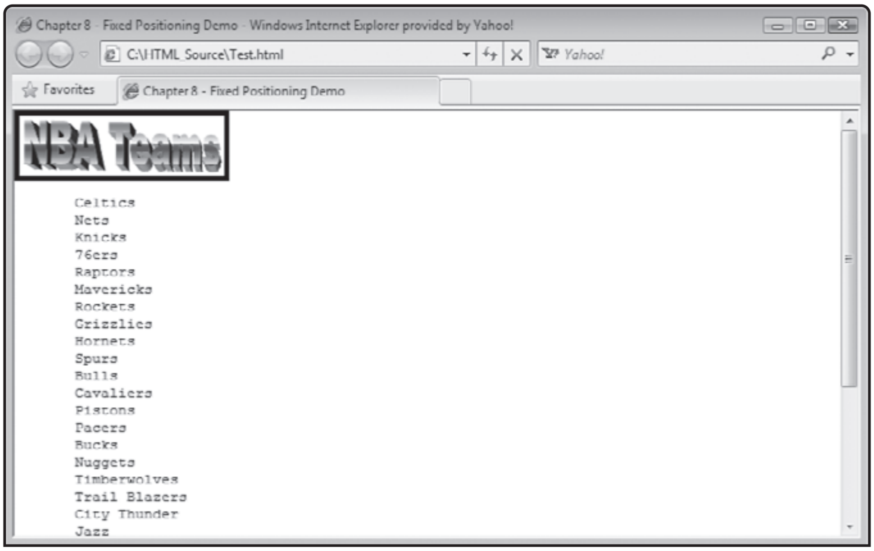
Fixed Positioning

Elements that are positioned using fixed positioning do not scroll or change position when the user scrolls up and down the web page. Instead, the elements remain visible at the same location while the rest of the page's content scrolls behind them. Fixed positioning is often used to keep report headings visible at the top of a web page at all times. The following example demonstrates how to work with fixed positioning.

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
  "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">

<html xmlns="http://www.w3.org/1999/xhtml" lang="en" xml:lang="en">

  <head>
    <meta http-equiv="Content-type" content="text/xhtml; charset=UTF-8" />
    <title>Chapter 8 - Fixed Positioning Demo</title>
    <style type = "text/css">
      p {border-width: thick; border-style: solid; height: 55px;
         width: 189px; position: fixed; top: -16px; left: 0px;}
    </style>
  </head>

  <body>
    <p>
      <img src = "nba.png" alt = "A graphic NBA header" />
    </p>
    <pre>
```

```
      Celtics
      Nets
      Knicks
      76ers
      Raptors
      Mavericks
      Rockets
      Grizzlies
      Hornets
      Spurs
      Bulls
      Cavaliers
      Pistons
      Pacers
      Bucks
      Nuggets
      Timberwolves
      Trail Blazers
      City Thunder
      Jazz
      Hawks
      Bobcats
      Heat
      Magic
      Wizards
      Warriors
      Clippers
      Lakers
      Suns
      Kings
    </pre>
  </body>

</html>
```

Note that the four blank lines in the previous document are no accident.

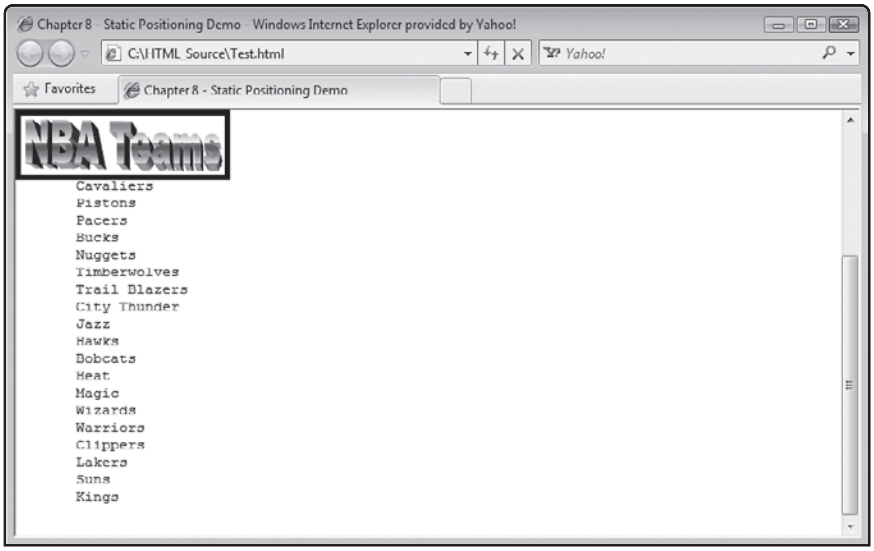

Chapter 8 - Fixed Positioning Demo - Windows Internet Explorer provided by Yahoo!
C:\HTML_Source\Test.html
Yahoo!
Favorites
Chapter 8 - Fixed Positioning Demo
NBA Teams

```
Celtics
Nets
Knicks
76ers
Raptors
Mavericks
Rockets
Grizzlies
Hornets
Spurs
Bulls
Cavaliers
Pistons
Pacers
Bucks
Nuggets
Timberwolves
Trail Blazers
City Thunder
Jazz
```

FIGURE 8.9

Using fixed positioning to set up a non-scrolling graphic heading.

Figure 8.10 demonstrates how this page will look when first loaded into the browser.


Chapter 8 - Static Positioning Demo - Windows Internet Explorer provided by Yahoo!
C:\HTML_Source\Test.html
Yahoo!
Favorites
Chapter 8 - Static Positioning Demo
NBA Teams

```
Cavaliers
Pistons
Pacers
Bucks
Nuggets
Timberwolves
Trail Blazers
City Thunder
Jazz
Hawks
Bobcats
Heat
Magic
Wizards
Warriors
Clippers
Lakers
Suns
Kings
```

FIGURE 8.10

The graphic heading remains in place as the page's content is scrolled.

Float Positioning

The position option supported by CSS is float. When an element is set up to float, it can be assigned to shift or float right or left on its current vertical line. The `float` property can be assigned any of the following values; `left`, `right`, `none`. A floated element will shift position in the specified direction until it makes contact with the edge of the container in which it is defined or until it makes contact with another float. Any text displayed on the page will follow down and around the floated element.

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
  "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">

<html xmlns="http://www.w3.org/1999/xhtml" lang="en" xml:lang="en">

  <head>
    <meta http-equiv="Content-type" content="text/xhtml; charset=UTF-8" />
    <title>Chapter 8 - Float Positioning Demo</title>
    <style type = "text/css">
      p {float: right;}
    </style>

  </head>

  <body>
    <p>
      <img src = "cats.jpg" alt = "Picture of two little kittens." />
    </p>
  </body>

</html>
```

Figure 8.11 demonstrates how this page will look when first loaded into the browser.

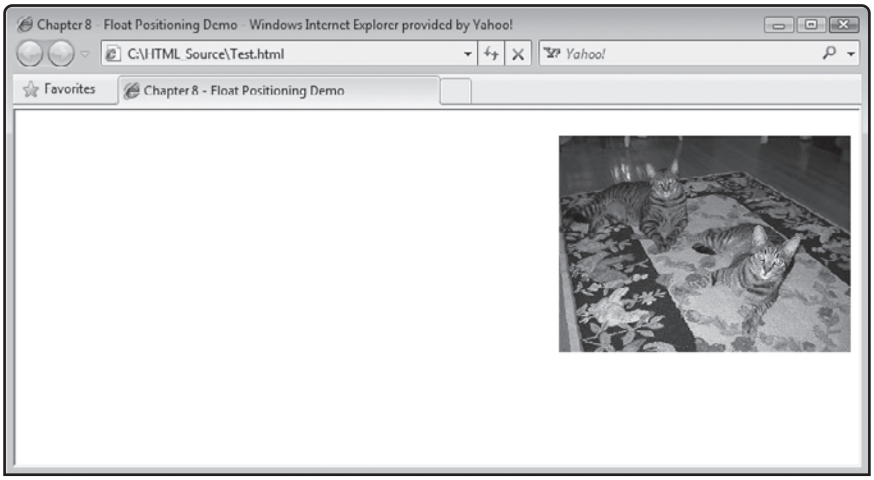

Chapter 8 - Float Positioning Demo - Windows Internet Explorer provided by Yahoo!
C:\HTML_Source\Test.html
Yahoo!
Favorites
Chapter 8 - Float Positioning Demo

FIGURE 8.11

Using float positioning to control the horizontal positioning of an element.

Using CSS to Style Your Lists

You can use CSS to style just about any content you embed in your web pages. This includes styling ordered and unordered lists. By default, ordered lists are displayed using Arabic numerals (1,2,3...) and ordered lists are displayed using a round disk shaped bullet. Using CSS, you can modify these defaults, displaying your lists using a variety of different markers.

Customizing Markers for Ordered Lists

If you prefer to use something other than the standard Arabic numerals when defining ordered lists, you can use the `list-style-type` property to select any of the alternative numeric styles listed in Table 8.2.

Table 8.2 Values Supported by the list-style Property

Numeric Style	Description	Example
Decimal	Displays Arabic numeral characters (default)	1, 2, 3, 4, 5, ...
lower-alpha	Displays lowercase alphabetic characters	a, b, c, d, e, ...
lower-roman	Displays lowercase Roman numeral characters	i, ii, iii, iv, v, ...
upper-alpha	Displays uppercase alphabetic characters	A, B, C, D, E, ...
upper-roman	Displays uppercase Roman numeral characters	I, II, III, IV, V, ...
none	Suppresses the display of characters	N/A

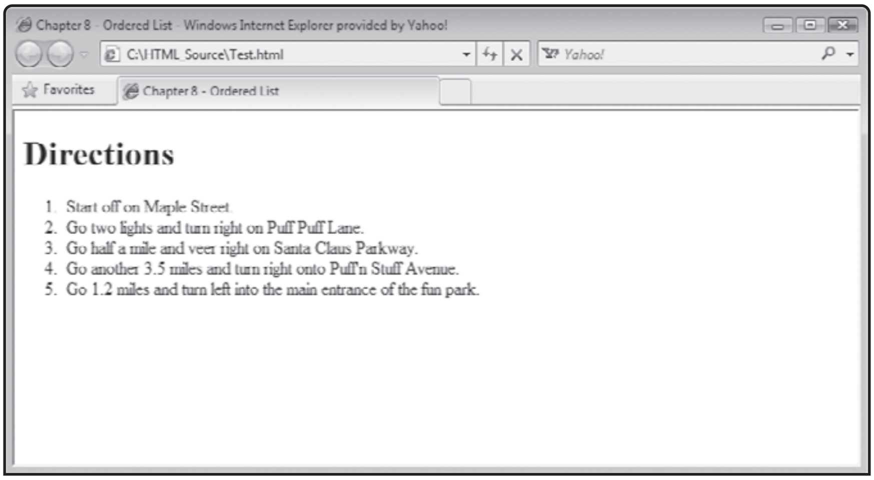
TRAP

Actually, CSS supports a number of other numeric styles, including: armenian, decimal-leading-zero, georgian, inherit, lower-greek, lower-latin, and upper-latin. However, Internet Explorer does not currently support any of these numeric style types, so it is recommended that you avoid their usage.

As an example of how to work with the `list-style-type` property to change the presentation of your ordered lists, look at the following:

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
  "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">

<html xmlns="http://www.w3.org/1999/xhtml" lang="en" xml:lang="en">

  <head>
    <meta http-equiv="Content-type" content="text/xhtml; charset=UTF-8" />
    <title>Chapter 8 - Ordered List</title>
  </head>

  <body>
    <h1>Directions</h1>
    <ol>
      <li>Start off on Maple Street.</li>
      <li>Go two lights and turn right on Puff Puff Lane.</li>
      <li>Go half a mile and veer right on Santa Claus Parkway.</li>
      <li>Go another 3.5 miles and turn right onto Puff'n Stuff Avenue.</li>
      <li>Go 1.2 miles and turn left into the main entrance of the fun
          park.</li>
    </ol>
  </body>

</html>
```

Here, a list of five ordered items has been defined. Figure 8.12 shows how this list looks when rendered by the browser.


Chapter 8 - Ordered List - Windows Internet Explorer provided by Yahoo!
C:\HTML Source\Test.html
Yahoo!
Favorites
Chapter 8 - Ordered List

Directions

1. Start off on Maple Street.
2. Go two lights and turn right on Puff Puff Lane.
3. Go half a mile and veer right on Santa Claus Parkway.
4. Go another 3.5 miles and turn right onto Puff'n Stuff Avenue.
5. Go 1.2 miles and turn left into the main entrance of the fun park.

Figure 8.12

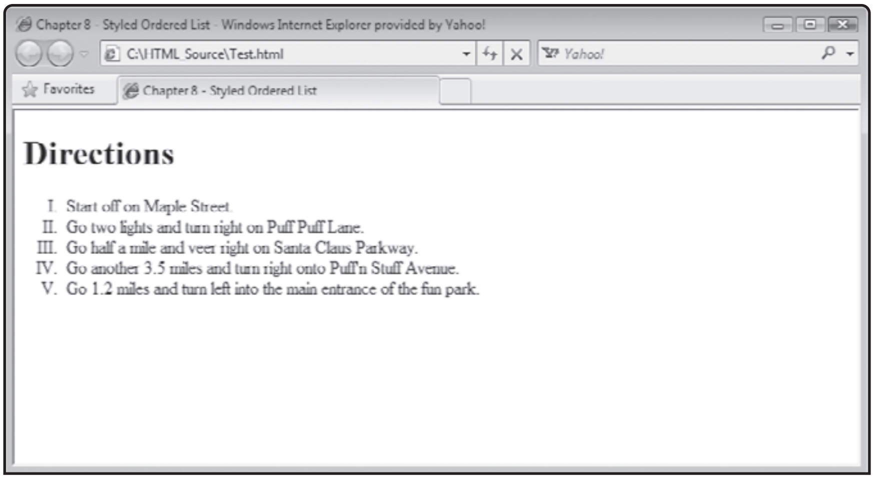
An example of an ordered list displayed using the default marker.

Now, let's rework the example as shown below.

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
  "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">

<html xmlns="http://www.w3.org/1999/xhtml" lang="en" xml:lang="en">
```

```
  <head>
    <meta http-equiv="Content-type" content="text/xhtml; charset=UTF-8" />
    <title>Chapter 8 - Styled Ordered List</title>
    <style type = "text/css">
      ol {list-style-type: upper-roman;}
    </style>
  </head>

  <body>
    <h1>Directions</h1>
    <ol>
      <li>Start off on Maple Street.</li>
      <li>Go two lights and turn right on Puff Puff Lane.</li>
      <li>Go half a mile and veer right on Santa Claus Parkway.</li>
      <li>Go another 3.5 miles and turn right onto Puff'n Stuff Avenue.</li>
      <li>Go 1.2 miles and turn left into the main entrance of the fun
          park.</li>
    </ol>
  </body>

</html>
```

As you can see, this example includes a style rule that uses the `list-style-type` property to alter the display of the marker to upper-roman, as shown in Figure 8.13.


Chapter 8 - Styled Ordered List - Windows Internet Explorer provided by Yahoo!
C:\HTML Source\Test.html
Yahoo!
Favorites
Chapter 8 - Styled Ordered List

Directions

I. Start off on Maple Street.
II. Go two lights and turn right on Puff Puff Lane.
III. Go half a mile and veer right on Santa Claus Parkway.
IV. Go another 3.5 miles and turn right onto Puff'n Stuff Avenue.
V. Go 1.2 miles and turn left into the main entrance of the fun park.

FIGURE 8.13

An example of an unordered list that has been configured to use the upper-roman marker.

Changing Markers for Unordered Lists

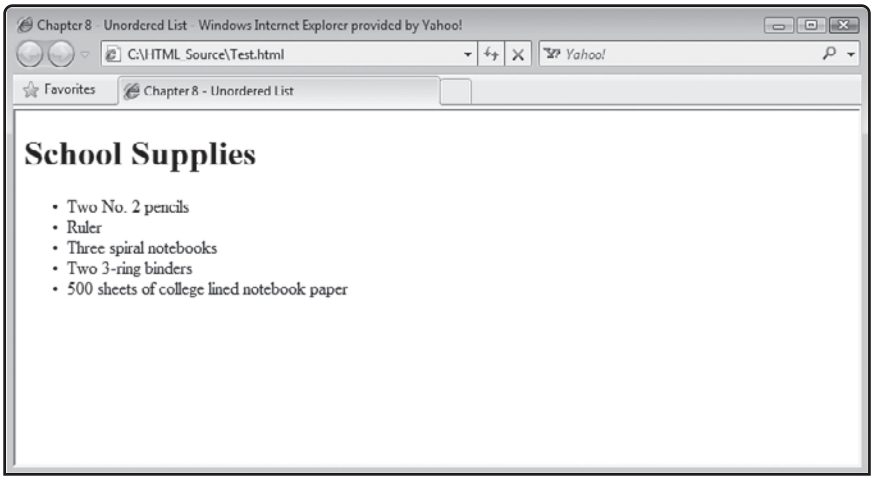
By taking advantage of the `list-style-type` property, you can modify the appearance of the markers used to style your unordered lists. You can even hide the markers altogether. The following lists outlines various markers that can be assigned to this property.

- **disc.** A darkened round marker (default).
- **circle.** A hollow or empty circle marker.
- **square.** A darkened square marker.
- **none.** A blank or hidden marker.

As an example of how to work with the `list-style-type` property to change the presentation of your unordered lists, look at the following.

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
  "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">

<html xmlns="http://www.w3.org/1999/xhtml" lang="en" xml:lang="en">

  <head>
    <meta http-equiv="Content-type" content="text/xhtml; charset=UTF-8" />
    <title>Chapter 8 - Unordered List</title>
  </head>

  <body>
    <h1>School Supplies</h1>
    <ul>
      <li>Two No. 2 pencils</li>
      <li>Ruler</li>
      <li>Three spiral notebooks</li>
      <li>Two 3-ring binders</li>
      <li>500 sheets of college lined notebook paper</li>
    </ul>
  </body>

</html>
```

Here, a list of five unordered items has been defined. Figure 8.14 shows how this list looks when rendered by the browser.


Chapter 8 - Unordered List - Windows Internet Explorer provided by Yahoo!
C:\HTML Source\Test.html
Yahoo!
Favorites
Chapter 8 - Unordered List

School Supplies

- Two No. 2 pencils
- Ruler
- Three spiral notebooks
- Two 3-ring binders
- 500 sheets of college lined notebook paper

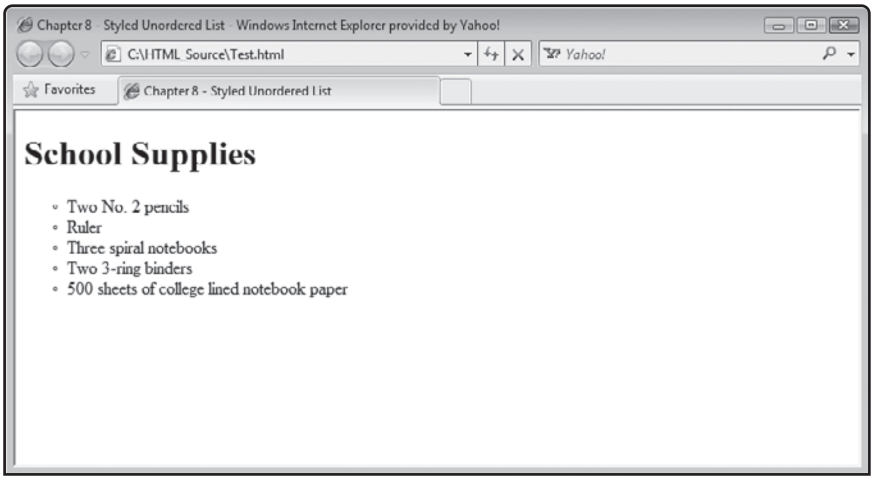
FIGURE 8.14

An example of an unordered list displayed using the default marker.

Now, let's rework the example as shown below.

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
  "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">

<html xmlns="http://www.w3.org/1999/xhtml" lang="en" xml:lang="en">

  <head>
    <meta http-equiv="Content-type" content="text/xhtml; charset=UTF-8" />
    <title>Chapter 8 - Styled Unordered List</title>
    <style type = "text/css">
      ul {list-style-type: circle;}
    </style>
  </head>

  <body>
    <h1>School Supplies</h1>
    <ul>
      <li>Two No. 2 pencils</li>
      <li>Ruler</li>
      <li>Three spiral notebooks</li>
      <li>Two 3-ring binders</li>
      <li>500 sheets of college lined notebook paper</li>
    </ul>
  </body>

</html>
```

As you can see, this example includes a style rule that uses the `list-style-type` property to alter the display of the maker to a circle, as shown in Figure 8.15.


Chapter 8 - Styled Unordered List - Windows Internet Explorer provided by Yahoo!
C:\HTML_Source\Test.html
Yahoo!
Favorites
Chapter 8 - Styled Unordered List

School Supplies

- Two No. 2 pencils
- Ruler
- Three spiral notebooks
- Two 3-ring binders
- 500 sheets of college lined notebook paper

FIGURE 8.15

An example of an unordered list that has been configured to use the circle marker.

As demonstrated in both of the previous examples, take note that regardless of which marker type you elect to use, CSS automatically sizes the marker to ensure that it is kept proportional with its associated text.

Creating Custom List Markers

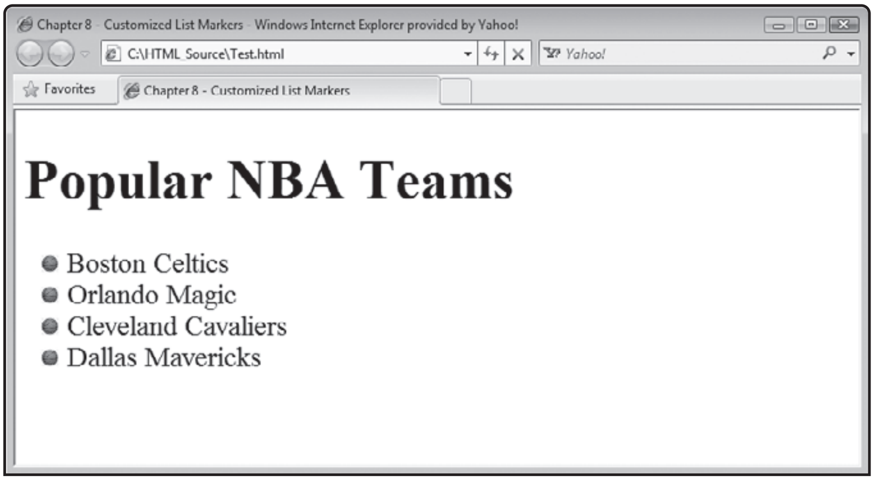
If you are not satisfied with the set of graphic markers made available by CSS, you can create and use your own marker using the `list-style-image` property. To use this property to supply your own custom marker, you must specify the URL of the graphic file where the marker resides. In addition, you must ensure that the marker is proportionally sized to match up with its associated text. CSS will not automatically scale your custom marker.

As an example of how to use the `list-style-image` property, look at the following document.

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
  "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">

<html xmlns="http://www.w3.org/1999/xhtml" lang="en" xml:lang="en">
```

```
  <head>
    <meta http-equiv="Content-type" content="text/xhtml; charset=UTF-8" />
    <title>Chapter 8 - Customized List Markers</title>
    <style type = "text/css">
      body {font-size: 1.6pc;}
      ul {list-style-image: url("ball.jpg");}
    </style>
  </head>

  <body>
    <h1>Popular NBA Teams</h1>
    <ul>
      <li>Boston Celtics</li>
      <li>Orlando Magic</li>
      <li>Cleveland Cavaliers</li>
      <li>Dallas Mavericks</li>
    </ul>
  </body>

</html>
```

As you can see, this example uses a custom graphic named ball.jpg as the basis for creating a customized marker. Figure 8.16 shows how this example looks when rendered by the browser.


Chapter 8 - Customized List Markers - Windows Internet Explorer provided by Yahoo!
C:\HTML_Source\Test.html
Yahoo!
Favorites
Chapter 8 - Customized List Markers

Popular NBA Teams

- Boston Celtics
- Orlando Magic
- Cleveland Cavaliers
- Dallas Mavericks

FIGURE 8.16

An example of an unordered list that uses a customized marker.

Styling Links

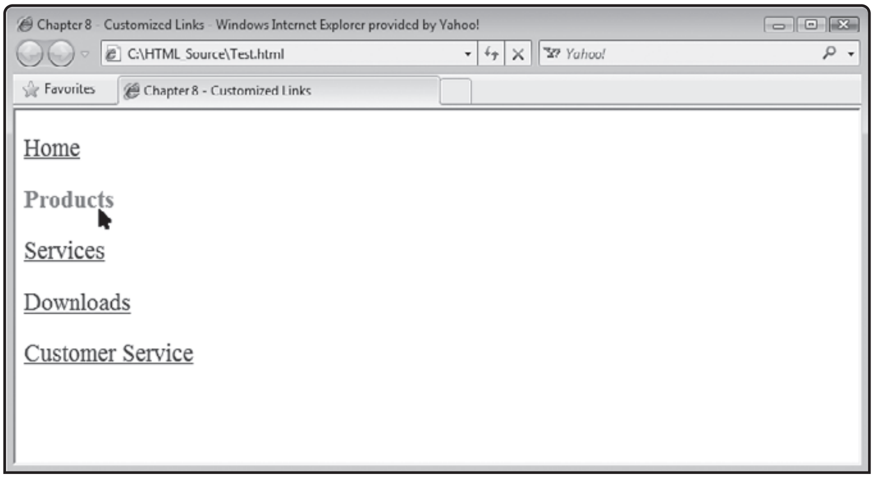
Another useful application of CSS is in style links. This includes both text and graphic links. For example, by default, browsers generally underline links and change the color in which they are displayed based on their status. Blue may be used to identify a link that has not been touched, while links that have been visited may be displayed in purple. If this color scheme is not consistent with the color scheme of your website you may want to modify the style of those links. This is accomplished by applying style rules to the anchor element while using pseudo classes.

Modifying the Presentation of Text Links

You were introduced to pseudo classes in Chapter 7. CSS supports five pseudo classes, all of which are designed to work with the anchor element. The following document demonstrates how you can use them in conjunction with CSS to take control over the appearance of all your links.

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
  "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">

<html xmlns="http://www.w3.org/1999/xhtml" lang="en" xml:lang="en">

  <head>
    <meta http-equiv="Content-type" content="text/xhtml; charset=UTF-8" />
    <title>Chapter 8 - Customized Links</title>
    <style type = "text/css">
      body {font-size: 1.4pc;}
      a:link {color: blue;}
      a:visited {color: green;}
      a:hover {color: red; font-weight: bolder; text-decoration: none;}
      a:active {color: green;}
      a:focus {color: red; font-weight: bolder; text-decoration: none;}
    </style>
  </head>

  <body>
    <p><a href = "index.html">Home</a></p>
    <p><a href = "products.html">Products</a></p>
    <p><a href = "services.html">Services</a></p>
    <p><a href = "downloads.html">Downloads</a></p>
    <p><a href = "custserv.html">Customer Service</a></p>
  </body>

</html>
```

Figure 8.17 demonstrates how things look when this document has been loaded into the browser and the mouse-pointer has been moved over the Products link.


Chapter 8 - Customized Links - Windows Internet Explorer provided by Yahoo!
C:\HTML Source\Test.html
Yahoo!
Favorites
Chapter 8 - Customized Links

Home

Products

Services

Downloads

Customer Service

FIGURE 8.17

An example of how to use CSS to alter the presentation of document text links.

Creating Graphical Links

Another common use of CSS is to convert text links into graphic links. To accomplish this, you need to use CSS to add a border to the link, and give it a background color, as demonstrated in the following example:

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
  "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">

<html xmlns="http://www.w3.org/1999/xhtml" lang="en" xml:lang="en">

  <head>
    <meta http-equiv="Content-type" content="text/xhtml; charset=UTF-8" />
    <title>Chapter 8 - Graphic Links</title>
    <style type = "text/css">
      a {border: 3px solid black;
         display: block;
```

```
        font: 14px Arial;
        text-decoration: none;
        text-align: center;
        width: 200px;
        height: 20px;
        background: yellow;
        color: blue;
      }
    a:focus, a:hover, a:active {
        background: red;
      }

    </style>
  </head>

  <body>
    <p><a href = "index.html">Home</a></p>
    <p><a href = "products.html">Products</a></p>
    <p><a href = "services.html">Services</a></p>
    <p><a href = "downloads.html">Downloads</a></p>
    <p><a href = "custserv.html">Customer Service</a></p>
  </body>

</html>
```

When this document is loaded into the browser, the first CSS style rule configures all of the links so that they are displayed with a solid black border that is 150 pixels wide and 30 pixels high. Their containers have been given a yellow background. The second rule uses two pseudo classes to modify the background color of each link's container whenever the mouse pointer is moved over it or when it becomes active. The result is a simple but impressive set of graphic menu controls that when clicked control navigation to other web pages.

Figure 8.18 demonstrates how things look when this document has been loaded into the browser and the mouse-pointer has been moved over the container for the Services link.

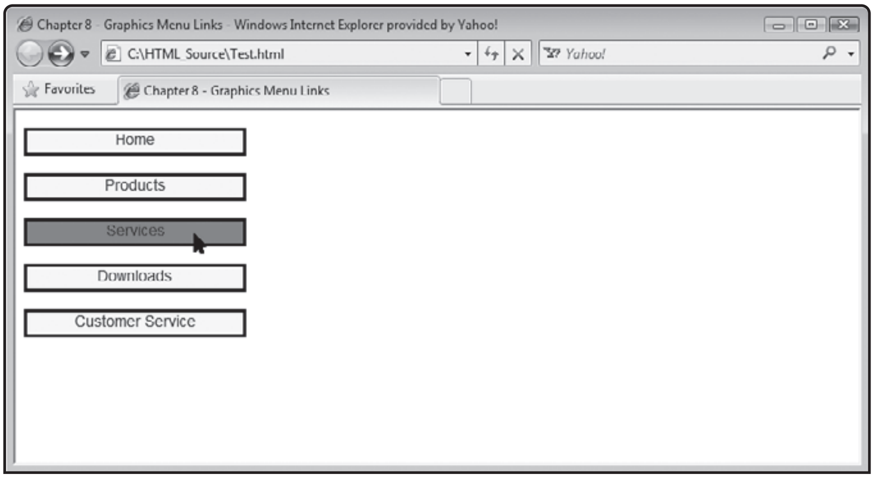

Chapter 8 - Graphics Menu Links - Windows Internet Explorer provided by Yahoo!
C:\HTML Source\Test.html
Yahoo!
Favorites
Chapter 8 - Graphics Menu Links
Home
Products
Services
Downloads
Customer Service

FIGURE 8.18

An example of how to use CSS to create basic graphic menu link controls.

If you want to get even fancier, you can use background images in place of background colors, as demonstrated in the following example.

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
  "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">

<html xmlns="http://www.w3.org/1999/xhtml" lang="en" xml:lang="en">

  <head>
    <meta http-equiv="Content-type" content="text/xhtml; charset=UTF-8" />
    <title>Chapter 8 - Graphic Button Links</title>
    <style type = "text/css">
      a {display: block;
         font: 14px Arial;
         text-decoration: none;
         text-align: center;
         width: 200px;
         height: 20px;
         /*background: yellow;*/
         background-image: url("yellow.png");
         color: blue;
      }
    a:focus, a:hover, a:active {
         /*background: red;*/
```

```
        background-image: url("red.png");
      }
    </style>
  </head>

  <body>
    <p><a href = "index.html">Home</a></p>
    <p><a href = "products.html">Products</a></p>
    <p><a href = "services.html">Services</a></p>
    <p><a href = "downloads.html">Downloads</a></p>
    <p><a href = "custserv.html">Customer Service</a></p>
  </body>

</html>
```

The differences between this example and the one before it are highlighted in bold. As you can see, instead of using the `background` property, this new version of the document uses the `background-image` property to display graphic files. This example makes use of two graphic image files. These two images contain a copy of an almost identical graphic button, the only difference being the color of the buttons. When loaded into the browser, CSS automatically centers and displays link text on the graphic button. Figure 8.19 demonstrates how this example looks when loaded into the browser.

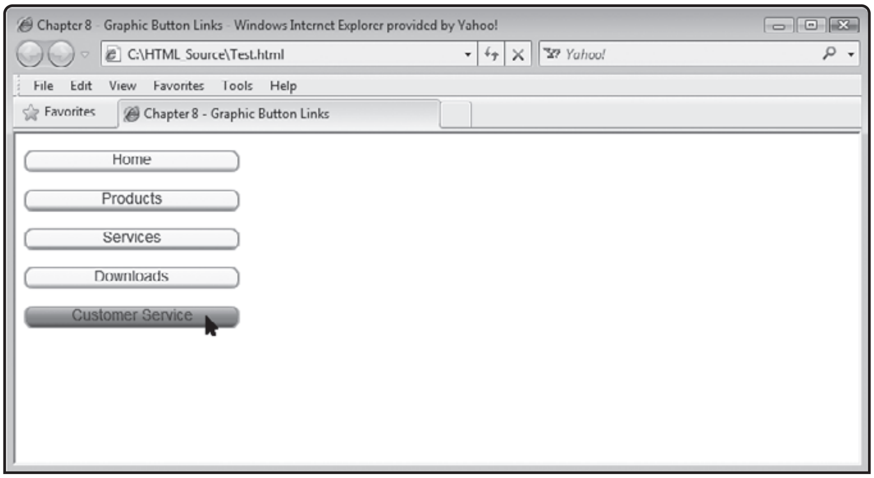

Chapter 8 - Graphic Button Links - Windows Internet Explorer provided by Yahoo!
C:\HTML_Source\Test.html
Yahoo!
File Edit View Favorites Tools Help
Favorites
Chapter 8 - Graphic Button Links
Home
Products
Services
Downloads
Customer Service

FIGURE 8.19

An example of how to use CSS to create fancy graphic menu link controls.

Using CSS to Better Integrate Text and Images

Another good use of CSS is to help better integrate and style the presentation of the flow of text and graphics on your web pages. Using CSS, you can, for example, wrap text around graphics in much the same way as is done in magazines and newspapers. CSS also makes easy work of adding graphics as backgrounds for your web pages.

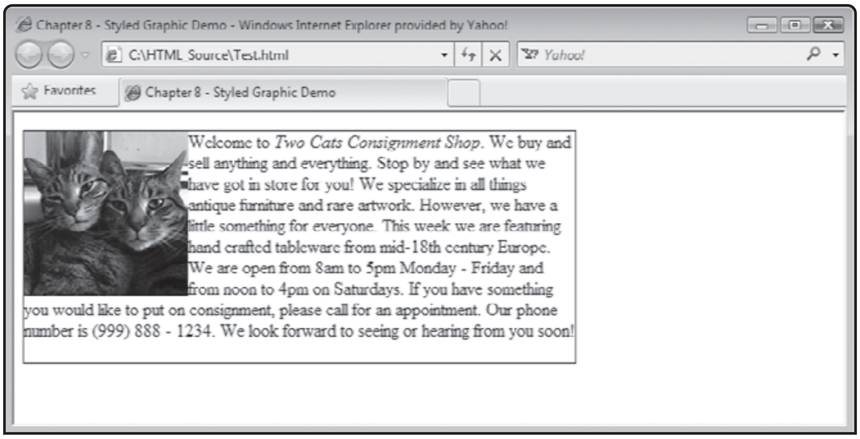
Wrapping Text Around Graphics

The trick to wrapping text around graphics lies in the application of the `float` property. An element that is floated, in this case a graphic, is shifted to the right or left, allowing text to flow around it. The `float` property supports the following values: `left`, `right`, and `none`. The following example demonstrates how to use the `float` property to flow text around the right side of a figure (as opposed to displaying it over or under the figure).

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
  "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">

<html xmlns="http://www.w3.org/1999/xhtml" lang="en" xml:lang="en">

  <head>
    <meta http-equiv="Content-type" content="text/xhtml; charset=UTF-8" />
    <title>Chapter 8 - Styled Graphic Demo</title>
    <style type = "text/css">
      p {
        border-width: thin;
        border-style: solid;
        width: 500px;
        height: 210px;
      }
      #logo {
        float: left;
      }
    </style>
  </head>

  <body>
    <p>
      <img id = "logo" src = "TwoCats.jpg" width = "149" height = "149"
        alt = "Two Cats Logo" />
      Welcome to <em>Two Cats Consignment Shop</em>. We buy and sell
      anything and everything. Stop by and see what we have got in
      store for you! We specialize in antique furniture and rare
      artwork. However, we have a little something for everyone. This week
      we are featuring hand-crafted tableware from mid-18th century Europe.
      We are open from 8am to 5pm Monday - Friday and from noon to 4pm on
      Saturdays. If you have something you would like to put on
      consignment, please call for an appointment. Our phone number is
      (999) 888 - 1234. We look forward to seeing or hearing from you soon!

    </p>
  </body>

</html>
```

If you look closely, you will see that the document's image `img` element has been assigned an ID of `Logo` and that a style rule has been used to apply the `float` property to the element. Note that the value of `float` has been set to `left` and that a paragraph rule has also been added that wraps up the paragraph and all its contents, including the `img` element, in a solid border.

Figure 8.20 shows how this example output is rendered by the browser.


Chapter 8 - Styled Graphic Demo - Windows Internet Explorer provided by Yahoo!
C:\HTML_Source\Test.html
Yahoo!
Favorites
Chapter 8 - Styled Graphic Demo

Welcome to *Two Cats Consignment Shop*. We buy and sell anything and everything. Stop by and see what we have got in store for you! We specialize in all things antique furniture and rare artwork. However, we have a little something for everyone. This week we are featuring hand crafted tableware from mid-18th century Europe. We are open from 8am to 5pm Monday - Friday and from noon to 4pm on Saturdays. If you have something you would like to put on consignment, please call for an appointment. Our phone number is (999) 888 - 1234. We look forward to seeing or hearing from you soon!

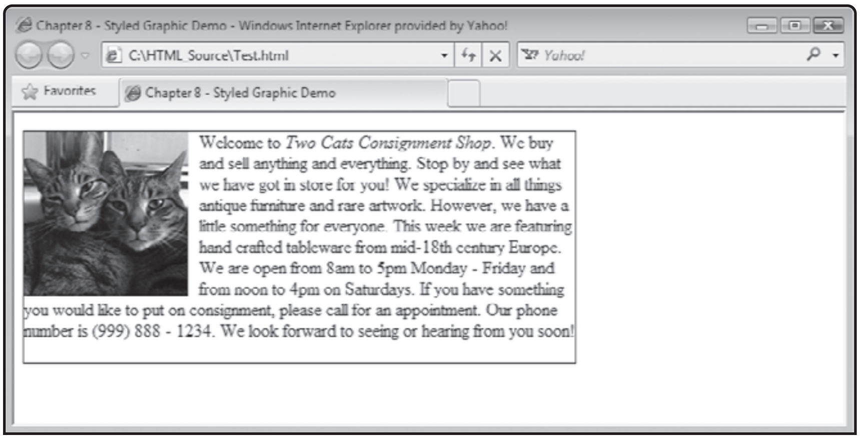
FIGURE 8.20

Using the `float` property to allow text to flow around the side of an image.

Note that while the text now wraps around the right side of the graphic, things look a little crowded because the text is so tightly placed up against the side of the graphic. However, this can easily be remedied by modifying the style rule and configuring the margin for the `img` element as demonstrated here.

```
<style type = "text/css">
  p {
    border-width: thin;
    border-style: solid;
    width: 500px;
    height: 210px;
  }
  #logo {
    float: left;
    margin-right: 10px;
    margin-bottom: 2px;
  }
</style>
```

As you can see, rather than use the `margin` property to set the margin for all four sides of the `img` element, the `margin-right` and `margin-bottom` properties were used to modify both the element's right and lower margins. Figure 8.21 shows how these changes to the style rule have affected the resulting web page.


Chapter 8 - Styled Graphic Demo - Windows Internet Explorer provided by Yahoo!
C:\HTML Source\Test.html
Yahoo!
Favorites
Chapter 8 - Styled Graphic Demo

Welcome to *Two Cats Consignment Shop*. We buy and sell anything and everything. Stop by and see what we have got in store for you! We specialize in all things antique furniture and rare artwork. However, we have a little something for everyone. This week we are featuring hand crafted tableware from mid-18th century Europe. We are open from 8am to 5pm Monday - Friday and from noon to 4pm on Saturdays. If you have something you would like to put on consignment, please call for an appointment. Our phone number is (999) 888 - 1234. We look forward to seeing or hearing from you soon!

FIGURE 8.21

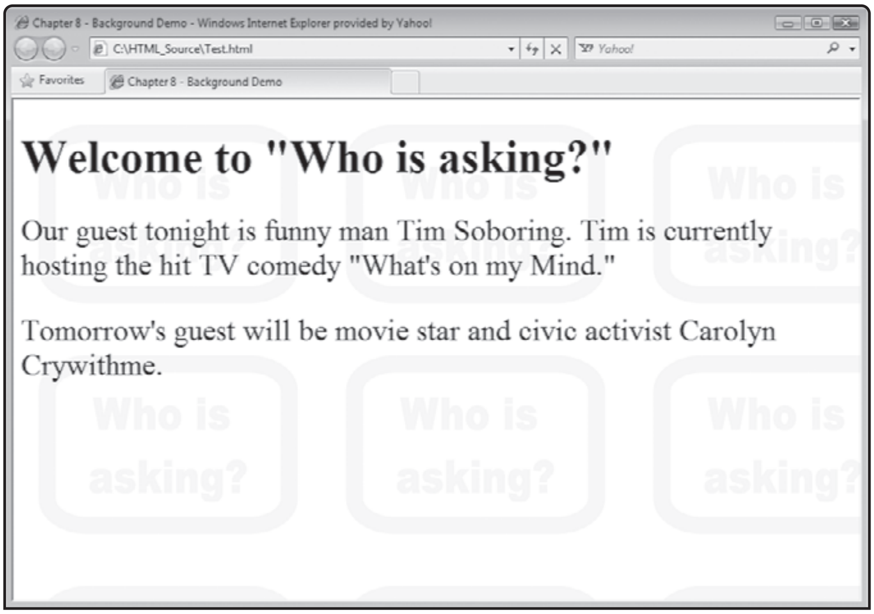
You can improve presentation by setting the image's right and bottom margins.

Adding a Background Image to Your Web Page

You have no doubt visited different websites that display graphic images as background on web pages. You can do the same thing using the `background-image` property. Of course, to be effective, the image you display must not overpower the content that you want to display. To use the `background-image` property, all you have to do is specify the URL of the graphic file you want to use, as demonstrated in the following example.

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
  "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">

<html xmlns="http://www.w3.org/1999/xhtml" lang="en" xml:lang="en">

  <head>
    <meta http-equiv="Content-type" content="text/xhtml; charset=UTF-8" />
    <title>Chapter 8 - Background Demo</title>
    <style type = "text/css">
      body {
        background-image: url(background.png);
      }
      h1 {
        font-size: 3pc;
      }
      p {
        font-size: 2pc;
      }
    </style>
  </head>

  <body>

    <h1>Welcome to "Who is asking?"</h1>

    <p>
      Our guest tonight is funny man Tim Soboring. Tim is currently hosting
      the hit TV comedy
      "What's on my Mind."
    </p>

    <p>
      Tomorrow's guest will be movie star and civic activist Carolyn
      Crywithme.
    </p>

  </body>

</html>
```

Here, an image file named background.png is used as the document's background, accomplished using the `background-image` property with a rule that styles the document's `body` element. Figure 8.22 shows an example of how this document looks when rendered by the browser.


Chapter 8 - Background Demo - Windows Internet Explorer provided by Yahoo!
C:\HTML_Source\Test.html
Yahoo!
Favorites
Chapter 8 - Background Demo

Welcome to "Who is asking?"

Our guest tonight is funny man Tim Soboring. Tim is currently hosting the hit TV comedy "What's on my Mind."

Tomorrow's guest will be movie star and civic activist Carolyn Crywithme.

FIGURE 8.22

An example of how to use CSS to add a background to your web pages.

Note that by default, the image is tiled, both horizontally and vertically, in the event the image is too small to fill the entire browser window. You can modify this behavior by adding the `background-repeat` property to the `body` element's style rule. This property supports the following values.

- **repeat-x.** Tiles the image horizontally.
- **repeat-y.** Tiles the image vertically.
- **no-repeat**. Disables image tiling.

STYLING YOUR TABLES

CSS provides extensive control over the presentation of tables. You can create CSS rules that add table borders, pad table cells, collapse individual cell borders, add background color, set border color, and control text alignment.

In Chapter 6, you learned how to add borders to your tables using the `table` element's `border` attribute. Though effective, it is better to use CSS's `border-style` property to add borders to your tables. Not only is using the `border-style` property considered better form, but it also provides the ability to specify any of the following values:

- dashed
- dotted
- double
- groove
- hidden
- inset
- none
- outset
- solid

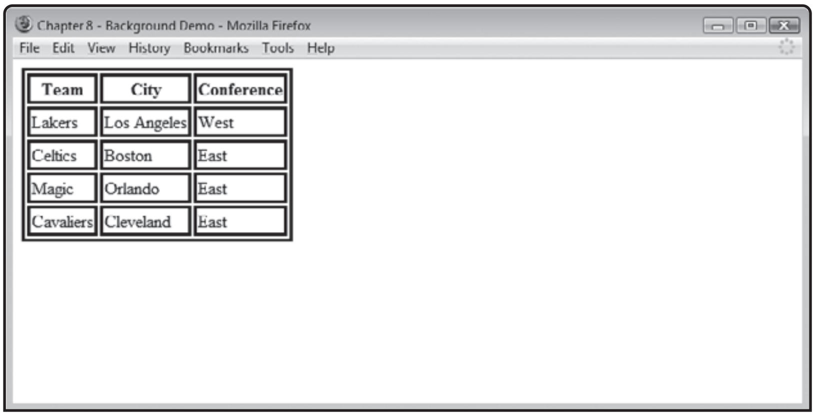
The following document provides an example of how to set a table's border using the `border-style` property.

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
  "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">

<html xmlns="http://www.w3.org/1999/xhtml" lang="en" xml:lang="en">

  <head>
    <meta http-equiv="Content-type" content="text/xhtml; charset=UTF-8" />
    <title>Chapter 8 - Background Demo</title>
    <style type = "text/css">
      table {
        border-style: solid;
      }
      th, td {
        border-style: solid;
      }
    </style>
  </head>

  <body>
```

```
    <table>
      <tr>
        <th scope = "col">Team</th>
        <th scope = "col">City</th>
        <th scope = "col">Conference</th>
      </tr>
      <tr>
        <td>Lakers</td>
        <td>Los Angeles</td>
        <td>West</td>
      </tr>
      <tr>
        <td>Celtics</td>
        <td>Boston</td>
        <td>East</td>
      </tr>
      <tr>
        <td>Magic</td>
        <td>Orlando</td>
        <td>East</td>
      </tr>
      <tr>
        <td>Cavaliers</td>
        <td>Cleveland</td>
        <td>East</td>
      </tr>
    </table>

  </body>

</html>
```

Figure 8.23 shows how the table produced by this example looks when rendered by the browser.

By default, the browser allocated only enough space to a row or column to accommodate its largest cell entry. You can often make a table look better by using the padding property to add a little extra white space to its cells. To demonstrate how this works, let's modify the embedded style sheet from the previous example as shown here:

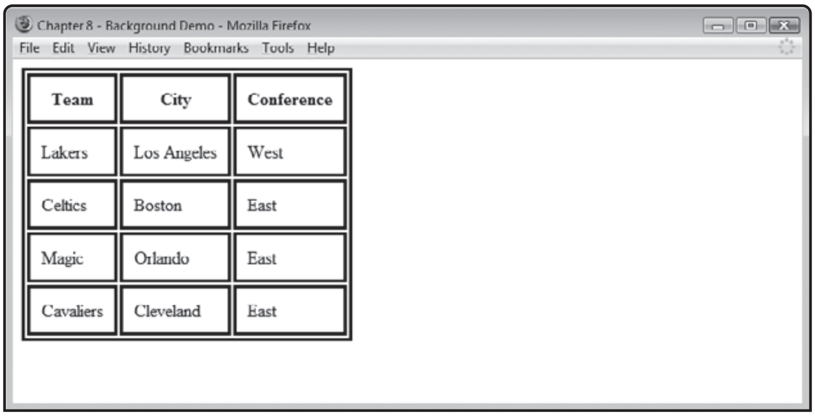

Chapter 8 - Background Demo - Mozilla Firefox
File Edit View History Bookmarks Tools Help

Team	City	Conference
Lakers	Los Angeles	West
Celtics	Boston	East
Magic	Orlando	East
Cavaliers	Cleveland	East

FIGURE 8.23
Using CSS to add a border to your table.

```
<style type = "text/css">
  table {
    border-style: solid;
  }
  th, td {
    border-style: solid;
    padding: 10px
  }
</style>
```

Figure 8.24 shows how the table looks once it has been rendered using the updated style rule.


Chapter 8 - Background Demo - Mozilla Firefox
File Edit View History Bookmarks Tools Help

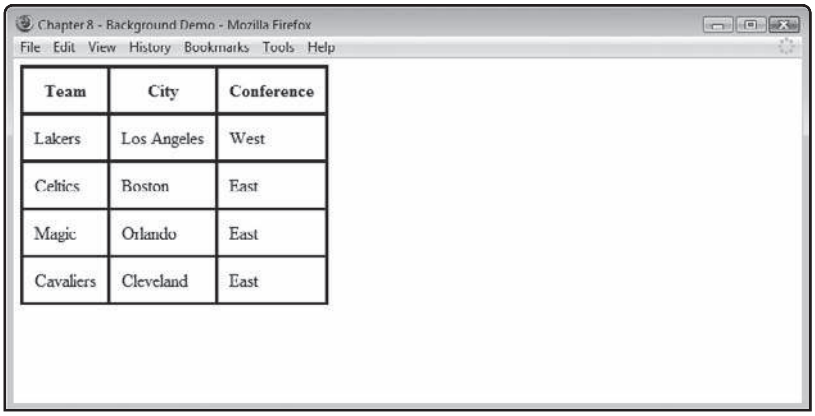
Team	City	Conference
Lakers	Los Angeles	West
Celtics	Boston	East
Magic	Orlando	East
Cavaliers	Cleveland	East

FIGURE 8.24
Using CSS to add a little padding to table cells.

If you look closely at Figure 8.24, you will notice that every cell in the table has its own individual border. If you want, you can modify the `table` element's style rule by adding the `border-collapse` property to it (with a value of `collapse`) to disable this presentation effect, as demonstrated here:

```
<style type = "text/css">
  table {
    border-style: solid;
    border-collapse: collapse;
  }
  th, td {
    border-style: solid;
    padding: 10px
  }
</style>
```

Figure 8.25 shows the effect that this rule change has on the table.


Chapter 8 - Background Demo - Mozilla Firefox
File Edit View History Bookmarks Tools Help

Team	City	Conference
Lakers	Los Angeles	West
Celtics	Boston	East
Magic	Orlando	East
Cavaliers	Cleveland	East

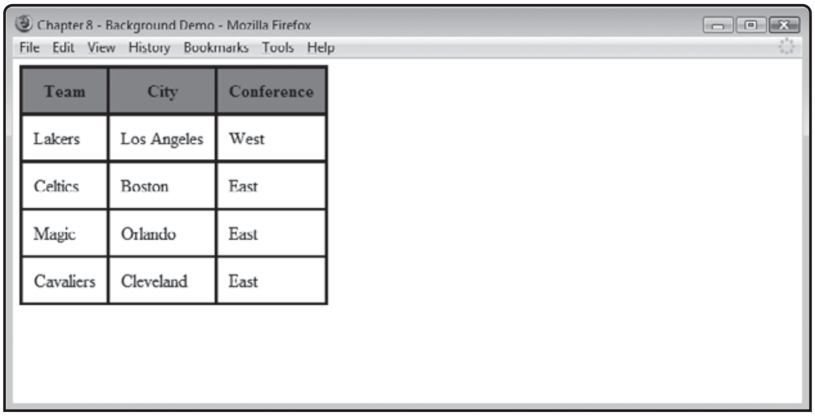
FIGURE 8.25

Using CSS to collapse individual cell borders.

Often table headings are highlighted by assigning a background color to them. This makes them stand out and helps make table data easier to scan. To add background color to cells in your tables, all you have to do is add the `background-color` property to the style rule for the appropriate table elements, as demonstrated here:

```
<style type = "text/css">
  table {
    border-style: solid;
    border-collapse: collapse;
  }
  th, td {
    border-style: solid;
    padding: 10px
  }
  th {
    background-color: gray;
  }
</style>
```

Figure 8.26 shows the effect that this rule change has on the table.


Chapter 8 - Background Demo - Mozilla Firefox
File Edit View History Bookmarks Tools Help

Team	City	Conference
Lakers	Los Angeles	West
Celtics	Boston	East
Magic	Orlando	East
Cavaliers	Cleveland	East

FIGURE 8.26
Using CSS to add background color to your tables.

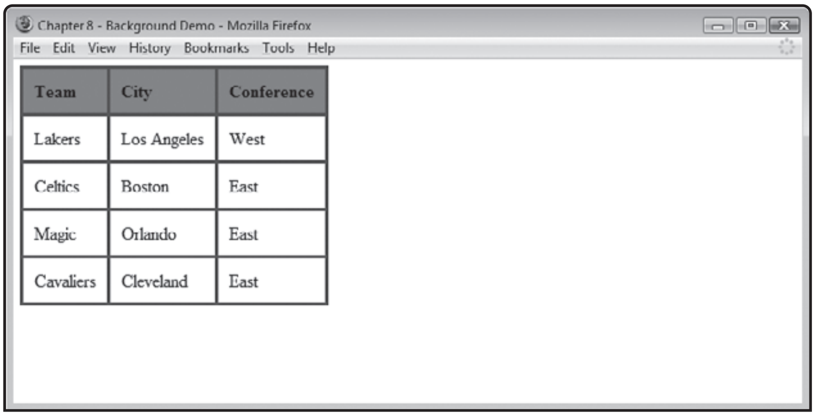
If you want, you can even specify the color of your tables by assigning a color to them, as demonstrated here:

```
<style type = "text/css">
  table {
    border-style: solid;
    border-collapse: collapse;
  }
  th, td {
    border-style: solid;
    padding: 10px;
    border-color: blue;
  }
  th {
    background-color: gray;
  }
</style>
```

You can even use CSS to control the alignment of text within labels using the `text-align` property, assigning it a value of `left`, `right`, `center`, or `justify`. To demonstrate how to work with this property, let's modify the previous style sheet as shown here:

```
<style type = "text/css">
  table {
    border-style: solid;
    border-collapse: collapse;
  }
  th, td {
    border-style: solid;
    padding: 10px;
    border-color: blue;
  }
  th {
    background-color: gray;
    text-align: left
  }
</style>
```

Figure 8.27 shows the effect that this style modification has on the table.


Chapter 8 - Background Demo - Mozilla Firefox
File Edit View History Bookmarks Tools Help

Team	City	Conference
Lakers	Los Angeles	West
Celtics	Boston	East
Magic	Orlando	East
Cavaliers	Cleveland	East

FIGURE 8.27

Using CSS to control cell alignment.

Styling Your Forms

One final area where we will look at applying CSS is in the modification of form presentation. Forms are challenging to style because every browser displays them differently. Form elements are closely tied to the operating system. To try to provide users with a consistent environment in which to work, browsers rely on the operating system to render form controls. As such, form appearance is heavily affected by the operating system's native presentation scheme. This scheme not only varies from operating system to operating system, but it also often varies between different versions of the same operating system. For example, Windows Vista's controls have a distinct look and feel from those provided by Windows XP.

Given the above restrictions, it is all but impossible to create forms that look exactly the same on every computer that loads your web documents. As such, it's often best to allow the browser to render forms using its default style. This brings with it the benefit that your visitors will see your form presented in a manner consistent with their experience and expectations. So, while you should avoid tinkering around with your form's controls, there is often plenty of value in modifying the presentation of the other elements on your forms, including headings, paragraphs, labels, and fieldsets.

As an example of how you might go about modifying a form's appearance, look at the following, which you might recognize from Chapter 6.

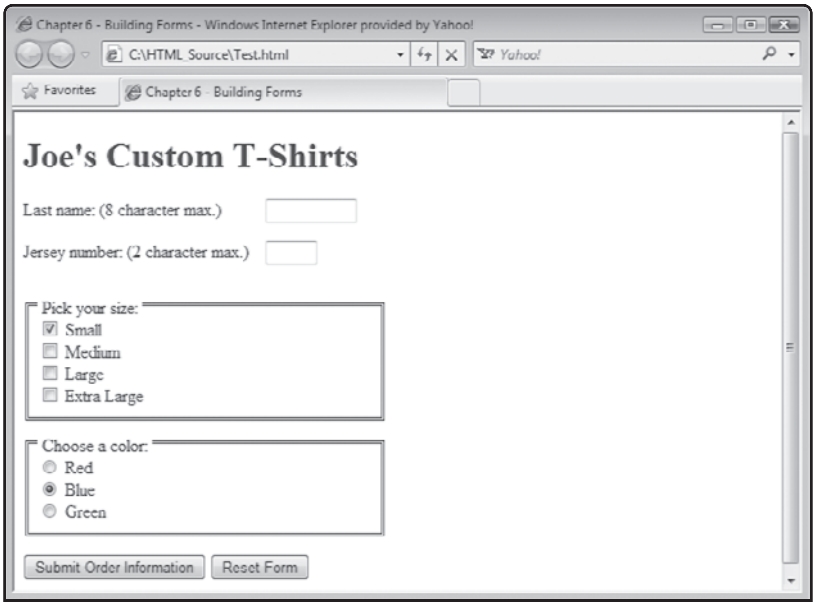
```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
  "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">

<html xmlns="http://www.w3.org/1999/xhtml" lang="en" xml:lang="en">

  <head>
    <meta http-equiv="Content-type" content="text/xhtml; charset=UTF-8" />
    <title>Chapter 6 - Building Forms</title>
    <style type = "text/css">
      body {
        color: blue;
      }
      fieldset {
        padding: 10px;
        max-width: 300px;
        border: double;
      }
      .textbox {
```

```
      float: left;
      width: 220px;
    }
  </style>
</head>

<body>

  <h1>Joe's Custom T-Shirts</h1>

  <form action = "/cgi-bin/processdata.cgi" method = "post">

    <p>
      <label class = "textbox" for = "name">Last name: (8 character
        max.)</label>
      <input type = "text" id = "name" name = "name" size = "8"
        maxlength = "8" />
    </p>

    <p>
      <label class = "textbox" for = "number">Jersey number: (2
        character max.)</label>
      <input type = "text" id = "number" name = "number" size = "2"
        maxlength = "2" />
    </p>

    <fieldset>
      <legend>Pick your size:</legend>
      <input type = "checkbox" id = "checkbox1" name = "checkbox1"
        value = "Small" checked = "checked" />
      <label for = "checkbox1">Small</label><br />
      <input type = "checkbox" id = "checkbox2" name = "checkbox2"
        value = "Medium" />
      <label for = "checkbox2">Medium</label><br />
      <input type = "checkbox" id = "checkbox3" name = "checkbox3"
        value = "Large"/>
      <label for = "checkbox3">Large</label><br />
      <input type = "checkbox" id = "checkbox4" name = "checkbox4"
```

```
        value = "Extra Large" />
      <label for = "checkbox4">Extra Large</label>
    </fieldset>

    <fieldset>
      <legend>Choose a color:</legend>
      <input type = "radio" id = "radio1" name = "radio"
        value = "Red" />
      <label for = "radio1">Red</label><br />
      <input type = "radio" id = "radio2" name = "radio"
        value = "Blue" checked = "checked" />
      <label for = "radio2">Blue</label><br />
      <input type = "radio" id = "radio3" name = "radio"
        value = "Green" />
      <label for = "radio3">Green</label><br />
    </fieldset>

    <p>
      <input type = "submit" id = "submit_button" name = "submit_button"
        value = "Submit Order Information" />
      <input type = "reset" id = "reset_button" name = "reset_button"
        value = "Reset Form" />
    </p>

  </form>

 </body>

</html>
```

If you look at this example's style rules, you will see that the first rule configures the display of the form's text in blue. In addition, a rule has been set up that modifies the appearance of the form's `fieldset` element in several ways. For starters, a little extra padding has been added to keep things from being too bunched up. Second, the width of the `fieldset` element is set to 300 pixels, overriding the default behavior of making the `fieldset` span the width of the entire browser window. Last, the `fieldset` element's border has been set to `double` just to spice things up a bit. The final style rule affects the presentation of the two labels assigned to the `textbox` class. The rule makes the labels float to the left and sets their width to 220 pixels.

As a result, the text control associated with the label becomes vertically aligned, as shown in Figure 8.28.


Chapter 6 - Building Forms - Windows Internet Explorer provided by Yahoo!
C:\HTML_Source\Test.html
Yahoo!
Favorites
Chapter 6 - Building Forms

Joe's Custom T-Shirts

Last name: (8 character max.)

Jersey number: (2 character max.)

Pick your size:
- [✓] Small
- Medium
- Large
- Extra Large

Choose a color:
- Red
- [✓] Blue
- Green

Submit Order Information | Reset Form

FIGURE 8.28

Using CSS to customize form presentation.

Styling Based on Output Device

Depending on your target audience, you may find that you need to configure CSS to present your content differently. For example, if your web content is targeted at an audience that is visually disabled, you might want to make sure your style is directed towards braille output devices. To define the type of media you want to target, you need to include the `style` element's optional `media` attribute. This attribute specifies the type of media your content should be styled for. The following list of values can be assigned to the `media` attribute.

- all
- aural
- braille
- handheld
- print
- projection
- screen

- tty
- tv

If you want, you can use the `@media` rule, which will allow you to specify more than one media type, as demonstrated in the following example.

```
<style type = "text/css">
  @media print {
    body { font-size: 12pt; }
  }
  @media screen {
    body { font-size: 14px; }
  }
  @media screen, print {
    body { color: blue; }
  }
</style>
```

Here, a font size of 12 points was specified for the text when printed. However, a font size of 14 points was specified for text when displayed in the browser window. In addition, the third rule has configured the use of blue when printing and displaying text.

Back to the Fortune Teller Game

All right, now it is time to return your attention to this chapter's project, the Fortune Teller game. This game simulates a session with a Fortune Teller, allowing the player to enter any number of questions. Answers to the player's questions are randomly generated and dynamically displayed in the browser window. To make the applications more appealing, an external style sheet will be used to improve various aspects of its presentation.

Designing the Application

To help make things easier to digest, this web project will be completed in a series of steps, as outlined here:

1. Create a new XHTML document.
2. Develop the document's markup.
3. Add meta and title elements.
4. Specify document content.
5. Create the document's script.
6. Create an external style sheet.
7. Load and test the Fortune Teller game.

Step 1: Creating a New XHTML Document

The first step in the development of the Fortune Teller game is to create a new web document. Do so using your preferred code or text editor. Save the document as a plain text file named FortuneTeller.html. This web document will make use of CSS style rules. Therefore, you will need to create a second file named ft.css.

Step 2: Developing the Document's Markup

The next step in the development of this project is to assemble the document's markup. To do so, add the following elements to the FortuneTeller.html file.

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
  "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">

<html xmlns="http://www.w3.org/1999/xhtml" lang="en" xml:lang="en">

  <head>

  </head>

  <body>

  </body>

</html>
```

Step 3: Adding meta and title Elements

Next, add the following elements to the document's `head` section.

```
<meta http-equiv="Content-type" content="text/xhtml; charset=UTF-8" />
<title>The Fortune Teller Game</title>
```

As previously stated, this web application will make use of an external style sheet named ft.css. To set this up, add the following link to the external style sheet to the end of the `head` section.

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

Step 4: Specifying Document Content

Now it is time to work on laying out this document's markup. Begin by adding the following elements to the document's `body` section.

```
<div>
  <img src = "title.png" width = "550px" height = "104px" alt = "Game Logo" />
</div>
```

As you can see, these elements place an image inside a `div` element. The `img` element is used to display a graphic showing the game's logo, which is displayed at the top of the browser window. The rest of the document's markup is made up of a form and its elements.

Note that the `form` element's `action` attribute has been set to the web page itself and not to a server-side form handler. Also, note that the form has no Submit button (and thus no need for a form handler). The form has two `input` elements, one with a text control and one for a button control. Both elements are assigned a unique ID. In addition, the `button` element ends with a JavaScript statement that uses the `onClick()` event handler to execute a JavaScript function named `AnswerQuestion()`.

The game will dynamically display text that shows the Fortune Teller answers each time a new question is asked. To facilitate the display of this information a `span` element has been added to the end of the form and assigned an ID of `answer`.

```
<form action = "FortuneTeller.html">

  <p>
    Instructions:
  </p>
  <p>
    To play this game all you have to do is type in a question that
    you would like the Fortune Teller to answer. Make sure that you
    phrase your question so that it can be answered with Yes/No styled
    responses. Once you have finished entering your question, click on
    the <em>Get Answer</em> button to see what the Fortune Teller has
    to say.
  </p>

  <p>
    <input type = "text" size = "68px" id = "inputField" />

    <input type = "button" value = "Get Answer" id = "checkBtn"
      Onclick = "AnswerQuestion()" />
  </p>
  <p><span id = "answer"> </span></p>

</form>
```

By assigning the `span` element a unique ID, you enable the application's JavaScript statements to dynamically update its content using the DOM.

Step 5: Creating the Document's Script

Now that the document's markup is complete, it is time to lay out its JavaScript code. Begin by adding the following statements to the document's `head` section.

```
<script type = "text/javascript">
<!-- Start hiding JavaScript statements

// End hiding JavaScript statements -->
</script>
```

These statements provide the markup needed to support the definition of the script. The JavaScript itself consists of two functions named `AnswerQuestion()` and `ResetScreen()`. The `AnswerQuestion()` function is executed whenever the player clicks on the game's button control. To create this function, embed the following statements inside the `script` element's opening `<script>` and closing `</script>` tags:

```
function AnswerQuestion() {
  var checkButton = document.getElementById("checkBtn");

  if (document.getElementById('inputField').value == "") {
    window.alert("You did not submit a question. Try again.");
  } else {

    randomNo = 1 + Math.random() * 9;
    randomNo = Math.round(randomNo);

    switch (randomNo) {
    case 1:
      document.getElementById('answer').innerHTML = "Yes!";
      break;
    case 2:
      document.getElementById('answer').innerHTML = "No.";
      break;
    case 3:
      document.getElementById('answer').innerHTML = "Maybe.";
      break;
    case 4:
      document.getElementById('answer').innerHTML = "Doubtful.";
      break;
    case 5:
      document.getElementById('answer').innerHTML =
        "Not in this lifetime.";
      break;
    case 6:
      document.getElementById('answer').innerHTML =
        "The answer is unclear.";
      break;
    case 7:
      document.getElementById('answer').innerHTML =
        "Ask this question again later.";
      break;
    case 8:
      document.getElementById('answer').innerHTML =
        "Today is your lucky day... Yes!";
      break;
    case 9:
      document.getElementById('answer').innerHTML =
        "Sorry but the answer is no.";
      break;
    case 10:
      document.getElementById('answer').innerHTML = "No way!";
      break;
    }
    setTimeout("ResetScreen()", 3000)

  }
}
```

The `AnswerQuestion()` function begins by declaring a variable named `checkButton`, which is used to set up a reference to the document's button control. Next, a check is made to ensure that the user actually typed a question before clicking on the button control. If not, an error message is displayed in a popup dialog window. If a question was submitted, a random number from 1 to 10 is generated and stored in a variable named `randomNo`. A `switch` statement code block is then used to compare the value stored in `randomNo` against ten `case` statements, each of which is assigned a value from 1 to 10. When a match is found, that `case` statement is executed. The first of these statements used dot notation and the DOM to assign (display) a text string inside the document's `span` element. This element assigned an ID of `answer`. The assignment is made using the `document` object's `getElementById()` method along with the `innerHTML` property. As soon as the assignment is made, a `break` statement is executed, terminating the execution of the rest of the `switch` statement code block.

The JavaScript `SetTimeout()` function is then executed. This function accepts two arguments, the name of a JavaScript to execute and a numeric value specifying how many milliseconds to pause before the specified function is executed (300 milliseconds equals 3 seconds).

So, after a three-second pause, the `ResetScreen()` function, shown below, is executed. This function uses dot notation and the DOM to clear out the player's questions and the game's answer in order to ready the game for a another question.

```
function ResetScreen() {
  document.getElementById('inputField').value = "";
  document.getElementById('answer').innerHTML = "";
}
```

Step 6: Creating an External Style Sheet

The FortuneTeller.html document is styled using an external style sheet named ft.css. The rules stored in this style sheet are shown here:

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

div {
  border-width: thin;
  border-style: solid;
  width: 550px;
  height: 104px;
  }

p {font-family: Arial;
  color: green;
  font-size: 1pc; }

span {
  font-size: 1.5pc }
  font-weight: bold;
  color: green;
  }
```

The first rule explicitly assigns white as the web page's background color. The second rule formats the document's `div` element, assigning it a thin, solid border that is 550 pixels wide and 104 pixels high (e.g., the exact dimension of the `img` element embedded inside the `div` element). The third rule assigns the font type, color, and size for all text stored in the document's paragraph elements. The last rule assigns the font size, height, and color of the text displayed inside the document's `span` element, making it a little larger than that of its paragraph text.

Step 7: Loading and Testing the Fortune Teller Game

Assuming you have followed along carefully, your copy of the FortuneTeller.html document should be complete. To make sure you have assembled it correctly, look at the following example, which shows a complete copy of the finished document.

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
  "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">

<html xmlns="http://www.w3.org/1999/xhtml" lang="en" xml:lang="en">

  <head>
    <meta http-equiv="Content-type" content="text/xhtml; charset=UTF-8" />
    <title>The Fortune Teller Game</title>
    <link href = "ft.css" type = "text/css" rel = "stylesheet" />

    <script type = "text/javascript">
    <!-- Start hiding JavaScript statements

      function AnswerQuestion() {
        var checkButton = document.getElementById("checkBtn");

        if (document.getElementById('inputField').value == "") {
          window.alert("You did not submit a question. Try again.");
```

```
} else {

  randomNo = 1 + Math.random() * 9;
  randomNo = Math.round(randomNo);

  switch (randomNo) {
  case 1:
    document.getElementById('answer').innerHTML = "Yes!";
    break;
  case 2:
    document.getElementById('answer').innerHTML = "No.";
    break;
  case 3:
    document.getElementById('answer').innerHTML = "Maybe.";
    break;
  case 4:
    document.getElementById('answer').innerHTML = "Doubtful.";
    break;
  case 5:
    document.getElementById('answer').innerHTML =
      "Not in this lifetime.";
    break;
  case 6:
    document.getElementById('answer').innerHTML =
      "The answer is unclear.";
    break;
  case 7:
    document.getElementById('answer').innerHTML =
      "Ask this question again later.";
    break;
  case 8:
    document.getElementById('answer').innerHTML =
      "Today is your lucky day... Yes!";
    break;
  case 9:
    document.getElementById('answer').innerHTML =
      "Sorry but the answer is no.";
    break;
```

```
        case 10:
          document.getElementById('answer').innerHTML = "No way!";
          break;
        }
        setTimeout("ResetScreen()", 3000)

      }
    }

    function ResetScreen() {
      document.getElementById('inputField').value = "";
      document.getElementById('answer').innerHTML = "";
    }

  // End hiding JavaScript statements -->
  </script>

</head>

<body>

  <div>
    <img src = "title.png" width = "550 height = "104"
     alt = "Game Logo" />
  </div>

  <form action = "FortuneTeller.html">

    <p>
      Instructions:
    </p>
    <p>
      To play this game all you have to do is type in a question that
      you would like the Fortune Teller to answer. Make sure that you
      phrase your question so that it can be answered with Yes/No styled
      responses. Once you have finished entering your question, click on
      the <em>Get Answer</em> button to see what the Fortune Teller has
      to say.
      </p>

      <p>
        <input type = "textfield" size = "68" id = "inputField">

        <INPUT type = "button" value = "Get Answer" id = "checkBtn"
          onclick= AnswerQuestion()>
      </p>

      <p><span id = "answer"> </span></p>

    </form>

  </body>

</html>
```

If you have not already done so, save your document and load it into your web browser to see how things have turned out.

HINT A complete copy of the source code for this project, including its style sheet and the graphics needed to create its graphic controls is available on the book's companion web page, located at www.courseptr.com/downloads.

Summary

This chapter rounded out your understanding of cascading style sheets by demonstrating its use in a variety of circumstances. This included learning how to use CSS to control the configuration of containers and including their placement, border, margins, and padding. You learned how to create CSS rules that modify the display of list markers and to create rollover links, controls, and buttons. This chapter also showed you how to use CSS to better integrate the presentation of text and images. On top of all this, you learned how to influence the presentation of forms and tables. This chapter wrapped things up by walking you through the creation of the Fortune Teller game.

Now, before you move on to Chapter 10, take a few minutes and enhance the Fortune Teller game by implementing the following challenges.

Challenges

1. As currently designed, the Fortune Teller Game displays one of ten randomly selected answers to player questions. However, it does not take long for the player to realize the limited range of answers. Make things more interesting by increasing the number of available answers.
2. As written, the Fortune Teller Game left justifies its graphic, text, and form. As a result, things look a little out of place if a user fully expands the browser window. To fix this, use CSS to center the display of all content.