

# Introduction to Cascading Style Sheets (CSS) Part 2

# 5.1 Introduction

- In this chapter, we introduce many features new to CSS3 (see the Objectives)
  - These capabilities are being built into the browsers, resulting in faster and more economical web development and better client-side performance
  - This reduces the need for JavaScript libraries and sophisticated graphics software packages such as Adobe Photoshop, Adobe Illustrator, Corel PaintShop Pro and Gimp to create interesting effects

## 5.2 Text Shadows

- The CSS3 text-shadow property makes it easy to add a text shadow effect to any text
- The text-shadow property has four values
  - Horizontal offset of the shadow
    - The number of pixels that the text-shadow will appear to the left or the right of the text
    - A negative value moves the text-shadow to the left
    - A positive value moves it to the right
  - Vertical offset of the shadow
    - The number of pixels that the text-shadow will be shifted up or down from the text
    - A negative value moves the shadow up
    - A positive value moves it down

## 5.2 Text Shadows (cont.)

- Blur radius
  - The blur (in pixels) of the shadow
  - A blur-radius of 0px would result in a shadow with a sharp edge = no blur
  - The greater the value, the greater the blurring of the edges
- Color
  - Determines the color of the text-shadow

```
1 <!DOCTYPE html>
2
3 <!-- Fig. 5.1: textshadow.html -->
4 <!-- Text shadow in CSS3. -->
5 <html>
6   <head>
7     <meta charset = "utf-8">
8     <title>Text Shadow</title>
9     <style type = "text/css">
10    h1
11    {
12      text-shadow: -4px 4px 6px dimgrey; /* add shadow */
13      font-size: 400%; /* increasing the font size */
14    }
15  </style>
16 </head>
17 <body>
18   <h1>Text Shadow</h1>
19 </body>
20 </html>
```

**Fig. 5.1** | Text shadow in CSS3. (Part I of 2.)



**Fig. 5.1** | Text shadow in CSS3. (Part 2 of 2.)

## 5.3 Rounded Corners

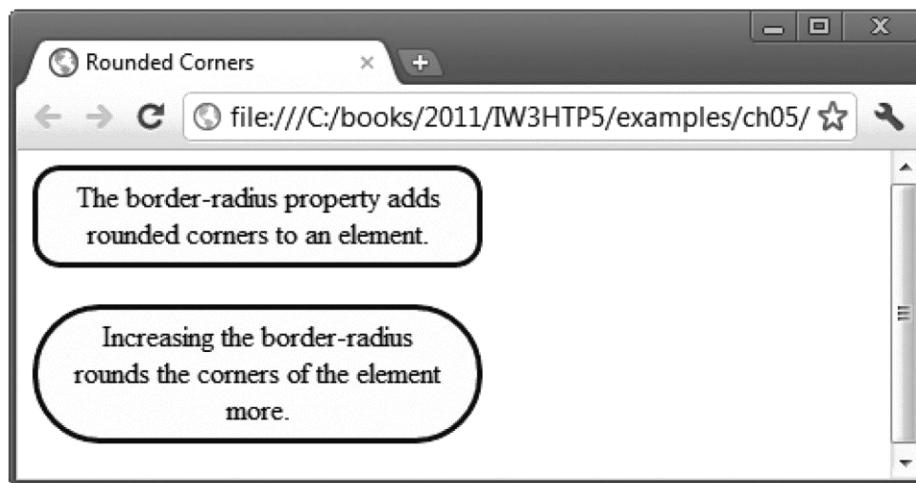
- The border-radius property allows you to add rounded corners to an element
- For the first rectangle, we set the border-radius to 15px
- This adds slightly rounded corners to the rectangle
- For the second rectangle, we increase the border-radius to 50px, making the left and right sides completely round
- Any border-radius value greater than half of the shortest side length produces a completely round end
- You can also specify the radius for each corner with border-top-left-radius, border-top-right-radius, border-bottom-left-radius and border-bottom-right-radius

```
1 <!DOCTYPE html>
2
3 <!-- Fig. 5.2: roundedcorners.html -->
4 <!-- Using border-radius to add rounded corners to two elements. -->
5 <html>
6   <head>
7     <meta charset = "utf-8">
8     <title>Rounded Corners</title>
9     <style type = "text/css">
10    div
11    {
12      border: 3px solid navy;
13      padding: 5px 20px;
14      background: lightcyan;
15      width: 200px;
16      text-align: center;
17      border-radius: 15px; /* adding rounded corners */
18      margin-bottom: 20px;
19    }
```

**Fig. 5.2** | Using border-radius to add rounded corners to two elements. (Part I of 3.)

```
20      #round2
21  {
22      border: 3px solid navy;
23      padding: 5px 20px;
24      background: lightcyan;
25      width: 200px;
26      text-align: center;
27      border-radius: 50px; /* increasing border-radius */
28  }
29  </style>
30 </head>
31 <body>
32     <div>The border-radius property adds rounded corners
33         to an element.</div>
34     <div id = "round2">Increasing the border-radius rounds the corners
35         of the element more.</div>
36 </body>
37 </html>
```

**Fig. 5.2 |** Using border-radius to add rounded corners to two elements. (Part 2 of 3.)



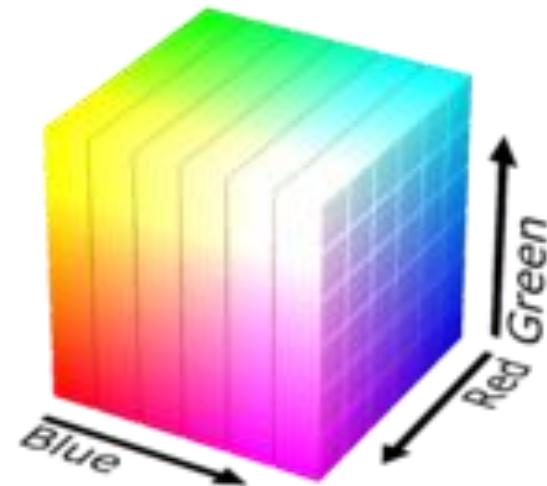
**Fig. 5.2** | Using border-radius to add rounded corners to two elements. (Part 3 of 3.)

## 5.4 Color

- CSS3 allows you to express color in several ways in addition to standard color names (such as Aqua) or hexadecimal RGB values (such as #00FFFF for Aqua)
- RGB (Red, Green, Blue) or RGBA (Red, Green, Blue, Alpha) gives you greater control over the exact colors in your web pages
  - The value for each color—red, green and blue
    - Range from 0 to 255
  - The alpha value—which represents opacity
    - Any value in the range 0.0 (fully transparent) through 1.0 (fully opaque)

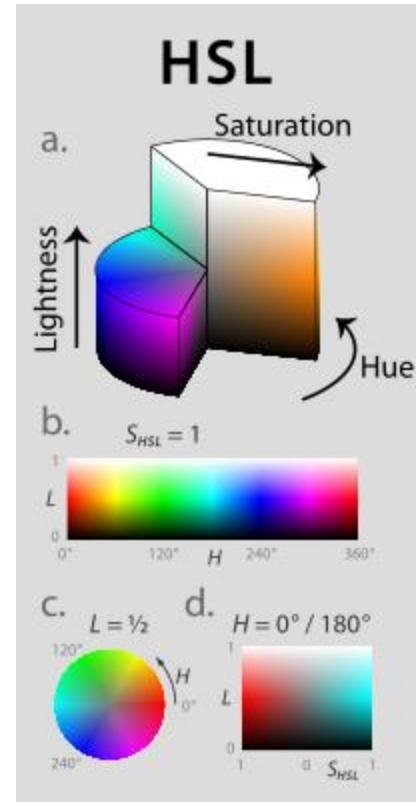
## 5.4 Color (cont.)

- If you were to set the background color as follows
  - **background: rgba(255, 0, 0, 0.5);**
- the resulting color would be a half-opaque red
- There are over 140 HTML color names, whereas there are 16,777,216 different RGB colors ( $256 \times 256 \times 256$ ) and varying opacities of each



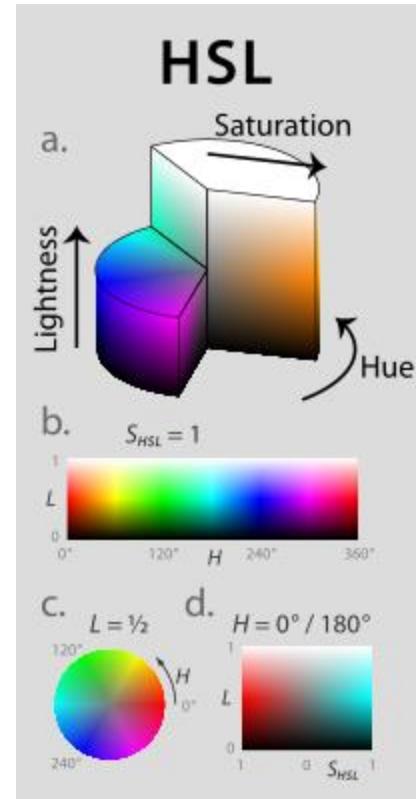
## 5.4 Color (cont.)

- CSS3 also allows you to express color using
  - HSL (hue, saturation, lightness)
  - HSLA (hue, saturation, lightness, alpha) values
- The hue is a color or shade expressed as a value from 0 to 359 representing the degrees on a color wheel (a wheel is 360 degrees)
- The colors on the wheel progress in the order of the colors of the rainbow—red, orange, yellow, green, blue, indigo and violet



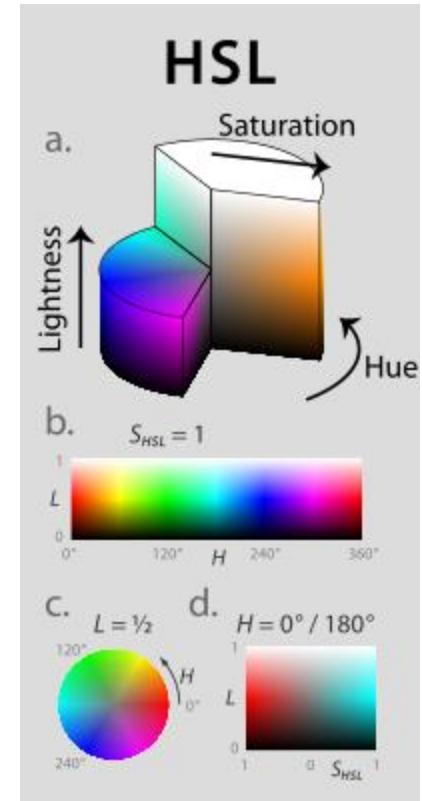
## 5.4 Color (cont.)

- The value for red, which is at the beginning of the wheel, is 0
- Green hues have values around 120 and blue hues have values around 240
- A hue value of 359, which is just left of 0 on the wheel, would result in a red hue
- The saturation
  - The intensity of the hue
    - Expressed as a percentage
      - 100% is fully saturated (the full color)
      - 0% is gray



## 5.4 Color (cont.)

- Lightness
  - The intensity of light or luminance of the hue is also expressed as a percentage
  - A lightness of 50% is the actual hue
  - If you decrease the amount of light to 0%, the color appears completely dark (black)
  - If you increase the amount of light to 100%, the color appears completely light (white)
  - For example, if you wanted to use an **hsla** value to get the same color red as in our example of an **rgba** value, you would set the background property as follows:
    - **background: hsla(0, 100%, 50%, 0.5);**      **rgba(255, 0, 0, 0.5);**



## 5.5 Box Shadows

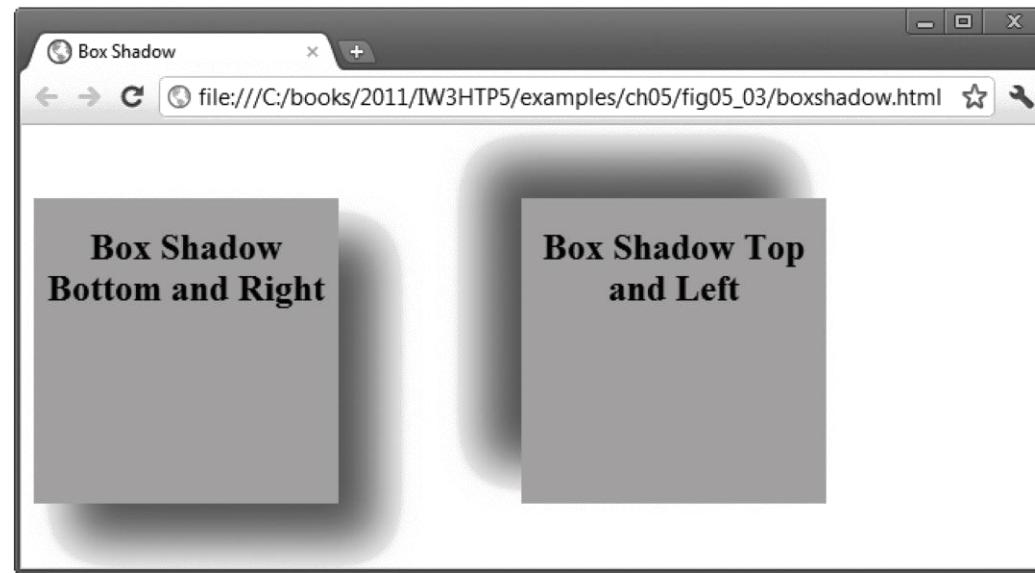
- You can shadow any block-level element in CSS3
- Figure 5.3 shows you how to create a box shadow
  - Horizontal offset of the shadow
    - The number of pixels that the box-shadow will appear to the left or the right of the box
    - A positive value moves the box-shadow to the right
  - Vertical offset of the shadow
    - The number of pixels the box-shadow will be shifted up or down from the box
    - A positive value moves the box-shadow down.
  - Blur radius is a blur-radius of 0px would result in a shadow with a sharp edge (no blur)
    - The greater the value, the more the edges of the shadow are blurred
  - Color is the box-shadow's color

```
1 <!DOCTYPE html>
2
3 <!-- Fig. 5.3: boxshadow.html -->
4 <!-- Creating box-shadow effects. -->
5 <html>
6   <head>
7     <meta charset = "utf-8">
8     <title>Box Shadow</title>
9     <style type = "text/css">
10    div
11    {
12      width: 200px;
13      height: 200px;
14      background-color: plum;
15      box-shadow: 25px 25px 50px dimgrey;
16      float: left;
17      margin-right: 120px;
18      margin-top: 40px;
19    }
20
```

**Fig. 5.3 |** Creating box-shadow effects. (Part 1 of 3.)

```
21      #box2
22  {
23      width: 200px;
24      height: 200px;
25      background-color: plum;
26      box-shadow: -25px -25px 50px dimgrey;
27  }
28  h2
29  {
30      text-align: center;
31  }
32  </style>
33  </head>
34  <body>
35      <div><h2>Box Shadow Bottom and Right</h2></div>
36      <div id = "box2"><h2>Box Shadow Top and Left</h2></div>
37  </body>
38  </html>
```

**Fig. 5.3** | Creating box-shadow effects. (Part 2 of 3.)



**Fig. 5.3** | Creating box-shadow effects. (Part 3 of 3.)

## 5.6 Linear Gradients; Introducing Vendor Prefixes

- Linear gradients are a type of image that gradually transitions from one color to the next horizontally, vertically or diagonally
- You can transition between as many colors as you like
  - Specify the points at which to change colors, called color-stops, represented in pixels or percentages along the gradient line
  - The angle at which the gradient extends
- You can use gradients in any property that accepts an image

## 5.6 Linear Gradients; Introducing Vendor Prefixes (cont.)

- In Fig. 5.4, we create three linear gradients
  - Vertical, horizontal and diagonal in separate rectangles
- The background property for each of the three linear gradient styles (vertical, horizontal and diagonal) is defined multiple times in each style
  - Once for WebKit-based browsers, once for Mozilla Firefox and once using the standard CSS3 syntax for linear gradients
  - WebKit also forms the basis for the experimental browser included with the Amazon Kindle e-book reader, as well as the default browser in the Apple iOS, BlackBerry Browser in OS 6
- This occurs frequently when working with CSS3, because many of its features are **not** yet finalized

Many of the browsers have gone ahead and begun implementing these features so you can use them now?

```
1  <!DOCTYPE html>
2
3  <!-- Fig. 5.4: lineargradient.html -->
4  <!-- Linear gradients in CSS3. -->
5  <html>
6      <head>
7          <meta charset = "utf-8">
8          <title>Linear Gradient</title>
9          <style type = "text/css">
10         div
11             {
12                 width: 200px;
13                 height: 200px;
14                 border: 3px solid navy;
15                 padding: 5px 20px;
16                 text-align: center;
17                 background: -webkit-gradient(
18                     linear, center top, center bottom,
19                     color-stop(15%, white), color-stop(50%, lightsteelblue),
20                     color-stop(75%, navy) );
21                 background: -moz-linear-gradient(
22                     top center, white 15%, lightsteelblue 50%, navy 75% );
```

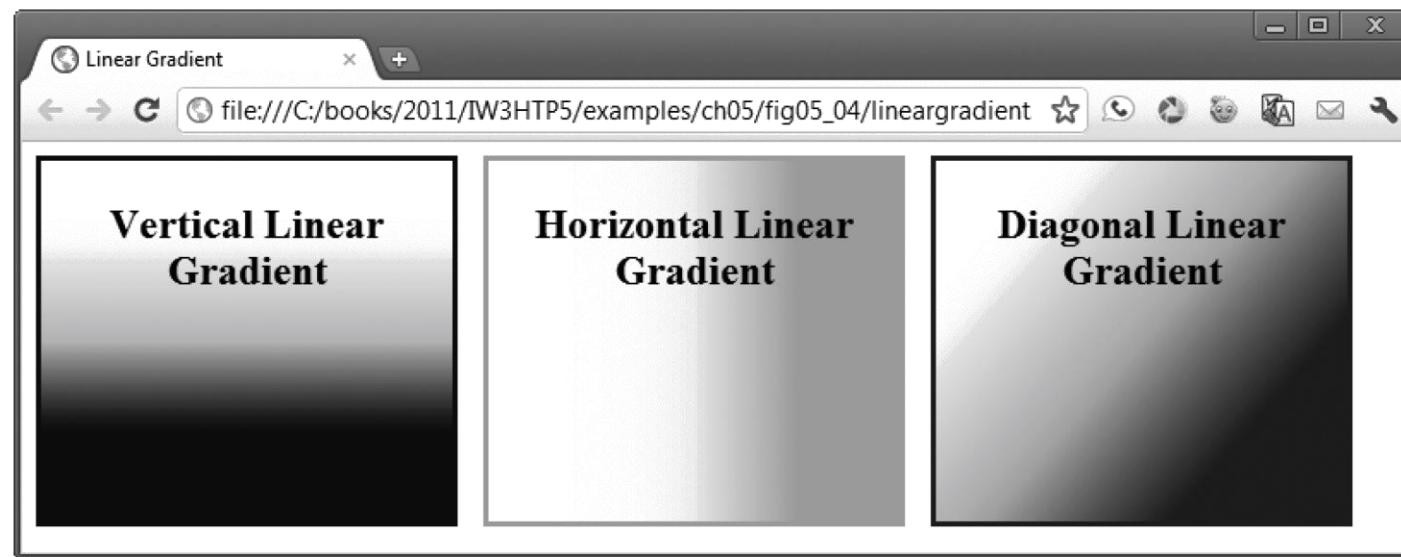
**Fig. 5.4** | Linear gradients in CSS3. (Part I of 4.)

```
23     background: linear-gradient(  
24         to bottom, white 15%, lightsteelblue 50%, navy 75% );  
25     float: left;  
26     margin-right: 15px;  
27 }  
28 #horizontal  
29 {  
30     width: 200px;  
31     height: 200px;  
32     border: 3px solid orange;  
33     padding: 5px 20px;  
34     text-align: center;  
35     background: -webkit-gradient(  
36         linear, left top, right top,  
37         color-stop(15%, white), color-stop(50%, yellow),  
38         color-stop(75%, orange) );  
39     background: -moz-linear-gradient(  
40         left, white 15%, yellow 50%, orange 75% );  
41     background: linear-gradient(  
42         90deg, white 15%, yellow 50%, orange 75% );  
43     margin-right: 15px;  
44 }
```

**Fig. 5.4** | Linear gradients in CSS3. (Part 2 of 4.)

```
45      #angle
46      {
47          width: 200px;
48          height: 200px;
49          border: 3px solid Purple;
50          padding: 5px 20px;
51          text-align: center;
52          background: -webkit-gradient(
53              linear, left top, right bottom,
54              color-stop(15%, white), color-stop(50%, plum),
55              color-stop(75%, purple) );
56          background: -moz-linear-gradient(
57              top left, white 15%, plum 50%, purple 75% );
58          background: linear-gradient(
59              45deg, white 15%, plum 50%, purple 75% );
60      }
61      </style>
62  </head>
63  <body>
64      <div><h2>Vertical Linear Gradient</h2></div>
65      <div id = "horizontal"><h2>Horizontal Linear Gradient</h2></div>
66      <div id = "angle"><h2>Diagonal Linear Gradient</h2></div>
67  </body>
68  </html>
```

Fig. 5.4 | Linear gradients in CSS3. (Part 3 of 4.)



**Fig. 5.4** | Linear gradients in CSS3. (Part 4 of 4.)

## 5.6 Linear Gradients; Introducing Vendor Prefixes (cont.)

- WebKit Vertical Linear Gradient
  - Begin with the background property
- The linear gradient syntax for WebKit differs slightly from that for Firefox
- For WebKit browsers, we use **-webkit-gradient**
- We then specify the type of gradient (linear) and the direction of the linear gradient, from center top to center bottom
- Next, we specify the color-stops for the linear gradient
- Within each color-stop are two values
  - The first is the location of the stop and the second is the color
- You can use as many color-stops as you like

## 5.6 Linear Gradients; Introducing Vendor Prefixes (cont.)

- Mozilla Vertical Linear Gradient
  - For Mozilla browsers, we use **-moz-linear-gradient**
  - **background: -moz-linear-gradient( left, white 15%, yellow 50%, orange 75% );**
- We specify the gradient-line (top center), which is the direction of the gradient
- After the gradient-line we specify each color and color-stop

## 5.6 Linear Gradients; Introducing Vendor Prefixes (cont.)

- Standard Vertical Linear Gradient
  - The standard CSS3 syntax for linear gradients is also slightly different
  - First, we specify the linear-gradient
  - **background: linear-gradient(135deg, white 15%, plum 50%, purple 75%);**
- We include the values for the gradient—the direction of the gradient (top), followed by each color and color-stop

## 5.6 Linear Gradients; Introducing Vendor Prefixes (cont.)

- Horizontal Linear Gradient
  - For WebKit, the direction of the gradient is left top to right top, followed by the colors and color-stops
  - For Mozilla, we specify the gradient-line (left), followed by the colors and color-stops
  - The standard CSS3 syntax begins with the direction (left), indicating that the gradient changes from left to right, followed by the colors and color-stops
- The direction can also be specified in degrees, with 0 degrees straight up and positive degrees progressing clockwise

## 5.6 Linear Gradients; Introducing Vendor Prefixes (cont.)

- Diagonal Linear Gradient
  - For WebKit, the direction of the gradient is left top to right bottom, followed by the colors and color-stops
  - For Mozilla, we specify the gradient-line (top left), followed by the colors and color-stops
  - The standard CSS3 syntax begins with the direction (135deg), indicating that the gradient changes at a 45-degree angle, followed by the colors and color-stops

## 5.6 Linear Gradients; Introducing Vendor Prefixes (cont.)

- Vendor Prefixes (Fig. 5.5)
  - Are used for properties that are still being finalized in the CSS specification but have already been implemented in various browsers
- Prefixes are not available for every browser or for every property
- If we remove the prefixed versions of the linear gradient styles in this example, the gradients will not appear when the page is rendered in a WebKit-based browser or Firefox
- If you run this program in browsers that don't support gradients yet, the gradients will not appear

<b>Vendor prefix</b>	<b>Browsers</b>
-ms-	Internet Explorer
-moz-	Mozilla-based browsers, including Firefox
-o-	Opera and Opera Mobile
-webkit-	WebKit-based browsers, including Google Chrome, Safari (and Safari on the iPhone) and Android

**Fig. 5.5 |** Vendor prefixes.

## 5.6 Linear Gradients; Introducing Vendor Prefixes (cont.)

- It's good practice to include the multiple prefixes when they're available so that your pages render properly in the various browsers
  - Many of the new CSS3 features have **not** yet been implemented in Internet Explorer
- When using vendor prefixes in styles, always place them before the nonprefixed version
  - The last version of the style that a given browser supports takes precedence and the browser will use it
  - By listing the standard non-prefixed version last, the browser will use the standard version over the prefixed version when the standard version is supported

## 5.7 Radial Gradients

- Radial gradients are similar to linear gradients, but the color changes gradually from an inner point (the start) to an outer circle (the end) (Fig. 5.6)
- In this example, the radial-gradient property has three values
  - **background: -webkit-radial-gradient(center, yellow, red);**
- The first is the position of the start of the radial gradient
  - In this case, the center of the rectangle
  - Other possible values for the position include top, bottom, left and right
- The second value is the start color (yellow), and the third is the end color (red)

## 5.7 Radial Gradients (cont.)

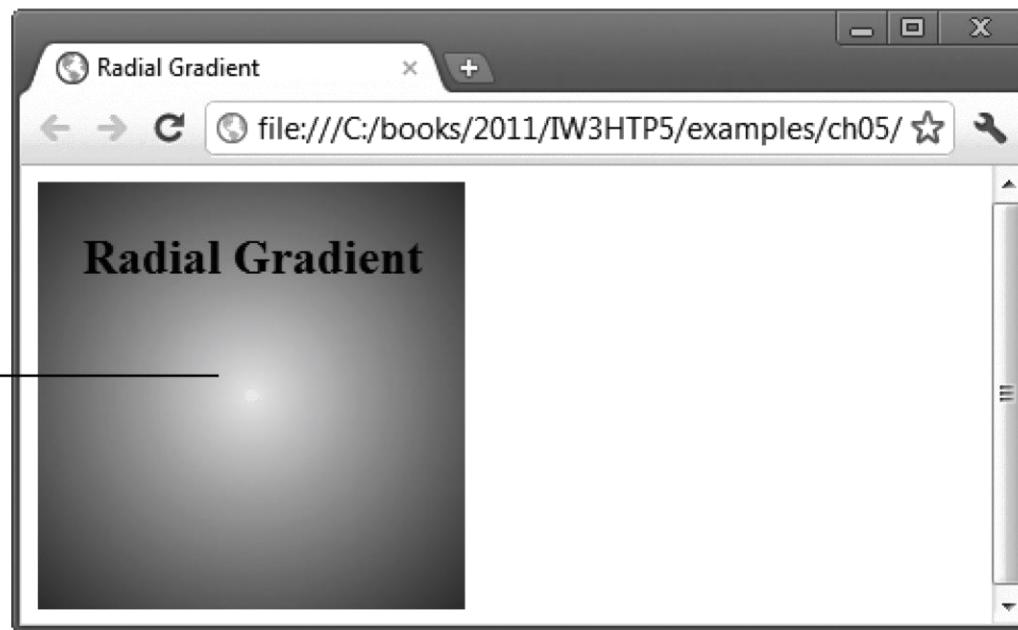
- The resulting effect is a box with a yellow center that gradually changes to red in a circle around the starting position
- In this case, notice that other than the **vendor prefixes**, the syntax of the gradient is identical for WebKit browsers, Mozilla and the standard CSS3 radial-gradient
  - -ms-

```
1  <!DOCTYPE html>
2
3  <!-- Fig. 5.6: radialgradient.html -->
4  <!-- Radial gradients in CSS3. -->
5  <html>
6      <head>
7          <meta charset = "utf-8">
8          <title>Radial Gradient</title>
9          <style type = "text/css">
10         div
11         {
12             width: 200px;
13             height: 200px;
14             padding: 5px;
15             text-align: center;
16             background: -webkit-radial-gradient(center, yellow, red);
17             background: -moz-radial-gradient(center, yellow, red);
18             background: radial-gradient(center, yellow, red);
19         }
20     </style>
21 </head>
```

**Fig. 5.6** | Radial gradients in CSS3. (Part 1 of 2.)

```
22      <body>
23          <div><h2>Radial Gradient</h2></div>
24      </body>
25  </html>
```

Radial gradient begins with yellow in the center, then changes to red in a circle as it moves toward the edges of the box



**Fig. 5.6 | Radial gradients in CSS3. (Part 2 of 2.)**

## 5.8 (Optional: WebKit Only) Text Stroke

- The **-webkit-text-stroke** property is a nonstandard property for WebKit-based browsers that allows you to add an outline (text stroke) around text
- Four of the seven browsers we use in this book are WebKit based
  - Safari and Chrome on the desktop and the mobile browsers in iOS and Android
  - In Fig. 5.7, we set the color of the h1 text to LightCyan
  - We add a **-webkit-text-stroke** with two values
    - The outline thickness (2px)
    - The color of the text stroke (black)
  - We used the font-size 500% for seeing the outline better

```
1  <!DOCTYPE html>
2
3  <!-- Fig. 5.7: textstroke.html -->
4  <!-- Text stroke in CSS3. -->
5  <html>
6      <head>
7          <meta charset = "utf-8">
8          <title>Text Stroke</title>
9          <style type = "text/css">
10         h1
11         {
12             color: lightcyan;
13             -webkit-text-stroke: 2px black; /* vendor prefix */
14             font-size: 500%; /* increasing the font size */
15         }
16     </style>
17     </head>
18     <body>
19         <h1>Text Stroke</h1>
20     </body>
21 </html>
```

**Fig. 5.7** | A text-stroke rendered in Chrome. (Part 1 of 2.)



**Fig. 5.7 |** A text-stroke rendered in Chrome. (Part 2 of 2.)

## 5.9 Multiple Background Images

- CSS3 allows you to add multiple background images to an element (Fig. 5.8)
- The style begins by adding two background-images, logo.png and ocean.png
- Next, we specify each image's placement using property `background-position`
  - The comma-separated list of values matches the order of the comma-separated list of images in the `background-image` property
  - The first value—**bottom right**—places the first image, logo.png, in the bottom-right corner of the background in the border-box

## 5.9 Multiple Background Images (cont.)

- The last value—**100% center**—centers the entire second image, ocean.png, in the content-box
  - It appears behind the content and stretches to fill the content-box
- The **background-origin** determines where each image is placed using the box model we discussed in Fig. 4.13
- The first image (logo.png) is in the outermost border-box, and the second image (ocean.png) is in the innermost content-box

```
1  <!DOCTYPE html>
2
3  <!-- Fig. 5.8: multiplebackgrounds.html -->
4  <!-- Multiple background images in CSS3. -->
5  <html>
6      <head>
7          <meta charset = "utf-8">
8          <title>Multiple Backgrounds</title>
9          <style type = "text/css">
10             div.background
11             {
12                 background-image: url(logo.png), url(ocean.png);
13                 background-position: bottom right, 100% center;
14                 background-origin: border-box, content-box;
15                 background-repeat: no-repeat, repeat;
16             }
17             div.content
18             {
19                 padding: 10px 15px;
20                 color: white;
21                 font-size: 150%;
22             }
23         </style>
24     </head>
```

**Fig. 5.8** | Multiple background images in CSS3. (Part I of 3.)

```
25 <body>
26     <div class = "background">
27         <div class = "content">
28             <p>Deitel & Associates, Inc., is an internationally recognized
29                 authoring and corporate training organization. The company
30                 offers instructor-led courses delivered at client sites
31                 worldwide on programming languages and other software topics
32                 such as C++, Visual C++®, C, Java®,
33                 C#®, Visual Basic®, Objective-C®, XML®,
34                 Python®, JavaScript, object technology,
35                 Internet and web programming, and Android and iPhone app
36                 development.</p>
37         </div></div>
38     </body>
39 </html>
```

**Fig. 5.8** | Multiple background images in CSS3. (Part 2 of 3.)



The second image (`ocean.png`) is centered behind the content and stretched as needed to fill the

The first image (`logo.png`) is placed at the bottom right with

**Fig. 5.8** | Multiple background images in CSS3. (Part 3 of 3.)

## 5.10 (Optional: WebKit Only) Reflections

- Figure 5.9 shows how to add a simple reflection of an image using the **-webkit-box-reflect** property
  - This is a nonstandard property
- The **-webkit-box-reflect** property's first value is the direction of the reflection
  - The direction value may be above, below, left, or right
- The second value is the offset, which determines the space between the image and its reflection
  - Optionally, you can specify a gradient to apply to the reflection
    - The first gradient causes the bottom reflection to fade away from top to bottom
    - The second gradient causes the right reflection to fade away from left to right

```
1  <!DOCTYPE html>
2
3  <!-- Fig. 5.9: reflection.html -->
4  <!-- Reflections in CSS3. -->
5  <html>
6      <head>
7          <meta charset = "utf-8">
8          <title>Reflection</title>
9          <style type = "text/css">
10         img { margin: 10px; }
11         img.below
12         {
13             -webkit-box-reflect: below 5px
14             -webkit-gradient(
15                 linear, left top, left bottom,
16                 from(transparent), to(white));
17     }
```

**Fig. 5.9** | Reflections in CSS3. (Part I of 3.)

```
18      img.right
19      {
20          -webkit-box-reflect: right 5px
21          -webkit-gradient(
22              linear, right top, left top,
23              from(transparent), to(white));
24      }
25  
```

</style>

```
26 </head>
27 <body>
28     <img class = "below" src = "jhtp.png" width = "138" height = "180"
29         alt = "Java How to Program book cover">
30     <img class = "right" src = "jhtp.png" width = "138" height = "180"
31         alt = "Java How to Program book cover">
32 </body>
33 </html>
```

**Fig. 5.9** | Reflections in CSS3. (Part 2 of 3.)



**Fig. 5.9** | Reflections in CSS3. (Part 3 of 3.)

## 5.11 Image Borders

- The CSS3 **border-image** property uses images to place a border around any block-level element (Fig. 5.10)
- We set a div's **border-width** to 30px, which is the thickness of the border we're placing around the element
- Next, we specify a width of 234px, which is the width of the entire rectangular border

```
1  <!DOCTYPE html>
2
3  <!-- Fig. 5.10: imageborder.html -->
4  <!-- Stretching and repeating an image to create a border. -->
5  <html>
6      <head>
7          <meta charset = "utf-8">
8          <title>Image Border</title>
9          <style type = "text/css">
10         div
11             {
12                 border-width: 30px;
13                 width: 234px;
14                 padding: 20px 20px;
15             }
16             #stretch
17             {
18                 -webkit-border-image: url(border.png) 80 80 80 80 stretch;
19                 -moz-border-image: url(border.png) 80 80 80 80 stretch;
20                 -o-border-image: url(border.png) 80 80 80 80 stretch;
21                 border-image: url(border.png) 80 80 80 80 stretch;
22             }
23             #repeat
```

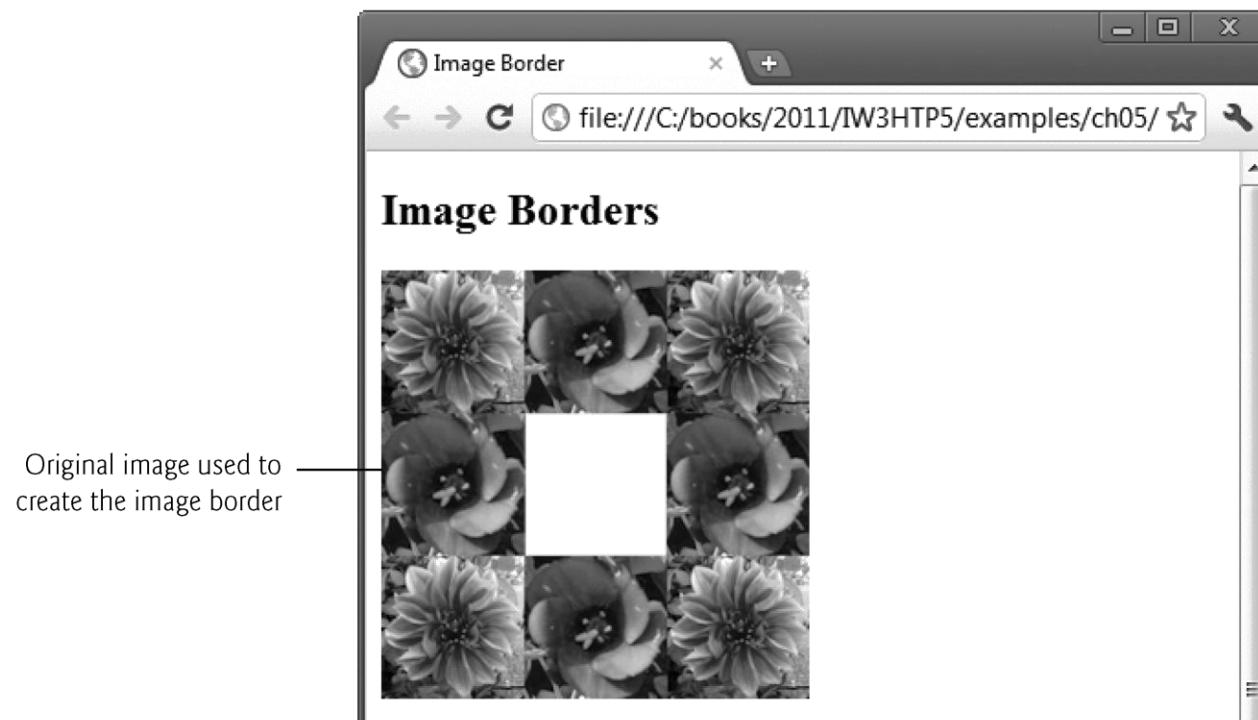
**Fig. 5.10** | Stretching and repeating an image to create a border. (Part 1 of 4.)

```
24      {
25          -webkit-border-image:url(border.png) 34% 34% repeat;
26          -moz-border-image:url(border.png) 34% 34% repeat;
27          -o-border-image:url(border.png) 34% 34% repeat;
28          border-image:url(border.png) 34% 34% repeat;
29      }
30  
```

</style>

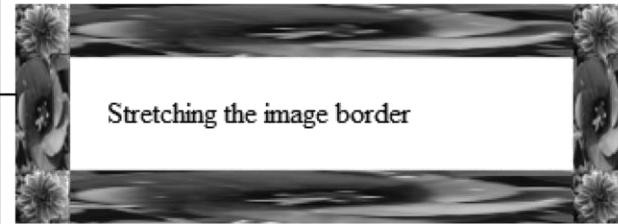
```
31 </head>
32 <body>
33     <h2>Image Borders</h2>
34     <img src = "border.png" alt = "image used to demonstrate borders">
35     <p><div id="stretch">Stretching the image border</div></p>
36     <p><div id="repeat">Repeating the image border</div></p>
37 </body>
38 </html>
```

**Fig. 5.10** | Stretching and repeating an image to create a border. (Part 2 of 4.)

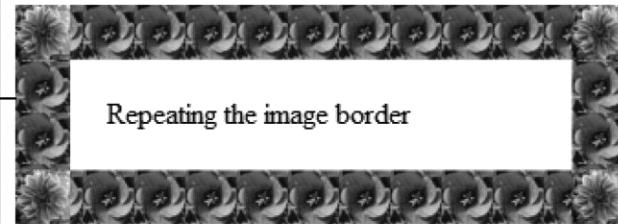


**Fig. 5.10** | Stretching and repeating an image to create a border. (Part 3 of 4.)

Corners of the image remain the same but the four sides of the image are stretched



Corners of the image remain the same but the four sides of the image are repeated



**Fig. 5.10 |** Stretching and repeating an image to create a border. (Part 4 of 4.)

## 5.11 Image Borders (cont.)

- Stretching an Image Border
  - In this example, we create two image border styles
  - In the first, we stretch (and thus distort) the sides of the image to fit around the element while leaving the corners of the border image unchanged (not stretched)
- The **border-image** property has six values
  - **border-image-source**
    - The URL of the image to use in the border
    - **url(border.png)**

## 5.11 Image Borders (cont.)

- **border-image-slice**

- Express with four space-separated values in pixels
- These values are the inward offsets from the top, right, bottom and left sides of the image (**80 80 80 80**)
- The **border-image-slice** divides the image into nine regions
  - Four corners
  - Four sides
  - A middle, which is transparent unless otherwise specified
- You may not use negative values
- We could express the border-image-slice in two values
  - The first value would represent the top and bottom
  - The second value the left and right
- The **border-image-slice** may also be expressed in percentages

## 5.11 Image Borders (cont.)

- **border-image-repeat**

- Specify how the regions of the border image are scaled and tiled (repeated)
- Indicating **stretch** just once creates a border that will stretch the top, right, bottom and left regions to fit the area
- You may specify two values for the **border-image-repeat** property
  - If we specified **stretch repeat**
    - The top and bottom regions of the image border would be stretched, and the right and left regions of the border would be repeated to fit the area
- Other possible values for the **border-image-repeat** property include **round** and **space**
  - If you specify round, the regions are repeated using only whole tiles, and the border image is scaled to fit the area
  - If you specify space, the regions are repeated to fill the area using only whole tiles, and any excess space is distributed evenly around the tiles

## 5.11 Image Borders (cont.)

- Next, we create an image border by repeating the regions to fit the space
- The **border-image** property includes four values
  - **border-image-source**
    - The URL of the image to use in the border
  - **border-image-slice**
    - We provided two values expressed in percentages for the top/bottom and left/right, respectively
  - **border-image-repeat**
    - The value repeat specifies that the tiles are repeated to fit the area, using partial tiles to fill the excess space
    - **border-image: url(border.png) 34% 34% repeat;**

## 5.12 Animation; Selectors

- In Fig. 5.11, we create a simple animation of an image that moves in a diamond pattern as it changes opacity

The animation starts and ends at the top of the diamond, moving the image in the counterclockwise direction initially. When the animation reaches the top of the diamond, the animation reverses, continuing in the clockwise direction. The animation terminates when the image reaches the top of the diamond for a second time.



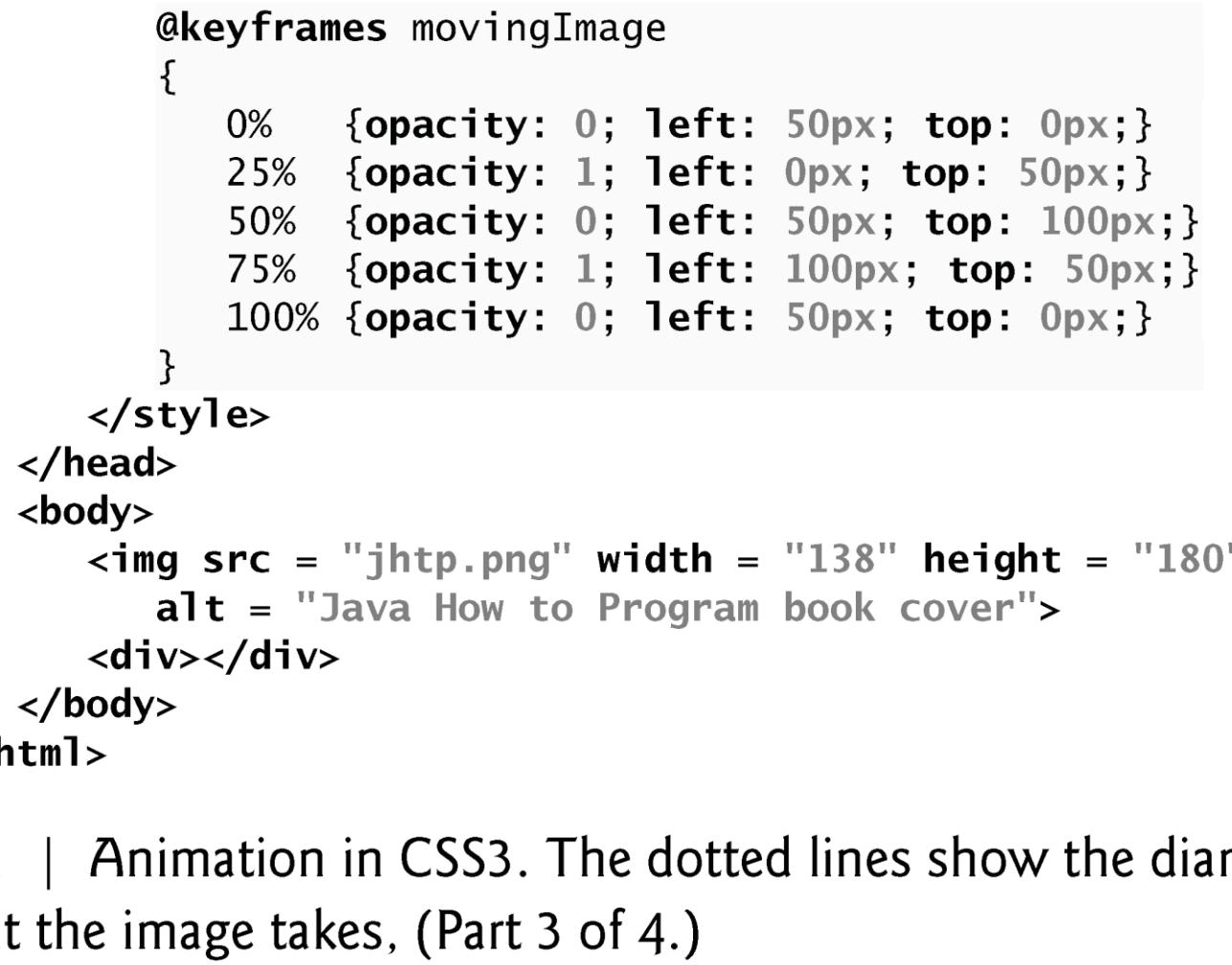
```
1 <!DOCTYPE html>
2
3 <!-- Fig. 5.11: animation.html -->
4 <!-- Animation in CSS3. -->
5 <html>
6   <head>
7     <meta charset = "utf-8">
8     <title>Animation</title>
9     <style type = "text/css">
10    img
11    {
12      position: relative;
13      -webkit-animation: movingImage linear 10s 1s 2 alternate;
14      -moz-animation: movingImage linear 10s 1s 2 alternate;
15      animation: movingImage linear 10s 2 1s alternate;
16    }
```

**Fig. 5.11** | Animation in CSS3. The dotted lines show the diamond path that the image takes, (Part 1 of 4.)

```
17 @-webkit-keyframes movingImage
18 {
19     0% {opacity: 0; left: 50px; top: 0px;}
20     25% {opacity: 1; left: 0px; top: 50px;}
21     50% {opacity: 0; left: 50px; top: 100px;}
22     75% {opacity: 1; left: 100px; top: 50px;}
23     100% {opacity: 0; left: 50px; top: 0px;}
24 }
25 @-moz-keyframes movingImage
26 {
27     0% {opacity: 0; left: 50px; top: 0px;}
28     25% {opacity: 1; left: 0px; top: 50px;}
29     50% {opacity: 0; left: 50px; top: 100px;}
30     75% {opacity: 1; left: 100px; top: 50px;}
31     100% {opacity: 0; left: 50px; top: 0px;}
32 }
```

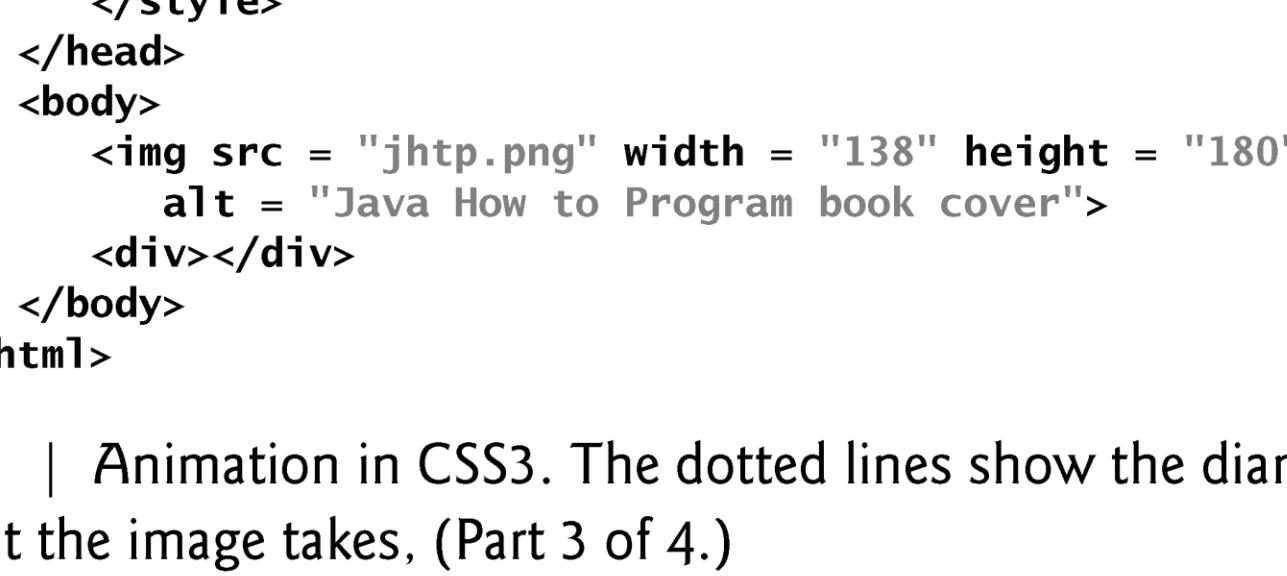
**Fig. 5.11** | Animation in CSS3. The dotted lines show the diamond path that the image takes, (Part 2 of 4.)

```
33      @keyframes movingImage
34      {
35          0% {opacity: 0; left: 50px; top: 0px;}
36          25% {opacity: 1; left: 0px; top: 50px;}
37          50% {opacity: 0; left: 50px; top: 100px;}
38          75% {opacity: 1; left: 100px; top: 50px;}
39          100% {opacity: 0; left: 50px; top: 0px;}
40      }
41  
```



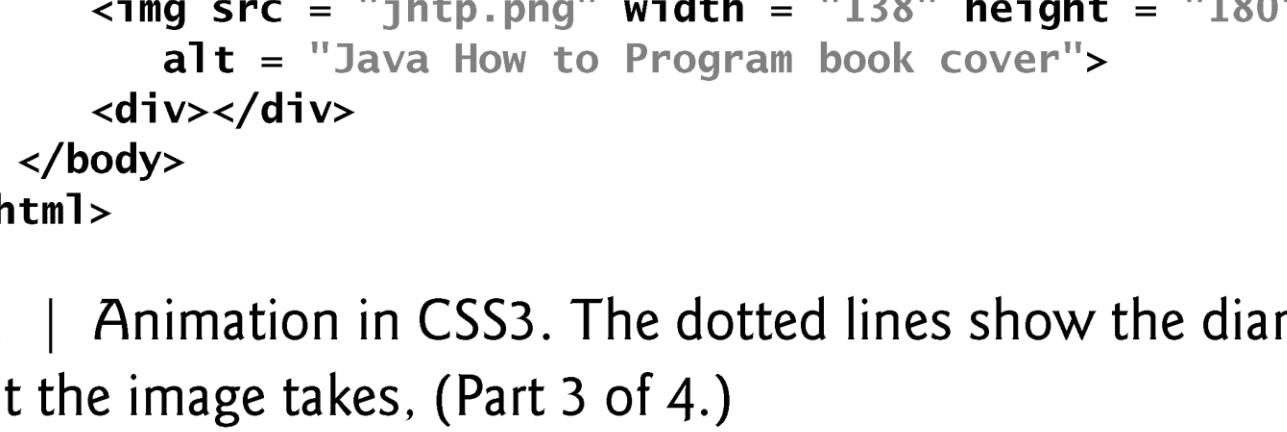
The diagram illustrates a diamond-shaped path for an image animation. It consists of four straight segments connecting vertices at (50, 0), (0, 50), (50, 100), and (100, 50). Dotted lines connect these points to form the diamond shape.

```
42  
```



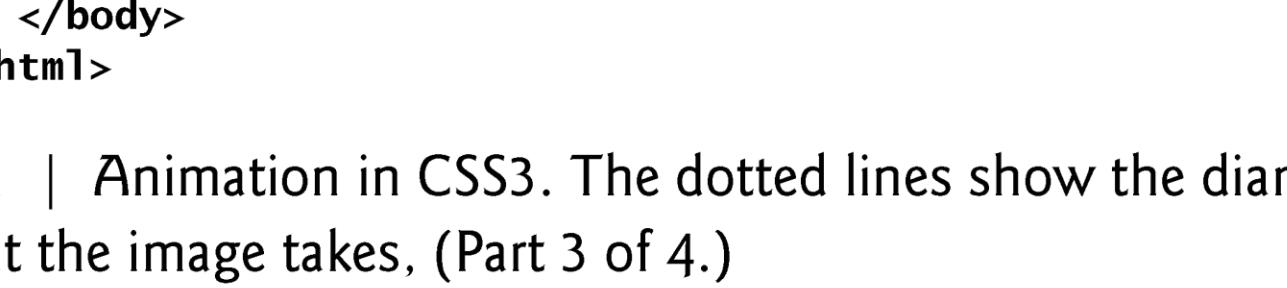
The diagram illustrates a diamond-shaped path for an image animation. It consists of four straight segments connecting vertices at (50, 0), (0, 50), (50, 100), and (100, 50). Dotted lines connect these points to form the diamond shape.

```
43  
```



The diagram illustrates a diamond-shaped path for an image animation. It consists of four straight segments connecting vertices at (50, 0), (0, 50), (50, 100), and (100, 50). Dotted lines connect these points to form the diamond shape.

```
44      <img src = "jhtp.png" width = "138" height = "180"
45          alt = "Java How to Program book cover">
46      <div></div>
47  
```



The diagram illustrates a diamond-shaped path for an image animation. It consists of four straight segments connecting vertices at (50, 0), (0, 50), (50, 100), and (100, 50). Dotted lines connect these points to form the diamond shape.

```
48  
```

**Fig. 5.11** | Animation in CSS3. The dotted lines show the diamond path that the image takes, (Part 3 of 4.)

## 5.12 Animation; Selectors

- The animation property allows you to represent several animation properties in a shorthand notation, rather than specifying each separately
  - **animation-name: movingImage;**
  - **animation-timing-function: linear;**
  - **animation-duration: 10s;**
  - **animation-delay: 1s;**
  - **animation-iteration-count: 2;**
  - **animation-direction: alternate;**

## 5.12 Animation; Selectors (cont.)

- In the shorthand notation, the values are listed in the following order
  - **animation-name**
    - Represents the name of the animation (movingImage)
  - **animation-timing-function**
    - Determines how the animation progresses in one cycle of its duration
    - **linear** specifies that the animation will move at the same speed from start to finish
    - The default value, **ease**, starts slowly, increases speed, then ends slowly
    - **ease-in** starts slowly, then speeds up
    - **ease-out** value starts faster, then slows down.
    - **ease-in-out** starts and ends slowly.
    - **cubic-bezier** allows you to customize the timing function with four values between 0 and 1, such as **cubic-bezier(1,0,0,1)**

## 5.12 Animation; Selectors (cont.)

- **animation-duration**

- Specify the time in seconds (s) or milliseconds (ms) that the animation takes to complete one iteration (10s in this case)

- **animation-delay**

- Specify the number of seconds (1s in this case) or milliseconds after the page loads before the animation begins

- **animation-iteration-count**

- Specify the number of times the animation will run
  - You may use the value **infinite** to repeat the animation continuously

- **animation-direction**

- Specify the direction in which the animation will run

## 5.12 Animation; Selectors (cont.)

- The value **alternate** used here specifies that the animation will run in alternating directions
  - In this case, counterclockwise (as we define with our keyframes), then clockwise
- The default value, **normal**, would run the animation in the same direction for each cycle
- The shorthand animation property cannot be used with the **animation-play-state** property
  - [http://www.w3schools.com/cssref/css3\\_pr\\_animation-play-state.asp](http://www.w3schools.com/cssref/css3_pr_animation-play-state.asp)
- If you do not include the **animation-play-state**, it defaults to running
  - **animation-play-state: running;**

## 5.12 Animation; Selectors (cont.)

- The **@keyframes** rule defines the element's properties that will change during the animation, the values to which those properties will change, and when they'll change
- The **@keyframes** rule is followed by the name of the animation (movingImage) to which the keyframes are applied
- CSS rules consist of one or more selectors followed by a declaration block in curly braces ({} )
- Selectors enable you to apply styles to elements of a particular type or attribute
- A declaration block consists of one or more declarations, each of which includes the property name followed by a colon (:), a value and a semicolon (;

## 5.12 Animation; Selectors (cont.)

- In this example, the **@keyframes** rule includes five selectors to represent the points-in-time for our animation
- You can break down the animation into as many points as you like

- **@keyframes movingimage**

```
{  
    0% {opacity: 0; left: 50px; top: 0px;}  
    25% {opacity: 1; left: 0px; top: 50px;}  
    50% {opacity: 0; left: 50px; top: 100px;}  
    75% {opacity: 1; left: 100px; top: 50px;}  
    100% {opacity: 0; left: 50px; top: 0px;}  
}
```

## 5.13 Transitions and Transformations

- With CSS3 transitions, you can change an element's style over a specified duration
- CSS3 transformations allow you to move, rotate, scale and skew elements
- Transitions are similar in concept to the animations, but transitions allow you to specify only the starting and ending values of the CSS properties being changed
- An animation's keyframes enable you to control intermediate states throughout the animation's duration

## 5.13.1 transition and transform Properties

- Figure 5.12 uses the transition and transform properties to scale and rotate an image 360 degrees when the cursor hovers over it
- For each property that will change, the transition property specifies the duration of that change
- We could specify a comma-separated list of property names that will change and the individual durations over which each property will change
  - **transition: transform 4s, opacity 2s;**
  - Indicates that a transform takes four seconds to apply and the opacity changes over two seconds—thus, the transform will continue for another two seconds after the opacity change completes

```
1 <!DOCTYPE html>
2
3 <!-- Fig. 5.12: transitions.html -->
4 <!-- Transitions in CSS3. -->
5 <html>
6   <head>
7     <meta charset = "utf-8">
8     <title>Transitions</title>
9     <style type = "text/css">
10    img
11    {
12      margin: 80px;
13      -webkit-transition: -webkit-transform 4s;
14      -moz-transition: -moz-transform 4s;
15      -o-transition: -o-transform 4s;
16      transition: transform 4s;
17    }
```

**Fig. 5.12** | Transitioning an image over a four-second duration and applying rotate and scale transforms. (Part 1 of 3.)

```
18     img:hover
19     {
20         -webkit-transform: rotate(360deg) scale(2, 2);
21         -moz-transform: rotate(360deg) scale(2, 2);
22         -o-transform: rotate(360deg) scale(2, 2);
23         transform: rotate(360deg) scale(2, 2);
24     }
25 
```

</style>

```
26 </head>
27 <body>
28     <img src = "cpphttp.png" width = "76" height = "100"
29         alt = "C++ How to Program book cover">
30 </body>
31 </html>
```

**Fig. 5.12** | Transitioning an image over a four-second duration and applying rotate and scale transforms. (Part 2 of 3.)

a)



b)



c)



d)



**Fig. 5.12** | Transitioning an image over a four-second duration and applying rotate and scale transforms. (Part 3 of 3.)

## 5.13.1 transition and transform Properties (cont.)

- The **:hover** pseudo-class formerly worked only for anchor elements but now works with any element
  - We use **:hover** to begin the rotation and scaling of the image
- The **transform** property specifies that the image will rotate 360deg and will scale to twice its original width and height when the mouse hovers over the image
- The **transform** property uses transformation functions, such as **rotate** and **scale**, to perform the transformations
- The **rotate transformation** function receives the number of degrees
  - Negative values cause the element to rotate left

## 5.13.1 transition and transform Properties (cont.)

- The **scale transformation** function specifies how to scale the width and height
  - The value 1 represents the original width or original height
  - Values greater than 1 increase the size and values less than 1 decrease the size
    - <http://www.w3.org/TR/css3-2d-transforms/#transform-functions>
    - [http://www.w3schools.com/cssref/css3\\_pr\\_transform.asp](http://www.w3schools.com/cssref/css3_pr_transform.asp)
- The **translation transformation** function specifies how to translate the position
  - **transform:translate(100px,100px);**

## 5.13.2 Skew

- CSS3 transformations also allow you to skew block-level elements, slanting them at an angle either horizontally (skewX) or vertically (skewY)
- We use the animation and transform properties to skew a rectangle and text horizontally by 45 degrees (Fig. 5.13)
- First we create a rectangle with a LightGreen background, a solid DarkGreen border and rounded corners
- The **animation** property specifies that the element will skew in a three-second (3s) interval for an infinite duration
- The fourth value, **linear**, is the **animation-timing-function**
- Next, we use the **@-webkit-keyframes** rule

```
1 <!DOCTYPE html>
2
3 <!-- Fig. 5.13: skew.html -->
4 <!-- Skewing and transforming elements in CSS3. -->
5 <html>
6   <head>
7     <meta charset = "utf-8">
8     <title>Skew</title>
9     <style type = "text/css">
10       .skew .textbox
11     {
12       margin-left: 75px;
13       background: lightgreen;
14       height: 100px;
15       width: 200px;
16       padding: 25px 0;
17       text-align: center;
18       font-size: 250%;
19       border: 3px solid DarkGreen;
20       border-radius: 15px;
21       -webkit-animation: skew 3s infinite linear;
22       -moz-animation: skew 3s infinite linear;
23       animation: skew 3s infinite linear;
24     }
```

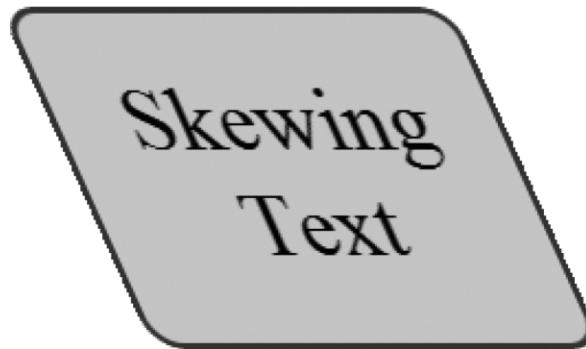
**Fig. 5.13** | Skewing and transforming elements in CSS3. (Part I of 3.)

```
25      @-webkit-keyframes skew
26      {
27          from { -webkit-transform: skewX(0deg); }
28          25% { -webkit-transform: skewX(45deg); }
29          50% { -webkit-transform: skewX(0); }
30          75% { -webkit-transform: skewX(-45deg); }
31          to { -webkit-transform: skewX(0); }
32      }
33      @-moz-keyframes skew
34      {
35          from { -webkit-transform: skewX(0deg); }
36          25% { -webkit-transform: skewX(45deg); }
37          50% { -webkit-transform: skewX(0); }
38          75% { -webkit-transform: skewX(-45deg); }
39          to { -webkit-transform: skewX(0); }
40      }
41      @-keyframes skew
42      {
43          from { -webkit-transform: skewX(0deg); }
44          25% { -webkit-transform: skewX(45deg); }
45          50% { -webkit-transform: skewX(0); }
46          75% { -webkit-transform: skewX(-45deg); }
47          to { -webkit-transform: skewX(0); }
48      }
49  
```

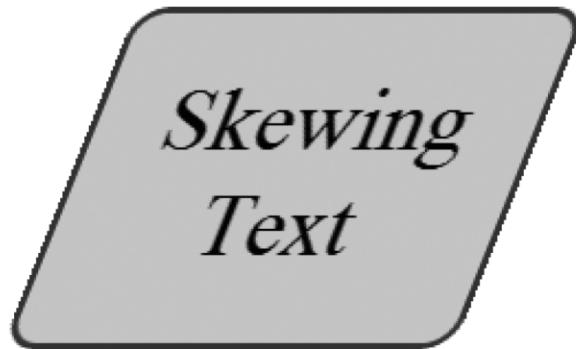
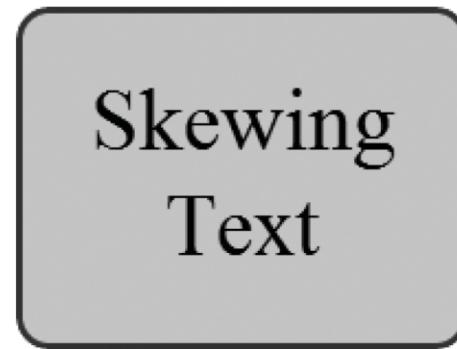
**Fig. 5.13** | Skewing and transforming elements in CSS3. (Part 2 of 3.)

```
50    </head>
51    <body>
52        <div class = "box skew">
53            <div class = "textbox">Skewing Text</div>
54        </div>
55    </body>
56 </html>
```

a) Bordered div at skewed left position      b) Bordered div at centered position



c) Bordered div at skewed right position



**Fig. 5.13 | Skewing and transforming elements in CSS3. (Part 3 of 3.)**

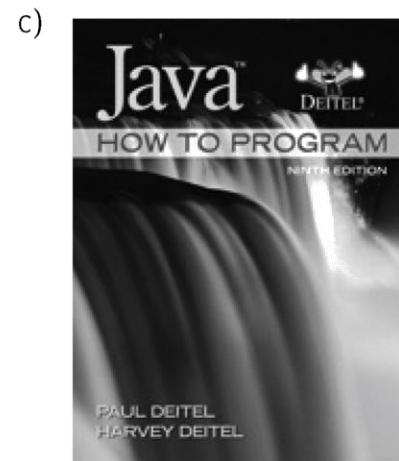
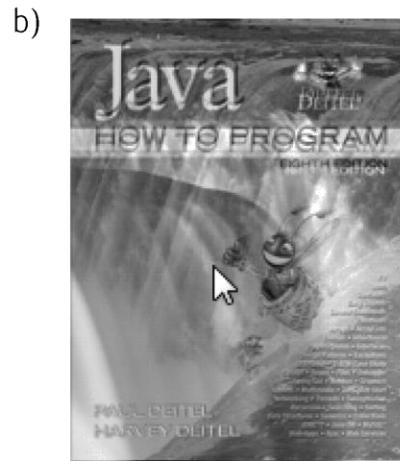
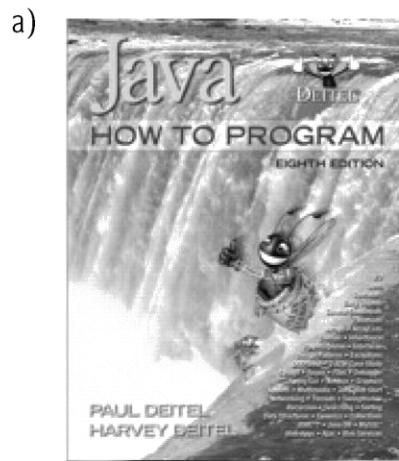
## 5.13.3 Transitioning Between Images

- We can use the **transition** property to create the visually beautiful effect of melting one image into another (Fig. 5.14)
- The **transition** property includes three values
  - First, we specify that the transition will occur on the opacity of the image.
  - The second value, 4s, is the transition-duration
  - The third value, **ease-in-out**, is the **transition-timing-function**
- Next, we define **:hover** with an opacity of 0, so when the cursor hovers over the top image, its opacity becomes fully transparent, revealing the bottom image directly behind it
  - We then add the bottom and top images, placing one directly behind the other

```
1 <!DOCTYPE html>
2
3 <!-- Fig. 5.14: meltingimages.html -->
4 <!-- Melting one image into another using CSS3. -->
5 <html>
6   <head>
7     <meta charset = "utf-8">
8     <title>Melting Images</title>
9     <style type = "text/css">
10       #cover
11       {
12         position: relative;
13         margin: 0 auto;
14       }
15       #cover img
16       {
17         position: absolute;
18         left: 0;
19         -webkit-transition: opacity 4s ease-in-out;
20         transition: opacity 4s ease-in-out;
21       }
22       #cover img.top:hover
23         { opacity:0; }
24     </style>
```

**Fig. 5.14** | Melting one image into another using CSS3. (Part I of 2.)

```
25    </head>
26    <body>
27        <div id = "cover">
28            <img class = "bottom" src = "jhttp.png" alt = "Java 9e cover">
29            <img class = "top" src = "jhttp8.png" alt = "Java 8e cover">
30        </div>
31    </body>
32 </html>
```



**Fig. 5.14** | Melting one image into another using CSS3. (Part 2 of 2.)

## 5.14 Downloading Web Fonts and the @font-face Rule

- Using the **@font-face rule**, you can specify fonts for a web page, even if they're not installed on the user's system
- You can use downloadable fonts to help ensure a uniform look across client sites
- In Fig. 5.15, we use the Google web font named “Calligraffiti”
- You can find numerous free, open-source web fonts at <http://www.google.com/webfonts>
- **<link href = 'http://fonts.googleapis.com/css?family=Tangerine' rel = 'stylesheet' type='text/css'>**
- Make sure the fonts you get from other sources are legal

```
1 <!DOCTYPE html>
2
3 <!-- Fig. 5.15: embeddedfonts.html -->
4 <!-- Embedding fonts for use in your web page. -->
5 <html>
6   <head>
7     <meta charset = "utf-8">
8     <title>Embedded Fonts</title>
9     <link href = 'http://fonts.googleapis.com/css?family=Calligraffiti'
10       rel = 'stylesheet' type = 'text/css'>
11   <style type = "text/css">
12     body
13     {
14       font-family: "Calligraffiti";
15       font-size: 48px;
16       text-shadow: 3px 3px 3px DimGrey;
17     }
18   </style>
19 </head>
```

**Fig. 5.15** | Embedding fonts for use in your web page. (Part I of 2.)

```
20 <body>
21   <div>
22     <b>Embedding the Google web font "Calligraffiti"</b>
23   </div>
24 </body>
25 </html>
```



**Fig. 5.15** | Embedding fonts for use in your web page. (Part 2 of 2.)

## 5.14 Downloading Web Fonts and the @font-face Rule (cont.)

- To get Google's Calligraffiti font, go to <http://www.google.com/webfonts> and use the search box on the site to find the font "Calligraffiti"
- Next, click Quick-use to get the link to the style sheet that contains the **@font-face rule**
- Paste that link element into the head section of your document

## 5.15 Flexible Box Layout Module and :nth-child Selectors

- Flexible Box Layout Module (FBLM) makes it easy to align the contents of boxes, change their size, change their order dynamically, and lay out the contents in any direction.
- In Fig. 5.16, we create flexible **divs** for four programming tips
  - When the mouse hovers over one of the **divs**, the **div** expands, the text changes from black to white, the background color changes and the layout of the text changes

```
1  <!DOCTYPE html>
2
3  <!-- Fig. 5.16: fb1m.html -->
4  <!-- Flexible Box Layout Module. -->
5  <html>
6      <head>
7          <meta charset = "utf-8">
8          <title>Flexible Box Layout Model</title>
9          <link href = 'http://fonts.googleapis.com/css?family=Rosario'
10             rel = 'stylesheet' type = 'text/css'>
11          <style type = "text/css">
12              .flexbox
13              {
14                  width: 600px;
15                  height: 420px;
16                  display: -webkit-box;
17                  display: box;
18                  -webkit-box-orient: horizontal;
19                  box-orient: horizontal;
20              }
```

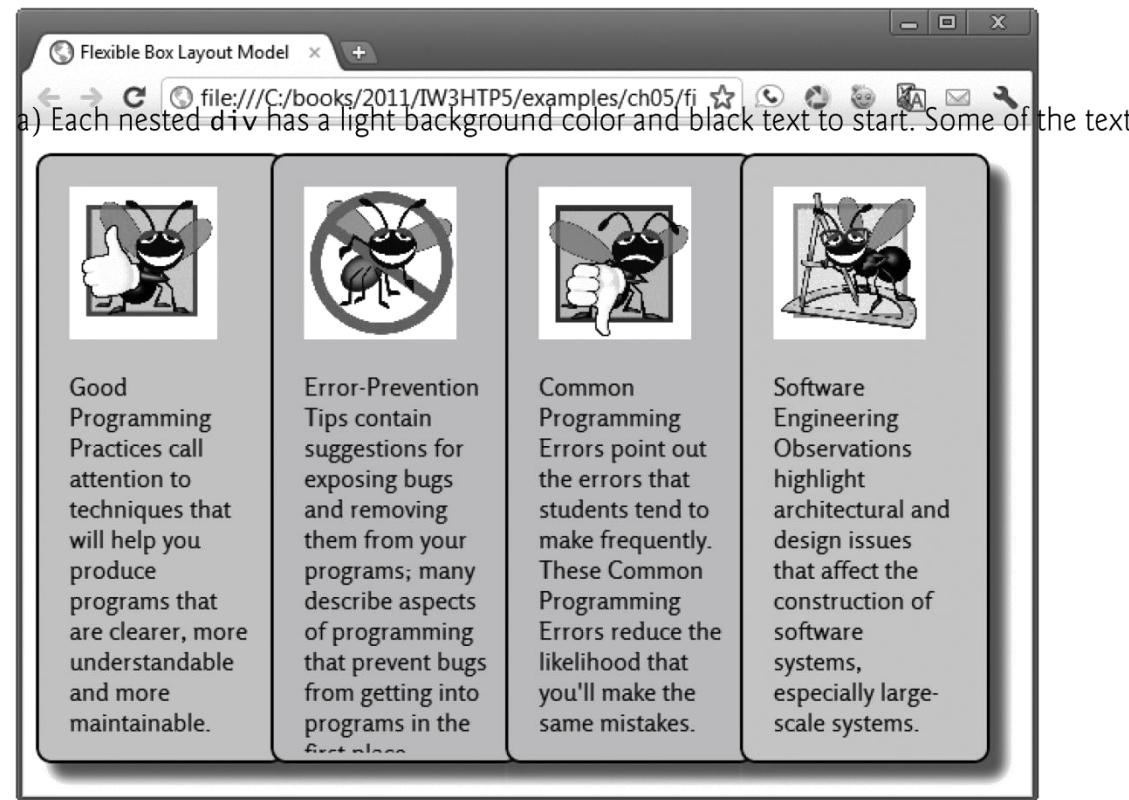
**Fig. 5.16** | Flexible Box Layout Module. (Part 1 of 5.)

```
21 .flexbox > div
22 {
23     -webkit-transition: 1s ease-out;
24     transition: 1s ease-out;
25     -webkit-border-radius: 10px;
26     border-radius: 10px;
27     border: 2px solid black;
28     width: 120px;
29     margin: 10px -10px 10px 0px;
30     padding: 20px 20px 20px 20px;
31     box-shadow: 10px 10px 10px dimgrey;
32 }
33 .flexbox > div:nth-child(1){ background-color: lightgrey; }
34 .flexbox > div:nth-child(2){ background-color: lightgrey; }
35 .flexbox > div:nth-child(3){ background-color: lightgrey; }
36 .flexbox > div:nth-child(4){ background-color: lightgrey; }
37
38 .flexbox > div:hover {
39     width: 200px; color: white; font-weight: bold; }
40 .flexbox > div:nth-child(1):hover { background-color: royalblue; }
41 .flexbox > div:nth-child(2):hover { background-color: crimson; }
42 .flexbox > div:nth-child(3):hover { background-color: crimson; }
43 .flexbox > div:nth-child(4):hover { background-color: darkgreen; }
44 p { height: 250px; overflow: hidden; font-family: "Rosario" }
45 </style>
```

Fig. 5.16 | Flexible Box Layout Module. (Part 2 of 5.)

```
46    </head>
47    <body>
48        <div class = "flexbox">
49            <div><img src = "GPP.png" alt = "Good programming practice icon">
50                <p>Good Programming Practices call attention to techniques that
51                    will help you produce programs that are clearer, more
52                    understandable and more maintainable.</p></div>
53            <div><img src = "EPT.png" alt = "Error prevention tip icon">
54                <p>Error-Prevention Tips contain suggestions for exposing bugs
55                    and removing them from your programs; many describe aspects of
56                    programming that prevent bugs from getting into programs in
57                    the first place.</p></div>
58            <div><img src = "CPE.png" alt = "Common programming error icon">
59                <p>Common Programming Errors point out the errors that students
60                    tend to make frequently. These Common Programming Errors reduce
61                    the likelihood that you'll make the same mistakes.</p></div>
62            <div><img src = "SEO.png"><p>Software Engineering Observations
63                highlight architectural and design issues that affect the
64                construction of software systems, especially large-scale
65                systems.</p></div>
66        </div>
67    </body>
68 </html>
```

**Fig. 5.16 |** Flexible Box Layout Module. (Part 3 of 5.)



**Fig. 5.16 |** Flexible Box Layout Module. (Part 4 of 5.)

- b) When the mouse hovers over :nth-child(2), the flexbox expands, the background-color changes to Crimson, the overflow text is revealed and the text changes to a bold



**Fig. 5.16 | Flexible Box Layout Module. (Part 5 of 5.)**

## 5.15 Flexible Box Layout Module and :nth-child Selectors (cont.)

- We define a **div** to which we apply the flexbox CSS class
  - That **div** contains four other **divs**
- The **flexbox class's display** property is set to the new CSS3 value **box**
- The **box-orient** property specifies the orientation of the box layout
  - The default value is horizontal (which we specified anyway)
  - You can also use vertical
- For the nested **divs**, we specify a one-second ease-out transition
- This will take effect when these the **:hover** pseudo-class style is applied to one of these **divs** to expand it

## 5.15 Flexible Box Layout Module and :nth-child Selectors (cont.)

- In CSS3, you can use selectors to easily select elements to style based on their attributes
- We use **:nth-child** selectors to select each of the four **div** elements in the flexbox **div** to style
- **div:nth-child(1)** selects the **div** element that's the first child of its parent and applies the **background-color** lightgrey
- Similarly, **div:nth-child(2)** selects the **div** element that's the second child of its parent, **div:nth-child(3)** selects the third child of its parent, and **div:nth-child(4)** selects the fourth child of its parent—each applies a specified background-color

## 5.15 Flexible Box Layout Module and :nth-child Selectors (cont.)

- Next, we define styles that are applied to the nested **div** elements when the mouse hovers over them
  - The **width (200px)**, **color (white)** and **font-weight (bold)**
- We use **:nth-child** selectors to specify a new background color for each nested **div**
- Finally, we style the **p** element—the text within each **div**
- In the output, notice that the text in the second child element (the Error-Prevention Tips), the **overflow** text is **hidden**
  - **overflow: hidden;** vs. **overflow: auto;**
- When the mouse hovers over the element, all of the text is revealed

## 5.16 Multicolumn Layout

- CSS3 allows you to easily create multicolumn layouts
- In Figure 5.17, we create a three-column layout by setting the **column-count** property to 3 and the **column-gap** property (the spacing between columns) to 30px
- We then add a thin black line between each column using the **column-rule** property
- When you run this example, try resizing your browser window
- The width of the columns changes to fit the three-column layout in the browser

```
1 <!DOCTYPE html>
2
3 <!-- Fig. 5.17: multicolumns.html -->
4 <!-- Multicolumn text in CSS3. -->
5 <html>
6   <head>
7     <meta charset = "utf-8">
8     <title>Multicolumns</title>
9     <style type = "text/css">
10    p
11      { margin:0.9em 0em; }
12      .multicolumns
13    {
14      /* setting the number of columns to 3 */
15      -webkit-column-count: 3;
16      -moz-column-count: 3;
17      -o-column-count: 3;
18      column-count: 3;
19      /* setting the space between columns to 30px */
20      -webkit-column-gap: 30px;
21      -moz-column-gap: 30px;
22      -o-column-gap: 30px;
23      column-gap: 30px;
```

**Fig. 5.17** | Multicolumn text in CSS3. (Part I of 5.)

```
24      /* adding a 1px black line between each column */
25      -webkit-column-rule: 1px outset black;
26      -moz-column-rule: 1px outset black;
27      -o-column-rule: 1px outset black;
28      column-rule: 1px outset black;
29  }
30  </style>
31 </head>
32 <body>
33   <header>
34     <h1>Computers, Hardware and Software<h1/>
35   </header>
36   <div class = "multicolumns">
37     <p>A computer is a device that can perform computations and make
38       logical decisions phenomenally faster than human beings can.
39       Many of today's personal computers can perform billions of
40       calculations in one second&mdash;more than a human can perform
41       in a lifetime. Supercomputers are already performing thousands
42       of trillions (quadrillions) of instructions per second! To put
43       that in perspective, a quadrillion-instruction-per-second
44       computer can perform in one second more than 100,000
45       calculations for every person on the planet! And&mdash;these
46       "upper limits" are growing quickly!</p>
47     <p>Computers process data under the control of sets of
48       instructions called computer programs. These programs guide
```

**Fig. 5.17 | Multicolumn text in CSS3. (Part 2 of 5.)**

49 the computer through orderly sets of actions specified by  
50 people called computer programmers. The programs that run on a  
51 computer are referred to as software. In this book, you'll  
52 learn today's key programming methodology that's enhancing  
53 programmer productivity, thereby reducing software-development  
54 costs&mdash;object-oriented programming.</p>  
55 <p>A computer consists of various devices referred to as hardware  
56 (e.g., the keyboard, screen, mouse, hard disks, memory, DVDs  
57 and processing units). Computing costs are dropping  
58 dramatically, owing to rapid developments in hardware and  
59 software technologies. Computers that might have filled large  
60 rooms and cost millions of dollars decades ago are now  
61 inscribed on silicon chips smaller than a fingernail, costing  
62 perhaps a few dollars each. Ironically, silicon is one of the  
63 most abundant materials&mdash;it's an ingredient in common  
64 sand. Silicon-chip technology has made computing so economical  
65 that more than a billion general-purpose computers are in use  
66 worldwide, and this is expected to double in the next few  
67 years.</p>  
68 <p>Computer chips (microprocessors) control countless devices.  
69 These embedded systems include anti-lock brakes in cars,  
70 navigation systems, smart home appliances, home security  
71 systems, cell phones and smartphones, robots, intelligent  
72 traffic intersections, collision avoidance systems, video game  
73 controllers and more. The vast majority of the microprocessors

**Fig. 5.17 | Multicolumn text in CSS3. (Part 3 of 5.)**

```
74      produced each year are embedded in devices other than general-
75      purpose computers.</p>
76      <footer>
77          <em>&copy; 2012 by Pearson Education, Inc.
78          All Rights Reserved.</em>
79      </footer>
80      </div>
81  </body>
82 </html>
```

**Fig. 5.17** | Multicolumn text in CSS3. (Part 4 of 5.)

A screenshot of a web browser window titled "Multicolumns". The address bar shows the URL "file:///C:/books/2011/IW3HTP5/examples/ch05/fig05\_17/multicolumns.html". The main content area displays a heading "Computers, Hardware and Software" followed by three columns of text. The first column discusses computer performance and supercomputers. The second column discusses computer hardware and software. The third column discusses silicon chips and their widespread use.

## Computers, Hardware and Software

A computer is a device that can perform computations and make logical decisions phenomenally faster than human beings can. Many of today's personal computers can perform billions of calculations in one second—more than a human can perform in a lifetime. Supercomputers are already performing thousands of trillions (quadrillions) of instructions per second! To put that in perspective, a quadrillion-instruction-per-second computer can perform in one second more than 100,000 calculations for every person on the planet! And—these "upper limits" are growing quickly!

Computers process data under the control of sets of instructions called computer programs. These programs guide the computer through orderly sets of actions specified by people called

computer programmers. The programs that run on a computer are referred to as software. In this book, you'll learn today's key programming methodology that's enhancing programmer productivity, thereby reducing software-development costs—object-oriented programming.

A computer consists of various devices referred to as hardware (e.g., the keyboard, screen, mouse, hard disks, memory, DVDs and processing units). Computing costs are dropping dramatically, owing to rapid developments in hardware and software technologies. Computers that might have filled large rooms and cost millions of dollars decades ago are now inscribed on silicon chips smaller than a fingernail, costing perhaps a few dollars each. Ironically, silicon is one of the most abundant materials—it's an ingredient in common sand. Silicon-chip technology has made computing so economical that more than a billion general-purpose computers are in use worldwide, and this is expected to double in the next few years.

Computer chips (microprocessors) control countless devices. These embedded systems include anti-lock brakes in cars, navigation systems, smart home appliances, home security systems, cell phones and smartphones, robots, intelligent traffic intersections, collision avoidance systems, video game controllers and more. The vast majority of the microprocessors produced each year are embedded in devices other than general-purpose computers.

**Fig. 5.17 | Multicolumn text in CSS3. (Part 5 of 5.)**

## 5.17 Media Queries

- With CSS3 media queries, you can determine the **finer attributes** of the media on which the user is viewing the page, such as the length and width of the viewing area on the screen, to better customize your presentation
- In Fig. 5.18, we modify the multicolumn example to alter the numbers of columns and the rules between columns based on the screen size of the device on which the page is viewed

```
1 <!DOCTYPE html>
2
3 <!-- Fig. 5.18: mediaqueries.html -->
4 <!-- Using media queries to reformat a page based on the device width. -->
5 <html>
6   <head>
7     <meta charset = "utf-8">
8     <title>Media Queries</title>
9     <style type = "text/css">
10    p
11      { margin: 0.9em 0em; }
12      /* styles for smartphones with screen widths 480px or smaller */
13      @media handheld and (max-width: 480px),
14        screen and (max-device-width: 480px),
15        screen and (max-width: 480px)
16      {
17        div {
18          -webkit-column-count: 1;
19          column-count: 1; }
20      }
```

**Fig. 5.18** | Using media queries to reformat a page based on the device width. (Part 1 of 8.)

```
21      /* styles for devices with screen widths of 481px to 1024px */
22      @media only screen and (min-width: 481px) and
23          (max-width: 1024px)
24      {
25          div {
26              -webkit-column-count: 2;
27              column-count: 2;
28              -webkit-column-gap: 30px;
29              column-gap: 30px;
30              -webkit-column-rule: 1px outset black;
31              column-rule: 1px outset black; }
32      }
33      /* styles for devices with screen widths of 1025px or greater */
34      @media only screen and (min-width: 1025px)
35      {
36          div {
37              -webkit-column-count: 3;
38              column-count: 3;
39              -webkit-column-gap: 30px;
40              column-gap: 30px;
41              -webkit-column-rule: 1px outset black;
42              column-rule: 1px outset black; }
43      }
```

**Fig. 5.18** | Using media queries to reformat a page based on the device width. (Part 2 of 8.)

```
44      </style>
45  </head>
46  <body>
47      <header>
48          <h1>Computers, Hardware and Software</h1>
49      </header>
50      <div>
51          <p>A computer is a device that can perform computations and make
52              logical decisions phenomenally faster than human beings can.
53              Many of today's personal computers can perform billions of
54              calculations in one second&mdash;more than a human can perform
55              in a lifetime. Supercomputers are already performing thousands
56              of trillions (quadrillions) of instructions per second! To put
57              that in perspective, a quadrillion-instruction-per-second
58              computer can perform in one second more than 100,000
59              calculations for every person on the planet! And&mdash;these
60              "upper limits" are growing quickly!</p>
61          <p>Computers process data under the control of sets of
62              instructions called computer programs. These programs guide
63              the computer through orderly sets of actions specified by
64              people called computer programmers. The programs that run on a
65              computer are referred to as software. In this book, you'll
66              learn today's key programming methodology that's enhancing
```

**Fig. 5.18 |** Using media queries to reformat a page based on the device width. (Part 3 of 8.)

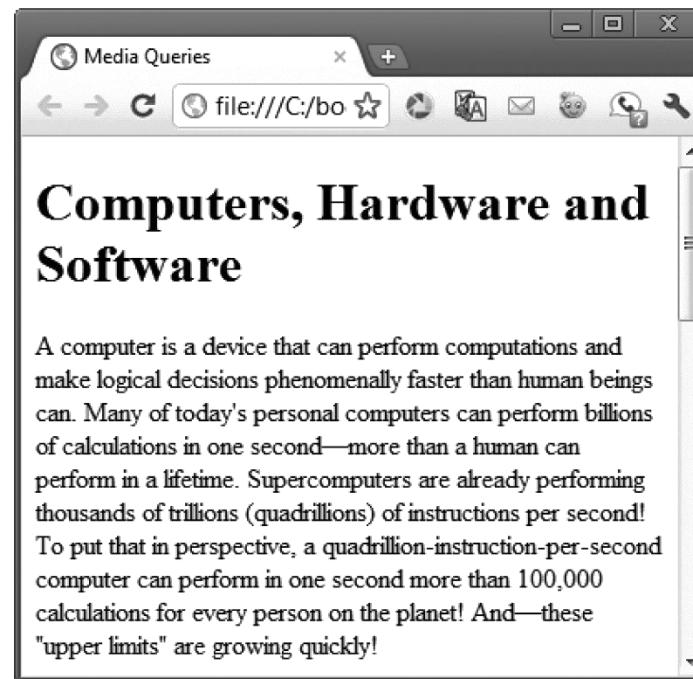
67 programmer productivity, thereby reducing software-development  
68 costs&mdash;object-oriented programming.</p>  
69 <p>A computer consists of various devices referred to as hardware  
70 (e.g., the keyboard, screen, mouse, hard disks, memory, DVDs  
71 and processing units). Computing costs are dropping  
72 dramatically, owing to rapid developments in hardware and  
73 software technologies. Computers that might have filled large  
74 rooms and cost millions of dollars decades ago are now  
75 inscribed on silicon chips smaller than a fingernail, costing  
76 perhaps a few dollars each. Ironically, silicon is one of the  
77 most abundant materials&mdash;it's an ingredient in common  
78 sand. Silicon-chip technology has made computing so economical  
79 that more than a billion general-purpose computers are in use  
80 worldwide, and this is expected to double in the next few  
81 years.</p>  
82 <p>Computer chips (microprocessors) control countless devices.  
83 These embedded systems include anti-lock brakes in cars,  
84 navigation systems, smart home appliances, home security  
85 systems, cell phones and smartphones, robots, intelligent  
86 traffic intersections, collision avoidance systems, video game  
87 controllers and more. The vast majority of the microprocessors  
88 produced each year are embedded in devices other than general-  
89 purpose computers.</p>

**Fig. 5.18** | Using media queries to reformat a page based on the device width. (Part 4 of 8.)

```
90      <footer>
91          <em>&copy; 2012 by Pearson Education, Inc.
92          All Rights Reserved.</em>
93      </footer>
94  </div>
95 </body>
96 </html>
```

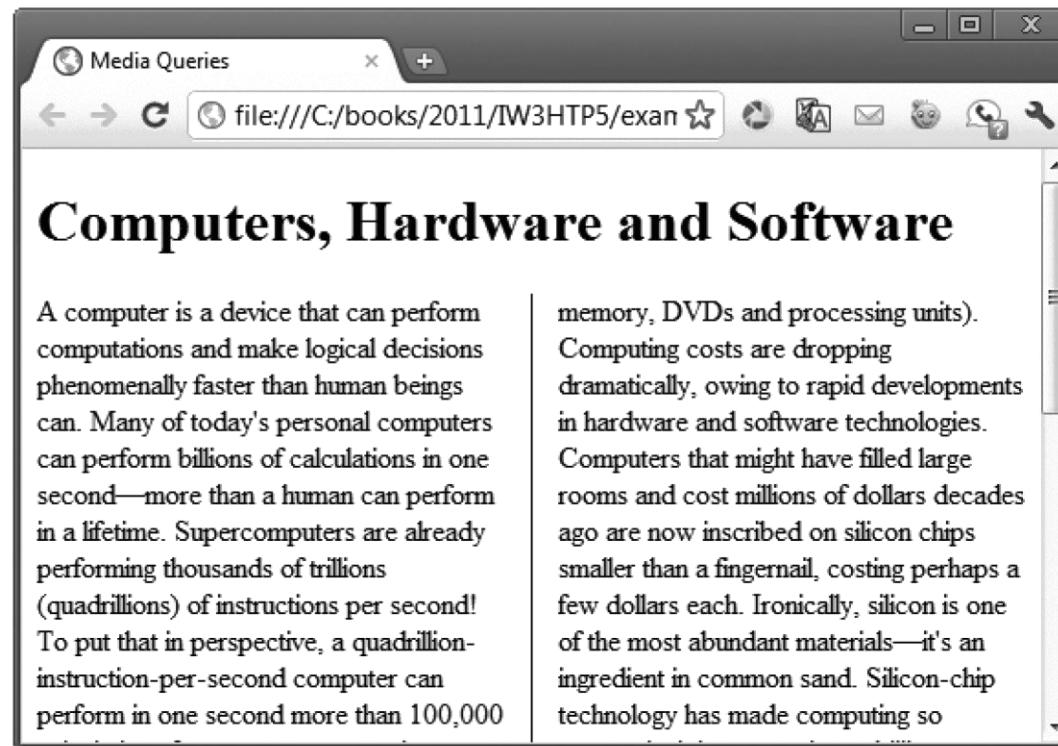
**Fig. 5.18** | Using media queries to reformat a page based on the device width. (Part 5 of 8.)

- a) Styles for smartphones  
with screen widths  
480px or smaller



**Fig. 5.18 |** Using media queries to reformat a page based on the device width. (Part 6 of 8.)

b) Styles for devices with screen widths of 481px to 1024px



**Fig. 5.18** | Using media queries to reformat a page based on the device width. (Part 7 of 8.)

c) Styles for devices with screen widths of 1024px or greater

The screenshot shows a web browser window with the title bar "Media Queries". The address bar displays the URL "file:///C:/books/2011/IW3HTP5/examples/ch05/fig05\_18/mediaqueries.html tml". The main content area contains the following text:

## Computers, Hardware and Software

A computer is a device that can perform computations and make logical decisions phenomenally faster than human beings can. Many of today's personal computers can perform billions of calculations in one second—more than a human can perform in a lifetime. Supercomputers are already performing thousands of trillions (quadrillions) of instructions per second! To put that in perspective, a quadrillion-instruction-per-second computer can perform in one second more than 100,000 calculations for every person on the planet! And—these "upper limits" are growing quickly!

Computers process data under the control of sets of instructions called computer programs. These programs guide the computer through orderly sets of actions specified by people called computer

programmers. The programs that run on a computer are referred to as software. In this book, you'll learn today's key programming methodology that's enhancing programmer productivity, thereby reducing software-development costs—object-oriented programming.

A computer consists of various devices referred to as hardware (e.g., the keyboard, screen, mouse, hard disks, memory, DVDs and processing units). Computing costs are dropping dramatically, owing to rapid developments in hardware and software technologies. Computers that might have filled large rooms and cost millions of dollars decades ago are now inscribed on silicon chips smaller than a fingernail, costing perhaps a few dollars each. Ironically, silicon is one of the most abundant materials—it's an ingredient in common sand. Silicon-chip technology has made computing so economical that more than a billion general-purpose computers are in use worldwide, and this is expected to double in the next few years.

Computer chips (microprocessors) control countless devices. These embedded systems include anti-lock brakes in cars, navigation systems, smart home appliances, home security systems, cell phones and smartphones, robots, intelligent traffic intersections, collision avoidance systems, video game controllers and more. The vast majority of the microprocessors produced each year are embedded in devices other than general-purpose computers.

© 2012 by Pearson Education, Inc. All Rights Reserved.

**Fig. 5.18 |** Using media queries to reformat a page based on the device width. (Part 8 of 8.)

## 5.17 Media Queries (cont.)

- The **@media rule** is used to determine the type and size of device on which the page is rendered
- When the browser looks at the rule, the result is either true or false
- The rule's styles are applied only if the result is true
- First, we use the **@media rule** to determine whether the page is being rendered on a handheld device (e.g., a smartphone) with a max-width of 480px, or a device with a screen that has a max-device-width of 480px, or on a screen having max-width of 480px
- If this is true, we set the column-count to 1
  - Render in a single column of 480px or less

## 5.17 Media Queries (cont.)

- If the condition of the first **@media rule** is false, a second **@media rule** determines whether the page is being rendered on devices with a min-width of 481px and a max-width of 1024px
- If this condition is true, we set the column-count to 2
  - The column-gap (the space between columns) to 30px and the column-rule (the vertical line between the columns) to 1px outset black

## 5.17 Media Queries (cont.)

- If the conditions in the first two **@media rules** are false, we use a third **@media rule** to determine whether the page is being rendered on devices with a min-width of 1025px
- If the condition of this rule is true, we set the column-count to 3
  - The column-gap to 30px (lines 39–40) and the column-rule to 1px outset black

## 5.18 Web Resources

- W3C home page for CSS3
  - <http://www.w3.org/Style/CSS/>
- The Deitel CSS3 Resource Center includes links to tutorials, examples, the W3C standards documentation
  - <http://www.deitel.com/css3/>
- Others
  - <http://layerstyles.org>
  - <http://www.colorzilla.com/gradient-editor/>
  - <http://css3generator.com/>
  - <http://css3please.com/>

## 5.18 Web Resources (cont.)

- Sites that help you generate cross-browser CSS3 code
  - <http://findmebyip.com/litmus/>
- Find the CSS3 features that are supported by each of the major browsers
  - <http://cssprefixer.appspot.com/>
- The CSSPrefixer tool helps you add vendor prefixes to your CSS3 code
  - <http://css-tricks.com/examples/HSLaExplorer/>