

# Box Properties

When you complete this chapter, you will be able to:

- ◎ Understand the CSS visual formatting model
- ◎ Use the CSS box model
- ◎ Apply the margin properties
- ◎ Apply the padding properties
- ◎ Apply the border properties
- ◎ Use the page layout box properties
- ◎ Create a simple page layout

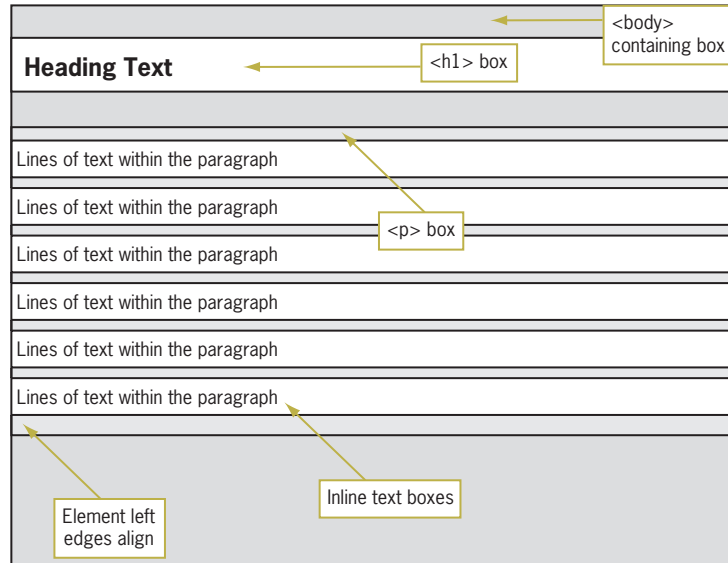
In this chapter, you will explore the CSS box properties. These properties let you control the margin, padding, and border characteristics of block-level elements. To understand how these properties work, you will first learn about the CSS visual formatting model and the box model. These models control the way content is displayed on a Web page. Then you will learn about the margin, padding, and border properties and how you can use them to enhance the display of content in the browser. Finally, you will see how the special box properties—width, height, float, and clear—let you create containers for content that you can position on your Web page. These box properties will become the basis for the page layout techniques you will learn about in the next chapter.

## Understanding the CSS Visual Formatting Model

The **CSS visual formatting model** describes how the element content boxes should be displayed by the browser, for example, whether scroll bars appear and how text is wrapped based on the browser window size. The visual formatting model is based on the hierarchical structure of the HTML document and the element display type. In HTML, elements fall into two primary box types:

- *Block*—Block-level boxes appear as blocks such as paragraphs. Block elements can contain other block elements or inline boxes that contain the element content.
- *Inline*—Inline-level boxes contain the content within the block-level elements. They do not form new blocks of content.

Figure 6-1 shows three different block-level elements: `<body>`, `<h1>`, and `<p>`. The `<h1>` and `<p>` elements contain inline content boxes.



**Figure 6-1** The CSS visual formatting model

Figure 6-1 also shows that parent elements contain child elements. The parent element is called the **containing box**. You can see that `<body>` is the containing box for the elements of a Web page. All other elements' boxes reside within `<body>`, some of which may contain their own child boxes. In Figure 6-1, the `<body>` element is the containing box for the `<h1>` and `<p>` elements. The `<p>` element is the containing box for the inline text that comprises the paragraph text. Inline text boxes are split as necessary to fit the dimensions of the containing box and to wrap text to the next line.

CSS lets you specify margin, border, and padding values for all block-level elements. In some instances, the values you specify depend on the containing box that is the parent of the element you want to affect. For example, if you choose a percentage value for a margin, the percentage value is based on the containing box. In Figure 6-1, a 10% margin value for the `<p>` element would create margins that are 10% of the width of the containing box, in this case, the `<body>` element.

CSS lets you change the display type of any box, either block or inline, with the `display` property, described next.

## Specifying the Display Type

### display property description

Value: block | inline | list-item | none | run-in | inline-block | table  
| inline-table | table-row-group | table-header-group |  
table-footer-group | table-row | table-column-group |  
table-column | table-cell | table-caption | none

Initial: inline

Applies to: all elements

Inherited: no

Percentages: N/A

The CSS display property determines how the browser displays an element. The display property values let you create block-level, inline, and list type elements. It also specifies values for a CSS table model, such as *table* and *table-row*, which are currently not supported by all browsers. The two most commonly used values are *block* and *inline*.

The display property is often used to create horizontal navigation lists, which you will learn about in Chapter 9. Here is a simple example that uses the display property with an `<li>` selector to make the list items appear horizontally across the page:

```
li {
  display: inline;
  list-style-type: none;
}
```

The list-style-type property hides the bullets that are normally displayed with an unordered list, as you saw in Chapter 5. The style rule applies to the following HTML code for an unordered list:

```
<ul>
  <li><a href="url">Home</a></li>
  <li><a href="url">Search</a></li>
  <li><a href="url">Products</a></li>
  <li><a href="url">Contact Us</a></li>
  <li><a href="url">Support</a></li>
  <li><a href="url">Downloads</a></li>
</ul>
```

The result of this style rule are list elements that can be used to create a navigation bar as shown in Figure 6-2.



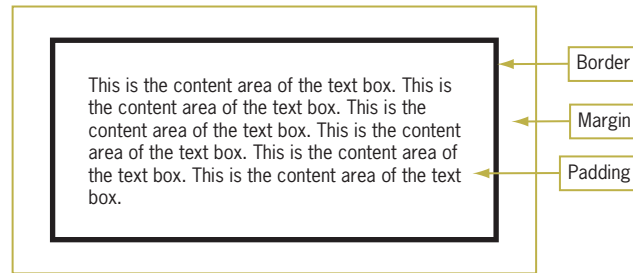
The display property *none* value lets you hide an element so that it is not displayed in the browser. This can be useful in a Web page where you want to display only some of the information on a page, or with JavaScript to create elements that can be displayed as a result of user action.



**Figure 6-2** Horizontal list created with the display property

## Using the CSS Box Model

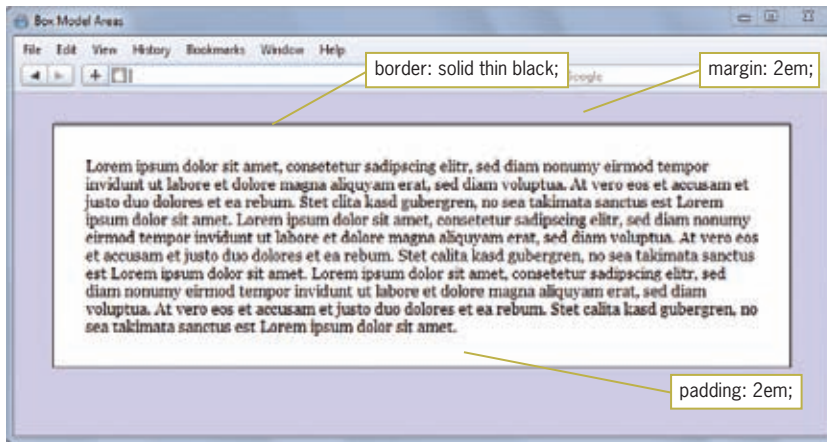
The CSS **box model** describes the rectangular boxes that contain content on a Web page. Each block-level element you create is displayed in the browser window as a box with content. Each content box can have margins, borders, and padding, as shown in Figure 6-3.



**Figure 6-3** CSS box model

As Figure 6-3 illustrates, the content box is the innermost box, surrounded by the padding, border, and margin areas. The padding area has the same background color as the content element, but the margin area is always transparent. The border separates the padding and margin areas.

Figure 6-4 shows the box model areas in a paragraph element. This paragraph has 2em padding, a thin black border, and 2em margins. Notice that the margin area is transparent.



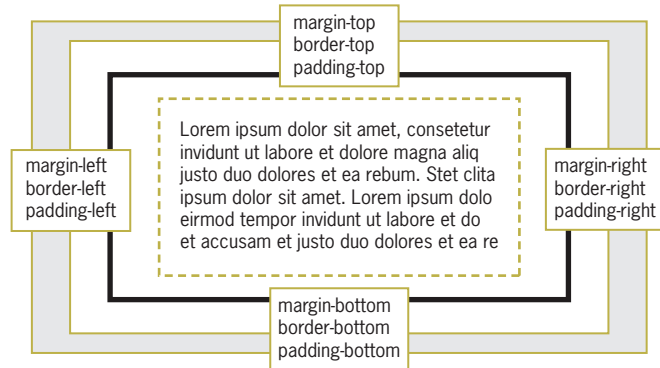
**Figure 6-4** CSS box model areas in a `<p>` element

The following code shows the style rule for the paragraph in Figure 6-4:

```
p {
  margin: 2em;
  padding: 2em;
  border: solid thin black;
  background-color: white;
}
```

The margin and padding properties set the length to 2em for all four sides of the box. The border property sets the border-style to solid, the border-weight to thin, and the border-color to black. The background-color property sets the paragraph background color to white.

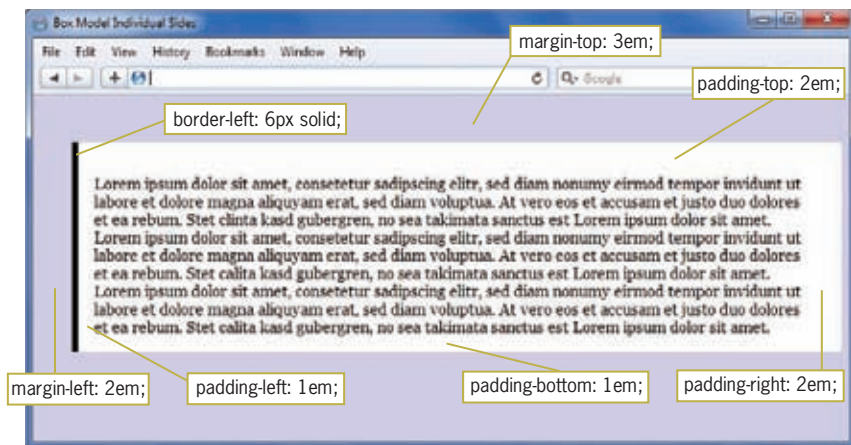
CSS lets you specify margin, padding, and border properties individually for each side of the box. Figure 6-5 shows that each area has a left, right, top, and bottom side. Each one of the sides can be referred to individually. If the browser supports the individual properties, you can select, for example, the padding-bottom, border-top, or margin-left properties if you prefer.



**Figure 6-5** CSS box model individual sides

Figure 6-6 shows a paragraph with a variety of box property settings. As you can see, CSS gives you complete control over the individual white space areas and borders in block elements. The style rule for the paragraph is:

```
p {
  background-color: white;
  border-left: 6px solid;
  margin-left: 2em;
  margin-top: 3em;
  padding-top: 2em;
  padding-right: 2em;
  padding-bottom: 1em;
  padding-left: 1em;
}
```



**Figure 6-6** CSS box model individual sides in a <p> element

## Measurement Values

The margin, border, and padding properties let you state two types of measurement values—either a length or a percentage. (For a full discussion of measurement values, see Chapter 5.) If you use a percentage value, the percentage is based on the width of the containing box, as described earlier. If you choose a length, you have to decide whether to use an absolute or relative value. As with font sizes, you are better off using relative units such as ems or percentages when you are stating margin, border, or padding sizes. The relative measurement values let you build scalable Web pages. Many Web designers prefer pixel measurements for certain properties such as borders when they know the size of the border line will remain constant across different monitors and devices. Pixel values for padding and margins are often specified in fixed page designs where the dimensions of the layout are constant.



Always use relative measurement values, such as ems or percentages, if you want your Web pages to adapt to different browser sizes or user-applied font sizes.

## Applying the Margin Properties

The **margin properties** let you control the margin area of the box model. Margins are always transparent, showing the background of their containing element. You can use margins to enhance the legibility of text, create indented elements, and add white space around images.

## Specifying Margins

### margin, margin-top, margin-right, margin-bottom, margin-left, property description

Value: <length> | <percentage>

Initial: 0

Applies to: all elements

Inherited: no

Percentages: refer to width of containing block

The margin properties let you specify margins with either a length or percentage value. You can use the margin property to state one value for all four margin sides, or you can specify settings for individual margins. The following style rule sets all four margins in a paragraph to 2em.

```
p {margin: 2em;}
```

You can also choose to specify individual margin properties that let you control each margin: margin-left, margin-right,



margin-top, and margin-bottom. The following style rules set the left and right margins for a paragraph element:

```
p {
    margin-left: 2em;
    margin-right: 3em;
}
```

### Margin Property Shorthand Notation

The shorthand notation syntax lets you state individual margin settings within the same rule. The individual margin settings change based on the number of values and their order within the rule. Table 6-1 shows how the syntax works.

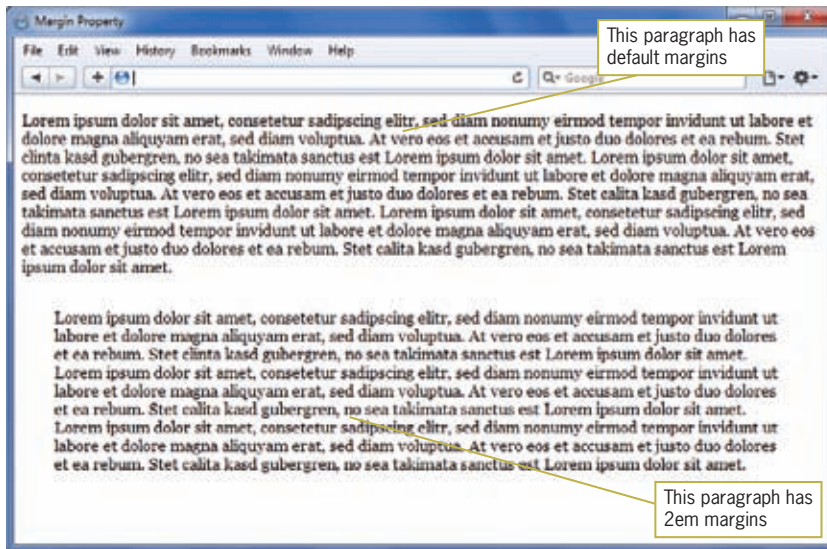
Number of Values	Example	Description
1 value	p {margin: 1em;}	All four margins are 1em
2 values	p {margin: 1em 2em;}	Top and bottom margins are 1em Left and right margins are 2em
3 values	p {margin: 1em 2em 3em;}	Top margin is 1em Right and left margins are 2em Bottom margin is 3em
4 values	p {margin: 1em 2em 3em 4em;}	Top margin is 1em Right margin is 2em Bottom margin is 3em Left margin is 4em

Table 6-1

Shorthand Notation for the margin Property

It is your choice whether you want to use the shorthand notation or to state the margin properties individually. Some designers prefer the more specific and easy to read individual margin properties rather than the shorthand notation. Others prefer the brief syntax of the shorthand notation. Either way, set a coding convention for your Web site and use it consistently.

Figure 6-7 shows two paragraph elements. The first paragraph has the default margin setting; the second has the margin set to 2em. Notice that the increased margins enhance the legibility of the text.



**Figure 6-7** Using the margin property

## Negative Margins

Margins are unusual because you can use negative values. You can set negative margin values to achieve special effects. For example, you can remove the default margins by setting a negative value or overlap elements on the page.

Figure 6-8 shows two `<h1>` elements, one with the default margins, and one with the bottom margin set to a negative value. The following rule sets a negative value of 20 pixels for the element's bottom margin:

```
h1 {margin-bottom: -20px;}
```



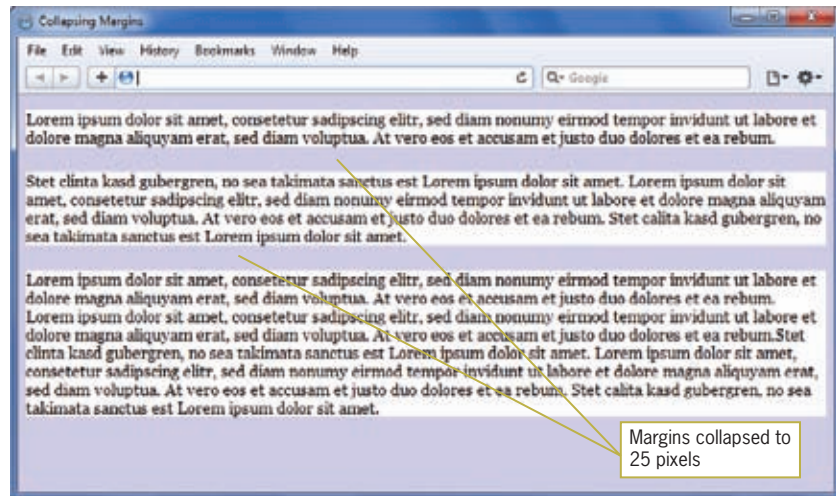
**Figure 6-8** `<h1>` element with a negative bottom margin

## Collapsing Margins

To ensure that the spacing between block-level elements is consistent, the browser collapses the vertical margins between elements. The vertical margins are the top and bottom element margins. The browser does not add the value of the two, but picks the greater value and applies it to the space between the adjoining elements. To illustrate this, consider the following rule:

```
p {
    margin-top: 15px;
    margin-bottom: 25px;
}
```

If the browser did not collapse the vertical margins, the paragraphs would have 40 pixels of space between each paragraph. Instead, the browser collapses the margin. Following the CSS convention, the browser sets the vertical margin between paragraphs to 25 pixels, the greater of the two values. Figure 6-9 shows how the browser maintains the space between the paragraphs.



**Figure 6-9** Browser collapses vertical margins

## Zeroing Margins

You can set margin values to zero if you want to remove the default margin spacing that is built into the browser. This technique is useful if you have problems getting your page designs to look consistent across multiple browsers and display devices. By setting the values in the body section to zero, the settings are

inherited for all elements on the page. Then you can set specific margin settings for each element as necessary for your design. The style rule for zeroing margins looks like this:

```
body {margin: 0; padding: 0;}
```

This technique is also handy when you want to place an image against the side of the browser window as shown in Figure 6-10.



**Figure 6-10** Zeroing margins to place images against the edge of the window

## Applying the Padding Properties

The CSS **padding properties** let you control the padding area in the box model. The padding area is between the element content and the border. The padding area inherits the background color of the element, so if a `<p>` element has a white background, the padding area will be white as well.



If you zero margins for the entire page, you must explicitly set margins for any elements that you do not want to place against the edge of the browser window.

**padding, padding-top, padding-right, padding-bottom, padding-left property descriptions**

Value: &lt;length&gt; | &lt;percentage&gt;

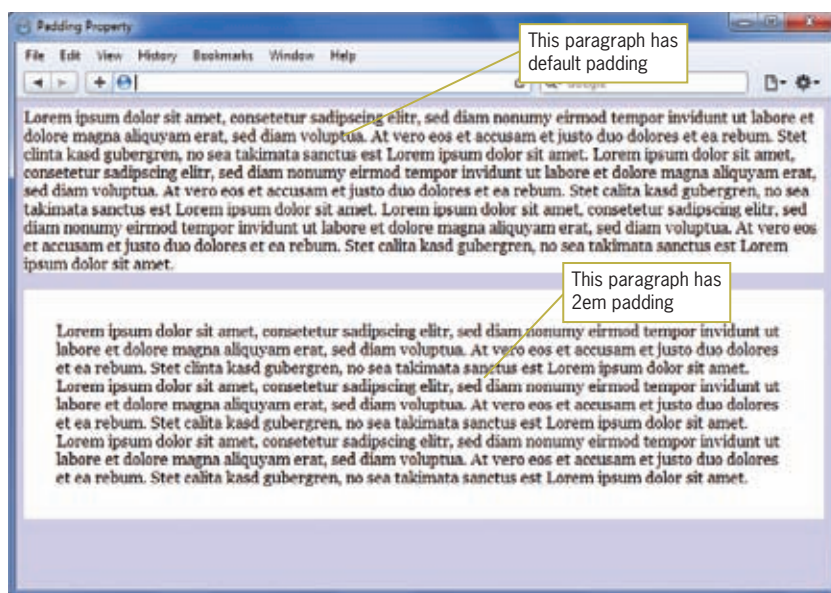
Initial: 0

Applies to: all elements

Inherited: no

Percentages: refer to width of containing block

Figure 6-11 shows how adding padding to an element increases the space between the content and the edge of the element box.



**Figure 6-11** Default padding and 2em padding

Like the margin property, the padding property lets you state one value for all four margin sides, or you can specify settings for individual padding areas. You can specify either a length or a percentage value. Unlike margins, you cannot collapse the padding area or set negative padding values. The following style rule sets all four padding sides in a paragraph to 2em.

```
p {padding: 2em;}
```

You can also choose to specify individual padding properties, which let you control each padding area: padding-left,



padding-right, padding-top, and padding-bottom. The following style rules set the left and right padding for a paragraph element:

```
p {
    padding-left: 2em;
    padding-right: 3em;
}
```

## Padding Property Shorthand Notation

The shorthand notation syntax lets you state individual padding settings. Like the margin shorthand property described earlier, the individual padding settings change based on the number of values and their order within the rule. Table 6-2 shows how the syntax works.

Number of Values	Example	Description
1 value	p {padding: 1em;}	Top, bottom, left, and right padding are 1em
2 values	p {padding: 1em 2em;}	Top and bottom padding are 1em Left and right padding are 2em
3 values	p {padding: 1em 2em 3em;}	Top padding is 1em Right and left padding are 2em Bottom padding is 3em
4 values	p {padding: 1em 2em 3em 4em;}	Top padding is 1em Right padding is 2em Bottom padding is 3em Left padding is 4em

**Table 6-2** Shorthand Notation for the padding Property

The following style rule sets the top and bottom padding areas for a paragraph, along with complementing borders, a white background, and margins to offset the paragraph from the browser sides:

```
p {
    padding-top: 2em;
    padding-bottom: 2em;
    border-top: solid thin black;
    border-bottom: solid thin black;
    background-color: white;
    margin: 2em;
}
```

As Figure 6-12 shows, the paragraph now has the default left and right padding with 2em top and bottom padding.

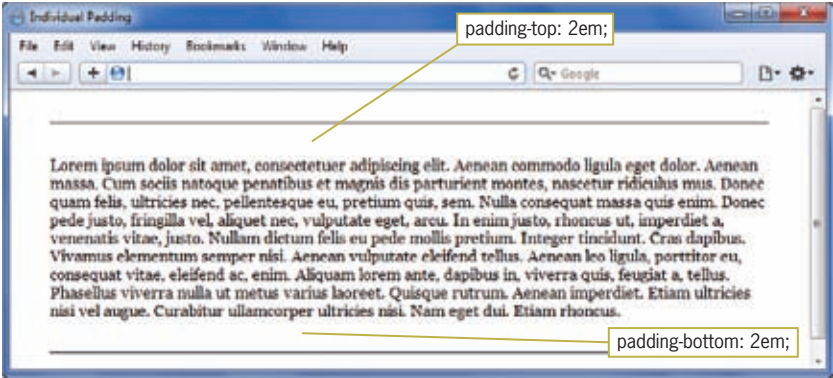


Figure 6-12 Using the individual padding properties

# Applying the Border Properties

The **border properties** let you control the appearance of borders around elements. The border area resides between the margin and padding. Border properties for each border side let you specify the style, width, and color of each border. You will most likely use the five border shorthand properties, which include:

- border
- border-left
- border-right
- border-top
- border-bottom

These shorthand properties let you state border-style, border-color, and border-width for all four borders or for any of the individual sides of the box. However, you can also state much more specific borders by using the border properties separately. Table 6-3 lists the entire range of 20 border properties.

Description	Property Name		
Overall shorthand property	border		
Individual side shorthand properties	border-left, border-top, border-right, border-bottom		
Specific shorthand property	border-style	border-width	border-color
Individual properties	border-left-style	border-left-width	border-left-color
	border-right-style	border-right-width	border-right-color
	border-top-style	border-top-width	border-top-color
	border-bottom-style	border-bottom-width	border-bottom-color

Table 6-3 Border Properties

Before you can use the shorthand properties, you must first understand the three border characteristics—border-style, border-color, and border-width.

## Specifying Border Style

### **border-style, border-top-style, border-right-style, border-bottom-style, border-left-style** property description

Value: <border-style>

Initial: none

Applies to: all elements

Inherited: no

Percentages: N/A

The border-style is the most important border property because it must be stated to make a border appear. The border-style property lets you choose from one of the following border style keywords:

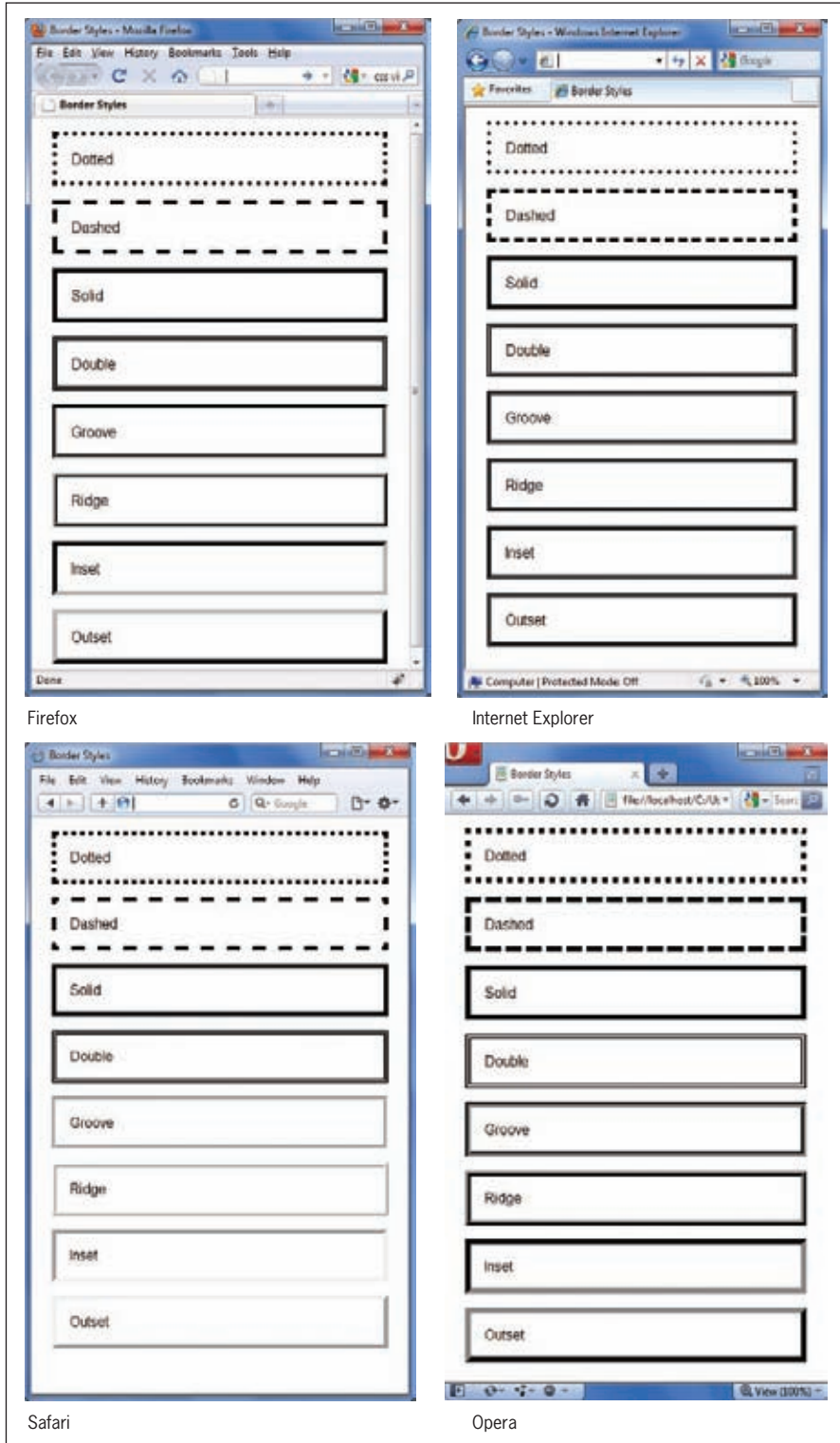
- *none*—No border on the element; this is the default setting
- *dotted*—Dotted border
- *dashed*—Dashed border
- *solid*—Solid line border
- *double*—Double line border
- *dot-dash*—Alternating dots and dashes (CSS3 value)
- *dot-dot-dash*—Two dots and a dash (CSS3 value)
- *wavy*—Wavy line border (CSS3 value)
- *groove*—Three-dimensional border that appears to be engraved into the page
- *ridge*—Three-dimensional border that appears to be embossed (or extend outward from the page)
- *inset*—Three-dimensional border that appears to set the entire box into the page
- *outset*—Three-dimensional border that appears to extend the entire box outward from the page

The following code shows an example of the border-style property in use:

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

Figure 6-13 shows examples of the borders. As you can see, each browser displays the border styles differently. If a border you specify is not supported, the border defaults to solid.





**Figure 6-13** Border styles in Firefox, Internet Explorer, Safari, and Opera

## Individual Border Styles

You can also specify individual borders styles with the following border-style properties:

- border-left-style
- border-right-style
- border-top-style
- border-bottom-style

These properties let you single out one border and apply a style. The following rule applies only to the left border of the element:

```
p {border-left-style: double;}
```

## Border-Style Property Shorthand Notation

The shorthand notation syntax lets you state individual border-style settings. The individual border style settings change based on the number of values and their order within the rule. Table 6-4 shows how the syntax works.

Number of Values	Example	Description
1 value	p {border-style: solid;}	All four borders are solid
2 values	p {border-style: solid double;}	Top and bottom borders are solid Left and right borders are double
3 values	p {border-style: solid double dashed;}	Top border is solid Right and left borders are double Bottom border is dashed
4 values	p {border-style: solid double dashed dotted;}	Top border is solid Right border is double Bottom border is dashed Left border is dotted

**Table 6-4** Shorthand Notation for the border-style Property

Of course, if you examine the rules in Table 6-4, you can see they will create odd effects. For example, a paragraph with a different border style for each side is not a common design technique. Remember to use restraint and keep the user in mind when working with border styles.

## Specifying Border Width

### **border-width, border-top-width, border-right-width, border-bottom-width, border-left-width property description**

Value: thin | medium | thick | <length>

Initial: medium

Applies to: all elements

Inherited: no

Percentages: N/A

The border-width property lets you state the width of the border with either a keyword or a length value. You can use the following keywords to express width:

- thin
- medium (default)
- thick

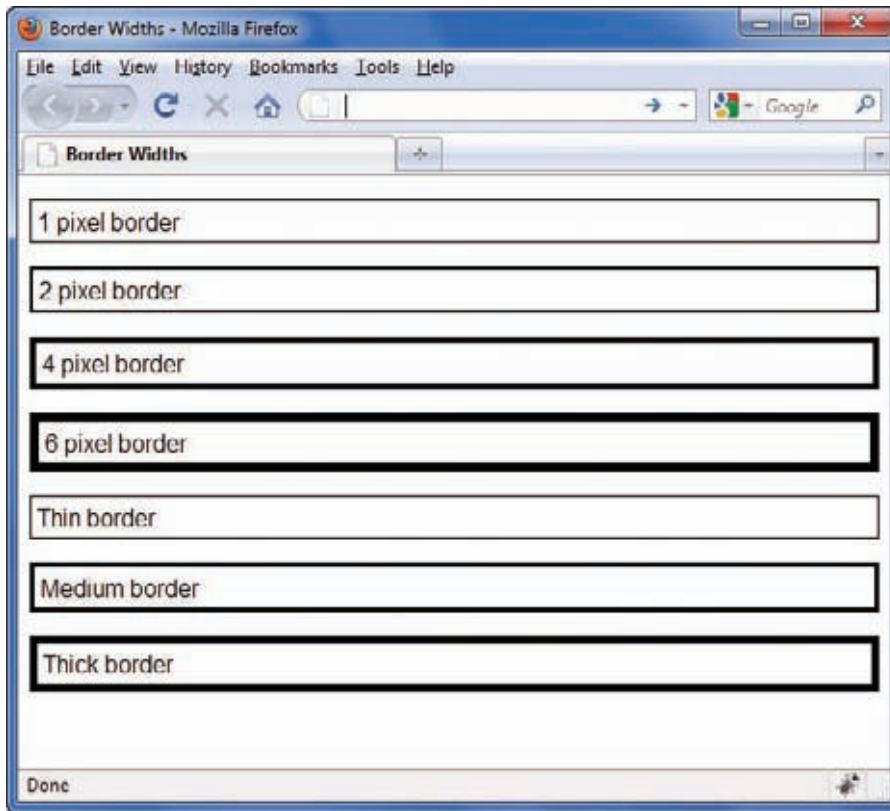
The width of the rule when you use these keywords is based on the browser. The length values let you state an absolute or relative value for the border; percentages are not allowed. Using a length value lets you create anything from a hairline to a very thick border. The following code shows an example of the border-width property in use:

```
p {border-width: 1px; border-style: solid;}
```

Remember that the border is not displayed unless the border-style property is stated. Figure 6-14 shows examples of different border widths.



When designing layouts, it is often handy to turn the borders on for an element, even though you may turn it off for the finished Web page. The border lets you see the dimensions of the element on the page. Remember that adding borders to an element increases the element's width.



**Figure 6-14** Different border widths

### *Individual Border Widths*

You can also specify individual borders widths with the following border-width properties:

- border-left-width
- border-right-width
- border-top-width
- border-bottom-width

These properties let you single out one border and apply a width. The following rule applies only to the left border of the element:

```
p {border-left-width: thin;}
```

### Border-Width Property Shorthand Notation

The shorthand notation syntax lets you state individual border-width settings. The individual border width settings change based on the number of values and their order within the rule. Table 6-5 shows how the syntax works.

Number of Values	Example	Description
1 value	p {border-width: 1px;}	All four borders are 1 pixel wide
2 values	p {border-width: 1px 2px;}	Top and bottom borders are 1 pixel wide Left and right borders are 2 pixels wide
3 values	p {border-width: 1px 2px 3px;}	Top border is 1 pixel wide Right and left borders are 2 pixels wide Bottom border is 3 pixels wide
4 values	p {border-width: 1px 2px 3px 4px;}	Top border is 1 pixel wide Right border is 2 pixels wide Bottom border is 3 pixels wide Left border is 4 pixels wide

Table 6-5

 Shorthand Notation for the border-width Property

### Specifying Border Color

border-color, border-top-color, border-right-color,  
border-bottom-color, border-left-color property description

Value: <color>  
Applies to: all elements  
Inherited: no  
Percentages: N/A

The border-color property lets you set the color of the element border. The value can be either a hexadecimal value, RGB value, or one of the 16 predefined color names shown in the following list:

Aqua	Navy	Gray	Silver
Black	Olive	Green	Tea
Blue	Purple	Lime	White
Fuchsia	Red	Maroon	Yellow



You can learn more about color values in Chapter 8.

To set a border color, use the property as shown in the following rule:

```
p {border-color: red; border-width: 1px; border-style: solid;}
```

The default border color is the color of the element content. For example, the following style rule sets the element text color to red. The border is also red because a border color is not specified.

```
p {  
    color: red;  
    font: 12pt arial;  
    border: solid;  
}
```

### *Individual Border Colors*

You can also specify individual border colors with the following border-color properties:

- border-left-color
- border-right-color
- border-top-color
- border-bottom-color

These properties let you single out one border and apply a color. The following property applies only to the left border of the element:

```
p {border-left-color: red; border-style: solid;}
```

### *Border-Color Property Shorthand Notation*

The shorthand notation syntax lets you state individual border-color settings. The individual border color settings change based on the number of values and their order within the rule. You can also choose to state individual border colors in the border-color property. The individual border color settings change based on the order within the rule. Table 6-6 shows how the syntax works.

Number of Values	Example	Description
1 value	p {border-color: black;}	All four borders are black
2 values	p {border-color: black red;}	Top and bottom borders are black Left and right borders are red
3 values	p {border-color: black red green;}	Top border is black Right and left borders are red Bottom border is green
4 values	p {border-color: black red green blue;}	Top border is black Right border is red Bottom border is green Left border is blue

Table 6-6

Shorthand Notation for the border-color Property

## Using the Border Shorthand Properties

The shorthand properties are the most common and easiest way to express border characteristics. When you use these shorthand properties, you are stating the style, color, and width of the border in one concise rule.

Border, border-top, border-right, border-bottom, border-left property description

Value: <border-width> | <border-style> | <color>

Initial: see individual properties

Applies to: all elements

Inherited: no

Percentages: N/A

The border property lets you state the properties for all four borders of the element. You can state the border-width, border-style, and border-color in any order. Border-style must be included for the border to appear. If you do not include border-width, the width defaults to medium. If you do not include border-color, the border appears in the same color as the element. The following example rules show different uses of the border property.

The following rule sets the border-style to solid. The border-weight defaults to medium. The border-color is the same as the color of the <p> element; because no color is stated, the border color is black.

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

The following rule sets the border-style to solid. The border-weight is 1 pixel. The border-color is red.

```
p {border: solid 1px red;}
```

The following rule sets the border-style to double. The border-weight is thin. The border-color is blue. Notice that the order of the values does not matter.

```
p {border: double blue thin;}
```

These let you state border-style, border-width, and border-color in one statement that selects a single element border. For example, the following rule sets border-style to solid and border-weight to thin for both the left and right borders. Because no color is stated, the borders default to the element color.

```
p {border-left: solid thin; border-right: solid thin;}
```

The following rule sets border-style to double and border-color to red for the top border. Because no border-weight is stated, the weight defaults to medium.

```
p {border-top: double red;}
```

## Specifying Rounded Borders

### **border-radius, border-top-right-radius, border-top-left-radius, border-bottom-right-radius, border-bottom-left-radius**

Value: [ <length> | <percentage> ]

Initial: 0

Applies to: all elements

Inherited: no

Percentages: N/A

The CSS 3 border-radius property lets you create rounded borders on block-level elements, an effect that many Web designers have wanted but were unable to easily create in a standardized way. Although not yet evenly supported by current browsers, the example in Figure 6-15 shows the Chrome browser's version of two styles of rounded corners on a border.



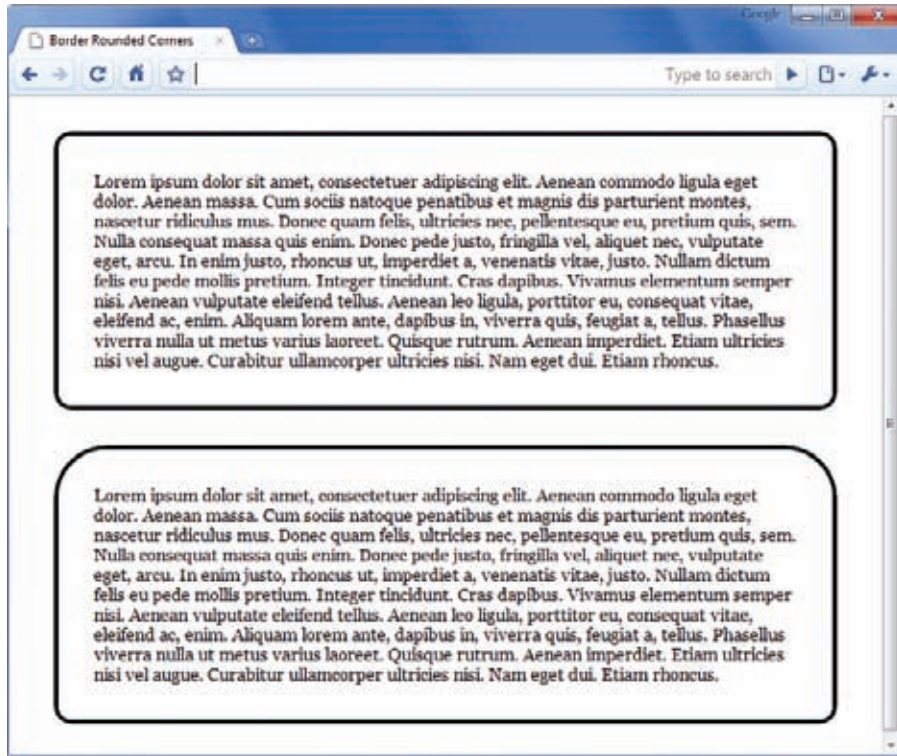


Figure 6-15 Rounded borders


The two length or percentage values determine the radius of each corner. These can be varied to create effects as shown in Figure 6-15.

The following rule sets the radius for all four corners to 1em:

```
border-radius: 1em;
```

You can also use individual properties to set each corner as shown in the bottom paragraph in Figure 6-15.

```
border-top-right-radius: 3em;
border-top-left-radius: 3em;
border-bottom-right-radius: 1em;
border-bottom-left-radius: 1em;
```

 The border-radius properties are CSS3 properties that may not work consistently across all browsers, so test your work carefully.

## Using the Page Layout Box Properties

The **page layout box properties** let you control the dimensions and position of content boxes. These properties are essential to building CSS page layouts, which you will learn about in Chapter 7.

Using these box properties, you can specify the exact shape of a content box and create columns and boxes of content. You can

set minimum and maximum widths and heights as well as control how your boxes are resized based on the size of the browser window. You can also align boxes to the left or right of other elements using the float property and allow text to wrap around images.

In this section, you will learn about the following box properties:

- width, min-width, max-width
- height, min-height, max-height
- float
- clear
- overflow

## Setting Element Width

### width, min-width, max-width property description

Value: <length> | <percentage>

Initial: auto

Applies to: all elements but text inline elements, table rows, and row groups

Inherited: no

Percentages: refer to width of containing block

The width property lets you set the horizontal width of an element using either a length value or a percentage. The percentage value is based on the width of the containing element box. The following is an example of width property usage:

```
div {width: 200px;}
```

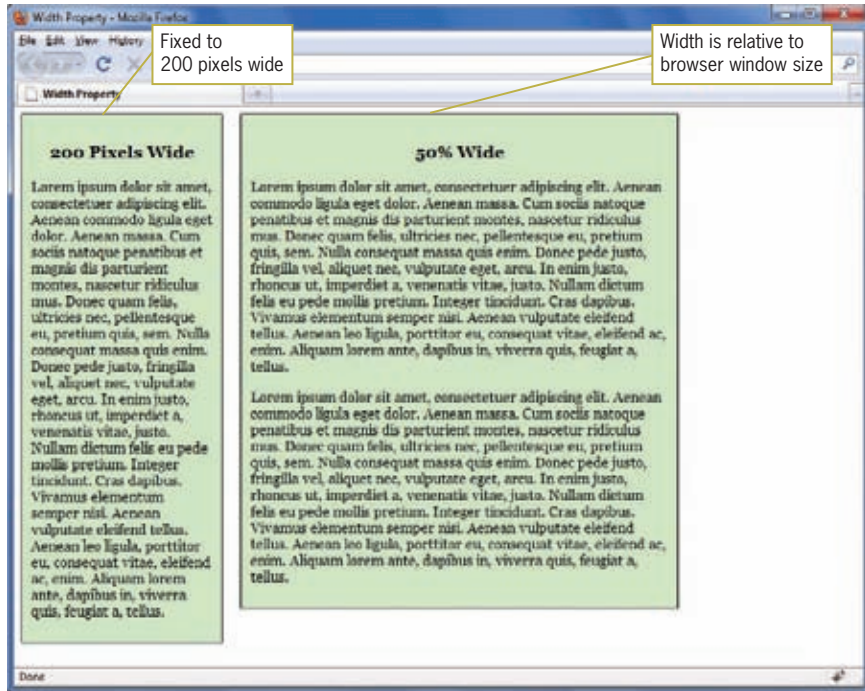
If you use percentages, the content boxes will adapt to the size of the browser or the containing element, allowing you to build flexible page layouts based on the browser size. If you are building fixed dimension layouts, use pixel values for width.

Figure 6-16 shows two <div> elements, one that is 200 pixels wide and one that is 50% wide. Each of the <div> elements in this example contains both headings and paragraphs. The left box is fixed at 200 pixels wide. Its height is determined by the content it contains. The right box is set to a percentage value, which means the box size will change based on the size of the browser window. This box's width will always be 50% of the current window size. If the user resizes the browser, the box width will change.



The <div> element is your best choice for creating content boxes

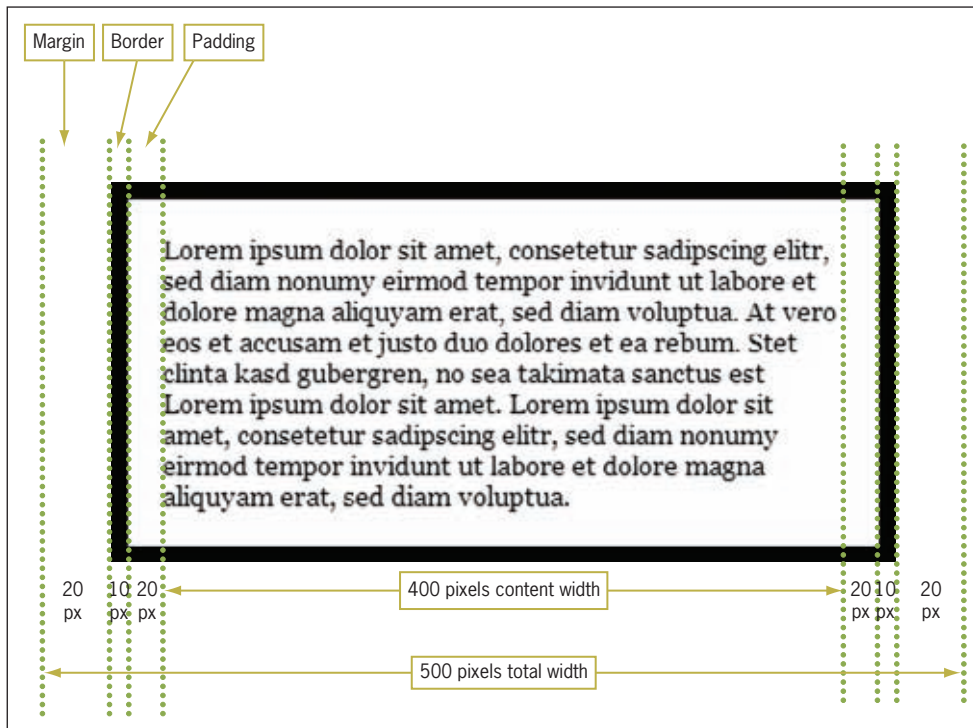
because it can contain other elements such as headings and paragraphs. Class or id names can be applied to the <div> element to apply style rules. As you will see here and in Chapter 7, most Web page layouts are built with <div> elements to create the columns and containers, each of which contains content.



**Figure 6-16** Using the width property

### Calculating Box Model Width

When you specify a width, you are specifying the width of the content only, not the entire element. In Figure 6-17, the paragraph width is 400 pixels as specified by the width property. The actual element width of 500 pixels is the result of adding the padding, border and margin properties to the content width.



**Figure 6-17** Calculating the box model width

## Setting Minimum and Maximum Widths

The `min-width` and `max-width` properties let you determine exactly how wide or narrow you want the width of a box to be. This is helpful when building flexible layouts, as you will see in Chapter 7. In Figure 6-16, the content element on the right will always be 50% of the browser window, but the user may occasionally make a window very narrow or very wide, and you don't want the box to become either a skinny narrow column or overly wide column. You can control the width with `min-width` and `max-width`. In the following example, a page content box is set to 100%, which will fill the browser window. However, the box will not be allowed to shrink to less than 750 pixels or expand to more than 1280 pixels.

```
div.content {
  width: 100%;
  min-width: 750px;
  max-width: 1280px;
}
```

## Setting Element Height

### height property description

Value: <length> | <percentage>

Initial: auto

Applies to: all elements but text inline elements, table columns, and column groups

Inherited: no

Percentages: N/A

The height property lets you set the vertical height of an element. Height should only be used in situations where you know the exact height of the element content, such as an image. At other times, you may need to create a box with specific dimensions for a design. It is a better practice to let the content determine the height of the element.

The height property accepts either a length value or a percentage. The percentage value is based on the height of the containing element box. The following is an example of height property usage:

```
div {height: 150px; width: 300px;}
```

### Setting Minimum and Maximum Height

The min-height and max-height properties let you determine exactly how tall or short you will allow the height of a box to be. These properties work exactly like the min-width and max-width properties described earlier.

## Floating Elements

### float property description

Value: left | right | none

Initial: none

Applies to: all elements except positioned elements and generated content (see Appendix B)

Inherited: no

Percentages: N/A

The float property lets you position an element to the left or right edge of its parent element. You can float an image to the left or right of text, as shown in Figure 6-18.





**Figure 6-18** Floating an image

The style rule for the floating image floats the image to the left and adds a margin to offset the text from the image. The style rule looks like this:

```
img {
    float: left;
    margin-right: 10px;
}
```

The float property can also be used to float a content box to the left or right of text. Used with the width element, you can create a content box as shown in Figure 6-19.

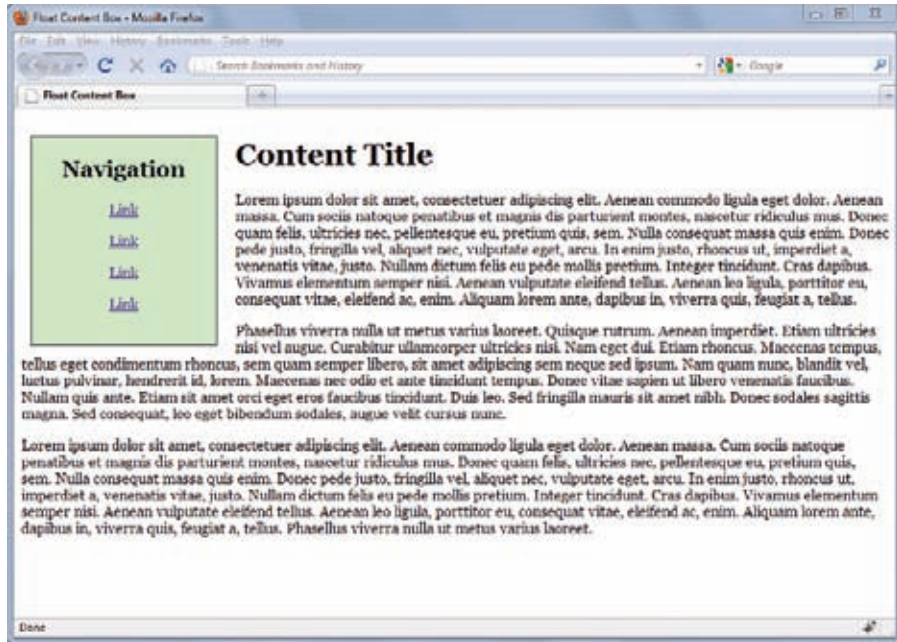


Figure 6-19 Floating content box

The rule for the left-floating content box looks like this:

```
#float {
  width: 200px;
  float: left;
  border: 1px solid black;
  padding-bottom: 20px;
  margin: 0px 20px 10px 10px;
  text-align: center;
  background-color: #cfc;
}
```

The style rule uses an id of *float* as the selector. The width property sets the width of the element to 200 pixels. The float property floats the box to the left of the page. Notice that the rule does not include a height property. The height of the box is determined by the content it contains and the padding values. The margin property states the top, right, bottom, and left margin values.

The floated content box is a `<div>` element that gets the id value of *float*:

```
<div id="float">
  <h2>Navigation</h2>
  <p><a href="url">Link</a></p>
  <p><a href="url">Link</a></p>
  <p><a href="url">Link</a></p>
  <p><a href="url">Link</a></p>
</div>
```

## Clearing Elements

### clear property description

Value: none | left | right | both

Initial: none

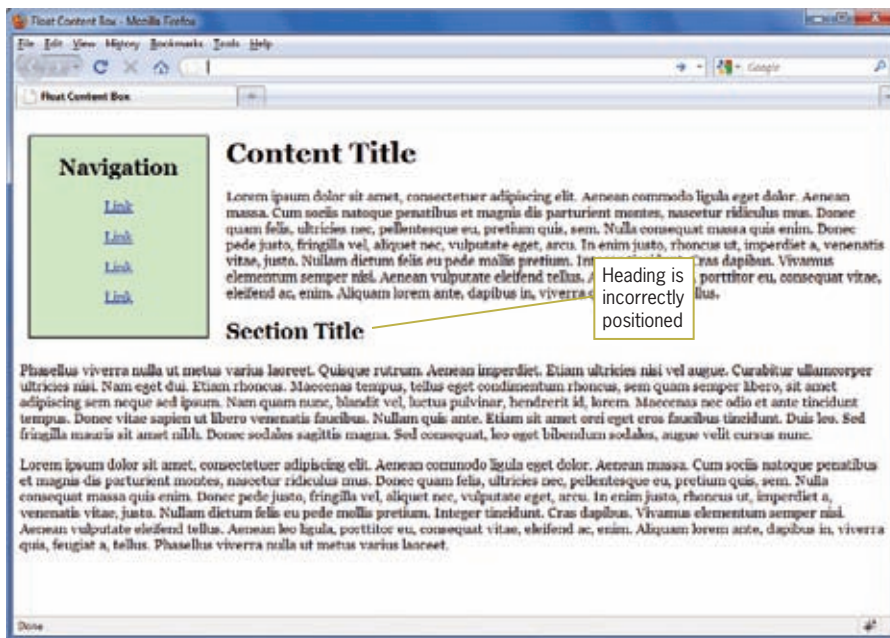
Applies to: block-level elements

Inherited: no

Percentages: N/A

275

The clear property lets you control the flow of text around floated elements. You only use the clear property when you are using the float property. Use the clear property to force text to appear beneath a floated element, rather than next to it. Figure 6-20 shows an example of normal text flow around an element.



**Figure 6-20** Normal text flow around a floating image

This figure shows the left floating content box from the previous example. The text flows down around the image on the right, which is the correct behavior. The second-level heading does not appear in the correct position. It should be positioned beneath the floating content box. To correct this problem, use the clear property. In this instance, the <h2> should be displayed clear of

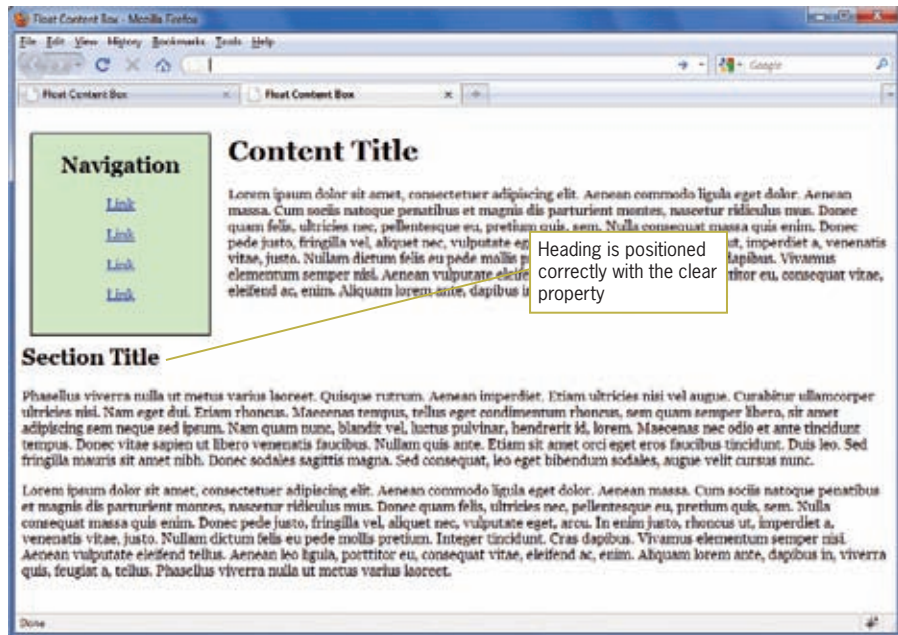


any left-floating images. Add this style rule directly to the <h2> element with the style attribute as follows:

```
<h2 style="clear: left;">
```

Figure 6-21 shows the result of adding the clear property.

276



**Figure 6-21** Using the clear property

Notice that the clear property lets you clear from either left- or right-floating images using the *left* and *right* values. The *both* value lets you control text flow if you have floating images on both the left and right sides of the text.

## Controlling Overflow

### overflow property description

Value: [visible | hidden | scroll | auto]

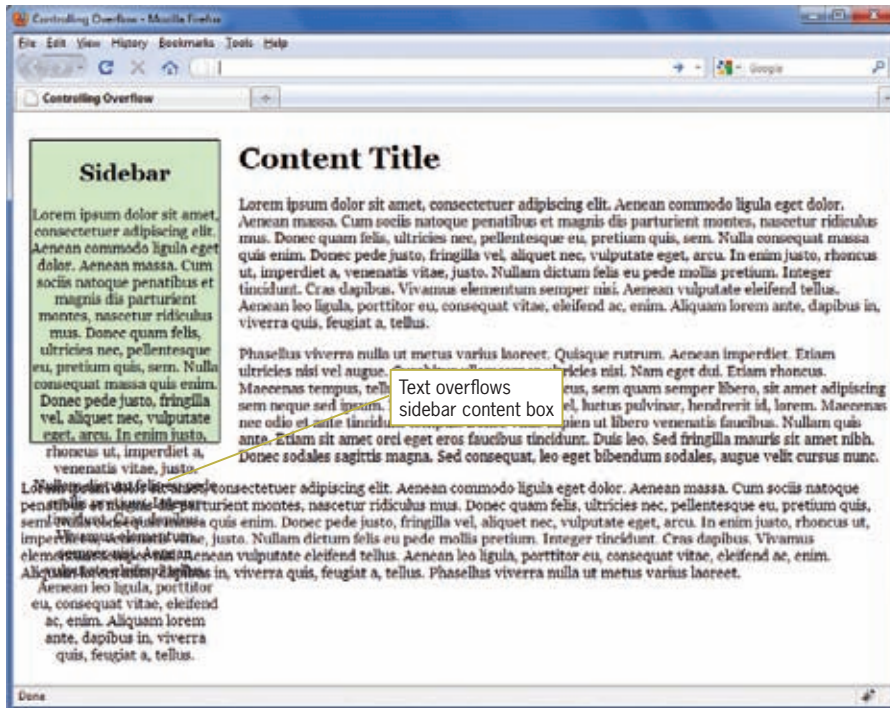
Initial: visible

Applies to: block-level elements

Inherited: no

Percentages: N/A

The overflow property lets you control when content overflows the content box that contains it. This can happen when the content is larger than the area it is designed for, especially when a height property is set for a content element. Figure 6-22 shows text overflow in a content box with fixed width and height.



**Figure 6-22** Text overflows a box with fixed width and height

To solve this problem, you can add an overflow property to the element. You can choose to show scroll bars or to hide the overflow content. In Figure 6-23, you can see the result of adding the overflow property set to auto, which automatically adds a scroll bar when the content overflows the box.

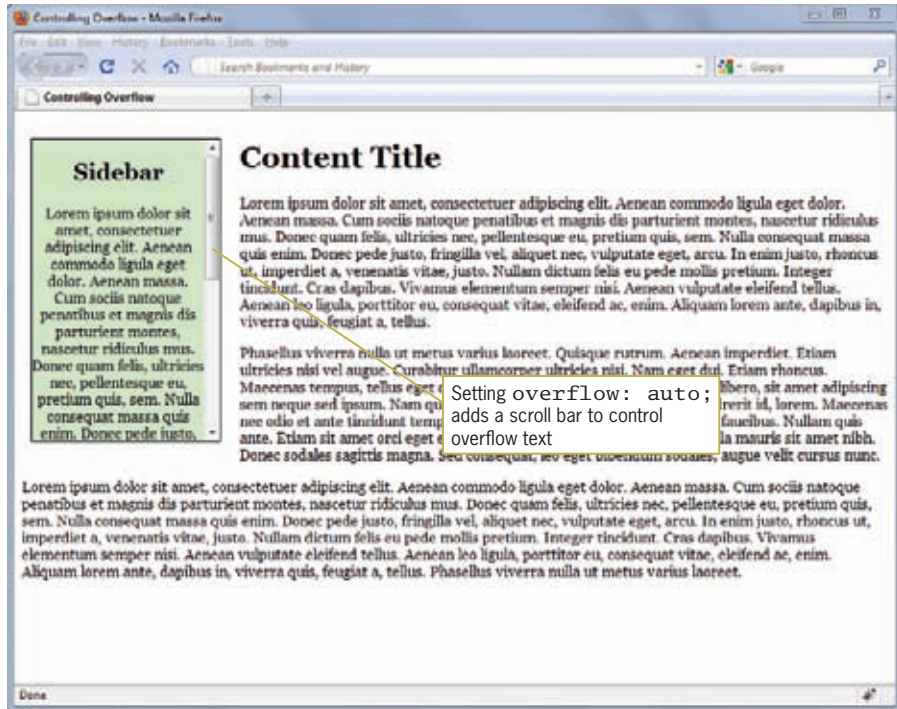


Figure 6-23 Controlling overflow with an auto scrollbar

## Creating Box Shadows

### box-shadow property description

Value: none | <shadow> [ , <shadow> ]\*

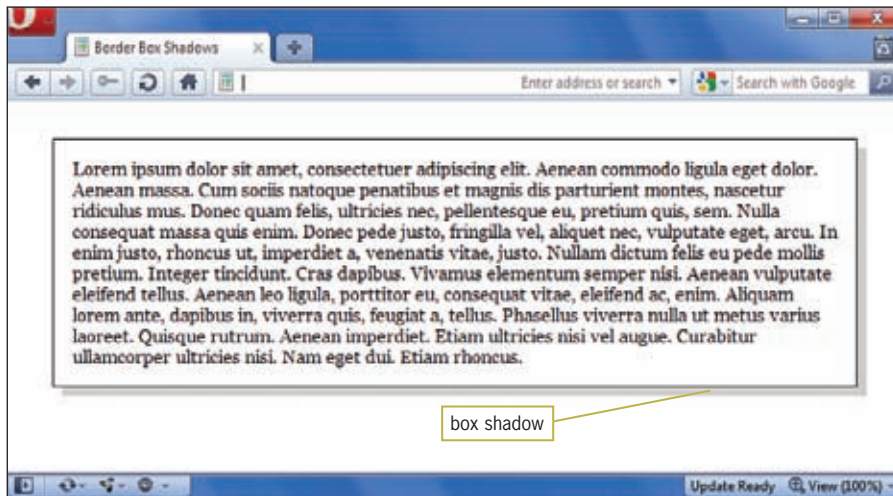
Initial: none

Applies to: all elements

Inherited: no

Percentages: N/A

The box-shadow property lets you add box shadows to an element to create a 3D effect. Figure 6-24 shows this in the Opera browser, which currently is the only browser to support this property.

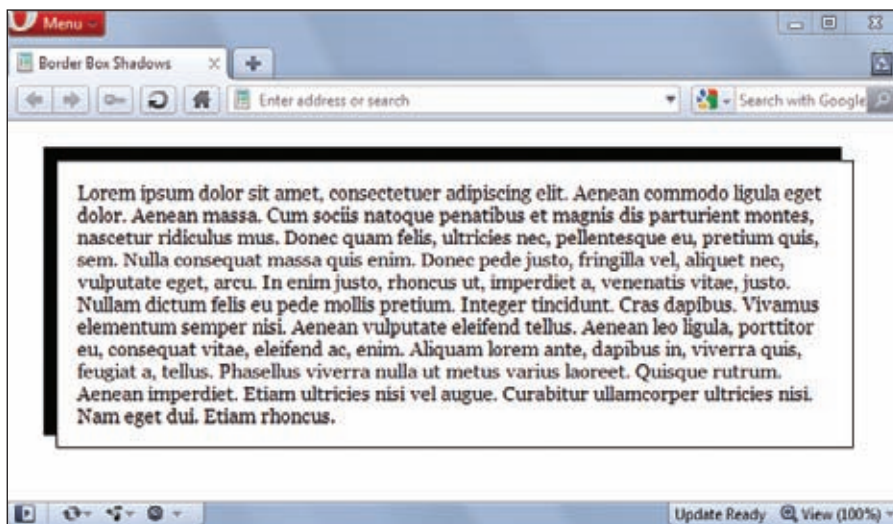


**Figure 6-24** Paragraph element with a box shadow

The box-shadow property lets you set both the horizontal and vertical measurement and color for the shadowed edges of the paragraph box. The style rule for this paragraph looks like this:

```
p {
  margin: 2em;
  border: thin solid;
  box-shadow: .5em .5em #ccc;
  padding: 1em;
}
```

The first value is the horizontal offset, and the second is the vertical offset. You can also use negative values to achieve results like the one shown in Figure 6-25.



**Figure 6-25** Paragraph element with negative box shadow

The style rule for the negative shadow looks like this:

```
p {  
  margin: 2em;  
  border: thin solid;  
  box-shadow: -10px -10px #000;  
  padding: 1em;  
}
```

## Activity: Creating a Simple Page Layout

In the following steps, you have a chance to apply some of the properties you learned about in this chapter to build a simple page layout. You will build on these skills in the next chapter to create more complex page designs. As you work through the steps, refer to Figure 6-29 to see the results you will achieve. New code that you will add is shown in **blue**. Save your file and test your work in the browser as you complete each step.

To apply the box properties:

1. Copy the **box model.html** file from the Chapter06 folder provided with your Data Files to the Chapter06 folder in your work folder. (Create the Chapter06 folder, if necessary.)
2. Open the file **box model.html** in your HTML editor, and save it in your work folder as **box model1.html**.
3. In your browser, open the file **box model1.html**. When you open the file, it looks like Figure 6-26.



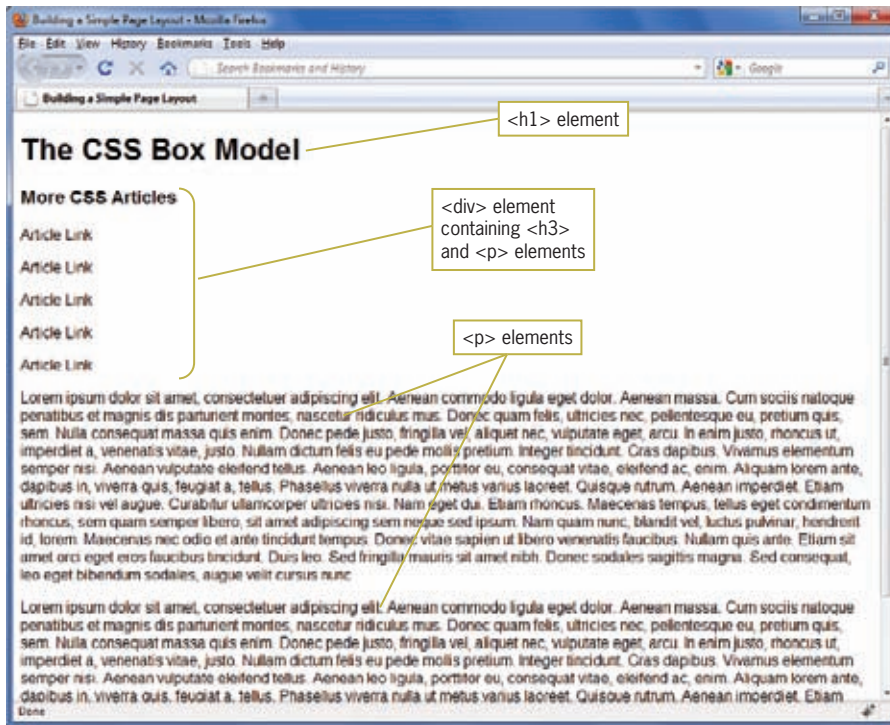


Figure 6-26 Original HTML document

- Examine the code. Notice the `<style>` section of the file. It contains one style rule to set the font-family for the page.

```
<style type="text/css">
body {font-family: arial, sans-serif;}
</style>
```

As shown in Figure 6-26, the HTML code in the file consists of:

- `<h1>` element for the heading
  - `<div>` content container that has an `<h3>` heading and `<p>` elements that simulate links
  - Two paragraph elements
- Start by styling the `<h1>` element with a 20-pixel left and right margin as shown in the following code:

```
h1 {
margin-left: 20px;
margin-right: 20px;
}
```

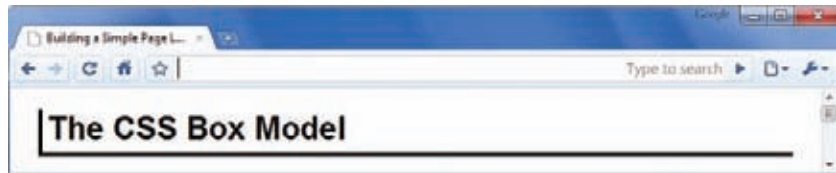
6. Add a border to the left and bottom sides of the `<h1>` for a distinctive look. Set the width of the border to 4 pixels.

```
h1 {
  margin-left: 20px;
  margin-right: 20px;
  border-bottom: solid 4px black;
  border-left: solid 4px black;
}
```

7. Add 5 pixels of padding to the left and bottom sides of the `<h1>` to offset the text from the border you just created.

```
h1 {
  margin-left: 20px;
  margin-right: 20px;
  border-bottom: solid 4px black;
  border-left: solid 4px black;
  padding-bottom: 5px;
  padding-left: 5px;
}
```

The finished `<h1>` element looks like Figure 6-27.



**Figure 6-27** Completed `<h1>` element

8. Now you will create the floating content box that contains the “More CSS Articles” heading and links. Write a style rule that uses an id of *sidebar*. Add a width of 200 pixels and a solid 1-pixel border.

```
#sidebar {
  width: 200px;
  border: 1px solid black;
}
```

9. In the `<body>` section of the document, find the `<div>` element that contains the “More CSS Articles” and add the id attribute to apply the sidebar style as shown:

```
<div id="sidebar">
  <h3>More CSS Articles</h3>

  <p>Article Link</p>
  <p>Article Link</p>
  <p>Article Link</p>
```

```
<p>Article Link</p>
<p>Article Link</p>
</div>
```

10. Float the division to the left side of the page. Add a left margin of 20 pixels to align the box with the heading and a right margin of 20 pixels to offset the paragraph text from the box.

```
#sidebar {
  width: 200px;
  border: 1px solid black;
  float: left;
  margin-left: 20px;
  margin-right: 20px;
}
```

11. Finish the content box by centering the text and adding a background color (#cff).

```
#sidebar {
  width: 200px;
  border: 1px solid black;
  float: left;
  margin-left: 20px;
  margin-right: 20px;
  text-align: center;
  background: #cff;
}
```

Figure 6-28 shows the completed floating sidebar content box. Notice that the paragraph text is very close to the left and right sides of the browser window.



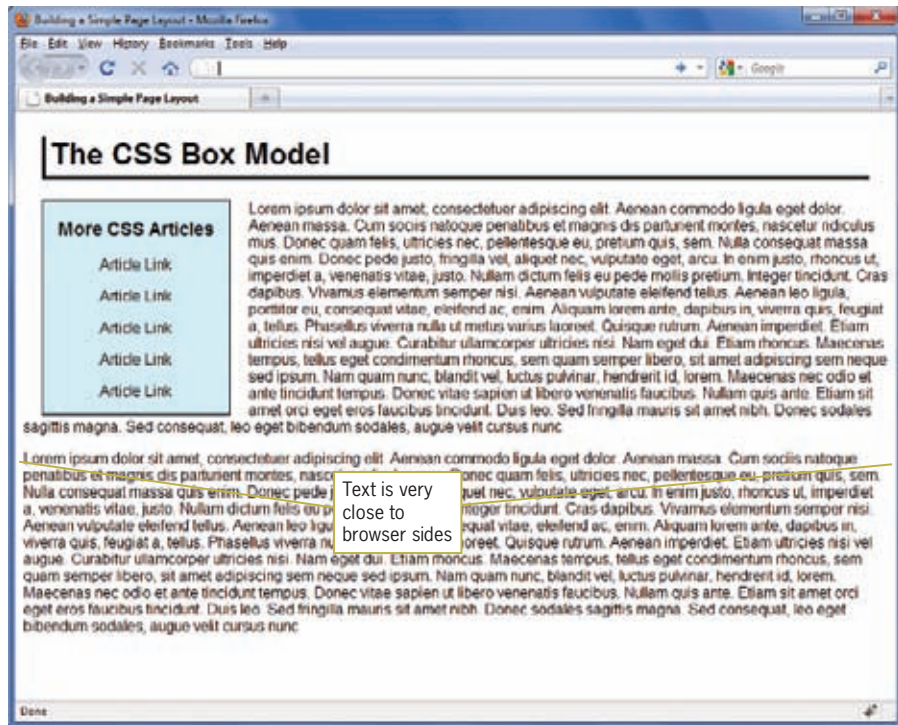
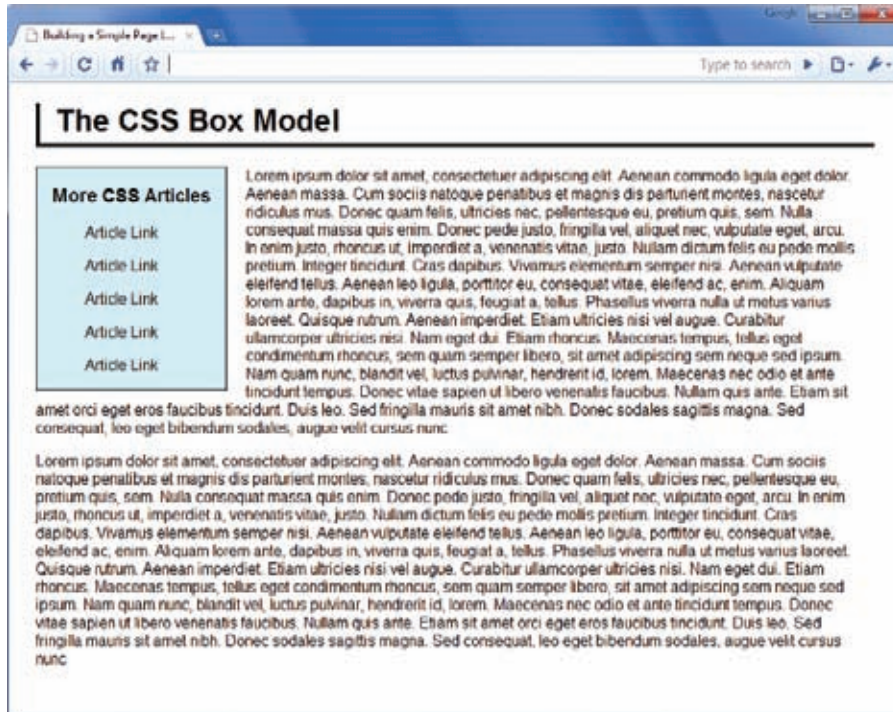


Figure 6-28 Floating content box

12. Finish the design by aligning the text properly using left and right margin properties. Write a rule that selects the <p> elements and sets the left margin to 20 pixels and the right to 40 pixels.

```
p {
    margin-left: 20px;
    margin-right: 40px;
}
```

Figure 6-29 shows the completed Web page design.



**Figure 6-29** Completed Web page design

The finished style sheet code:

```
<style type="text/css">
body {font-family: arial, sans-serif;}

h1 {
  margin-left: 20px;
  margin-right: 20px;
  border-bottom: solid 4px black;
  border-left: solid 4px black;
  padding-bottom: 5px;
  padding-left: 5px;
}

#sidebar {
  width: 200px;
  border: 1px solid black;
  float: left;
  margin-left: 20px;
  margin-right: 20px;
  text-align: center;
  background: #cfc;
}
```

```
p {  
    margin-left: 20px;  
    margin-right: 40px;  
}  
  
</style>
```

## Chapter Summary

In this chapter you learned about the concepts of the CSS box and visual formatting models. You saw how the margin, padding, and border properties let you control the space around block-level elements on a Web page. By using these properties judiciously, you can enhance the legibility of your content. You also learned how the page layout box properties create boxes and columns for content, wrap text, and float images.

- The CSS box model lets you control spacing around the element content.
- You can state values of margin, border, and padding for all four sides of the box or individual sides.
- To build scalable Web pages, choose relative length units such as ems or percentages.
- To build fixed pages, choose pixel measurements.
- The browser collapses vertical margins to ensure even spacing between elements.
- Margins are transparent, showing the color of the containing element's background color. Padding takes on the color of the element to which it belongs.
- The border properties let you add borders to all individual sides or all four sides of an element. The three border characteristics are style, color, and width. Style must be stated to make the border appear.
- The page layout box properties let you create floating content boxes and wrap text around images.
- Remember to use margin, border, and padding properties to enhance the legibility of your content.

## Key Terms

**border properties**—CSS properties that let you control the appearance of borders around elements.

**box model**—A CSS element that describes the rectangular boxes containing content on a Web page.

**containing box**—The containing rectangle, or parent element, of any child element. The absolute containing element is the window area of the browser. All other elements are displayed within this containing box, which equates to the <body> element of an HTML document. Within <body>, elements such as <div> or <p> are parents, or containing boxes, for their child elements.

**CSS visual formatting model**—A model that describes how CSS element content boxes should be displayed by the browser.

**margin properties**—CSS properties that let you control the margin area of the box model.

**padding properties**—CSS properties that let you control the padding area in the box model.

**page layout box properties**—CSS properties that let you control the dimensions and position of content boxes.

## Review Questions and Exercises

1. What are the three space areas in the box model?
2. Which space area is transparent?
3. What does the visual formatting model describe?
4. What is the visual formatting model based on?
5. What are percentage measurement values based on?
6. What are the preferred length units for margins and padding?
7. In the following rule, what is the size of the vertical margins between paragraphs?

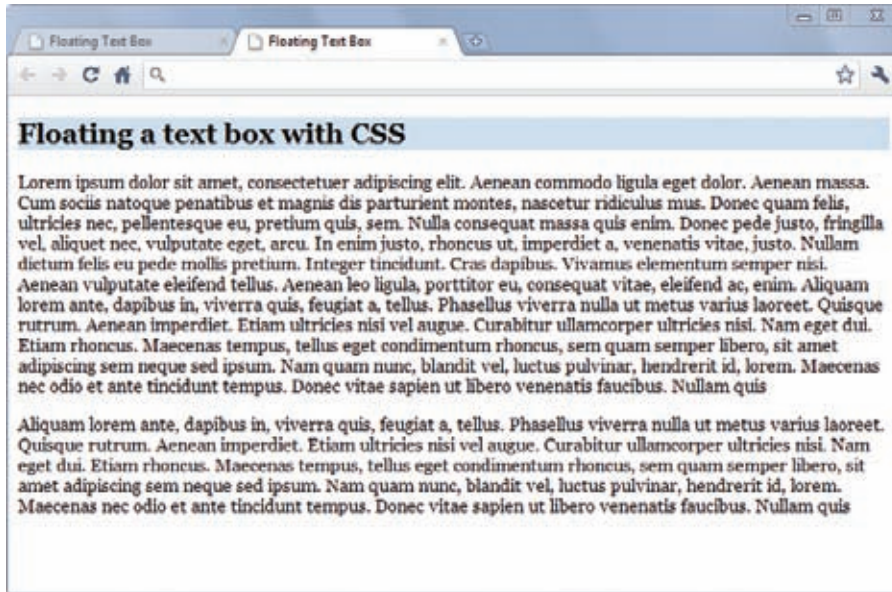
```
p {margin-top: 15px; margin-bottom: 10px;}
```

8. Where is the padding area located?
9. What are the five most common border properties?
10. What is the default border style?
11. What is the default border weight?
12. What is the default border color?
13. What does the float property let you do?
14. What does the clear element let you do?
15. Write a style rule for a `<p>` element that sets margins to 2em, padding to 1em, and a black, solid 1-pixel border.
16. Write a style rule for an `<h1>` element that sets top and bottom padding to .5em with a dashed thin red border on the bottom.
17. Write a style rule for a `<p>` element that creates left and right padding of 1em, a left margin of 30 pixels, and a left black medium double border.

## Hands-On Projects

1. Browse the Web and choose a site that you feel exhibits good handling of white space to increase the legibility of the content. Write a short design critique of why the white space works effectively.
2. Browse the Web and choose a site that can benefit from the box properties available in CSS. Print a copy of the page and indicate where you would change the spacing and border properties. Write a short essay that describes the changes you want to achieve and how they would increase the legibility of the page content.
3. In this project, you will create a floating text box.
  - a. Copy the **floatactivity.html** file from the Chapter06 folder provided with your Data Files to the Chapter06 folder in your work folder. (Create the Chapter06 folder, if necessary.)

- b. Open the file **floatactivity.html** in your HTML editor, and save it in your work folder as **floatactivity1.html**.
- c. In your browser, open the file **floatactivity1.html**.  
When you open the file, it looks like Figure 6-30.



**Figure 6-30** Original HTML file for Project 3

- d. Examine the page code. Notice that an existing style rule sets a background-color for a floatbox class, as shown in the following code fragment:  

```
.floatbox {background-color: #ccddee;}
```
- e. This class is applied to the first <p> element in the document, as shown in Figure 6-30. Your goal is to use a variety of box properties to create a finished page that looks like Figure 6-31.



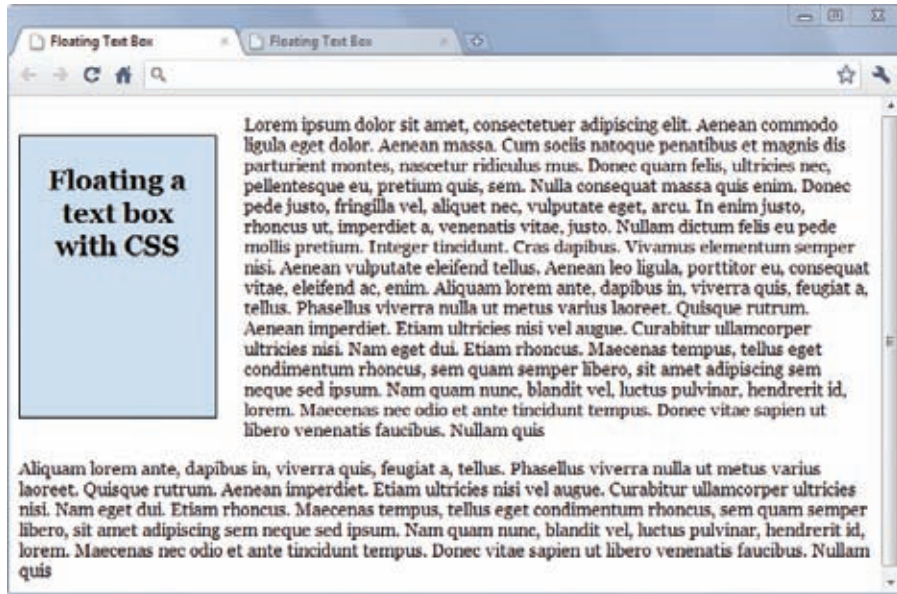


Figure 6-31 Completed HTML file for Project 3

f. Use the following properties to create the finished floating text box:

- width
- height
- float
- padding
- margin-right
- border
- text-align

Experiment with the different properties until you achieve results that look as close to the finished page as possible.

4. In this project, you will have a chance to test the border properties. Save and view the file in your browser after completing each step.

a. Using your HTML editor, create a simple HTML file (or open an existing file) that contains heading and paragraph elements. Save the file in your Chapter06 folder as **borders.html**.



- b. Add a `<style>` element to the `<head>` section as shown in the following code:

```
<head>
<title>CSS Test Document</title>
<style type="text/css">
</style>
</head>
```

- c. Experiment with the different border styles. Start by applying any of the following style rules to your document's elements:

```
h1 {border: solid 1px black;}
h2 {border-top: solid 1px; border-bottom: solid 3px;}
p {border-left: double red; border-right: solid 1px;}
```

- d. Experiment with adding padding properties to your style rules to offset the borders from the text. The following style rules have sample padding properties to try:

```
h1 {border: solid 1px black; padding: 20px;}
h2 {border-top: solid 1px; border-bottom: solid 3px;
padding-top: 15px; padding-bottom: 30px;}
p {border-left: double red; border-right: solid 1px;
padding-left: 30px; padding-right: 20px;}
```

- e. Continue to experiment with the border and padding properties. Try adding color and margin properties to see how the elements are displayed.

## Individual Case Project

Create the box model conventions for your Web site. Build on the typographic classes you created in Chapter 5. Think about the different spacing requirements for your content and decide how the legibility can be enhanced using the box properties. Add this information to the type specification HTML page that shows examples of the different typefaces and sizes and how they will be used. Decide on margins, padding, and borders and select the elements that will benefit from their use. Create before and after sample HTML pages that reflect the enhanced design.

## Team Case Project

Work with your team to decide on the box model conventions for your project Web site. These include any spacing specifications for your content that will increase legibility and clarity.

You may need to create sample mock-up HTML pages to present your ideas on these characteristics to your team members. Decide on margins, borders, and padding and how you will use these in your Web site. Think about creating floating boxes within sections for text highlights, how to standardize the look of different sections of your site with increased white space, and how borders can help to emphasize headings and columns.

After you have reached a general consensus, go back to work on your page template. Create more finished mock-ups of your page design. Trade the page layout examples with your team members to reach consensus.