

# USING THE BOX PROPERTIES

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

- ◆ Understand the CSS visual formatting model
- ◆ Use the CSS box model
- ◆ Use the margin properties
- ◆ Use the padding properties
- ◆ Use the border properties
- ◆ Use the special box properties
- ◆ Apply the box properties

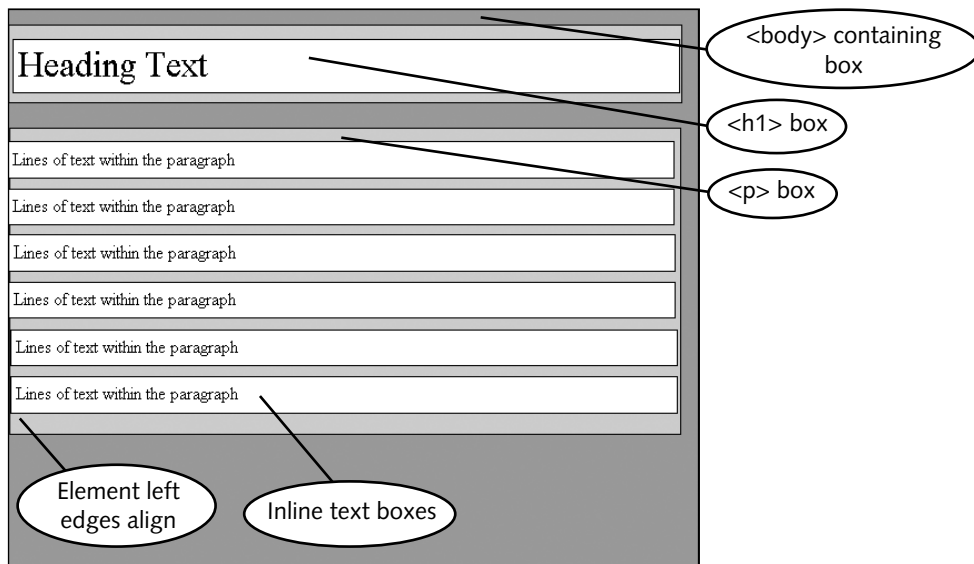
In this chapter you will explore the CSS box properties. These properties let you control the margin, padding, and border characteristics of block-level elements. To understand how these properties work, you will first learn about the CSS visual formatting model and box model. These models control the 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 floating text boxes.

## THE CSS VISUAL FORMATTING MODEL

The CSS visual formatting model describes how the element content boxes should be displayed by the browser. The visual formatting model is based on the hierarchical structure of the HTML document and the element display type. In XHTML, elements fall into three display type categories:

- *Block*—Block-level elements appear as blocks such as paragraphs. Block elements contain inline boxes that contain the element content.
- *Inline*—Inline-level elements contain the content within the block-level elements. They do not form new blocks of content.
- *List-item*—List-item elements create a surrounding containing box and list-item inline boxes.

Figure 9-1 shows three different block-level elements, `<body>`, `<h1>`, and `<p>`. The `<h1>` and `<p>` elements contain inline content boxes.



**Figure 9-1** The CSS visual formatting model

Figure 9-1 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 element boxes reside within `<body>`. In Figure 9-1, 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.

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 9-1, 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.

## Specifying Element Display Type

### display property description

Value: block | inline | list-item | none

Initial: inline

Applies to: all elements

Inherited: no

Percentages: N/A

The CSS display property determines the display type of an element. When you are working with XHTML, you will probably not use the display property very often because the browser contains default element display information for each element. For example: `<h1>`, `<p>`, and `<blockquote>` elements are block-level elements; `<b>` and `<i>` are inline elements. In most cases you want the browser to display each element using the default element type. However, there may be times when you want to manipulate this property. The following style rule changes the default display type for an `<h1>` element from block to inline.

```
h1 {display: inline;}
```

The result of this style rule is an `<h1>` element that behaves like an inline element, as shown in Figure 9-2.



**TIP**

The display property “none” value lets you hide an element so that it is not displayed in the browser. This could be useful in a style sheet where you may want to display only some of the information on a page.

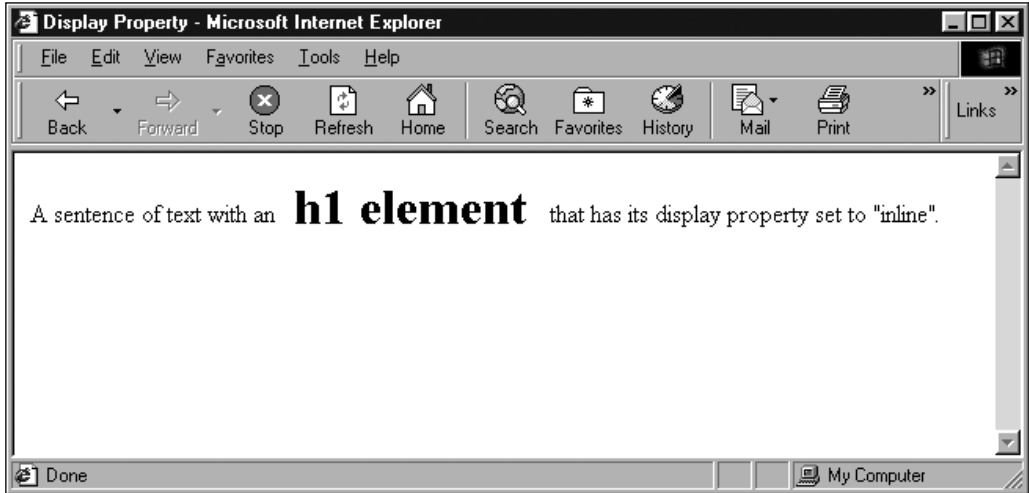


Figure 9-2 Manipulating the display property

## 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 containing content. Each content box can have margins, borders, and padding, as shown in Figure 9-3.

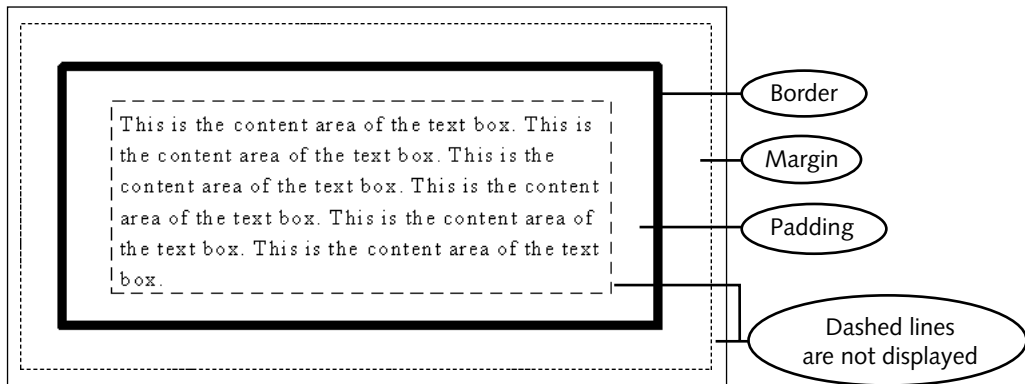
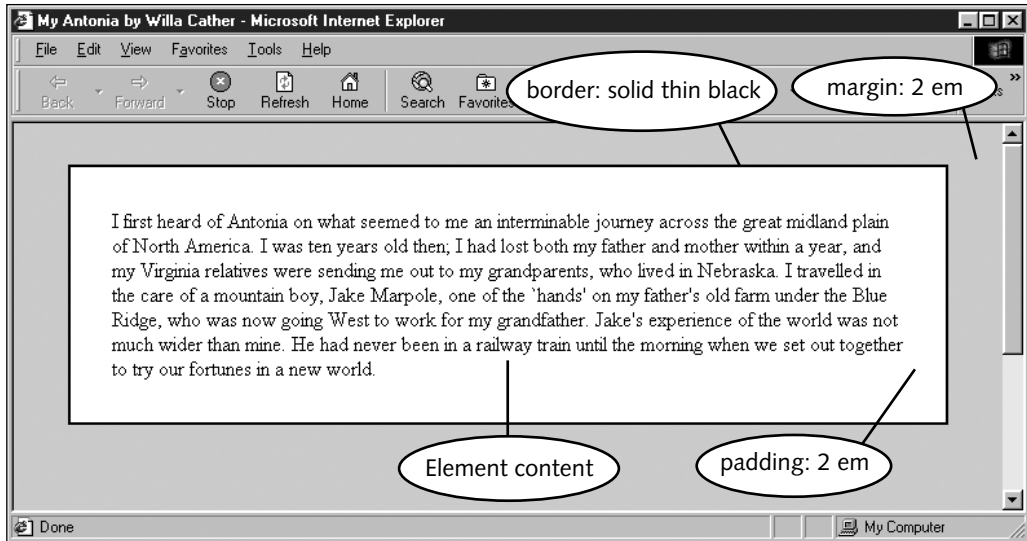


Figure 9-3 The CSS box model

As Figure 9-3 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 9-4 shows the box model areas in a paragraph element. This paragraph has 2-em padding, a thin black border, and 2-em margins. Notice that the margin area is transparent.



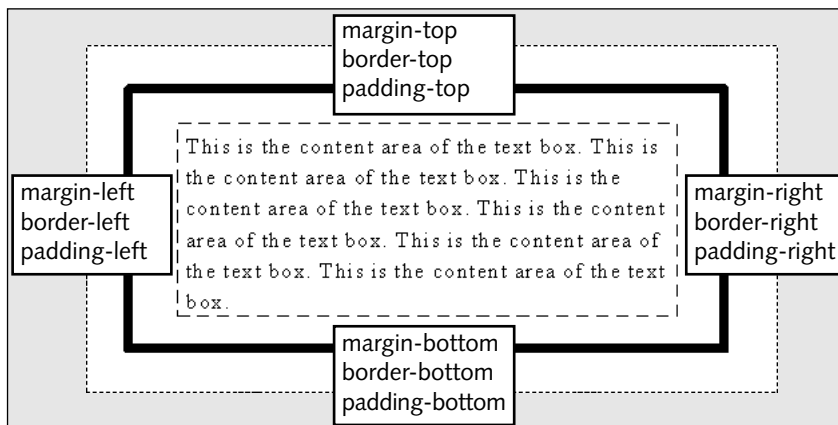
**Figure 9-4** The CSS box model areas in a `<p>` element

The following code shows the style rule for the paragraph in Figure 9-4:

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

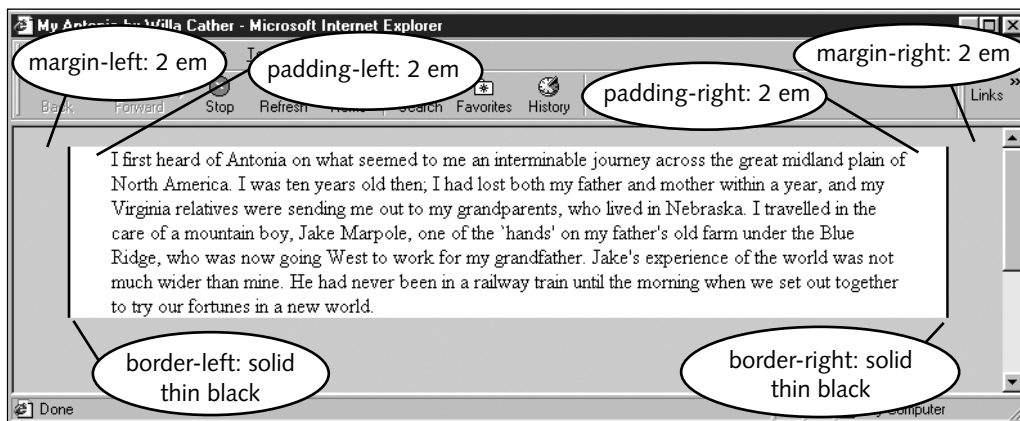
The margin and padding properties set the length to 2 em for all four sides of the box. The border property is a shorthand property that sets the border-style property to solid, the border-weight property to thin, and the border-color property to black. The background-color property sets the paragraph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

CSS lets you specify margin, padding, and border properties individually for each side of the box. Figure 9-5 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. Netscape 4.x does not support the individual properties, but Opera 6.0, Netscape 7.0, and Internet Explorer 6.0 do.



**Figure 9-5** The CSS box model individual sides

Figure 9-6 shows the same paragraph with `margin-left` and `margin-right` set to 2 em; `padding-left` and `padding-right` set to 2 em, and `border-left` and `border-right` set to solid thin black.



**Figure 9-6** The CSS box model individual sides in a `<p>` element

The style rule for the paragraph is:

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

## 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 7.) 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 pixels when you are stating margin, border, or padding sizes. The relative measurement values let you build scalable Web pages. In some instances, it's preferable to use the absolute values, such as the point, but these are generally more useful when you know the exact measurements of the output device.

---

## USING 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 property description**

**Value:** <length> | <percentage>

**Initial:** 0

**Applies to:** all elements

**Inherited:** no

**Percentages:** refer to width of containing block

The margin property is a shorthand property that lets you set all four individual margins with one property. You can specify either a length or percentage value. The most commonly supported usage of the margin property is to state one value for all four margin sides, as shown in the following style rule:

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

You also can choose to state individual margin settings within this same rule. This can be confusing, because the individual margin settings change based on the order within the rule. Table 9-1 shows how the syntax works.

Table 9-1 Shorthand notation for the margin property

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

To make your style rules more specific and easy to read, you can use the individual margin properties, such as `margin-left`, rather than the shorthand notation. The individual margin properties are described in the next section.

Figure 9-7 shows two paragraph elements. The first paragraph has the margin set to 2 em, the second has the default margin setting.

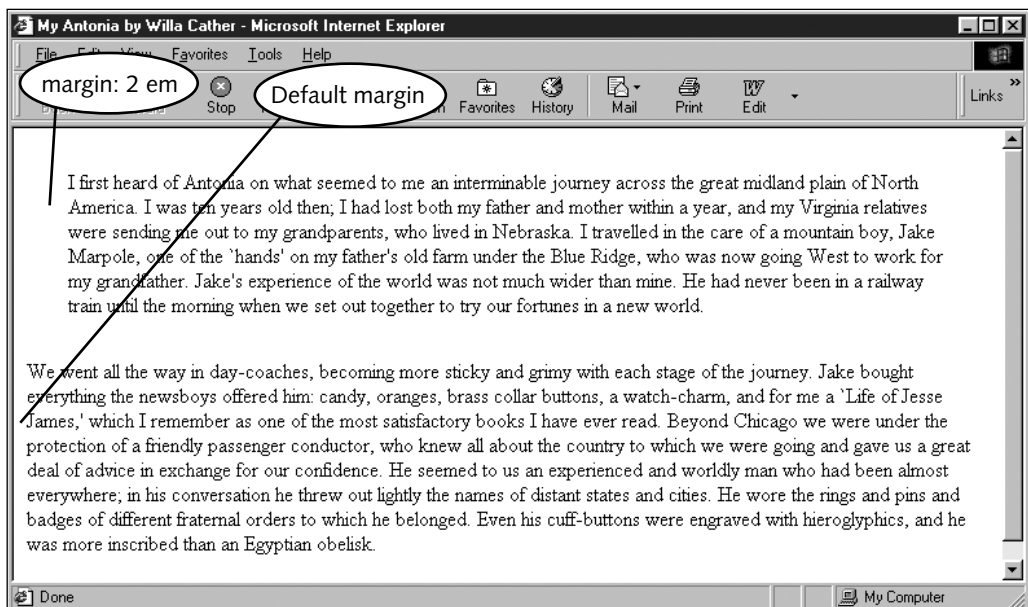


Figure 9-7 Using the margin property

Notice that the increased margins enhance the legibility of the text. However, only the horizontal margins on the left and right sides of the paragraph are beneficial. The extra vertical margin between the paragraphs is too large and breaks up the flow of the text. To solve this problem, you can use the individual margin properties, described next.

## Specifying the Individual Margin Properties

### **margin-left, margin-right, margin-top, margin-bottom individual property descriptions**

Value: <length> | <percentage>

Initial: 0

Applies to: all elements

Inherited: no

Percentages: refer to width of containing block

The individual margin properties let you control each of the individual margins: margin-left, margin-right, margin-top, and margin-bottom. The following style rule sets the left and right margins for the paragraph element:

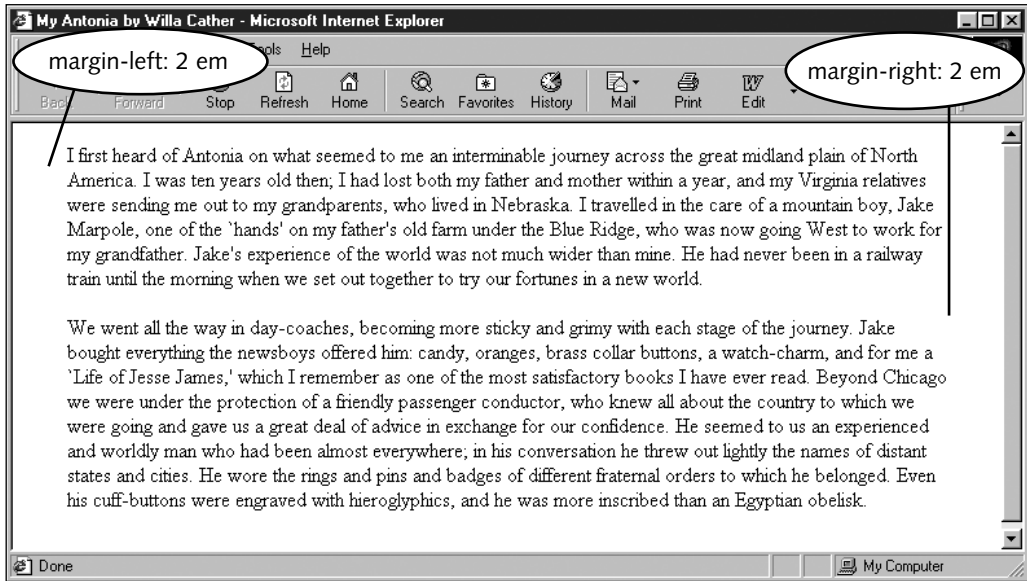
```
p { margin-left: 2em;
    margin-right: 2em;
}
```

As Figure 9-8 shows, both paragraphs now have the default vertical margins with 2-em left and right horizontal margins.

## Negative Margins

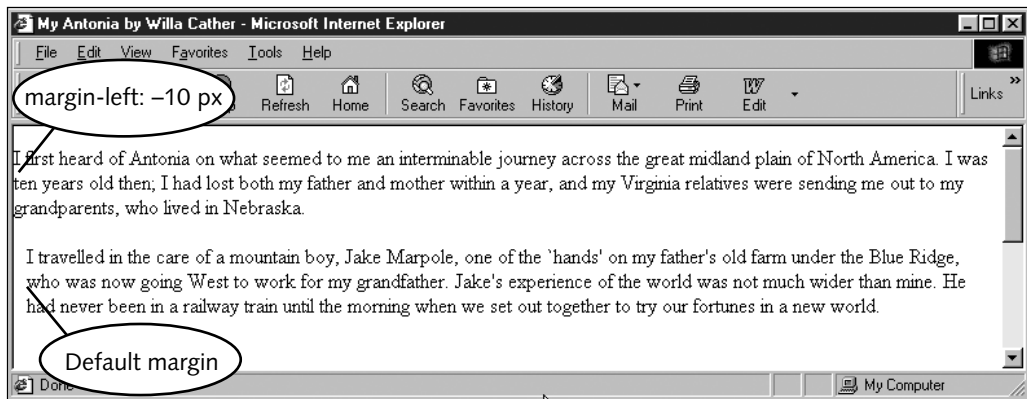
You can set negative margin values to achieve special effects. For example, you can override the default browser margin by setting a negative value. Although it varies by browser, the default left margin is approximately 10 pixels. The following rule sets a negative value of -10 pixels:

```
p {margin-left: -10px;}
```



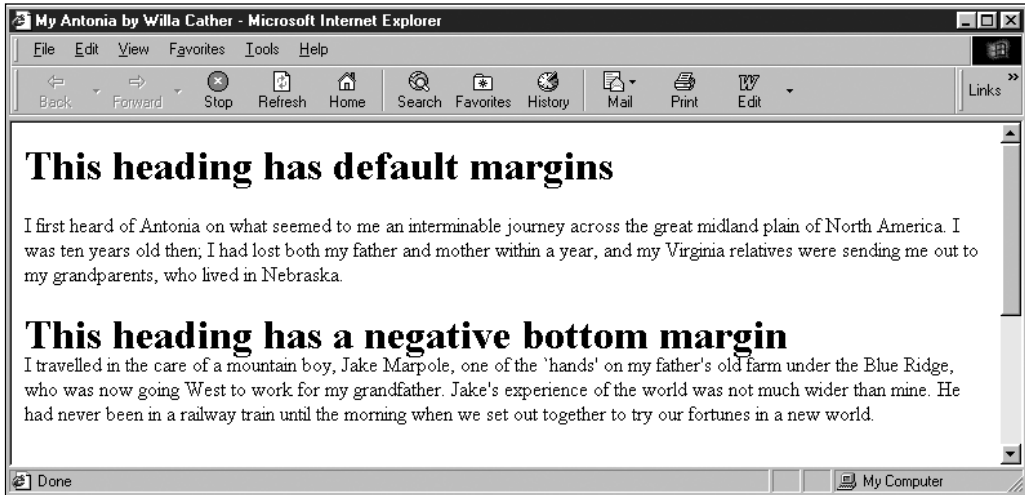
**Figure 9-8** Using the individual margin properties

Figure 9-9 shows two paragraphs, one with the default margin and one with a negative margin.



**Figure 9-9** A `<p>` element with negative left margin

You can also use negative margins to remove the default margins from other elements. Figure 9-10 shows two `<h1>` elements, one with the bottom margin set to a negative value and one with the default margins.



**Figure 9-10** An `<h1>` element with negative bottom margin

Although neither of the preceding negative margin results really enhances the legibility of the text, you might someday encounter a design problem that requires a negative margin solution; it's helpful to know that CSS allows you to do this.

## 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 9-11 shows the results of collapsing the margins with the preceding style rule.

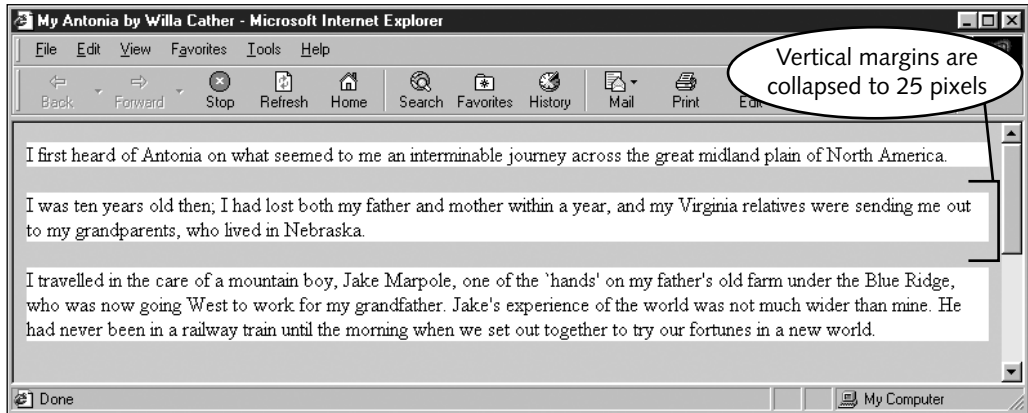


Figure 9-11 The browser collapses vertical margins

## USING 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 add a border to an element, you will almost always want to use padding to increase the white space between the content and the border, as shown in Figure 9-12.

### Specifying Padding

#### **padding property description**

Value: `<length>` | `<percentage>`

Initial: 0

Applies to: all elements

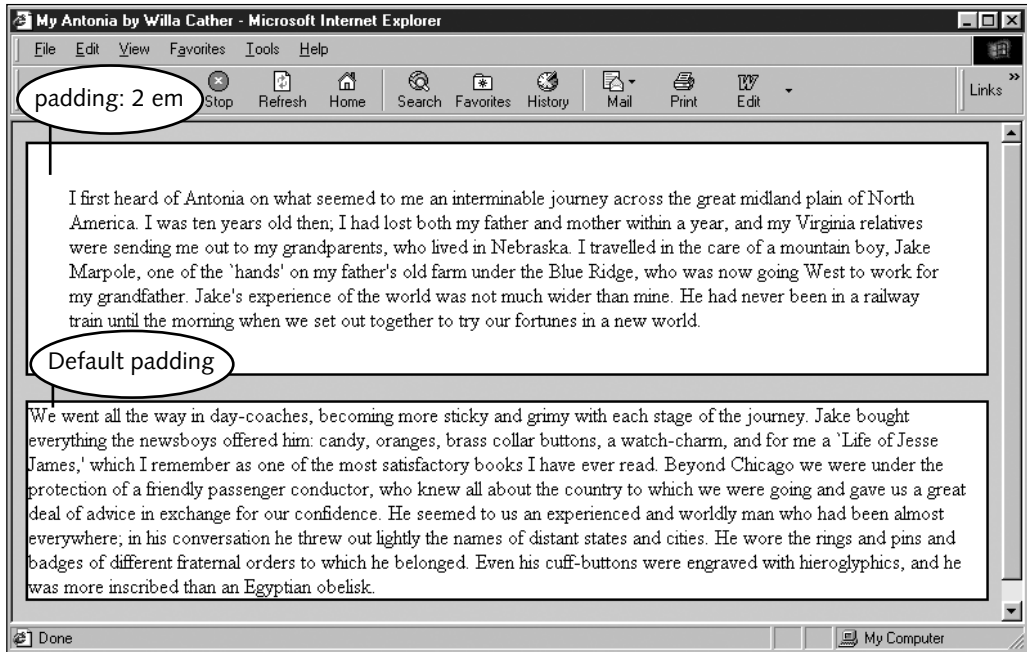
Inherited: no

Percentages: refers to width of containing block

The padding property is a shorthand property that lets you set all four individual padding values with one rule. You can specify either a length or a percentage value. Unlike margins, you cannot collapse the padding area or set negative padding values.

The most common usage of the padding property is to state one value for all four padding sides, as shown in the following style rule:

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



**Figure 9-12** Default padding and 2-em padding

You can also choose to state individual padding settings in the padding property. Like the margin shorthand property described earlier, the individual padding settings change based on the order within the rule. Table 9-2 shows how the syntax works.

**Table 9-2** Shorthand notation for the padding property

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

## Specifying the Individual Padding Properties

### padding-left, padding-right, padding-top, padding-bottom individual property descriptions

Value: <length> | <percentage>

Initial: 0

Applies to: all elements

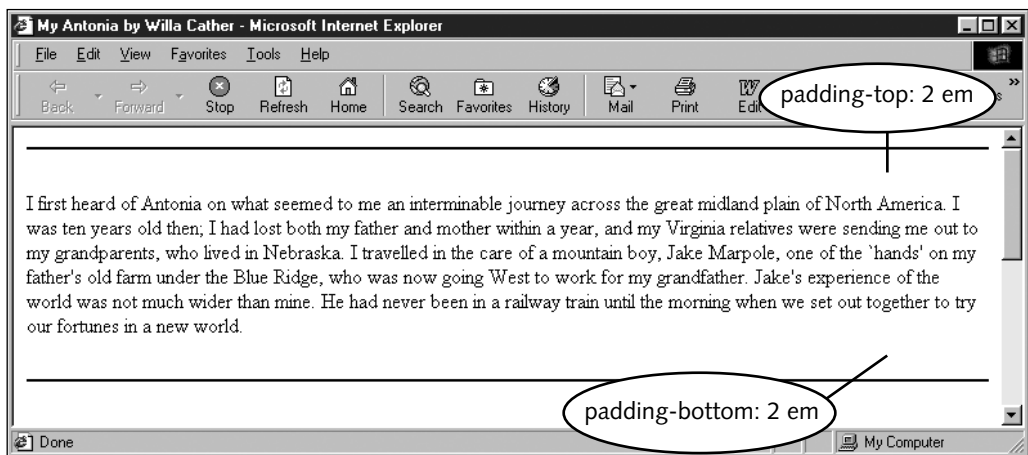
Inherited: no

Percentages: refer to width of containing block

The individual padding properties let you control the individual padding areas: padding-left, padding-right, padding-top, and padding-bottom. The following style sets the top and bottom padding areas for the paragraph, along with complementing borders and a white background:

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

As Figure 9-13 shows, the paragraph now has the default left and right padding with 2-em top and bottom padding.



**Figure 9-13** Using the individual padding properties

## USING THE BORDER PROPERTIES

The border properties let you control the appearance of borders around elements. The border area resides between the margin and padding. You can set 20 border properties, many of which are too specific for common use. 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 9-3 lists the entire range of 20 border properties.

**Table 9-3** 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-right-style border-top-style border-bottom-style	border-left-width border-right-width border-top-width border-bottom-width	border-left-color border-right-color border-top-color border-bottom-color

To use the shorthand properties you must first understand the three border characteristics—border style, border color, and border width. Then you will learn how to use the border shorthand properties.

## Specifying Border Style

### border-style property description

Value: <border-style>

Initial: none

Applies to: all elements

Inherited: no

Percentages: N/A

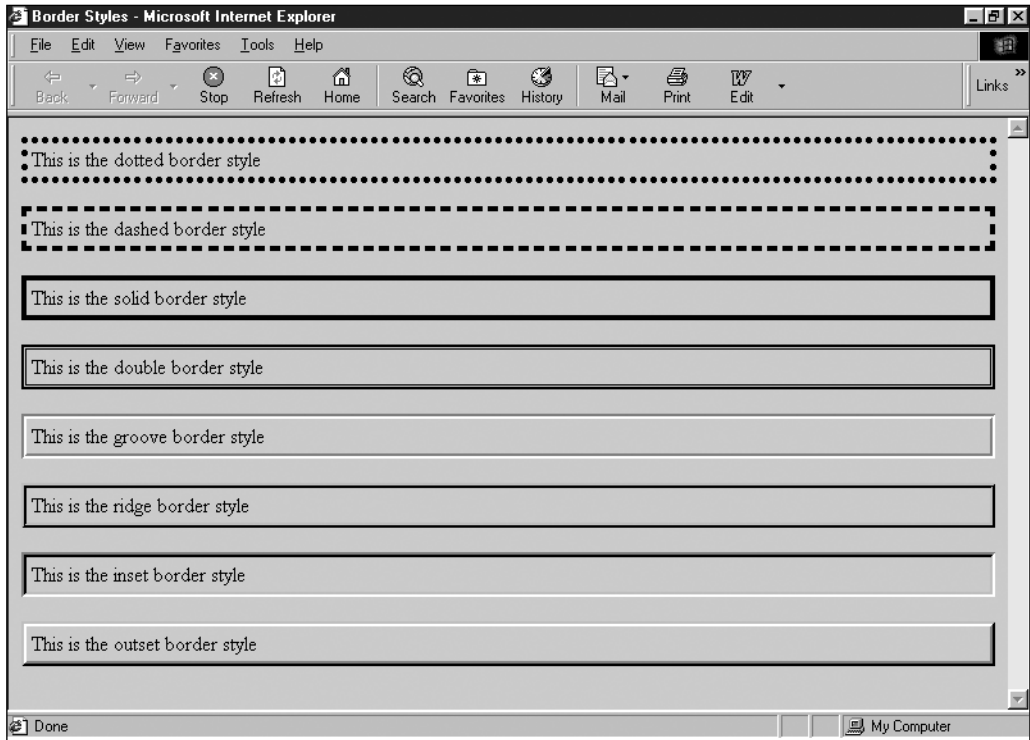
The border style is the most important border property because it must be stated to make a border appear. The border-style property lets you choose from one of the following border-style keywords:

- *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
- *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 9-14 shows examples of the borders. The gray background for this page enhances the display of the three-dimensional styles, which do not look the same on a white background. Not all borders are supported by all browsers, so test your work carefully. If you specify a border style that is not supported, the border defaults to solid.



**Figure 9-14** Different border styles

You can also choose to state border styles for individual sides using the border-style property. The individual border-style settings change based on the order within the rule. Table 9-4 shows how the syntax works.

**Table 9-4** Shorthand notation for the border-style property

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

Of course, if you examine the rules in Table 9-4, 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.

## Individual Border Styles

You can also specify individual border 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;}
```

## Specifying Border Width

### **border-width property description**

Value: thin | medium | thick | <length>

Initial: medium

Applies to: all elements

Inherited: no

Percentages: N/A

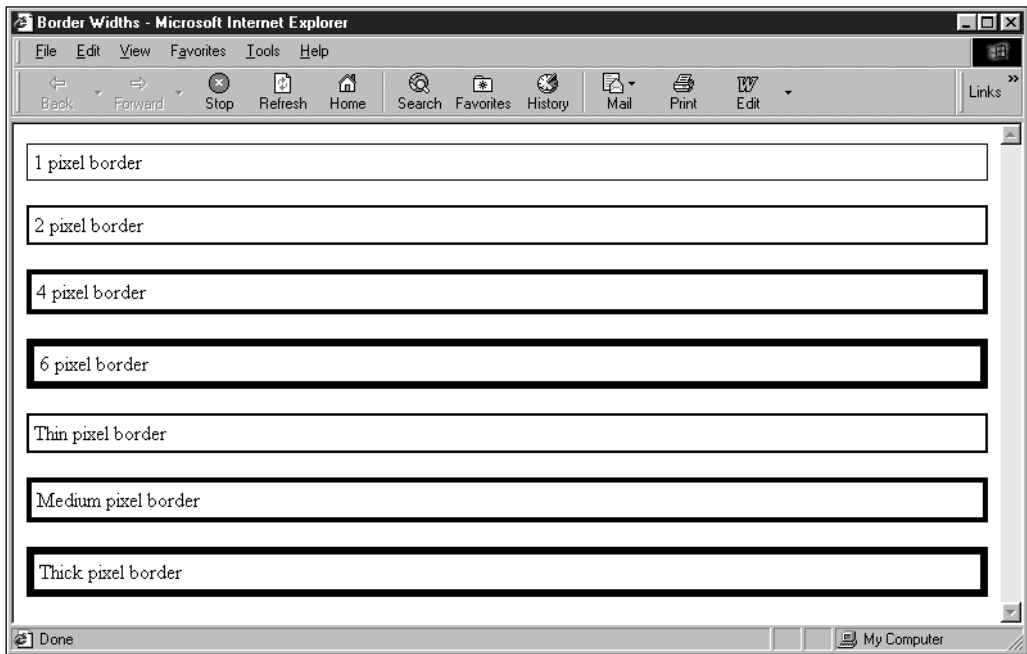
The border-width property lets you state the width of the border with either a keyword or a length value. You can use the following keywords to express width:

- 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 9-15 shows examples of different border widths.



**Figure 9-15** Different border widths

You can also choose to state individual border widths in the border-width property. The individual border-width settings change based on the order within the rule. Table 9-5 shows how the syntax works.

Table 9-5 Shorthand notation for the border-width property

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

### 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;}
```

## Specifying Border Color

### border-color property description

Value: <color>

Applies to: all elements

Inherited: no

Percentages: N/A

The border-color element 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 listed in Table 9-6.

**Table 9-6** Predefined color names

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

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 color to red. The border is also red because a border color is not specified.

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

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 9-7 shows how the syntax works.

Table 9-7 Shorthand notation for the border-color property

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

## 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 rule applies only to the left border of the element:

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

## 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.

## Specifying Borders

### **border**

Value: <border-width> | <border-style> | <border-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;}
```

## Specifying Individual Borders

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

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

Initial: See individual properties

Applies to: all elements

Inherited: no

Percentages: N/A

You can set individual border properties using the individual border shorthand properties. 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.

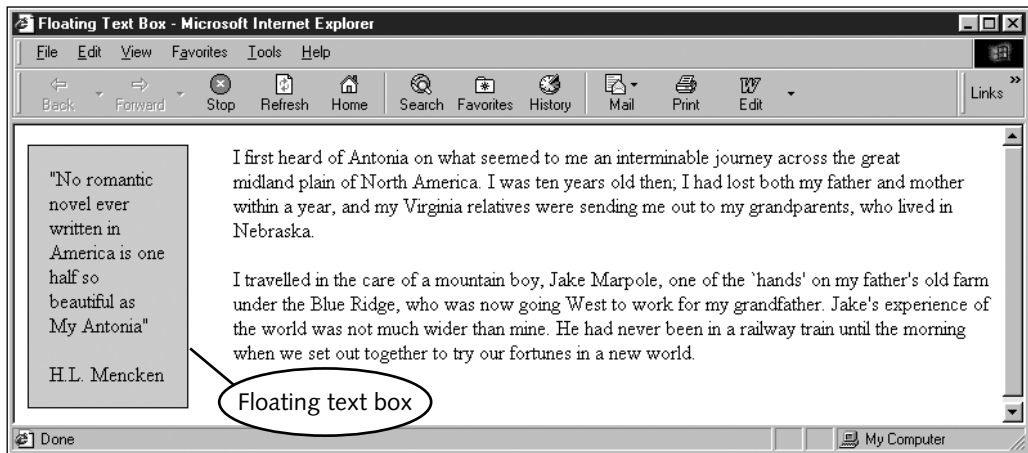
```
p {border-left: solid thin; border-right: solid thin;}
```

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;}
```

## USING THE SPECIAL BOX PROPERTIES

The special box properties are used primarily with objects such as images. In most instances you will not set these properties for normal block-level elements. However, you can use them to create floating text boxes as shown in Figure 9-16.



**Figure 9-16** A floating text box

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

- width
- height
- float
- clear

After you have learned about the four properties, you will see how they are used to create a floating text box like the one shown in Figure 9-16.

## Width

### **width property description**

**Value:** <length> | <percentage>

**Initial:** auto

**Applies to:** all elements but nonreplaced 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. Width is not intended for normal block-level elements, but you can use it to create floating text boxes or with images. In most cases you set the padding or margins of the elements rather than explicitly stating a width.

The width property accepts either a length value or a percentage. The percentage value is based on the width of the containing element box. By default, the value of width is set to auto, which is based on the content of the element minus the padding, border, and margins, if applicable. The following is an example of width property usage:

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

Use percentages or relative measurement values, such as pixels (px), to ensure that your widths are portable across different screen resolutions.

## Height

### **height property description**

**Value:** <length> | <percentage>

**Initial:** auto

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

**Inherited:** no

**Percentages:** N/A

The height property lets you set the vertical height of an element. Like width, height is not intended for normal block-level elements, but you can use it to create floating text

boxes or images. In most cases you set the padding or margins of the elements rather than explicitly stating a height.

The height property accepts either a length value or a percentage. The percentage value is based on the height of the containing element box. By default the value of height is set to auto, which is based on the content of the element minus the padding, border, and margins if applicable. The following is an example of height property usage:

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

## Float

### 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. Float is most commonly used for <img> elements, allowing alignment of an image to the left or right of text. You can also use the float property to align a text box to the left or right edge of its parent.

### Floating Text Boxes

The float property can also be used to float a text box to the left or right of text. Used with the width and height properties, you can create a text box of the type shown earlier in Figure 9-16. The advantage to this type of layout is that no HTML tables are used to create the design; rather, a simple CSS rule is all that is necessary.

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

```
.floatbox {
    width: 125px;
    height: 200px;
    float: left;
    background-color: #cccccc; /* light gray */
}
```

This rule states a class named floatbox. The rule sets the width, height, and float properties of the element, and sets the background color to a hexadecimal color value that is a light gray.

The class floatbox can then be applied to an element in the document. The code from the document follows. Notice that a `<p>` is contained within the first `<div>` element. The `<p>` is the element to which the class is applied.

```

<body>

<div>

<p class="floatbox">

    "No romantic novel ever written in America is one half so
    beautiful as My Antonia"<br></br>H.L. Mencken

</p>

    I first heard of Antonia on what seemed to me an interminable
    journey across the great midland plain of North America. I
    was ten years old then; I had lost both my father and mother
    within a year, and my Virginia relatives were sending me out
    to my grandparents, who lived in Nebraska.

</div>

<p>

    I travelled in the care of a mountain boy, Jake Marpole,

    one of the 'hands' on my father's old farm under the Blue
    Ridge, who was now going West to work for my grandfather.

    Jake's experience of the world was not much wider than mine.
    He had never been in a railway train until the morning when
    we set out together to try our fortunes in a new world.

</p>

</body>

```

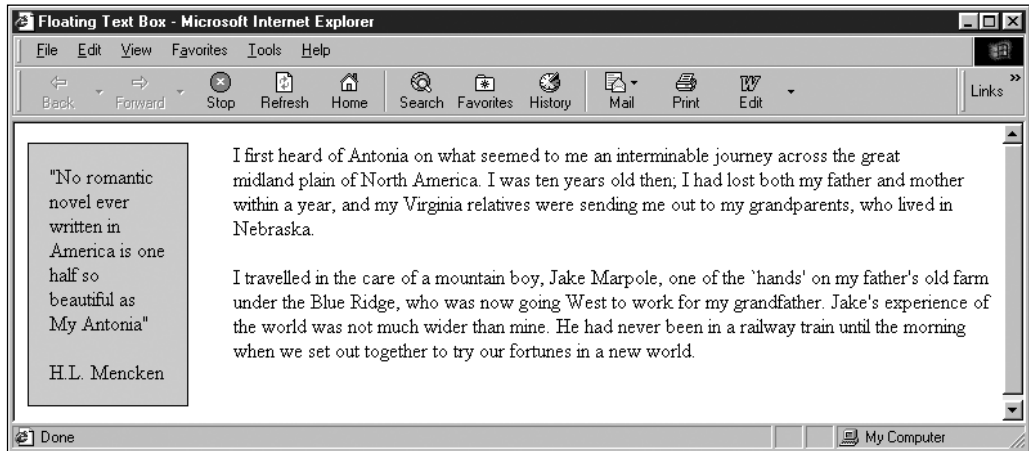
The floatbox can be enhanced by adding some of the other properties you learned about in this chapter. The following rule adds 1-em padding for the entire element, a 2-em right margin, and a 1-pixel solid, black rule.

```

.floatbox {
    width: 125px;
    height: 200px;
    float: left;
    background-color: #cccccc; /* light gray */
    padding: 1em;
    margin-right: 2em;
    border: solid black 1px;
}

```

The enhanced float box is much more legible and improves the page layout. Figure 9-17 shows the result of the new properties added to the rule.



**Figure 9-17** An enhanced floating text box

Float is supported by Internet Explorer 6.0, Opera 5.0, and Netscape 7.1. The float property is not supported in Netscape 4.x.

## Clear

### **clear property description**

**Value:** none | left | right | both  
**Applies to:** block-level elements  
**Inherited:** no  
**Percentages:** N/A

The clear property lets you control the flow of text around floated elements. You use the clear property only when you are using the float property. Clear lets you force text to appear beneath a floated element, rather than next to it. Figure 9-18 shows an example of normal text flow around an element.

This figure shows an image with the float property set to “left”. The text flows down around the image on the right, which is the correct behavior. The second-level heading and paragraph, however, are not appearing in the correct position. They should be positioned beneath the floating image. To correct this problem, use the clear property. In this instance, the `<h2>` should display 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;">
```

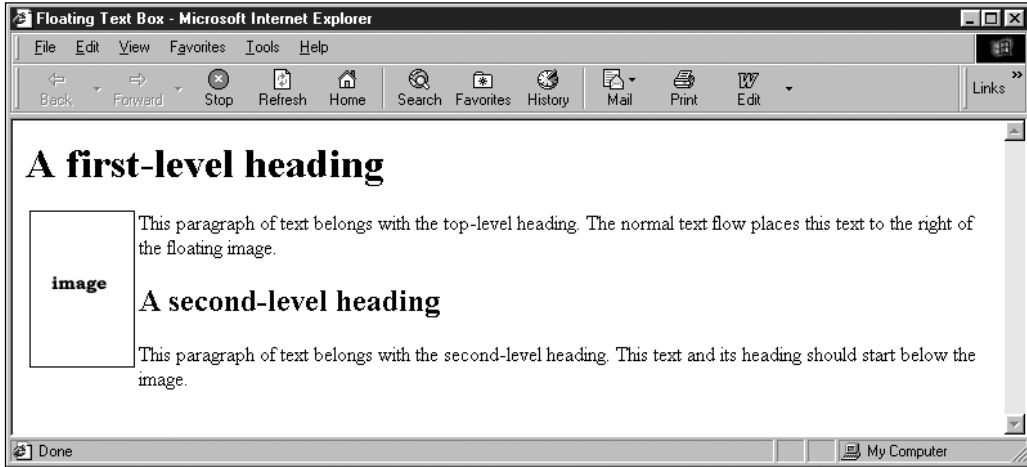


Figure 9-18 Normal text flow around a floating image

Figure 9-19 shows the result of adding the clear property.

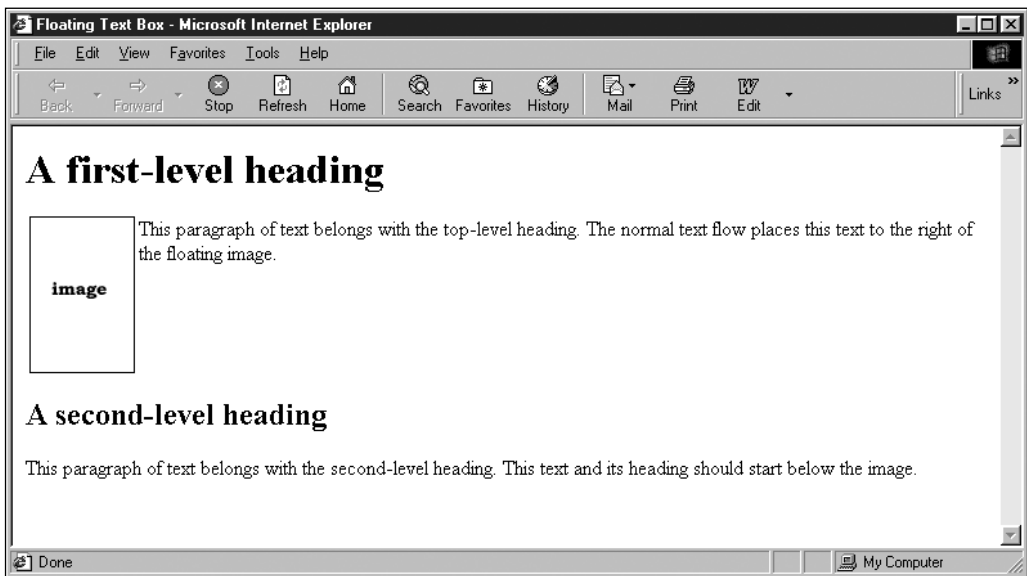


Figure 9-19 Using the clear property

The code for this page follows:

```

<html>

<head>

<title>Floating Text Box</title>

<style type="text/css">

img.left {float: left;}

</style>

</head>

<body>

<h1>A first-level heading</h1>

<p>

This paragraph of text
belongs with the top-level heading. The normal text flow
places this text to the right of the floating image.

</p>

<h2 style="clear: left;">A second-level heading</h2>

<p>

This paragraph of text belongs with the second-level heading.
This text and its heading should start below the image.

</p>

</body>

</html>

```

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 in the event you have floating images on both the left and right sides of the text.

## APPLYING THE BOX PROPERTIES

In the following steps you have a chance to apply some of the properties you learned about in this chapter. As you work through the steps, refer to Figure 9-20 to see the results you will achieve. Save your file and test your work in the browser as you complete each step.

To apply the box properties:

1. Copy the **boxtext.htm** file from the Chapter09 folder provided with your Data Files to a Chapter09 folder in your work folder. (Create the Chapter09 folder, if necessary.)
2. Open the file **boxtest.htm** in your HTML editor and save it in your work folder as **boxtest1.htm**.
3. In your browser, open the file **boxtest1.htm**. When you open the file it looks like Figure 9-20.

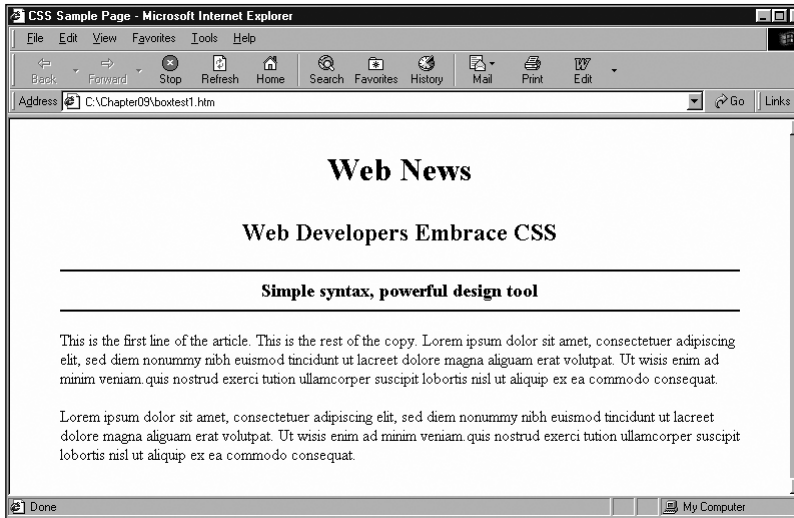


Figure 9-20 The original HTML document

4. Examine the code. Notice the `<style>` section of the file. It contains three basic style rules that center the `<h1>`, `<h2>`, and `<h3>` elements. The complete code for the page follows:

```
<html>
<head>
<title>CSS Sample Page</title>
<style type="text/css">
h1 {text-align: center;}
h2 {text-align: center;}
```

```

h3 {text-align: center;}
</style>
</head>
<body>

<h1>Web News</h1>
<h2>Web Developers Embrace CSS</h2>
<h3>Simple syntax, powerful design tool</h3>

<p>This is the first line of the article. This is the rest
of the copy. Lorem ipsum dolor sit amet, consectetuer
adipiscing elit, sed diam nonummy nibh euismod tincidunt ut
laccet dolore magna aliquam erat volutpat. Ut wisis enim
ad minim veniam.quis nostrud exerci tution ullamcorper
suscipit lobortis nisl ut aliquip ex ea commodo consequat.

</p>
<p>Lorem ipsum dolor sit amet, consectetuer adipiscing
elit,sed diam nonummy nibh euismod tincidunt ut laccet
dolore magna aliquam erat volutpat. Ut wisis enim ad minim
veniam.quis nostrud exerci tution ullamcorper suscipit
lobortis nisl ut aliquip ex ea commodo consequat.</p>

</body>
</html>

```

5. Start by styling the `<h1>` element with a 1-em margin. Add the margin property to the existing `<h1>` style rule:

```
h1 {text-align: center; margin: 1em;}
```

6. The `<h2>` element has a margin-bottom property of 1 em. Add this property to the existing style rule:

```
h2 {text-align: center; margin-bottom: 1em;}
```

7. The `<h3>` has the most complex style effects, so break the style rule into separate lines. Start by adding .5 em of padding to the top and bottom of the element. Add the properties to the existing style rule:

```
h3 { text-align: center;
padding-top: .5em;
padding-bottom: .5em;
}
```

8. Now add top and bottom borders to the `<h3>`. Set the style to solid and the weight to thin. Because the finished rules are black, you do not have to state a color:

```
h3 { text-align: center;
padding-top: .5em;
padding-bottom: .5em;
```

```
border-top: solid thin;  
border-bottom: solid thin;  
}
```

9. Finish the `<h3>` by setting the left and right margins to 40 pixels:

```
h3 { text-align: center;  
padding-top: .5em;  
padding-bottom: .5em;  
border-top: solid thin;  
border-bottom: solid thin;  
margin-left: 40px;  
margin-right: 40px;  
}
```

10. Finish styling the document by setting the `<p>` left and right margins to 40 pixels, to line up with the borders of the `<h3>`:

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

Figure 9-21 shows the results of the style rules.



Figure 9-21 The finished Web page

---

## 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 special box properties let you create interesting text effects such as floating text boxes without the use of HTML table elements.

- 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 pixels.
- 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 special box properties let you create floating images or text boxes.
- Remember to use margin, border, and padding properties judiciously to enhance the legibility of your content, rather than just for novelty effects.

---

## REVIEW QUESTIONS

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 are the two types of color values?
14. What does the float property let you do?
15. What does the clear element let you do?
16. Write a style rule for a `<p>` element that sets margins to 2 em, padding to 1 em, and a black, solid 1-pixel border.
17. Write a style rule for an `<h1>` element that sets top and bottom padding to .5 em with a dashed, thin, red border on the bottom.
18. Write a style rule for a `<p>` element that creates left and right padding of 1 em, 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 increase the legibility of the page content.
3. In this project, you will create a floating text box.
  - a. Copy the **float.htm** file from the Chapter09 folder provided with your Data Files to a Chapter09 folder in your work folder. (Create the Chapter09 folder, if necessary.)
  - b. Open the file **float.htm** in your text editor and save it in your work folder as **float1.htm**.
  - c. In your browser, open the file **float1.htm**. When you open the file it looks like Figure 9-22.

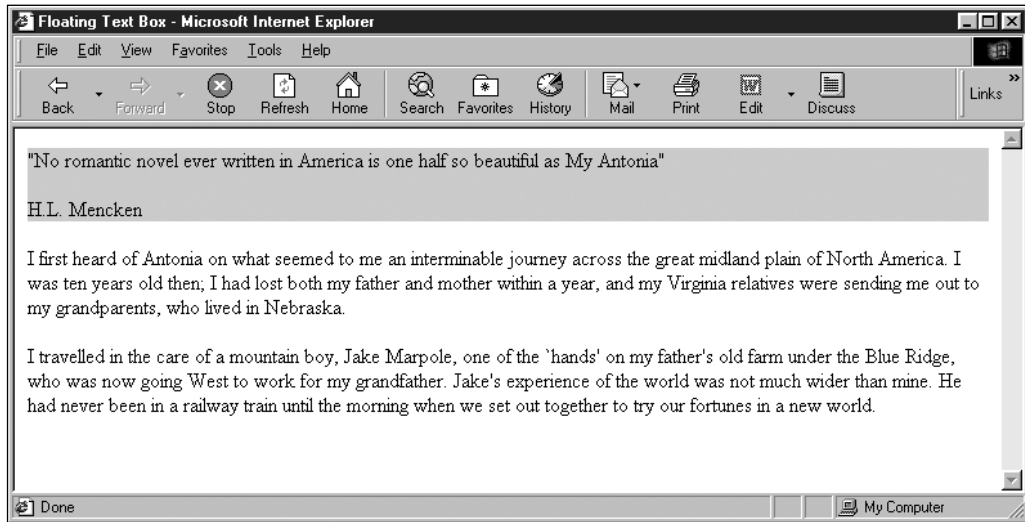


Figure 9-22 The original XHTML file for Project 3

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

```
.floatbox {background-color: #ccc;}
```

- e. This class is applied to the first <p> element in the document, as shown in Figure 9-22. Your goal is to use a variety of box properties to create a finished page that looks like Figure 9-23.

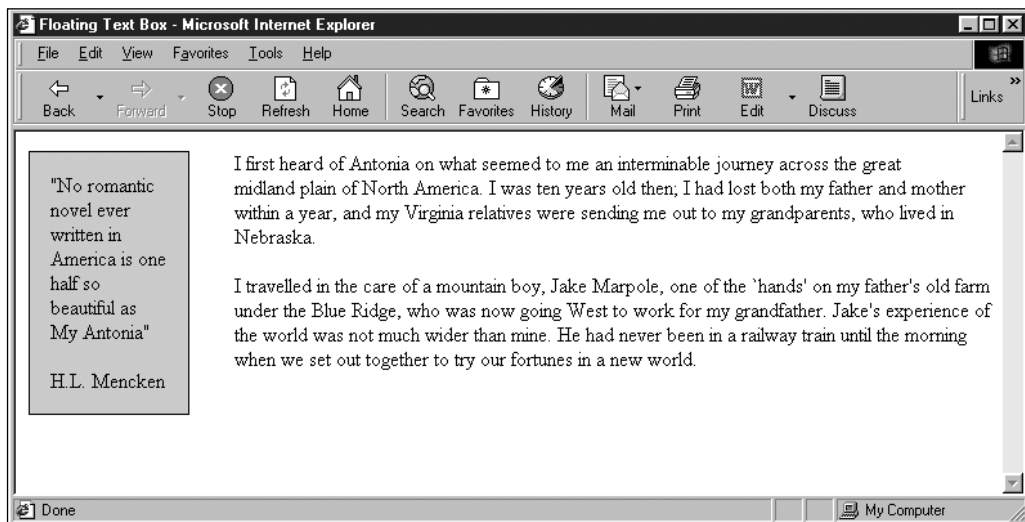


Figure 9-23 The finished XHTML file for Project 3

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

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

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 text editor, create a simple HTML file (or open an existing file) that contains heading and paragraph elements. Save the file in your Chapter09 folder as **borders.htm**.
  - 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.

---

## CASE PROJECT



### CASE PROJECTS

Create the box element spacing conventions for your Web site. Build on the typographic classes you created in Chapter 7. 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 XHTML page that shows examples of the different typefaces and sizes and how they will be used. Decide on margins, padding, and borders and which elements will benefit from their use. Create before and after sample XHTML pages that reflect the enhanced design.