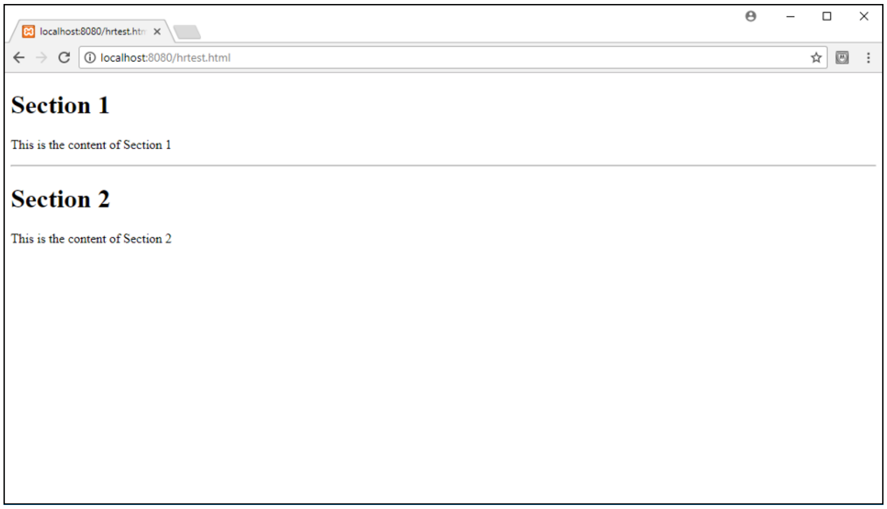


Section 1

This is the content of Section 1

Section 2

This is the content of Section 2

FIGURE 1-4: Using the hr element in a web page.

Marking Your Text

The opposite of block-level elements are *text-level elements.* Text-level elements allow you to apply styles to a section of content within a block. This section shows you the text-level elements you can apply to the content in your web page.

Formatting text

The text-level elements apply predefined formats to text without the need for CSS styling. The most popular of the text-level elements are the *b* and *i elements*, which apply the bold and italic styles, respectively:

```
<p>I <i>wanted</i> the <b>large</b> drink size.</p>
```

Text-level elements are also called *inline*, because they appear in the same line as the content. You can embed text-level elements to apply more than one to the same text:

```
<p>I wanted the <b><i>large</i></b> drink size.</p>
```

REMEMBER

When applying two or more text-level elements to text, make sure you close the tags in the opposite order that you open them.

HTML5 supports lots of different text-level elements for using different styles of text directly, without the help of CSS. Table 1-2 lists the text-level elements available in HTML5.

TABLE 1-2 **HTML5 Text-Level Elements**

Element	Description
`abbr`	Displays the text as an abbreviation
`b`	Displays the text as boldface
`cite`	Displays the text as a citation (often displayed as italic)
`code`	Displays the text as program code (often displayed with a fixed-width font)
`del`	Displays the text as deleted (often displayed with a strikethrough font)
`dfn`	Displays the text as a definition term (often displayed as italic)
`em`	Displays the text as emphasized (often displayed as italic)
`i`	Displays the text as italic
`ins`	Displays the text as inserted (often displayed with an underline font)
`kbd`	Displays the text as typed from a keyboard (often as a fixed-width font)
`mark`	Displays the text as marked (often using highlighting)
`q`	Displays the text as quoted (often using quotes)
`samp`	Displays the text as sample program code (often displayed with a fixed font)
`small`	Displays the text using a smaller font than normal
`strong`	Displays the text as strongly emphasized (often using boldface)
`sub`	Displays the text as subscripted
`sup`	Displays the text as superscripted
`time`	Displays the text as a date and time value
`var`	Displays the text as a program variable (often using italic)

As you can see in Table 1-2, you have lots of options for formatting text without even having to write a single line of CSS code!

Using hypertext

In Book 1, Chapter 1, I mention that hyperlinks are the key to web pages. Hyperlinks are what tie all the individual web pages in your website together, allowing site visitors to jump from one page to another.

The element that creates a hyperlink is the *anchor* text-level element. At first, that may sound somewhat counterintuitive — you'd think an anchor would keep you in one place instead of sending you someplace else. But think of it the other way around: The anchor element is what anchors another web page to your current web page. Following the anchor takes you to the other web page!

Formatting a hyperlink

Because the anchor element is a text-level element, you use it to mark text inside a block. That text then becomes the hyperlink. You add an anchor element using the `<a>` tag. The anchor element is two-sided, so it has both an opening tag (`<a>`) and a closing tag (`</a>`). The text inside the opening and closing tags becomes the hyperlink text.

A few different attributes are available for the `<a>` tag, but the most important one is the `href` attribute. The `href` attribute specifies where the hyperlink takes your site visitors:

```
<a href="http://www.google.com">Click here to search.</a>
```

When a site visitor clicks the hyperlink, the browser automatically takes the visitor to the referenced web page in the same browser window. If you prefer, you can also specify the `target` attribute, which specifies how the browser should open the new web page. Here are your options for the `target` attribute:

- `_blank`: Opens the specified page in a new tab or window.
- `_self`: Opens the specified page in the current tab or window. This is the default behavior in HTML5, so it's not necessary to add it unless you want to for clarification in your code.
- `_parent`: Opens the specified page in the parent window of a frame embedded within the window. Embedded frames aren't popular anymore in HTML5, so this option is seldom used.
- `_top`: Opens the specified page in the main window that contains the frame embedded within other frames. This is seldom used.

You use the target attribute like this:

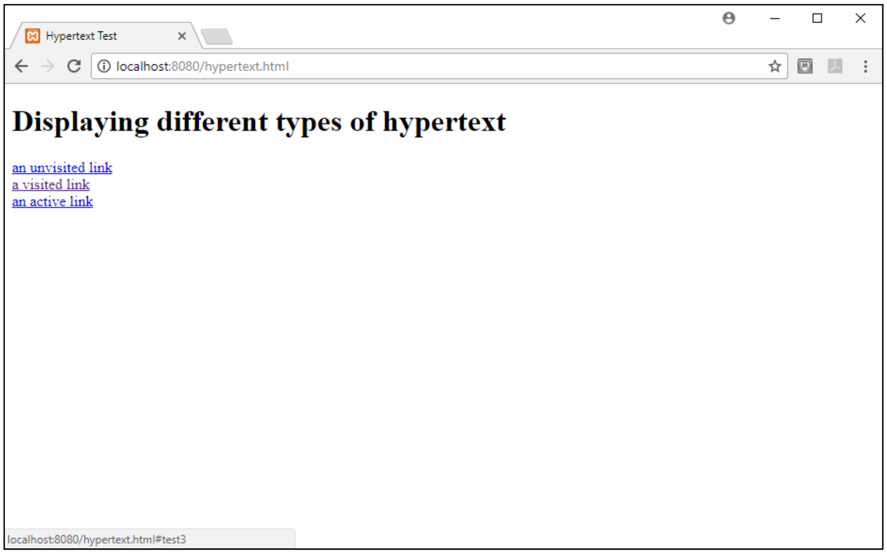
```
<a href="http://www.google.com" target="_blank">Click here to search.</a>
```

TIP

There's no set rule regarding how to handle opening new web pages, but generally it's a good idea to open other pages on your own website in the same browser tab or window, but open remote web pages in a new tab or window. That way your site visitors can easily get back to where they left off on your website if needed.

Displaying a hyperlink

When you specify a hyperlink in the text, the browser tries to make it stand out from the rest of the text, as shown in Figure 1-5.



Displaying different types of hypertext

an unvisited link
a visited link
an active link

localhost:8080/hypertext.html#test3

FIGURE 1-5: Displaying hypertext in a document.

By default, browsers will display the anchor element text using a different format than the rest of the block text:

- Unvisited links appear underlined in blue.
- Visited links appear underlined in purple.
- Active links are when you click an unvisited or visited link with your mouse. When you click your mouse, the link becomes active and appears underlined in red.

You can change these formats to your own liking using CSS styles, as I explain in the next chapter.

Specifying a hyperlink

The `href` attribute defines the location of the web page that you want the browser to open for your site visitor, but there are a few different formats you can use to specify that location:

» A different location on the same document

» A separate web page in the same website

» A web page in a remote website

You can use hyperlinks to force the browser to jump to a specific location inside the same web page. This is handy for long web pages that require lots of scrolling to get to sections at the bottom of the page. To use this method, you must first identify the locations in the web page by applying the `id` attribute to a block-level element, such as a heading or a paragraph element:

```
<h1 id="chicago">Chicago News</h1>
```

To create an anchor element to jump to that section, you use the `id` attribute value, preceded by a number sign or hash mark (`#`):

```
<a href="#chicago">See Chicago News</a>
```

When the site visitor clicks the link, the browser automatically scrolls to place the section in the viewing area of the window.

When jumping to another web page on the same server, you don't need to include the full `http://` address in the `href` attribute. Instead, you can specify a *relative address.* The relative address isn't where your uncle lives; it's shorthand for finding another file on the same web server. If the file is in the same folder on the same server, you can just specify the filename:

```
<a href="store.html">Shop in our online store.</a>
```

You can also place files in a subfolder under the location of the current web page. To do that, specify the subfolder without a leading slash:

```
<a href="store/index.php">Shop in our online store.</a>
```

In both cases, the browser will send an HTTP request to retrieve the file to the same server where it downloads the original page from.

To specify a web page on a remote website, you'll need to use an *absolute address.* The absolute address specifies the location using the *Uniform Resource Locator* (URL), which defines the exact location of a file on the Internet using the following format:

```
protocol://host/filename
```

The `protocol` part specifies the network protocol the browser should use to download the file. For web pages, the protocol is either `http` (for unencrypted connections) or `https` (for encrypted connections). The `host` part specifies the host name, such as `www.google.com` for Google. The `filename` part specifies the exact folder path and filename to reach the file on the server. If you omit the filename, the remote web server will offer the default web page in the folder (usually, `index.html`).

TIP

You can also specify local filenames using an absolute path address. Just precede the folder name with a forward slash (`/`). The leading forward slash tells the server to look for the specified folder at the `DocumentRoot` location of the web server, instead of in a subfolder from the current location.

Working with Characters

No, I'm not talking about Disneyland. I'm talking about the letters, numbers, and symbols that appear on your web pages. Humans prefer to see information as letters, words, and sentences, but computers prefer to work with numbers. To compensate for that, programmers developed a way to represent all characters as number codes so computers can handle them. The computer just needs a method of mapping the number codes to characters.

Character sets

The character-to-number mapping scheme is called a *character set.* A character set assigns a unique number to every character the computer needs to represent. In the early days of computing in the United States, the American Standard Code for Information Interchange (ASCII) became the standard character set for mapping the English-language characters and symbols in computers.

As the computing world became global, most programs needed to support more than just the English language. The Latin-1 and ISSO 8859-1 character sets became popular, because they include characters for European languages. But that still didn't cover everything!

Because it's supported worldwide, the HTML5 standard required more than just European-language support. The Unicode character set supports characters from all languages of the world; plus, it has room for expansion. Because of its huge size, though, a subset of Unicode, called UTF-8, became more popular. UTF-8 also supports all languages, but with a smaller footprint; it has become the standard for HTML5.

Although the HTML5 standard specifies a default character set, it's a good idea to specify the character set in your web pages so that you're sure the client browser is using the same character set to interpret your content. You do that using the *meta element.* Because the meta element provides additional information about your web page, you have to place it inside the head element section of the HTML code.

The meta element uses the single-sided `<meta>` tag. To specify the character set in HTML5 you use the following format:

```
<meta charset="UTF-8">
```

If your HTML code requires a different character set, you specify it here.

TIP

The `<meta>` tag allows you to specify other features of your web page to the browser so that it knows how to process the body of the web page, and identify the content of the web page to servers that automatically scan your web pages for search engines. I talk some more about the `<meta>` tag in Book 4, Chapter 4.

Special characters

The UTF-8 character set supports lots of fancy characters that you won't find on your keyboard, such as the copyright symbol (©), the cent symbol (¢), and the degree symbol (°). These are commonly referred to as *special characters.*

You can use special characters in your web page content because they're valid UTF-8 characters. You just need to use a different way of specifying them. Instead of typing these characters using your keyboard, you use a code to specify them.

HTML5 uses two types of codes to specify special characters:

» **Numeric character reference:** The numeric character reference uses the UTF-8 numeric code assigned to the character. Place an ampersand (&) and a hash (#) in front of the character number, and a semicolon (;) after the character number. For example, to display the copyright symbol, use the following:

```
&#169;
```

» **Character entity reference:** The character entity reference uses a short name to represent the character. Place an ampersand (&) in front of the character short name, and a semicolon (;) after the character short name:

```
&copy;
```

Both methods work equally well, so use whichever method you're most comfortable with. The list of special characters available in UTF-8 is pretty long, so I won't include them here. If you search the web for *UTF-8 characters,* you'll find plenty of websites that show the mappings between the UTF-8 numbers and character short names.

Making a List (And Checking It Twice)

The world is full of lists — to-do lists, wish lists, grocery lists . . . the list just goes on and on. It's no surprise that the HTML5 developers created a way to easily present lists in web pages. There are three basic types of lists available for you to use in HTML5: unordered lists, ordered lists, and description lists. This section covers how to use each type of list in your web pages.

Unordered lists

Some lists have no specific order to the items contained in them (like a grocery list). In the word-processing world, these are called *bulleted lists,* as each item in the list is preceded by a generic bullet icon. In HTML5, they're referred to as *unordered lists.*

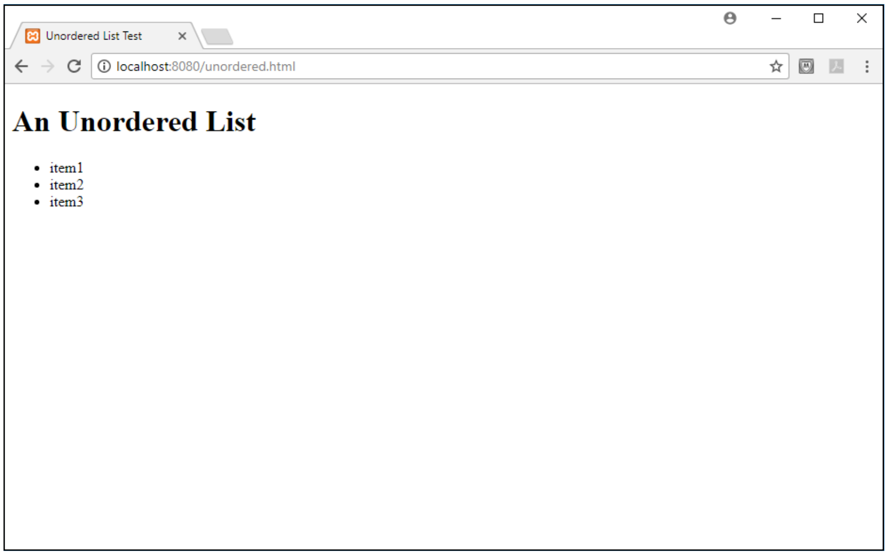
The HTML5 standard uses the *ul element* to display an unordered list using bullets. The ul element is a two-sided element, so you use the `<ul>` tag to open the list and the `</ul>` tag to close the list.

You must identify each item in the list using the *li element.* The li element is also a two-sided element, so you use the `<li>` tag to open each item description and the `</li>` tag to close it. The overall structure for an unordered list looks like this:

```
<ul>
   <li>item1</li>
   <li>item2</li>
   <li>item3</li>
</ul>
```

Because HTML5 doesn't care about white space in code, it's common to indent the list items in the definition as shown here, to help make it easier to read the code. However, indenting isn't necessary.

Figure 1-6 shows the default way most browsers display unordered lists in the web page.



An Unordered List

- item1
- item2
- item3

FIGURE 1-6: Displaying an unordered list.

The bullet marks are fairly generic, similar to what you'd see in most word-processing documents. Fortunately, you can spice things up a little using CSS by defining different types of bullets to use.

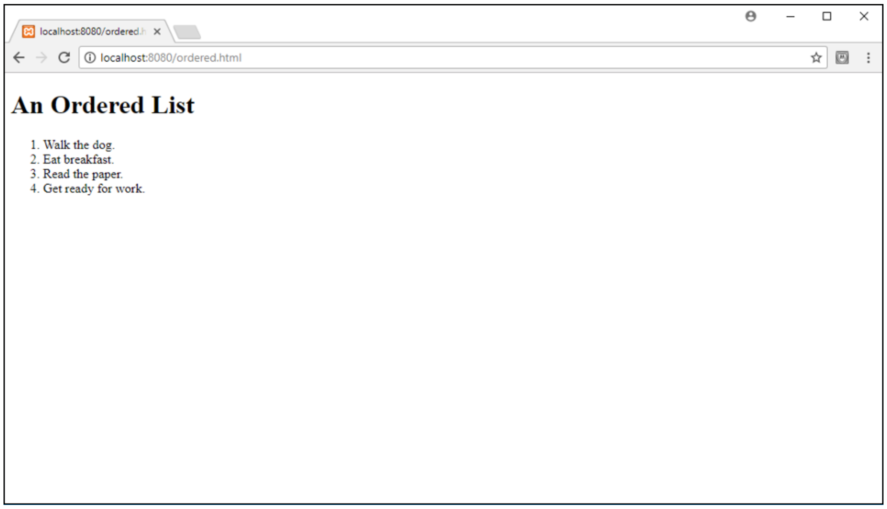
Ordered lists

Some lists have a specific order in which the items should appear and be processed. In word-processing, these lists are called *numbered lists.* In HTML5, they're called *ordered lists.*

The HTML5 standard uses the *ol element* to display an ordered list. The ordered list also uses the li element to mark the individual items contained in the list:

```
<ol>
   <li>Walk the dog.</li>
   <li>Eat breakfast.</li>
   <li>Read the paper.</li>
   <li>Get ready for work.</li>
</ol>
```

By default, browsers assign each item in the list a number, starting at 1, and increasing for each list item, as shown in Figure 1-7.


localhost:8080/ordered.html

An Ordered List

1. Walk the dog.
2. Eat breakfast.
3. Read the paper.
4. Get ready for work.

FIGURE 1-7: The display default for an ordered list.

If you want the list to be in reverse order, add the `reversed` attribute:

```
<ol reversed>
```

If you'd like to start at a different number, add the `start` attribute, and specify the starting number as the value:

```
<ol start="10">
```

If you don't want to use numbers, there are a few other options available with the `type` attribute. Table 1-3 shows the different ordered list types available.

TABLE 1-3 **Ordered List Types**

Type	Description
1	Numerical list (the default)
A	Alphabetical list, using uppercase
a	Alphabetical list, using lowercase
I	Roman numerals, using uppercase
i	Roman numerals, using lowercase

As you can probably guess, you can also embed lists within lists:

```
<ol type="I">
   <li>First item</li>
   <ol type="A">
      <li>Item 1, Subitem 1</li>
      <li>Item 1, Subitem 2</li>
   </ol>
   <li>Second item</li>
   <ol type="A">
      <li>Item 2, Subitem 1</li>
      <li>Item 2, Subitem 2</li>
   </ol>
</ol>
```

WARNING

When using embedded lists, it's very important to match up the opening and closing tags for each item in the list, as well as the lists themselves. Any mismatches will confuse the browser and will cause unpredictable results.

Description lists

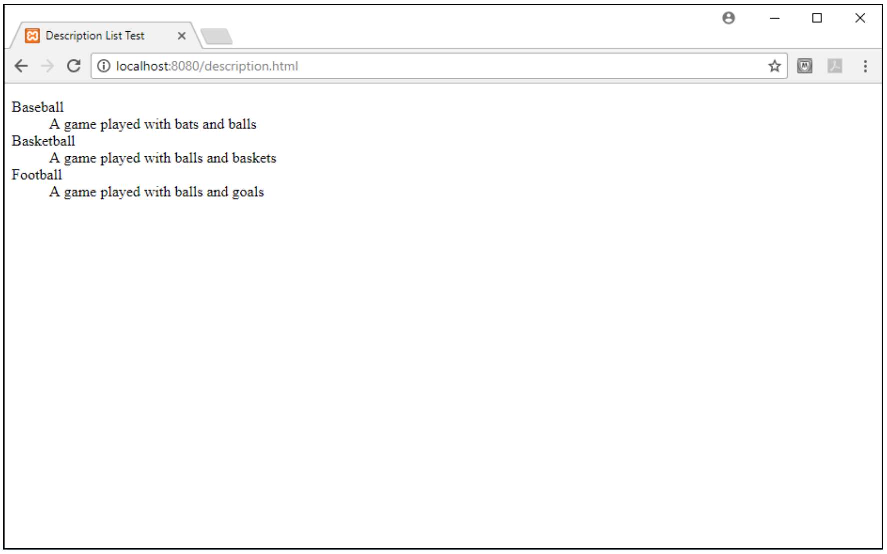
Another common use of lists is to provide descriptions for terms, such as a glossary. The HTML5 standard uses *description lists* to provide an easy way to do that.

Description lists use the *dl element* to define the list but use a slightly different method of defining the items in the list than the unordered and ordered lists. The description list breaks the items into terms and descriptions. You define a term using the *dt two-sided element* and the associated description using the *dd two-sided element.*

Because it's important to match the correct term with the correct description, be careful to alternate between the two in the list definition:

```
<dl>
<dt>Baseball</dt>
<dd>A game played with bats and balls</dd>
<dt>Basketball</dt>
<dd>A game played with balls and baskets</dd>
<dt>Football</dt>
<dd>A game played with balls and goals</dd>
</dl>
```

Figure 1-8 shows how this table is rendered in the browser.



Baseball
    A game played with bats and balls
Basketball
    A game played with balls and baskets
Football
    A game played with balls and goals

FIGURE 1-8: Displaying a description list.

The browser automatically separates the terms from the descriptions in the display, making it easier to tell which is which.

Building Tables

No, don't get out your hammer and saw. I'm talking about data tables. The world is filled with data, and a very common use of web pages is to present that data to the world. This section describes the data table features built into HTML5 that you can use to easily present your data. The general process for creating a table involves three steps:

1. **Define the table element.**
2. **Define the table rows and columns.**
3. **Define the table headings.**

This section walks through each of these steps to show you how to create tables for your data.

Defining a table

To add a table to your web page, you start out with the HTML5 *table element.* The table element is a two-sided element, so it opens with a `<table>` tag and closes with a `</table>` tag:

```
<table>
</table>
```

That creates the table, but it’s not too exciting because there’s nothing in it yet. The next step is to define cells for the data.

WARNING

Prior versions of HTML added attributes to the `<table>` tag to define the table appearance, such as the border type, cell spacing, and width. HTML5 has dropped all these attributes, so avoid using them if possible. You should now define those features using CSS styles instead.

Defining the table’s rows and columns

If you’re familiar with standard spreadsheet software, such as Microsoft Excel or Apple Numbers, you’re used to defining tables using cells, referenced by letters (for the columns) and numbers (for the columns). Unfortunately, HTML5 uses a different method for defining table cells.

To build the cells in a table you must define two separate elements:

» **A row in the table:** You use the *tr element* to define the row inside the table. The tr element is a two-sided element, so you use the `<tr>` tag to open a row and the `</tr>` tag to close the row.

» **The cell inside the row:** Inside the row you use the *td element* to define individual cells. Again, the td element is two-sided, so you use the `<td>` tag to open a cell and the `</td>` tag to close a cell.

So, with all that info, you can create your first table. Just follow these steps:

1. **Open your text editor, program editor, or IDE package and type the following code:**

```
<!DOCTYPE html>
<html>
<head>
<title>My First Table</title>
</head>
<body>
<h1>Bowling Scores</h1>
<table>
```

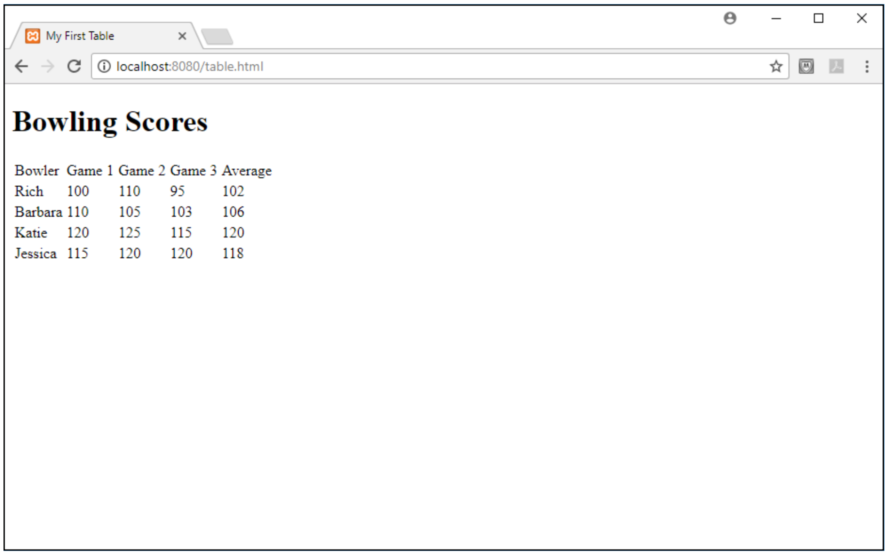
```
   <tr>
      <td>Bowler</td>
      <td>Game 1</td>
      <td>Game 2</td>
      <td>Game 3</td>
      <td>Average</td>
   </tr>
   <tr>
      <td>Rich</td>
      <td>100</td>
      <td>110</td>
      <td>95</td>
      <td>102</td>
   </tr>
   <tr>
      <td>Barbara</td>
      <td>110</td>
      <td>105</td>
      <td>103</td>
      <td>106</td>
   </tr>
   <tr>
      <td>Katie</td>
      <td>120</td>
      <td>125</td>
      <td>115</td>
      <td>120</td>
   </tr>
   <tr>
      <td>Jessica</td>
      <td>115</td>
      <td>120</td>
      <td>120</td>
      <td>118</td>
   </tr>
</table>
</body>
</html>
```

2. **Save the file in the XAMPP `DocumentRoot` folder as `mytable.html`.**
3. **Make sure the XAMPP servers are running.**

4. **Open your browser and enter the following URL:**

```
http://localhost:8080/mytable.html
```

You may need to change the 8080 port number in the URL to match the Apache web server in your setup. When you display the web page it should look like Figure 1-9.



Bowling Scores

```
Bowler  Game 1 Game 2 Game 3 Average
Rich    100    110    95     102
Barbara 110    105    103    106
Katie   120    125    115    120
Jessica 115    120    120    118
```

FIGURE 1-9: Displaying the table in Chrome.

By default, the table doesn't contain any gridlines, but you can change that using CSS, as you see in the next chapter. Also, the table column headings appear just like the data rows. You fix that next.

Defining the table headings

You can apply special formatting to table headings without the use of CSS by using the *th element* instead of the td element for the heading cells:

```
<tr>
   <th>Bowler</th>
   <th>Game 1</th>
   <th>Game 2</th>
   <th>Game 3</th>
   <th>Average</th>
</tr>
```

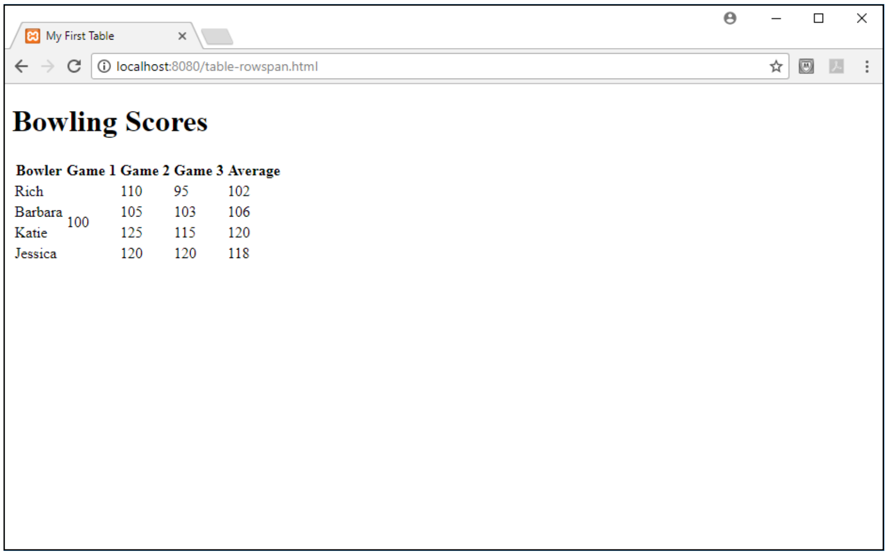
The th element causes the browser to display the heading cells using a bold font.

Often, in tables, you'll run into situations where a data cell must span two or more columns or rows. You can emulate that in your HTML5 tables using the `rowspan` and `colspan` attributes in the `<td>` tag.

To span two or more rows in a single data cell, just add the `rowspan` attribute, and specify the number of rows to span. For example, if all the bowlers had the same score in the first game, you could do this:

```
<tr>
   <td>Rich</td>
   <td rowspan=4>100</td>
   <td>110</td>
   <td>95</td>
   <td>102</td>
</tr>
```

Now the second cell will span the next four rows in the table. Remember, though, when entering data for the other three rows, you must omit the first cell of data, because the first row will take up that space, as shown in Figure 1-10.



Bowling Scores

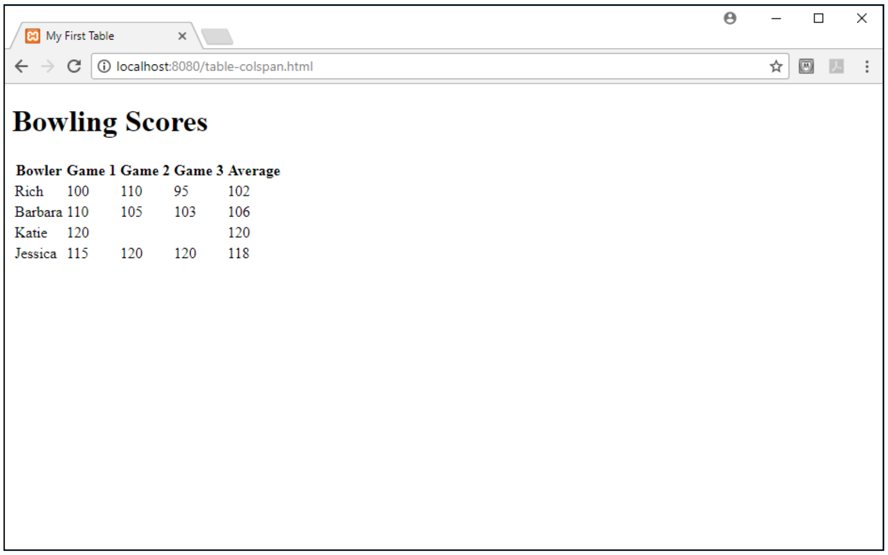
Bowler	Game 1	Game 2	Game 3	Average
Rich	100	110	95	102
Barbara		105	103	106
Katie		125	115	120
Jessica		120	120	118

FIGURE 1-10: Using the `rowspan` attribute in a table.

Likewise, if one of the bowlers had the same score in all three games, you could use the `colspan` attribute to combine all three columns into one cell:

```
<tr>
   <td>Katie</td>
   <td colspan=3>120</td>
</tr>
```

Now the second cell in the row will span all three data columns for that row, as shown in Figure 1-11.



Bowling Scores

Bowler	Game 1	Game 2	Game 3	Average
Rich	100	110	95	102
Barbara	110	105	103	106
Katie	120			120
Jessica	115	120	120	118

FIGURE 1-11: Using the `colspan` attribute in a table.

IN THIS CHAPTER

» **Defining styles**

» **Formatting text**

» **Using the box model**

» **Sprucing up your tables**

» **Positioning elements where you want them**

Chapter 2 The Basics of CSS3

In the last chapter, I explain how to use HTML5 to display content on your web page. However, when you just use HTML5, things look pretty boring! This chapter shows you how to incorporate style into your web pages to help liven things up (even if you're not an artist).

First, I explain how to use CSS style sheets to style elements contained in the web page. Then I show you how to work with styles to change the color and font of text, make fancier lists, and spruce up your tables within your web pages. Finally, I explain how to work with the CSS positioning features to arrange your content in an appealing manner on the page.

Understanding Styles

When you specify an HTML5 element in your web page, the web browser decides just how that element should look. Browsers use a default styling to determine the difference between the text in an h1 element and the text in a blockquote element.

Fortunately, another standard is available to work with HTML5 that helps you make your web pages unique. Back in Book 1, Chapter 1, I explain how Cascading Style Sheets (CSS) work to style HTML5 content on the web page. That's the key to making your website stand out from the crowd!

The CSS standard specifies ways to define the color, size, and font type of text that appears on web pages. It also provides some styles for adding background colors and images and styling other types of elements on the web page.

TIP

The CSS standard has evolved some over the years. At the time of this writing, it's currently at version 3 — you'll often see it referred to as CSS3, and that's what I call it in this book.

Now you're ready to take a deeper dive into just how to use CSS3 in your web applications. This section walks through how CSS3 works and how you can use it to make your web pages look good.

Defining the rules of CSS3

CSS3 uses *rules* to define how the browser should display content on the web page. Each rule consists of two parts: a *selector* that defines what elements the rule applies to and one or more *declarations* that define the style to apply.

The format of the CSS3 rule looks like this:

```
selector {declaration; declaration; ...}
```

In the rule definition, there are five ways to define the selector:

- **Element type:** The rule applies to all elements of the specified type.
- `id` **attribute:** The rule applies to the specific element with the specified id value.
- `class` **attribute:** The rule applies to all elements with the specified class value.
- **Pseudo-element:** The rule applies to a specific part within an element.
- **Pseudo-class:** The rule applies to elements in a specific state.

Each declaration defines a CSS3 style property and its associated value. Each property sets a specific style (such as a color or a font) to the element the rule applies to. You must end each declaration with a semicolon, and you can list as many declarations as needed in the rule.

Here's the format of the property and its value as you list them in the declaration:

```
property: value
```

In the following sections, I explain in more detail the five ways to define a selector.

Element type

You can apply the same styling to all elements of a specific type in your web page by specifying the element name as the selector:

```
h1 {color: red;}
```

This rule ensures that the browser displays all h1 elements in the web page using a red font color.

If you want to apply the same styles to multiple types of elements, you can either list them as separate rules or group the elements together in the selector by separating the element names with commas, like this:

```
h1, p {color: red;}
```

This rule instructs the browser to style all h1 and p elements using a red font color.

id attribute

If you need to define a rule that applies to just a single element in the web page, use the `id` attribute as the selector. To specify an `id` attribute as the selector, place a pound sign (`#`) in front of the id name:

```
#warning {color: red;}
```

To use the rule in your HTML5 code, just apply the `id` attribute value to the element you need to style:

```
<h1 id="warning">This is a bad selection.</h1>
```

The browser will apply that rule to the specific element that uses the `id` attribute value.

class attribute

If you need to define a rule that applies to multiple elements, but not necessarily all the elements of that type, use the `class` attribute as the selector. To specify a `class` attribute as the selector, place a period in front of the class name:

```
.warning {color: red;}
```

Then just apply that class attribute to whichever elements you need to style using that rule:

```
<h1 class="warning">This is a bad selection.</h1>
<p class="warning">Please make another selection.</p>
```

As you can see from this example, you can apply the same class attribute value to different element types, making this a very versatile way of styling sections of your web page!

If you decide you only need to apply a rule to one particular element type of the class, you can specify the element type in the selector before the class value:

```
p.warning {color: red;}
```

This rule will apply only to p elements with the `class` attribute value of `warning`.

Pseudo-element

The CSS standard defines a handful of special cases where you can apply styles to a subsection of the element content, and not the entire content of an element. These rules are called *pseudo-elements.*

To use a pseudo-element rule, separate the rule from the selector it applies to using a double colon (`::`):

```
selector::pseudo-element
```

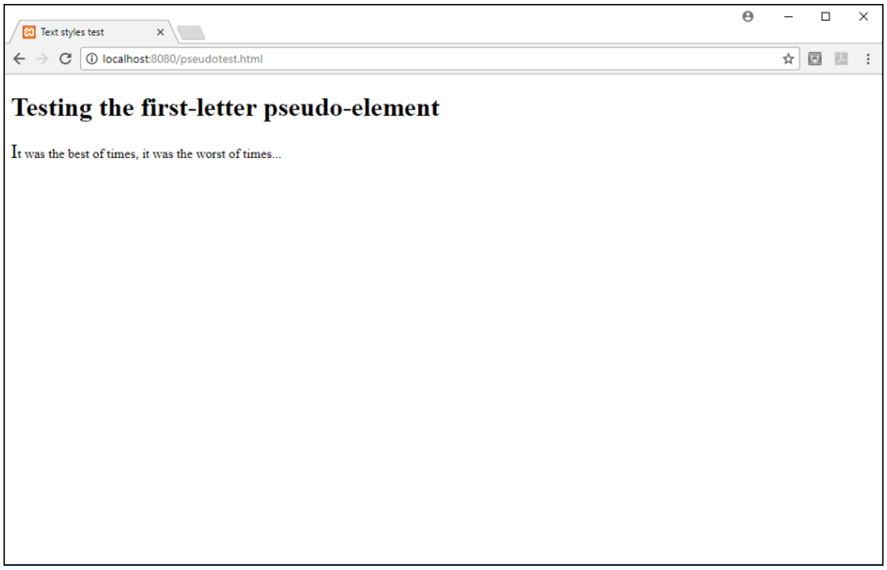
CSS3 supports a set of keywords for the pseudo-element names. For example, if you want to get fancy and style the first letter of a paragraph of text differently from the rest of the text, you can use the `first-letter` pseudo-element keyword:

```
p::first-letter {font-size: 20px}
```

The `first-letter` pseudo-element matches with only the first letter of the p element, as shown in Figure 2-1.

CSS3 defines only a handful of pseudo-elements. Table 2-1 lists them.

There aren't a lot of pseudo-elements available, but these few pseudo-elements can come in handy for trying special formatting of your web pages.



Testing the first-letter pseudo-element

It was the best of times, it was the worst of times...

FIGURE 2-1: Using the first-letter pseudo-element on text.

TABLE 2-1 CSS3 Pseudo-Elements

Pseudo-Element	Description
`after`	Places an object before the selected element content
`before`	Places an object after the selected element content
`first-letter`	Applies the rule to the first letter of the selected element content
`first-line`	Applies the rule to the first line of the selected element content
`selection`	Applies the rule to the content area selected by the site visitor

TIP

The `after` and `before` pseudo-elements may sound a bit strange, because there's no content to style before or after an element. They're most commonly used for placing images after or before the content in the element.

Pseudo-class

A *pseudo-class* applies the defined styles to an element that is in a specific state on the web page. The state refers to how the element behaves, such as buttons that are disabled, check boxes that are checked, or input boxes that have the browser focus.

These rules are commonly applied to hypertext links on the web page to help site visitors distinguish links they've already visited. You do that by using a series of four pseudo-class style rules:

- `link`: Applies the rule to a normal, unvisited link
- `visited`: Applies the rule to a link that the site visitor has already visited
- `hover`: Applies the rule when the site visitor hovers the mouse pointer over the link
- `active`: Applies the rule when the site visitor clicks the mouse on the link

You specify pseudo-class rules using a single colon to separate it from the selector in the rule definition:

```
a: link {color: orange;}
a: visited {color: purple;}
a: hover {color: green;}
a: active {color: red;}
```

All these pseudo-class rules apply to all the anchor elements in the web page and apply different colors to the hyperlink text depending on the hyperlink state.

WARNING

It's extremely important to list the anchor element pseudo-class rules in the order shown here, or they won't work!

TIP

If you want to remove the underline that most browsers apply to hypertext links, add the following property to the pseudo-element style rule:

```
text-decoration:none;
```

There are lots of pseudo-classes that you can use to apply rules to specific elements in the your web pages. Table 2-2 shows the list of available pseudo-classes in CSS3.

Many of the pseudo-class style rules (such as `first-child` and `last-child`) work with the location of an element within the Document Object Model (DOM). Book 3, Chapter 2, discusses the DOM and how to use it to reference elements on the web page.

TABLE 2-2 **The CSS3 Pseudo-Classes**

Pseudo-Class	Description
`active`	The rule applies to hypertext links while the site visitor clicks them.
`checked`	The rule applies to input check boxes and radio options that are selected (checked).
`disabled`	The rule applies to input elements that are disabled.
`empty`	The rule applies to elements that have no children.
`enabled`	The rule applies to input elements that are enabled.
`first-child`	The rule applies to the first child element of a parent element.
`first-of-type`	The rule applies to the first child element of the same type as the parent.
`focus`	The rule applies to elements that have the browser focus.
`hover`	The rule applies to elements that the mouse pointer is hovering over.
`in-range`	The rule applies to elements whose value is within the specified range.
`invalid`	The rule applies to elements whose value is invalid.
`lang(`*`language`*`)`	The rule applies to elements with the `lang` attribute specified.
`last-child`	The rule applies to the last child element of a parent element.
`last-of-type`	The rule applies to the last child element of the same type as the parent.
`link`	The rule applies to unvisited hypertext link elements.
`not(`*`selector`*`)`	The rule applies to all elements except the specified selector elements.
`nth-child(`*`n`*`)`	The rule applies to the *n*th child of the parent element.
`nth-last-child(`*`n`*`)`	The rule applies to the *n*th child of the parent element counting backward from the last element.
`nth-of-type(`*`n`*`)`	The rule applies to the *n*th child element with the same type as the parent.
`only-of-type`	The rule applies to every element that is the only element of the same type as the parent.
`only-child`	The rule applies to every element that is the same only child of a parent.
`optional`	The rule applies to input elements that do not have the `required` attribute.
`out-of-range`	The rule applies to elements with a value out of the specified range.
`read-only`	The rule applies to elements with a `readonly` attribute specified.

(continued)

TABLE 2-2 ***(continued)***

Pseudo-Class	Description
`read-write`	The rule applies to elements without a `readonly` attribute specified.
`required`	The rule applies to elements with a `required` attribute specified.
`root`	The rule applies to the document's root element.
`target`	The rule applies to the current active element specified.
`valid`	The rule applies to elements that have a valid value.
`visited`	The rule applies to hypertext links that the site visitor has already visited.

Applying style rules

In Book 1, Chapter 1, I discuss the different ways to apply CSS3 styles to an HTML5 document. To refresh your memory, there are three ways to do that:

- **Inline styles:** Place the style properties inside the HTML5 element opening tag, using the `style` attribute:

  ```
  <h1 style="color: red;">Warning</h1>
  ```

- **Internal styles:** Use the `<style>` tag to define a set of styles that apply to the entire document:

  ```
  <style>
  h1 {color: red;}
  </style>
  ```

- **External styles:** Use an external file to contain the style definitions, and then add the `<link>` tag in the HTML5 document to reference the external style sheet:

  ```
  <link rel="stylesheet" href="mystyles.css">
  ```

Note that with the inline style definitions, you leave off the selector part of the rule. Because the rule applies only to the element that declares it, there's no need for the selector. With both the inline and external style sheet methods, you define the set of rules separately within the style sheet. The great benefit of using the external style sheet method is that you can then apply the same style sheet to all the pages of your website!

You can use any of these locations to define your style rules, or you can use them all at the same time! If two or more style rules apply to the same element on the web page, the cascading feature of CSS3 kicks in. CSS3 defines a specific process on how the browser applies conflicting rules to an element to ensure everything happens in order. The next section explains how that works.

Cascading style rules

As the name suggests, if you define multiple style rules for a web page, the rules cascade down from the lower-priority rules, which are applied first, to the higher-priority rules, which are applied later.

TIP

Saying "down" from a lower to a higher priority may seem counterintuitive, but it's common jargon in CSS circles. Just remember that the higher-priority rules take precedence over the lower-priority rules.

The CSS3 standard defines a strict process for how browsers should apply style rules to each element. In Book 1, Chapter 1, I outline an abbreviated version of the cascading rules. There are actually ten different rule levels that the CSS3 standard defines for applying rules! However, most web designers don't use all ten levels to define rules, so things don't usually get that complicated.

Table 2-3 shows the official CSS3 cascading rules process.

TABLE 2-3 The CSS3 Cascading Rules Process

Rule Type	Description	Priority Level
Importance	Rules contain the `!important` property and override all other rules	1
Inline	Rules defined using the `style` attribute in an element opening tag	2
Media	Rules defined for a specific media type	3
User defined	Accessibility features defined in the browser by the site visitor	4
Specific selector	A selector referring to an id, class, pseudo-element, or pseudo-class	5
Rule order	When multiple rules apply to an element, the last rule declared wins	6
Inheritance	Rules inherited from parent elements in the web page	7
Internal	Rules defined in internal style sheets	8
External	Rules defined in external style sheets	9
Browser default	The default styles built into the browser, the lowest priority	10

REMEMBER

Notice that accessibility features have a special place in the cascading rule order. Many of your website visitors may have some type of viewing disability preventing them from viewing your content as you style it. Most browsers allow users to define their own style features, such as specifying foreground and background contrasting colors or changing the font size to make text more readable.

Now that you've seen how to define CSS3 rules and where to define them, the next step is to start learning some rules to apply to your web pages. The CSS3 standard defines a myriad of styles for you to use. Entire books have been written trying to cover all the different rules and features, such as *CSS3 For Dummies* by John Paul Mueller (Wiley). The remaining sections in this chapter walk you through some of the more commonly used rules that you'll want to keep in mind as you design your dynamic web applications.

Styling Text

No place is styling more important than with the text that appears on your web page. You can transform a dull, boring website with just a few changes to the text styles. This section walks through the options you have available for styling text to help liven up your website.

Setting the font

A *font* defines how a medium displays the characters in the content. Whether it's etching words into stone, setting text on paper using a printing press, or displaying pixels on a computer monitor, fonts help control how readers interpret the content.

When you place text on your web page using HTML5, the browser selects a default font style, size, and format based on the element type, and it uses that same setting for all the text in those elements on your web page. That not only makes for a boring web page, but can also confuse your site visitors if all the content blends together.

This section describes how you can change the font features the browser uses to display text in your web pages.

Finding a family

The CSS3 standard defines the `font-family` style property to allow you to change the style of font. Here's the format for the `font-family` property:

```
font-family: fontlist;
```

The *fontlist* value is a comma-separated list of font names. There are two ways to specify a font in the list:

- **Using a specific font name:** Specific font names require the browser to use that specific font to display the text, such as Times New Roman, Arial, or Helvetica. Browsers are limited to using only the fonts that are installed on the workstation, so specifying a specific font name can be a gamble. If that font isn't available on the site visitor's workstation, the browser will revert to the default font. It has become common practice to provide several options of font names in the `font-family` property. The browser will try to use the font presented first in the list, and if that's not available, it'll try the next font listed, and continue down the list. If no font is available, the browser reverts to the default font.
- **Using a generic font group:** Generic font groups give the browser a little more leeway in selecting a font to use. Instead of looking for a specific font, the browser can use any font that's included in the font group. CSS3 defines the following font groups:
  - `cursive`: A font that mimics handwritten text
  - `fantasy`: An ornamental font used for special text
  - `monospace`: A font that uses the same spacing for all characters
  - `sans-serif`: A font without any ornamentation added to characters
  - `serif`: A font that uses ornamentation at the tail of each character

It's common practice to list specific font names first in the font list and then, as a last resort, add a generic font group, like this:

```
font-family: Arial, 'Times New Roman', sans-serif;
```

With this rule, the browser will try to use the Arial font. If that's not available on the visitor's workstation, it will try to use the Times New Roman font. If Times New Roman is also not available, the browser will look for a font from the sans-serif font group.

REMEMBER

Note that for font names that contain spaces, you must enclose the name in single quotes.

TIP

The CSS3 standard defines an exciting new feature called *web fonts.* Web fonts allow you to define your own font on a server so that browsers can download them along with the web page. I dive into using web fonts in more detail in Chapter 4 of this minibook.

Picking a size

After selecting a font style to use, the next step is to decide what size the font should be. Browsers have built-in sizes for separating out the different header levels, as well as standard text. However, you can change that by using the `font-size` property:

```
font-size: size;
```

You'd think specifying a font size would be easy, but CSS3 actually allows you to specify the size in one of five different methods:

- As an absolute unit of measurement
- As a relative unit of measurement
- As a percentage of the space assigned to the element
- Using a size keyword
- Using a size keyword relative to the space assigned to the element

You specify absolute units using a specific size value of measurement. To complicate things even more, CSS allows you to use six different units of measurements, shown in Table 2-4.

TABLE 2-4 CSS Font-Size Absolute Units of Measurement

Unit	Description
`cm`	Centimeters
`in`	Inches
`mm`	Millimeters
`pc`	Picas
`pt`	Points
`px`	Pixels

The first three units of measurement shown in Table 2-4 are easily recognizable, but the last three aren't as common. There are 6 picas in an inch, and 72 points in an inch. The pixel unit originally matched up to pixels on a standard computer monitor, but with the advancement of monitor technology, that isn't the case anymore.

You can specify the size using either a whole number or a decimal value:

```
font-size: 0.25in;
font-size: 48pt;
```

The relative units of measurement set the size of the font relative to other elements on the web page. Table 2-5 shows the relative size units in CSS3.

TABLE 2-5 **CSS Font-Size Relative Units of Measurement**

Unit	Description
`ch`	Relative to the size of the zero character
`em`	Relative to the size of the normal font size of the elements
`ex`	Relative to the normal height of the font size currently used
`rem`	Relative to the height of the root element
`vh`	Relative to 1% of the browser window height
`vw`	Relative to 1% of the browser window width
`vmax`	Relative to 1% of the larger of the browser window width or height
`vmin`	Relative to 1% of the smaller of the browser window width or height
`%`	As a percentage of the normal element size

The `em` relative unit size is the most popular. It sizes the element relative to the text in the web page. For example, here's a common rule that you'll see:

```
h1 {font-size: 2em;}
```

This tells the browser to size the h1 element twice the size of the text in the web page. By using relative units, you can easily change the size of headings based on the size of the text in the content. If you decide to change the font size of the text in the web page, the headings will automatically change size to stay in the same proportion.

To make things simpler, CSS also allows you to set the text size using a human-readable keyword. There are both absolute and relative keywords available:

- **Absolute:** `xx-small`, `x-small`, `small`, `medium`, `large`, `x-large`, `xx-large`
- **Relative:** `smaller`, `larger`

Using the keywords makes setting font sizes easier, but you're still a little at the mercy of the browser. It's up to the browser to determine just what is a small, medium, or large size font.

Playing with color

By default, browsers display all text in black on a white background. Things don't get any more boring than that! One of the first steps in livening up your website is to change the text color scheme.

There are two CSS3 properties that you need to do that:

- `color`: Selects the color the browser uses for the text font
- `background-color`: Selects the color the browser uses for the background

You have a vast palette of colors to choose from for your color scheme. Usually, it's a good idea to pick a color scheme for your website and try to stick with that for most of the web pages contained in the website. Often, a corporation will set the color scheme of its website based on the colors used in the company logo. Occasionally, you may need some content to pop out at visitors, so you'll need to deviate some from the scheme.

The original CSS standard provided three ways to define colors used in styles:

- **With color names:** You can choose a text value from a standard list of color names. CSS3 defines many different colors by name. If you plan on using a standard color, most likely you can call it just by its name:

  ```
  p {color: red; background-color: white;}
  ```

- **With RGB hexadecimal values:** If you want to fine-tune the colors your web page elements use, you can select the intensity of the red, green, and blue colors based on hexadecimal values from 00 to FF. If you're into hexadecimal numbers, define the color as three hexadecimal values preceded by a pound sign:

  ```
  p {color: #ffa500;}
  ```

  The `ffa500` value sets the red hue at full intensity, sets the green hue a little lower, and turns the blue hue off, producing the orange color.