

10 Creating Navigational Aids

In this chapter, you will learn how to

- ✓ Plan your site's organization.
 - ✓ Create a text-based navigation bar.
 - ✓ Create a graphical navigation bar.
 - ✓ Create an image map.
 - ✓ Redirect to another URL.
-

If you worked through the exercises in Parts 1 and 2 of this book, you have acquired most of the basic skills you need to create simple Web sites. Now it's a matter of putting all these skills together to make attractive and easy-to-use sites, and that's what you'll focus on in Part 3.

One way to make your Web site easily accessible is to place a consistent navigation bar on each page. A *navigation bar* is a set of hyperlinks that connect to the major pages of your Web site. These hyperlinks can be either text-based or graphical. You already saw how to create both kinds of hyperlinks in Chapters 5, "Creating Hyperlinks and Anchors," and Chapter 9, "Displaying Graphics," so creating a navigation bar is a logical next step. You'll learn how to plan your site's organization, and then create a suitable navigation bar to match it.

This chapter also explains a couple of other useful techniques to help users navigate your site. You'll learn how to redirect users from one page to another and how to create an image map that hyperlinks defined areas of a graphic to specific pages.

See Also Do you need only a quick refresher on the topics in this chapter? See the Key Points at the end of this chapter.

Practice Files Before you can use the practice files provided for this chapter, you need to install them from the book's companion content page to their default locations. See "Using the Practice Files" in the beginning of this book for more information.

Planning Your Site's Organization

Navigation bars can be easy to create, but they require some planning to be effective. Up to this point in the book, you've been creating single pages with a common theme for eventual inclusion in a Web site, but you probably have not yet given a lot of thought to how the pages fit together. So before creating a navigation bar, you want to consider the overall structural plan for the site.

A navigation bar should contain links to the most important sections of the Web site, plus a link to the Home page.



The Garden Company

Helping you help your gardens grow since 1975

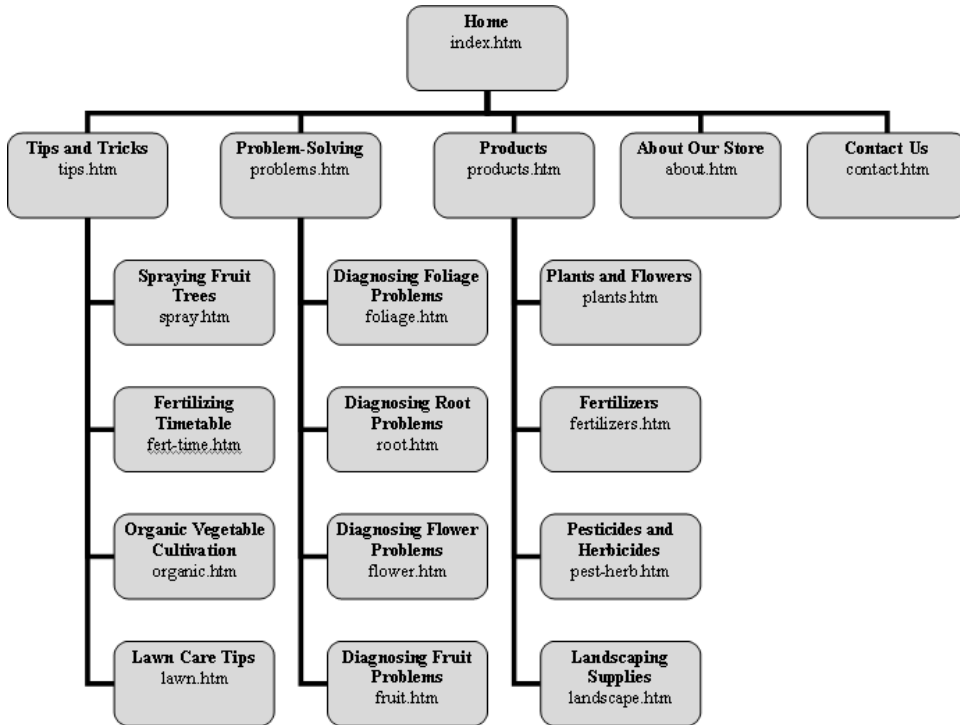
[Home](#) [Tips and Tricks](#) [Problem-Solving](#) [Products](#) [About Our Store](#) [Contact Us](#)

The navigation bar should not contain hyperlinks to every page in the site unless the site is extremely small and simple. Although there is no hard-and-fast rule about the number of items a navigation bar can contain, most people try for somewhere between four and seven. With fewer than four, your site doesn't look very content-rich; with more than seven, the navigation bar becomes crowded and confusing. In addition, on low-resolution displays or in narrow browser windows, your navigation bar might wrap to a second (or even third) line if it's a horizontal bar, or it might force the user to scroll down if it's a vertical bar. This chapter discusses only horizontal bars, but you'll learn how to make vertical navigation bars in Chapter 12, "Creating Tables."

Note Some Web sites have navigation bars in which each hyperlink opens a menu of options when the user points to it or clicks it. You can't create that with plain HTML; those are constructed with JavaScript or another Web-based programming language.

Before building your navigation bar, create a diagram that outlines the site's planned structure. It doesn't matter if you haven't created all the pages yet. You can be as fancy or as plain as you want with your chart. It can be scrawled on the back of a napkin or built using SmartArt (through a Microsoft Office application), Microsoft Visio, or some other charting tool. Choose file names for each planned page, so you can start referring to them in hyperlinks even if they don't exist yet.

The organization of The Garden Company's site, which you've been creating pages for in this book's examples, might look something like this.

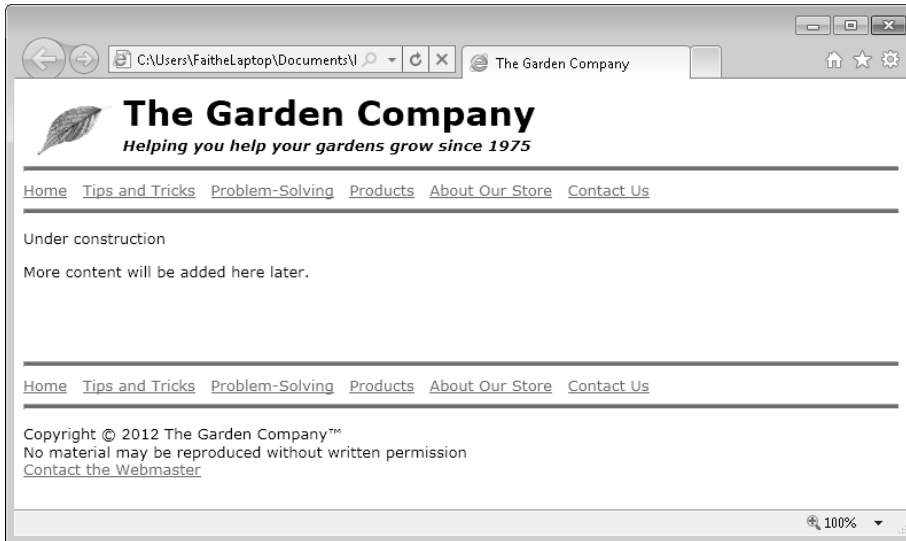


Notice that the level directly below the Home page consists of five pages. The navigation bar will contain hyperlinks to each of these pages. Three of these are introductory pages for larger sections of content; the introductory pages of those sections will link to each page within that section. This Web site is modest in scope initially, but there is plenty of room for future expansion. You could add dozens of additional tips, problem-solving techniques, and products. You could even create subsections within one of the main areas if the content becomes too overwhelming for a single page.

Notice also that not every page referenced from the navigation bar is a major section. Three of them—Home, About Our Store, and Contact Us—are simply pages that are important for visitors to be able to access quickly from any page.

Creating a Text-Based Navigation Bar

A text-based navigation bar is the simplest and easiest, and it is also very user-friendly. On simple Web pages, text-based navigation bars are usually placed at the top of the page, in a single horizontal line. Some Web designers also place a copy at the bottom of each page so visitors don't need to scroll back up to the top of a page to access the links.



Tip When you place a navigation bar at the bottom of a page, it is customarily a text-based bar rather than a graphical one.

HTML5 includes a `<nav>` tag, a two-sided container tag in which you can optionally place the code for a navigation bar. The `<nav>` tag is designed to help browsers and style sheets identify sets of links as a navigational element, and handle them appropriately. If the browser does not support the `<nav>` tag, it is ignored. You'll use the `<nav>` tag in this chapter because it's good practice to start including HTML5 tags in your code, but you won't be doing anything special with the `<nav>` tag's attributes. However, in sites you create yourself, you are free to define style attributes for the `<nav>` tag in internal or external style sheets; this can be a way to help ensure consistency among the navigation bars throughout all the pages in your Web site.

In this exercise, you will add a text-based navigation bar to the top and bottom of a Web page.



SET UP Be sure to use the files provided specifically for this exercise, and not earlier versions. Use the `index.htm` and `default.css` files in the practice folder for this topic. These files are located in the `Documents\Microsoft Press\HTML5 SBS\10Navigation\CreatingTextBar` folder. Open the `index` file in Microsoft Notepad and in Microsoft Internet Explorer.

1. At the first `<hr>` tag, add a `<nav>` container and add the text for a navigation bar.

```
<nav>
<hr>
<p>Home Tips and Tricks Problem-Solving Products About Our Store Contact
Us</p>
</nav>
```

2. Save the file, and then refresh Internet Explorer.

The text of the intended navigation bar appears, but the items are not clearly separated.

Home Tips and Tricks Problem-Solving Products About Our Store Contact Us

HTML ignores multiple spaces, so you must instead use the nonbreaking space code (* *) if you want to insert extra spaces between words without creating a table or some other structural container.

3. Insert a nonbreaking space (and a normal space following it) between each section title, like this:

```
<p>Home &nbsp; Tips and Tricks &nbsp; Problem-Solving &nbsp; Products
&nbsp; About Our Store &nbsp; Contact Us</p>
```

4. Save the file, and then refresh Internet Explorer.

5. To help set off the navigation bar from the rest of the text, insert a second horizontal line below the navigation bar text, but above the closing *</nav>* tag.

```
<nav>
<hr>
<p>Home &nbsp; Tips and Tricks &nbsp; Problem-Solving &nbsp; Products &nbsp;
About Our Store &nbsp; Contact Us</p>
<hr>
</nav>
```

6. Save the file, and then refresh Internet Explorer.

Home Tips and Tricks Problem-Solving Products About Our Store Contact Us

The horizontal spacing looks okay, but the navigation bar would look better if the green lines were closer to it at the top and bottom.

7. Set the margin for the paragraph to zero.

```
<p style="margin:0px">Home &nbsp;   Tips and Tricks &nbsp;   Problem-Solving  
&nbsp;   Products &nbsp;   About Our Store &nbsp;   Contact Us</p>
```

8. Save the file, and then refresh Internet Explorer.

Now the lines are closer to the text.

Home Tips and Tricks Problem-Solving Products About Our Store Contact Us

9. Add hyperlinks to each of the six items in the navigation bar to the corresponding pages.

Note Line breaks are added in the following code for ease of reading, but they are not required.

```
<nav>  
<hr>  
<p style="margin:0px">  
<a href="index.htm">Home</a> &nbsp;    
<a href="tips.htm">Tips and Tricks</a> &nbsp;    
<a href="problems.htm">Problem-Solving</a> &nbsp;    
<a href="products.htm">Products</a> &nbsp;    
<a href="about.htm">About Our Store</a> &nbsp;    
<a href="contact.htm">Contact Us</a></p>  
<hr>  
</nav>
```

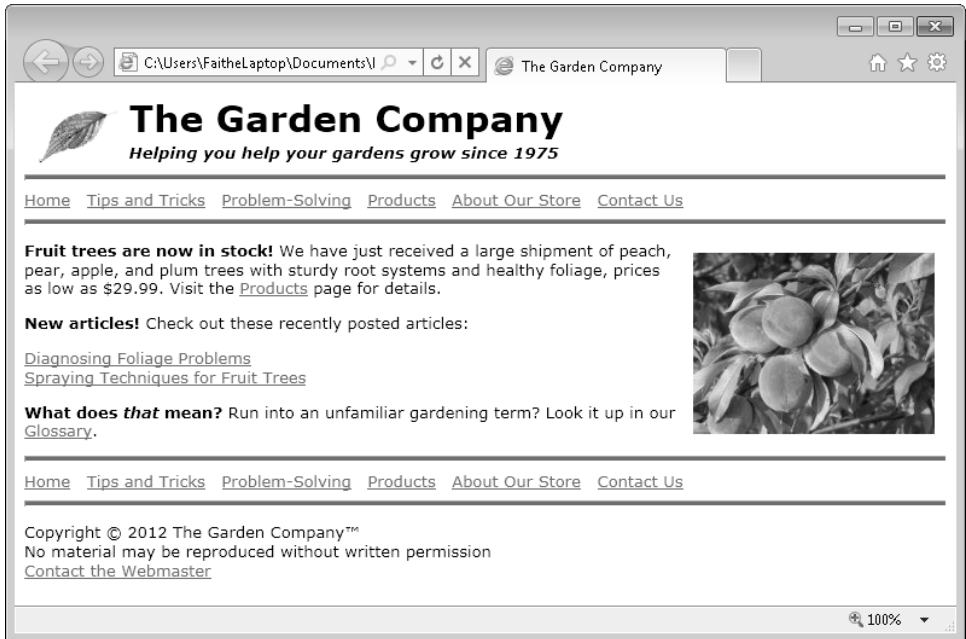
10. Save the file, and then refresh Internet Explorer.

The navigation bar is complete.

[Home](#) [Tips and Tricks](#) [Problem-Solving](#) [Products](#) [About Our Store](#) [Contact Us](#)

11. Select the code for the entire navigation bar, including the `<nav>` and `</nav>` tags, and press **Ctrl+C** to copy it to the Clipboard.
12. Select the `<hr>` tag at the bottom of the document, and press **Ctrl+V** to replace it with a copy of the navigation bar.
13. Save the file, and then refresh Internet Explorer.

Two navigation bars appear, one above and one below the main content of the page.



✕ CLEAN UP Close the Notepad and Internet Explorer windows.

Creating a Graphical Navigation Bar

Text hyperlinks are clear and unambiguous, but not all that attractive. You might prefer to create a navigation bar that uses buttons or other graphics instead of text links. You can create the graphics yourself in a graphics-editing program. If you do create your own, it's a good idea to follow these guidelines:

- Keep the size of each button small (150 pixels wide at the most).
- Make each button the same size and shape. The only variation should be in the text that they present.
- Save each button as a separate file in GIF or JPG format.

If you have no talent or inclination for art projects, search the Web; there are thousands of sites with free graphical buttons that you can download. Make several copies of a

button you like, and then use a text tool in a graphics-editing program to place different text on each copy. Here are a couple of links to free button sites to get you started:

- <http://www.aaa-buttons.com>
- http://www.eosdev.com/eosdev_Buttons.htm

Most professional Web site designers do not create their own buttons, nor do they acquire them from others; instead they use *button-creation programs* to generate them. Such programs make it very easy to create groups of identical buttons with different text on each one. There are both commercial standalone programs that make buttons, and also free Web utilities. Here are a few sites; you can find many more with a simple Web search.

- <http://www.crystalbutton.com>
- <http://www.buttongenerator.com>

Note The buttons provided for the exercises in this book were created with Crystal Button.

You set up a graphical navigation bar just like a text-based navigation bar, but instead of hyperlinks from the text, you hyperlink from the graphic by placing the `` tag within the `<a>` tag, like this:

```
<a href="product.htm"></a>
```

In this exercise, you will convert a text-based navigation bar to a graphics-based one.



SET UP Be sure to use the files provided specifically for this exercise, and not earlier versions. Use the *index.htm* file in the practice folder for this topic. This file is located in the Documents\Microsoft Press\HTML5 SBS\10Navigation\CreatingGraphicBar folder. Open the *index* file in Notepad and in Internet Explorer.

1. In Notepad, in the upper navigation bar, change the hyperlinks so that they reference the button graphics in the */images* folder rather than displaying text.

```
<nav>
<hr>
<p style="margin:0px">
<a href="index.htm"></a>
<a href="tips.htm"></a>
```



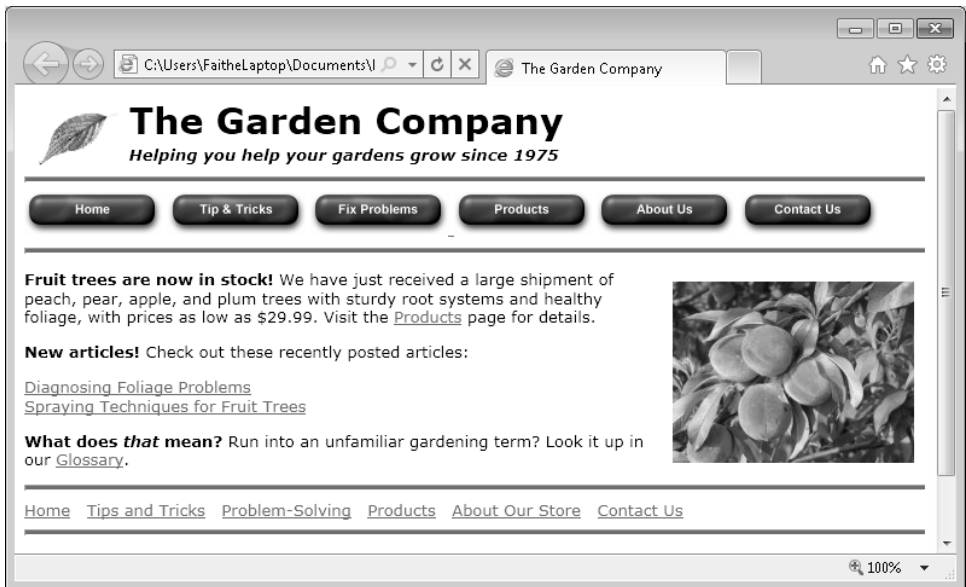
```

<a href="problems.htm">
</a>
<a href="products.htm"></a>
<a href="about.htm"></a>
<a href="contact.htm">
</a></p>
<hr>
</nav>

```

Note The preceding code also removes the spaces you previously placed between the links, because the spacing is now provided by the graphics themselves.

2. Save the file, and then refresh Internet Explorer to view your work.



CLEAN UP Close the Notepad and Internet Explorer windows.

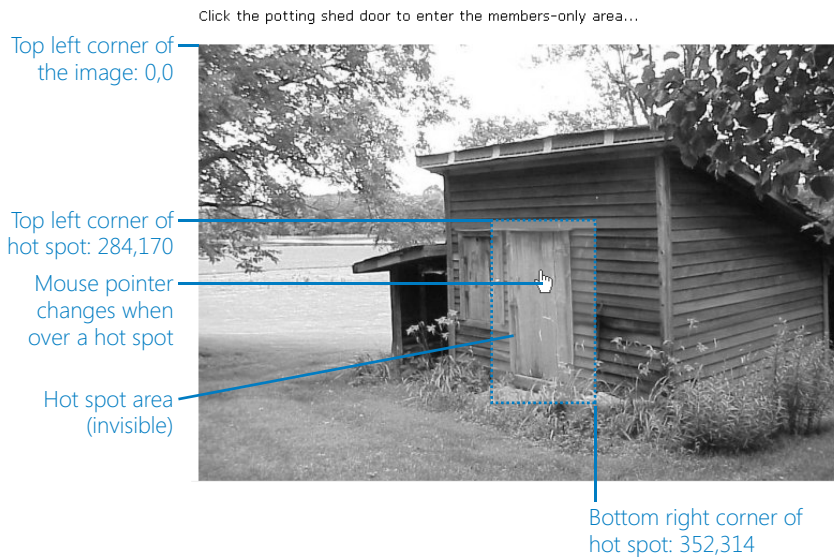
Creating an Image Map

You have seen how to make an image function as a hyperlink, but sometimes you might want different areas of the image to hyperlink to different locations. For example, suppose you have a map of the United States, and you want the user to be able to click individual states to view a page containing information specific to her location. To create such an effect, you must use an *image map*.

An image map is an overlay for a graphic that assigns hyperlinks to specifically defined areas (*hotspots*) on the image. The hotspots can be rectangular, circular, or irregularly shaped (called a *poly hotspot*).

The position of a rectangular hotspot is defined by two points: its upper-left and lower-right corners. Each point is expressed as a pair of numbers that represent the horizontal and vertical distance (in pixels) from the upper-left corner of the image. For example, in the following image, the shed door is defined as a hotspot. The upper-left corner of the shed door is located at 284,170—in other words, 284 pixels to the right and 170 pixels down from the upper-left corner of the image. The lower-right corner of the shed door is at 352,314. Therefore, the code for defining this particular hotspot is as follows:

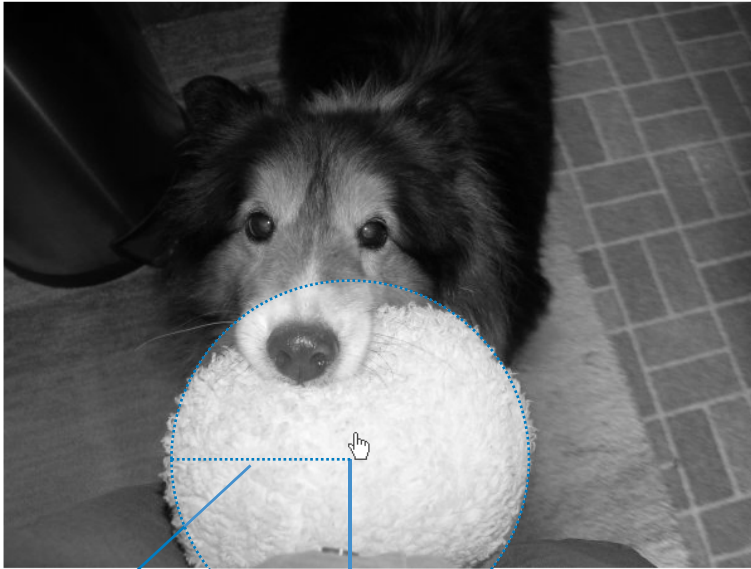
```
<area shape="rect" coords="284,170,352,314" href="enter.htm">
```



To define a circular hotspot, you use three coordinates: two for the circle's center point (horizontal and vertical values), and one for the radius of the circle.

```
<area shape="circle" coords="270,364,144" href="index.htm">
```

Click the ball to enter the Dog Toy Zone...



Radius of the circle is 270 pixels

The center of the circle is 270,364

To define a poly hotspot, you use as many coordinates as are needed to define all the vertexes of the shape. Poly hotspots consist of straight lines that connect each of the points you define. For example, here's one with four vertices:

```
<area shape="poly" coords="287,71,413,286,314,446,188,267" href="index.htm">
```

Click the moth to see the Pesticides area...

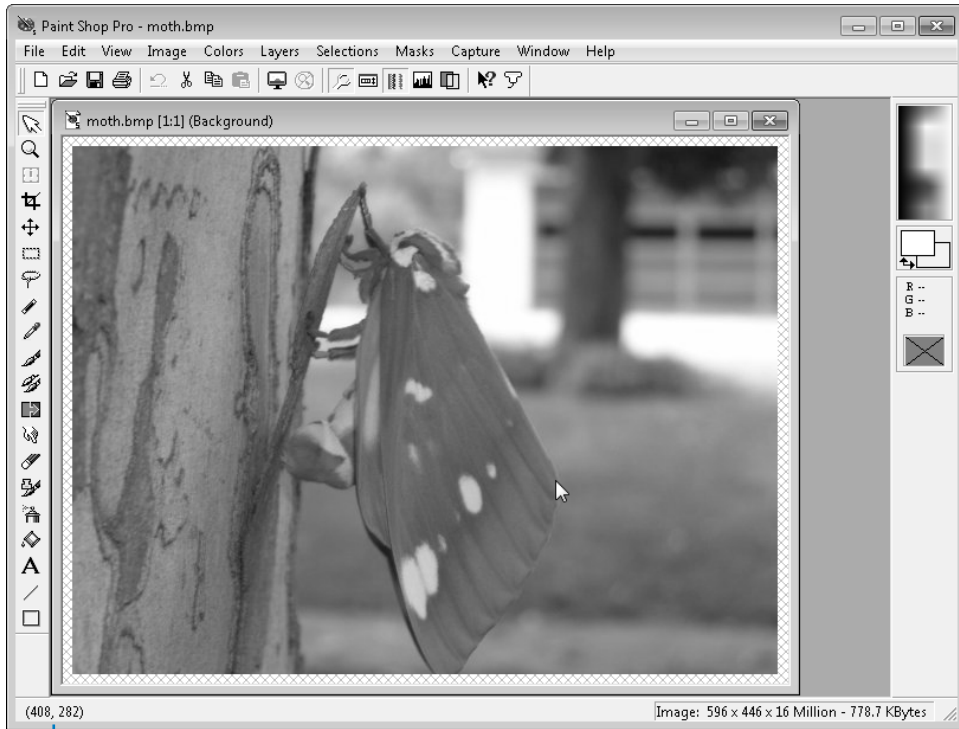


188,267

314,446

413,286

The easiest way to determine the coordinates of various points is to open the image in a graphics-editing program such as Paint Shop Pro, which displays the mouse pointer position in the status bar as you work. Move the mouse pointer over any spot on the image, and the program will display its coordinates.



Mouse pointer position

If you don't have access to a graphics-editing program, you can use trial-and-error to position the points.

To construct an image map, start with a two-sided `<map>` tag. In it, place *name* and *id* attributes. The name and ID can be the same; you need the *name* for the map itself, and you can use the *id* to refer to the image map in the style sheet, if desired.

```
<map name="moth" id="moth">
</map>
```

Then within the `<map>` tag, insert the points for the areas:

```
<map name="moth" id="moth">
<area shape="poly" coords="287,71,413,286,314,446,188,267" href="index.htm">
</map>
```

Just as with hyperlinks, you can include a *title* attribute in an `<area>` tag to make a ScreenTip appear when a user hovers the mouse over it. This is especially helpful when there is no text in the area, such as on a map or a photo.

```
<map name="moth" id="moth">
<area shape="poly" coords="287,71,413,286,314,446,188,267" href="index.htm"
title="Home page">
</map>
```

Finally, reference the map's name in the `` tag for the image with the *usemap* attribute. You must include a pound or hash sign (#) before the map name, as shown in the following:

```

```

In this exercise, you will create an image map that uses one graphic as a navigation bar with multiple hyperlinks.



SET UP Be sure to use the files provided specifically for this exercise, and not earlier versions. Use the *index.htm* file in the practice folder for this topic. This file is located in the Documents\Microsoft Press\HTML5 SBS\10Navigation\CreatingImageMap folder. Open the *index* file in Notepad and in Internet Explorer.

1. Immediately after the `` tag that contains the *bar.jpg* graphic, add an image map definition.

```
<nav>

<map>
</map>
</nav>
```

2. Name the map **navbar**, and then set its ID to **navbar**.

```
<nav>

<map name="navbar" id="navbar">
</map>
</nav>
```

3. Within the `<map>` tag, create the following hotspots:

```
<nav>

<map name="navbar" id="navbar">
<area shape="rect" coords="0,0,60,30" href="home.htm">
<area shape="rect" coords="70,0,155,30" href="tips.htm">
<area shape="rect" coords="165,0,250,30" href="problem.htm">
```

```
<area shape="rect" coords="260,0,325,30" href="products.htm">
<area shape="rect" coords="335,0,400,30" href="about.htm">
<area shape="rect" coords="410,0,490,30" href="contact.htm">
</map>
</nav>
```

4. In the `` tag, reference the name of the image map.

```

```

5. Save the file, and then refresh Internet Explorer. Position the mouse pointer over each name in the navigation bar. Notice that the URL displays in the browser's status bar.



Note Depending on your screen resolution and browser window size, the entire URL might not be visible in the status bar. After you publish the site to a Web server, however, the URL shown in the status bar will be much shorter and easier to read.

6. Edit each hyperlink to display a ScreenTip when the mouse pointer is positioned over it.

```
<nav>

<map name="navbar" id="navbar">
<area shape="rect" coords="0,0,60,30" href="home.htm" title="Home">
<area shape="rect" coords="70,0,155,30" href="tips.htm" title="Tips
& Tricks">
```

```

<area shape="rect" coords="165,0,250,30" href="problem.htm" title="Fix
Problems">
<area shape="rect" coords="260,0,325,30" href="products.htm" title=
"Products">
<area shape="rect" coords="335,0,400,30" href="about.htm" title= "About Us">
<area shape="rect" coords="410,0,490,30" href="contact.htm" title=
"Contact Us">
</map>
</nav>

```

Note Even though ScreenTips simply display the text that the user is clicking, they are still useful because they indicate that the text is clickable.

Notice the *&* used in the second hotspot definition. Remember that HTML uses the ampersand as a special character, so to display an ampersand, you must use *&*; so that it will render as an ordinary symbol.

7. Save the file, and then refresh Internet Explorer. Position the mouse pointer over each name in the navigation bar to display the ScreenTips.



✘ CLEAN UP Close the Notepad and Internet Explorer windows.

Redirecting to Another URL

After you have managed your own Web site for a while, you might decide you want to restructure its organization by renaming some pages, placing pages in folders, or hosting your site at a different location with a different URL. All that is fine, but what about the people who bookmarked the original page? They'll be faced with an unfriendly *Page Not Found* message if you remove the old content entirely, and they won't have any way of finding the page in its new location.

To help your past visitors find the new page, you can leave the old page in place and replace its text with a hyperlink that tells them where the new page is located. You already know how to create a hyperlink—that's simple. But you can take it one step further and set up the old page to actually *redirect* to the new page. In other words, you can make the old page automatically display the new page.

It is customary for a redirection to include five seconds of delay, so users can cancel the redirect operation if desired. It is also customary to include a text hyperlink to the new page, in case the redirect operation fails for some reason (such as the browser not supporting it, although this is uncommon).

You implement a redirect operation by adding an attribute to a `<meta>` tag in the `<head>` section of the page (as you learned in Chapter 2, "Setting Up the Document Structure"). You must create a new `<meta>` tag for this operation; you cannot add the attributes to any existing `<meta>` tag that the document might have. For example, to redirect to the page `support.microsoft.com` after a five-second delay, use the following:

```
<meta http-equiv="refresh" content="5; url=http://support.microsoft.com">
```

Be sure to use a semicolon (not a comma) between the delay (the `content` attribute) and the `url` attribute.

In this exercise, you will redirect one page to another page automatically after five seconds.



SET UP Be sure to use the files provided specifically for this exercise, and not earlier versions. Use the `foliage.htm` file in the practice folder for this topic. This file is located in the Documents\Microsoft Press\HTML5 SBS\10Navigation\Redirecting folder. Open the `foliage` file in Notepad and Internet Explorer.

1. In the `<head>` section, add a new `<meta>` tag as follows:

```
<meta http-equiv="refresh" content="5; url=foliage-new.htm">
```

2. In the `<body>` section, make the text `click here` into a hyperlink to `foliage-new.htm`.

```
<p>This page has been moved. <br>
If your browser supports automatic redirection, the new page will appear in
5 seconds. <br>
If the new page does not appear, <a href="foliage-new.htm">click here
</a>.</p>
```

3. Save the file, and then refresh Internet Explorer.



After five seconds, the *foliage-new* page appears.

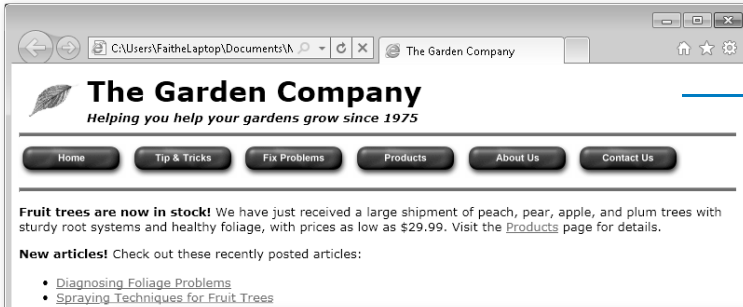
4. Click the browser's **Back** button, and then quickly click the **Click here** hyperlink to test it.

✕ CLEAN UP Close the Notepad and Internet Explorer windows.

Key Points

- A navigation bar contains a list of hyperlinks to the major pages on your site. It need not include every page in the site. The optimal number of links is between four and seven.
- In HTML5, you can use the `<nav>` tag as a container to indicate that a group of links constitutes a navigation element.
- Plan your site's organization before you create the navigation bar. Draw a diagram of all the pages and their connections to one another, and choose a file name for each page.
- Navigation bars are traditionally placed at the top or left side of a page. Placing a bar to the side requires the use of layout techniques discussed later in this book.
- Many Web designers place a text version of their navigation bar at the bottom of each page for user convenience.
- A text-based navigation bar is simply a series of hyperlinks.
- A graphical navigation bar uses small graphics for the hyperlinks. You can create these graphics using a graphics program such as Photoshop or a utility designed specifically for creating Web buttons.
- To redirect a page to a different URL, create a `<meta>` tag in the `<head>` section with the `http-equiv` attribute, like this: `<meta http-equiv="refresh" content="5; url=http://support.microsoft.com">` .
- You use an image map to specify individual sections of a single graphic that should act as hyperlinks. Use the `<map>` tag to create the map. Within it, define hotspots with `<area>` tags, and then reference the map as an attribute in the `` tag.
- To create image areas for your image maps, remember that points in an image are defined by their distances from the upper-left corner of the graphic. For example, the coordinates 10,15 refer to a point on the graphic that is 10 pixels to the right and 15 pixels below the upper-left corner.

Chapter at a Glance

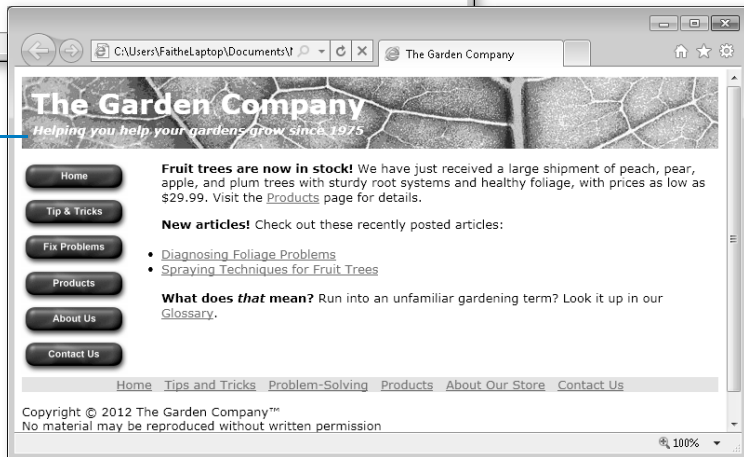


Create divisions,
page 188



Position divisions,
page 192

Format divisions,
page 197



11 Creating Division-Based Layouts

In this chapter, you will learn how to

- ✓ Understand HTML5 semantic tags.
 - ✓ Begin to think in divisions.
 - ✓ Create divisions.
 - ✓ Create an HTML5 semantic layout.
 - ✓ Position divisions.
 - ✓ Format divisions.
-

Until a few years ago, tables were the most popular way of structuring a Web page. You'll learn about tables and their formatting in Chapter 12, "Creating Tables," and Chapter 13, "Formatting Tables," in case that's the route you want to go with your site's design. However, as Web designers move increasingly toward separating style and content, division-based layouts are becoming more appealing.

A *division-based layout* defines the area of a page with `<div>` tags, or some of the new HTML5 semantic tags such as `<article>` and `<aside>`, and then applies formatting to each area using styles. One big advantage of division-based layouts is that you can place the styles in an external style sheet, and then make style changes to many pages at once simply by modifying the style sheet. For example, moving the navigation bar from the left to the right on a dozen pages is easy with a division-based layout that uses an external style sheet, but it's a huge chore with a table-based layout. Another advantage is that division-based layouts reduce the number of lines of code needed to produce a page.

In this chapter, you will learn how to create a separate area of a page (a *division*) in a document, and how to control division and element positions. Then you'll learn how to format a division (which is mostly a matter of applying the same formatting styles that you've learned about in previous chapters) and how to overcome any problems introduced by the formatting.

See Also Do you need only a quick refresher on the topics in this chapter? See the Key Points at the end of this chapter.

Practice Files Before you can use the practice files provided for this chapter, you need to install them from the book's companion content page to their default locations. See "Using the Practice Files" in the beginning of this book for more information.

Understanding HTML5 Semantic Tags

HTML5 adds some *semantic tags* to define layouts in more intuitive ways than the generic `<div>` tag is capable of. A semantic tag is one in which the name of a tag reflects its purpose.

Here are the major semantic tags you should know:

- **<header>** Defines the masthead or other header information on the page. Typically the header is repeated on every page of a site, although that is not required.
- **<footer>** Defines the text at the bottom of a page, such as the copyright or contact information. Again, it is typically repeated on every page of the site.
- **<article>** Defines a block of text that represents a single article, story, or message. An article can be distinguished from other text in that it can logically stand alone. For example, on a news site, each news story is an article.
- **<aside>** Defines a block of text that is tangential to the main discussion, such as a note, tip, or caution. An aside can be distinguished from other text in that it could be pulled out and discarded without disrupting the main document in which it appears.
- **<section>** Defines a generic content or application section. Examples of sections would be book chapters or the numbered sections of a thesis; a site's home page could be split into sections such as Introduction, News, and Contact Information. A section begins with a heading such as `<h1>` followed by other content. A general rule is to use `<section>` if the area being defined would be included in an outline of the document or page.

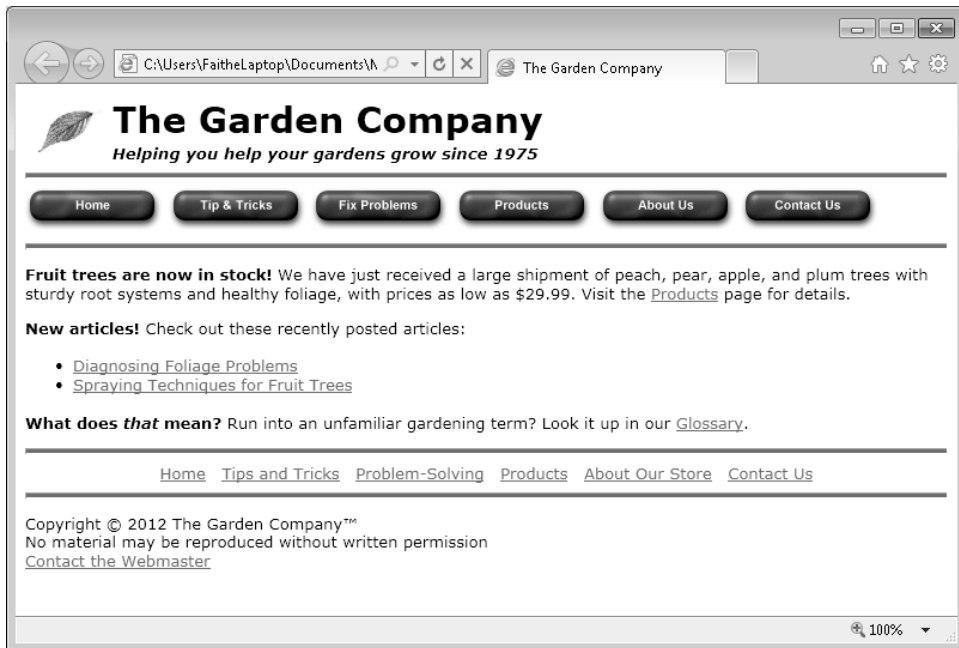
Note The `<section>` tag might sound similar to the `<div>` tag, but the HTML5 standard differentiates them, saying that `<section>` should not be used merely to define formatting. A section defines a particular type of meaningful content, not just a block of contiguous text that should be formatted the same way.

If you use semantic tags to structure your page and someone views it with a browser that doesn't support HTML5, the page might not look the way you want it to; the browser will ignore the tags it doesn't understand. That's why, for the time being, creating the page structure using `<div>` tags is the safest way to go. However, it's important that you learn the HTML5 semantic tags too, for future reference.

In this chapter, you'll learn to mark up a document both ways: with generic `<div>` tags that are readable in any browser, and with the new HTML5 semantic tags.

Beginning to Think in Divisions

In an effective division-based layout, each part of the page you want to format separately should be a *division*. For now, don't think about whether the division will be a vertical or horizontal area on the page, or how large or small it will be; just think about the content. For example, look at the following Web page. How many natural divisions do you see here?



If you were designing with `<div>` tags, you might break down this page like this: the masthead, the top navigation bar, the body text, the bottom navigation bar, and the copyright notice.

If you were designing with HTML5 semantic tags, you might break it down like this: `<header>` for the masthead, `<nav>` for the navigation bars, and `<footer>` for the copyright notice. Formatting each of the paragraphs in the body with its own `<article>` tag might be overkill for this page, but in a page with more content, you might use `<article>`, `<aside>`, or `<section>` to break content down into manageable pieces.

Creating Divisions

You use an *id* attribute to give a name to a division, like this:

```
<div id="masthead">
```

Each ID must be unique within the document, but multiple documents can use the same division names. Such reuse is good, in fact, because it lets you define the formatting of multiple documents with a single style sheet.

In this exercise, you will create divisions within a page. Then in later exercises, you will position and format those divisions.



SET UP Use the *index.htm* file in the practice file folder for this topic. This file is located in the Documents\Microsoft Press\HTML5 SBS\11Divisions\CreatingDivisions folder. Open the *index* file in Microsoft Notepad and Microsoft Internet Explorer.

1. Enclose the logo, company name, and tagline in a `<div>` tag, and name the tag **masthead**.

```
<body>
<div id="masthead">
<a href="http://www.contoso.com" title="Home page">
</a>
<h1 class="pagetitle">The Garden Company</h1>
<h5 class="tagline"><i>Helping you help your gardens grow since 1975</i></h5>
</div>
```

2. Enclose the top navigation bar in a `<div>` tag, and name the tag **topnav**.

```
<div id="topnav">
<hr>
<a href="index.htm"></a>
<a href="tips.htm"></a>
<a href="problems.htm"></a>
<a href="products.htm"></a>
<a href="about.htm"></a>
<a href="contact.htm"></a>
<hr>
</div>
```

Note Make sure that you include the `<hr>` tags in the `topnav` division.

Note As you learned in Chapter 10, “Creating Navigational Aids,” the `<nav>` tag is an HTML5 semantic tag that serves the same purpose as defining a `<div>` tag, but it is intended for a navigation bar. You’ll use `<nav>` in the next exercise in the chapter, where you apply HTML5 semantic tags.

3. Enclose the body paragraphs in a `<div>` tag, and name the tag **main**.

```
<div id="main">
<p><b>Fruit trees are now in stock! </b>We have just received a large
shipment of peach, pear, apple, and plum trees with sturdy root systems
and healthy foliage, with prices as low as $29.99. Visit the <a href=
"products.htm">Products</a> page for details.</p>
<p><b>New articles!</b> Check out these recently posted articles:
<ul>
<li><a href="foliage.htm">Diagnosing Foliage Problems</a></li>
<li><a href="spray.htm">Spraying Techniques for Fruit Trees</a></li>
</ul>
<p><b>What does <i>that</i> mean?</b> Run into an unfamiliar gardening term?
Look it up in our <a href="glossary.htm" target="_blank">Glossary</a>.</p>
</div>
```

4. Enclose the bottom navigation bar in a `<div>` tag, and name the tag **bottomnav**.

```
<div id="bottomnav">
<hr>
<p style="margin:0px; text-align: center">
<a href="index.htm">Home</a> &nbsp;
<a href="tips.htm">Tips and Tricks</a> &nbsp;
<a href="problems.htm">Problem-Solving</a> &nbsp;
<a href="products.htm">Products</a> &nbsp;
<a href="about.htm">About Our Store</a> &nbsp;
<a href="contact.htm">Contact Us</a></p>
<hr>
</div>
```

5. Enclose the copyright notice in a `<div>` tag, and name the tag **copy**.

```
<div id="copy">
<p>Copyright &copy; 2012 The Garden Company&trade;<br>
No material may be reproduced without written permission<br>
<a href="mailto:webmaster@contoso.com?subject=Question/Comment" title=
"webmaster@contoso.com">Contact the Webmaster</a></p>
</div>
```

6. Save the file.

Note You do not need to view your work in Internet Explorer this time because you have not made any changes that change the rendering or appearance of the page. You will do that later in the chapter.

 **CLEAN UP** Close the Notepad and Internet Explorer windows.

Creating an HTML5 Semantic Layout

If you prefer to use the HTML5 semantic tags to create your layout, you choose the appropriate tags based on the *purpose* of the text. It's conceptually very much the same as using a `<div>` tag with an *id* attribute, but the tag itself provides the context. For example, instead of the `<div id="masthead">` tag, you would use the `<header>` tag.

In this exercise, you will change a division-based document to one that uses semantic tags to define the layout.



SET UP Use the *index2.htm* file in the practice file folder for this topic. This file is located in the Documents\Microsoft Press\HTML5 SBS\11Divisions\UsingSemantic folder. Open the *index2* file in Microsoft Notepad and Microsoft Internet Explorer.

1. Replace the `<div id="masthead">` tag with `<header>`, and change its closing `</div>` tag to `</header>`.

```
<body>
<header>
<a href="http://www.contoso.com" title="Home page">
</a>
<h1 class="pagetitle">The Garden Company</h1>
<h5 class="tagline"><i>Helping you help your gardens grow since
1975</i></h5>
</header>
```

2. Replace the `<div id="topnav">` tag with `<nav>`, and change its closing `</div>` tag to `</nav>`.

```
<nav>
<hr>
<a href="index.htm"></a>
<a href="tips.htm"></a>
<a href="problems.htm"></a>
<a href="products.htm"></a>
<a href="about.htm"></a>
<a href="contact.htm"></a>
<hr>
</nav>
```

Note Because the bottom navigation bar should be formatted differently than the top one, leave it formatted as a division. That way you can use the `<nav>` tag to define the formatting for only the top navigation bar.

3. Delete the `<div id="main">` tag and its closing `</div>` tag.
4. Enclose the first paragraph of the body text with an `<article>` tag.

```
<article>
<p><b>Fruit trees are now in stock! </b>We have just received a large
shipment of peach, pear, apple, and plum trees with sturdy root systems
and healthy foliage, with prices as low as $29.99. Visit the <a href=
"products.htm">Products</a> page for details.</p>
</article>
```

Note In practical usage, the individual paragraphs of body text on this page would probably not warrant their own semantic tags because this page contains so little content overall. However, for example purposes, you will mark them up anyway.

5. Enclose the second paragraph and the bulleted list that follows it with an `<article>` tag.

```
<article>
<p><b>New articles!</b> Check out these recently posted articles:
<ul>
<li><a href="foliage.htm">Diagnosing Foliage Problems</a></li>
<li><a href="spray.htm">Spraying Techniques for Fruit Trees</a></li>
</ul></article>
```

6. Enclose the last body paragraph with an `<aside>` tag.

```
<aside>
<p><b>What does <i>that</i> mean?</b> Run into an unfamiliar gardening term?
Look it up in our <a href="glossary.htm" target="_blank">Glossary</a>.</p>
</aside>
```

Leave the bottom navigation bar's `<div>` tag as is.

7. Replace the `<div id="copy">` tag with `<footer>`, and change its closing `</div>` tag to `</footer>`.

```
<footer>
<p>Copyright ©; 2012 The Garden Company&trade;.<br>
No material may be reproduced without written permission.<br>
<a href="mailto:webmaster@contoso.com?subject=Question/Comment" title=
"webmaster@contoso.com">Contact the Webmaster</a></p>
</footer>
```

8. Save the file.

Note You do not need to view your work in Internet Explorer this time because the changes you have made do not change the rendering.

 **CLEAN UP** Close the Notepad and Internet Explorer windows.

Positioning Divisions

There are two ways of positioning a division (or equivalent semantic-tagged block): you can use the *float* style rule, as you did with pictures in Chapter 9, “Displaying Graphics”, or you can use the *position* style rule. The following sections explain each of these methods.

Note In the rest of this chapter, for simplicity, I use the term *division* generically to mean both the `<div>` tag and the HTML5 semantic tags. In most cases, browsers handle the formatting and positioning the same way.

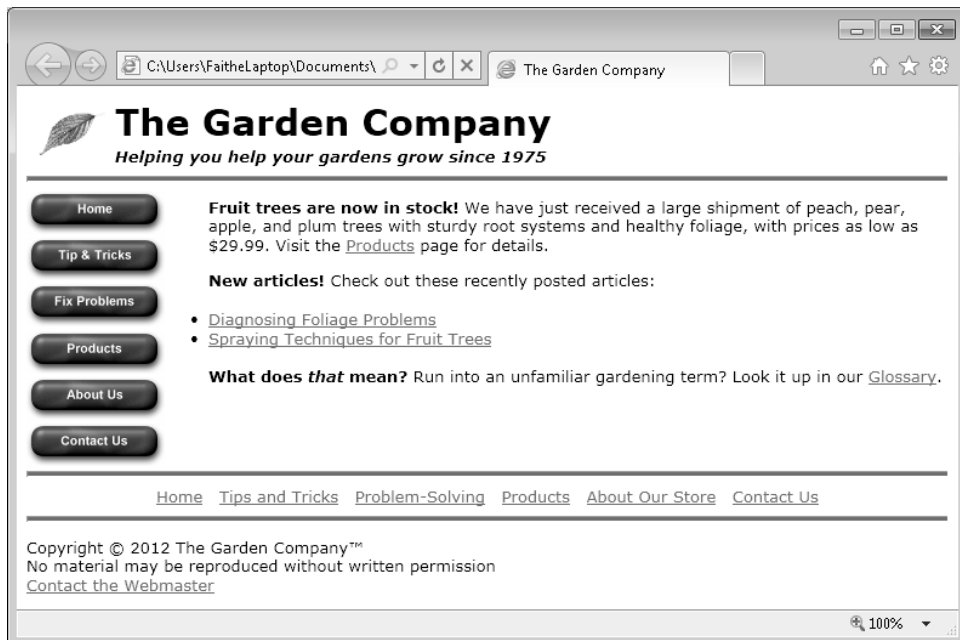
Floating a Division to the Right or Left

The easiest way to place one division beside another is to use the *float* style rule. For example, to make a navigation bar that floats to the left of the main body text, you can set the navigation bar’s division to a certain width (perhaps 150 pixels or so), and then float it like this:

```
<div id="topnav" style="width: 150px; float: left">
```

Alternatively, if you were using the `<nav>` tag for the navigation bar, it would look like this:

```
<nav style="width: 150px; float: left">
```



Because the main advantage of using divisions is to promote consistency across documents, you would probably want to set up the style rule in an external style sheet rather than in the individual division tag or an internal style sheet.

In a style sheet, you precede the names of unique elements such as divisions with a pound sign (#), as shown in the following:

```
#topnav {width: 150px; float: left}
```

Alternatively, if you were using the `<nav>` tag for the navigation bar, the style rule in the style sheet would look like this:

```
nav {width: 150px; float: left}
```

Positioning a Division on the Page

If you need a division to be in a specific spot on the page, use the *position* style rule, which has three possible values:

- *position: absolute* This value specifies a fixed position with respect to the parent element. Unless the element is within some other tag, the parent element is generally the `<body>` tag; in this case, the element would have a fixed position relative to the upper-left corner of the page.
- *position: relative* This value specifies an offset from the element's natural position. Other elements on the page are not affected, even if the new position causes elements to overlap.
- *position: fixed* This value specifies a fixed position within the browser window that doesn't change even when the display is scrolled up or down. Internet Explorer does not support this setting.

You must use each of these values in conjunction with a *top*, *right*, *bottom*, and/or *left* style rule that specifies the location to which the *position* rule refers. For example, to position a division called *main* exactly 100 pixels from the top of the page and 200 pixels from the left side, create this style rule in the style sheet:

```
#main {position: absolute; top: 100px; left: 200px}
```

Note When using semantic tags, you won't have one that defines the entire main body of the page content, so you might want to create a division for that purpose if you want to specify an exact position for all the body text on the page. As this example illustrates, it's okay to mix up semantic tags and `<div>` tags in your work. The `<div>` tag is not deprecated in HTML5; it's still perfectly valid.

You can combine positioning with a *width* specification to position each division in a precise rectangular area on the screen. For example, to place the top navigation bar exactly 100 pixels from the top of the page and make it 150 pixels wide, use the following:

```
#topnav {position: absolute; top: 100px; width: 150px}
```

Or, if you are using the `<nav>` tag instead, use this:

```
nav {position: absolute; top: 100px; width: 150px}
```

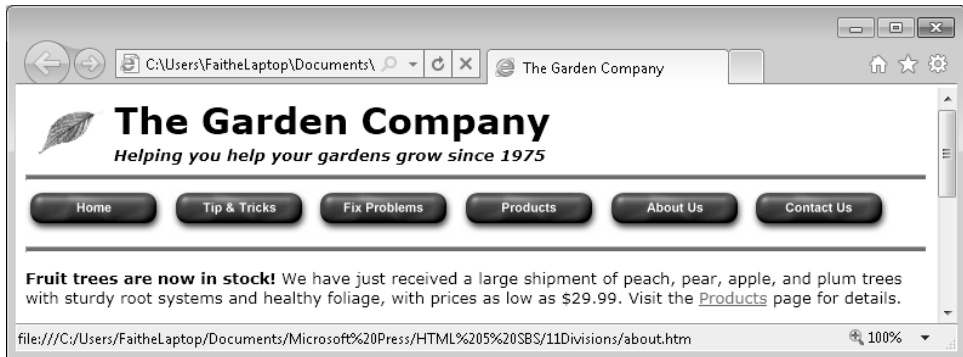
The *position* style rule results in positioning that does not take into regard other elements on the page. This can get you in trouble because elements can potentially overlap unattractively, but it can also be used to intentionally create overlapping elements. For example, you can use this feature to overlay text on a photo.

In this exercise, you will specify a size and position for several divisions by creating rules that refer to those divisions in an external style sheet. This example file uses a mixture of HTML5 semantic tags and generic `<div>` tags.



SET UP Use the *default.css* and *index3.htm* files in the practice file folder for this topic. These files are located in the Documents\Microsoft Press\HTML5 SBS\11Divisions\PositioningDivisions. Open the *default.css* style sheet in Notepad, and open the *index3.htm* file in Internet Explorer.

1. In Internet Explorer, view the *index3* file. Note the position of the top navigation bar.



2. In Notepad, in *default.css*, add the following style rule:

```
nav {float: left; width: 150px; padding-top: 15px}
```

Note You can add the style rule anywhere in *default.css*; adding it at the end of the file is fine.

3. Save the file, and then refresh Internet Explorer.

The navigation bar now appears at the left side of the page.

Note Notice that when the navigation bar is laid out vertically, the horizontal rule below it looks awkward.



4. Open `index3.htm` in Notepad and remove the `<hr>` tag immediately before the `</nav>` tag. Save your work, and then refresh Internet Explorer to view the change.
5. Reopen `default.css` in Notepad if necessary. Add a style rule that limits the width of the main division to 500 pixels.

```
#main {width: 500px}
```

6. Save the file, and then refresh Internet Explorer.
Notice that the body text begins higher on the page than the top button, which looks a bit awkward. We'll fix that next.
7. Specify an absolute position for the top of the main division that is 85 pixels from the top and 140 pixels from the left

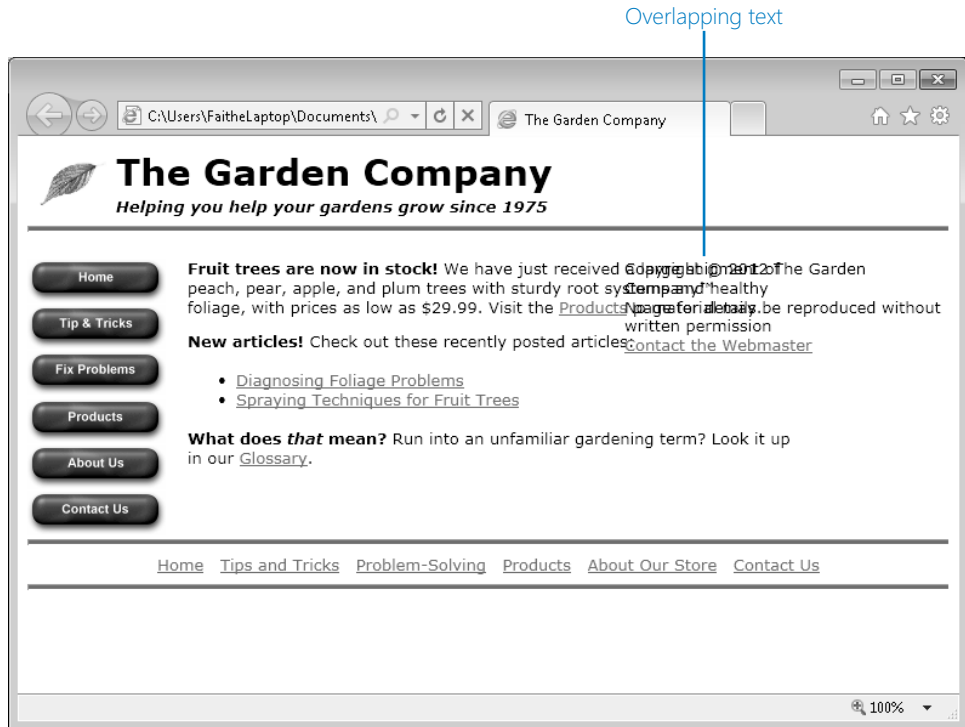
```
#main {width: 500px; position: absolute; top: 85px; left: 140px}
```

8. Save the file, and then refresh Internet Explorer.
Notice that the text in the main division now aligns nicely with the top of the buttons.

9. (Optional) Experiment with the top and left settings in *default.css*, saving your work and refreshing Internet Explorer to see the changes. Return the style rule to **top: 85px** and **left: 140px** when you are finished experimenting.
10. Position the upper-left corner of the `<footer>` section 85 pixels from the top and 500 pixels from the left side of the page.

```
footer {position: absolute; top: 85px; left: 500px}
```

11. Save the file, and then refresh Internet Explorer.
Notice that the main and copy divisions overlap.

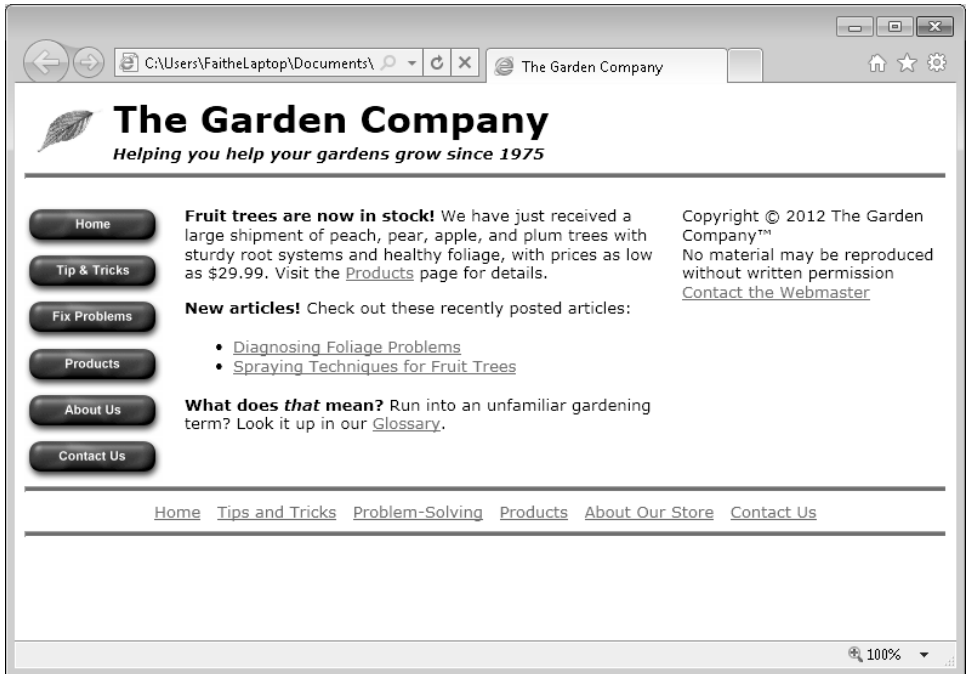


12. Modify the style rules for the main division and the footer tag so that the main division is only 400 pixels wide, and the footer starts at 550 pixels from the left:

```
#main {width: 400px; position: absolute; top: 85px; left: 140px}
```

```
footer {position: absolute; top: 85px; left: 550px}
```

13. Save the file, and then refresh Internet Explorer.
Now the divisions share the horizontal space more attractively.



✕ CLEAN UP Close the Notepad and Internet Explorer windows.

Formatting Divisions

You format divisions as you would any other elements. You can use styles to specify the font family, font style, font weight, alignment, color, and everything else covered so far in this book.

You can change the background color of a division with the *background-color* style rule. For example, to add a khaki-colored background to the navigation bar, use the following:

```
nav {float: left; width: 150px; padding-top: 15px; background-color: khaki}
```

When you start applying colors to divisions, however, you might uncover some underlying problems with your page. For example, the page for The Garden Company from the previous example looks pretty good when everything has a white background, but watch what happens when you add that khaki background to the navigation bar, as shown in the image that follows.



There are several problems with this layout. One is that the *main* division, which has an absolute position, is overlapping the navigation bar. The root cause is that the navigation bar is wider than it needs to be. Also, the button graphics in the navigation bar have a rectangular white background—a fact that was not obvious until now.

You can fix the size and positioning issues easily enough by modifying the styles. For example, you could decrease the width of the navigation bar to 100 pixels, as shown in the following:

```
nav {float: left; width: 100px; padding-top: 15px; background-color: khaki}
```

Unfortunately, you can't fix the button background problem with HTML; you'd need to edit the button graphics in a program that supports transparency, setting each button's background to be transparent. If your graphics-editing program does not support transparency, one solution is to change each button's background color to khaki. That method is not as good, though, because you might decide to make the navigation bar some other color later. With a transparent background, the buttons will blend nicely into any background color.

Note Recall from Chapter 9 that GIF and PNG graphics formats support transparency, but JPG does not.

In this exercise, you will apply a colored background to a division and edit that division's formatting to fine-tune it.



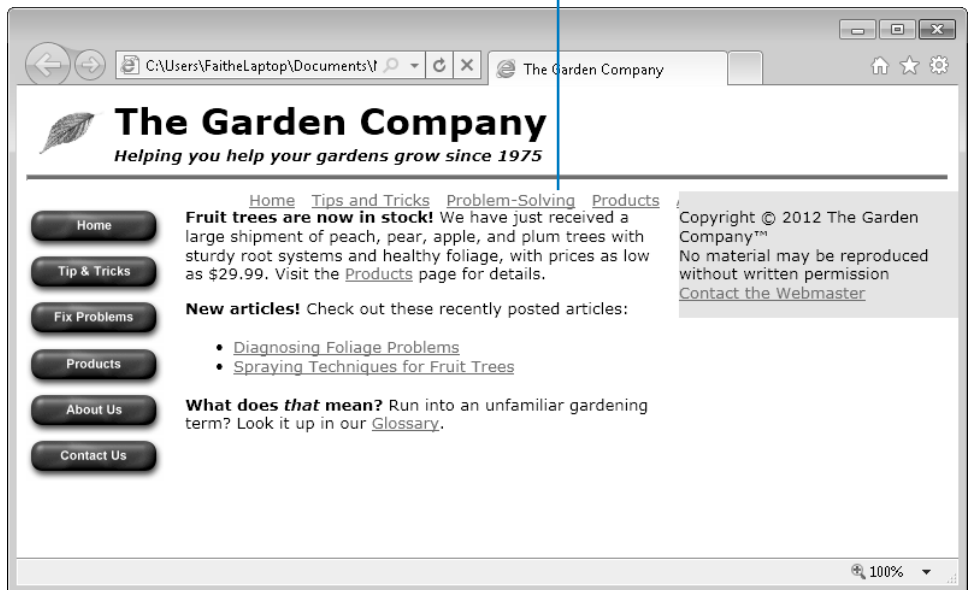
SET UP Use the *default.css* and *index3.htm* files in the practice file folder for this topic. These files are located in the Documents\Microsoft Press\HTML5 SBS\11Divisions\FormattingDivisions folder. Open the *default.css* style sheet in Notepad, and open the *index3.htm* file in Internet Explorer.

1. In the *default.css* style sheet, add the khaki background color to the footer.


```
footer {position: absolute; top: 85px; left: 550px; background-color: khaki}
```
2. Save the file, and then refresh Internet Explorer.
3. Open the *index3* file in Notepad and delete the `<hr>` tags from the *bottomnav* division.
4. Save the file, and then refresh Internet Explorer.

Oops! Look what has happened. Those horizontal lines were holding that division at the bottom of the page, where it belongs. Without them, the text shifted up. The browser ignores all the other divisions except the masthead because they are all absolutely positioned.

Bottom navigation bar has shifted up



At this point you have two choices: you can set an absolute position for the *bottomnav* division, or you can get rid of the absolute positioning for all the divisions and go back to a simple float for the top navigation bar. Let's do the latter.

5. In the *default.css* style sheet, delete the *main* division's style rule as well as the style rules for the *footer*.

6. Add a style rule that changes the bottom navigation bar to khaki.

```
#bottomnav {background-color: khaki}
```

7. Save the file, and then refresh Internet Explorer.

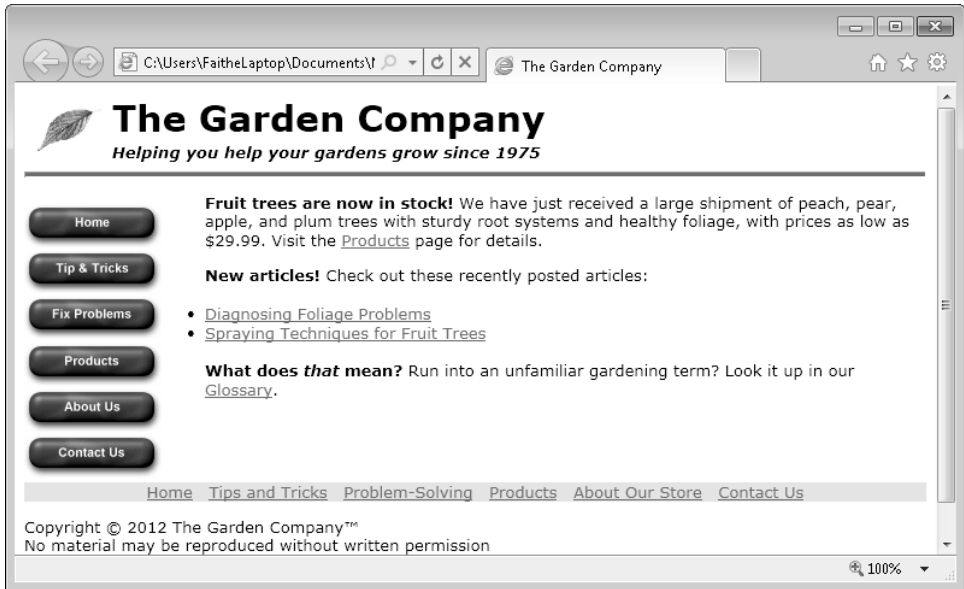
Now the bottom navigation bar is back down at the bottom of the page, but it doesn't clear the navigation bar at the left.



8. In the *default.css* style sheet, set the *bottomnav* division to clear any items positioned to its left.

```
#bottomnav {background-color: khaki; clear: left}
```

9. Save the file, and then refresh Internet Explorer.



Now give the site a new look by getting rid of the graphic and horizontal line in the masthead and inserting a background image in the *masthead* division.

10. In the *index3* file, in the header, delete the `` tag for the leaf and its associated `<a>` hyperlink. Delete the horizontal line, as well. This leaves the `<header>` looking like this:

```
<header>
<h1 class="pagetitle">The Garden Company</h1>
<h5 class="tagline"><i>Helping you help your gardens grow since 1975</i></h5>
</header>
```

Some browsers don't interpret `<header>` correctly, and the masthead is a fairly important page element to get right, so in the interest of compatibility, turn that `<header>` back into a division whose name is *header* before going any further.

11. Change the `<header>` tag to a `<div>` tag.

```
<div id="header">
<h1 class="pagetitle">The Garden Company</h1>
<h5 class="tagline"><i>Helping you help your gardens grow since 1975</i></h5>
</div>
```

12. Save the file.
13. In the *default.css* style sheet, add a style rule for the header division that applies an image as its background:

```
#header {background-image: url(images/leaf-green.jpg)}
```

14. Save the file, and then refresh Internet Explorer.

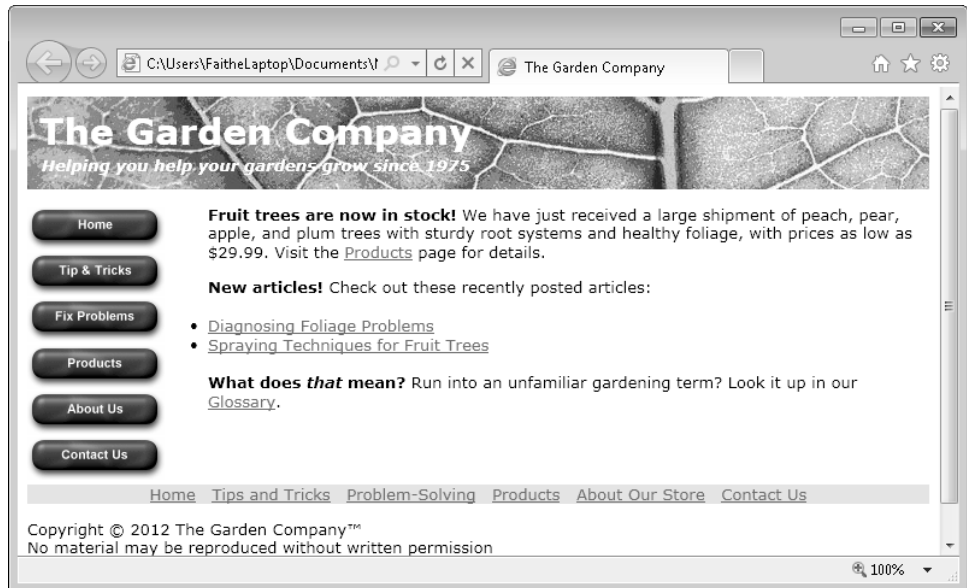


The new masthead looks interesting, but you need to add some padding and make the text easier to read.

15. Modify the style rule for the header division to use white text and to add 10 pixels of padding on all sides:

```
#header {background-image: url(images/leaf-green.jpg); padding: 10px; color: white}
```

16. Save the file, and then refresh Internet Explorer to see the new masthead.

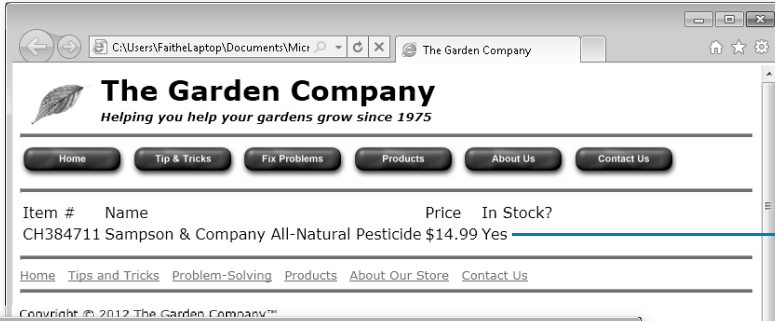


✖ CLEAN UP Close the Notepad and Internet Explorer windows.

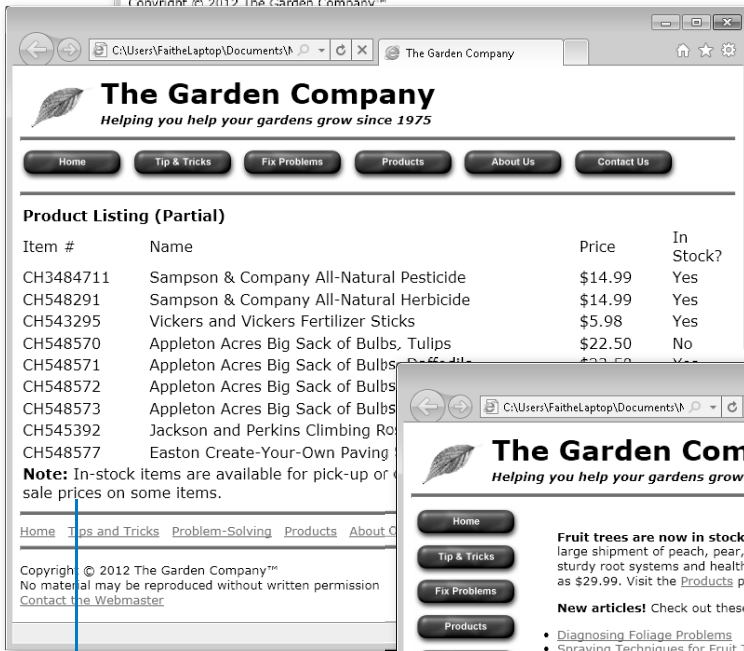
Key Points

- To create a division, surround a section of a page with a `<div>` tag.
- HTML5 uses semantic tags to define sections of a page. Some of the most common of these are `<header>`, `<footer>`, `<nav>`, `<article>`, `<aside>`, and `<section>`. Not all browsers support these tags yet. Internet Explorer 9.0 and higher does, as do the current versions of Google Chrome and Firefox.
- Each division tag has an `id` attribute that should be unique within that document. Multiple documents can have the same division names, though, and in fact, this is encouraged so that one external style sheet can format multiple documents.
- One way to position a division is with a `float` attribute. For example, to place a division at the left (for use as a navigation bar), use `float: left`.
- Another way to position a division is with a position attribute. The valid values are `absolute`, `relative`, and `fixed`. When you use the `position` attribute, you must also use a `top`, `bottom`, `left`, and/or `right` attribute to specify the numeric value for the position.
 - With absolute positioning, the element is positioned absolutely within its parent element, which is usually the `<body>` tag, so the element is positioned absolutely on the page.
 - With relative positioning, the element is positioned in relation to its default position.
 - With fixed positioning, the element is positioned in relation to the browser window.
- Divisions can be formatted by using the same character, paragraph, and page formatting styles you learned throughout the book, including `background-color` and `background-image`.

Chapter at a Glance

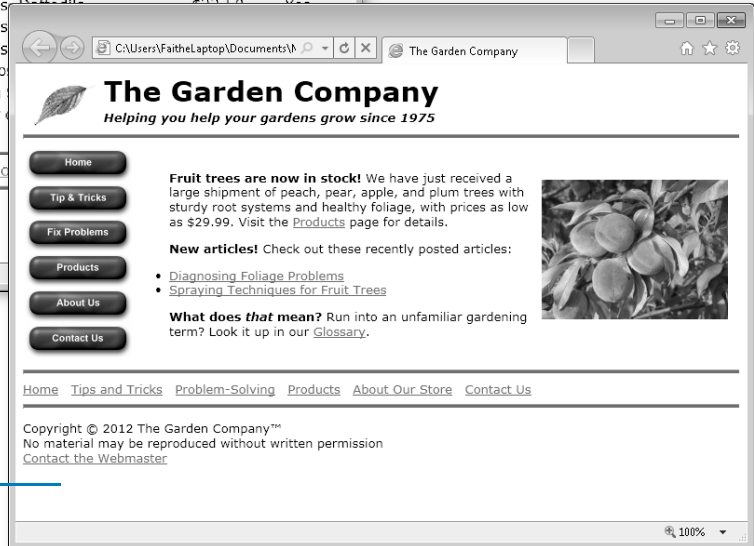


Create a table,
page 207



Merge table cells,
page 220

Use tables for
page layout,
page 224



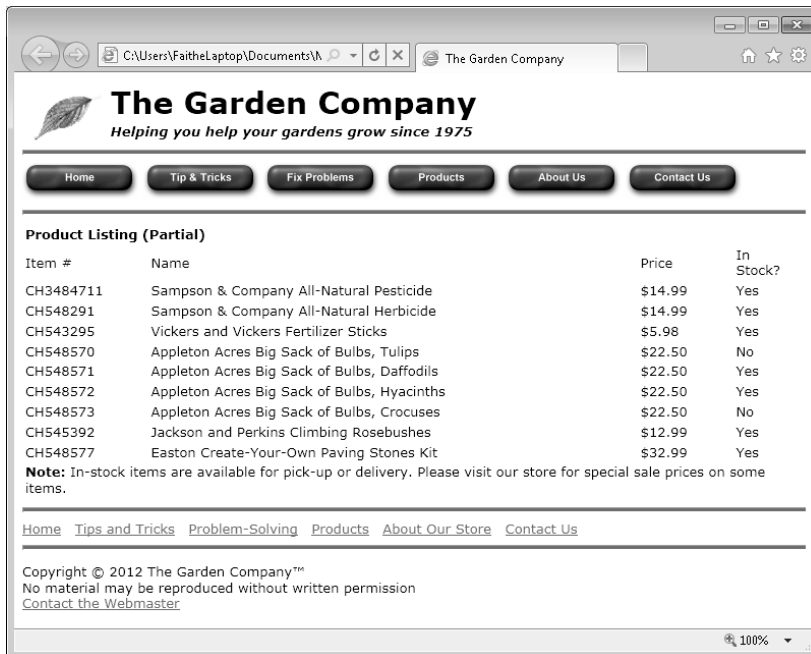
12 Creating Tables

In this chapter, you will learn how to

- ✓ Create a simple table.
- ✓ Specify the size of a table.
- ✓ Specify the width of a column.
- ✓ Merge table cells.
- ✓ Use tables for page layout.

If you've used a word-processing program before, you're probably already familiar with the task of creating tables. A *table* is a grid of rows and columns, the intersections of which form *cells*. Each cell is a distinct area, into which you can place text, graphics, or even other tables.

HTML handles tables very well, and you can use them to organize and present complex data to your site visitors. For example, you could display your store's inventory in a table.



The screenshot shows a web browser window with the address bar displaying 'C:\Users\Faithelaptop\Documents\A...' and the page title 'The Garden Company'. The page features a logo with a leaf and the text 'The Garden Company Helping you help your gardens grow since 1975'. Below the logo is a navigation menu with buttons for 'Home', 'Tip & Tricks', 'Fix Problems', 'Products', 'About Us', and 'Contact Us'. The main content area is titled 'Product Listing (Partial)' and contains a table with the following data:

Item #	Name	Price	In Stock?
CH3484711	Sampson & Company All-Natural Pesticide	\$14.99	Yes
CH548291	Sampson & Company All-Natural Herbicide	\$14.99	Yes
CH543295	Vickers and Vickers Fertilizer Sticks	\$5.98	Yes
CH548570	Appleton Acres Big Sack of Bulbs, Tulips	\$22.50	No
CH548571	Appleton Acres Big Sack of Bulbs, Daffodils	\$22.50	Yes
CH548572	Appleton Acres Big Sack of Bulbs, Hyacinths	\$22.50	Yes
CH548573	Appleton Acres Big Sack of Bulbs, Crocuses	\$22.50	No
CH545392	Jackson and Perkins Climbing Rosebushes	\$12.99	Yes
CH548577	Easton Create-Your-Own Paving Stones Kit	\$32.99	Yes

Note: In-stock items are available for pick-up or delivery. Please visit our store for special sale prices on some items.

At the bottom of the page, there is a footer with the text: 'Copyright © 2012 The Garden Company™ No material may be reproduced without written permission Contact the Webmaster'.

The most popular use of tables in HTML, however, is as a page-layout tool. You can create a large table that occupies the entire page, and then place content into the cells to position that content on the page. In the following figure, for example, each of the sections (the masthead, the navigation bar, the body, and the footer) resides in its own separate table cell. You will create this layout later in the chapter, learning the details of how it's done.



Note Now that division-based layouts are becoming more common (see Chapter 11, “Creating Division-Based Layouts”), some experts will tell you that table-based design is on its way out. However, for a small, non-professional Web site for personal or organizational use, tables are still a very viable way of laying out your pages.

In this chapter, you’ll learn the basic HTML for creating tables, rows, and cells. You’ll also learn how to specify cell sizes and merge cells to create larger areas. After you master these skills, you’ll put them to work by creating a table-based page layout grid. Then, in the next chapter, you’ll learn how to format tables.

See Also Do you need only a quick refresher on the topics in this chapter? See the Key Points at the end of this chapter.

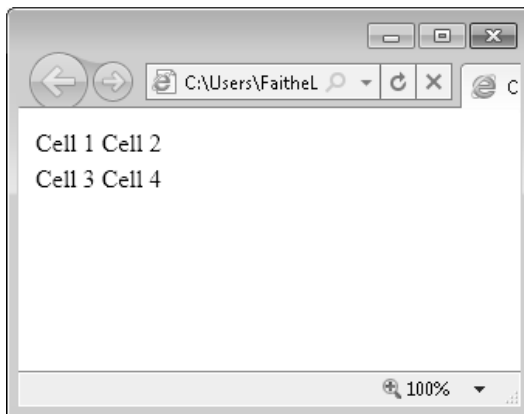
Practice Files Before you can use the practice files provided for this chapter, you need to install them from the book's companion content page to their default locations. See "Using the Practice Files" in the beginning of this book for more information.

Creating a Simple Table

The `<table>` tag creates an HTML table. Within that tag, you include one or more `<tr>` tags, which define the table's rows, and within each `<tr>` tag, you define one or more `<td>` tags, which define the cells.

```
<table>
  <tr>
    <td>Cell 1</td>
    <td>Cell 2</td>
  </tr>
  <tr>
    <td>Cell 3</td>
    <td>Cell 4</td>
  </tr>
</table>
```

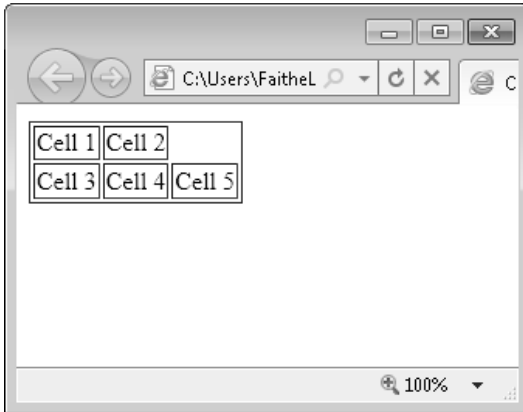
Displayed in a browser, the code just shown creates a table that looks like the following:



This table is not very interesting to look at in its default state. And because by default, HTML tables have no borders or shading, you can barely even tell it's a table at all. The text simply appears where it's supposed to appear according to the table's specification. (That's a big hint about how you will use tables for layout later in the chapter.)

The number of columns within a table is equal to the largest number of `<td>` tags in any given row. Watch what happens when I add another `<td>` tag to the second row. I'm also going to add a `border="1"` attribute in the `<table>` tag to make the table borders visible. so you can see what's going on more clearly. (You'll learn more about that attribute in Chapter 13, Formatting Tables.) The additions are shown in bold text in the following code:

```
<table border="1">
  <tr>
    <td>Cell 1</td>
    <td>Cell 2</td>
  </tr>
  <tr>
    <td>Cell 3</td>
    <td>Cell 4</td>
    <td>Cell 5</td>
  </tr>
</table>
```



Notice that because the rows do not have the same number of cells, the browser inserts a blank space in the row that doesn't include the extra cell. In the section "Merging Table Cells" on page 220, you will learn how to merge multiple cells into a single cell.

In this exercise, you will create a simple table.



SET UP Use the `products.htm` file in the practice file folder for this topic. This file is located in the `Documents\Microsoft Press\HTML5 SBS\12Tables\CreatingTable` folder. Open the `products` file in Microsoft Notepad and in Microsoft Internet Explorer.

1. In Notepad, in the empty space between the two consecutive `<hr>` tags, create the table, leaving a few blank lines between the opening and closing tags.

```
<table>

</table>
```

2. Within the table, create three rows. Indenting the lines as shown in the following code is optional but recommended.

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

3. Within the first row, create four columns.

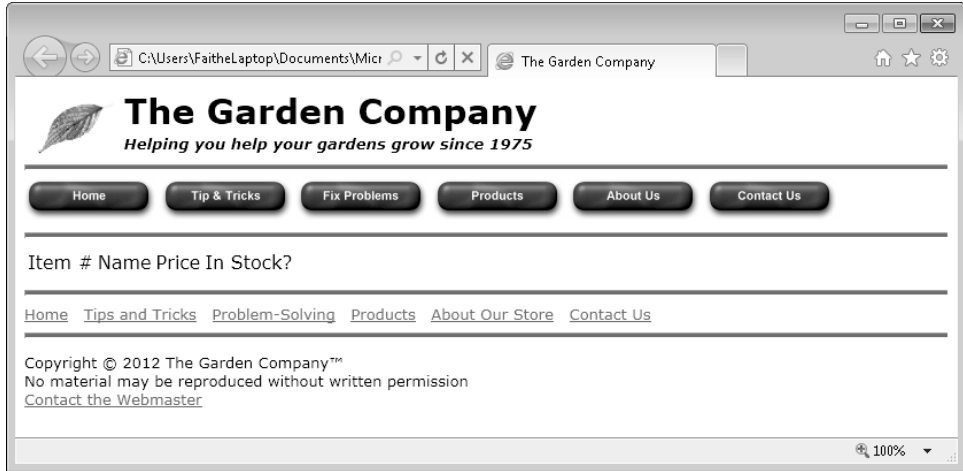
```
<table>
  <tr>
    <td> </td>
    <td> </td>
    <td> </td>
    <td> </td>
  </tr>
  <tr>
  </tr>
  <tr>
  </tr>
</table>
```

4. Enter the text that will appear in the first row of the table, as follows:

```
<table>
  <tr>
    <td>Item #</td>
    <td>Name</td>
    <td>Price</td>
    <td>In Stock?</td>
  </tr>
  <tr>
  </tr>
  <tr>
  </tr>
</table>
```

Tip If you like, you can use `<th>` tags instead of `<td>` tags to indicate table headings. Some browsers automatically format table heading cells differently.

5. Save the file, and then refresh Internet Explorer.



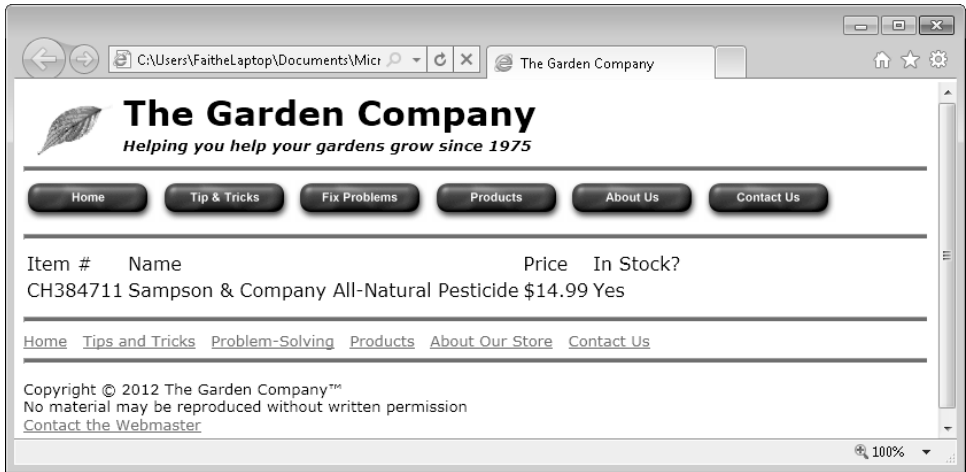
Notice that the browser ignores the two empty rows.

6. In Notepad, enter a product name in each cell of the first empty row, as shown here:

```
<table>
  <tr>
    <td>Item #</td>
    <td>Name</td>
    <td>Price</td>
    <td>In Stock?</td>
  </tr>
  <tr>
    <td>CH384711</td>
    <td>Sampson & Company All-Natural Pesticide</td>
    <td>$14.99</td>
    <td>Yes</td>
  </tr>
  <tr>
  </tr>
</table>
```

7. Save the file, and then refresh Internet Explorer.

Notice that the columns have expanded to accommodate the longest entries.



✕ CLEAN UP Close the Notepad and Internet Explorer windows.

Specifying the Size of a Table

By default, a table sizes itself to accommodate all of its cells, and each cell's height and width changes to accommodate the largest entry in that row or column. The table structure expands or contracts as needed when you add or remove cells or content within cells.

With these default settings, a table can end up looking rather cramped, especially if you don't use borders between cells (which you'll learn more about in Chapter 13). In the table from the previous exercise, for example, some extra space would be welcome.

One way to add extra spacing in this instance would be to set the overall size of the table to 100 percent. This forces the table to expand horizontally to fill the available space in the browser window. To do this, add a width attribute to the opening `<table>` tag like this:

```
<table width=100%>
```

Alternatively, you can place the width specification in a style, like this:

```
<table style="width: 100%">
```

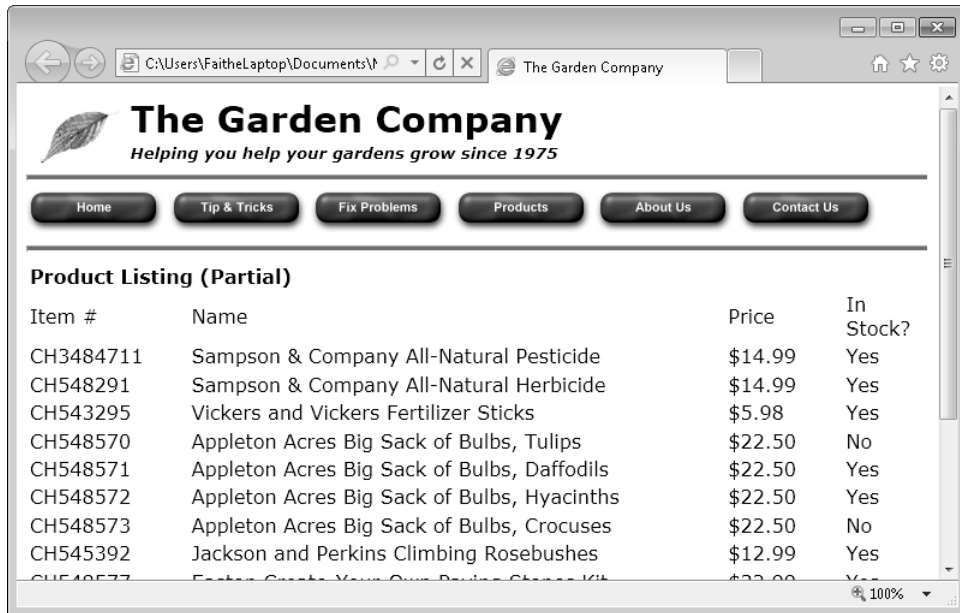
To apply the width specification to all tables, place it in a style sheet, as shown here:

```
table {width: 100%}
```

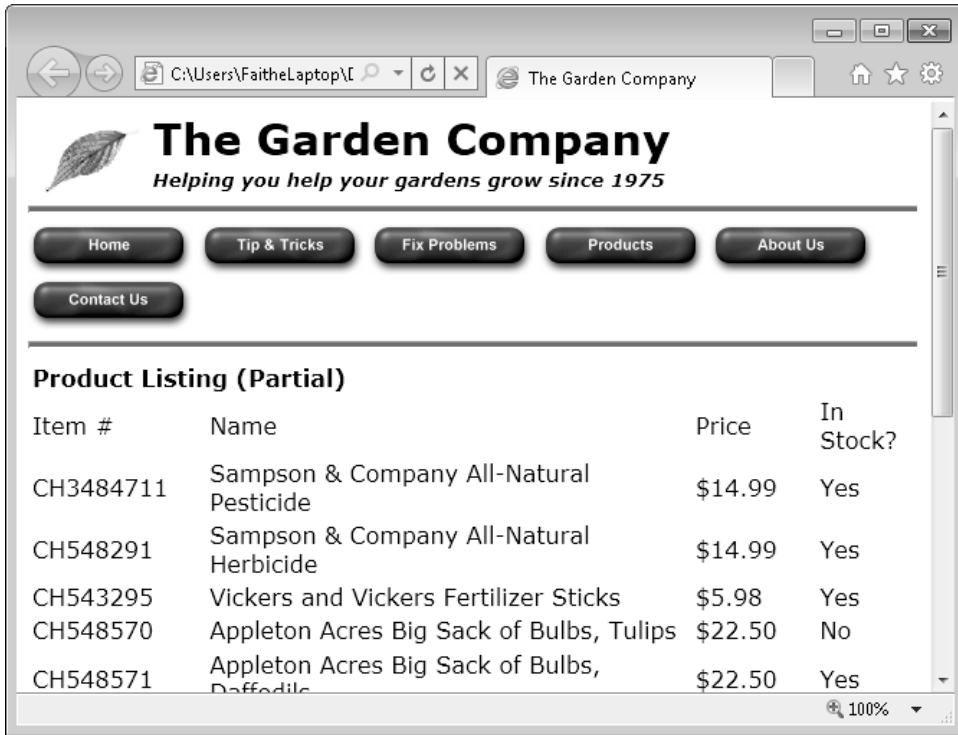
You don't need to specify 100 percent; you could also set the table's width to 50, 75, or any other percentage. You can do the same thing with table height, making it expand to fill the entire browser window vertically by using the following:

```
table (height: 100%)
```

The only drawback to specifying width and/or height by percentage is that you cannot be certain which size browser window the visitors to your site will be using. This example looks great in an 800 × 600 window, such as demonstrated here:



But in a smaller window, it becomes just as cramped as before, and the text wraps to multiple lines.



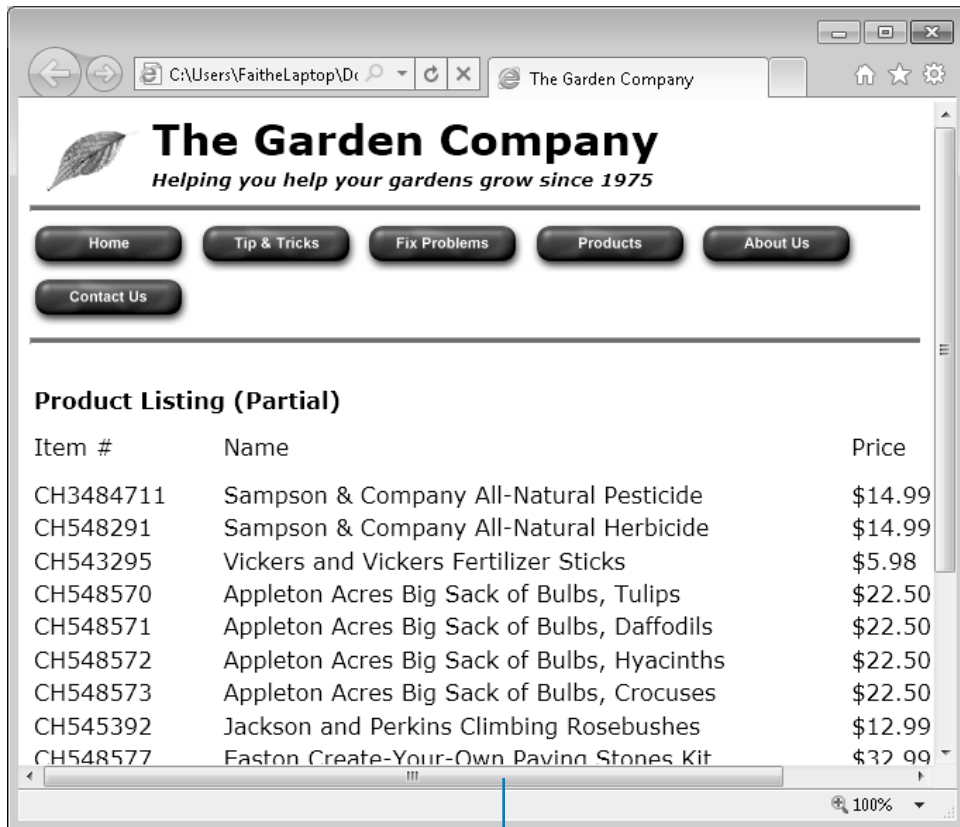
And in a larger window, the extra space between the columns becomes exaggerated.



An alternative approach is to specify a number of pixels for the table's width. That way, the width the table requires in pixels does not change no matter what the size of the browser window. For example, to lock the table to a width of 750 pixels, use the following:

```
<table width="750px">
```

When a browser renders a fixed-width table in a smaller browser window, a horizontal scroll bar appears in the window.



Horizontal scroll bar

When displayed in a larger window, extra horizontal space appears to the right of the table (assuming the table is left-aligned) rather than being distributed throughout the table.