

CSS



David Sawyer McFarland

POGUE PRESS™
O'REILLY®

Beijing • Cambridge • Farnham • Köln • Paris • Sebastopol • Taipei • Tokyo

CSS: The Missing Manual

by David Sawyer McFarland

Copyright © 2006 O'Reilly Media, Inc. All rights reserved.

Printed in the United States of America.

Published by O'Reilly Media, Inc., 1005 Gravenstein Highway North, Sebastopol, CA 95472.

O'Reilly books may be purchased for educational, business, or sales promotional use. Online editions are also available for most titles (*safari.oreilly.com*). For more information, contact our corporate/institutional sales department: (800) 998-9938 or *corporate@oreilly.com*.

Printing History:

August 2006: First Edition.

Nutshell Handbook, the Nutshell Handbook logo, the O'Reilly logo, and “The book that should have been in the box” are registered trademarks of O'Reilly Media, Inc. *CSS: The Missing Manual*, The Missing Manual logo, Pogue Press, and the Pogue Press logo are trademarks of O'Reilly Media, Inc.

Many of the designations used by manufacturers and sellers to distinguish their products are claimed as trademarks. Where those designations appear in this book, and O'Reilly Media, Inc. was aware of a trademark claim, the designations have been printed in caps or initial caps.

While every precaution has been taken in the preparation of this book, the publisher and author assume no responsibility for errors or omissions, or for damages resulting from the use of the information contained herein.



This book uses RepKover™, a durable and flexible lay-flat binding.

ISBN-13: 978-0-596-52687-0

[C]

[3/08]

Building Navigation Bars

Every site needs good navigation features to guide visitors to the information they're after—and help them find their way back. Most sites are organized in sections, such as Products, Contact Info, Corporate Blog, and so on. This structure lets visitors know what information to expect and where they can find it. Much of the time, you find links to a site's principal sections in a *navigation bar*. CSS makes it easy to create a great looking navigation bar, rollover effects and all.

Using Unordered Lists

At heart, a navigation bar's nothing more than a bunch of links. More specifically, it's actually a *list* of the different sections of a site. Back in Chapter 1 you learned HTML's mission is to provide meaningful structure to your content. Accordingly, you should always use a tag that's appropriate to the meaning of that content. For a list of items, that's the `` or unordered list tag—the same one you use to create bulleted lists. It doesn't matter whether you want your list to have *no* bullets or to stretch horizontally across the top of the page: You can do all that by styling the `` tag with CSS. Figure 9-4 shows an example.

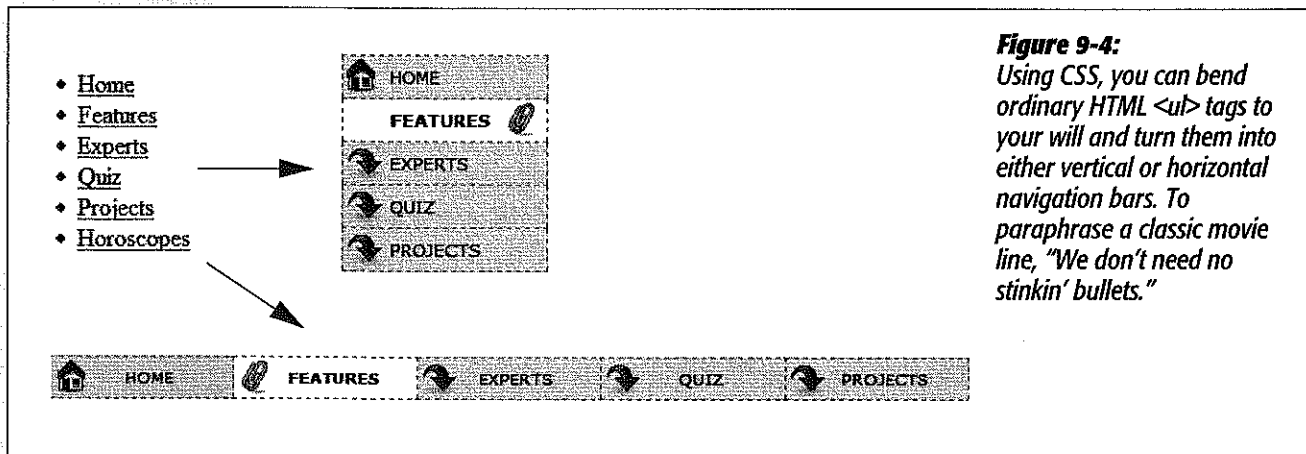


Figure 9-4:
Using CSS, you can bend ordinary HTML `` tags to your will and turn them into either vertical or horizontal navigation bars. To paraphrase a classic movie line, "We don't need no stinkin' bullets."

The HTML for a nav bar is straightforward. There's a single link inside each individual list item. Also, you need a way to style just that unordered list. (You don't want *actual* lists of items to look like navigation bars.) Applying a class or id to the `` tag is a good approach:

```
<ul class="nav">
  <li><a href="index.html">Home</a></li>
  <li><a href="news.html">News</a></li>
  <li><a href="reviews.html">Reviews</a></li>
</ul>
```

The CSS varies a bit depending on whether you want a horizontal or vertical navigation bar. In either case, you need to do two things:

- **Remove the bullets.** Unless the navigation bar is supposed to look like a bulleted list, remove the bullets by setting the *list-style-type* property to none:

```
ul.nav { list-style-type: none; }
```

- **Eliminate padding and margins.** Since browsers indent list items from the left, you need to remove this added space as well. Some browsers do the indenting using *padding* and others use *margin*, so you need to set both to 0:

```
ul.nav {  
    list-style-type: none;  
    padding-left: 0;  
    margin-left: 0;  
}
```

These two steps essentially make each list item look like any plain old block-level element, such as a paragraph or headline (except that a browser doesn't insert margins between list items). At this point, you can begin styling the links. If you want a vertical navigation bar, read on; for horizontal nav bars, see page 222.

Vertical Navigation Bars

A vertical navigation bar is just a bunch of links stacked one on top of the next. Removing the bullets and left margin and padding (as explained in the previous section) gets you most of the way there, but you need to know a few additional tricks to get things looking right:

1. **Display the link as a block.**

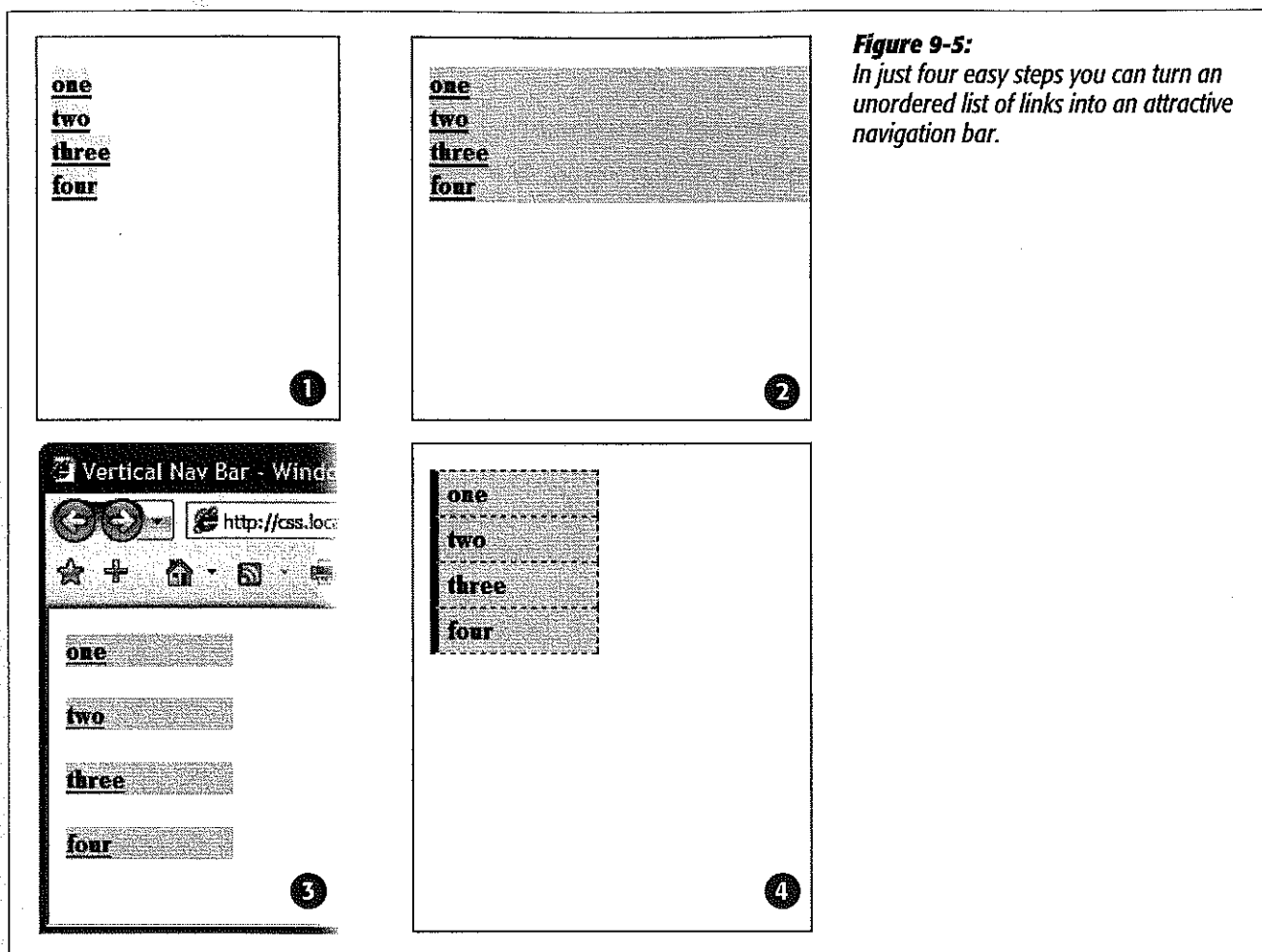
Since the `<a>` tag is an inline element, it's only as wide as the content inside it. Buttons with different length link text (like Home and Our Products) are different widths. The staggered appearance of different width buttons stacked on top of each other doesn't look good, as you can see in #1 in Figure 9-5. In addition, top and bottom padding and margins have no effect on inline elements. To get around these limitations, style the link as a block element:

```
ul.nav a { display: block; }
```

The block value not only makes each button the same width, it also makes the entire area of the link clickable. That way, when your visitors click areas where there's no link text (like the padding around the link), they still trigger the link. (Internet Explorer 6 and earlier has problems with this technique, so look for the fix on page 246.)

2. **Constrain the width of the buttons.**

Making links block level elements also means they're as wide as the tag they're nested in. So when they're just sitting in a page, those links stretch the width of the browser window (see #2 in Figure 9-5). You have several ways to make them



a little narrower. First you can just set the width of the `<a>` tag. If you want each button to be 8 ems wide, for example, then add that to the *width* property:

```
ul.nav a {
    display: block;
    width: 8em;
}
```

Setting a width for any of the tags that wrap around those links—such as the `` or `` tags—also works.

If the button text occupies only one line, you can also center the text vertically so there's equal space above and below the link text. Just add a height to the link and set its *line-height* property to the same value: `a { height: 1.25em; line-height: 1.25em; }`.

Note: You may not need to set an explicit width if the nav bar's located inside a page layout element that itself has a width. As you'll read in Part 3, it's easy to create a sidebar that hugs the left (or right) edge of a page. The sidebar has a set width, so plopping the unordered list nav bar inside it automatically constrains the width of the buttons.

Unfortunately, when you don't set an explicit width for the `<a>` tag, IE has a couple of problems with these links. First, every version (including 7, as of this writing), displays large gaps between each link (see #3 in Figure 9-5). (If you set an explicit width on the `<a>` tags in the nav bar, then you've already taken care of this IE bug. Skip steps 3 and 4.)

3. Remove the gap in Internet Explorer.

To remove this space, you must add a style for the `` tag:

```
ul.nav li { display: inline; }
```

This situation's one of many where you have to send Internet Explorer some nonsense CSS to make it behave. Fortunately, this workaround has no ill effects for other browsers.

4. Expand the clickable area in Internet Explorer 6 and earlier.

Another IE bug (one that's been fixed in IE 7) appears whenever you set a link to display as a *block*. Even though other browsers make the entire block area clickable, IE 6 still limits clicking to just the text inside the link. To fix this, add a style that sets a height for the `<a>` tag. Because this new style would cause the links to shrink to an unreadable height in IE 7 and other browsers, you have to hide it from them. One way to do so is by using the ** html* hack you learned about in Chapter 7 (page 152):

```
* html ul.nav a { height: 1px; }
```

Tip: Another way to create a style that applies only to IE 6 and earlier is to use conditional comments. You can read about them in depth in Chapter 14 (page 399).

Now that all this busywork is out of the way, you can style the buttons to your heart's content. Add padding, background colors, margins, images, or whatever tickles your artistic fancy. If you want to spread the buttons out so they don't touch, then add a bottom (or top) margin to each link.

WORKAROUND WORKSHOP

When Borders Bump

If the buttons in your nav bar touch and you apply a border around each link, then the borders double up. In other words, the bottom border from one button touches the top border of the next button.

To get around this, add the border to only the *top* of each link. That way, you'll get just one border line where the bottom from each button touches the top from the next.

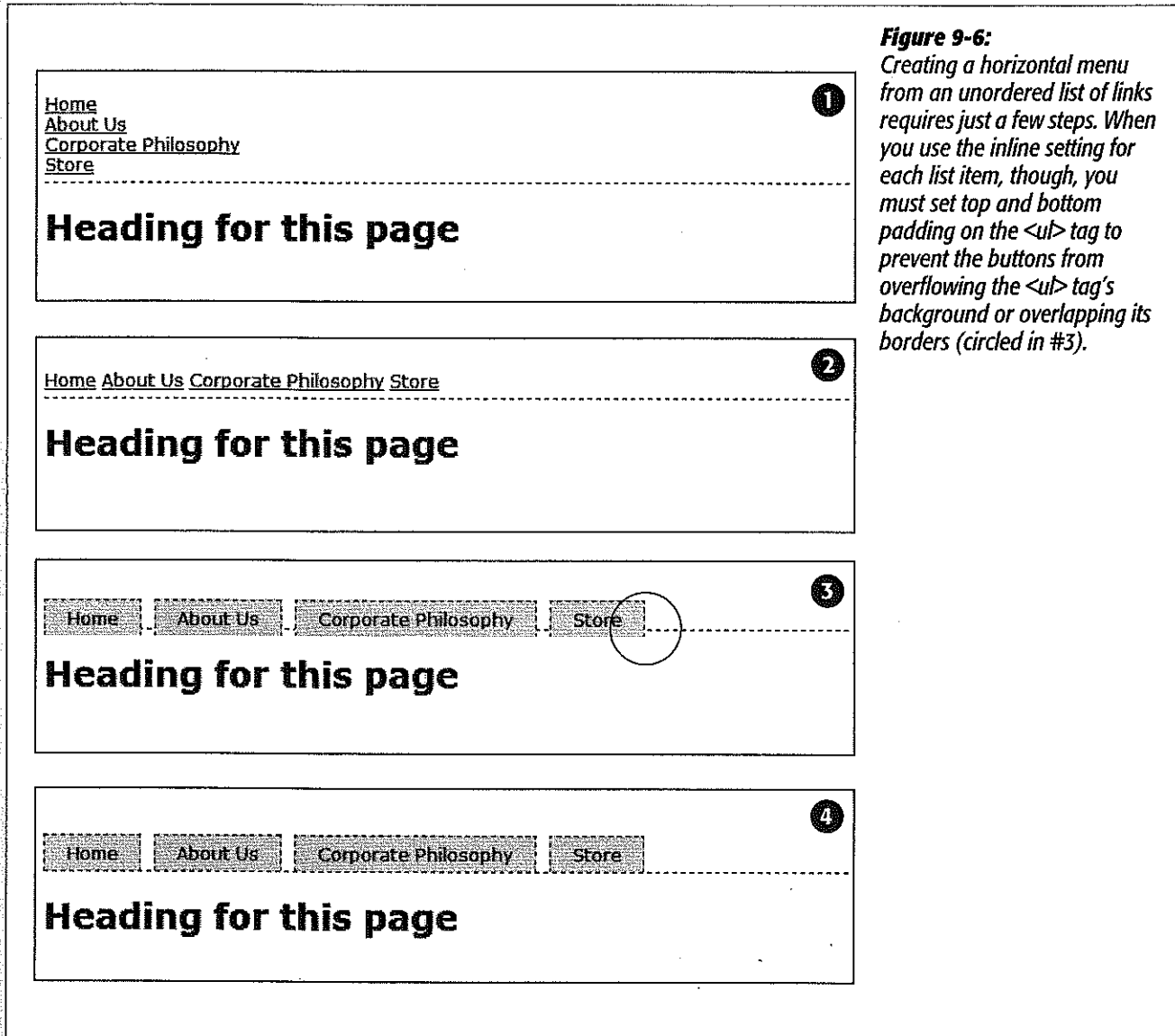
This workaround, however, leaves the entire nav bar borderless below the last link. To fix that problem, you can either create a class with the correct bottom border style and apply it to the last link, or better yet, add a bottom border to the `` tag that encloses the nav bar. (You'll see this trick in action in this chapter's tutorial, on page 243.)

Horizontal Navigation Bars

CSS lets you turn a set of stacked list items into a side-by-side presentation of links, like the one shown back in Figure 9-4. This section shows you two common ways to create a horizontal navigation bar from a list. The first—using the *display: inline* property—is easy, but it can't create equal-size buttons. If a uniform look is what you crave, then turn to the *float*ed `` method described on page 225.

Whichever method you use, start by removing the bullets and left space from the `` tag, as illustrated in #1 of Figure 9-6.

Figure 9-6: Creating a horizontal menu from an unordered list of links requires just a few steps. When you use the inline setting for each list item, though, you must set top and bottom padding on the `` tag to prevent the buttons from overflowing the `` tag's background or overlapping its borders (circled in #3).



Using *display: inline*

The simplest method of creating a horizontal navigation bar involves changing the *display* property of the list items from *block* to *inline*. It's easy to do using CSS.

1. Make the list items inline elements.

Inline elements don't create a line break before or after them as block-level elements do. Setting the *display* property of the `` tags to *inline* makes them appear one beside the other (see #2 in Figure 9-6).

```
ul.nav li { display: inline; }
```

You need to make sure you don't have too many buttons, though. If they won't all fit side by side, some will drop down below the first row.

2. Style the links.

You can remove the underline beneath the links and add a border around them instead. You can also add background color or a background image to provide visual depth. Add padding if you need more room around each link's text. If you want some space between each button, then apply a right margin. The following style gives links a button-like appearance, as shown in #3 and #4 in Figure 9-6:

```
ul.nav a {  
    border: 1px dashed #000;  
    border-bottom: none;  
    padding: 5px 15px 5px 15px;  
    margin-right: 5px;  
    background-color: #EAEAEA;  
    text-decoration: none;  
    color: #333;  
}
```

Note: When you use the inline method to create a horizontal nav bar, don't set the links' *display* property to *block*. If you do, then each link will appear one on top of the other and span the entire width of the page (or entire width of the list's containing block).

3. Add top and bottom padding to the `` tag.

Because `<a>` tags are inline elements, adding top and bottom padding doesn't actually increase the height of a link. Instead, that padding just causes borders and backgrounds on links to overlap elements above and below the link, like the example shown way back in Figure 7-6. In Internet Explorer, the padding can also make top borders on your links disappear. In this case, the `<a>` tag's padding is also making the border on the bottom of the `` tag appear a little above and behind the links (circled in image #3 in Figure 9-6).

The solution's to add padding to the `` tag, which creates space to accommodate the links' overflowing backgrounds and borders. For the `` tag's bottom padding, use the same value as the link's bottom padding. To determine the `` tag's top padding value, add 1 pixel to the link's top padding. (If you're using ems, just make sure the `` tag's top padding is greater than the

top padding used for the link.) For example, the `` tag style to accompany the links in step 2 would look like this:

```
ul.nav {
    margin-left: 0;
    list-style: none;
    padding-left: 0;
    padding-top: 6px;
    padding-bottom: 5px;
    border-bottom: 1px dashed #000;
}
```

As you can see in #4 in Figure 9-6, the bottom padding lets the bottom border of a `` fit in place nicely. One problem with this approach is that there's always a gap between each button, so if you want buttons that touch, then you need to float the links or set a negative right margin on them. Read on for another solution.

Tip: To make this horizontal nav bar appear in the center of the page, add `text-align: center;` to the `` tag's style.

FREQUENTLY ASKED QUESTION

Pop-up Menus

How do I create those cool pop-up menus that display a submenu of links when someone rolls his mouse over a button?

Navigation bars that have multiple levels of menus that pop up or slide out are extremely popular. They're a perfect way to cram a lot of link options into a compact navigation bar. You can create them in a couple of ways.

First, there's the CSS-only approach. One popular drop-down menu technique is called Son of Suckerfish. (The earlier version was called Suckerfish.) You can learn about both here: www.htmldog.com/articles/suckerfish/dropdowns/.

As for creating a multi-level, horizontal drop-down menu, there's a nice easy tutorial at: www.tanfa.co.uk/css/examples/menu/tutorial-h.asp.

The same site provides a tutorial for creating vertical menus with pop-out submenus: www.tanfa.co.uk/css/examples/menu/tutorial-v.asp.

The one disadvantage to the CSS approach is that the submenus disappear instantly if your visitor's mouse strays. You can hope that all your visitors have excellent reflexes, or you can try a different approach: Use CSS to style the buttons and JavaScript to control the actions of the submenus. You can find various free JavaScript-driven dynamic menus at YADM (Yet Another Dynamic Menu): <http://www.onlinetools.org/tools/yadm/>.

For a continuously updated and very powerful commercial JavaScript-driven menu system, check out Ultimate Drop-down Menu (www.udm4.com). It's compliant with current CSS standards, and accessible to search engines, screen readers, and other alternate browsing devices. A commercial license costs around \$70 for a single site. But educational and charitable organizations, as well as people running personal, non-commercial Web sites can get Ultimate Drop-down Menu for free.

Using floats for horizontal navigation

Although the *display: inline* technique for creating a horizontal nav bar is simple, it has one fundamental flaw: there's no way to create equally sized buttons. Setting a width on either the `` or `<a>` tags has no effect, because they're inline elements. To get around this problem, you need to use a little trickier solution—floats.

Note: Nav bars made up of floated elements are hard to center horizontally in the middle of a page. When you need to do that, the *inline* method described on page 222 is better.

1. Float the list items.

Adding a left float to the `` tags removes them from the normal top-down flow of elements:

```
ul.nav li { float: left; }
```

The floated list items (along with their enclosed links) slide right next to each other, just like the images in the photo gallery tutorial on page 188. (You can just as easily float them right if you want those buttons to align to the right edge of the screen or containing sidebar.)

2. Add *display: block* to the links.

Links are inline elements, so width values (as well as top and bottom padding and margins) don't apply to them. Making a browser display the links as block elements lets you set an exact width for the button and add a comfortable amount of white space above and below each link:

```
ul.nav a { display: block; }
```

3. Style the links.

Add background colors, borders, and so on. This part of the process is identical to step 2 on page 223.

4. Add a width.

If you want the nav buttons to have identical widths, set a width for the `<a>` tag (page 146). When you set the *width* property, it's a good idea to use em units because they scale. That way, the link text won't get bigger than the buttons if a visitor increases the browser's font size. The exact width you use depends on how much text is in each button. Obviously for a link like Corporate Philosophy, you need a wider button.

Tip: To center the text in the middle of the button, add *text-align: center;* to the links' style.

5. Float the `` tag.

If it has a border, background color, or image, you should also float the `` tag to the left. If you don't, the list items float outside the `` and cause it to

collapse in height. (Preventing this overflow is known as *containing a float*, and it's discussed in depth on page 293.)

Note: You may want the floated `` to span the entire width of the page (or a containing block like a sidebar)—if you've added a background color to the `` tag that you'd like to span across the page to form a solid stripe. Just set the `` tag's width to 100 percent, as shown in the code following these numbered steps.

Finally, because the `` is floated, there's the potential that content following it may attempt to wrap around the right side of the navigation bar. To prevent this, you need to *clear* the float by applying the CSS `clear` property to whatever follows the `` tag.

6. Clear stuff after the float.

One easy way is to create a class style named `.clear`, like this `.clear { clear: both; }` and then apply it to the first tag after the list.

Tip: You can learn more about clearing floats in Chapters 7 (page 155) and 11 (page 293).

Here are the styles required to create the navigation bar pictured in Figure 9-7. Notice that the buttons are the same width, and the button text is centered.

```
ul.nav {
    margin-left: 0px;
    padding-left: 0px;
    list-style: none;
    border-bottom: 1px dashed #000;
    float:left;
    width: 100%;
}
ul.nav li {
    float: left;
}
ul.nav a {
    width: 12em;
    display: block;
    border: 1px dashed #000;
    border-bottom: none;
    padding: 5px;
    margin-right: 5px;
    background-color: #EAEAEA;
    text-decoration:none;
    color: #333;
    text-align: center;
}
```

Home

About Us

Corporate Philosophy

Store

Heading for this page

Figure 9-7: Floating list items let you create equal width buttons for a navigation bar like this one. You can see the actual CSS that created this bar on the facing page.

FREQUENTLY ASKED QUESTION

Where to Get Navigation Bar Help

I've never made a nav bar before, but I really want my site to have one. I just don't think I can put it all together on my own. Is there something that walks me through the whole process for the first time?

Yes. In fact, there's a tutorial in this very chapter that shows you step by step how to create a navigation bar. Just flip to page 239.

Online, you can also find tutorials, plus tools that do some of the work for you.

For more information on turning ordinary lists into extraordinary navigation elements, visit the step-by-step list tutorial at: <http://css.maxdesign.com.au/listutorial/>.

You can also find loads of cool list-based navigation designs at <http://css.maxdesign.com.au/listamatic/>.

If you want to create tabs for your navigation (like the ones at the top of every Amazon.com page), check out the resources on this page: <http://css-discuss.incutio.com/?page=ListTabs>.

Finally, if you just don't want to bother creating your own, then try the List-O-Matic wizard at www.accessify.com/tools-and-wizards/developer-tools/list-o-matic/ or List-U-Like CSS Generator at www.listulike.com/generator/. Both sites ask for certain information, like fonts and colors, and can create the CSS you need for list-based navigation.

Advanced Link Techniques

If you've mastered the basic `:hover` principle, and know how to add a background image to a link, you're probably hungry for more elaborate ways to spruce up your site's navigation. In the following sections, you'll meet a few of the most popular techniques.

Big Clickable Buttons

The `:hover` pseudo-class is a great way to add an interactive feel to a Web page. But what if you want to highlight an area that's bigger than just a two-word navigation link? Suppose you have a list of news stories in a sidebar. Each item includes the title on one line, followed by a paragraph summary of the story. And suppose you want to highlight the area around both title and summary when a visitor mouses over them (see Figure 9-8).

Fortunately, Internet Explorer 7, Firefox, Safari, and Opera all understand the `:hover` pseudo-class when applied to all kinds of elements, not just links. So if you want to highlight a paragraph when the mouse moves across it, you can do so like this:

```
p:hover { background-color: yellow;}
```