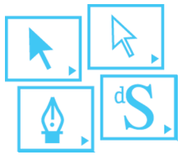
Chapter Opener art not processed.

Chapter 6

CSS Menus



Use CSS menus on just one site. I guarantee you will never look back. If you are looking for professional appearance, easy construction and maintenance, and great navigation for your site, CSS menus are the closest thing to a magic bullet that exists in the Web world.

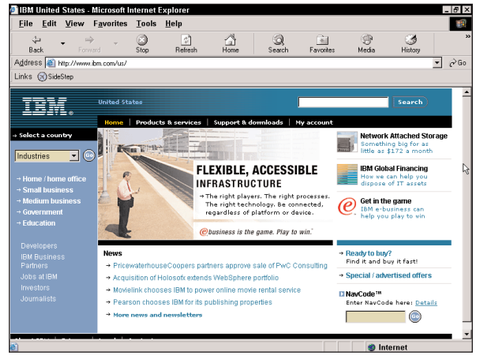
WENDY PECK

I've brought you some solid reasons to use CSS for your content text in the previous chapters of this book. However, the best is yet to come. Putting the power of CSS to work creating menus is the wisest design decision you can make, for yourself and your visitors. In this chapter you learn how to use CSS techniques to create stylized menus with no graphic assistance. CSS menus offer fast download and exceptionally easy editing, and can be very attractive.

CSS and Menus: A Heavenly Match

The objections are probably already forming in your mind: "But you have to work with a tiny selection of fonts to use CSS-based menus?" True, but it's worth it. "But won't CSS menus look different on different monitors?" Yes, but it's worth it. "But don't I lose the opportunity to have pretty pictures in my menus with CSS menus?" Not completely; but anyway, it's worth it. The overriding theme here is: It's worth it!

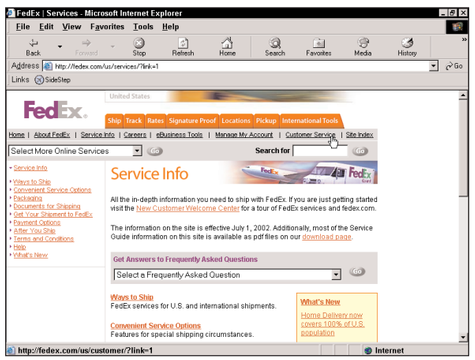
I don't intend to spend a lot of energy trying to explain why you should use CSS menus whenever possible. I'm not alone in my belief that they are the best solution for most menus. Take a quick tour through the top sites on the Web, and I'm confident that you will find CSS-based menus on almost any that you seek. Try Google.com, with all-text menus, or Amazon.com, with all but the main menus controlled by CSS. IBM.com (6.1) features nothing but



● 6.1

CSS menus through the site. FedEx.com (6.2) presents graphic menu tabs for the main menu, but CSS for every other menu item on the site. I don't know about you, but I'm happy to use the same techniques as some of the most innovative companies on the Web, especially when the technique provides faster menu creation and maintenance.

However, it's only fair that I present the pros and the cons for using CSS menus. As in everything bound for the Web, there are compromises. I don't use CSS menus for every menu I create, even today. In fact, my own site (`wpeck.com`) does not use CSS for the main menu, although a text menu is at the bottom of each page. My site is very small, and exceedingly simple, with only the menus (6.3) and page titles providing visual interest. Part III, "Graphic Type for the Web," of this book is devoted to text created as a graphic, with a focus on menus. CSS is not the solution for every menu.



6.2

6.3

Why and Where to Use CSS Menus

Offering definitive "rules" of any type for Web menus is pretty tough. The subject is so complex that I have written an entire book, Web Menus with Beauty and Brains (Hungry Minds, 2000), devoted to the subject of creating menus. However, I do want to give you some guidelines to help you with your menu decisions.

CSS menus provide:

- Smaller page sizes than graphic menus
- Easy construction — just type after the CSS styles are in place
- Fast, easy editing — type in new menu items or delete unnecessary items as text
- Quick design changes — no saving new images
- Rollover effects with no JavaScript code required
- Dependable links
- A unified appearance because they mix well with content text
- "Food" for search engine spiders
- Recognition as links for visitors — not all graphic menus are obvious

CSS menus are perfect for:

- Menus where content changes frequently (news-based sites are an example)
- Sites where visitor feedback is monitored and changes are made on that basis
- Submenus on interior pages
- Main menus with medium-sized text
- Sites maintained by a team — no graphic files to share
- Sites with huge menus — download speed and editing savings multiply
- Sites that also contain mini-sites for special exhibitions, products, events, and so on
- Sites in testing phase

CSS menus should not be used for:

- Menus that require specific fonts to match corporate style
- Menus using novelty fonts
- Menus with large fonts (no antialiasing)
- Menus with tiny fonts (cannot use specialty fonts designed for tiny size)
- Sites that require special characters in menus
- Menus that require any typography control (kerning, varied character sizes, and so on)
- Varied color menus
- Sites that require menus to have graphics rollover effects
- Sites that must deliver text rollover effects in Netscape 4 version browsers

Behind the Scenes

Eric Dunham and NPR: Creating a Visitor-Driven Site

Eric Dunham knows the value of teamwork. Leaving South Carolina with a Fine Arts degree, he made his way through work with an ad agency in Washington, D.C, and a stint with the Baltimore Museum of Art, before arriving at National Public Radio (NPR). He was hired originally to create button graphics, but by becoming friendly with the programmers, and learning about databases, he worked his way into more Web responsibility.

Today, Eric and Katherine Parker form the design team for the NPR site. Visitor value is top priority, and they do work with outside consultants to test and design the best possible architecture for NPR listeners. This team is driven by visitor feedback, and NPR listeners are a vocal crowd. According to Eric, "our listeners are loyal and very possessive about our service." When the team asks for feedback, they get it. Unsolicited email is a constant report card for the Web team, and they track it carefully. It may be impossible to determine whether the visitors get what they require from the NPR site because they are a vocal bunch, or if NPR visitors are a vocal bunch because the Web team listens. Whatever the reason, there is a lesson in the NPR experience for all designers.

Q: What is your role for the NPR Website, and what was your route to working with the NPR Web site?

A: I've been the Senior Web Designer at NPR.org for 31/2 years. I'm responsible for page designs, page graphics, interface designs, and online promotional materials. I came to NPR after building sites for other companies in the Baltimore/DC area.

Q: What was the design team asked to accomplish for the NPR site?

A: The design "team" is very small: three to five core designers/programmers. The goals for the site are constantly evolving as we respond to user needs. Initially, the site was news focused. Based on user research, we've moved the design from being "today's news" focused to "content" focused. Recognizing the importance of good search functionality to Web users, we've been tasked with improving search results and search results display.

Q: Why did you decide to do your text control with CSS?

A: We implemented our first CSS with the last major redesign of the site (June '01). A team of writer/editors was being built to create expanded Web-only coverage of radio pieces. With a diverse group creating compelling content many times a day, a central stylesheet was the best route to providing a consistent user experience. For the next rev, we are planning even more layout control with CSS in addition to XML.

Q: Are there any features to the text presentation on the NPR site that should be noted?

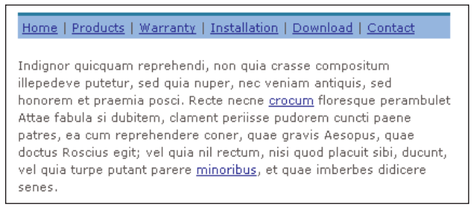
A: We made a significant error in our CSS implementation that we plan to correct with the next major overhaul of the site: pixel based font-size properties. Some browsers cannot scale type with a pixel size specified by a stylesheet. A large percentage of our users are older and prefer a larger type size. We unwittingly did them a disservice by "locking" the font-size. On the other hand, we've used CSS

(continues)

Starting with CSS Menus

Here's the good news. If you have a simple page structure, and require the same color menu items as your links, you already learned all you need to know in Chapter 5. Remember when you were learning to create link effects? They are the ticket to a simple CSS menu. It makes no difference whether text menus are presented horizontally (6.4) or vertically (6.5). That's a layout issue that I cover later in this chapter.

To create menu items as I have shown in 6.4 and 6.5, you simply type the menu titles that you require and assign a link. The code is very simple for the horizontal menu.



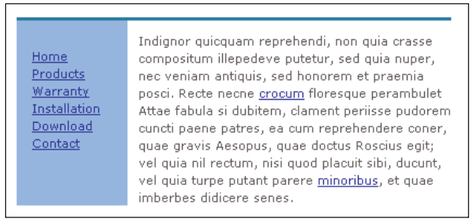
● 6.4

[NOTE]

I ignore the code for a vertical menu (6.5) until later in this chapter because the formatting for this layout is mixed with the menu items.

```
<a href="index.html">Home</a>
| <a href="prod.html">Products</a>
| <a href="warr.html">Warranty</a>
| <a href="inst.html">
    Installation</a>
| <a href="down.html">Download</a>
| <a href="contact.html">Contact</a>
```

I'm not making a great argument for CSS menus with the examples in 6.4 and 6.5, however. You can

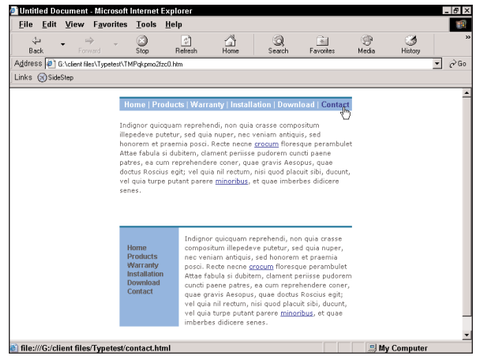


● 6.5

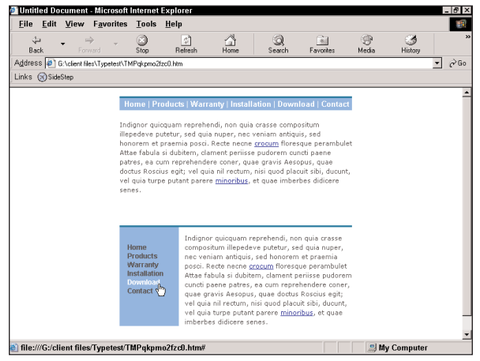
see that the links in the menu and the links in the body copy are the same. Of course, they are, because the same CSS styles format all the links. Even if the color works for both menus and body copy, it is rare that the same link styles are best for both menus and body copy. I strongly recommend that links in content be underlined, yet menu links, especially in a vertical stack, are better without underlining for easy reading. That sounds like a contradiction to my hardline "underline links" stance, but menus present themselves as links by formatting and position. I also don't have any color recommendation on menu text, other than it be easily read, and work with the page.

Defining Multiple Link Styles on a Page with Class Styles

Luckily, you have an easy way around links with different requirements on the same page using CSS. A special type of CSS style, called a class style, solves the problem. You create a uniquely named style and assign whatever attributes you desire, including link attributes. You can apply that special style to any selector, and when you use it for links, you have no limit to the number of different type link effects you can place in a page. This example page shows three different link styles and rollover effects all on one page. The rollover effect for the links in the top menu (6.6) and side menu (6.7) are controlled by class



● 6.6



● 6.7

(continued)

text to replace a number of graphics with CSS text, improving site maintanence, site updating, and download times.

Q: Did the team hit any trouble spots with the design, and if so, how were they solved?

A: No significant trouble spots were encountered. We are always adding to and working on an internal list of site improvements based on user, business, and programming needs.

Q: I really like how easy it is to find information on your pages, even though there are many, many links. Can you talk a little about how the information is presented and what the route was to this format?

A: We adhere to basic best practices for interface design while recognizing our unique position on the Web. Through user research, we found that, generally, our users map their experience of the radio to the Web site. This situation raises some interesting problems (time zones, local vs. national, brand blur, Member station sites and schedules) that a consistent, clear interface can solve.

The site has become more and more modular over time which helps users quickly find information or "parse" chunks of the pages. Modularity is achieved by standardizing the presentation of site elements. For example,

a link to audio has it's own icon and is placed at the top of an expanded coverage page. In addition, we have formats for particular "types" of content or features: search results look like program rundowns which look like news items. This consistency gives the site modularity and improves the user experience: users become familiar with how the site operates across a variety of sections.

Q: How is the site content maintained?

A: There are a number of different types of content on the site. Audio, search, e-commerce, news, Web-only content, and expanded coverage are all handled by different systems. NPR.org's talented programmers are building a custom CMS that will bring the different types of content under one umbrella.

Q: How important is bandwidth, page download time, etc, to the Web team at NPR?

A: We work hard to keep the pages small and minimize impact on the users' machines. However, our metrics show that a large percentage of our users access the site over a broadband connection, usually during work hours, which gives us some flexibility. The most important factor is metrics: page views, time on page, user paths, e-mailed stories, etc. With strong data, we can make informed decisions about what features need to be improved or highlighted. We also attempt to answer most user e-mail. User e-mail is very powerful for addressing problems with the site that may not have occurred to us or surfaced during user testing. It's also a wonderful tool for ideas!

Q: Is there anything you would like to add about the NPR site?

A: NPR is unique on the Web: it provides content that is deep and rich that listeners/users are passionate about. The audio content is wonderful to serve online: users can listen while continuing to work, catch a story they missed on their commute, e-mail links to their favorite stories to friends and family, and enjoy online-specific content that supports and extends their radio experience. Providing a positive user experience in that context is very rewarding for NPR.org's small, dedicated team.

Q: Do you have any tips to pass on about Web design?

A: Always remember the user. Know them, talk to them, and learn from them. Always remember the design community. Network and interact with your colleagues online and at conferences: take advantage of the medium by sharing ideas for creating the best user experience.

Blue, Underlined Links are Never Wrong

I talk in earlier chapters about why you should use underlining for links. Underlined text is an instant link flag for your visitor. As a surfer myself, I find it irritating when I have to use my mouse to see whether links exist, rather than having them presented clearly to me. Even with as much time as I spend on the Web, I'm never sure whether a colored, boldface word in text is highlighted for emphasis, or is a link. The only way I can tell is to move my mouse over the link and see whether I either receive the hand icon indicating a link, or have another mouseover effect signal the link. That uncertainty is frustrating for visitors. Please underline your links. If you have too many links in your copy, and the underlining is distracting from easy reading, perhaps your page would be improved by adding a small menu to direct traffic.

The issue of whether to use blue or not is a little less critical, in my opinion. If you have underlined links, and there is just no way that blue looks good with your body text, then using another color for your links sometimes makes sense. However, and this is a big however, you *are* reducing your visitor's instant recognition of your links. Many blues are available, even within the Web-safe palette. I've not yet designed a site that doesn't work with blue links. The links are not always bright blue. Some sites have blue-gray links, and others feature a blue-green. Finding a blue that works for your site is worth it. You have enough ways to inadvertently slow your visitor's progress through your pages. Why gamble with one of the few truly predictable visitor behaviors?

If you use links without underlining, and a different color than blue – and some of you are not going to like this next statement – you are clearly putting your design entertainment ahead of your visitor's needs. If you are designing a site for your own pleasure, that is acceptable. If you are creating a site that depends on visitors, do so at your peril. Remember, your visitors have the power of the mouse. They can and *will* use that power to leave a site that is even slightly confusing to navigate.

styles, named `.topmenu` and `.sidemenu`, respectively (I chose the names). The body copy rollover effect (6.8) comes from the definitions for the default link definition (`a:link`) in the style sheet.

Creating the various states for a menu link style is exactly the same as creating states for the default link style, which you learned in Chapter 5, with one exception. You must first name a class style. You can call the class anything, but using meaningful names makes future editing much easier. This is not the place to be creative, however. I usually name my styles with literary gems like `topmenu`, `rightmenu`, and `servicemenu`. Resist using mixed-case names (see nearby sidebar on naming styles).

If you worry that you may have trouble identifying a style in the future, you can use a comment within the style to jog your memory. Use the following syntax to include a comment in your style sheet:

```
/* Type any comment here. */
```

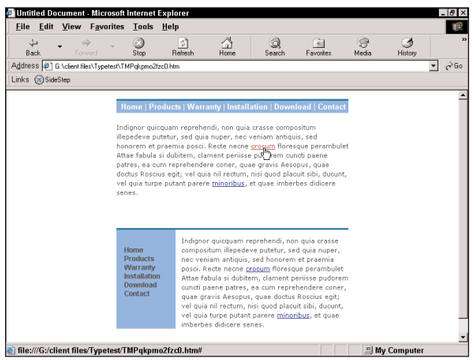
Class styles always start with a period (.). A style you want to name topmenu appears in a style sheet as a selector called .topmenu. Styles for a menu named `leftmenu` are defined `.leftmenu`. The basic syntax for a style called `topmenu` follows:

```
.topmenu {
   property: value
}
```

I want you to ignore that class styles are often used for menus for a few minutes. Look at the following defined style:

```
.redtext {
   font-weight: bold;
   color: #FF0000;
}
```

If this style is called in your document, the text that is affected by the style will be displayed in boldface and red. The following code displays a portion of the paragraph in red, boldface text, as defined by the CSS style shown in the previous example (6.9). Attributes



● 6.8

Naming Styles

Eric Meyer, the CSS expert featured in an interview in Chapter 4, lists mixed-case style names, and including illegal characters in style names as one of the common errors he sees on Web sites. He advises using lowercase names for styles you create and name, and understanding which characters can be used. (See `http://www.w3.org/TR/REC-CSS2/syndata.html#q4` for the rules on characters and recommendations on style name case.)

Many designers use class style names like TopMenu or LeftMenu to make their code easier to read, but this practice is not good. A class style named topmenu will always be acceptable. TopMenu may cause problems in some documents.

CSS is not case-sensitive, but documents using CSS may be case-sensitive. In particular, although class names are not case sensitive for HTML documents, they are for XML documents. You simply cannot go wrong by using all lowercase names, with no characters other than A–Z and 0–9 in your style names.

that are not specified in the class style are inherited from parent tags, in this case `<p>`.

```
<p>You can apply a group of text
    attributes, <span
    class="redtext">as
defined in the CSS class
    style</span>, to any portion of
    text on the page.
Simply enclose that text you wish to
    affect within a style.</p>
```

In this case, the class style is applied to the `<span>` tag (6.9). This tag allows you to enclose only the text that you want to affect, not an entire block, as with `<p>` or `<h?>` tags.

You can apply a class style to any selector, however. Suppose you wanted this entire paragraph to be formatted with the class style. The following code applies the class style to the entire first paragraph (6.10). The second paragraph has no class style applied, so it is displayed with the defined paragraph attributes.

Specify a class style and you can apply a sets of text attributes, **as defined in a CSS style**, to any portion of text on the page. Simply enclose that text you wish to affect within a style.

● 6.9

```
<p class="redtext">If you apply a
    class style to a block element,
    like
the paragraph tag, the entire block
    is affected.</p>

<p>This paragraph does not have the
    class style applied, so follows
    the
defined paragraph style.</p>
```

Defining Link Attributes for Class Styles

I hope that your design mind is racing with the possibilities if you have just been introduced to class styles. But remember that you have a lot of design power for menus when you assign link attributes to a class style. This method is how I managed to create the document with three different link definitions that you saw earlier in this chapter (6.6, 6.7, 6.8).

In this section, I step you through creating link styles for three distinct menu styles (6.11). You'll learn to create a simple horizontal menu, a menu

If you apply a class style to a block element, like the paragraph tag, the entire block is affected.

This paragraph does not have the class style applied, so follows the defined paragraph style.

● 6.10

● 6.11

with decorative border styling and one that is formatted using a table. These menus can form patterns that you can use for any menu.

First, here is the complete code for a very simple class style named `simple.`, with link attributes (6.12). You can refer to this basic example if you get confused when the number of attributes grow in the more stylized menus.

```
.simple {
   font-family: Arial, Helvetica,
    sans-serif;
   font-size: 12px;
   font-weight: bold;
   color: #003300;
}


.simple a:link {
   color: #006633;
   text-decoration: none;
}


.simple a:visited {
   color: #CCCC99;
   text-decoration: none;
}


.simple a:hover {
   color: #996633;
}
```

[TIP]

The style `.simple` displays where there is no link in your menu. When you create a menu, most text will be a link, and is controlled by the link styling for the class style. However, the original class style controls the dividing characters between menu items, or headlines that are used to divide menu categories but are not links themselves. You can work with this feature to add extra design savvy to your menus.

● 6.12

Block and Inline Elements with Class Styles

Block elements are HTML elements that have line breaks, including `<p>`, `<h?>`, `<li>`, `<table>`, and `<form>` to name just a few.

Inline elements do not have line breaks, and include `<strong>`, `<em>`, and `<span>`.

You can apply class styles to both block and inline elements, and you can use inline elements within block elements. For example, you can enclose any text within a paragraph or table (block elements) with a `<span>` tag (inline element), and apply a CSS style to the `<span>` tag. The class style affects only the text between the `<span>` and `</span>` tags.

On the other hand, applying a class style to a `<p>` or `<table>` tag affects the entire paragraph or table. All text within a paragraph controlled by `<p class="redtext">` displays both the `<p>` tag attributes and the attributes defined in the class style. The styles in the class style override any attributes it shares with the `<p>` tag.

How do you decide where the class style should be applied? Try to place class styles as high in the layout order of the page as possible to reduce the code required. In other words, use a block element rather than an inline element whenever you can. If all text within a table should have the class style applied, include the class style with the `<table>` tag, `<table class="redtext">`. If only a table cell requires the class style, which is common for menu application, apply the class style to the `<td>` tag, `<td class="redtext">`.

After you're comfortable with creating a simple class style, and specifying link styles to go with that style, you can move on to using the power of CSS decorative features. This sample page (6.11) contains four separate menus. You cannot see a text menu at the bottom of the page. That menu is controlled by the page link attributes that you learned to create in Chapter 5. In this section, I show you how to create the other three menus on the page.

[NOTE]

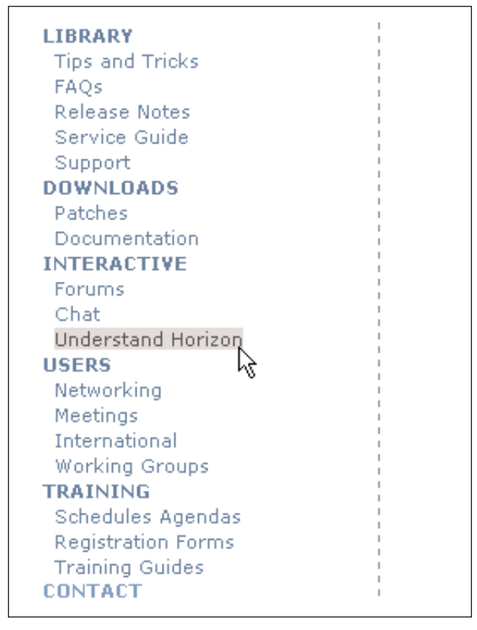
Each of you create your CSS in a different way, some with a text editor, some with an on-board visual editor CSS creation, and others with dedicated programs like TopStyle. I step you through the process with descriptive passages for why and how a technique is done, along with the pure CSS code that is required for that effect.

The main menu (6.13) runs horizontally. This menu is quite simple with a class style applied to the table cell containing the menu, and the dotted line specified as part of the style.

The long side menu at the left edge of the page (6.14) is formatted in a table, with the class style controlling the links. The dotted line to the right of the menu is controlled by this style.

Finally, the upper-right menu (6.15) is created fully with CSS. Class styles containing all the border, margin, and font information are applied to `<p>` tags to form the individual boxes containing the links.

● 6.13



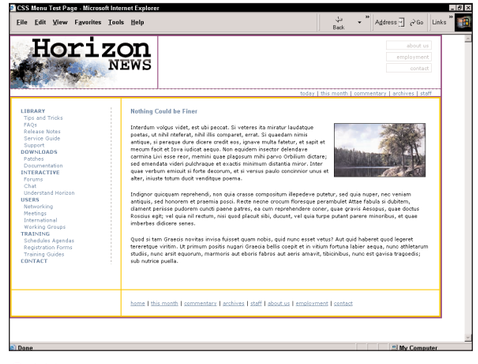
● 6.14

To give you an idea of how the page is laid out, I created a sample with table borders applied (6.16). The purple border represents the main table for the page. This table has no cell padding or spacing, and provides the overall layout for the page. In the sample shown here, I reduced the table size to 90% from 100% to make the borders easier to see. Note that a cell that spans two cells holds the main menu so that the CSS style in the menu (dotted line) stretches across the page.

The yellow table provides column layout and padding for the content area of the page. Note that the dotted line is contained within the left column, as it is added through the CSS for that cell. These layout tables are the only ones on the page, although the left menu is formatted with a table.

The margin for the page is set to zero through the CSS margin style.

● 6.15



● 6.16

In this view, you can see the text menu at the bottom of the page.

If you want to create this page exactly as I have done, you must add the following CSS styles to your style sheet before you begin to create the menus:

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


p {
   font-size: 11px;
   font-family: Verdana, Arial,
    Helvetica, sans-serif;
   line-height: 150%;
}


h1 {
   font-size: 120%;
   font-family: Arial, Helvetica,
    sans-serif;
   color: #336699;
}


a:link {
   color: #336699;
}


a:visited {
   color: #660033;
}


a:hover {
   color: #000000;
}
```

[NOTE]

You can find the HTML code for this page, as well as the completed CSS style sheet in Appendix C.

Creating the Basic Main Menu Style

The main menu is placed in a two-cell span to allow the dotted line to stretch across the page. Text is typed into the spanned cells, with the cell alignment set to right.

1. **Enter the text for the menu and assign links using the following code:**

```
<a href="today.html">today</a> |
<a href="month.html">this month</a> |
<a ref="comment.html">commentary</a>
    |
<a href="archives.html">archives</a>
    |
<a href="staff.html">staff</a>
```

[WARNING]

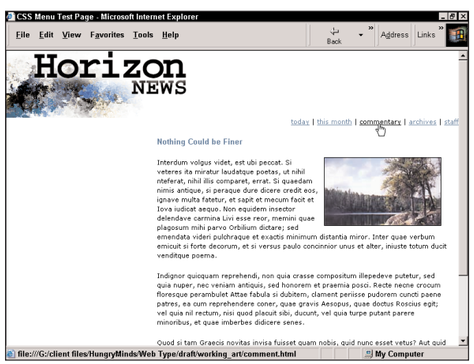
There is a space following the ⊢character in the previous code.

Your text should be at the right edge of the page (6.17) and have the properties of the `a:link` style for the page. At this point, you have not added any style to the menu, so it is picking up the default values, which includes underlining for links.

2. **Create a class style named `main menu` and assign the following values:**

```
.mainmenu {
   font-family: Verdana, Arial,
    Helvetica, sans-serif;
   font-size: 11px;
   color: #000066;
}
```

This code sets the color and font for the style, but adds no styling. Before you address any link attributes, having your menu format in place is



● 6.17

a good idea, so you can see the total effect of the menu as you add effects. In this case, you must add a dotted border above the text, and move the menu away from the margin.

3. **To add the style to the table cell, add the class style to the `<td>` tag as follows:**

```
<td class="mainmenu">
```

There will be table formatting attributes already in the `<td>` tag. It doesn't matter where the `class="mainmenu"` goes, as long as it is enclosed in the `<td>` tag. The `mainmenu` class style now controls your menu text.

[TIP]

CSS provides both margin and padding settings for any style. The best way to become familiar with these attributes, and how they interact with borders and background, is to devote some time to "playing." I was having a hard time remembering what did what with CSS decorative attributes from the time I created one style sheet to the next. Finally, I spent almost an entire afternoon specifying and previewing backgrounds, various borders, padding, margins, and so on. I tested every "what if" I could imagine. That short timeout from building active CSS style sheets has paid off many times over.

4. **Because you want the style to display right to the edge of the page (remember the dotted line), but want the text to end before the end of the style, padding is the correct attribute to adjust in this case. Add the following value to your `mainmenu` class style:**

```
padding-right: 20px;
```

The text has now moved away from the right margin (6.18).

5. **Use the border attribute to create the dotted line. Add the following code to your `mainmenu` style:**

```
border-top-width: 1px;
   border-top-style: dotted;
   border-top-color: #666666;
```

These attributes add a gray dotted line 1-pixel wide along the top border of your style (6.19).

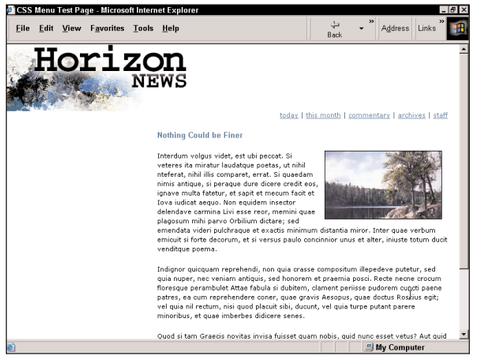
6. **The line style is starting to shape up, but the border is a little too close to the text in the menu. Add a little white space by specifying a 2-pixel padding value for the top of the style as follows:**

```
padding-top: 2px;
```

It's amazing what a couple of pixels can do. Your `mainmenu` basic style is complete (6.20).

Adding Link Attributes to the Main Menu

The basic style for your main menu is in place, so now is the time to set the attributes for your links. Nothing in the styling you added in the previous exercise is meant to change as the menu is used. You can let the values from the basic `mainmenu` class style provide the setting for the text, and simply specify the values you want to have for links and mouseovers and to mark pages that have been visited.



● 6.18

● 6.19

● 6.20

Add the following styles to your mainmenu style sheet.

1. **Set your link colors by adding an `a:link` style to the `mainmenu` class style. The following code removes the underline from links and sets the color (6.21):**

```
.mainmenu a:link {
   color: #336699;
   text-decoration: none;
}
```

2. **Set the specifications for how the links will display when the visitor has already visited the page for that link, in this case as a medium gray (6.22):**

```
.mainmenu a:visited {
   color: #999999;
   text-decoration: none;
}
```

[WARNING]

You must visit the linked page and return to the page you are previewing in order to see the visited style in action.

3. **Finally, you can include a mouseover effect for your menu links. Adding a background to a mouseover effect delivers a very clear message that the text is a link, as well as marking clearly which link is highlighted. Add a background as well as confirm the text color for a mouseover effect by adding the following code:**

```
.mainmenu a:hover {
   color: #336699;
   background-color: #CCCCCC;
}
```

6.21

6.22

[WARNING]

You should specify text color again, as a visited link will display the text color for that style. At times, that can cause disappearing text, so covering all your bases by restating the color is best, even if it is the same as the link color.

Your main menu is now complete (6.23). For practice, try applying the mainmenu style to a variety of text menus in different forms. Remember that you can add a class style to any selector.

Adding Decorative Styling: Top Menu

If you worked through the previous exercise, you will have no trouble creating the menu that is shown in the top-right corner of our sample page (6.11). The same methods are used to create both the main menu and this smaller menu, but in this case, you work with more border attributes to create a stylized menu (6.24).

6.23

6.24

To create a stylized menu:

1. **The text for this menu is entered in a separate table cell with top alignment. Enter the code and text as follows:**

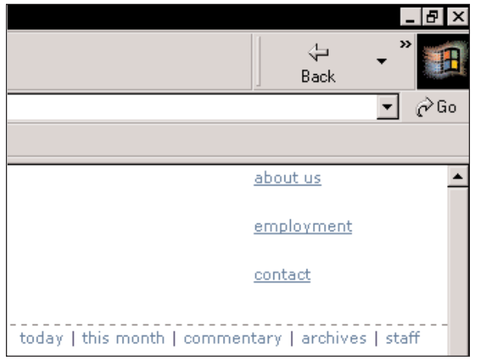
```
<td valign="top">
     <p><a href="about.html">about
   us</a></p>
    <p><a
   ref="employ.html">employment</a><
   /p>
    <p><a
   href="contact.html">contact</a></
   p></td>
```

I included the table cell tags so you can make sure that your cell matches this example. Table cell formatting could interfere with attributes in the CSS decorated style. The text will be tight against the top and left borders of the cell (6.25).

2. **Create a class style named `toprightmenu` with the following attributes.**

```
.toprightmenu {
   font-family: Verdana, Arial,
    Helvetica, sans-serif;
   font-size: 11px;
   color: #999999;
   text-align: right;
   text-decoration: none;
}


.toprightmenu a:link {
   color: #999999;
   text-decoration: none;
}


.toprightmenu a:visited {
   color: #6699CC;
   text-decoration: none;
}


.toprightmenu a:hover {
   color: #336699;
}


.toprightmenu a:visited {
   color: #6699CC;
text-decoration: none;
}


.toprightmenu a:hover {
   color: #336699;
}
```

These styles control the text formatting for this menu, although you will see no change to the text until you add the style to the text in the next step.



6.25

3. **Add the class style `toprightmenu` to each of the `<p>` tags in the menu text. Your code should look like this:**

```
<p class="toprightmenu"><a
  href="about.html">about
  us</a></p>
<p class="toprightmenu"><a
  href="employ.html">employment</a>
  </p>
<p class="toprightmenu"><a
  href="contact.html">contact</a></
  p>
```

The text moves to the right margin of the cell, and contains the text formatting and rollover style that you placed in the class style in Step 2 (6.26).

[NOTE]

You add the class style to each paragraph separately so that the borders will surround each menu item. If you applied the style to a `<span>` or `<td>` tag, the border would appear around the entire area, not individual menu items.

4. **Add a border to all sides by adding the following code to the main `toprightmenu` style:**

   ```
   border: 1px solid #CCCCCC;
   ```

   The borders are added to the menu items, and extend the full width of the table cell (6.27).

5. **To control the size of the box containing the menu items, and therefore control where the borders display, add the following code to the `toprightmenu` style (6.28):**

   ```
   height: 18px;
   width: 100px;
   ```

6. **Now that the box has a defined size, the menu has moved to the left in the cell. The text is still aligned to the right of the box, but the box is following default cell alignment, or left. Change the cell alignment to right (6.29).**

7. **Take a good look at the text within the box, and the box within the page to plan the next steps. You can control where the text appears in relation to the box, and where the box appears in relation to the cell containing the menu through padding and margin attributes. Padding controls the distance between the edge of the box, defined by the border in this case, and the contents of the box (text). Margin values control how far the box is in relation to the element containing it, or table cell in this case.** Use the following code to format the padding and margins for this menu:

   ```
   padding: 1px 5px 1px 1px;
   margin-bottom: -5px;
   margin-right: 20px;
   margin-top: 10px;
   ```

Your `toprightmenu` style should be as follows.

```
.toprightmenu {
   font-family: Verdana, Arial,
    Helvetica, sans-serif;
   font-size: 11px;
   color: #999999;
   text-align: right;
   text-decoration: none;
   border: 1px solid #CCCCCC;
   height: 18px;
   width: 100px;
   padding: 1px 5px 1px 1px;
   margin-bottom: -5px;
   margin-right: 20px;
   margin-top: 10px;
}
```

● 6.26

● 6.27

Your menu is now complete (6.30). Try creating a new menu, and applying this style. The new menu should look the same as this menu. That's the power of CSS.

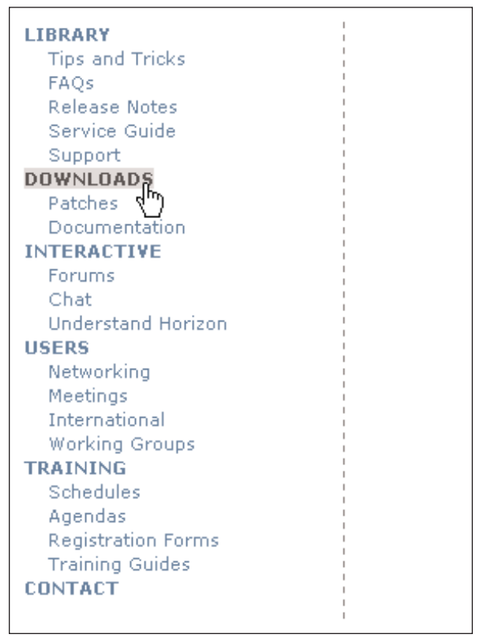
Formatting a Menu in a Table

The left menu on this page does not depend on CSS styling for design. The text and rollover effects are created with CSS, as is a dividing line between the menu and page, but unlike the previous sample, the you accomplish the formatting for this menu with a table (6.31). The two column format allows you to

● 6.28

● 6.29

● 6.30



● 6.31

set an indent exactly as you require. I've added borders so you can see the table structure (6.32).

NOTE

This menu has headlines for categories that are also links. If they were plain text, and not links, you could create the class style with no padding, and add left padding to the link styles for the class, creating the indent.

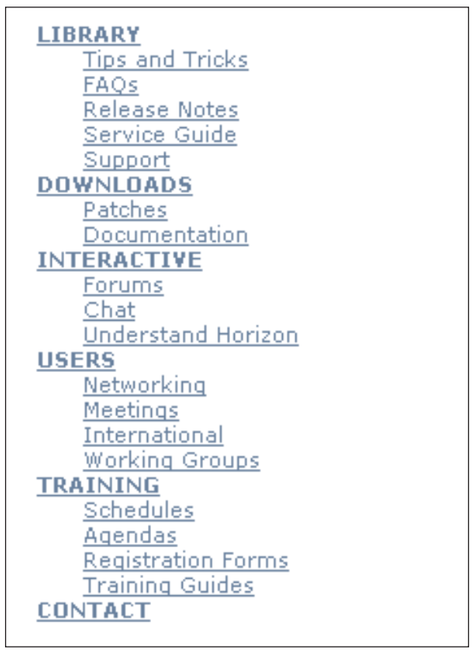
With headlines as text, you could also achieve this effect by creating two separate class styles, and applying padding to the indented style. However, applying this type of formatting with a table is often easier, and more logical. I prefer to keep only one class style in a menu so that future editing is simple and fast.

● 6.32

The table for the sample menu is 12 rows by 2 columns, with no cellspacing or margins. I used 100% for the table width to fill the menu area. Headline rows are created by a 2-cell span, and are typed in all caps. Menu items are typed with initial caps in the second column of the row. A transparent GIF file holds the indent cell open.

1. **Create the table and linked text with the following code. This code creates a table and adds the links, but does no text formatting. The links will follow the style for the default links on your page (6.33).**

```
<table width="100%" border="0"
    cellpadding="0" cellspacing="0">
<tr valign="top">
<td colspan="2"><a
    href="library.html"><strong>LIBRA
    RY</strong></a></td>
</tr>
```



● 6.33


```
<tr valign="top">
     <td><img src="art/spacer.gif"
    width="15" height="1"></td>
     <td><a href="library_tips.html">
    Tips and Tricks</a><br>
     <a href="library_faq.html">
    FAQs</a><br>
     <a href="library_release.html">
    Release Notes</a><br>
     <a href="library_serviceref.
    html">Service Guide</a><br>
     <a href="library_refdocs.
    html">Support</a></td>
</tr>
<tr valign="top">
     <td colspan="2"><strong><a
    href="download.html">DOWNLOADS
    </a>
     </strong></td>
</tr>
<tr valign="top">
     <td> </td>
     <td><a
    href="down_patches.html">Patches
    </a><br>
     <a href="down_documentation.
    html">Documentation</a></td>
</tr>
<tr valign="top">
     <td colspan="2"><a href=
    "interactive.html"><strong>
    INTERACTIVE
     </strong></a></td>
</tr>
<tr valign="top">
     <td> </td>
     <td><a href="inter_forum.html">
    Forums</a><br>
     <a href="inter_chat.html">
    Chat</a><br>
     <a href="inter_understand.html">
    Understand Horizon</a></td>
</tr>
<tr valign="top">
     <td colspan="2"><a href=
    "usergroup.html"><strong>USERS
     </strong></a></td>
</tr>
<tr valign="top">
     <td> </td>
     <td><a href="user_networkprof.
    html">Networking</a><br>
     <a href="user_meet.html">
    Meetings</a><br>
     <a href="user_international.
    html">International</a><br>
     <a href="user_workgroup.html">
    Working Groups</a></td>
</tr>
<tr valign="top">
     <td colspan="2"><a href=
    "training.html"><strong>
     TRAINING</strong></a></td>
</tr>
<tr valign="top">
     <td> </td>
     <td><a href="train_sched.html">
    Schedules</a><br>
     <a href="train_agendas.html">
    Agendas</a><br>
     <a href="train_regforms.html">
    Registration Forms</a><br>
     <a href="train_guides.html">
    Training Guides</a></td>
</tr>
<tr valign="top">
     <td colspan="2"><strong><a href=
    "contact.html">CONTACT</a>
     </strong></td>
</tr>
</table>
```

2. **With the basic formatting handled by the table, your class style is very simple. Add the following class style code to your style sheet:**

```
.leftmenu {
   border-right-width: 1px;
   border-right-style: dotted;
   border-right-color: #666666;
}

.leftmenu a:link {
   color: #336699;
   text-decoration: none;
   font-family: Verdana, Arial,
    Helvetica, sans-serif;
   font-size: 11px;
   line-height: 130%;
}
```

```
.leftmenu a:visited {
   color: #6699CC;
   text-decoration: none;
}

.leftmenu a:hover {
   color: #333333;
   background-color: #CCCCCC;
}
```

You have already added the attributes used in these styles to other styles in this chapter. Now you must apply the style to your menu. In this case, you want the border line at the right to include the entire area of the menu, so the style is applied to affect the full table. Adding the style to the `<td>` tag that contains the table applies the style to the full menu and places the dotted line right at the edge of the cell.

3. **Add the `leftmenu` style to the table cell containing the menu table, using the following code:**

```
<td width="200" valign="top"
    class="leftmenu">
```

That's it for the special menus on your page (6.34). Add a text menu to the bottom of the page for easy navigation (6.35), and your page is complete. This page displays consistently in all current browsers, and degrades reasonably well to display in Netscape version 4 browsers (6.36). The links will work, and although the upper menu is out of place and the left menu is missing the dividing line, visitors can navigate easily.

The techniques you used to create the menus on this page give you the tools to create many, many menu styles. Without CSS, you would have to create these menus in graphic programs, export them, and include them in JavaScript rollover scripts. CSS is a much more efficient way to accomplish many menu styles.

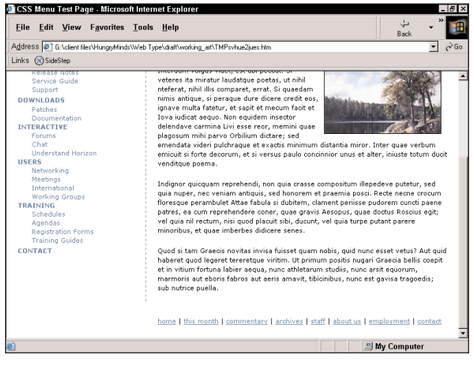
In a different section of your site, you can use the same menu styles with completely different menu items. If you were using graphic menus, you would have to start over and design the second menu, exporting and placing in rollovers as for the original. There is just no comparison in the time required to create a second menu using CSS or graphics. With CSS, you type the menu titles and links, apply the CSS style, and your menu is complete.

Design Tips for CSS Menus

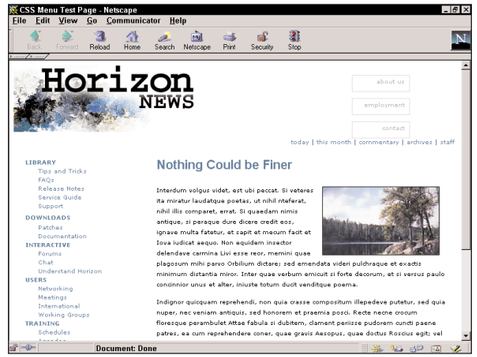
Knowing how to create a menu controlled by CSS is one thing. Creating a CSS menu that looks good is another. I've seen some pretty hideous CSS menus out there. Of course, some of the menus created with graphic text are much worse, as the designer has no restrictions. However, you do need to keep design principles in mind for type, even when using CSS. I've compiled a list of tips to help you avoid most of the errors I have seen.

- **Use identical fonts for related styles.** Make sure all your link styles have the same base font, including type style (6.37). Using a boldface or italic font for one of the link states causes your text to move so that the characters in the different style fit (6.38). At best, these fonts are uncomfortable for the visitor because the text moves on mouseover. At worst, they can break your page (see the sidebar "One Tiny Error = One Broken Page").
- **Don't use large type for CSS menus.** Remember that HTML text, the type that CSS controls, does not offer antialiasing to keep character outlines smooth (6.39). If you must fill a large space, use a background color or image for the menu area, or enhance the text with graphic symbols. Bigger is not always better; when it comes to text, large text is rarely the answer to design problems.

6.34



6.35



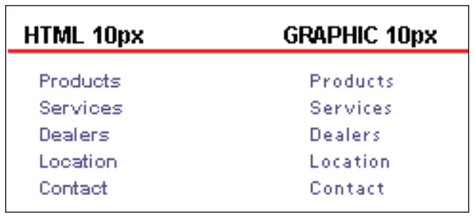
6.36

6.37

6.38

6.39

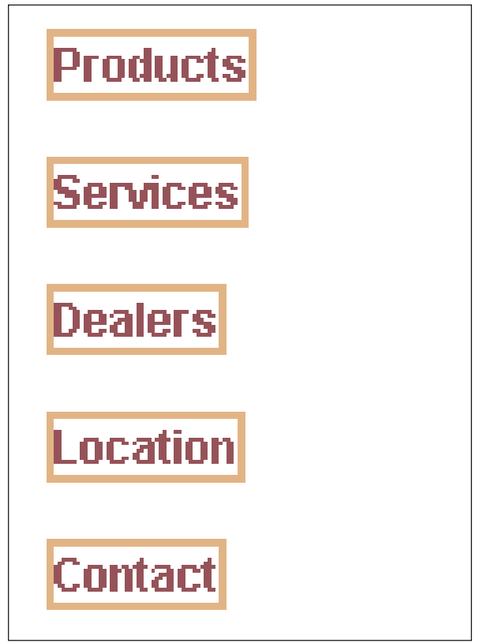
- **Don't use overly small text for CSS menus.** Text at small sizes will likely fall apart on a Mac monitor. If you must use very small text with CSS, make sure you test on a Mac. Better yet, if the text must be very small, switch to graphic menus, using specially designed fonts for small sizes, often called pixel fonts (6.40). These fonts look better and display consistently on both Mac and PC systems.
- **Watch your spelling and typos.** Of course, this tip is not just for CSS, but it seems that CSS menus have errors more often (6.41). My guess is that the text is so easy to create that the same attention does not go into making sure all is perfect. With graphic menu items, you must export and place the files, providing more working opportunity to catch any slips. With CSS text, you just type and run, perhaps never paying close attention to that word again. Be warned!
- **Leave enough white space.** Again, white space is important for all design, but I have seen so many CSS menus with decorative styling that leave the text gasping for air (6.42). Use padding attributes to ensure that your borders leave text room to breathe. Margin attributes leave breathing room between the edge of your CSS styling and other page elements (6.43). Even if you're not using border styling, you may need to use padding or margins to position your text.



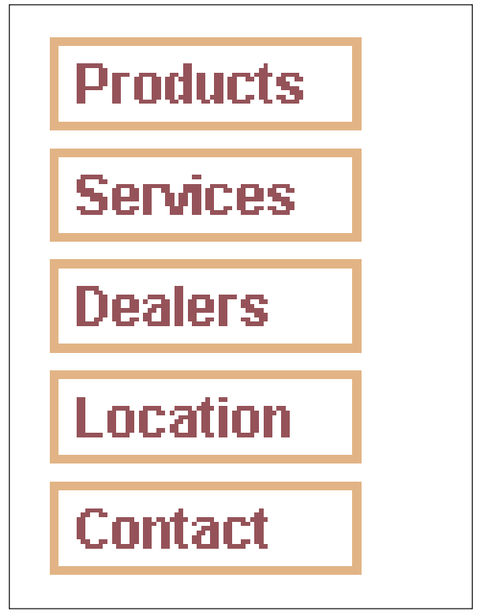
● 6.40

● 6.41

- **Don't leave too much space between related text items.** Once again, too much is not good. Menu text items are related, and they are only effective if the visitor sees them as a unit. This is not usually a problem with horizontal menus, as you must work hard with HTML text to include extra spacing. However, with vertical text, seeing the menu items hanging out all by themselves is very



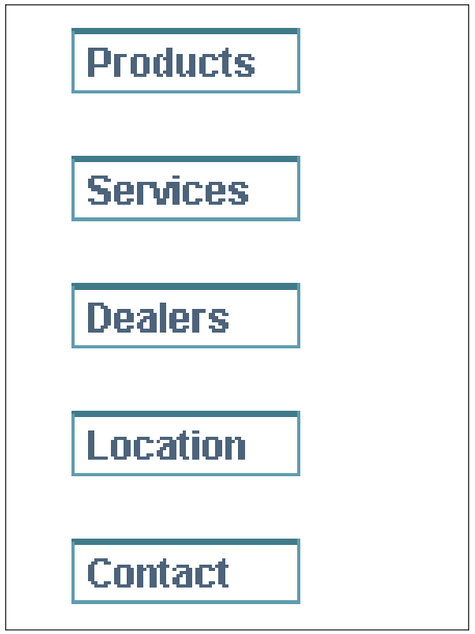
● 6.42



● 6.43

common (6.44). Unless you set the margin values for your style, it is rare that you should use hard breaks between vertical menu items. Use the
 tag, and adjust line spacing for the style if you find the items are too close (6.45).

- **Make sure text doesn't disappear with any link states.**
  I'll bet you have seen pages where menu items disappear. Pay attention to all states of your link styling. I've been caught myself when I have designed a menu, and then changed the background color where the menu is located. Make sure that you test all states of your menu whenever you make a change. White text does not show on a light yellow background, even with the power of CSS working for you (6.46).
- **When using border attributes, understand where they are going.**
  Borders and lines can be tricky to work with unless you really grasp the idea of where they will be applied. Consider all possibilities for your design when you create a style (6.47). Applying a border effect to a paragraph or a table cell delivers totally different results with the same CSS styling. I've seen cases where the designer obviously didn't get this idea, and the page resulted in quite a mess. My guess is that he or she set up the style, then made changes, then tried to fix the problem, without understanding where the problem originated. Trust me when I say that the time spent understanding exactly where CSS borders go is time well spent (and best accomplished by playing).
- **Don't use italic styling on your menu items.**
  Just don't.



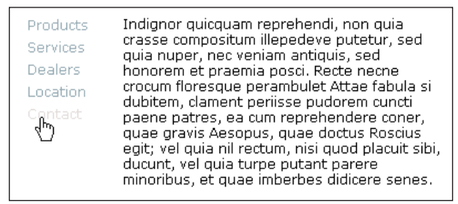
6.44

6.45

6.46

- **Make sure your menu stands apart but lets the content be the star.**
  With CSS styling, you often use the same font for menus as you do for content text, and the menu can be so demure it is not clearly available (6.48). Make sure that you create a separate area

6.47



6.48

One Tiny Error = One Broken Page

Recently, I saw one small CSS error that initially made me think the page was disintegrating. It was only my curiosity that made me stick around to find out what caused the problem.

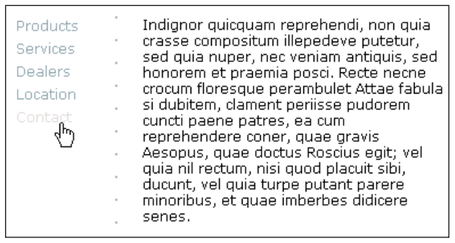
The problem page had a normal weight font specified as the `a:link` style, and a bold specified for the `a:hover` style. That can make you a little seasick at the best of times, as bold font characters take more space than the same character with a normal weight font, and the text moves.

In one spot on the bad page, the bold type increased the amount of space required for a long word in a link, and the link jumped to the next line whenever the mouse passed over. Imagine what that did to the display on the screen as a long link was bumped to the next line, then back up when the mouse was removed, down again when the mouse passed over. All the content on the page following this spot moved as the new line was created and returned to normal. Small error. Disastrous results.

for your menus. You can accomplish this task by using bold for your menus, white space, background color or images for menu areas, or dividing lines or borders on your menu. Even a simple, almost invisible dotted line does the job (6.49). But also be aware that the menu should be obvious, but allow visitors to concentrate on the text. Screaming red backgrounds with bold, white text will win over demure content every time (6.50). Keep the balance.

Obviously, the designer of this page did not test with the resolution I was using. You must always test your pages with popular resolutions, in all current browsers. Unless you're certain your users will be using current browsers, I do urge you to test your page with a Netscape 4 version browser as well. When it comes to CSS, Netscape 4 version browsers require special attention. Great design is lost if your visitors cannot see your pages correctly.

● 6.50



● 6.49