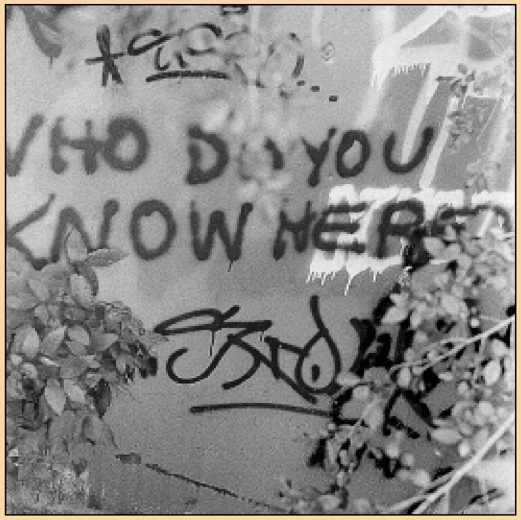


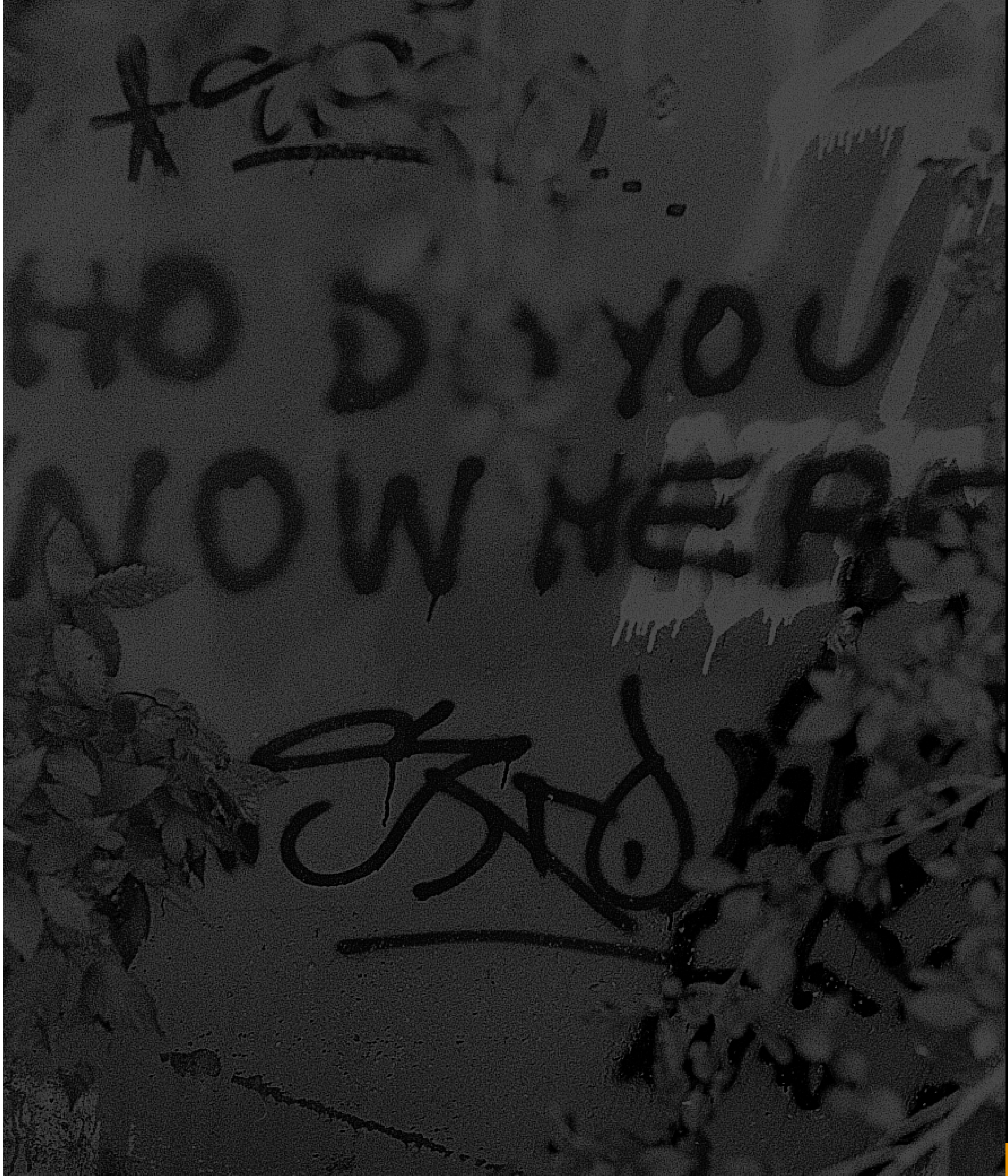
“For a list of all the ways technology has

failed to improve the quality of life,

please press three.”

—ALICE KAHN

MANAGING A WEEKLY PUBLICATION

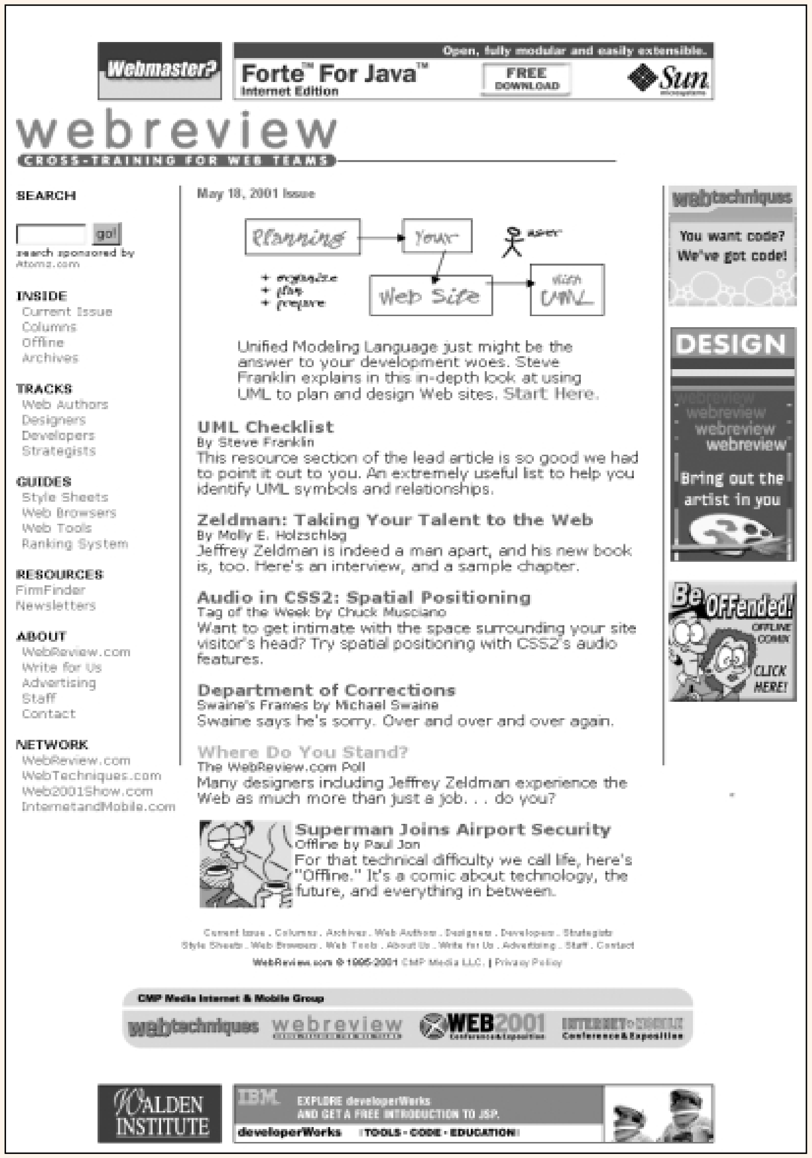


In Chapter 2, "Updating a Daily News Site," you put together a daily newspaper. You learned how to fashion columns and create readable style. But more magic is needed to manage the design and technology of publications—especially large publications and information sites. In this chapter, you explore some of these issues through `WebReview.com`, an award-winning resource site for web developers and designers.

Project 2

Managing a Weekly Publication

By Molly E. Holzschlag


Webmaster?

Open, fully modular and easily extensible.
Forte™ For Java™
Internet Edition
FREE DOWNLOAD
Sun

webreview
CROSS-TRAINING FOR WEB TEAMS

SEARCH
go!
search sponsored by Atomz.com

INSIDE
Current Issue
Columns
Offline
Archives

TRACKS
Web Authors
Designers
Developers
Strategists

GUIDES
Style Sheets
Web Browsers
Web Tools
Ranking System

RESOURCES
FirmFinder
Newsletters

ABOUT
WebReview.com
Write for Us
Advertising
Staff
Contact

NETWORK
WebReview.com
WebTechniques.com
Web2001Show.com
InternetandMobile.com

May 18, 2001 Issue

Unified Modeling Language just might be the answer to your development woes. Steve Franklin explains in this in-depth look at using UML to plan and design Web sites. Start Here.

UML Checklist
By Steve Franklin
This resource section of the lead article is so good we had to point it out to you. An extremely useful list to help you identify UML symbols and relationships.

Zeldman: Taking Your Talent to the Web
By Molly E. Holzschlag
Jeffrey Zeldman is indeed a man apart, and his new book is, too. Here's an interview, and a sample chapter.

Audio in CSS2: Spatial Positioning
Tag of the Week by Chuck Musciano
Want to get intimate with the space surrounding your site visitor's head? Try spatial positioning with CSS2's audio features.

Department of Corrections
Swaine's Frames by Michael Swaine
Swaine says he's sorry. Over and over and over again.

Where Do You Stand?
The WebReview.com Poll
Many designers including Jeffrey Zeldman experience the Web as much more than just a job. . . do you?

Superman Joins Airport Security
Offline by Paul Jon
For that technical difficulty we call life, here's "Offline." It's a comic about technology, the future, and everything in between.

Current Issue . Columns . Archives . Web Authors . Designers . Developers . Strategists
Style Sheets . Web Browsers . Web Tools . About Us . Write for Us . Advertising . Staff . Contact
WebReview.com © 1995-2001 CMP Media LLC. | Privacy Policy

CMP Media Internet & Mobile Group
webtechniques webreview WEB2001 Conference&Exposition Internet & Mobile Conference&Exposition

webtechniques
You want code? We've got code!

DESIGN
webreview webreview webreview webreview
Bring out the artist in you

Be OFFended! OFFLINE COMIX CLICK HERE!

WALDEN INSTITUTE

IBM. EXPLORE developerWorks AND GET A FREE INTRODUCTION TO JSP.
developerWorks TOOLS · CODE · EDUCATION

Project Snapshot

The problem: Managing larger, regularly updated, content-rich sites.

This chapter is for anyone who is looking for solutions to manage navigation, presentation, and effective markup of large, content-rich sites.

Technical Specs

The following are the technical specifications that you need to manage a weekly publication:

- **Markup used**—XHTML 1.0 (You can also use HTML 4 if you prefer.)
- ***Document type definition* (DTD) used**—Transitional.

From a markup perspective, I used XHTML 1.0 transitional when I developed the site. My rationale for doing so was two-fold. First, a site that's predominantly made up of text conceivably can be simplified to ensure crisp separation of document formatting and presentation, even using complicated tables to ensure the layout remains intact across platforms and browsers. The rationale was a good one, but during the production process, I learned my choice was compromised by the amount of ads I had to design into the site. The more complex a layout becomes, the more difficult it becomes to separate document formatting from presentation.

The second reason I selected XHTML 1.0 was that I wanted to make a statement that showed that XML—in the form of XHTML—could easily be used on the web. I learned some interesting lessons from making the choice to use XHTML, especially in terms of JavaScript, as you see later in this chapter.

Because the exercises in this chapter are comparative, you can choose to use HTML 4 or XHTML 1.0. My only recommendation is that you stick to the transitional DTD to ensure utmost flexibility in a design that's still primarily accessed through the web. If you're shifting toward a publication that appeals to users accessing with wireless and other devices, consider moving to strict markup instead.

Here are the additional technologies or skills that you need:

- Familiarity with HTML.
- Familiarity with a text editor, such as Notepad, SimpleText, or a favorite HTML editor.

Browser considerations: Cross-browser compatible site.

You must use an external style sheet and an embedded style sheet.

Here's how you should structure your site. In the case of `WebReview.com`, the site uses a hierarchical structure. Top-level pages are used daily. The second tier contains information by year, and the third level contains the individual issues and their dependents by date.

How a real-world site is structured will be determined greatly by individual needs. So if you have a publication site that is updated monthly, you'll have different archival management needs. What's more, you might already be working on a site that has legacy problems with structure and have to make do. See the sidebar, "Structure Inspiration" for some ideas on how to solve structure problems.

Structure Inspiration

A primary concern with any regularly published, content-rich site is how to effectively manage a site's infrastructure. There's no definite answer here—much depends on your publication's specific needs. However, it's a good idea to grab a pen and paper, a great big white board or charting software, such as Visio, and work out the site's physical structure before you attempt to write any of the markup.

A strong physical site structure helps ensure that your markup is more consistent. Where things such as directories for images, media, and style sheets, archived information, and so on, are placed will immediately be reflected in the way you write your internal links. This, in turn, reflects on the markup and the speed at which you can troubleshoot problematic documents.

Unfortunately, many of you will walk into situations that you can't change; pre-existing problems must be dealt with as best as possible. Streamline wherever you can. Cleaner markup that relies on style sheets makes so much sense when it's put into this perspective. Imagine how easy it would be to update a site simply by changing its style sheet and not having to rebuild it from the ground up? Now that's practical.

How It Works

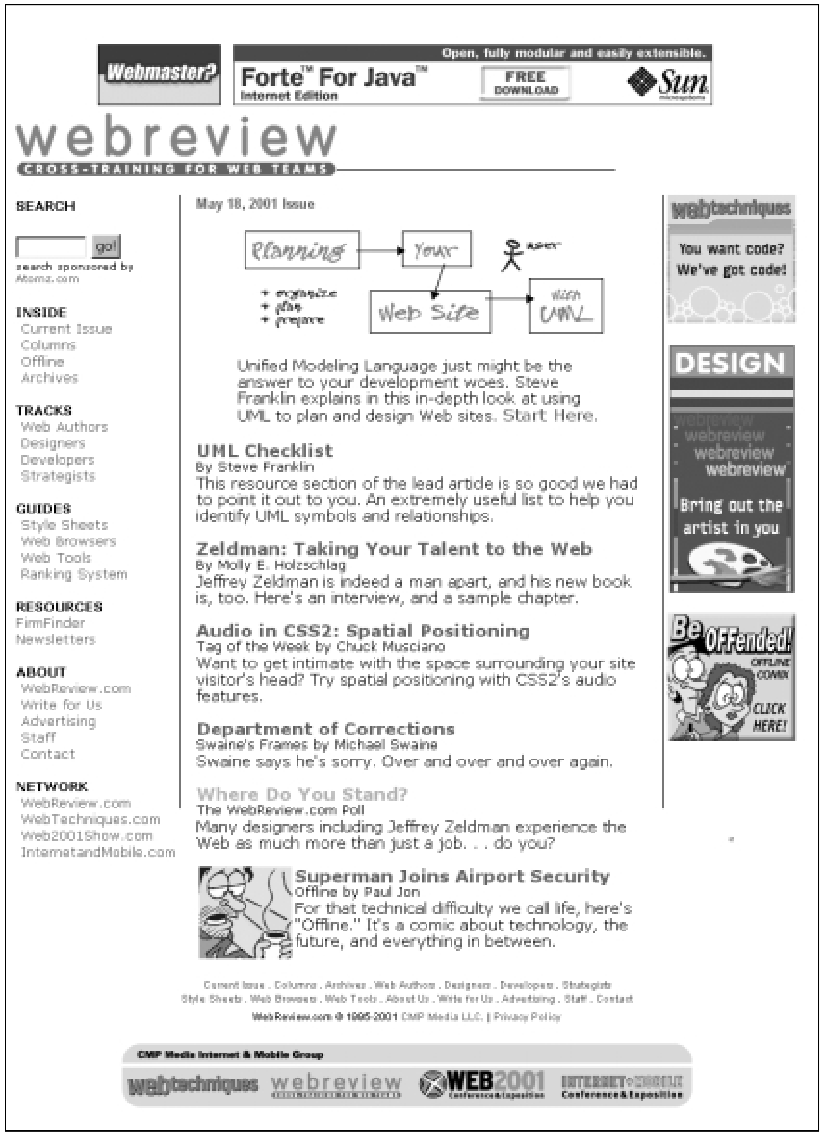
It's just coincidental that I was working on this book when `WebReview.com` was being restructured. This restructuring meant having to take a hard look at markup and structure. Some of the detailed problems I needed to tackle included the following:

- **Make the most of available screen space**—The old site design was a fixed-width table, centered on the page. Three columns were then within that table. The look was a bit old-fashioned and cramped. The solution? Use dynamic tables.
- **Solve problems with site structure and navigation**—`WebReview.com` was created in 1995, and as it grew, it became like a ramshackle house—rooms

upon rooms with some rooms staring to fall down. The site really needed a navigation and structure update. To solve this problem, I reorganized the site structure and made the navigation global. I also put the navigation into *Server Side Includes* (SSIs).

- **Manage consistency from page to page more effectively**—Headers were inconsistent in style and color; sometimes, graphics were used instead of text headers; and navigation was incredibly problematic due to the growth of sections. No consistent navigation existed on the site. Style sheets came to the rescue in terms of achieving consistency.
- **Incorporate advertisements effectively into the interface and solve scripting problems**—A major challenge with interface design on a commercial site is how to place numerous ads on a page and still keep the content in focus. What's more, ad delivery often comes from external vendors who serve up their own brand of markup. Correcting problem markup and escaping characters properly when using XHTML helped to successfully address these problems.

Managing issues such as these is standard fare for the professional web designer these days. You must analyze an existing site and make complex decisions that will be matched by equally complex technology solutions.



SEARCH

search sponsored by Atomz.com

INSIDE
Current Issue
Columns
Offline
Archives

TRACKS
Web Authors
Designers
Developers
Strategists

GUIDES
Style Sheets
Web Browsers
Web Tools
Ranking System

RESOURCES
FirmFinder
Newsletters

ABOUT
WebReview.com
Write for Us
Advertising
Staff
Contact

NETWORK
WebReview.com
WebTechniques.com
Web2001Show.com
InternetandMobile.com

May 18, 2001 Issue

Unified Modeling Language just might be the answer to your development woes. Steve Franklin explains in this in-depth look at using UML to plan and design Web sites. Start Here.

UML Checklist
By Steve Franklin
This resource section of the lead article is so good we had to point it out to you. An extremely useful list to help you identify UML symbols and relationships.

Zeldman: Taking Your Talent to the Web
By Molly E. Holzschlag
Jeffrey Zeldman is indeed a man apart, and his new book is, too. Here's an interview, and a sample chapter.

Audio in CSS2: Spatial Positioning
Tag of the Week by Chuck Musciano
Want to get intimate with the space surrounding your site visitor's head? Try spatial positioning with CSS2's audio features.

Department of Corrections
Swaine's Frames by Michael Swaine
Swaine says he's sorry. Over and over and over again.

Where Do You Stand?
The WebReview.com Poll
Many designers including Jeffrey Zeldman experience the Web as much more than just a job. . . do you?

Superman Joins Airport Security
Offline by Paul Jon
For that technical difficulty we call life, here's "Offline." It's a comic about technology, the future, and everything in between.

Current Issue . Columns . Archives . Web Authors . Designers . Developers . Strategists
Style Sheets . Web Browsers . Web Tools . About Us . Write for Us . Advertising . Staff . Contact
WebReview.com © 1995-2001 CMP Media LLC. | Privacy Policy

Use a mixed fixed width and dynamic table layout to manage navigation, content, and multiple ads. The `WebReview.com` home page juggles

Managing Layout

For this example, you'll use a mixed fixed width and dynamic table layout in XHTML 1.0 transitional that effectively manages navigation, content, and graphics along with a total of seven ads on the page.

The virtues of the fixed and dynamic layout combinations are numerous, including cross-resolution compatibility, visual control over specific areas of the document, and dynamic flow of specific areas.

Download the code samples related to this chapter from this book's web site and follow along. You will want to have the `master_document.html` and the `table_template.html` documents available.

To achieve a combination of fixed and dynamic layout tables, follow these steps:

1 Open `table_template.html` in your editor.

```
<table>
<tr>

  <td>

  </td>

  <td>

  </td>

  <td>

  </td>

</tr>

</table>
```

2 In the table tag itself, insert the following border,

It should read

```
<table border="0" width="100%" cellpadding="0"
```

Table Cells and Single-Pixel GIFs

Fixed-table cells require either a graphic of the cell's exact width (such as a header), or a single-pixel GIF shim set to the width of the table to ensure the cell does not collapse.

Graphic shims have been a long-debated issue. The idea of a graphic shim goes right to the heart of the argument that says document formatting and presentation should be separated. If browsers conform to current CSS2 standards, and if all people had browsers that conformed, we could toss the idea of a shim out the window and rely on style-sheet positioning to control presentation instead.

Until then, the shim is the only invisible method to ensure that a fixed cell does not collapse.

width, and padding attributes:

The width is dynamic, set to 100 percent. This ensures that the entire document dynamically adjusts to the available screen space. Setting the defaults of padding and spacing to `0` eliminates any problems with spliced graphics within the table by ensuring that no unwanted gaps appear.

3 In the first table cell, add the following attributes:

```
<td
width="200">
```

You'll notice that this cell has a numeric value, like its `width` attribute. This fixes the size.

4 In the second table cell, add the following attributes:

```
<td
width="100%">
```

In this case, the cell is dynamic—its content stretches to the available resolution. Notice that the value is 100 percent. This means that the content portions of the site will be dynamic to the most-available space.

5 In the third table cell, add the following attributes:

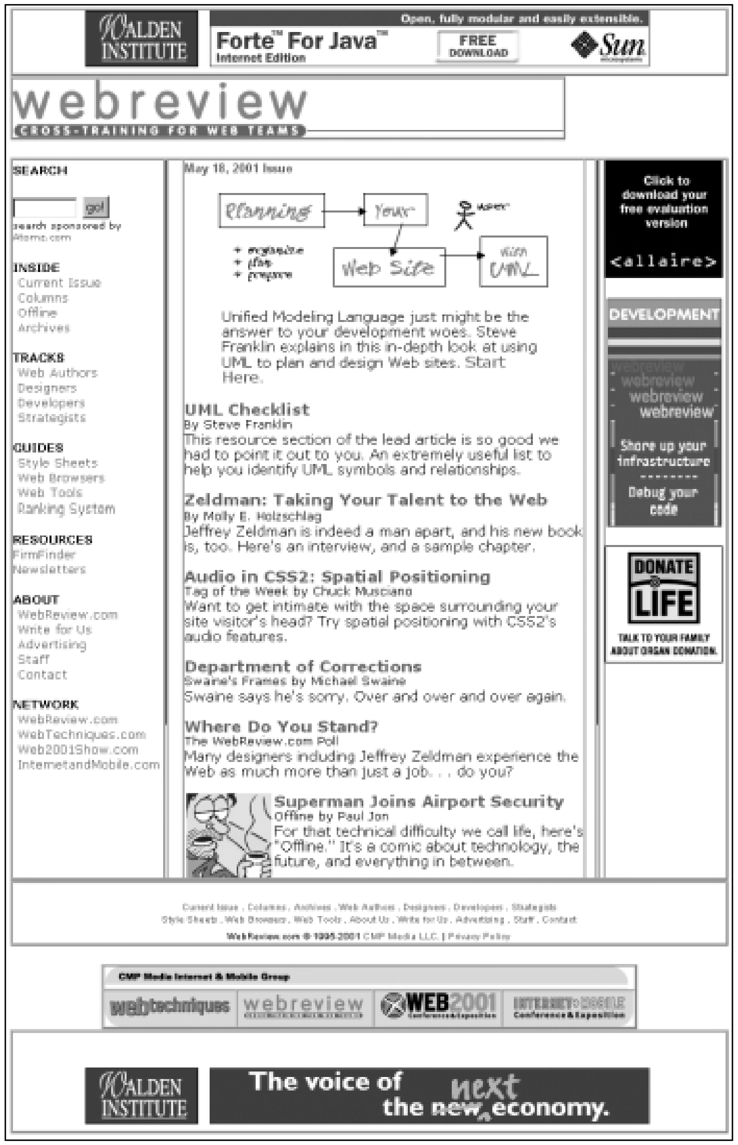
```
<td
```

This cell is fixed. Again, a graphic or shim set to the same width of the cell will be helpful to ensure against collapsing cells.

6 Insert this table into an HTML or XHMTL document template of your own and add content to see how it works.

You can also modify which cells are fixed and which are dynamic—it just depends on on your needs. If you examine `master_document.html`, you see that the main table has additional cells. Play around with the number and type of cells and see what might work the best magic for you. In the figure, I turned on the table borders so you can see the table structure of `WebReview.com`'s home page.

Tip: Here's a trick that many web designers use to assist them during the development of page layouts: Set table borders to a value of 1. This way, the grid is clearly visible and adjustments can be easily made. Switch the borders back to a value of 0 to see the end result.



May 18, 2001 Issue

Unified Modeling Language just might be the answer to your development woes. Steve Franklin explains in this in-depth look at using UML to plan and design Web sites. Start Here.

UML Checklist
By Steve Franklin
This resource section of the lead article is so good we had to point it out to you. An extremely useful list to help you identify UML symbols and relationships.

Zeldman: Taking Your Talent to the Web
By Molly E. Holzschlag
Jeffrey Zeldman is indeed a man apart, and his new book is, too. Here's an interview, and a sample chapter.

Audio in CSS2: Spatial Positioning
Tag of the Week by Chuck Musciano
Want to get intimate with the space surrounding your site visitor's head? Try spatial positioning with CSS2's audio features.

Department of Corrections
Swaine's Frames by Michael Swaine
Swaine says he's sorry. Over and over and over again.

Where Do You Stand?
The WebReview.com Poll
Many designers including Jeffrey Zeldman experience the Web as much more than just a job. . . do you?

Superman Joins Airport Security
Offline by Paul Jon
For that technical difficulty we call life, here's "Offline." It's a comic about technology, the future, and everything in between.

Current Issue . Columns . Archives . Web Authors . Designers . Developers . Strategists
Style Sheets . Web Browsers . Web Tools . About Us . Write for Us . Advertising . Staff . Contact
WebReview.com © 1995-2001 CMP Media LLC. | Privacy Policy

The `WebReview.com` home page with table borders on. This way, you can see each individual cell, the entire table structure, and how the various text and image elements fit into each cell.

Adding Look and Feel by Using *Style*

With a developed structure in place, you can now move on to creating a look and feel for your site. At WebReview.com, several choices were made regarding the global style for all documents. I wanted WebReview.com to have good onscreen readability (characteristics for good onscreen readability are described in Chapter 2).

The bulk of the site uses high contrast: black text on a white background. I also chose sans-serif fonts. I made the choice to use points when sizing fonts (a controversial choice, but I prefer the fixed rather than dynamic results), and I added a variety of features needed for the color palette, link behaviors, and specialty text, such as block quotes and code samples.

To create a style sheet that's suitable for highly readable documents, follow these steps:

```
p {
        font-size : 11pt;
        font-family : verdana, helvetica, arial, sans-serif;
        font-weight : normal;
        font-style : normal;
        color : #000000;
        line-height : 12pt;
        text-decoration : none;
}
```

1 Download and open the master_style.css document to see WebReview.com's sheet for comparison.

2 Begin a new, blank document in your text editor;

Listing 3.1 Header and Link Settings for WebReview.com

```
h1 {
        font-size : 15pt;
        font-family : verdana, helvetica, arial, sans-serif;
        font-weight : bold;
        font-style : normal;
        color : #000000;
        text-decoration : none;
}
```

continues

continued

name it `style.css`.

3 In the document, add the following:

4 Add the header and link information.

Listing 3.1 shows the header and link information for `WebReview.com`.

```
h2  {
        font-size : 14pt;
        font-family : verdana, helvetica, arial, sans-serif;
        font-weight : bold;
        font-style : normal;
        color : #000000;
        text-decoration : none;
}


h3  {
        font-size : 13pt;
        font-family : verdana, helvetica, arial, sans-serif;
        font-weight : bold;
        font-style : normal;
        color : #000000;
        text-decoration : none;
}


h4  {
        font-size : 11pt;
        font-family : verdana, helvetica, arial, sans-serif;
        font-weight : bold;
        font-style : normal;
        color : #000000;
        line-height : 11pt;
        text-decoration : none;
}


a {
        color: #336699;
        text-decoration : none;
}


a:visited  {
        color: #336699;
        text-decoration : none;
}


a:active  {
        color: #CC9933;
        text-decoration : none;
}


a:hover  {
        color : #FF9933;
}
```

5 To ensure that your list items pick up style in as many browsers as possible, be sure to apply the style to the list types, as follows:

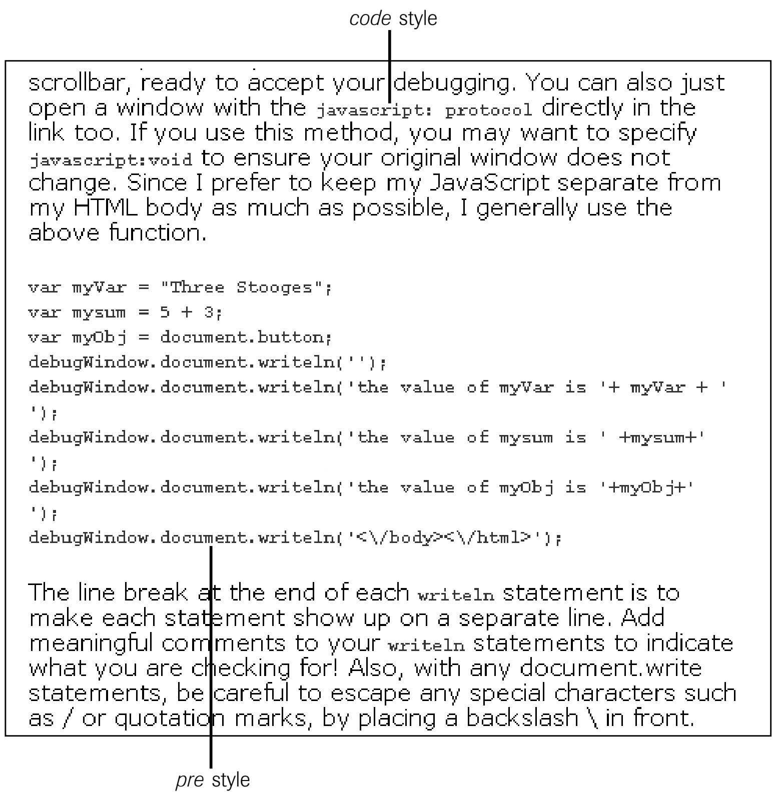
Caution: Netscape 4.*x* doesn't properly read styles assigned to the list item tag (li). So it's important to apply those styles to the list type (ul, ol, or dl), because Netscape 4.*x* browsers will then apply the styles more consistently.

```
ul {
        font-size : 11pt;
        font-family : verdana, helvetica, arial, sans-serif;
        font-weight : normal;
        font-style : normal;
        color : #000000;
        text-decoration : none;
}


ol {
        font-size : 11pt;
        font-family : verdana, helvetica, arial, sans-serif;
        font-weight : normal;
        font-style : normal;
        color : #000000;
        text-decoration : none;
}



dl {
        font-size : 11pt;
        font-family : verdana, helvetica, arial, sans-serif;
        font-weight : normal;
        font-style : normal;
        color : #660000;
        line-height : 12pt;
        text-decoration : none;
}
```

6 To add inline code, preformatted text, and block quote text, use the following code:

Remember that any time you require special text features, you can add style rules as needed. The figure shows how preformatted text (used for a number of reasons in HTML and XHTML, but here for denoting code) is now styled on any page linking to this sheet.

```
pre {
        font-size: 9pt;
        font-family: courier, courier new,
        ➥monospace;
        font-weight: normal;
        font-style: normal;
        color: #660000;
        line-height: 12pt;
        text-decoration: none;
}

code {
        font-size: 9pt;
        font-family: courier, courier new,
        ➥monospace;
        font-weight: normal;
        font-style: normal;
        color: #660000;
        line-height: 12pt;
        text-decoration: none;
}

blockquote  {
        font-size : 11pt;
        font-family : verdana, helvetica,
        ➥arial, sans-serif;
        font-weight : normal;
        font-style : normal;
        color : #000000;
        line-height : 12pt;
        text-decoration : none;
}
```



scrollbar, ready to accept your debugging. You can also just open a window with the `javascript: protocol` directly in the link too. If you use this method, you may want to specify `javascript:void` to ensure your original window does not change. Since I prefer to keep my JavaScript separate from my HTML body as much as possible, I generally use the above function.

```
var myVar = "Three Stooges";
var mysum = 5 + 3;
var myObj = document.button;
debugWindow.document.writeln('');
debugWindow.document.writeln('the value of myVar is '+ myVar + '
');
debugWindow.document.writeln('the value of mysum is ' +mysum+'
');
debugWindow.document.writeln('the value of myObj is '+myObj+'
');
debugWindow.document.writeln('<\/body><\/html>');
```

The line break at the end of each `writeln` statement is to make each statement show up on a separate line. Add meaningful comments to your `writeln` statements to indicate what you are checking for! Also, with any document.write statements, be careful to escape any special characters such as / or quotation marks, by placing a backslash \ in front.

Within the browser you'll note that code and preformatted text styles now appear in monospace font.

7 Save the file. Link any documents you'd like to have the style sheet influence by adding the following markup to the head section of that document:

```
<link href="style.css" type="text/css" rel="stylesheet" />
```

Tip: You'll want to adjust the href attribute to the proper location within your own directories. When I have only one or two style sheets for a web site, I tend to put them on the top level. When I have more than two or three style sheets, I keep them in a separate style directory.

Tip: To see if your style passes muster, use the W3C's CSS Validator service, available at jigsaw.w3.org/css-validator/.

Headers and Links, Oh My!

Briefly mentioned in Chapter 2, there is an historical basis for why headers in HTML are formulated the way that they are. In publishing, the need to designate top-level content from supporting subjects is important. Look, for example, at this chapter. It uses a system of headers that denote this process. If a topic is primary, it received a certain type of header, with specific styles attached. If it is secondary, that header is likely to be smaller and differently styled, and so on.

As you know, h1 headers, by default, will be larger than an h6 header. It's my recommendation that you use headers consistent with the use that was originally intended for them. h1 headers should be larger and perhaps bolder than any lower-level headers you use. You can choose different colors for header levels if you like; it just depends on the aesthetic that you're trying to achieve. What you do not want to do is randomly assign styles to header levels. If you do this, you won't achieve consistent results.

When it comes to links, no hard and fast rules exist. In fact, there's some debate as to the appropriateness of coloring links at all and simply allowing them to remain default colors—blue for standard, purple for visited, and so forth. This argument is based on ideas set forth by various usability specialists, who feel that the consistency across all sites achieved by this is a benefit.

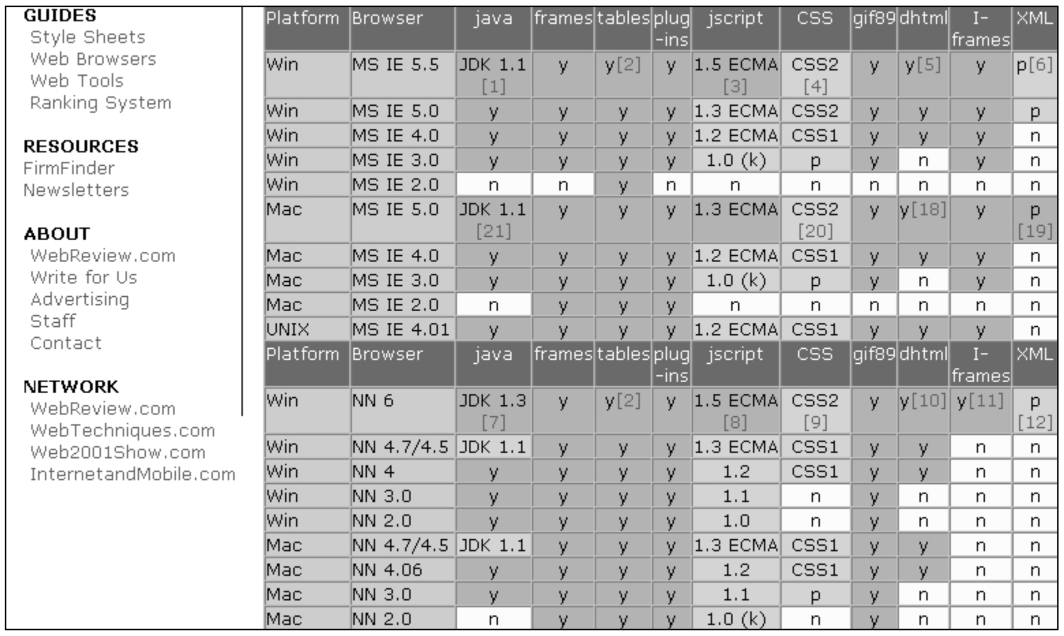
I don't agree. As a designer, matching link colors to the esthetic of your design is important. I will say, however, that keeping your link colors consistent throughout your site is imperative for a professional-looking site.

Controlling Table Style

In certain instances, such as when information must be accessible through a visible grid system, `WebReview.com` uses tables for the display of information. In these instances, the requirements for detailed style become necessary. For example, embedded style is an excellent style method choice when you have only a few pages that require the particular style applied to them.

To embed a style sheet, follow these steps:

1 Open an HTML or XHTML document. You can use `master_document.html` if you like.



GUIDES
Style Sheets
Web Browsers
Web Tools
Ranking System

RESOURCES
FirmFinder
Newsletters

ABOUT
WebReview.com
Write for Us
Advertising
Staff
Contact

NETWORK
WebReview.com
WebTechniques.com
Web2001Show.com
InternetandMobile.com

Platform	Browser	java	frames	tables	plug -ins	jscript	CSS	gif89	dhtml	I-frames	XML
Win	MS IE 5.5	JDK 1.1 [1]	y	y[2]	y	1.5 ECMA [3]	CSS2 [4]	y	y[5]	y	p[6]
Win	MS IE 5.0	y	y	y	y	1.3 ECMA	CSS2	y	y	y	p
Win	MS IE 4.0	y	y	y	y	1.2 ECMA	CSS1	y	y	y	n
Win	MS IE 3.0	y	y	y	y	1.0 (k)	p	y	n	y	n
Win	MS IE 2.0	n	n	y	n	n	n	n	n	n	n
Mac	MS IE 5.0	JDK 1.1 [21]	y	y	y	1.3 ECMA	CSS2 [20]	y	y[18]	y	p [19]
Mac	MS IE 4.0	y	y	y	y	1.2 ECMA	CSS1	y	y	y	n
Mac	MS IE 3.0	y	y	y	y	1.0 (k)	p	y	n	y	n
Mac	MS IE 2.0	n	y	y	y	n	n	n	n	n	n
UNIX	MS IE 4.01	y	y	y	y	1.2 ECMA	CSS1	y	y	y	n
Platform	Browser	java	frames	tables	plug -ins	jscript	CSS	gif89	dhtml	I-frames	XML
Win	NN 6	JDK 1.3 [7]	y	y[2]	y	1.5 ECMA [8]	CSS2 [9]	y	y[10]	y[11]	p [12]
Win	NN 4.7/4.5	JDK 1.1	y	y	y	1.3 ECMA	CSS1	y	y	n	n
Win	NN 4	y	y	y	y	1.2	CSS1	y	y	n	n
Win	NN 3.0	y	y	y	y	1.1	n	y	n	n	n
Win	NN 2.0	y	y	y	y	1.0	n	y	n	n	n
Mac	NN 4.7/4.5	JDK 1.1	y	y	y	1.3 ECMA	CSS1	y	y	n	n
Mac	NN 4.06	y	y	y	y	1.2	CSS1	y	y	n	n
Mac	NN 3.0	y	y	y	y	1.1	p	y	n	n	n
Mac	NN 2.0	n	y	y	y	1.0 (k)	n	y	n	n	n

Use specialty tables, such as these on `WebReview.com`, when you need detailed style.

2 In the `head` portion of the document, insert the opening and closing `style` tags:

```
<style type="text/css">

</style>
```

3 Insert the style rules:

```
td.bighead {
        vertical-align: top;
        background-color: #666699;
        font-family: verdana, helvetica, arial, sans-serif;
        font-size: 1em;
        color: #FFFFFF;
```

continues

continued

```
td.head {
        border-color: #333366;
        border-style: solid;
        vertical-align: top;
        background-color: #666699;
        font-family: verdana, helvetica, arial, sans-serif;
        font-size: .8em;
        color: #FFFFFF;
}

td.side {
        border-color: #333366;
        border-style: solid;
        vertical-align: top;
        text-align: center;
        background-color: #DDDDF0;
        font-family: verdana, helvetica, arial, sans-serif;
        font-size: .8em;
        color: #000000;
}

td.yes {
        vertical-align: top;
        background-color: #ff9900;
        font-family: verdana, helvetica, arial, sans-serif;
        font-size: .8em;
        color: #000000;
}
```

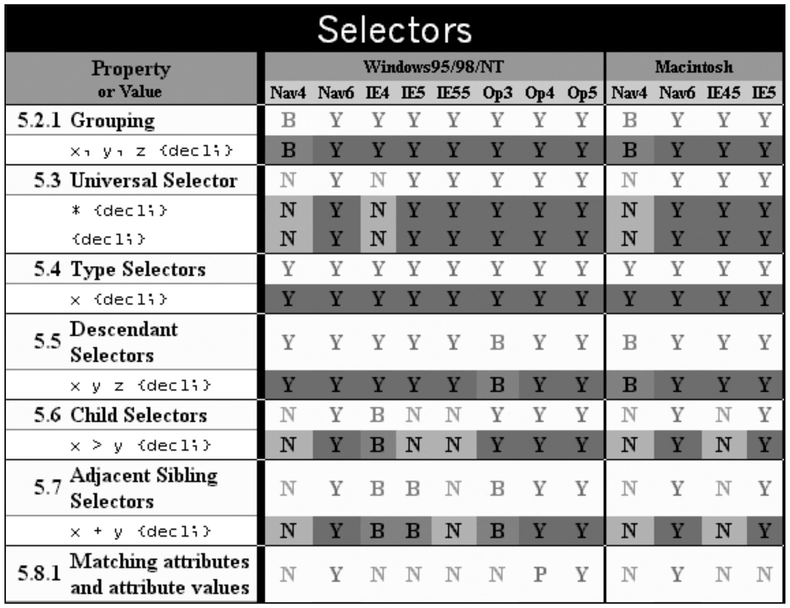
```
td.no {
        vertical-align: top;
        background-color: #ffffcc;
        font-family: verdana, helvetica, arial, sans-serif;
        font-size: .8em;
        color: #000000;
}

td.part {
        vertical-align: top;
        background-color: #ffcc33;
        font-family: verdana, helvetica, arial, sans-serif;
        font-size: .8em;
        color: #000000;
}

td.cont {
        border-color: #333366;
        border-style: solid;
        vertical-align: top;
        background-color: #ffffff;
        font-family: verdana, helvetica, arial, sans-serif;
        font-size: .8em;
        color: #000000;
}
```

Note how this sample controls table style. You can have as many style rules as you like. Just be sure not to overburden the document. It's helpful to create a consistent formatting style for your *Cascading Style Sheets* (CSS). Other techniques, such as grouping (which is discussed in Chapter 1 "About Web Markup: XML, HTML, XHTML," can also be used to streamline your documents.

Tip: If the style rules become too long, you might consider creating an external style sheet.

CSS Printing Features

Style sheets offer you the ability to hide certain portions of a page. This, in turn, makes it easy to be more printer-friendly—especially in instances like `WebReview.com`, which uses dynamic, rather than fixed, tables.



Selectors

Property or Value	Windows95/98/NT Nav4	Nav6	IE4	IE5	IE55	Op3	Op4	Op5	Macintosh Nav4	Nav6	IE45	IE5
5.2.1 Grouping	B	Y	Y	Y	Y	Y	Y	Y	B	Y	Y	Y
`x, y, z {decl;}`	B	Y	Y	Y	Y	Y	Y	Y	B	Y	Y	Y
5.3 Universal Selector	N	Y	N	Y	Y	Y	Y	Y	N	Y	Y	Y
`* {decl;}`	N	Y	N	Y	Y	Y	Y	Y	N	Y	Y	Y
`{decl;}`	N	Y	N	Y	Y	Y	Y	Y	N	Y	Y	Y
5.4 Type Selectors	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y
`x {decl;}`	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y
5.5 Descendant Selectors	Y	Y	Y	Y	Y	B	Y	Y	B	Y	Y	Y
`x y z {decl;}`	Y	Y	Y	Y	Y	B	Y	Y	B	Y	Y	Y
5.6 Child Selectors	N	Y	B	N	N	Y	Y	Y	N	Y	N	Y
`x > y {decl;}`	N	Y	B	N	N	Y	Y	Y	N	Y	N	Y
5.7 Adjacent Sibling Selectors	N	Y	B	B	N	B	Y	Y	N	Y	N	Y
`x + y {decl;}`	N	Y	B	B	N	B	Y	Y	N	Y	N	Y
5.8.1 Matching attributes and attribute values	N	Y	N	N	N	N	P	Y	N	Y	N	N

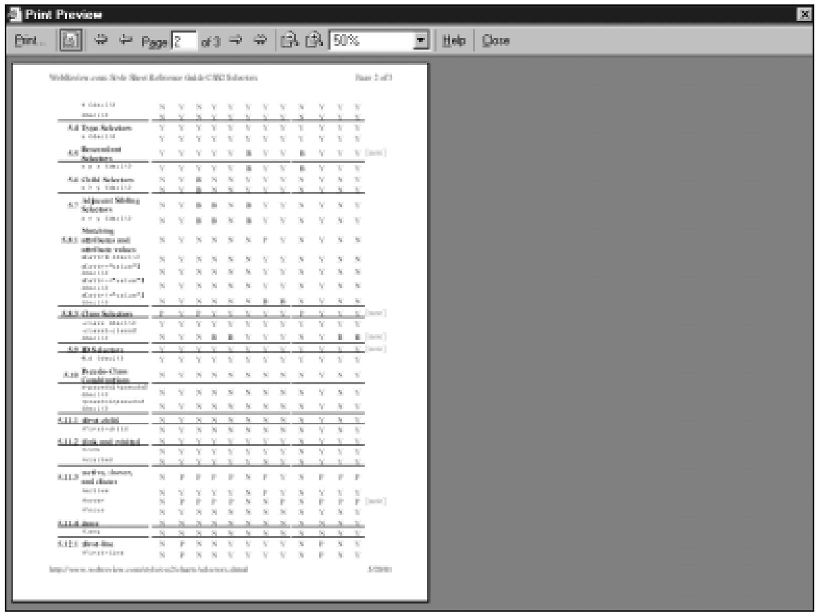
Dynamic tables can pose a printing problem.

The style sheet guide at `WebReview.com`—written by the *World Wide Web Consortium* (W3C) style sheet working group member Eric Meyer—uses the `display` property along with classes to ensure that certain parts of the page won't print.

To recreate this feature, follow these steps:

1. Open an existing HTML or XHTML document along with the `print_style.css` document found on the web site.

Note: For more information on class and grouping, see Chapter 1.



Use the `display` property to hide some items for easier printing.

2 In the print_style.css document, note the class selectors for elements that are to be prevented from printing:

```
table.top-ads, table.bottom-ads, table.footer, td.left-nav *,
td.right-ads *, p.style-nav {display: none}
```

3 Add the class attribute to the tags in question.

Here's a sample from the Style Guide at WebReview.com:

```
<td width="125" valign="top" align="right" class="right-ads">
```

4 Choose to embed or link externally to the file and enter the appropriate code.

I choose to link externally, so I entered the following into the head portion of my HTML document:

```
<link href="printout.css" type="text/css" rel="stylesheet"
```

Caution: The display property is not supported by any 4.0 versions or below of Netscape or Internet Explorer. In this case, you won't be able to print out the printer-friendly version of the page, but the page will print. The display property is supported by Internet Explorer 5.*x* and higher on Windows and Macintosh, and Netscape 6.*x* for Windows and Mac. It is also supported by Opera 4.*x* and 5.*x*, and many versions of Mozilla.

Managing Navigation and Ads with Server-Side Includes

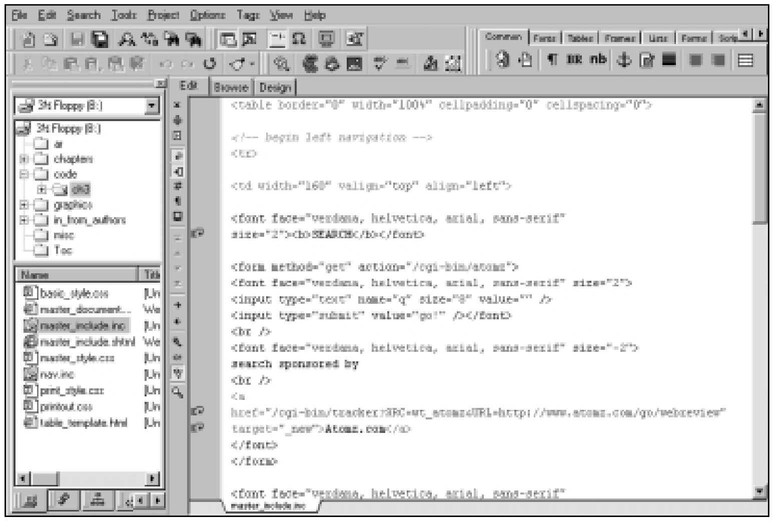
Consistent navigation is a key component to making your content accessible. Weighing the various options, I felt that, on a site this large, it might be helpful to pull that navigation out into a SSI. This way, any changes to that navigation can be made to just one file instead of the thousands files that reside on the site.

WebReview.com sits on an Apache server using OpenBSD as its OS. On this setup, it's possible to use SSIs to address multiple concerns. With an SSI, you can insert an include statement that asks the server to supply a snippet of code in place of the include when the page is served. It's an awesome technology when you're trying to manage large sites.

In the case of `WebReview.com`, SSIs helped me find solutions, such as navigation consistency, easy update access (the navigation SSI is one file and changes made to it apply to the entire site), and a means to separate the advertising markup from our own markup. The ad markup requires regular updating, so it makes sense to use SSIs.

Of course, different servers use different `include` methods. The one `WebReview.com` uses is fairly common. You can check with your hosting service for details. Otherwise, the following method is fairly general.

To view the `include` file, simply open it in an editor.



```
<table border="0" width="100%" cellpadding="0" cellspacing="0">

<!-- begin left navigation -->
<tr>

<td width="160" valign="top" align="left">

<font face="verdana, helvetica, arial, sans-serif"
size="2"><b>SEARCH</b></font>

<form method="get" action="/cgi-bin/atomz">
<font face="verdana, helvetica, arial, sans-serif" size="2">
<input type="text" name="q" size="8" value="" />
<input type="submit" value="go!" /></font>
<br />
<font face="verdana, helvetica, arial, sans-serif" size="-2">
search sponsored by
<br />
<a
href="/cgi-bin/tracker?SRC=wt_atomz&URL=http://www.atomz.com/go/webreview"
target="_new">Atomz.com</a>
</font>
</form>

<font face="verdana, helvetica, arial, sans-serif"
```

Viewing the contents of the `include` file in an editor.

To better understand an SSI:

1 Open `master_document.html` and find the navigation section, under the `begin left navigation` comment tag.

It should begin with the following:

```
<!-- begin left navigation -->

<font face="verdana, helvetica, arial, sans-serif"
➥size="2"><b>SEARCH</b></font>
<form method="get" action="/cgi-bin/atomz">
<font face="verdana, helvetica, arial, sans-serif" size="2">
<input type="text" name="q" size="8" value="" /> <input
➥type="submit"
value="go!" /></font>
<br />

<font face="verdana, helvetica, arial, sans-serif" size="-
➥2">search sponsored by<br />
<a href="/cgi-
bin/tracker?SRC=wt_atomz&amp;URL=http://www.atomz.com
/go/
```

continues

continued

```
</form>
<font face="verdana, helvetica, arial, sans-serif"
➥size="2"><b>INSIDE</b><br
/>
&nbsp;<a href="/index.shtml">Current Issue</a><br />
&nbsp;<a href="/columns.shtml">Columns</a><br />
&nbsp;<a href="/offline/2001/05_18_01.shtml">Offline</a><br
/>
```

2 Compare this to the file `master_include.inc`.

Notice that the navigation section and the `include` file contain the exact same markup, including portions of the table that's necessary to make the `include` fit accordingly into the main document, such as the following:

```
<tr>
<td width="160" valign="top" align="left">
```

All the navigation code has been removed from the originating file, and replaced with the `include` statement, which you see how to do now.

3 Open `master_include.shtml`, and note the `include` statement:

```
<!-- #include virtual='/includes/nav.inc' -->
```

This document has the `include` statements that you'll create and use on your site. The server will then input the markup automatically. When you call up the page and view source, you'll notice that the source displays completely. You don't see the `include` statements because the server already processed the document.

Note: Although SSIs are familiar to some readers, others haven't had the opportunity to explore them. It's an old, standard practice that's extremely efficient. As you can see, we were able to create one in a few steps. Yet, the way they work adds much power to the developer by controlling numerous pages with a single file—similar to a style sheet. The main challenge for those readers who want to try out SSIs is this: Be sure that you find out from your server administrator how they are properly achieved on the particular server that you are using. Sometimes, this information is available on your ISP's web site. In some cases, you unfortunately might not have access to include SSIs at all.

MANAGING JAVASCRIPT IN AN XHTML SITE

Perhaps the most frustrating thing about keeping the XHTML 1.0 documents on `WebReview.com` clean came about when the advertising code was delivered. The ad code is created by an outside source, and like so many professional companies, standards aren't an issue. The HTML they generate doesn't conform to any particular DTD, and they use a combination of Iframes and JavaScript to deliver the ads.

Listing 3.2 shows the markup the ad folk submitted to me.

LISTING 3.2 MARKUP FROM AN EXTERNAL SOURCE

```
<IFRAME WIDTH=120 HEIGHT=60 MARGINWIDTH=0
MARGINHEIGHT=0 ➥HSPACE=0 VSPACE=0
FRAMEBORDER=0 SCROLLING=no BORDERCOLOR=#000000
➥SRC="http://newads.cmpnet.com/html.ng/
site=webreview
&pagepos=topleftbutton">
<SCRIPT LANGUAGE="JavaScript"
SRC="http://newads.cmpnet.com/
js.ng/Params.richmedia=yes
&site=webreview
&pagepos=topleftbutton">
</SCRIPT>
</IFRAME>
```

Taking a quick look at this markup, you can already tell that it doesn't conform to the more rigid rules derived from XML and found in XHTML (See Chapter 1). Furthermore, when you use recommended markup, such as XHTML 1.0 or HTML 4, it creates some concerns. In this case, the problems are two-fold. First, the introduction of any proprietary or ill-used tag or attribute causes the document to be invalid (although it will still likely operate). Second, the introduction of the inline JavaScript that hasn't been escaped also causes the page to be invalid.

When you use JavaScript with XHTML, it's ideal if you can put your JavaScript into a separate document. This only works with scripts that would normally be embedded, however. (Inline scripting is another story.)

In the case of inline scripts, you have to escape special characters in order for the document to validate. The peskiest of these characters is the ampersand (&), which is used frequently in JavaScript. So you'll need to escape it whenever possible.

To escape character entities in JavaScript by hand, follow these steps:

1 Open `master_script.html`, which says the following:

```
<iframe width="120" height="60" marginwidth="0"
➥marginheight="0"
frameborder="0" scrolling="no"
src="http://newads.cmpnet.com/html.ng/site=webreview
&pagepos=topleftbutton">
<script type="text/javascript" language="JavaScript"
src="http://newads.cmpnet.com/js.ng/Params.richmedia=yes
&site=webreview
&pagepos=topleftbutton">

</script>
```

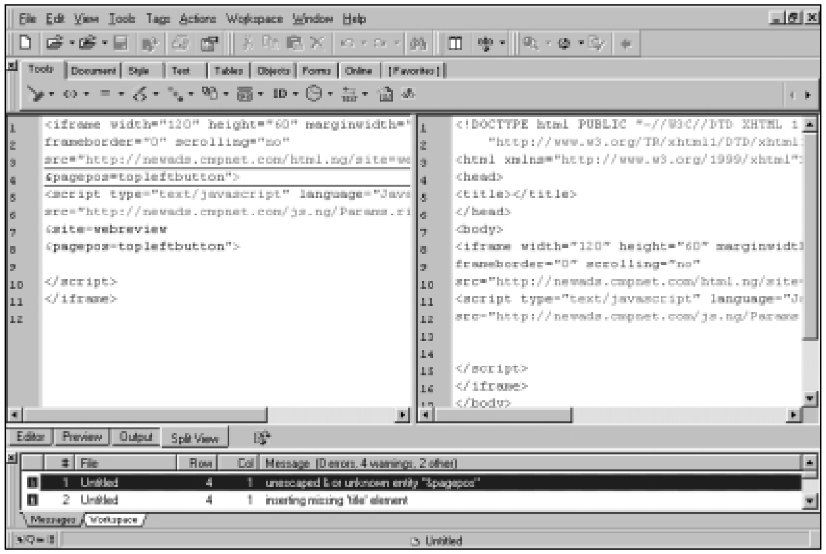
2 Open `master_script_escaped.html`, which contains the following:

```
<iframe width="120" height="60" marginwidth="0"
marginheight="0" frameborder="0" scrolling="no"
src="http://newads.cmpnet.com/html.ng/site=webtechniques&am
p;
➥pagepos=topleftbutton">
<script type="text/javascript" language="JavaScript"
src="http://newads.cmpnet.com/js.ng/Params.richmedia=yes&amp
;
➥site=webtechniques&amp;pagepos=topleftbutton">
```

3 Manually escape the attributes or use HTML Tidy to escape the character entities.

HTML Tidy is a faster and happier method, so I use the Tidy plug-in to HTML-Kit from Chami. The figure shows the conversion of the `master_script.html` to an escaped version.

Note: HTML Tidy is an excellent tool to test, repair, and convert files. Check it out at **`www.w3.org/`**. HTML-Kit can be found at **`www.chami.com/`**.



File Edit View Tools Tags Actions Workspace Window Help
Tools | Document | Style | Text | Tables | Objects | Forms | Online | [Favorites]
Editor | Preview | Output | Split View
Messages | Workspace
Untitled

```
<iframe width="120" height="60" marginwidth="
frameborder="0" scrolling="no"
src="http://newads.cmpnet.com/html.ng/site=we
&pagepos=topleftbutton">
<script type="text/javascript" language="Java
src="http://newads.cmpnet.com/js.ng/Params.ri
&site=webreview
&pagepos=topleftbutton">

</script>
</iframe>
```

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1
    "http://www.w3.org/TR/xhtml1/DTD/xhtml1
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
<title></title>
</head>
<body>
<iframe width="120" height="60" marginwidt
frameborder="0" scrolling="no"
src="http://newads.cmpnet.com/html.ng/site=
<script type="text/javascript" language="J
src="http://newads.cmpnet.com/js.ng/Params


</script>
</iframe>
</body>
```

	#	File	Row	Col	Message (0 errors, 4 warnings, 2 other)
	1	Untitled	4	1	unescaped & or unknown entity "&pagepos"
	2	Untitled	4	1	inserting missing 'title' element

Convert the document by using the HTML Tidy plug-in in HTML-Kit from Chami.

More Magic

Of course, you might want to make some practical changes to the exercises in this project to incorporate your own style—or to expand your own skill set.

Table Manners

Although I recommend dynamic, or a combination of fixed and dynamic, pages for most of today's contemporary web designs, many people use fixed designs. This is especially true for designers who want tight control out of their graphical layouts.

Here are some tips for keeping fixed tables under control:

- **Set the table tag width to a pixel width that's appropriate for your audience**—At the time of this writing, the most widespread resolution is 800 × 600, with a substantially growing number of individuals using higher resolutions.
- **Ensure that, in the `table` tag you set, the cell spacing and padding attributes to a `0` value** —This helps avoid gapping between table-cell content.
- **Do your math**—Each table cell must have a pixel width that matches its contents appropriately and adds up to the total width found in the table itself.
- **Put the closing `</td>` table cells on the same line as the table-cell contents**—This helps avoid gapping.
- **Place fixed-width layout tables in the center of the page**—This placement helps equalize any white space and avoid "left-heavy" pages.

Following these simple rules helps ensure that your tables are strongly built and attractively displayed to your site visitors.

Style Source

Searching for some style resources to help you learn to use style effectively? Try these:

- `www.w3.org/Style/CSS/`—The W3C style sheets home page.
- `style.webreview.com/`—`WebReview.com` Style Guide by Eric Meyer is a comprehensive resource that includes browser support comparison charts and style features.
- `www.meyerweb.com/eric/css/references/css1ref.html`—Eric Meyer's CSS1 quick-find W3C property reference for CSS1.
- `www.meyerweb.com/eric/css/references/css2ref.html`—CSS2 quick-find W3C property reference.

These style resources are considered the most authoritative available. Get familiar with them—you'll be glad that you did.

Server-Side Magic

One important way to help in the ever-present need to keep markup clean is to shift some responsibilities to the server. You saw how this can be done using SSIs.

Languages, such as Java, Perl and PHP, and applications, such as ColdFusion, Active Server Pages, and .NET, can work to your benefit by removing certain activities from the client.

A prime example of this is an alternative print method to the CSS2 print method. By using any of these applications (Perl is an extremely popular choice), you can create scripts that enable print processing of pages without having to rely on the HTML layout or CSS markup.