Page Layout With HTML

HTML's design capabilities have come a long way from the earliest days, when browsers simply stacked each element, one after another, in the browser window, flush left. First, Netscape extensions and table layout tags added tools for creating grids and white space. And now, full control over layout and typography are at hand with HTML style sheets, which designers can use to specify typeface, leading, indents, and even exact placement for each page element.

In the last chapter we described how HTML developed, beginning with the most basic structural tags, then adding design features willy-nilly with each new browser release until, at last, a truce was reached around HTML 4.0, which established a standard layout method: applying style sheets to structural elements, just as designers are used to doing in other media. Style sheets, with their ability to control position on the page as well as typographic settings, should finally provide a working solution to Web page layout. Until the world is using browsers that support the new standard, though, designers need to design pages with the entire history of HTML in mind. In this chapter, we'll show how layout effects can be achieved with a variety of methods and describe the benefits of each approach.

Structural Tags: The Building Blocks of HTML Layout

As we explained at the beginning of this book (→21), HTML has a lot in common with the typesetting systems used in the days before desktop publishing. Codes embedded in the file tag each element. Then the layout of each element is determined by specifications programmed into the typesetting software.

On the Web, the typesetting software is in each browser, which interprets the HTML code according to its own particular layout instructions. The tricky part is that, for older browsers, designers have no control over the typesetting specs that determine the look of the page; they're preprogrammed into the browser, and nothing can affect them. Browsers that support HTML 3.2 give designers control over a few different effects—typefaces and the placement of graphics, for instance—but rely on the built-in defaults for most specs. With style sheets, designers at last gain that control. Style sheets provide type specifications for each element that override the built-in defaults—returning to a traditional method of page layout.

Until most Web users have upgraded to browsers that support style sheets, the trick to HTML layout is to work with all these methods at once. Web designers need to code pages so that they work in early browsers—the ones that make up their own minds about the layout of each element. Then, designers can add HTML 3.2 style tags and style sheets for those browsers that understand them.

The key to both approaches is structural tags. In early browsers, the structural tags you use will control layout. In browsers that support style sheets, the structural tags provide the backbone on which you hang the layout instructions that will determine the page's look.

We tend to do pages that support a variety of browsers, which is much easier than supporting different pages for different browsers. But the level of differentiation between the two sets of code is really dependent on how much money the client is willing to invest.

STEFAN FIELDING-ISAACS, ART & SCIENCE

START TAG	DEFINITION	END TAG	SPECIFICATION
<BLOCKQUOTE>	Block (indented)	</BLOCKQUOTE>	12-pt. Times, indented 48 pixels from the left and right quotation margins
<P>	Paragraph		12-pt. Times, 16 pixels space above and below
<HR>	Horizontal rule		2 pixel line, flush left
<H1>	Level-1 head	</H1>	24-pt. Times bold, flush left
<H2>	Level-2 head	</H2>	18-pt. Times bold, flush left
<H3>	Level-3 head	</H3>	14-pt. Times bold, flush left
<H4>	Level-4 head	</H4>	12-pt. Times bold, flush left
<H5>	Level-5 head	</H5>	10-pt. Times bold, flush left
<H6>	Level-6 head	</H6>	8-pt. Times bold, flush left
<DIR>	Directory list	</DIR>	12-pt. Times indented 48 pixels, preceded by a bullet, indented 36 pixels. <LI> starts a new item.
<DL> **<DT>** **<DD>**	Definition list	</DL> </DT> </DD>	<DT> (definition term) 12-pt. Times, flush left <DD> (definition) 12-pt. Times, indented 48 pixels
<MENU>	Menu list	</MENU>	12-pt. Times indented 48 pixels, preceded by a bullet, indented 36 pixels. <LI> starts a new item.
<OL> **<LI>**	Ordered (numbered) list	</OL>	12-pt. Times, indented 48 pixels, first line preceded by Arabic numeral and indented 33 pixels. (The *type=* attribute can be used to change the numbering style to upper- or lowercase letters or Roman numerals.) <LI> starts a new item.
<UL> **<LI>**	Unordered (bulleted) list	</UL>	12-pt. Times, indented 48 pixels, first line preceded by a bullet and indented 36 pixels. (The *type=* attribute can be used to change the bullet style.) <LI> starts a new item.

HTML'S STRUCTURAL TAGS name a document element, not a particular design, but most browsers use similar specifications for each tag. The specifications shown here are those used on a PC in Netscape Navigator. Other browsers may use slightly different specs.

Using Structural Tags for Layout

Used as they were meant to be, structural tags create the kind of layout you might have last used typing up a college term paper: a couple levels of heads, double spaces between flush-left paragraphs, and one basic typeface, Times. In short, the preset styles create a lowest-common-denominator layout—readable but boring.

The default settings are usually the same from browser to browser. Text tagged <BLOCKQUOTE> will usually be 12-point Times, indented about half an inch on each side, for example.

Structural tags aren't really meant for design. Using HTML as it was meant to be used means simply tagging each element appropriately: <H1> for a top-level head, for instance, and <BLOCKQUOTE> used only for indented quotations. (This ensures that HTML files can be used, according to the tenets of SGML, in other applications as well.) Of course, that leads to a very boring page, and HTML doesn't have enough structural tags to label any except the most basic elements, anyway, so very few page authors actually use HTML that way.

Instead, designers quickly figured out that they could use HTML tags for their design attributes, rather than their structural purpose. If they wanted their text in 14-point Times rather than the default 12-point, they could simply tag all their text <H3>. A half-inch margin around text could be achieved by tagging all the text with <BLOCKQUOTE>. Two <BLOCKQUOTE> tags result in a deeper margin. Because the basic tags are so few, and because the available typefaces are, usually, limited to two, the variety is limited. In the early days of the Web, though, this was the only available method of exerting any control at all over the look of pages.

Designers should remember that this approach isn't foolproof. The defaults you count on may change from browser to browser and even from one release of a particular browser to another. And all your assumptions are blown as soon as independent-minded readers decide to set their own preferences. Text you thought safe at 12-point Times might actually be seen as 16-point Chancery Script.

About The World Wide Web Consortium

The World Wide Web Consortium exists to realize the full potential of the Web.

W3C is an industry consortium which develops common standards for the evolution of the Web by producing specifications and reference software. Although W3C is funded by industrial members, its products are freely available to all.

The Consortium is run in the United States by the MIT Laboratory for Computer Science and in Europe by INRIA, in collaboration with CERN where the Web originated. For details on the joint activities and the contributions of CERN, INRIA, and MIT, please see the statement on the start World Wide Web initiative.

Membership Information

W3C Membership is open to any organization which signs a membership agreement, further detailed in the Prospectus.

To learn about individual members and visit their World Wide Web sites, see the Members of the W3 Consortium.

Services

The W3C provides a number of public services:

- A repository of information about the World Wide Web for developers and users, especially specifications about the Web
- A reference code implementation to embody and promote standards
- Various prototype and sample applications to demonstrate use of new technology

Any products of the consortium are available during development and on initial release to members. However, one month after a formal internal release, all software produced by or explicitly contributed to the W3C is available for general public use, commercial or otherwise.

Further information

- Frequently Asked Questions about W3C
- Help wanted: come and work with the W3C Team at MIT or INRIA
- Consortium slides about W3C
- People of the W3C
- How to contact W3C
- Press Information

W3C is hosted by the MIT Laboratory for Computer Science and in Europe by INRIA with support from DARPA and the European Commission.

http://www.w3.org/consortium/

USING HTML 2.0 as it was intended results in a lowest-common-denominator layout reminiscent of a college term paper—or word processing circa 1983.

ad319

In February of 1993, ad319 was born from the simultaneous efforts of three artists and designers trained in traditional mediums, all of whom were attempting to embrace new digital technologies. The founding members of ad319 were Kathleen Chmielewski, Nan Goggin, and Joseph Squier. The idea of working as a collective seemed an effective way to pool our knowledge, and an efficient means of addressing the issues we face as contemporary artists, designers and educators. One outgrowth of this collaborative approach has been the ghost gallery.

Feedback is welcomed.

ad319 members also collaborate on the creation of electronic artwork. Their most recent piece is Body Space Idiocracy, which exists as a separate Web site. It was very favorably reviewed by HotWired in April of 1995. A CD-ROM version of Body,Space,Idiocracy was exhibited at the Centre George Pompidou in Paris from November 1994 to January 1995.

Additionally, Joseph Squier maintains his own site for Web-based artwork, which is called the place.

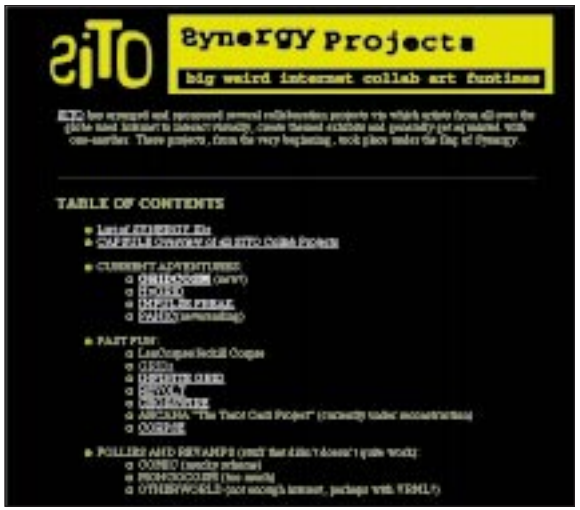
ad319 also curated an exhibition of electronic art for the Krannert Art Museum. *Art as Signal: Inside the Loop* was a survey of the best contemporary electronic art from around the world. It ran from November 1995 to January 1996. An accompanying CD-ROM catalog will be produced during the summer of 1996.

These projects were made possible through the substantial support of several groups at the University of Illinois; including the Center for Graphic Technology, the Advanced Information Technologies Group, and the Women, Information Technology, and Scholarship group.

AD319
http://www.art.uiuc.edu/@art/ad319/ad319.html

AVALANCHE
http://www.avsi.com/

USING STRUCTURAL TAGS as they were never meant to be used adds new layout options. Using five <BLOCKQUOTE> tags (as in the ad319 page, left) creates a deep indent. Multiple <LI> tags (as used in the page from Avalanche's old site, above) creates a similar white space. (The headings in Avalanche's page were created using <PRE>.) The Sito site uses straightforward list layout, but adds space around its intro text with <BLOCKQUOTE>. Using HTML 2.0 codes, like these, rather than the HTML 3.2 table tags or the HTML 4.0 style sheets that would create the same effect, ensures that the intended effect will be seen in even the oldest browsers.



ED STASNY
http://www.sito.org/synergy/

Controlling Layout With <PRE>

An HTML 2.0 trick that deserves special mention is the <PRE> (preformatted text) tag, which was created to let page authors specify an exact layout for text. Any text between <PRE> and </PRE> tags is displayed exactly as it is typed, including extra spaces and any returns (which are usually ignored inside other tags). Using <PRE>, designers can arrange text painstakingly with the space bar—an effect especially useful for poetry or for other layouts in which the placement of sparse text is key. By default, most browsers display <PRE> text as 10-point Courier.

TEXT TAGGED AS <PRE> is laid out exactly as it is typed in the HTML file (right), allowing designers to lay out text precisely. By default, <PRE> text is displayed in the Courier typeface.

JULIET MARTIN

http://www.bway.net/~juliet/oooxxooo/Parenthesis.html

START TAG	ATTRIBUTES	END TAG	EXPLANATION
<PRE>		</PRE>	Marks text that should be laid out exactly as typed

THE ALIGN= ATTRIBUTE for the <IMG> tag lets designers wrap text around graphics. The *hspace=* and *vspace=* attributes create space between the image and the surrounding text.

the ONION • 25 FEBRUARY 1998

Clinton Threatens To Drop Da Bomb On Iraq

CHOCOLATE CITY--In an address before an emergency session of Parliament Monday, George Clinton said he is prepared to drop Da Bomb on Iraq if Saddam Hussein does not loosen up and comply with U.N. weapons inspectors by the Clinton-imposed deadline of March 1.

"For Saddam Hussein to refuse to let U.N. officials inspect Iraqi weapons facilities as per the terms of Iraq's 1991 Gulf War surrender is decidedly unfunky of him," Clinton said. "While the decision to drop Da Bomb is never an easy one, unless Saddam gets down with this whole U.N.-inspection thang and seriously refunktatizes his stance by March 1, we will have no choice but to tear the roof off Baghdad."

Preparations for the military strike, dubbed Operation Supergroovalistoppitsifinkstication Storm, are already underway. The Mothership is ready and on standby at Starchild Air Force Base in Detroit, where more than 5,000 bop gunners are making final preparations for deployment to the Persian Gulf. Clinton has also ordered an additional 2,500 Aquaboogie Amphibious Assault units to the Gulf, bringing the total P-Funk Nation military presence in the region to 33,000.

According to General William "Bootsy" Collins, the primary goal of the ground assault is to breach Hussein's presidential palace, capture the Iraqi leader, and "put some serious funk in his trunk."

Collins acknowledged that the mission would not be easy.

George Clinton

JACK SZWERGOLD / ANDREW WELYCZKO

http://www.theonion.com/onion3307/clintondropsdabomb.html

Orphaned Tags

Most of the extensions introduced by Netscape and Microsoft were deemed useful and were later incorporated into official HTML. But not all of them. The history of HTML is littered with tags that failed to enter the language because they were considered too annoying or too antithetical to HTML principles or because they were replaced by standard HTML methods for achieving the same effects.

<BLINK> (blinking text), introduced by Netscape in Navigator 1.1, fell into the first category. Other orphaned tags include Microsoft's <MARQUEE> (a scrolling message across the browser window, introduced in Internet Explorer 2.0), and Netscape's <MULTICOL>, <SPACER>, and <LAYER> tags, which were introduced with Navigator 3.0 and 4.0, after the Web community clearly saw that such specific style tags were the wrong way to go about extending HTML.

These tags live on the browsers of the companies that introduced them, largely so that any pages created with them in the past will display correctly. But few Web designers still use them.

HTML Extensions and HTML 3.2

The layout tags described in HTML 3.2 actually crept onto the Web bit by bit, introduced piecemeal by early versions of Netscape's and Microsoft's browsers, as Netscape or Microsoft extensions (→**74**). These extensions included new structural tags, like those for tables (→**90**). Here, we'll talk about what we called style tags in the last chapter (→**72**): tags that were designed to control the way elements look on screen.

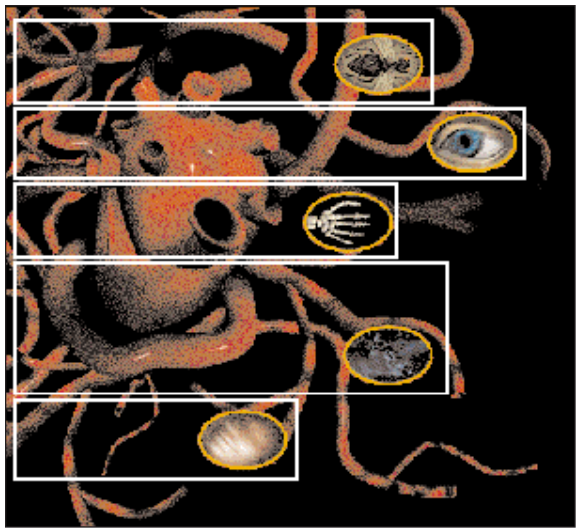
For layout purposes, perhaps the most important of the extensions was the *align=* attribute for the <IMG> tag (→**124**), which allowed designers to wrap text around graphics. Used straightforwardly, the new attributes allowed designers to place graphics at the right or left margin, with text wrapping around them as in a magazine layout. The accompanying *vspace=* and *hspace=* attributes were used to create space between the graphic and the surrounding text. Even more cleverly, though, designers used it to wrap text around graphics with transparent backgrounds (→**122**), opening up expanses of white space on HTML pages for the first time.

Other tags that gave designers a bit of control were the <CENTER> tag, allowing designers to center text and graphics on the page, and Microsoft's *leftmargin=* and *topmargin=* attributes for the <BODY> tag, creating a way to add space around the edges of a page without resorting to <BLOCKQUOTE>.

As you can see, the controls were far from rich, but Web designers got some great mileage out of them, creating some surprisingly elegant pages.

The next step forward, introduced in Navigator 2.0 and adopted by Internet Explorer 3.0 (and then canonized by HTML 3.2) were the table and frame tags, structural tags that allowed designers to create a real grid for Web pages.

START TAG	ATTRIBUTES	END TAG	EXPLANATION
<BODY>		</BODY>	Marks the text to be displayed in the browser window
	leftmargin=*n*		Sets a left margin, described as a number of pixels
	topmargin=*n*		Sets a top margin, described as a number of pixels
<CENTER>		</CENTER>	Marks text that should be centered in the window
<HR>			Inserts a horizontal rule
	align="right" OR "left" OR "center"		Specifiies the rule's placement
	color="*#RRGGBB*" OR "*name*"		A color for the rule, specified in hexadecimal or as a color name
	noshade		Removes the rule's default drop shadow
	size=*n*		The width (height) of the rule, in pixels
	width=*n* OR "*n%*"		The length of the rule, in pixels or as a percentage of the window width
<MULTICOL>		</MULTICOL>	Marks text that should be set in multiple columns
	cols=*n*		The number of columns
	gutter=*n*		The amount of space between columns, in pixels
	width=*n*		The width of the column set, in pixels
<SPACER>		</SPACER>	Creates a blank space in the page layout
	align="left" OR "right" OR "top" OR "texttop" OR "middle" OR "absmiddle" OR "baseline" OR "bottom" OR "absbottom"		For *type=block,* tells the browser how to wrap the adjoining text around the space
	height=*n*, width=*n*		For *type=block*, the width and height of the empty space
	size=*n*		For *type=horizontal* or *type=vertical,* the size of the empty space, in pixels
	type="horizontal" OR "vertical" OR "block"		Tells the browser to create a space in the current line (*horizontal*), to create a vertical space above the next item (*vertical*), or to create a rectangular space (*block*)



BILL DOMONKOS
http://www.bdom.com/documents/homepage.html

```
<HTML><HEAD><TITLE>B Domonkos Home</TITLE></HEAD>
<BODY BGCOLOR="#000000"
BACKGROUND="../images/backhome.gif">

<TABLE CELLPADDING="0">
<TR>
<TD WIDTH="349" HEIGHT="68" ALIGN=RIGHT VALIGN=BOTTOM>
<A HREF="illustration.html"><IMG SRC="../images/fly.gif" HEIGHT="61"
WIDTH="84" BORDER="0"></A></TD>
</TR>
</TABLE>

<TABLE CELLPADDING="0">
<TR>
<TD WIDTH="428" HEIGHT="52" ALIGN=RIGHT VALIGN=TOP>
<A HREF="movies.html"><IMG SRC="../images/eye.gif" HEIGHT="57"
WIDTH="82" BORDER="0"></A></TD>
</TR>
</TABLE>

<TABLE CELLPADDING="0">
<TR>
<TD WIDTH="320" HEIGHT="61" ALIGN=RIGHT VALIGN=BOTTOM>
<A HREF="3d.html"><IMG SRC="../images/hand.gif" HEIGHT="60"
WIDTH="81" BORDER="0"></A></TD>
</TR>
</TABLE>

<TABLE CELLPADDING="0">
<TR>
<TD WIDTH="361" HEIGHT="110" ALIGN=RIGHT VALIGN=BOTTOM>
<A HREF="blueroom.html"><IMG SRC="../images/blue.gif"
HEIGHT="66" WIDTH="92" BORDER="0"></A></TD>
</TR>
</TABLE>
```

HTML TABLES let you position items at particular pixel locations on screen. In this example, the designer uses table cells of different widths and heights to stagger the graphic buttons (aligned at the bottom right of each cell) on top of a background graphic.

Page Layout With Tables

HTML's table layout tools were designed with traditional tables in mind—the kind that hold statistics within a page—but designers quickly adopted them for structuring whole pages. With tables, designers can specify different-width columns to break up a page horizontally and discrete rows to control vertical space. Table cells can include text, graphics, and even other tables.

HTML tables don't provide the same kind of flexibility you can get from page layout programs such as QuarkXPress or PageMaker. The grids they create can't accommodate overlapping columns or some other niceties that add the sophisticated asymmetry possible with print layouts. In the environment of the Web, tables can also be problematic for other reasons. Complex tables can choke some browsers, and the fact that visitors' font and screen sizes are unpredictable means that fixed-size rows and columns may cut off parts of the table's content from view. (Table cells can also be flexible, expanding with the cells' contents and reflowing as viewers change their window sizes.) On the other hand, tables do allow designers to specify, to the pixel, exactly where text or images will be placed on a page—a control they never had before.

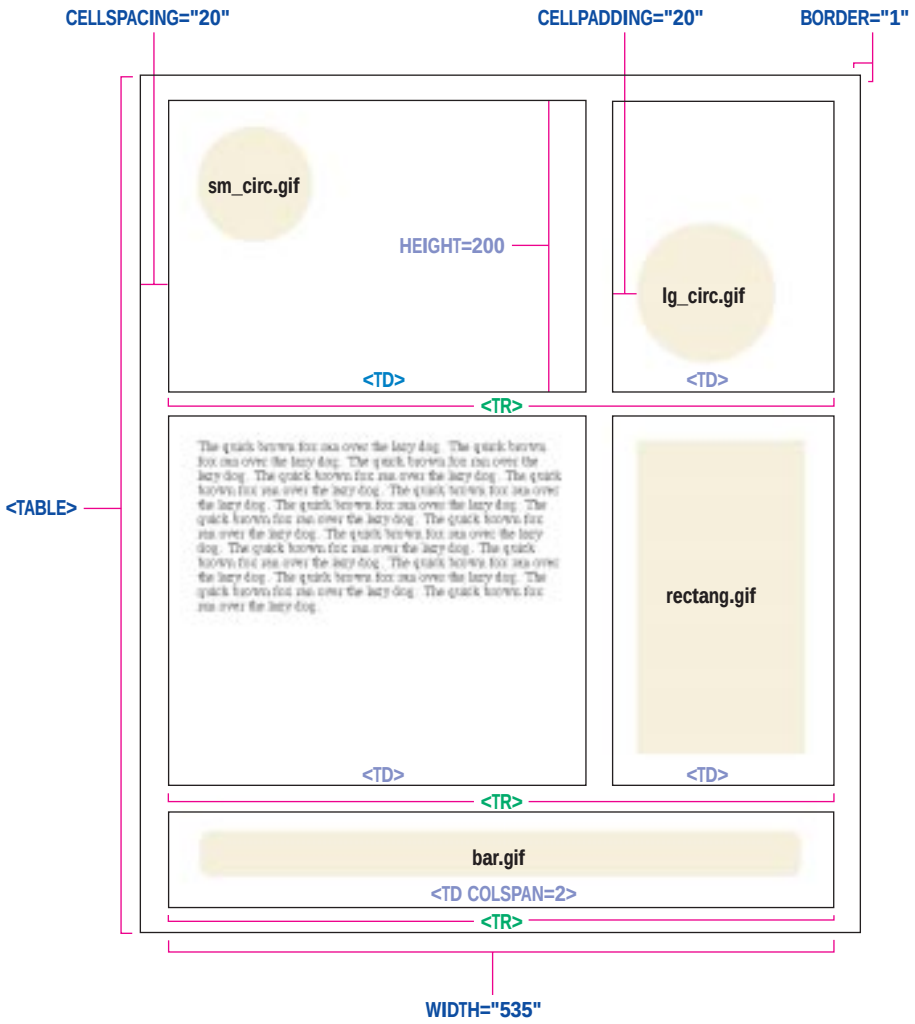
AMY FRANCESCHINI / DAVID KARAM / MICHAEL MACRONE / OLIVIER LAUDE
http://atlasmagazine.com/photo/laude_despair/index.html

AVALANCHE
http://www.ear1.com/

HTML TABLES can create strict grids or more freeform shapes.

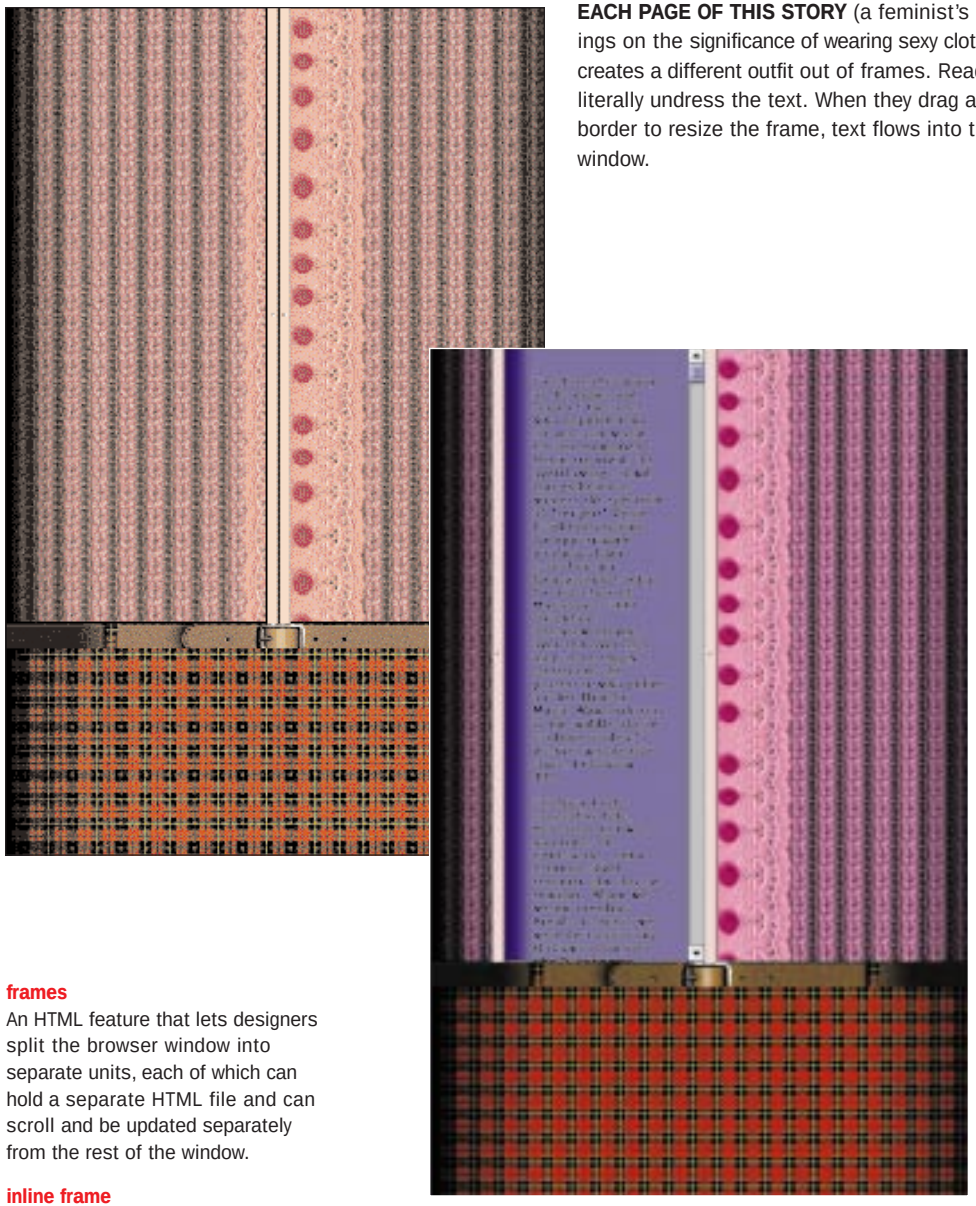
LOOKING - DESIGN FOR COMMUNICATION
http://www.lacountyarts.org/ford.html

START TAG	ATTRIBUTES	END TAG	EXPLANATION
<TABLE>		</TABLE>	Surround all the tags that make up the table
	align="left" OR "right" OR "center"		The table's alignment in the window
	background="*URL*"		An image file to be used as the table's background
	bgcolor="*#RRGGBB*" OR "*name*"		The color of the table's background, using RGB values (expressed in hexadecimal) or a color name
	border=*n*		A width for the table's border, in pixels. *border=0* means no border.
	cellpadding=n		The space between each cell's border and its contents, specified in pixels
	cellspacing=*n*		The space between each cell's contents, specified in pixels
	cols=*n*		The number of columns in the table
	height=*n*, width=*n* or *n*%		The table's total height and width, specified in pixels or (for width) as a percentage of the window size
	rules="none" OR "groups" OR "rows" OR "cols" OR "all"		Specifies which rules will appear in the table
<CAPTION>		</CAPTION>	Creates a caption for the table
<COL> **<COLGROUP>**		</COLGROUP>	Create column groupings. <COL> allows authors to set attributes for several columns at once; <COLGROUP> groups the columns structurally, so that they will be laid out together in a browser window.
	align="left" OR "right" OR "center" OR "justify" OR "char"		The alignment of the cells' contents
	span=*n*		The number of columns in the group
	valign="top" OR "middle" OR "bottom" OR "baseline"		The vertical alignment of the cell's contents relative to its borders
	width=*n* OR "0*"		A default width for each column in the group. "0*" means each column should be just wide enough to hold its contents.
<TD> **<TH>**		</TD> </TH>	Mark the data (<TD>) or heading (<TH>) that goes in each table cell
	align="left" OR "right" OR "center"		The data's alignment in the cell
	background="*URL*"		An image file to be used as the cell's background
	bgcolor="*#RRGGBB*" OR "*name*"		A color for the cell's background
	colspan=*n*		The number of columns the cell spans
	height=*n*, width=*n* OR *n*%		The height and width of the table cell, in pixels or (for width) as a percentage of the table size
	rowspan=*n*		The number of rows the cell spans
	valign="top" OR "middle" OR "bottom" OR "baseline"		The vertical alignment of the cell's contents relative to its borders
<THEAD> **<TFOOT>** **<TBODY>**		</THEAD> </TFOOT> </TBODY>	Group cells into a table heading, table footer, and table body, respectively. Browsers may scroll table bodies while leaving the header and footer in place.
<TR>		</TR>	Creates a new table row. <TR> and </TR> contain a set of table cells defined by <TD> and <TH>.
	align="left" OR "right" OR "center" OR "justify" OR "char"		The alignment of the contents of the row's cells
	bgcolor="#RRGGBB" OR "name"		A color for the table row's background
	valign="top" OR "middle" OR "bottom" OR "baseline"		The vertical alignment of the row's contents relative to the cell's borders



AN HTML TABLE is set up from a series of rows and columns. The <TABLE> tag's *width=, height=,* and other attributes define the overall dimensions of the table. Then the table is constructed row by row. A <TR> (table row) tag, creates each row; <TH> (table head) and <TD> (table data) tags mark the content of each cell. Cells can hold any kind of data, including graphics or other media.

```
<HTML>
<HEAD><TITLE>TABLE</TITLE></HEAD>
<BODY BGCOLOR="WHITE">

<TABLE WIDTH="535" BORDER="1" CELLSPACING="20"CELLPADDING="20">

    <TR ALIGN=LEFT>

        <TD VALIGN="TOP" HEIGHT="200">
        <IMG SRC="sm_circ.gif">
        </TD>

        <TD VALIGN="bottom">
        <IMG SRC="lg_circ.gif">
        </TD>

    </TR>

    <TR ALIGN=LEFT>

        <TD VALIGN="TOP">
        The contents of each cell is embedded  between &lt;TD&gt; (table
        data) tags. It can include graphics, plug-ins, or even other tables. You
        can align cell contents horizontally and vertically in a number of ways
        within the cell. The text (or other content) is inset by the number of
        pixels set in the cellpadding= attribute. You can set a specific height
        and width for a cell using the &lt;TD&gt; or &lt;TH&gt; tag's height=
        and width= tags. If you don't use those attributes, the cell will be as
        wide and tall as it needs to be to fit the cells' contents.
        <IMG SRC= "pixel.gif" WIDTH=300 HEIGHT="1">
        </TD >

        <TD>
        <IMG SRC="rectang.gif" WIDTH="140" HEIGHT="260">
        </TD>

    </TR>

    <TR ALIGN=CENTER>

        <TD COLSPAN=2>
        <IMG SRC="bar.gif">
        </TD>

    </TR>

</TABLE>

</BODY>
</HTML>
```



EACH PAGE OF THIS STORY (a feminist's musings on the significance of wearing sexy clothes) creates a different outfit out of frames. Readers literally undress the text. When they drag a frame border to resize the frame, text flows into the new window.

JASON HUANG / YOSHI SODEOKA
http://www.word.com/desire/garterbelt/

frames
An HTML feature that lets designers split the browser window into separate units, each of which can hold a separate HTML file and can scroll and be updated separately from the rest of the window.

inline frame
A frame that is not part of a frameset but is defined individually.

Dividing the Window With Frames

Similar in some ways to tables, but offering some different advantages (and disadvantages) are **frames**. Like tables, frames let designers divide a window into any number of horizontal and vertical rows and columns. But unlike table cells, each frame can hold a separate HTML file, and each frame can scroll separately.

Frames can be individually named. That name can then be used as the target for a hyperlink (→78) so that a click in one frame can change the contents of another. This makes frames a natural solution for setting off the navigation controls for a site; the navigation controls always remain on screen while new pages are loaded into a separate frame.

Like table rows and columns, frames can be fixed in size or scale to fit the content. They can also be designed with or without borders and scrollbars.

HTML 4.0 includes a new feature called **inline frames**, free-floating frames that can be placed at any pixel coordinate within a window.

In the opinion of many Web users, frames have been a mixed blessing. You can't print or bookmark a page that's inside a frameset, and some users find navigating within frames confusing. Even given these caveats, though, frames can sometimes provide a practical solution to Web design problems.

AVALANCHE
http://www.oneclub.com/

WINNERS OF THE ONE SHOW are listed in the right-hand frame of this site. Clicking on a link there displays the selected work in the large center frame.

ROGER LOS
http://www.austinhealey.com/big.html

SITE NAVIGATION is set off in a frame across the bottom of the Austin-Healey site. On this page, a timeline fills the upper frame; users scroll horizontally to move through the years.

LANCE ARTHUR
http://www.glassdog.com/the_lab/toppage.html

A CONTROL PANEL and the site's branding are set off in frames across the top and left side of this site for GlassDog Labs.

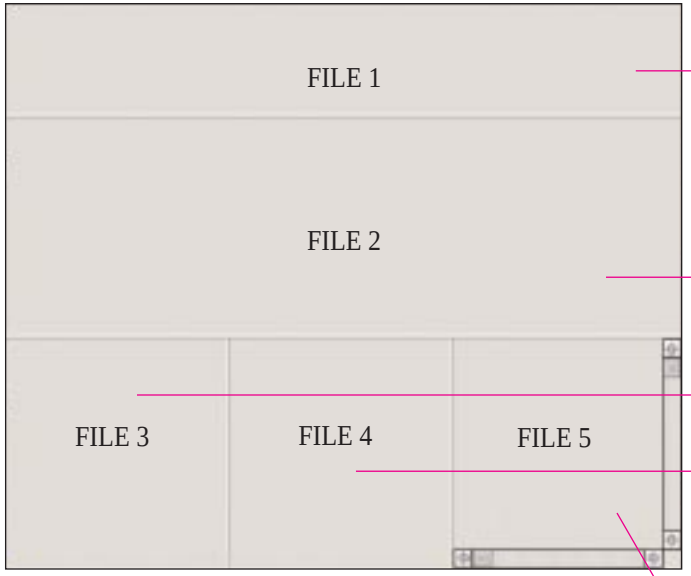
Frames are a great concept, but the way they're implemented doesn't work that well. We no longer use them.

FRED SOTHERLAND, CNET

START TAG	ATTRIBUTES	END TAG	EXPLANATION
<FRAMESET>		</FRAMESET>	Encloses all the tags that make up a set of frames
	border=*n*		Sets a border (*1* or "*yes*") or omits a border (*0* or "*no*") around a frame. (Microsoft's browser uses the numbers, Netscape's the words.)
	bordercolor="*#RRGGBB*" OR "*name*"		A color for the border, specified as RGB values (in hexadecimal) or as a color name
	cols="*col1, col2, col3, ...*"		Sets up a frameset as a set of "columns." The set of columns is specifed by giving a width for each one. Widths can be specified in pixels, as a percentage of the window size, or as an asterisk (*), meaning that the column should take up the remaining space. If more than one column is specified with an asterisk, the space is divided evenly among them.
	frameborder=0 OR 1 OR "yes" OR "no"		Sets a border (*1* or "*yes*") or omits a border (*0* or "*no*") around a frameset. (Microsoft's browser uses the numbers, Netscape's the words.)
	rows="*row1, row2, row3,...*"		Sets up a frameset as a set of "rows." The set of rows is specifed by giving a width for each one. Widths can be specified in pixels, as a percentage of the window size, or as an asterisk (*), meaning that the row should take up the remaining space. If more than one row is specified with an asterisk, the space is divided evenly among them.
<FRAME>			Specifes the attributes of one frame within a frameset
	bordercolor="*#RRGGBB*" OR "*name*"		A color for the border, specified as RGB values (in hexadecimal) or as a color name
	frameborder=0 OR 1 OR "yes" OR "no"		Sets a border (*1* or "*yes*") or omits a border (*0* or "*no*") around a frame.
	marginheight=*n*		Creates a margin at the top and bottom of the frame (specified in pixels)
	marginwidth=*n*		Creates a margin at the left and right sides of the frame (specified in pixels)
	name="*name*"		A target name for the frame (used by <A> tags to send linked files to that particular frame)
	noresize		Prevents users from resizing the frame (by omitting the resize box)
	scrolling="yes" OR "no" OR "auto"		Includes or omits a scroll bar for the frame. By default (or using "*auto*") a scroll bar appears if the frame's contents go beyond its borders.
	src="*URL*"		The URL of the file to be placed in the frame
<NOFRAMES>		</NOFRAMES>	Marks content that should be displayed in browsers that don't support frames. Browsers that support frames ignore any code marked with <NOFRAMES>.

FRAMESETS ARE SET UP either as a set of rows or as a set of columns. Multiple framesets can be nested to create columns within rows, or vice versa. Once the frameset is defined, individual <FRAME> tags are used to name the content and set the style of each frame; a separate file is loaded into each frame. The <NOFRAMES> tag sets off copy that will be shown on browsers (prior to Navigator 2.0 and Internet Explorer 3.0) that don't support frames. The <IFRAME> tag, new in HTML 4.0, creates an inline frame.

```
<HTML>
<FRAMESET rows="*,*,*">
    <FRAME src="file1.html">
    <FRAME src="file2.html">
    <FRAMESET cols="*,*,*">
        <FRAME src="file3.html">
        <FRAME src="file4.html">
        <FRAME src="file5.html">
    </FRAMESET>
</FRAMESET>
</HTML>
```




```
<HTML>
<HEAD>
<TITLE>FILE1</TITLE>
</HEAD>
<BODY>
<H1>FILE1</H1>
</BODY>
</HTML>
```

FILE1.HTML

```
<HTML>
<HEAD>
<TITLE>FILE2</TITLE>
</HEAD>
<BODY>
<H1>FILE2</H1>
</BODY>
</HTML>
```

FILE2.HTML

```
<HTML>
<HEAD>
<TITLE>FILE3</TITLE>
</HEAD>
<BODY>
<H1>FILE3</H1>
</BODY>
</HTML>
```

FILE3.HTML

```
<HTML>
<HEAD>
<TITLE>FILE4</TITLE>
</HEAD>
<BODY>
<H1>FILE4</H1>
</BODY>
</HTML>
```

FILE4.HTML

```
<HTML>
<HEAD>
<TITLE>FILE5</TITLE>
</HEAD>
<BODY>
<H1>FILE5</H1>
</BODY>
</HTML>
```

FILE5.HTML

START TAG	ATTRIBUTES	END TAG	EXPLANATION
<IFRAME>			Creates an "inline frame," which flows with the document text
	frameborder=0 OR 1 OR "yes" OR "no"		Sets a border (*1* or "*yes*") or omits a border (*0* or "*no*") around a frame. (Microsoft's browser uses the numbers, Netscape's the words.)
	hspace=*n*, vspace=*n*		The horizontal and vertical space, in pixels, between the frame and the surrounding text
	height=*n*, width=*n*		The height and width of the frame, in pixels
	marginheight=*n*		Creates a margin at the top and bottom of the frame (specified in pixels)
	marginwidth=*n*		Creates a margin at the left and right sides of the frame (specified in pixels)
	name="*name*"		A target name for the frame (used by <A> tags to send linked files to that particular frame)
	scrolling="yes" OR "no" OR "auto"		Includes or omits a scroll bar for the frame. By default (or using "*auto*") a scroll bar appears if the frame's contents go beyond its borders.
	src="*URL*"		The URL of the file to be placed in the frame

STYLE SHEETS ARE simply lists of layout specifications for different HTML elements. In CSS, each specification consists of a selector, which names the element the styles apply to, and a list of style properties and their values, enclosed in brackets. (The CSS specification defines the possible properties and values.) A colon separates the property from its value. Multiple property settings can be provided for a single element; in such a list, the property and value pairs are separated by semicolons.

cascading style sheets (CSS)
The most widely supported style sheet language for Web publishing.

CSS1
The first version of the cascading style sheet language.

DSSSL
Document semantics and style specification language, a popular style language for SGML publishing.

style sheet
Layout specifications added to an HTML file.

XSL
Extensible style language, a style language under development as a companion to XML.

Specifying Layout With Style Sheets

Beginning with HTML 3.2, HTML includes support for a layout solution that, at last, strikes a balance between designers' need to control the layout of Web pages and HTML's premise of specifying structure, not layout, to ensure a document's usefulness across applications. That solution, which promises to revolutionize Web design, is **style sheets**.

Web style sheets work much like the style sheets used in popular word processing and page layout programs. Standard HTML structural tags (<H1>, <P>, and so on) mark each element. Instead of using default layouts for each element, though, browsers will look for specifications—style sheets—defined by the designer. (Browsers will fall back on the default if no style sheets are provided or if they don't support style sheets.)

Just as with style sheets in word processing and page layout programs, a single style definition can be used to style every instance of a certain element with a single command, and designs for entire documents—even entire sites—can be easily changed by simply changing the centrally defined style attributes.

Exactly what style sheets can do is controlled by the specific style sheet language you use. Right now, the standard style sheet language for the Web is called **cascading style sheets** (**CSS** for short). Other style

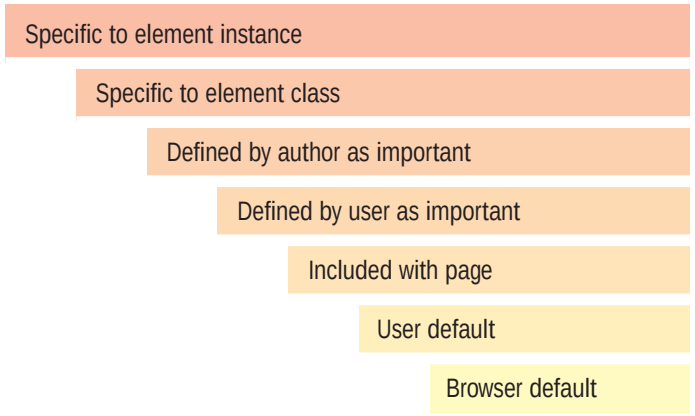
sheet languages, such as **DSSSL** (document semantics and style specification language), were developed for use with other applications of SGML. And **XSL** (→**145**), the extensible style language, is being developed as a companion for XML (→**142**). For now, though, CSS is center of attention: **CSS1** (the first version of CSS) is supported by Microsoft's and Netscape's current browsers.

A style sheet is simply a list of layout specifications for each HTML element in a document. CSS gives a lot more control over HTML layout than has any solution that has come before, letting designers specify such attributes as point size, line spacing (leading), and indents for text. And CSS layout specs can use standard design and typographic measurements like points, picas, and ems, as well as pixels and percentages, to describe a page.

You can add stylesheets to your Web pages in a few different ways, depending on how widely you want the styles to be used. You can import external style sheets (describing, say, standard styles used by your company or publication) using the <LINK> tag in a document's heading. You can use the HTML <STYLE> tag in the document's heading to add document-wide styles. Or you can use a *style=* attribute with just about any HTML tag, to describe styles that pertain only to that element. (For instance, you could add a style property to a <DIV> tag to set styles for the elements within that division or to a <P> tag to affect a

START TAG	ATTRIBUTES	END TAG	EXPLANATION
<STYLE>		</STYLE>	Enclose the style sheets for an HTML document
	type="*MIME-type*"		The style sheet language, defined as a MIME type (e.g., css/text)
	media="screen" OR "print" OR "projection" OR "braille" OR "aural" OR "all"		The media types the style sheet should be used for
<LINK>		</LINK>	Links an external document to the current file
	href="*URL*"		The location of the linked document
	rel="*description*"		The relationship of the linked file to the current document. For style sheets, the setting would be *rel="stylesheet"*
	type="*MIME-type*"		The MIME type of the linked content; for style sheets, usually "text/css"

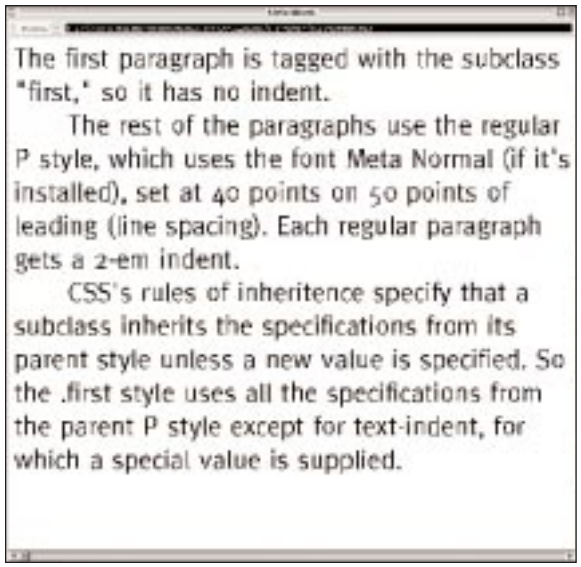
SEVERAL STYLE SHEETS can be combined in a single document. CSS's cascading order defines which style definitions take precedence when more than one is defined.



What Cascading Means

The name "cascading style sheets" comes from the way style sheets are applied to a document's elements. A certain element may have several styles associated with it. The browser uses a default style sheet for everything it displays. Site visitors may also specify a particular style sheet they like as their own default. The page author may import a companywide style sheet using the <LINK> tag, apply additional document-specific styles using the <STYLE> tag in the page heading, and then add special treatments to particular paragraphs or phrases in the document's body. The CSS specification spells out exactly which styles get priority in such cases.

As a rule, each style definition listed in the chart at left takes precedence over the one under it: The user's preference overrides the browser's default, and as a rule, the author's styles override the user's. (Users can override author styles by not accepting external style sheets or by naming some of their preferences as "important.") And more specific specifications override less specific sets.




```
<HEAD>
<TITLE>Style Sheet</TITLE>
<STYLE TYPE="text/css">
P       {
        font-family: Meta-Normal, Syntax, Helvetica, Arial;
        font-size: 40pt;
        text-indent: 2 em;
        text-align: justify;
        line-height: 50 pt;
        }
.first  {
         text-indent: 0 em;
         }

</STYLE>
</HEAD>

<BODY BGCOLOR="white">

<P CLASS="first">The first paragraph is tagged with the subclass “first,” so it
has no indent. . </P>

<P> The rest of the paragraphs use the regular P style, which uses the font Meta
Normal (if it’s installed), set at 40 points on 50 points of leading (line spacing).
Each regular paragraph gets a 2-em indent.</P>

<P> CSS’s rules of inheritence specify that a subclass inherits the specifications
from its parent style unless a new value is specified. So the  .first style uses all
the specifications from the parent P style except for text-indent, for which a spe-
cial value is supplied. </P>

</BODY>
```

THE STYLE SHEET SHOWN HERE creates a standard style for paragraphs and another for a special class of the P element (P.first) used for first paragraphs under headings. Notice the use of the *font* property, a shorthand notation you can use to combine several font speci-fications. Also notice CSS's inheritance rules at work here. The subclass (P.first) inherits all the specifications from the parent element (P), so only the differences need to be specified in the subclass's style definition.

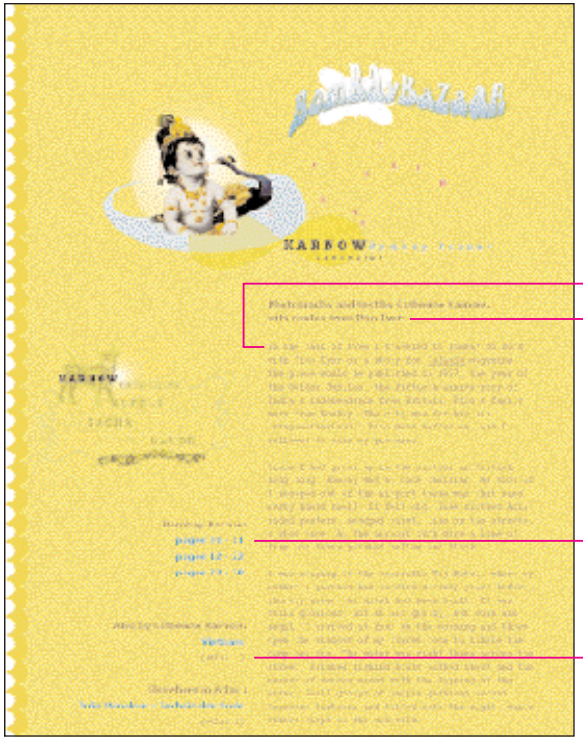
class
In HTML 4.0, a group of elements defined by the author. You add an element to a class using the *class=* attribute. With cascading style sheets, designers can assign layout attributes to all the members of a class.

inheritance
In CSS, the principle that elements use the same style properties as any element that contains them, unless those properties are specifically overridden.

single paragraph.) For style sheets to work, browsers must support those tags (introduced with HTML 3.2) as well as the style sheet language itself.

CSS coding is made somewhat simpler by the idea of **inheritance**. In CSS, an element inherits the style attributes of its parent element: for example, text tagged as <EM> would inherit the settings from the <P> (or other) element that contains it. A <P> element could inherit styles given to a <DIV> element above it, or even to the <HTML> tag itself. That way, a page author can define general page attributes (such as a standard typeface or page margin, for instance) just once, then add to or override those styles as needed for particular elements.

An especially powerful use of style sheets is the ability to go beyond the basic HTML tag set by creating element **classes**. Say you had a document for which you wanted to specify two types of paragraph styles: one for standard paragraphs, with an indent on the first line, and another, with no indent, for the first paragraph under a heading. You could tag each standard paragraph with the standard <P> style and create a special class of the <P> tag for the first paragraphs under heads, using the HTML 3.2 *class=* attribute (*<P class="first">*, for instance). Your style sheet could then provide styles for each type (as shown on the example on this page). In that way, you can essentially extend the HTML tag set indefinitely.

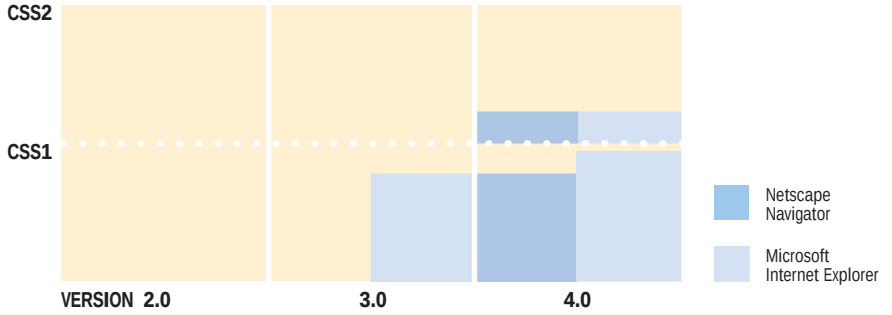


ATLAS MAGAZINE uses linked style sheets to apply consistent styles across the publication. Each element is named as part of a class, which has a corresponding style.

```
.text1   { font: 12px/16px 'Courier New', CourierNew, monospace; }

.text2   { font: bold 11px/16px Verdana, Arial, sans-serif; }

.text3   { font: bold 11px/16px Verdana, Arial, sans-serif;  color:
#94992b; }

.text4   { font: bold 11px Verdana, Arial, sans-serif; color: #94992b; posi-
tion: relative; top: 3px; }

.text5   { font: bold 11px/16px Verdana, Arial, sans-serif;  text-align: right;
margin-right: 24px; }

.caption { font: bold 11px/18pt Verdana, Arial, sans-serif;   margin-right:
24px; text-align: right; }

.cap2    { font: bold 10px/16px Verdana, Arial, sans-serif;  color:
#94992b; }

.debold  { font-weight: normal; }

A:link   { text-decoration: none; }
```

ATLAS MAGAZINE/CATHERINE KARNOW
http://www.atlasmagazine.com/photo/karnow6/index_C.html

BOTH NETSCAPE AND MICROSOFT support cascading style sheets in their browsers, beginning with Microsoft Internet Explorer 3.0 and Netscape Navigator 4.0. Current versions of both browsers support most of CSS1 and the parts of CSS2 having to do with CSS positioning.



IN CSS, EVERY ELEMENT is treated as if it were in a box. CSS properties can be used to control things like the box's size and the use of padding, borders, and margins. Other properties can be used to add background colors, scroll bars, and other features to the box, as if the box were its own frame.

```
<HTML>
<HEAD>
<TITLE>cssbox</TITLE>
<STYLE>

#square{

                background-color: green;
                padding: 15 px;
                border-width: 3 px;
                border-color: red;
                border-style: solid;
                margin-top: .5 in;
                margin-bottom: 40 px;
                margin-left: 40 px;
                margin-right: 2 in;
                width: auto;
                height: auto;
                clear: both

                }


</STYLE>

<BODY BGCOLOR="white">

<DIV ID="square">

Come back, the Caterpillar called after her. I've something important to
say!

</DIV>


<BLOCKQUOTE>This sounded promising, certainly. Alice turned and
came back again.</BLOCKQUOTE>


</BODY>
<HTML>
```

Boxes: The Layout Model of CSS

To begin to understand the layout capabilities of CSS, you must first understand that CSS treats each HTML element as if it were in a box. That means you can add things like background colors, borders, and other details to any element, just as if it were in its own table cell. CSS's box properties can be applied to any HTML element, allowing you to adjust the margin and "padding" around it as well as adding a border of any style or color. In addition, you can use CSS's color properties to not only color the type but also to add a background image or background color behind the element, filling the element's box.

By default, an element's box will grow to fit the content inside it, but you can also use the *width* and *height* properties to specify a certain size for the box. (If the size is too small for the box's content, a scroll bar can be made to appear, as if the element were in its own frame) (→**94**).

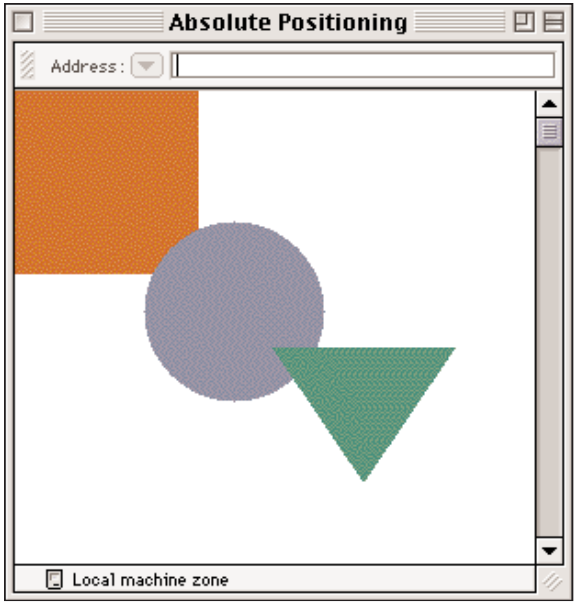
Understanding CSS boxes and the CSS properties that control them will let you fine-tune a page's measurements to an unprecedented degree. It will also help you understand CSS positioning, a way of using CSS to create complex, layered layouts, as we'll describe in the next section.

CSS Positioning

CSS positioning is not actually part of CSS1 but was introduced to the W3C as a separate proposal and was adopted by both Netscape and Microsoft in version 4.0 of their browsers. (The principles of CSS are built into CSS2.) With CSS positioning, you simply add CSS properties to tell the browser just how and where you want each element's box placed on screen.

The first decision to make is *how* you want the box positioned: with "absolute" or "relative" positioning. Absolute positioning places the element at a named coordinate in the browser window. Relative positioning positions it relative to its default position or to another element that contains it. Another positioning property, *float,* floats the element right or left, to the parent element's boundary, and lets text flow around it.

The position on the page is set with the *top* and *left* properties, measured from the top left of the browser window (or parent element) and described in terms of an x-y grid. (You can specify placement in just about any unit you wish: pixels, ems, points, picas, or percentages of window size.) You can also specify a **z-index**, or layering order, for the element, giving the item a position above or below other elements on the page. By default, the element is positioned wherever it falls in the flow of the document, with a box size large enough to contain the element's content, and a

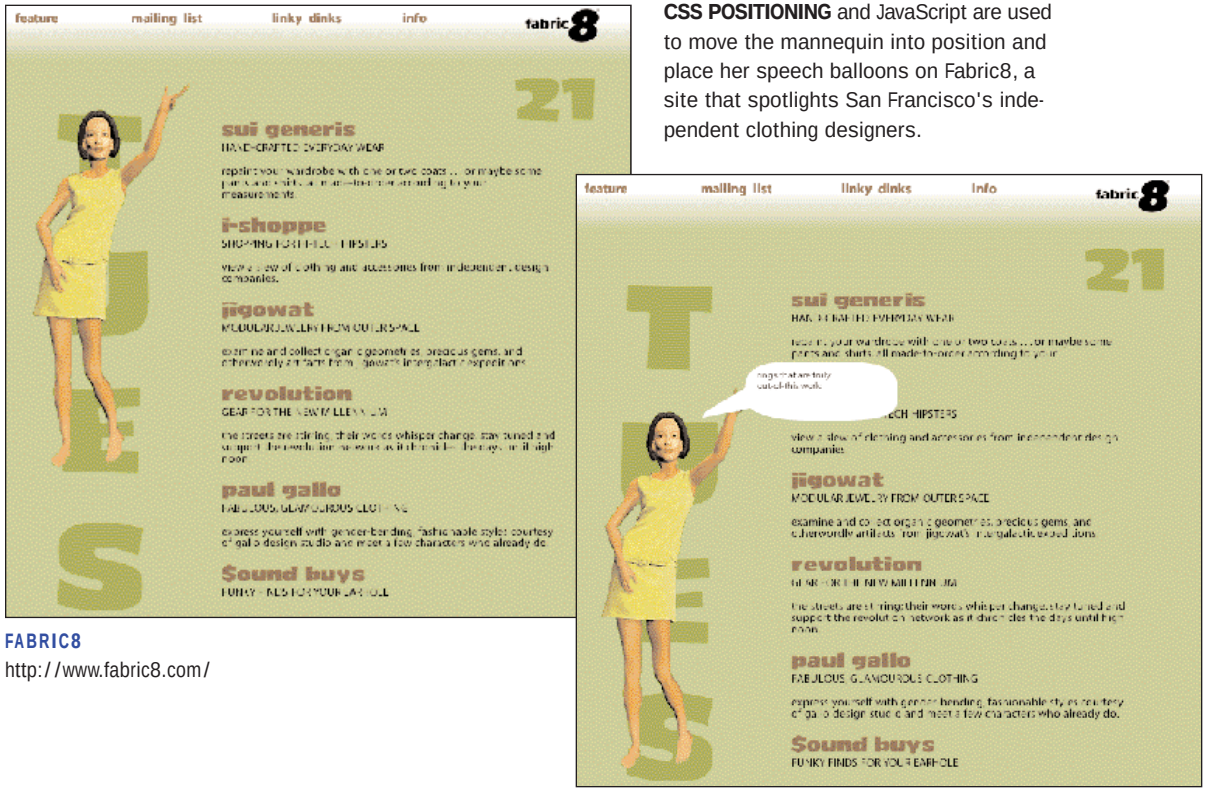


CSS POSITIONING lets you name the exact window coordinates for each element. The *top* and *left* properties name the window coordinates for each iitem. The *z-index=* property defines its layering position. The <DIV> tag separates the document into its component units, and its *id=* attribute provides a name by which each document section can be called from the style sheet.

```
<HEAD>
<TITLE>Absolute Positioning</TITLE>
<STYLE>
#square{ position: absolute; top: 0px; left: 0 px; z-index: 0; }

#circle{ position: absolute; top: 80px; left: 80 px; z-index: 1; }

#triangle{ position: absolute; top: 160px; left: 160 px; z-index: 2; }
</STYLE>
</HEAD>

<BODY BGCOLOR="white">

<DIV ID="square">
<IMG SRC= "square.gif" width="100" height="100"></DIV>

<DIV ID="circle">
<IMG SRC= "circle.gif" width="100" height="100"></DIV>

<DIV ID="triangle">
<IMG SRC= "triangle.gif" width="100" height="100"></DIV>

</BODY>
```

z-index
In CSS positioning, the layering order of an element. The term refers to the element's position in an x-y-z Cartesian coordinate system.

START TAG	ATTRIBUTES	END TAG	EXPLANATION
<DIV>		</DIV>	Groups the enclosed elements so that the attributes of the <DIV> tag apply to those elements
	align="left" OR "center" OR "right" OR "justify"		The alignment of the grouped elements
	id="*name*"		A name for the group. In CSS positioning, this name is used by the style sheet.
<SPAN>		</SPAN>	Groups a set of words inside a block-level element
	align="left" OR "center" OR "right" OR "justify"		The alignment of the grouped words
	id="*name*"		A name for the group. In CSS positioning, the name is used to identify the group in the style sheet.



CSS POSITIONING and JavaScript are used to move the mannequin into position and place her speech balloons on Fabric8, a site that spotlights San Francisco's independent clothing designers.

FABRIC8
http://www.fabric8.com/

```
<BODY BGCOLOR="#99CC66" TEXT="#000000" LINK="#663300"
VLINK="#669900" ALINK="#000000">

<DIV ID="lyrFeature" STYLE="position:absolute; left:10px; top:8px;
width:119px; height:30px; z-index:501" CLASS="nav">

<A HREF="feature.html" ONMOUSEOVER="rollover(1,'just one
of the many<BR>great products we offer',event); return true;"
onmouseout="rollover(0,'',event)">feature</A>

</DIV>

<DIV ID="lyrList" STYLE="position:absolute; left:130px; top:8px;
width:150px; height:30px; z-index:502" CLASS="nav">

<A HREF="list.html" ONMOUSEOVER="rollover(1,'join our<BR>
international scene',event)" ONMOUSEOUT="rollover(0,'',event)">
<NOBR>mailing list</NOBR></A>

</DIV>

<DIV ID="lyrLinks" STYLE="position:absolute; left:275px; top:8px;
width:150px; height:30px; z-index:503" CLASS="nav">

<A HREF="links.html" ONMOUSEOVER="rollover(1,'web styles we
dig',event)" ONMOUSEOUT="rollover(0,'',event)"><NOBR>linky
dinks</NOBR></A>

</DIV>
```

z-index depending on the order in which the elements are named in the file.

The next question is how you break up your document into the various layers that you'll place on the page. You'll need to group elements that you want to place together and give a name to each group so that you can identify it in your style sheet. Currently, the way you do this is with HTML's <DIV> or <SPAN> tags.

<DIV> and <SPAN> are useful whenever you want to assign an attribute to more than one item. The difference between them is that <DIV> (division) is used to group a set of elements (such as a group of paragraphs), while <SPAN> is used inside an element (to group a set of words). With CSS positioning, you use <DIV> and <SPAN> to define each group you want to position separately in your document. The tags' *id=* attribute lets you give each group a name by which you can address it from the style sheet.

CSS positioning is a cornerstone of dynamic HTML (→**76**), a group of technologies that lets Web designers control HTML elements in all sorts of new ways within the newest browsers. (In fact, CSS positioning is sometimes referred to as DHTML positioning.) In later chapters, we'll talk about how CSS positioning can be used with JavaScript to animate pages and make them interactive. Here, we'll just point out that CSS positioning can give you all the page layout controls of QuarkXPress or PageMaker. In the long run, it should also lead to dramatically simplified HTML coding, creating a clean and logical way to describe a page's layout, with no ambiguity and without using complex embedded tables and other kinds of troublesome, nonstandard HTML. Once it's better supported by browsers, CSS positioning should finally be the key to powerful, WYSIWYG page layout for the Web.

Now that you understand how a page's architecture can be constructed with CSS positioning, we'll talk about how CSS relates to other technologies for controlling a page's finer points: its typographic layout.

Online: Page Layout With HTML

Cascading Style Sheets and CSS Positioning
http://www.hotwired.com/webmonkey/stylesheets/
http://www.microsoft.com/workshop/author/default.asp#css
http://www.useit.com/alertbox/9707a.html
http://www.w3.org/Style/css/
http://www.webreview.com/guides/style/

Dynamic HTML
http://www.dhtmlzone.com/
http://www.hotwired.com/webmonkey/dynamic_html/
http://www.insidedhtml.com/
http://www.microsoft.com/workshop/author/dhtml
http://www.projectcool.com/developer/dynamic/
http://www.webdeveloper.com/categories/advhtml/
http://www.webreview.com/wr/pub/Dynamic_HTML/

Frames
http://www.cnet.com/Content/Builder/Authoriing/Frames/
http://www.projectcool.com/developer/alchemy/04-frames. html
http://www.webreference.com/dev/frames/

Tables
http://www.projectcool.com/developer/alchemy/03-tables. html

Dynamic HTML is the answer to a designer's prayers. Finally we can control placement to the pixel. We can create compelling and dynamic sites that aren't nightmarish to download. The ability to layer things has made me the happiest webgeek on earth. The fun has just begun.

ANNETTE LOUDON, CONSTRUCT