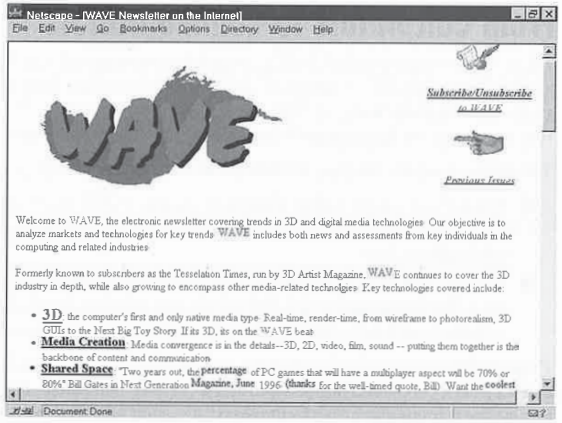

FIGURE 41.1.
HTML's raison d'etre is to provide nicely formatted information that is easy to navigate.

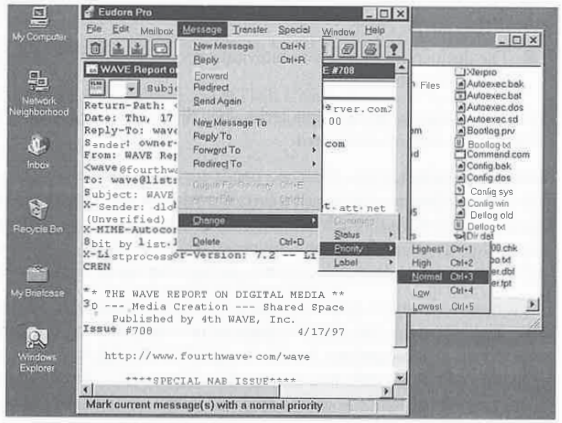

FIGURE 41.2.
Today's computer interfaces are simply too confusing and hard to learn for many of tomorrow's applications. (Note that the actual information content shown here is the same newsletter as in Figure 41.1.)

From the Desktop to the Conference Table

When personal computers first started appearing on desks, people bought them to help with the kind of tasks that take place on a desktop: reading, writing, calculating, and shuffling pages of information around. This reflected the business model of the mid-20th century: Isolated

workers and companies spent most of their time processing information in their own offices and a much smaller portion of their time directly communicating that information between individuals and companies.

As the century draws to a close, that business model is showing severe signs of strain. More and more time is spent in communication, and less and less in isolation. Workers who were trained in the older way of doing business complain that they now spend so much time attending meetings and conferences, or answering voice mail and e-mail, that no time is left to "get any work done." Those from the newer generation of employees and entrepreneurs, on the other hand, complain when coworkers or business associates create communication bottlenecks by being "out of touch" and "hard to reach."

The old desktop metaphor is increasingly useless for doing the most crucial tasks in a communications-oriented business world. As the personal computer becomes more of an interpersonal communicator, the interface must be more of a conference table than a desktop. Like a conference room, your computer is now primarily a place to exchange textual and graphical documents, audiovisual presentations, and verbal interaction. Unlike most conference rooms, your computer allows you to directly exchange information with hundreds of millions of people all over the world.

HTML as the New User Interface

As the role of the computer evolves, HTML is becoming more and more central to nearly everything we do with computers. HTML is *the* global standard for connecting all types of information together in a predictable and presentable way.

HTML provides a painless and reliable way to combine and arrange text, graphics, sound, video, and interactive programs. And unlike older proprietary page layout standards, it was designed from the start for efficient communication between all kinds of computers worldwide. At this point, the likelihood of any other standard unseating it as king of the communications hill seems remote.

The prominence of HTML, however, does *not* mean that Web browsers will be a major category of software application in the coming years. In fact, the Web browser as a distinct program has already nearly disappeared. Microsoft Internet Explorer 4.0, for instance, does much more than retrieve pages from the World Wide Web. It lets you use HTML pages as the interface for organizing and navigating through the information on your own computer, including directory folders and the Windows desktop itself. In conjunction with HTML-enabled software such as Office 97, HTML becomes the common standard interface for word processing, spreadsheets and databases as well. The new Netscape Communicator 4.0 is also much more than a Web browser. It uses HTML to integrate all types of media into e-mail, discussion groups, schedule management, business documents, and collaborative project management. (See Figure 41.3.)

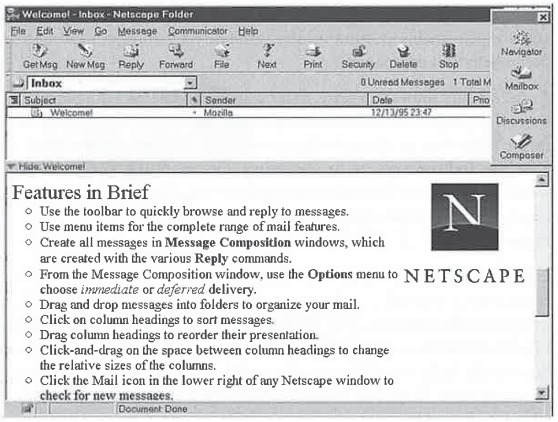

FIGURE 41.3.
Netscape Communicator 4.0 offers a vision of the future by integrating HTML into e-mail, discussion groups, and other business communication.

Meanwhile, HTML support is being included in every major software release so that every program on your computer will soon be able to import and export information in the form of HTML pages. In a nutshell, HTML is the glue that holds together all the diverse types of information on our computers and ensures that it can be presented in a standard way that will look the same to anyone in the world.

In a business world that now sees fast, effective communication as the most common and most important task of its workers, the information glue of HTML has the power to connect more than different types of media. It is the hidden adhesive that connects a business to its customers, and connects individual employees to form an efficient team. Knowing how to apply that glue—the skills you gained from this book—puts you in one of the most valuable roles in any modern organization.

The Digital Media Revolution

The most important changes in the next few years may not be in HTML itself, but in the audience you can reach with your HTML pages. Many Web site developers hope that Internet-based content will have enough appeal to become the mass-market successor to television and radio. Less optimistic observers note that the global communications network has a long way to go before it can even deliver television-quality video to most users.

I won't pretend to have a magic mirror that lets me see how and when HTML becomes a mass-market phenomenon. But one thing is certain: All communication industries, from television to telephony, are moving rapidly toward exclusively digital technology. As they do so, the lines between communication networks are blurring. New Internet protocols promise to optimize

multimedia transmissions at the same time that new protocols allow wireless "broadcasters" to support two-way interactive transmissions. The same small satellite dish can give you both Internet access and high-definition TV.

Add to this trend the fact that HTML is the only widely supported worldwide standard for combining text content with virtually any other form of digital media. Whatever surprising turns the future of digital communication takes, it's difficult to imagine that HTML won't be sitting in the driver's seat.

More than a million people can already access the Internet without a "real computer" through TV set-top boxes and from WebTV Inc., cable TV companies, and digital satellite services. These devices are only the first wave of much more ubiquitous appliances that provide HTML content to people who wouldn't otherwise use computers.

NOTE

The prospect of mass-market HTML access is obviously a great opportunity for HTML page authors. But it also presents a number of challenges when you are designing HTML pages because many people might see your pages on low-resolution TV screens or small hand-held devices. See the "What You Can Do Today to Be Ready for Tomorrow" section, later in this chapter, for some pointers on making sure your HTML pages can be enjoyed and understood by the widest possible audience.

Unity in Diversity

So far in this chapter, I've noted how HTML is in the right place at the right time to enable several key changes in business and interpersonal communication. As the people and companies of the world becomes more interconnected and interdependent, HTML's capability to make all information technology easier to use and less constrained by geography seems almost magical.

Even more magically, HTML has enabled an explosion of new media formats and incompatible file types while at the same time providing the first truly universal format for exchanging all types of information.

But there is no secret mystical force behind this apparent paradox. The power of HTML comes from a very intentional (and seemingly mundane) aspect of its design. Quite simply, HTML standardizes the format of the most common types of information while freely allowing unlimited special cases for proprietary formats and new technology. This means that you can both ensure complete compatibility between the widest variety of software and easily develop unique information formats to meet your individual needs.

Early multimedia Web sites were perhaps the first examples of how to meet these two (apparently conflicting) goals simultaneously. Text and graphics were visible to all visitors (as shown

in Figure 41.4), while Shockwave movies, Java applets, or other formats were used to provide additional audiovisual or interactive content for those who had the necessary plug-ins or helper applications (as shown in Figure 41.5). Though multimedia formats have come a long way toward standardization, you can still freely include proprietary file formats in ordinary HTML pages and use the helper app, Netscape plug-in, ActiveX object, or Java program of your choice to handle them.

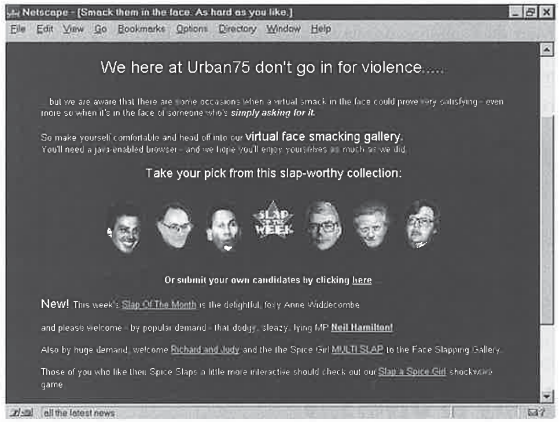

FIGURE 41.4.
This site uses HTML to present text and images that everyone can see.

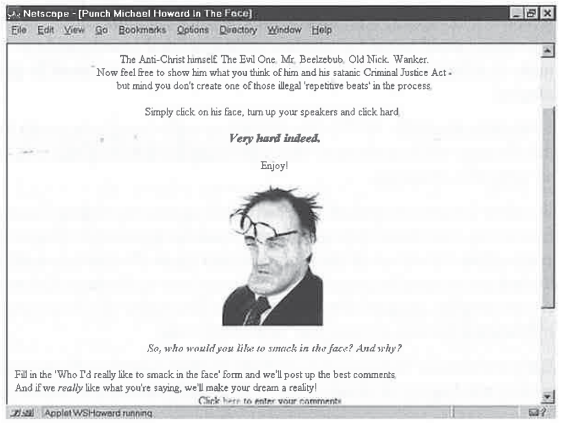

FIGURE 41.5.
If you explore the site shown in Figure 41.4 further, you'll find multimedia elements that require custom applets and/or plug-ins.

The ability to extend HTML pages with custom data types is far more than a way to embed a nifty movie or virtual reality scene into your Web page. To show how much more, the next two sections highlight some of the most exciting up-and-coming uses of HTML.

HTML as a Programming Language

The near-universal compatibility of HTML provides a big incentive to format any important document as a Web page, even if you have no immediate plans for putting it on the World Wide Web. You can create a single page that can be printed on paper, sent as an e-mail message, displayed during a board meeting presentation, and posted for reference on the company intranet. Or, you can take the traditional route and format the page separately for each of these applications—and edit each file with a different software program when the information needs to be updated. Now that most business software supports the HTML standard, many organizations are trying to get employees to consistently use it for all important documents.

But the great migration to HTML goes beyond what you might have thought of as "documents" in the old days. Combined with Java, ActiveX and other new technologies, HTML is being used for creating full-blown software applications that would traditionally have been written in more unwieldy languages such as C++.

As one example, consider M. Casco Associates, a small technical software company based in rural Pennsylvania. For years, the company has developed and published a line of interactive science simulation software for the educational market using the C and C++ languages. Figure 41.6 shows the Windows version of one of their programs, called Order: A Closer Look at Chaos. (The program uses interactive simulations to teach the basics of chaos theory.)

Because they are a small company, Casco could not afford to develop for more than one operating system or computer type at a time, even though their potential market included DOS, Windows, Macintosh, and UNIX users. Creating an effective interface to combine the tutorial and interactive elements of their software was a significant part of the development effort.

Figure 41.7 shows the latest version of their software, which was rewritten as interactive Java applets embedded in an HTML tutorial and guide. Not only does this new version eliminate much of the interface programming, but it is also instantly compatible with any type of computer that can run a Java-enabled Web browser. As a bonus, M. Casco was able to put part of the program directly on the World Wide Web for potential customers to evaluate without rewriting a line of code. (See for yourself at `http://www.mcasco.com`.) The small Java modules and HTML tutorial are also considerably easier to debug and maintain than a large C++ program requiring a recompile every time a change is made.

Transferring traditional programs to HTML-based presentations usually involves a shift in how you think about the user's experience. Essentially, instead of an application with a hypertext help system, you have a hypertext help system with an application inside it. This gives new users an obvious and friendly way to get to know a program, and still allows the HTML to step out of the way whenever the Java applets or other interactive media should be the focus of the user's attention.

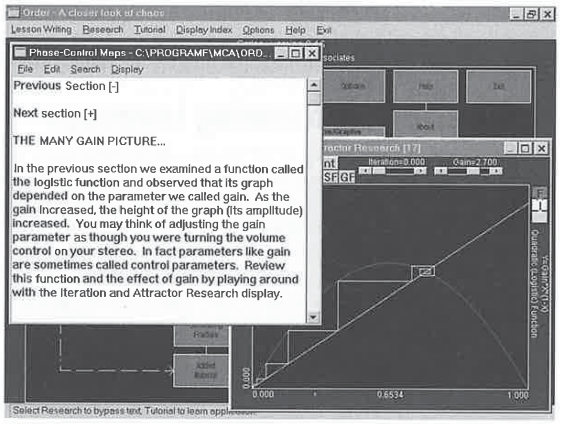

FIGURE 41.6.
This fairly sophisticated Windows program includes both interactive scientific simulations and an extensive online help tutorial.

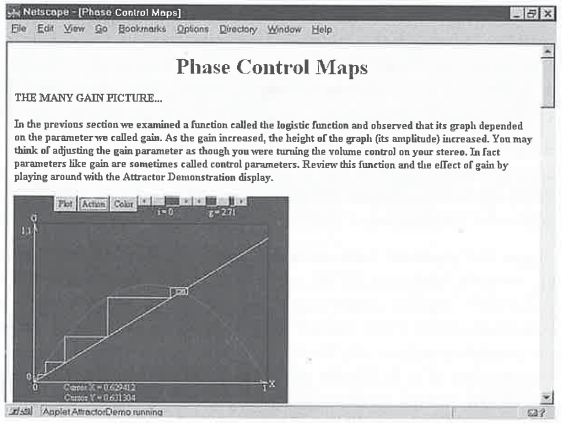

FIGURE 41.7.
With HTML and Java, all the functionality of the large C++ program in Figure 41.6 can be implemented in simple, cross-platform compatible parts.

HTML Applications of the Future

You've seen that HTML-based presentations can in many cases replace what was once done with proprietary data formats, specialized software, or more traditional programming languages. Here are a few of the other areas where HTML is finding application beyond the Web:

- Kiosks with HTML-based interactive content are popping up everywhere. They look like ATM machines on steroids, and they're helping sell records and theme park tickets, expand department store displays, and even automate the paying of parking tickets. The number of kiosks using intranet and/or HTML technology is projected to soar from under 90,000 today to more than 500,000 by the end of 1998.
- Information-rich CD-ROM titles are migrating to HTML fast. Encyclopaedia Britannica is already entirely HTML based, which enables them to offer their content on CD-ROM, the Web, or a combination of both for maximum speed and up-to-the-minute currency. Because CD-ROM drives display multimedia so much faster than most Internet connections, dynamic HTML presentations become possible that just couldn't be done on today's World Wide Web. The new DVD-ROM drives will be even faster and will hold much more information, making them ideally suited to large multimedia "sites."
- Corporate newsletters are now often created in HTML for the company intranet, and then printed on paper for delivery to employees or customers who wouldn't see them on the Web. The traditional difference between online and paper presentations was that graphics needed to be high-resolution black-and-white for printing and low-resolution color for computer screens. Today's inexpensive color printers, however, do a great job making low-res color images look great in an HTML-based newsletter.
- Teachers are finding that tests and educational worksheets are easier to administer as HTML pages and can include many types of interactive content that wouldn't be possible on paper. Even for students who lack access to the Internet, simple HTML documents can be passed out on floppy disks.
- Vertical market users often buy a computer specifically to run a certain custom-designed application or set of applications. The VARs and systems integrators that provide these systems are delivering machines configured to start up displaying HTML pages. This helps step users through the use of the machine, or replace old-fashioned "idiot menus" with a more attractive and sophisticated interface without sacrificing ease of use.
- Even game developers are getting into the HTML act. While some programmers still contend that HTML-based content will never be the best solution for calculation-intensive applications, others have proceeded to produce Java and ActiveX modules that do high-speed 3D graphics rendering (which is arguably the most intensive application of all) quite nicely. These engines are being used for both Web-based *virtual reality* worlds and cutting-edge entertainment titles to be distributed on CD-ROM.

I could list many more creative and beneficial uses of HTML beyond run-of-the-mill Web pages, but the point should be clear: If you need to present any type of information, seriously consider HTML as an alternative to the software or programming tools that you would have used a couple years ago for the job.

What You Can Do Today to Be Ready for Tomorrow

If you've made your way through most of the chapters of this book, you already have one of the most important ingredients for future success in the new digital world: a solid working knowledge of HTML.

Chances are that your primary reason for learning HTML at this time is to create some Web pages, but I hope this chapter has convinced you that you'll be using HTML for far more than that in the future. Here are some of the factors your should consider when planning and building your Web site today, so that it will also serve you well tomorrow:

- The multimedia and interactive portions of your site are likely to need more revisions to keep up with current technology than the text and graphics portions. When possible, keep the more cutting-edge parts of your site separate and take especially good care to document them well with the `<COMMENT>` tag.
- Though new technologies such as Java and Shockwave may be the wave of the future, avoid them today except when developing for disk-based media or a fast local intranet. Even when everyone is using the new 33Kbps and 56Kbps modems, many people still will move on to a different site before they'll wait for an applet or interactive movie to download, initialize, and start working.
- Because style sheets will soon give you complete control over the choice and measurements of type on your Web pages, it would be a good idea to study basic typography now if you aren't familiar with it. Understanding and working with things such as *leading, kerning, em spaces,* and *drop caps* have long been essential for producing truly professional quality paper pages. It will soon be essential for producing outstanding Web pages, too.
- One of the most popular and important features that will be added to many Web sites in the near future is interactive discussions and work groups. If you only have time to evaluate one new technology, that might be the one to pick. The new Netscape Communicator 4.0 package has especially strong support for group collaboration and communication.
- When you design your pages, don't assume that everyone who sees them will be using a computer. Televisions, video-telephones, game consoles, and many other devices may have access to them as well. Some of these devices have very low resolution screens (with as few as 320×200 pixels). Although it's difficult to design a Web page to look good at that resolution, you'll reach the widest possible audience if you do.
- Remember that a majority of people are still using versions 2.0 or 3.0 of Netscape Navigator and Microsoft Internet Explorer to view Web pages. If you use HTML 4.0 features such as layering and style sheets, make sure your pages still look good in browsers that don't support those features!

- Whenever you run into something that you'd like to do on a Web page, but can't with HTML as it stands today, include a comment in the page so you can add that feature when it becomes possible in the future.

Summary

This chapter provides a birds-eye view of the future of HTML and the new roles that HTML will play in global communications. It also outlines several applications of HTML that are likely to be much more common in the near future than they are today. Finally, it offers some advice for planning and constructing HTML pages today that will continue to serve you well into the future.

Glossary

by Dennis Báthory Kitsz

GLOSSARY

A

ActiveX: A technique developed by Microsoft to run applications on client Microsoft-based computers through an active Web page interface. Like plug-ins, once loaded, ActiveX remains part of the browser. *See also* **helper application, Java, plug-in, viewer**

agent: (*also called* autonomous agent, information agent, intelligent agent) An application or applet that carries the user's preferences into a virtual environment such as the Web and acts independently on those preferences. Infinitely configurable, an agent is the user's doppelgänger in cyberspace.

American National Standards Institute (ANSI): An organization dedicated to promoting national technology standards. ANSI is privately funded and includes both public and private members. *See also* **International Organization for Standardization**

American Standard Code for Information Interchange (ASCII): A code that represents a set of 256 unformatted characters—letters, numbers, symbols, and terminal actions (such as carriage returns and line feeds) in a standard format of 8 bits (although 128-character, 7-bit ASCII exists as well).

anchor: A portion of HTML code that identifies a specific document or a position (destination) inside a specific document. Also, the visible portion of the code, usually represented by underlined words. *See also* **hyperlink**

ANSI: *See* **American National Standards Institute**

applet: A dependent piece of computer software, usually small and compact, that requires an application with or within which to operate.

application: A complete piece of computer software, whether it is a compact utility consisting of a few bytes or a suite of programs that work together. An application stands alone inside its environment processor, operating system, browser, and so on.

ASCII: *See* **American Standard Code for Information Interchange**

attribute: Additional information added to element tags to specify characteristics of or render the element, such as size, color, borders, and other qualifications. Attributes are author-specified values.

authentication: A software technique that guarantees the identity of an individual or company; used in secure sites.

B

bandwidth: The capacity to carry data. In a network, this is usually bits per second and is critical for browsing speed, especially in sites heavy in hypermedia content.

bit: A binary digit, the smallest atom of digital information, represented by the number 0 or 1.

bitmap graphics: The resolution of an image into a series of dots, represented by a map of binary digits (bits). *See also* **raster graphics, vector graphics**

body: The "visible" content of an HTML document that a browser displays.

bookmark: A popular term for a hyperlink name and URL that the user saves for future use in a bookmark list or hotlist.

block element: A tag within an HTML document that identifies the document's structural areas by grouping text content in terms of its meaning. *See also* **container, text element**

bps: Bits per second. *See* **Kbps**

browser: Client software for displaying Web pages and using hyperlinks to navigate the Web.

byte: A combination of eight bits that constitutes a basic digital "word" size, capable of forming 256 possible combinations of bits.

C

C and C++: Advanced programming languages that dominate network applications. C is an older, structured language; C++ is its object-oriented successor.

cable-modem: A device that combines television and telephone technology to enable high-speed receipt of information via cable and normal-speed transmission via modem. The cable-modem is anticipated by content providers because it will make Web page (and hypermedia) downloads appear instantaneous while keeping data-thin and slower user interactions on ordinary phone lines. *See also* **modem, push**

cache: A temporary storage area for data. Specifically in terms of browsers, a cache is the trail of Web documents stored on the user's computer so that the user can instantaneously view a document again. Because some Web sites frequently update documents, caching can have the unwelcome side effect of holding on to "stale" pages.

cascading style sheets (CSS): A layout and display system for Web pages. The user can specify fonts and sizes, layouts, colors, and other viewing and printing parameters. Because CSS are optional and not part of HTML itself, they do not break HTML's platform independence for layout purposes. *See also* **frames, layers**

cell: A portion of a table in which information is displayed. A cell can encompass one or more of a table's rows or columns.

CERN: *See* **European Laboratory for Particle Physics**

CGI: *See* **common gateway interface**

character encoding: A numerical range of codes, repertoire of characters, and their mapping. *See also* **character set**

character set: The letters, numbers, and symbols available for display or printing. Character sets in HTML are defined to provide platform-independent consistency in all display systems for all languages.

chunk: A manageable piece of information. The definition is flexible but usually means the division of larger text into a single theme, concept, topic, or idea. Good "chunking" helps present nonlinear information.

class: The definition for a type of object in the form of a general software blueprint used in object-oriented programming. Classes may share elements with each other and may inherit characteristics of the parent blueprint. *See also* **inheritance**

client: A combination of computer and software capable of receiving information and instructions from a remote host computer or server.

client-side: Applications, activities, or events that take place on the client computer, such as image maps or applets.

code page: The character encoding developed for a specific language. *See also* **character encoding, character set**

common gateway interface (CGI): A programming interface between the client (browser) and server. CGI accepts data from the user and acts on it, sometimes returning information to the user. Common examples are forms and searches.

compression: The reduction of raw data by removing redundancies, making predictions, and describing events, rather than replicating them. This allows for smaller files and faster downloads but, especially in the case of audio, noticeable degradation.

container: An HTML element made up of an opening and closing tag. Information is contained within the tags. *See also* **block element, text element**

content: The information a Web page provides. Content is often highly debated and often redefined to defend a Web page creator's design. Web purists largely view content as text.

cookie: A line of text stored in the client browser that the server can retrieve. It may contain encoded passwords, shopping-cart contents, user identification, and so on. Cookies are controversial because commercial Web designers use them extensively to track visitors.

copyright: An international treaty setting the right to intellectual property. Although copyright exists the moment an idea is manifested as a work of authorship (in any medium), the nature of the Web culture has not taken copyright into account until recently.

Cougar: The latest W3C proposed draft for HTML specifications that distinguish between markup (content) and appearance of Web pages. Like most HTML specifications, it lags behind and attempts to codify existing practice into universally accessible definitions and incorporates significant access techniques for hypermedia. Cougar may become HTML 4.0, succeeding the present 3.2 specification.

CSS: *See* **cascading style sheets**

D

database: An organized collection of information stored on a server.

declaration: A statement about a document and its structure, including its type, syntax, scope, character set, entities, naming rules, and so on.

dialog box: A box displayed on the screen that asks for user selections. The dialog box is a core component of a graphical user interface.

dialup: The method of accessing the Internet via modem and traditional telephone lines.

DNS: *See* **Domain Name Service**

document type definition (DTD): The hierarchy of elements used to define the structure of a document. The DTD is set forth in the HTML specification.

domain name: A name for an organization, business, or government agency expressed in word-like form. There are national domains and six major organization domain types (`.com`, `.edu`, `.org`, `.gov`, `.mil`, and `.net`). Combined with a server name, they are translated by domain name services into a numeric IP address called the "dotted quad" (such as 204.246.11.197).

Domain Name Service (DNS): A computer containing a cross-reference of domain names to respective numerical IP addresses.

DTD: *See* **document type definition**

dynamic: In Web parlance, particularly in terms of Web pages, dynamic refers to server responses created based on active user input (such as forms) or passive user input (such as browser identification or cookies).

dynamic HTML: A method that Microsoft promotes to produce pages with active or changing information.

E

element: Though commonly referred to as a tag, an element is in fact a component of hierarchical structure defined by a document type definition. The element is actually described (or delimited) by the tag. This clarity is obscured, however, by elements without contents (such as line breaks), and those that may not contain other elements (such as anchors). *See also* **block element, text element**

embedded object: An object included as part of a document in its complete form, such as an illustration, audio file, or application. *See* **object**

encoding (character): *See* **character encoding**

encryption: A method of altering information to make it unreadable by anyone except users who possess the rules to decode it. Public-key encryption is popular because anyone can encrypt and send a message, but only the recipient can decode it.

entity: A character that cannot be typed on the keyboard (such as é or ©) or that has a special meaning to HTML (such as <). It is represented by its numeric form (called a character reference: `©` for ©) or mnemonic form (`©` for ©).

European Laboratory for Particle Physics (CERN): *Conseil Européan pour la Recherche Nucléaire.* This laboratory in Geneva was the birthplace of the Web concept and execution in 1989.

event: An incident or occurrence meaningful to a program, such as an operation's completion, a request for data, or an interruption of activity.

event handler: A portion of a program that responds to an event by processing its details, such as a mouse click, keyboard stroke, or "out of paper" signal.

Extensible Markup Language (XML): A subset of SGML that parses a document's structure and elements in terms of its document type definition.

extension: (1) The method of identifying the purpose of files and applications within the filename, following a dot. Common to the Web as well as UNIX and DOS operating systems, extensions can identify files such as HTML (`index.html`), MIDI (`song.mid`), images (`picture.jpg`), and so on. File extensions are not found in some operating systems, such as Macintosh.

extension: (2) Additions to HTML outside the existing specifications. Companies often contribute, encourage, promote, and even implement these extensions to advance features they would like to include in future releases of the HTML specification.

extranet: Internal web sites (intranet sites) accessible to the outside world, usually via passwords. *See also* **intranet**

e-zine: An online electronic publication, specifically (like a print 'zine) an amateur, artistic, or non-mainstream publication and presentation.

F

fat: In reference to a client, *fat* is a general term indicating a machine that is a complete, standalone computer with an extensive operating system and powerful applications. Although such a computer can access a server, it does not require one.

file path: *See* **path**

file transfer protocol (ftp): The method for remote login to a computer to exchange files over a TCP/IP network. Ftp usually requires a username and password, but anonymous ftp can also be made available, especially with Web browsers.

firewall: Protective software that prevents unauthorized users or software from entering an internal network. It is used to protect theft of data and to keep out viruses but has the side effect of making Web use difficult, particularly for hypermedia.

font: A set of typographical characters (numbers, letters, and symbols). A complete Western (Latin) font, for example, includes upper- and lowercase letters as well as a full range of accented characters. Special software fonts such as Cherokee have helped sustain and expand their respective cultures. *See also* **kerning**, **monospaced font**, **proportional font**

form: An HTML component that enables Web authors to have input fields on their pages, permitting feedback from users and offering interactive options.

format: The presentation of a document, including fonts, voice, paging, colors, and other features. Format is also achieved by usage (such as paragraphs, lists, and titles) and logical elements. *Compare* **cascading style sheets**, **layout**, **structure**

frames: Valuable but confounding extensions to HTML that present information as a concurrently displayed group of pages, each with a unique URL but concatenated by a master document. Frames are a kind of simultaneous hypertext. Because they are visually organized, they are not considered platform-independent and are the subject of bitter controversy. *See also* **cascading style sheets**, **layers**

ftp: *See* **file transfer protocol**

G

GIF: *See* **Graphics Interchange Format**

glyph: A character in an ideographic writing system, usually not phonetically based. Ideographic systems can use tens of thousands of unique glyphs, as opposed to the hundred or so in phonetic systems.

gopher: An early method of accessing information on the Internet consisting of document indexes, file transfer, database browsing, and Telnet.

graphical user interface (GUI): Generally, an operating environment in which icons represent files and applications on a monitor display, and a point-and-click device (such as a mouse, trackball, or pen) is used to activate them. GUIs usually employ drag-and-drop file movement and launching applications, plus buttons, toolbars, drop-down menus, and other conventions not available with line-based (keyboard) methods.

Graphics Interchange Format (GIF): An image compression method CompuServe developed and the first that Web browsers accommodated. Files are large and the palette is limited to 256 colors, but no information is lost in the compression.

GUI: *See* **graphical user interface**

H

handheld personal computer (HPC): A complete computer in a small package, with a limited display. The HPC and its kin concern Web page authors because the small display and limited features call for HTML's platform independence.

head: Part of the HTML document containing information about the document itself, including the document title, searchability, keywords, and other contents including scripts.

header: The client sends the HTTP request header to the server to indicate what information it wants. Among those requests are document title, body, authorizations, contents, MIME types, and interaction with CGI.

helper application: A program that is not a native part of a Web browser nor a software plug-in provided for the browser. Helper applications are used to display non-text content, particularly newly developed or custom multimedia content, that goes beyond the browser's capabilities at the time of its release. *See also* **ActiveX, Java, plug-in, viewer**

hex (hexadecimal) notation: An effective method of representing binary numbers in systems with a 4-bit architectural heritage. Four binary digits (0000 through 1111) can be represented by hexadecimal numbers 0 through F; eight digits, 00 through FF; sixteen digits, 0000 through FFFF, and so on.

hit: A visit to a Web page or, sometimes, to each file on that page.

host: A server. In Web terms, a computer with a specific name on the Internet. *See also* **node**

hotspot: The portion of an image that is mapped to a hyperlink.

HPC: *See* **handheld personal computer**

HTML: *See* **HyperText Markup Language**

HTTP: *See* **HyperText Transfer Protocol**

HyperCard: The first practical implementation of hypertext theory in the Apple computer.

hyperlink: A connection between one file and another file or inside a file. It is one of the HTML anchors.

hypermedia: Multimedia as presented on the Web, including passive and active graphics, audio, video, virtual reality, and other media concepts under development.

HyperTalk: The scripting language used to create Apple HyperCard documents. *See also* **HyperCard, script**

hypertext: The practical and conceptual heart of the Web, hypertext is a system of relating points within and outside a text to each other nonlinearly. Hypertext, as manifested with hyperlinks, moves a user from text to text at will.

HyperText Markup Language (HTML): A platform-independent method of identifying a document's structure and references. In its ideal form, HTML makes no reference to the ultimate user's hardware; in practice, HTML includes presentation information as well as structural elements.

HyperText Transfer Protocol (HTTP): A protocol for transferring data formats between a server and client. Data formats include plain text, hypertext, images, sound, public or proprietary formats specified as a MIME type, and metainformation about the data.

I

icon: A visual representation of a file or application.

image (IMG): Originally created to include visual images on a Web page, the IMG tag has expanded to include many types of hypermedia, including sound, video, and interactivity.

image map (imagemap): An image with hotspots that serve as mouse-activated hyperlinks.

inheritance: Characteristics of a class shared with a parent or child. *See* **class**

integrated services digital network (ISDN): Digital communications standards for voice, data, and video over standard phone lines at 128 Kbps. It is better established in Germany and is developing slowly in the U.S. because of anticipation of cable-modem services.

intellectual property: The ownership of an idea's manifestation—that is, a work put into communicable form. The communicable form can include paper, electronic media, or stone, and encompasses all forms of authorship. It has been the subject of debate and revisions to the copyright treaties because of the "Wild West" nature of the Web. *See also* **copyright, watermark**

interface: A generic term for the method used to join hardware components, software components, or users and systems. For example, a user interface is a concept and method of interaction; a printer interface consists of cabling, electronic components, and software drivers; a data interface is software to translate data formats; an equestrian interface is a saddle and reins.

International Organization for Standardization (ISO): Founded in 1946, this voluntary, nontreaty organization creates international standards and definitions. These standards apply to both hard and soft technologies, methods, and contents, and they are not limited to any particular group of fields or subjects. *See also* **American National Standards Institute**

Internet: A worldwide network of computers in communication using agreed-on protocols. A simple definition does not explain the Internet as a *concept*, but it is an opportunity for universal access to human knowledge, wherever it is distributed. *See also* **nonlinear, World Wide Web**

Internet Explorer: A Microsoft Web browser that gained popularity by being bundled with the Windows 95 and NT 4.0 operating systems.

Internet Protocol (IP): A system of network connections using numerical (IP) addresses. No direct connection between the sending and receiving computer is assumed; the IP address is used to establish an indirect pathway through multiple networks.

Internet service provider (ISP): A business that offers connection to the Internet, usually through dialup or dedicated telephone lines. ISPs can be little more than a local server, a full-featured system of various accounts including server Web space, or a commercial provider with proprietary features and chatrooms along with Internet access.

intranet: A network internal to an organization or corporation set up to mimic Internet features. Intranets may also connect to the Internet.

IP: *See* **Internet Protocol**

IP address: A quartet of digits separated by dots (such as `204.246.11.197`) that represents a unique computer connected to the Internet. Domain names are resolved to IP addresses before information is exchanged.

ISDN: *See* **integrated services digital network**

ISO: *See* **International Organization for Standardization.** (ISO is actually not an acronym, but the prefix for the standards, derived from the Greek prefix *iso*, meaning "same.")

ISP: *See* **Internet service provider**

J

Java: A programming language Sun Microsystems developed to permit standalone applications to run on a client computer through an active Web page interface on a "Java virtual machine." *See also* **ActiveX**, **helper application**, **plug-in**, **viewer**

JavaScript: A scripting language that is interpreted and runs on a client computer within an HTML document and provides active and interactive applets to Web pages.

JavaScript style sheets: A page formatting method that Netscape promotes and JavaScript controls. *See also* **cascading style sheets**

Joint Photographic Experts Group Format (JPEG): A method of compressing images; the second such method adopted to display images on the Web. The palette is large (16 million colors) and file sizes can be made small with JPEG's variable compression, but the method is *lossy*, meaning that the original image's detail is not fully restored during decompression. It is used largely for photographs, where the eye makes up for compression losses.

JPEG *See* **Joint Photographic Experts Group Format**

K

Kbps: A thousand bits per second, often used in reference to Internet connection speeds. Early modem speeds were 300 bps (bits per second), whereas 28.8 Kbps is common today, with compression that squeezes an effective rate of 34 Kbps through the line. Other frequently encountered speeds are ISDN at 128 Kbps, T1 connections at 1.5 Mbps (million bits per second), T2 at 3 Mbps, and T3 at 45 Mbps. *See also* **bps**, **Mbps**

kerning: A typographical technique that overlaps characters for visual balance.

L

layers: A Netscape-specific method of organizing multiple documents on a page, including layout features and identification of viewable content. Layers are related to frames but offer more author control. *See also* **cascading style sheets**, **frames**

layout: The visual appearance of a document. Because HTML identifies a document's structure rather than its appearance, Web authors have sought techniques to make their pages display to their tastes. Although some HTML workarounds have been developed, along with display elements within the HTML specification itself, advanced layout calls for other methods, including images and cascading style sheets. *Compare* **cascading style sheets**, **format**, **structure**

LCD: *See* **liquid crystal display**

leading: The vertical space between lines of text.

linear: A concept of organizing information in a known sequence. With few exceptions and until recently, this has been the method of storing, retrieving, and exchanging information throughout the history of the written word. Linear information is considered "self-contextualizing" because the references appear in a past, present, and future sequence.

link: *See* **hyperlink**

Linux: An operating system based on UNIX and created as freeware. Many small Internet servers run Linux, which independent programmers continue to maintain.

liquid crystal display (LCD): Low-power display hardware in which a liquid becomes semi-opaque in the presence of an electrical current.

listserv: An automated mailing list for exchanging messages. Documents sent to a listserv are redistributed to all its subscribers.

LiveScript: *See* **JavaScript**

logical markup: The identification of a text's characteristics, content, and structure. *Compare* **physical markup**

Lynx: A text-only browser found on older or slower computers, sometimes used for high-speed surfing or text-to-speech access. It has a hardcore following and is also useful for testing both the platform independence and the potential ambiguity (or complete opaqueness) of a Web page design.

M

map: *See* **image map**

markup: The identification of a text's characteristics, content, structure, and presentation by use of language-like descriptions. *See also* **logical markup, physical markup**

Mbps: Million bits per second. *See* **Kbps**

menu: A list of choices. On Web pages, menus can be presented as hyperlinks, drop lists, checkbox groups, maps, and even hypermedia controls.

MIDI: *See* **Musical Instrument Digital Interface**

MIME: *See* **Multipurpose Internet Mail Extensions**

modem: *Modulator-demodulator.* A device that converts digital information into a sound format that can be transmitted over acoustic phone lines.

monospaced font: Characters displayed or printed in equally spaced "typewriter" format.

Moore's Law: Originally put forward in 1965 by Intel co-founder Gordon Moore, it states that computer power doubles every 18 months. It has held true for 30 years. Software and hardware creators have exploited technological developments to the chagrin of budget-conscious users and have presented a formidable barrier to those concerned with universal access to computing power.

Mosaic: The first successful graphical browser, created in 1993 at the NCSA.

Motion Picture Experts Group (MPEG): Video and audio compression standards. There are presently three MPEG standards with several "layers" each.

Mozilla: From "Mosaic killer," the original prototype name for Netscape Navigator.

MPEG: *See* **Motion Picture Experts Group**

multimedia: A term expropriated from the art world (where it means a performance or artwork including several human senses), multimedia is commonly used to mean the integration of at least audio and video elements into a computer presentation. Its Net equivalent (*See* **hypermedia**) is the bane of pure HTML advocates because it does not offer platform independence.

Multipurpose Internet Mail Extensions (MIME): A method of encoding binary data for electronic mail transmission, including audio, graphics, video, word processing, music scores, and applications. MIME was also adopted as the standard for Web transmission.

Musical Instrument Digital Interface (MIDI *or* Midi): The specification of sonic parameters (such as note, volume, duration, and controllers) as a serial stream of data and the specification of the required connectivity. MIDI stepped out of the production studio onto the Web because of both the compact size of the files and its platform independence.

N

National Center for Supercomputing Applications (NCSA): The University of Illinois at Urbana-Champaign center responsible for developing the original Mosaic browser.

navigate: To move through the Web using hyperlinks. Also, to move through Web pages using hyperlinks, droplists, toolbars, maps, hypermedia controls, and other techniques.

NC: *See* **Network Computer**

NCSA: *See* **National Center for Supercomputing Applications**

Netscape Navigator: Web browser created by NCSA expatriates (*See* **Mozilla**), quickly rising to popularity by a combination of aggressive giveaway marketing and the promotion of presentation-oriented Web pages, graphics, sound, and other hypermedia and a plug-in structure to accommodate new forms of media.

Network Computer (NC): A simplified computer (sometimes called a *thin client*) whose applications and files are stored on one or more servers, rather than locally, and without the features of a full operating system.

news: *See* **Usenet**

node: (1) The point of entrance to or exit from a Web page, considered a standalone construct.

node: (2) A computer connected to a network capable of generating, receiving, or forwarding data.

nonlinear: A concept of storing, obtaining, and exchanging information through multiple, indirect, layered, and sometimes unpredictable paths with no predefined order. Human thinking frequently follows such paths, but until modern technology made it possible, media representation of nonlinear information was virtually unknown outside its rudimentary use in printed footnotes or operatic motifs. It is now exemplified by the Web's hyperlinks.

O

object: A self-contained piece of information ranging from a single byte to a complete document or application.

object-oriented: Programming and databases using self-contained and interchangeable software routines or data elements rather than linear (and often custom) coding. Focusing on ways of interacting with data, object orientation is based on building-block thinking.

ODBC: *See* **Open Database Connectivity**

Open Database Connectivity (ODBC): Microsoft standard for exchanging software components and data within applications.

operating system (OS): The software control system for a computer. It usually refers to a disk-based system, although single-board and harsh-environment computers may use ROM-based operating systems. The operating system may have several layers to create an environment (such as older Windows operating through DOS) or work directly with the computer (such as the Macintosh, Windows 95, or UNIX).

OS: *See* **operating system**

P

page: A Web document in HTML with a single URL. Pages are usually *chunks* (*See* **chunk**), but in the case of linear information, pages can sometimes be long, text-like documents or lists of links.

parameter: A range of possibilities or actions in an application.

parse: To examine code or markup for underlying structure, grammar, and syntax; to determine whether errors are present; and to prepare the results for an action—page presentation in the case of HTML.

password: String of characters known only to a system or application and its user. Passwords are a simple method of keeping information confidential, but their proliferation has complicated lives for frequent computer users. Authentication methods have been proposed to replace passwords in some situations.

path: The trail through a computer's directory structure to a document.

PDA: *See* **personal digital assistant**

Perl: *See* **Practical Extraction and Report Language**

personal digital assistant (PDA): A small, handheld, computer device with a subset of the features found in larger operating systems. PDAs with Web-browsing capabilities present a challenge for page designers and HTML authors because of their small monochrome displays.

physical markup: The markup of a document's presentation. Physical markup is consistent with standard HTML practice in which it contributes to structure and meaning as identified in the typographical world. This remains a contentious issue, as in the case of emphasized and strong text versus italic and bold text. Some recognize the former as creating well-formed HTML with a structural meaning; others recognize the latter as creating well-formed HTML with typographical meaning.

PICS: *See* **Platform for Internet Content Selection**

pixel: From *picture element*, the pixel is the basic element on a raster display (*See* **raster graphics**), represented as a dot containing intensity and color information.

platform: The essential combination of hardware (microprocessor, architecture, and memory organization) and software (operating system and applications) that makes up a computer. Because hundreds of platforms exist beyond the well-known DOS, Windows, Macintosh, and UNIX systems, the HTML author strives for platform independence.

Platform for Internet Content Selection (PICS): Proposed standard that gives parents the technical means to block indecent material from children. It defines file formats for independent content rating, and the published results can be used by PICS-enabled software.

platform independence: The capability of applications or data to be used on any computer. Never actually achieved, platform independence is attempted by HTML by identifying a document's structure and liberating a document's elements from their presentation.

plug-in: A dependent application that extends a browser's capability to present advanced or custom content without changing the browser itself. Originally developed as helper applications, plug-ins are now provided by media content developers for the most popular browsers. *See also* **ActiveX**, **helper application**, **Java**, **viewer**

PNG: *See* **Portable Network Graphic**

port: A hardware or software entry point to a computer system, largely software ports in Internet protocol terms (HTTP uses port 80, Telnet port 23, IRC port 6667, and so on).

Portable Network Graphic (PNG): Compressed graphics format with a large color palette and low information loss. PNG is a public-domain technique that has been proposed as a replacement for GIF. *See also* **Graphics Interchange Format**

Practical Extraction and Report Language (Perl): The most popular language for programming CGI. *See* **common gateway interface**

proportional font: Characters presented according to their width rather than equally, as exemplified by handwriting and typesetting. Proportional fonts have become the graphics interface standard, but monospaced fonts are still used in terminals and for special circumstances.

protocol: A standard for network communication. Different protocols exist for unique purposes: HTTP moves Web documents, FTP transfers files, TCP manages network traffic, IP handles data packets. A group of simultaneously operating protocols is a stack. Protocols are also used for communications with peripheral devices such as printers.

proxy gateway: Intermediary computer and software that pass data between an isolated computer and the outside world. Proxy gateways can be configured to pass only "safe" information to the client. *See also* **firewall**

proxy server: A computer that stores and distributes documents it has obtained from other servers. This technique is used to lower access costs or keep frequently used documents at hand. One disadvantage is that updated Web pages might not be available to clients until the proxy server is next updated. *See also* **cache**

pull: Actions initiated by a client. Most browser applications request, or *pull*, information from the server.

push: Actions initiated by a server, especially for sending updated information to a client browser, such as news, stock quotes, and live video.

Q

QuickTime: Proprietary Apple format for audio and video frequently found in CD-ROM applications and occasionally found on the Web. *See also* **MPEG**

R

RAM: *Random-access memory.* Advanced clients require large amounts of high-speed RAM to handle graphics, sound, and video without requiring frequent access to a hard disk.

raster graphics: A method of drawing images by altering the intensity and color of an electron beam sweeping across video display terminal phosphors in a regular pattern of horizontal lines—the raster. It is an established television-style technology.

RealAudio: A popular audio compression system capable of streaming audio on the Internet. RealAudio offers free encoders and decoders but charges for the live-streaming servers. RealAudio automatically detects a computer's Internet connection speed and configures itself for highest audio quality.

Request for Comments (RFC): Published documents that include approved and proposed Internet standards, as well as agreed-upon plans and concepts. They are independently proposed by technical experts and remain RFCs even if they become standards.

RFC: *See* **Request for Comments**

robot: Automated software that performs tasks independent of users, often said to "live online." Also called a bot or an agent. *See also* **agent**

S

script: A written set of instructions for performing tasks, a script is usually application-specific commands that customize an application's operations. Scripts are found in most Internet applications, with examples found in the Web's JavaScript, in dialer scripts, IRC scripts, server scripts, and so on.

scrolling: The movement of screen information horizontally and vertically. Scrolling is important to Web authors both as a convenience issue (vertical scrolling is expected, horizontal scrolling is considered inconvenient) or access issue (sometimes, as with WebTV, horizontal scrolling is impossible).

search: To use Web-based, interactive tools to locate specific information or Web sites; it is the opposite of surfing.

Secured HyperText Transfer Protocol (SHTTP): An HTTP protocol in which an encrypted connection is established before data is exchanged.

security: The preservation of privacy in data exchanges. Security usually calls for encrypting data and authenticating users and may also include passwords.

Sequential Query Language (SQL): Database-access and database-retrieval language used for searching large quantities of information and answering questions; pronounced "sequel." Other popular database techniques are Informix, Oracle, and Sybase.

server: A combination of computer, software, and network connection that acts as a central location and acts on requests for resources, such as applications, Web pages, or other data. It also processes responses to interactive pages, such as forms or searchers.

server-side: Applications, activities, or events that take place on the host computer or server, where information and responses are processed.

SGML: *See* **Standard Generalized Markup Language**

ShockWave: A popular browser plug-in with which to create animation and sound and, recently, other hypermedia content. Newer versions of ShockWave use streaming techniques. *See also* **streaming**

shopping cart: A software method to keep track of online purchases. Shopping carts are difficult to implement because the server cannot know the browser's location on the Web. Web authors resort to cookies, scripts, and other techniques to maintain shopping carts.

SHTTP: *See* **Secured HyperText Transfer Protocol**

site: *See* **Web site**

socket: A method for making a "virtual connection" between software processes by identifying data type, direction, and port. Originally developed for UNIX, it is now used by other systems including Windows and Macintosh.

spider: A searching robot that explores and indexes the contents of all local hyperlinks from a given Web page.

SQL: *See* **Sequential Query Language**

Standard Generalized Markup Language (SGML): An international standard for text information processing, distribution, search, and retrieval. SGML documents have content and structural information about content. Format is not specified in SGML.

stream: A continuous flow of information.

streaming: The use of a data stream to present time-based content (such as audio or video) as it happens, without waiting for a complete file to download. Streaming can be used with existing complete documents or "live" from digital information created on the fly.

structure: The organization of a document. In HTML terms, structure identifies the markup of consistent elements that identify their purpose; the definition of its content. Structuring a document frees the viewer to determine its use, particularly in a nonlinear, hypertext situation. *Compare* **format**, **layout**

style sheets: *See* **cascading style sheets**

surf: To browse or navigate Web sites without a specific informational goal in mind. *Compare* **search**

T

table: Originally conceived as a way of marking up tabular information, the table has evolved into a document presentation element. Using borderless tables, Web designers achieve layout-like results in graphical browsers. The use of tables for design is controversial because they are used to format a display rather than to provide document structure.

tag: Markup instruction in an HTML document presented in angled brackets. A tag may contain a declaration, statement, and so on, as well as attributes.

TCP: *See* **Transport Control Protocol**

TCP/IP: *See* **Transmission Control Protocol/Internet Protocol**

Telnet: Remote login to a computer that enables a user to use a computer from a distance, including manipulating files and executing applications.

template: A document or file containing generic information used to build new or updated documents by "filling in the blanks" or altering variable information. Templates are particularly useful for Web sites that update information frequently, maintain an identical presentation and structure, and archive it all (such as online publications, reviews, and directories).

terminal: The interactive user end of a network, usually only a display, keyboard, and interface, but sometimes generally referring to any interactive end-user point on a network, including full-blown computers or, in earlier days, Teletype machines.

text element: A tag within an HTML document that identifies the characteristics of words or characters, rather than the structure of the content. *See also* **block element**, **container**

thin: In reference to a client, *thin* is a general term that indicates a machine without the computing capability of a complete standalone computer with an extensive operating system and powerful applications. Most of its power comes from a server.

toolbar: (*sometimes* **navigation bar**) A set of icons that are hyperlinks to other Web pages or hypermedia.

Transport Control Protocol (TCP): A means by which a connection is established between a client and a server. Data is broken into packets that are addressed to the receiving computer together with information for their reassembly.

Transmission Control Protocol/Internet Protocol (TCP/IP): The combination of client/server connection and data transfer (TCP) and network connection system (IP) used on the Internet. *See* **Transport Control Protocol**, **Internet Protocol**

transportability: *See* **platform independence**

U

UCS: *See* **Universal Multiple-Octet Coded Character Set**

UCS Transformation Format: A method of switching between standard ASCII and UCS character sets within a serial stream of data.

Unicode: 16-bit character encoding system designed to display phonetic and ideographic languages. Although incomplete, Unicode is implemented on the latest generation of Web browsers.

Uniform Resource Citation (URC): A set of attributes that describe an object's characteristics, such as size, origin, authorship, copyright, data type, and so on.

Uniform Resource Identifier (URI): Generic set of identifications that encompasses URCs, URLs, and URNs.

Uniform Resource Locator (URL): The path to a specific document on the Internet that consists of the protocol (`http://`, `ftp://`, `gopher://`, `news://`, and so on), a domain name or IP address, a directory structure, and the document name.

Uniform Resource Name (URN): Any identifier other than a URL. This seemingly ambiguous definition leaves open the door to flexible ways of accessing Internet information in the future.

Universal Multiple-Octet Coded Character Set (UCS): A 32-bit character encoding system designed to accommodate phonetic and ideographic (glyph) writing systems.

UNIX, or Unix: The world's dominant operating system, dating from the early 1970s. UNIX is considered to provide the optimum combination of portability, flexibility, and power.

URC: *See* **Uniform Resource Characteristic**

URI: *See* **Uniform Resource Identifier**

URL: *See* **Uniform Resource Locator**

URN: *See* **Uniform Resource Name**

Usenet: Growing out of local electronic bulletin boards, Usenet is an informal collection of tens of thousands of thematic groups. Users post ASCII or binary messages that propagate from server to server, sometimes over a period of days. Usenet traffic ranges from authoritative to fictional, accurate to fraudulent, art to pornography.

username: A computer user's identification, nickname, or "handle" for a specific site, application, or connection. For security, the username is combined with a password or other authentication method.

UTF: *See* **UCS Transformation Format**

V

valid document: Usually referring to a Web page, a valid document passes all the requirements of the particular HTML specification under which is was created, including structure, syntax, and characters.

Validator: An application that checks a document for validity.

value: A numeric or text variable that determines the range or function of a parameter. *See also* **parameter**

VBScript: Microsoft entry into the scripting languages for Web sites. *See also* **JavaScript**, **script**

vector graphics: A technique of displaying images by drawing them directly with lines and curves of changing intensity and color. *See also* **bitmap graphics**, **raster graphics**

viewer: A helper application originally intended for graphics, then audio. Largely superseded by the term *helper application. See also* **ActiveX**, **helper application**, **Java**, **plug-in**

Viola: An early UNIX-based browser organized on the HyperCard model.

Virtual Reality Modeling Language (VRML): A method of displaying, moving within, changing, and hyperlinking three-dimensional scenes. VRML deals with concepts and techniques similar to HTML but is concerned with the unique non-textual aspects of three-dimensional images, such as lighting and shadow, viewpoints, backgrounds, and the z-axis.

virus: A program, usually destructive or mischievous, designed to hide inside digital information, usually an application or active library, and remain outside the control of the user. Viruses usually self-replicate, hence the name, to infect other programs or systems via diskettes or networks.

VRML: *See* **Virtual Reality Modeling Language**

W

W3C: *See* **World Wide Web Consortium**

WAIS: *See* **wide area information service**

watermark: A technique that permanently marks a digital document, usually with its creator's name and rights information. Digital watermarking is at its infant stage because of the public nature of Web documents. *See also* **copyright**, **intellectual property**

Web: *See* **World Wide Web**

Web page: *See* **page**

Web site: A combination of related Web pages gathered under one domain or directory, usually dealing with a single theme or personality.

WebTV: A television-based Web appliance, usually containing a subset of Web features, operated by remote control. WebTV concerns Web designers because of its smaller display size and lack of horizontal scrolling.

well-formed document: A document that not only passes HTML markup validation but also possesses a logical structure and is platform-independent.

wide area information service (WAIS): An early Internet method of indexing and retrieving information.

window: A portion of a display screen that contains an operating application or open file. Windows are often thought of as the pieces of paper on a desk (hence the term *desktop*).

Windows: A Microsoft operating system. Windows 3.*x* was a shell formed over DOS, whereas Windows 95 is a true operating system that also contains a Web browser and dialup software. Windows NT is a network system that is operating an increasing number of Web servers.

World Wide Web (WWW): The Web is a subset of the Internet that implements the concept of hyperlinks to exchange information and operates according to the HyperText Transfer Protocol. The Web is distinguished as the first widespread implementation of nonlinear information concepts.

World Wide Web Consortium (W3C): An industry consortium to develop common standards for the Web. Members propose standards and protocols for use of hypermedia.

WWW: *See* **World Wide Web**

WYSIWYG: What you see is what you get; simplified and largely inaccurate abbreviation for software that presents screen displays resembling a final product, such as a printed document or a Web page.

XYZ

XML: *See* **Extensible Markup Language**

APPENDIX A

HTML Quick Reference

IN THIS APPENDIX

This appendix is a reference to the HTML tags you can use in your documents. Unless otherwise noted, all of the tags listed here are supported by both Microsoft Explorer 3.0 and Netscape Navigator 3.0. Note that some other browsers do not support all the tags listed.

The proposed HTML style sheet specification is also not covered here. Refer to the Netscape (`http://home.netscape.com/`) or Microsoft (`http://www.microsoft.com/`) Web sites for details on this and other late-breaking changes to the HTML standard.

HTML Tags

These tags are used to create a basic HTML page with text, headings, and lists. An (MS) beside the attribute indicates that it is only supported by Microsoft Internet Explorer.

Comments

Tag	Attribute	Function
`<!-- ... -->`		Creates a comment. Can also be used to hide JavaScript from browsers that do not support it.
`<COMMENT>...</COMMENT>`		The new official way of specifying comments.

Structure Tags

Tag	Attribute	Function
`<HTML>...</HTML>`		Encloses the entire HTML document.
`<HEAD>...</HEAD>`		Encloses the head of the HTML document.
`<BODY>...</BODY>`		Encloses the body (text and tags) of the HTML document.
	`BACKGROUND="..."`	The name or URL of the image to tile on the page background.
	`BGCOLOR="..."`	The color of the page background.
	`TEXT="..."`	The color of the page's text.
	`LINK="..."`	The color of unfollowed links.
	`ALINK="..."`	The color of activated links.
	`VLINK="..."`	The color of followed links.
	`BGPROPERTIES="..."`(MS)	Properties of background image. Currently allows only the value `FIXED`, which prevents the background image from scrolling.

Tag	Attribute	Function
	`TOPMARGIN="..."`(MS)	Top margin of the page, in pixels.
	`BOTTOMMARGIN="..."`(MS)	Bottom margin of the page, in pixels.
`<BASE>`		Indicates the full URL of the current document. This optional tag is used within `<HEAD>`.
	`HREF="..."`	The full URL of this document.
`<ISINDEX>`		Indicates that this document is a gateway script that allows searches.
	`PROMPT="..."`	The prompt for the search field.
	`ACTION="..."`	Gateway program to which the search string should be passed.
`<LINK>`		Indicates a link between this document and some other document. Generally used only by HTML-generating tools. `<LINK>` represents a link from this entire document to another, as opposed to `<A>`, which can create multiple links in the document. Not commonly used.
	`HREF="..."`	The URL of the document to call when the link is activated.
	`NAME="..."`	If the document is to be considered an anchor, the name of that anchor.
	`REL="..."`	The relationship between the linked-to document and the current document; for example, `"TOC"` or `"Glossary"`.
	`REV="..."`	A reverse relationship between the current document and the linked-to document.
	`URN="..."`	A Uniform Resource Number (URN), a unique identifier different from the URL in `HREF`.

continues

Tag	Attribute	Function
	TITLE="..."	The title of the linked-to document.
	METHODS="..."	The method by which the document is to be retrieved; for example, FTP, Gopher, and so on.
<META>		Indicates meta-information about this document (information about the document itself); for example, keywords for search engines, special HTTP headers to be used for retrieving this document, expiration dates, and so on. Meta-information is usually in the form of a key/value pair. Used in the document <HEAD>.
	HTTP-EQUIV="..."	Creates a new HTTP header field with the same name as the attribute's value; for example, HTTP-EQUIV="Expires". The value of that header is specified by the CONTENT attribute.
	NAME="..."	If meta-data is usually in the form of key/value pairs, NAME indicates the key; for example, Author or ID.
	CONTENT="..."	The content of the key/value pair (or of the HTTP header indicated by HTTP-EQUIV).
<NEXTID>		Indicates the "next" document to this one (as might be defined by a tool to manage HTML documents in series). <NEXTID> is considered obsolete.

Headings and Title

Tag	Attribute	Function
<H1>...</H1>		A first-level heading.
<H2>...</H2>		A second-level heading.
<H3>...</H3>		A third-level heading.

Tag	Attribute	Function
<H4>...</H4>		A fourth-level heading.
<H5>...</H5>		A fifth-level heading.
<H6>...</H6>		A sixth-level heading.
<TITLE>...</TITLE>		Indicates the title of the document. Used within <HEAD>.

All heading tags accept the following attribute:

	ALIGN="..."	Possible values are CENTER, LEFT, and RIGHT.

Paragraphs and Regions

Tag	Attribute	Function
<P>...</P>		A plain paragraph. The closing tag (</P>) is optional.
	ALIGN="..."	Align text to CENTER, LEFT, or RIGHT.
<DIV>...</DIV>		A region of text to be formatted.
	ALIGN="..."	Align text to CENTER, LEFT, or RIGHT.

Links

Tag	Attribute	Function
<A>...</A>		With the HREF attribute, creates a link to another document or anchor; with the NAME attribute, creates an anchor that can be linked to.
	HREF="..."	The URL of the document to be called when the link is activated.
	NAME="..."	The name of the anchor.
	REL="..."	The relationship between the linked-to document and the current document; for example, "TOC" or "Glossary". REL="..." is not commonly used.

continues

Tag	Attribute	Function
	REV="..."	A reverse relationship between the current document and the linked-to document (not commonly used).
	URN="..."	A Uniform Resource Number (URN), a unique identifier different from the URL in HREF (not commonly used).
	TITLE="..."	The title of the linked-to document (not commonly used).
	METHODS="..."	The method by which the document is to be retrieved; for example, FTP, Gopher, and so on (not commonly used).
	TARGET="..."	The name of a frame that the linked document should appear in.

Lists

Tag	Attribute	Function
<OL>...</OL>		An ordered (numbered) list.
	TYPE="..."	The type of numerals to label the list. Possible values are A, a, I, i, and 1.
	START="..."	The value with which to start this list.
<UL>...</UL>		An unordered (bulleted) list.
	TYPE="..."	The bullet dingbat to use to mark list items. Possible values are DISC, CIRCLE (or ROUND), and SQUARE.
<MENU>...</MENU>		A menu list of items.
<DIR>...</DIR>		A directory listing; items are generally smaller than 20 characters.
<LI>		A list item for use with <OL>, <UL>, <MENU>, or <DIR>.

Tag	Attribute	Function
	TYPE="..."	The type of bullet or number to label this item with. Possible values are DISC, CIRCLE (or ROUND) SQUARE, A, a, I, i, and 1.
	VALUE="..."	The numeric value this list item should have (affects this item and all below it in <OL> lists).
<DL>...</DL>		A definition or glossary list.
	COMPACT	The COMPACT attribute specifies formatting that takes less whitespace to present.
<DT>		A definition term as part of definition list.
<DD>		The corresponding definition to a definition term as part of definition list.

Character Formatting

Tag	Attribute	Function
<EM>...</EM>		Emphasis (usually italic).
<STRONG>...</STRONG>		Stronger emphasis (usually bold).
<CODE>...</CODE>		Code sample (usually Courier).
<KBD>...</KBD>		Text to be typed (usually Courier).
<VAR>...</VAR>		A variable or placeholder for some other value.
<SAMP>...</SAMP>		Sample text (not commonly used).
<DFN>...</DFN>		A definition of a term.
<CITE>...</CITE>		A citation.
<B>...</B>		Boldface text.
<I>...</I>		Italic text.
<TT>...</TT>		Typewriter (monospaced) font.

continues

Tag	Attribute	Function
<PRE>...</PRE>		Preformatted text (exact line endings and spacing will be preserved—usually rendered in a monospaced font).
<BIG>...</BIG>		Text is slightly larger than normal.
<SMALL>...</SMALL>		Text is slightly smaller than normal.
_{...}		Subscript.
^{...}		Superscript.
<STRIKE>...</STRIKE>		Puts a strikethrough line in text.

Other Elements

Tag	Attribute	Function
<HR>		A horizontal rule line.
	SIZE="..."	The thickness of the rule, in pixels.
	WIDTH="..."	The width of the rule, in pixels or as a percentage of the document width.
	ALIGN="..."	How the rule line will be aligned on the page. Possible values are LEFT, RIGHT, and CENTER.
	NOSHADE	Causes the rule line to be drawn as a solid line instead of a transparent bevel.
	COLOR="..." (MS)	Color of the horizontal rule.
 		A line break.
	CLEAR="..."	Causes the text to stop flowing around any images. Possible values are RIGHT, LEFT, and ALL.
<NOBR>...</NOBR>		Causes the enclosed text not to wrap at the edge of the page.
<WBR>		Wraps the text at this point only if necessary.

Tag	Attribute	Function
<BLOCKQUOTE>... </BLOCKQUOTE>		Used for long quotes or citations.
<ADDRESS>... </ADDRESS>		Used for signatures or general information about a document's author.
<CENTER>... </CENTER>		Centers text or images.
<BLINK>...</BLINK>		Causes the enclosed text to blink in an irritating manner.
<FONT>...</FONT>		Changes the size of the font for the enclosed text.
	SIZE="..."	The size of the font, from 1 to 7. Default is 3. Can also be specified as a value relative to the current size; for example, +2.
	COLOR="..."	Changes the color of the text.
	FACE="..."	Name of font to use if it can be found on the user's system. Multiple font names can be separated by commas, and the first font on the list that can be found will be used.
<BASEFONT>		Sets the default size of the font for the current page.
	SIZE="..."	The default size of the font, from 1 to 7. Default is 3.

Images, Sounds, and Embedded Media

Tag	Attribute	Function
<IMG>		Inserts an inline image into the document.
	ISMAP	This image is a clickable image map.
	SRC="..."	The URL of the image.

continues

Tag	Attribute	Function
	ALT="..."	A text string that will be displayed in browsers that cannot support images.
	ALIGN="..."	Determines the alignment of the given image. If LEFT or RIGHT, the image is aligned to the left or right column, and all following text flows beside that image. All other values such as TOP, MIDDLE, and BOTTOM, or the Netscape-only TEXTTOP, ABSMIDDLE, BASELINE, and ABSBOTTOM determine the vertical alignment of this image with other items in the same line.
	VSPACE="..."	The space between the image and the text above or below it.
	HSPACE="..."	The space between the image and the text to its left or right.
	WIDTH="..."	The width, in pixels, of the image. If WIDTH is not the actual width, the image is scaled to fit.
	HEIGHT="..."	The height, in pixels, of the image. If HEIGHT is not the actual height, the image is scaled to fit.
	BORDER="..."	Draws a border of the specified value in pixels to be drawn around the image. In the case of images that are also links, BORDER changes the size of the default link border.
	LOWSRC="..."	The path or URL of an image that will be loaded first, before the image specified in SRC. The value of LOWSRC is usually a smaller or lower resolution version of the actual image.
	USEMAP="..."	The name of an image map specification for client-side image mapping. Used with <MAP> and <AREA>.

Tag	Attribute	Function
	DYNSRC="..." (MS)	The address of a video clip or VRML world (dynamic source).
	CONTROLS (MS)	Used with DYNSRC to display a set of playback controls for inline video.
	LOOP="..." (MS)	The number of times a video clip will loop. (-1, or INFINITE, means to loop indefinitely.)
	START="..." (MS)	When a DYNSRC video clip should start playing. Valid options are FILEOPEN (play when page is displayed) or MOUSEOVER (play when mouse cursor passes over the video clip).
<BGSOUND> (MS)		Plays a sound file as soon as the page is displayed.
	SRC="..."	The URL of the WAV, AU, or MIDI sound file to embed.
	LOOP="..." (MS)	The number of times a video clip will loop. (-1, or INFINITE, means to loop indefinitely.)
<SCRIPT>		An interpreted script program.
	LANGUAGE="..."	Currently only JAVASCRIPT is supported by Netscape. Both JAVASCRIPT and VBSCRIPT are supported by Microsoft.
	SRC="..."	Specifies the URL of a file that includes the script program.
<OBJECT>		Inserts an image, video, Java applet, or ActiveX control into a document.

NOTE

Usage of the <OBJECT> tag is not yet finalized. Check http://www.w3.org/ for the latest attributes supported by the HTML 3.2 standard.

Tag	Attribute	Function
<APPLET>		Inserts a self-running Java applet.
	CLASS="..."	The name of the applet.
	SRC="..."	The URL of the directory where the compiled applet can be found (should end in a slash as in http://mysite/myapplets/). Do not include the actual applet name, which is specified with the CLASS attribute.
	ALIGN="..."	Indicates how the applet should be aligned with any text that follows it. Current values are TOP, MIDDLE, and BOTTOM.
	WIDTH="..."	The width of the applet output area, in pixels.
	HEIGHT="..."	The height of the applet output area, in pixels.
<PARAM>		Program-specific parameters. (Always occurs within <APPLET> or <OBJECT> tags.)
	NAME="..."	The type of information being given to the applet or ActiveX control.
	VALUE="..."	The actual information to be given to the applet or ActiveX control.
	REF="..."	Indicates that this <PARAM> tag includes the address or location of the object.
<EMBED> (Netscape only!)		Embeds a file to be read or displayed by a plug-in application.

NOTE

In addition to the following standard attributes, you can specify applet-specific attributes to be interpreted by the plug-in that displays the embedded object.

Tag	Attribute	Function
	SRC="..."	The URL of the file to embed.
	WIDTH="..."	The width of the embedded object in pixels.
	HEIGHT="..."	The height of the embedded object in pixels.
	ALIGN="..."	Determines the alignment of the media window. Values are the same as for the <IMG> tag.
	VSPACE="..."	The space between the media and the text above or below it.
	HSPACE="..."	The space between the media and the text to its left or right.
	BORDER="..."	Draws a border of the specified size in pixels to be drawn around the media.
<NOEMBED>...</NOEMBED>		Alternate text or images to be shown to users who do not have a plug-in installed.
<MAP>...</MAP>		A client-side image map, referenced by <IMG USEMAP="...">. Includes one or more <AREA> tags.
<AREA>		Defines a clickable link within a client-side image map.
	SHAPE="..."	The shape of the clickable area. Currently, only RECT is supported.
	COORDS="..."	The left, top, right, and bottom coordinates of the clickable region within an image.
	HREF="..."	The URL that should be loaded when the area is clicked.
	NOHREF	Indicates that no action should be taken when this area of the image is clicked.

Forms

Tag	Attribute	Function
<FORM>...</FORM>		Indicates an input form.
	ACTION="..."	The URL of the script to process this form input.
	METHOD="..."	How the form input will be sent to the gateway on the server side. Possible values are GET and POST.
	ENCTYPE="..."	Normally has the value application/x-www-form-urlencoded. For file uploads, use multipart/form-data.
	NAME="..."	A name by which JavaScript scripts can refer to the form.
<INPUT>		An input element for a form.
	TYPE="..."	The type for this input widget. Possible values are CHECKBOX, HIDDEN, RADIO, RESET, SUBMIT, TEXT, SEND FILE, or IMAGE.
	NAME="..."	The name of this item, as passed to the gateway script as part of a name/value pair.
	VALUE="..."	For a text or hidden widget, the default value; for a checkbox or radio button, the value to be submitted with the form; for Reset or Submit buttons, the label for the button itself.
	SRC="..."	The source file for an image.
	CHECKED	For checkboxes and radio buttons, indicates that the widget is checked.
	SIZE="..."	The size, in characters, of a text widget.
	MAXLENGTH="..."	The maximum number of characters that can be entered into a text widget.

Tag	Attribute	Function
	ALIGN="..."	For images in forms, determines how the text and image will align (same as with the <IMG> tag).
<TEXTAREA>...</TEXTAREA>		Indicates a multiline text entry form element. Default text can be included.
	NAME="..."	The name to be passed to the gateway script as part of the name/value pair.
	ROWS="..."	The number of rows this text area displays.
	COLS="..."	The number of columns (characters) this text area displays.
	WRAP="..."	Controls text wrapping. Possible values are OFF, VIRTUAL, and PHYSICAL.
<SELECT>...</SELECT>		Creates a menu or scrolling list of possible items.
	NAME="..."	The name that is passed to the gateway script as part of the name/value pair.
	SIZE="..."	The number of elements to display. If SIZE is indicated, the selection becomes a scrolling list. If no SIZE is given, the selection is a pop-up menu.
	MULTIPLE	Allows multiple selections from the list.
<OPTION>		Indicates a possible item within a <SELECT> element.
	SELECTED	With this attribute included, the <OPTION> will be selected by default in the list.
	VALUE="..."	The value to submit if this <OPTION> is selected when the form is submitted.

Tables

Tag	Attribute	Function
`<TABLE>...</TABLE>`		Creates a table that can contain a caption (`<CAPTION>`) and any number of rows (`<TR>`).
	`BORDER="..."`	Indicates whether the table should be drawn with or without a border. In Netscape, `BORDER` can also have a value indicating the width of the border.
	`CELLSPACING="..."`	The amount of space between the cells in the table.
	`CELLPADDING="..."`	The amount of space between the edges of the cell and its contents.
	`WIDTH="..."`	The width of the table on the page, in either exact pixel values or as a percentage of page width.
	`ALIGN="..."` (MS)	Alignment (works like `IMG ALIGN`). Values are `LEFT` or `RIGHT`.
	`BGCOLOR="..."`	Background color of all cells in the table that do not contain their own `BACKGROUND` or `BGCOLOR` attribute.
	`BACKGROUND="..."` (MS)	Background image to tile within all cells in the table that do not contain their own `BACKGROUND` or `BGCOLOR` attribute.
	`BORDERCOLOR="..."` (MS)	Border color (used with `BORDER="..."`).

Tag	Attribute	Function
	`BORDERCOLORLIGHT="..."` (MS)	Color for light part of 3D-look borders (used with `BORDER="..."`).
	`BORDERCOLORDARK="..."` (MS)	Color for dark part of 3D-look borders (used with `BORDER="..."`).
	`VALIGN="..."` (MS)	Alignment of text within the table. Values are `TOP` and `BOTTOM`.
	`FRAME="..."` (MS)	Controls which external borders will appear around a table. Values are `void` (no frames), `above` (top border only), `below` (bottom border only), `hsides` (top and bottom), `lhs` (left side), `rhs` (right side), `vsides` (left and right sides), and `box` (all sides).
	`RULES="..."` (MS)	Controls which internal borders appear in the table. Values are `none`, `basic` (rules between `THEAD`, `TBODY`, and `TFOOT` only), `rows` (horizontal borders only), `cols` (vertical borders only), and `all`.
`<CAPTION>...</CAPTION>`		The caption for the table.
	`ALIGN="..."`	The position of the caption. Possible values are `TOP` and `BOTTOM`.

→continues

Tag	Attribute	Function
<TR>...</TR>		Defines a table row, containing headings and data (<TR> and <TH> tags).
	ALIGN="..."	The horizontal alignment of the contents of the cells within this row. Possible values are LEFT, RIGHT, and CENTER.
	VALIGN="..."	The vertical alignment of the contents of the cells within this row. Possible values are TOP, MIDDLE, BOTTOM, and BASELINE.
	BGCOLOR="..."	Background color of all cells in the row that do not contain their own BACKGROUND or BGCOLOR attributes.
	BACKGROUND="..."(MS)	Background image to tile within all cells in the row that do not contain their own BACKGROUND or BGCOLOR attributes.
	BORDERCOLOR="..."(MS)	Border color (used with BORDER="...").
	BORDERCOLORLIGHT="..."(MS)	Color for light part of 3D-look borders (used with BORDER="...").
	BORDERCOLORDARK="..."(MS)	Color for dark part of 3D-look borders (used with BORDER="...").
<TH>...</TH>		Defines a table heading cell.

Tag	Attribute	Function
	ALIGN="..."	The horizontal alignment of the contents of the cell. Possible values are LEFT, RIGHT, and CENTER.
	VALIGN="..."	The vertical alignment of the contents of the cell. Possible values are TOP, MIDDLE, BOTTOM, and BASELINE.
	ROWSPAN="..."	The number of rows this cell will span.
	COLSPAN="..."	The number of columns this cell will span.
	NOWRAP	Does not automatically wrap the contents of this cell.
	WIDTH="..."	The width of this column of cells, in exact pixel values or as a percentage of the table width.
	BGCOLOR="..."	Background color of the cell.
	BACKGROUND="..." (MS)	Background image to tile within the cell.
	BORDERCOLOR="..." (MS)	Border color (used with BORDER="...").
	BORDERCOLORLIGHT="..." (MS)	Color for light part of 3D-look borders (used with BORDER="...").
	BORDERCOLORDARK="..." (MS)	Color for dark part of 3D-look borders (used with BORDER="...").
<TD>...</TD>		Defines a table data cell.

continues

Tag	Attribute	Function
	ALIGN="..."	The horizontal alignment of the contents of the cell. Possible values are LEFT, RIGHT, and CENTER.
	VALIGN="..."	The vertical alignment of the contents of the cell. Possible values are TOP, MIDDLE, BOTTOM, and BASELINE.
	ROWSPAN="..."	The number of rows this cell will span.
	COLSPAN="..."	The number of columns this cell will span.
	NOWRAP	Does not automatically wrap the contents of this cell.
	WIDTH="..."	The width of this column of cells, in exact pixel values or as a percentage of the table width.
	BGCOLOR="..."	Background color of the cell.
	BACKGROUND="..." (MS)	Background image to tile within the cell.
	BORDERCOLOR="..." (MS)	Border color (used with BORDER="...").
	BORDERCOLORLIGHT="..." (MS)	Color for light part of 3D-look borders (used with BORDER="...").
	BORDERCOLORDARK="..." (MS)	Color for dark part of 3D-look borders (used with BORDER="...").

Frames

Tag	Attribute	Function
<FRAMESET>... </FRAMESET>		Divides the main window into a set of frames that can each display a separate document.
	ROWS="..."	Splits the window or frameset vertically into a number of rows specified by a number (such as 7), a percentage of the total window width (such as 25%), or an asterisk (*) indicating that a frame should take up all the remaining space or divide the space evenly between frames (if multiple * frames are specified).
	COLS="..."	Works similar to ROWS, except that the window or frameset is split horizontally into columns.
	BORDER="..."	Size of frame border in pixels (0 turns off borders). This tag is Netscape-specific; Microsoft IE uses FRAMEBORDER and FRAMESPACING instead.
	FRAMEBORDER="..." (MS)	Specifies whether to display a border for a frame. Options are YES and NO.
	FRAMESPACING="..." (MS)	Space between frames, in pixels.
<FRAME>		Defines a single frame within a <FRAMESET>.
	SRC="..."	The URL of the document to be displayed in this frame.

continues

Tag	Attribute	Function
	NAME="..."	A name to be used for targeting this frame with the TARGET attribute in <A HREF> links.
	MARGINWIDTH="..."	The amount of space to leave to the left and right side of a document within a frame, in pixels.
	MARGINHEIGHT="..."	The amount of space to leave above and below a document within a frame, in pixels.
	SCROLLING="..."	Determines whether a frame has scrollbars. Possible values are YES, NO, and AUTO.
	NORESIZE	Prevents the user from resizing this frame (and possibly adjacent frames) with the mouse.
<NOFRAME>...</NOFRAME>		Provides an alternative document body in <FRAMESET> documents for browsers that do not support frames (usually encloses <BODY>...</BODY>).

Character Entities

Table A.1 contains the possible numeric and character entities for the ISO-Latin-1 (ISO8859-1) character set. Where possible, the character is shown.

NOTE

Not all browsers can display all characters, and some browsers might even display characters different from those that appear in the table. Newer browsers seem to have a better track record for handling character entities, but be sure to test your HTML files extensively with multiple browsers if you intend to use these entities.

Table A.1. ISO-Latin-1 character set.

Character	*Numeric Entity*	*Character Entity (if any)*	*Description*
	&#00;–&#08;		Unused
			Horizontal tab
	Line feed
	&#11;–&#31;		Unused
			Space
!	!		Exclamation mark
"	"	"	Quotation mark
#	#		Number sign
$	$		Dollar sign
%	%		Percent sign
&	&	&	Ampersand
'	'		Apostrophe
(	(		Left parenthesis
)	)		Right parenthesis
*	*		Asterisk
+	+		Plus sign
,	,		Comma
-	-		Hyphen
.	.		Period (fullstop)
/	/		Solidus (slash)
0–9	0–9		Digits 0–9
:	:		Colon
;	;		Semicolon
<	<	<	Less than
=	=		Equal sign
>	>	>	Greater than
?	?		Question mark
@	@		Commercial "at"
A–Z	A–Z		Letters A–Z

continues

Table A.1. continued

Character	*Numeric Entity*	*Character Entity (if any)*	*Description*
[	[		Left square bracket
\	\		Reverse solidus (backslash)
]	]		Right square bracket
^	^		Caret
—	_		Horizontal bar
`	`		Grave accent
a–z	a–z		Letters a–z
{	{		Left curly brace
|	&#124		Vertical bar
}	}		Right curly brace
~	~		Tilde
	&#127;–		Unused
¡	¡	¡	Inverted exclamation
¢	¢	¢	Cent sign
£	£	£	Pound sterling
¤	¤	¤	General currency sign
¥	¥	¥	Yen sign
¦	¦	¦ or &brkbar;	Broken vertical bar
§	§	§	Section sign
¨	¨	¨	Umlaut (dieresis)
©	©	© (Netscape only)	Copyright
ª	ª	ª	Feminine ordinal
«	«	«	Left angle quote, guillemot left
¬	¬	¬	Not sign

Character	*Numeric Entity*	*Character Entity (if any)*	*Description*
			Soft hyphen
®	®	® (Netscape only)	Registered trademark
¯	¯	¯	Macron accent
°	°	°	Degree sign
±	±	±	Plus or minus
²	²	²	Superscript two
³	³	³	Superscript three
´	´	´	Acute accent
µ	µ	µ	Micro sign
¶	¶	¶	Paragraph sign
·	·	·	Middle dot
¸	¸	¸	Cedilla
¹	¹	¹	Superscript one
º	º	º	Masculine ordinal
»	»	»	Right angle quote, guillemot right
¼	¼	¼	Fraction one-fourth
½	½	½	Fraction one-half
¾	¾	¾	Fraction three-fourths
¿	¿	&iquest	Inverted question mark
À	À	À	Capital A, grave accent
Á	Á	Á	Capital A, acute accent
Â	Â	Â	Capital A, circumflex accent

continues

Table A.1. continued

Character	*Numeric Entity*	*Character Entity (if any)*	*Description*
Ã	Ã	Ã	Capital A, tilde
Ä	Ä	Ä	Capital A, dieresis or umlaut mark
Å	Å	Å	Capital A, ring
Æ	Æ	Æ	Capital AE diphthong (ligature)
Ç	Ç	Ç	Capital C, cedilla
È	È	È	Capital E, grave accent
É	É	É	Capital E, acute accent
Ê	Ê	Ê	Capital E, circumflex accent
Ë	Ë	Ë	Capital E, dieresis or umlaut mark
Ì	Ì	Ì	Capital I, grave accent
Í	Í	Í	Capital I, acute accent
Î	Î	Î	Capital I, circumflex accent
Ï	Ï	Ï	Capital I, dieresis or umlaut mark
Ð	Ð	Ð	Capital Eth, Icelandic
Ñ	Ñ	Ñ	Capital N, tilde
Ò	Ò	Ò	Capital O, grave accent

Character	*Numeric Entity*	*Character Entity (if any)*	*Description*
Ó	Ó	Ó	Capital O, acute accent
Ô	Ô	Ô	Capital O, circumflex accent
Õ	Õ	Õ	Capital O, tilde
Ö	Ö	Ö	Capital O, dieresis or umlaut mark
×	×	×	Multiply sign
Ø	Ø	Ø	Capital O, slash
Ù	Ù	Ù	Capital U, grave accent
Ú	Ú	Ú	Capital U, acute accent
Û	Û	Û	Capital U, circumflex accent
Ü	Ü	Ü	Capital U, dieresis or umlaut mark
Ý	Ý	Ý	Capital Y, acute accent
Þ	Þ	Þ	Capital THORN, Icelandic
ß	ß	ß	Small sharp s, German (sz ligature)
à	à	à	Small a, grave accent
á	á	á	Small a, acute accent
â	â	â	Small a, circumflex accent

continues

Table A.1. continued

Character	*Numeric Entity*	*Character Entity (if any)*	*Description*
ã	ã	ã	Small a, tilde
ä	ä	ä	Small a, dieresis or umlaut mark
å	å	å	Small a, ring
æ	æ	æ	Small ae diphthong (ligature)
ç	ç	ç	Small c, cedilla
è	è	è	Small e, grave accent
é	é	é	Small e, acute accent
ê	ê	ê	Small e, circumflex accent
ë	ë	ë	Small e, dieresis or umlaut mark
ì	ì	ì	Small i, grave accent
í	í	í	Small i, acute accent
î	î	î	Small i, circumflex accent
ï	ï	ï	Small i, dieresis or umlaut mark
ð	ð	ð	Small eth, Icelandic
ñ	ñ	ñ	Small n, tilde
ò	ò	ò	Small o, grave accent
ó	ó	ó	Small o, acute accent
ô	ô	ô	Small o, circumflex accent

Character	*Numeric Entity*	*Character Entity (if any)*	*Description*
õ	õ	õ	Small o, tilde
ö	ö	ö	Small o, dieresis or umlaut mark
÷	÷	÷	Division sign
ø	ø	ø	Small o, slash
ù	ù	ù	Small u, grave accent
ú	ú	ú	Small u, acute accent
û	û	û	Small u, circumflex accent
ü	ü	ü	Small u, dieresis or umlaut mark
ý	ý	ý	Small y, acute accent
þ	þ	þ	Small thorn, Icelandic
ÿ	ÿ	ÿ	Small y, dieresis or umlaut mark

APPENDIX B

HTML 3.2 Reference Specification

by Dave Raggett (dsr@w3.org)

IN THIS APPENDIX

Status of This Document

This document has been reviewed by W3C members and other interested parties and has been endorsed by the Director as a W3C Recommendation. It is a stable document and may be used as reference material or cited as a normative reference from another document. W3C's role in making the Recommendation is to draw attention to the specification and to promote its widespread deployment. This enhances the functionality and interoperability of the Web.

A list of current W3C Recommendations and other technical documents can be found at `http://www.w3.org/pub/WWW/TR/">http://www.w3.org/pub/WWW/TR/`.

Abstract

The Hypertext Markup Language (HTML) is a simple markup language used to create hypertext documents that are portable from one platform to another. HTML documents are SGML documents with generic semantics that are appropriate for representing information from a wide range of applications. This specification defines HTML Version 3.2. HTML 3.2 aims to capture recommended practice as of early '96 and as such to be used as a replacement for HTML 2.0 (RFC 1866).

Contents

Introduction to HTML 3.2

HTML 3.2 is W3C's specification for HTML, developed in early '96 together with vendors including IBM, Microsoft, Netscape Communications Corporation, Novell, SoftQuad, Spyglass, and Sun Microsystems. HTML 3.2 adds widely deployed features such as tables, applets and text flow around images, while providing full backwards compatibility with the existing standard HTML 2.0.

W3C is continuing to work with vendors on extensions for accessibility features, multimedia objects, scripting, style sheets, layout, forms, math, and internationalization. W3C plans on incorporating this work in further versions of HTML.

HTML as an SGML Application

HTML 3.2 is an SGML application conforming to International Standard ISO 8879—Standard Generalized Markup Language. As an SGML application, the syntax of conforming HTML 3.2 documents is defined by the combination of the SGML declaration and the document type definition (DTD). This specification defines the intended interpretation of HTML 3.2 elements and places further constraints on the permitted syntax that are otherwise inexpressible in the DTD.

The SGML rules for record boundaries are tricky. In particular, a record end immediately following a start tag should be discarded. For example:

```
<P>
Text
```

is equivalent to

```
<P>Text
```

Similarly, a record end immediately preceding an end tag should be discarded. For example:

```
Text
</P>
```

is equivalent to

```
Text</P>
```

Except within literal text (for example, the PRE element), HTML treats contiguous sequences of white space characters as being equivalent to a single space character (ASCII decimal 32). These rules allow authors considerable flexibility when editing the marked-up text directly. Note that future revisions to HTML may allow for the interpretation of the horizontal tab character (ASCII decimal 9) with respect to a tab rule defined by an associated style sheet.

SGML entities in PCDATA content or in CDATA attributes are expanded by the parser; for example, `é` is expanded to the ISO Latin-1 character decimal 233 (a lowercase letter *e* with an acute accent). This could also have been written as a named character entity, for example, `é`. The `&` character can be included in its own right using the named character entity `&`.

HTML allows CDATA attributes to be unquoted provided the attribute value contains only letters (*a* to *z* and *A* to *Z*), digits (0 to 9), hyphens (ASCII decimal 45), or periods (ASCII decimal 46). Attribute values can be quoted using double or single quote marks (ASCII decimal 34 and 39, respectively). Single quote marks can be included within the attribute value when the value is delimited by double quote marks, and vice versa.

Note that some user agents require attribute minimization for the following attributes: COMPACT, ISMAP, CHECKED, NOWRAP, NOSHADE, and NOHREF. These user agents don't accept syntax such as COMPACT=COMPACT or ISMAP=ISMAP although this is legitimate according to the HTML 3.2 DTD.

The SGML declaration and the DTD for use with HTML 3.2 are given in appendices. Further guidelines for parsing HTML are given in WD-html-lex.

The Structure of HTML Documents

HTML 3.2 documents start with a <!DOCTYPE> declaration followed by an HTML element containing a HEAD and then a BODY element:

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 3.2 Final//EN>
<HTML>
<HEAD>
<TITLE>A study of population dynamics</TITLE>
... other head elements
</HEAD>
<BODY>
... document body
</BODY>
</HTML>
```

In practice, the HTML HEAD and BODY start and end tags can be omitted from the markup as these can be inferred in all cases by parsers conforming to the HTML 3.2 DTD.

Every conforming HTML 3.2 document **must** start with the <!DOCTYPE> declaration that is needed to distinguish HTML 3.2 documents from other versions of HTML. The HTML specification is not concerned with storage entities. As a result, it is not required that the document type declaration reside in the same storage entity (that is, file). A Web site may choose to dynamically prepend HTML files with the document type declaration if it is known that all such HTML files conform to the HTML 3.2 specification.

Every HTML 3.2 document must also include the descriptive title element. A minimal HTML 3.2 document thus looks like this:

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 3.2 Final//EN>
<TITLE>A study of population dynamics</TITLE>
```

NOTE

The word "Final" replaces "Draft" now that the HTML 3.2 specification has been ratified by the W3C member organizations.

The HEAD Element

This contains the document head, but you can always omit both the start and end tags for HEAD. The contents of the document head are an unordered collection of the following elements:

- The TITLE element
- The STYLE element
- The SCRIPT element
- The ISINDEX element
- The BASE element
- The META element
- The LINK element

```
<!ENTITY % head.content "TITLE & ISINDEX? & BASE?">
<!ENTITY % head.misc "SCRIPT|STYLE|META|LINK">

<!ELEMENT HEAD O O  (%head.content) +(%head.misc)>
```

The %head.misc entity is used to allow the associated elements to occur multiple times at arbitrary positions within the HEAD. The following elements can be part of the document head:

- TITLE defines the document title, and is always needed.
- ISINDEX is for simple keyword searches; see PROMPT attribute.
- BASE defines base URL for resolving relative URLs.
- SCRIPT is reserved for future use with scripting languages.
- STYLE is reserved for future use with style sheets.
- META is used to supply meta info as name/value pairs.
- LINK is used to define relationships with other documents.

TITLE, SCRIPT and STYLE are containers and require both start and end tags. The other elements are not containers so that end tags are forbidden. Note that conforming browsers won't render the contents of SCRIPT and STYLE elements.

TITLE

```
<!ELEMENT TITLE - -  (#PCDATA)* -(%head.misc)>
```

Every HTML 3.2 document **must** have exactly one TITLE element in the document's HEAD. It provides an advisory title that can be displayed in a user agent's window caption and so on. The content model is PCDATA. As a result, character entities can be used for accented characters and to escape special characters such as & and <. Markup is not permitted in the content of a TITLE element.

Example TITLE element:

```
<TITLE>A study of population dynamics</TITLE>
```

STYLE and SCRIPT

```
<!ELEMENT STYLE  - - CDATA -- placeholder for style info -->
<!ELEMENT SCRIPT - - CDATA -- placeholder for script statements -->
```

These are placeholders for the introduction of style sheets and client-side scripts in future versions of HTML. User agents should hide the contents of these elements.

These elements are defined with CDATA as the content type. As a result they may contain only SGML characters. All markup characters or delimiters are ignored and passed as data to the application, except for ETAGO (</) delimiters followed immediately by a name character [a-zA-Z]. This means that the element's end tag (or that of an element in which it is nested) is recognized, while an error occurs if the ETAGO is invalid.

ISINDEX

```
<!ELEMENT ISINDEX - O EMPTY>
<!ATTLIST ISINDEX
        prompt CDATA #IMPLIED -- prompt message -->
```

The ISINDEX element indicates that the user agent should provide a single-line text input field for entering a query string. There are no restrictions on the number of characters that can be entered. The PROMPT attribute can be used to specify a prompt string for the input field, for example:

```
<ISINDEX PROMPT="Search Phrase">
```

The semantics for ISINDEX are currently well defined only when the base URL for the enclosing document is an HTTP URL. Typically, when the user presses the Enter (return) key, the query string is sent to the server identified by the base URL for this document. For example, if the query string entered is "ten green apples" and the base URL is

```
http://www.acme.com/
```

then the query generated is

```
http://www.acme.com/?ten+green+apples"
```

Note that space characters are mapped to + characters and that normal URL character escaping mechanisms apply. For further details, see the HTTP specification.

NOTE

Note that in practice, the query string is restricted to Latin-1 as there is no current mechanism for the URL to specify a character set for the query.

BASE

```
<!ELEMENT BASE - O EMPTY>
<!ATTLIST BASE
        href %URL  #REQUIRED
        >
```

The BASE element gives the base URL for dereferencing relative URLs, using the rules given by the URL specification, for example,

```
<BASE href="http://www.acme.com/intro.html">
    ...
    <IMG SRC="icons/logo.gif">
```

The image is dereferenced to

```
http://www.acme.com/icons/logo.gif
```

In the absence of a BASE element, the document URL should be used. Note that this is not necessarily the same as the URL used to request the document, as the base URL may be overridden by an HTTP header accompanying the document.

META

```
<!ELEMENT META - O EMPTY    -- Generic Metainformation -->
<!ATTLIST META
        http-equiv  NAME    #IMPLIED  -- HTTP response header name  --
        name        NAME    #IMPLIED  -- metainformation name       --
        content     CDATA   #REQUIRED -- associated information     --
        >
```

The META element can be used to include name/value pairs describing properties of the document, such as author, expire date, a list of key words, and so on. The NAME attribute specifies the property name, while the CONTENT attribute specifies the property value, for example,

```
<META NAME="Author" CONTENT="Dave Raggett">
```

The HTTP-EQUIV attribute can be used in place of the NAME attribute and has a special significance when documents are retrieved via the Hypertext Transfer Protocol (HTTP). HTTP servers may use the property name specified by the HTTP-EQUIV attribute to create an RFC 822 style header in the HTTP response. This can't be used to set certain HTTP headers, though; see the HTTP specification for details.

```
<META HTTP-EQUIV="Expires" CONTENT="Tue, 20 Aug 1996 14:25:27 GMT">
```

will result in the HTTP header:

```
Expires: Tue, 20 Aug 1996 14:25:27 GMT
```

This can be used by caches to determine when to fetch a fresh copy of the associated document.

LINK

LINK provides a media-independent method for defining relationships with other documents and resources. LINK has been part of HTML since the very early days, although few browsers as yet take advantage of it (most still ignore LINK elements).

LINK elements can be used *in principle* in the following ways:

- For document-specific navigation toolbars or menus
- To control how collections of HTML files are rendered into printed documents
- For linking associated resources such as style sheets and scripts
- To provide alternative forms of the current document

```
<!ELEMENT LINK - O EMPTY>
<!ATTLIST LINK
        href    %URL    #IMPLIED    -- URL for linked resource --
        rel     CDATA   #IMPLIED    -- forward link types --
        rev     CDATA   #IMPLIED    -- reverse link types --
        title   CDATA   #IMPLIED    -- advisory title string --
        >
```

`href` Specifies a URL designating the linked resource.

`rel` The forward relationship also known as the "link type." It specifies a named relationship from the enclosing document to the resource specified by the HREF attribute. HTML link relationships are as yet unstandardized, although some conventions have been established.

`rev` This defines a reverse relationship. A link from document A to document B with REV=*relation* expresses the same relationship as a link from B to A with REL=*relation*. REV=made is sometimes used to identify the document author, either the author's e-mail address with a mailto URL, or a link to the author's home page.

`title` An advisory title for the linked resource.

Here are some proposed relationship values:

`rel=top`

The link references the top of a hierarchy, for example, the first or cover page in a collection.

`rel=contents`

The link references a document serving as a table of contents.

`rel=index`

The link references a document providing an index for the current document.

`rel=glossary`

The link references a document providing a glossary of terms that are relevant to the current document.

`rel=copyright`

The link references a copyright statement for the current document.

`rel=next`

The link references the next document to visit in a guided tour. It can be used, for example, to preload the next page.

`rel=previous`

The link references the previous document in a guided tour.

`rel=help`

The link references a document offering help, for example, describing the wider context and offering further links to relevant documents. This is aimed at reorienting users who have lost their way.

`rel=search`

The link references a page for searching material related to a collection of pages

Example LINK elements:

```
<LINK REL=Contents HREF=toc.html>
<LINK REL=Previous HREF=doc31.html>
<LINK REL=Next HREF=doc33.html>
<LINK REL=Chapter REV=Contents HREF=chapter2.html>
```

The BODY Element

This contains the document body. Both start and end tags for BODY may be omitted. The body can contain a wide range of elements:

- Headings (H1–H6)
- The ADDRESS element
- Block-level elements
- Text-level elements

The key attributes are BACKGROUND, BGCOLOR, TEXT, LINK, VLINK, and ALINK. These can be used to set a repeating background image, plus background and foreground colors for normal text and hypertext links.

```
<!ENTITY % body.content "(%heading | %text | %block | ADDRESS)*">
<!ENTITY % color "CDATA" -- a color specification: #HHHHHH @@ details? -->
```

```
<!ENTITY % body-color-attrs "
        bgcolor %color #IMPLIED
        text %color #IMPLIED
        link %color #IMPLIED
        vlink %color #IMPLIED
        alink %color #IMPLIED
        ">
<!ELEMENT BODY O O  %body.content>
<!ATTLIST BODY
        background %URL #IMPLIED  -- texture tile for document background --
        %body-color-attrs;  -- bgcolor, text, link, vlink, alink --
        >
```

Example:

```
<body bgcolor=white text=black link=red vlink=maroon alink=fuschia>
```

`bgcolor`	Specifies the background color for the document body. See following for the syntax of color values.
`text`	Specifies the color used to stroke the document's text. This is generally used when you have changed the background color with the `BGCOLOR` or `BACKGROUND` attributes.
`link`	Specifies the color used to stroke the text for unvisited hypertext links.
`vlink`	Specifies the color used to stroke the text for visited hypertext links.
`alink`	Specifies the highlight color used to stroke the text for hypertext links at the moment the user clicks on the link.
`background`	Specifies a URL for an image that will be used to tile the document background.

Colors are given in the sRGB color space as hexadecimal numbers (for example, `COLOR="#C0FFC0"`), or as one of 16 widely understood color names. These colors were originally picked as being the standard 16 colors supported with the Windows VGA palette.

Color Names and sRGB Values

`Black = "#000000"`	`Green = "#008000"`
`Silver = "#C0C0C0"`	`Lime = "#00FF00"`
`Gray = "#808080"`	`Olive = "#808000"`
`White = "#FFFFFF"`	`Yellow = "#FFFF00"`
`Maroon = "#800000"`	`Navy = "#000080"`
`Red = "#FF0000"`	`Blue = "#0000FF"`
`Purple = "#800080"`	`Teal = "#008080"`
`Fuchsia = "#FF00FF"`	`Aqua = "#00FFFF"`

Block and Text-Level Elements

Most elements that can appear in the document body fall into one of two groups: block-level elements which cause paragraph breaks, and text-level elements which don't. Common block-level elements include `H1` to `H6` (headers), `P` (paragraphs) `LI` (list items), and `HR` (horizontal rules). Common text-level elements include `EM`, `I`, `B` and `FONT` (character emphasis), `A` (hypertext links), `IMG` and `APPLET` (embedded objects) and `BR` (line breaks). Note that block elements generally act as containers for text-level and other block-level elements (excluding headings and address elements), while text-level elements can only contain other text-level elements. The exact model depends on the element.

Headings

```
<!--
  There are six levels of headers from H1 (the most important)
  to H6 (the least important).
-->
<!ELEMENT ( %heading )  - -  (%text;)*>
<!ATTLIST ( %heading )
        align  (left¦center¦right) #IMPLIED
        >
```

`H1`, `H2`, `H3`, `H4`, `H5` and `H6` are used for document headings. You always need the start and end tags. `H1` elements are more important than `H2` elements and so on, so that `H6` elements define the least important level of headings. More important headings are generally rendered in a larger font than less important ones. Use the optional `ALIGN` attribute to set the text alignment within a heading, for example,

```
<H1 ALIGN=CENTER>... centered heading ...</H1>
```

The default is left alignment, but this can be overridden by an enclosing `DIV` or `CENTER` element.

ADDRESS

```
<!ENTITY % address.content "((%text;) ¦ P)*">
<!ELEMENT ADDRESS - - %address.content>
```

The `ADDRESS` element requires start and end tags and specifies information such as authorship and contact details for the current document. User agents should render the content with paragraph-breaks before and after. Note that the content is restricted to paragraphs, plain text, and text-like elements as defined by the `%text` entity.

Example:

```
<ADDRESS>
Newsletter editor<BR>
J.R. Brown<BR>
8723 Buena Vista, Smallville, CT 01234<BR>
Tel: +1 (123) 456 7890
</ADDRESS>
```

Block Elements

- **P** *paragraphs:* The paragraph element requires a start tag, but the end tag can always be omitted. Use the ALIGN attribute to set the text alignment within a paragraph, for example, <P ALIGN=RIGHT>
- **UL** *unordered lists:* These require start and end tags, and contain one or more LI elements representing individual list items.
- **OL** *ordered (numbered) lists:* These require start and end tags, and contain one or more LI elements representing individual list items.
- **DL** *definition lists:* These require start and end tags and contain DT elements that give the terms, and DD elements that give corresponding definitions.
- **PRE** *preformatted text:* Requires start and end tags. These elements are rendered with a monospaced font and preserve layout defined by whitespace and line break characters.
- **DIV** *document divisions:* Requires start and end tags. It is used with the ALIGN attribute to set the text alignment of the block elements it contains. ALIGN can be one of LEFT, CENTER, or RIGHT.
- **CENTER** *text alignment:* Requires start and end tags. It is used to center text lines enclosed by the CENTER element. See DIV for a more general solution.
- **BLOCKQUOTE** *quoted passage:* Requires start and end tags. It is used to enclose extended quotations and is typically rendered with indented margins.
- **FORM** *fill-out forms:* Requires start and end tags. This element is used to define a fill-out form for processing by HTTP servers. The attributes are ACTION, METHOD, and ENCTYPE. Form elements can't be nested.
- **ISINDEX** *primitive HTML forms:* Not a container, so the end tag is forbidden. This predates FORM and is used for simple kinds of forms which have a single text input field, implied by this element. A single ISINDEX can appear in the document head or body.
- **HR** *horizontal rules:* Not a container, so the end tag is forbidden. Attributes are ALIGN, NOSHADE, SIZE, and WIDTH.
- **TABLE** *can be nested:* Requires start and end tags. Each table starts with an optional CAPTION followed by one or more TR elements defining table rows. Each row has one or more cells defined by TH or TD elements. Attributes for TABLE elements are WIDTH, BORDER, CELLSPACING, and CELLPADDING.

Paragraphs

```
<!ELEMENT P      - O (%text)*>
<!ATTLIST P
        align  (left|center|right) #IMPLIED
        >
```


The P element is used to mark up paragraphs. It is a container and requires a start tag. The end tag is optional as it can always be inferred by the parser. User agents should place paragraph breaks before and after P elements. The rendering is user agent-dependent, but text is generally wrapped to fit the space available.

Example:

```
<P>This is the first paragraph.
<P>This is the second paragraph.
```

Paragraphs are usually rendered flush left with a ragged right margin. The ALIGN attribute can be used to explicitly specify the horizontal alignment:

align=left	The paragraph is rendered flush left.
align=center	The paragraph is centered.
align=right	The paragraph is rendered flush right.

For example:

```
<p align=center>This is a centered paragraph.
<p align=right>and this is a flush right paragraph.
```

The default is left alignment, but this can be overridden by an enclosing DIV or CENTER element.

Lists

List items can contain block-level and text-level items, including nested lists, although headings and address elements are excluded. This limitation is defined via the %flow entity.

Unordered Lists

```
<!ELEMENT UL - -  (LI)+>
<!ENTITY % ULStyle "disc|square|circle">
<!ATTLIST UL -- unordered lists --
        type    (%ULStyle)    #IMPLIED    -- bullet style --
        compact (compact)     #IMPLIED    -- reduced interitem spacing --
        >
<!ELEMENT LI - O %flow -- list item -->
<!ATTLIST LI
        type    (%LIStyle)    #IMPLIED    -- list item style --
        >
```

Unordered lists take the form:

```
<UL>
      <LI>... first list item
      <LI>... second list item
      ...
    </UL>
```

The UL element is used for unordered lists. Both start and end tags are always needed. The LI element is used for individual list items. The end tag for LI elements can always be omitted. Note that LI elements can contain nested lists. The COMPACT attribute can be used as a hint to the user agent to render lists in a more compact style.

The TYPE attribute can be used to set the bullet style on UL and LI elements. The permitted values are disc, square, or circle. The default generally depends on the level of nesting for lists.

with `<li type=disc>`

with `<li type=square>`

with `<li type=circle>`

This list was chosen to cater for the original bullet shapes used by Mosaic in 1993.

Ordered (Numbered) Lists

```
<!ELEMENT OL - -  (LI)+>
<!ATTLIST OL -- ordered lists --
        type      CDATA      #IMPLIED   -- numbering style --
        start     NUMBER     #IMPLIED   -- starting sequence number --
        compact   (compact)  #IMPLIED   -- reduced interitem spacing --
        >

<!ELEMENT LI - O %flow -- list item -->
<!ATTLIST LI
        type      CDATA      #IMPLIED   -- list item style --
        value     NUMBER     #IMPLIED   -- set sequence number --
        >
```

Ordered (or, *numbered*) lists take the form:

```
<OL>
      <LI>... first list item
      <LI>... second list item
      ...
    </OL>
```

The OL START attribute can be used to initialize the sequence number (by default it is initialized to 1). You can set it later on with the VALUE attribute on LI elements. Both of these attributes expect integer values. You can't indicate that numbering should be continued from a previous list, or to skip missing values without giving an explicit number.

The COMPACT attribute can be used as a hint to the user agent to render lists in a more compact style. The OL TYPE attribute allows you to set the numbering style for list items:

Type	*Numbering style*
1	Arabic numbers: 1, 2, 3, ...
a	Lower alpha: a, b, c, ...
A	Upper alpha: A, B, C, ...
i	Lower Roman: i, ii, iii, ...
I	Upper Roman: I, II, III, ...

Definition Lists

```
<!-- definition lists - DT for term, DD for its definition -->

<!ELEMENT DL    - -  (DT|DD)+>
<!ATTLIST DL
        compact (compact) #IMPLIED -- more compact style --
        >

<!ELEMENT DT - O  (%text)*>
<!ELEMENT DD - O  %flow;>
```

Definition lists take the form:

```
<DL>
    <DT> term name
     term definition
    ...
  </DL>
```

DT elements can only act as containers for text-level elements, while DD elements can hold block-level elements as well, excluding headings and address elements.

For example:

```
<DL>
<DT>Term 1This is the definition of the first term.
<DT>Term 2This is the definition of the second term.
</DL>
```

which could be rendered as:

Term 1 This is the definition of the first term.

Term 2 This is the definition of the second term.

The COMPACT attribute can be used with the DL element as a hint to the user agent to render lists in a more compact style.

DIR and MENU

```
<!ELEMENT (DIR¦MENU) - -  (LI)+ -(%block)>
<!ATTLIST (DIR¦MENU)
        compact (compact) #IMPLIED
        >
```

These elements have been part of HTML from the early days. They are intended for unordered lists similar to UL elements. User agents are recommended to render DIR elements as multicolumn directory lists, and MENU elements as single column menu lists. In practice, Mosaic and most other user agents have ignored this advice and instead render DIR and MENU in an identical way to UL elements.

Preformatted Text

```
<!ELEMENT PRE - - (%text)* -(%pre.exclusion)>
<!ATTLIST PRE
        width NUMBER #implied
        >
```

The PRE element can be used to include preformatted text. User agents render this in a fixed pitch font, preserving spacing associated with white space characters such as space and newline characters. Automatic word-wrap should be disabled within PRE elements.

Note that the SGML standard requires that the parser remove a newline immediately following the start tag or immediately preceding the end tag.

PRE has the same content model as paragraphs, excluding images and elements that produce changes in font size, for example, IMG, BIG, SMALL, SUB, SUP, and FONT.

A few user agents support the WIDTH attribute. It provides a hint to the user agent of the required width in characters. The user agent can use this to select an appropriate font size or to indent the content appropriately.

Here is an example of a PRE element—a verse from Shelley ("To a Skylark"):

```
<PRE>
       Higher still and higher
         From the earth thou springest
       Like a cloud of fire;
         The blue deep thou wingest,
And singing still dost soar, and soaring ever singest.
</PRE>
```

which is rendered as:

```
       Higher still and higher
         From the earth thou springest
       Like a cloud of fire;
         The blue deep thou wingest,
And singing still dost soar, and soaring ever singest.
```


The horizontal tab character (encoded in Unicode, US ASCII, and ISO 8859-1 as decimal 9) should be interpreted as the smallest nonzero number of spaces which will leave the number of characters so far on the line as a multiple of 8. Its use is strongly discouraged since it is common practice when editing to set the tab-spacing to other values, leading to misaligned documents.

XMP, LISTING, and PLAINTEXT

```
<![ %HTML.Deprecated [

<!ENTITY % literal "CDATA"
        -- historical, non-conforming parsing mode where
           the only markup signal is the end tag
           in full
        -->

<!ELEMENT (XMP¦LISTING) - -  %literal>
<!ELEMENT PLAINTEXT - O %literal>

]]>
```

These are obsolete tags for preformatted text that predate the introduction of PRE. User agents may support these for backwards compatibility. Authors should avoid using them in new documents!

DIV and CENTER

```
<!ELEMENT DIV - - %body.content>
<!ATTLIST DIV
        align   (left¦center¦right) #IMPLIED -- alignment of following text --
        >

<!-- CENTER is a shorthand for DIV with ALIGN=CENTER -->
<!ELEMENT center - - %body.content>
```

DIV elements can be used to structure HTML documents as a hierarchy of divisions. The ALIGN attribute can be used to set the default horizontal alignment for elements within the content of the DIV element. Its value is restricted to LEFT, CENTER, or RIGHT, and is defined in the same way as for the paragraph element <P>.

Note that because DIV is a block-like element, it will terminate an open P element. Other than this, user agents are **not** expected to render paragraph breaks before and after DIV elements. CENTER is directly equivalent to DIV with ALIGN=CENTER. Both DIV and CENTER require start and end tags.

CENTER was introduced by Netscape before they added support for the HTML 3.0 DIV element. It is retained in HTML 3.2 on account of its widespread deployment.

BLOCKQUOTE

```
<!ELEMENT BLOCKQUOTE - - %body.content>
```

This is used to enclose block quotations from other works. Both the start and end tags are required. It is often rendered indented, for example:

> They went in single file, running like hounds on a strong scent, and an eager light was in their eyes. Nearly due west the broad swath of the marching Orcs tramped its ugly slot; the sweet grass of Rohan had been bruised and blackened as they passed.
>
> from "The Two Towers" by J.R.R. Tolkien.

FORM

```
<!ENTITY % HTTP-Method "GET ¦ POST"
        -- as per HTTP specification
        -->

<!ELEMENT FORM - - %body.content -(FORM)>
<!ATTLIST FORM
        action %URL #IMPLIED  -- server-side form handler --
        method (%HTTP-Method) GET -- see HTTP specification --
        enctype %Content-Type; "application/x-www-form-urlencoded"
        >
```

This is used to define an HTML form, and you can have more than one form in the same document. Both the start and end tags are required. For very simple forms, you can also use the `ISINDEX` element. Forms can contain a wide range of HTML markup, including several kinds of form fields such as single- and multiline text fields, radio button groups, checkboxes, and menus.

`action`	This specifies a URL which is either used to post forms via e-mail, for example `action="mailto:foo@bar.com"`, or used to invoke a server-side forms handler via HTTP, for example `action="http://www.acme.com/cgi-bin/register.pl"`.
`method`	When the action attribute specifies an HTTP server, the method attribute determines which HTTP method will be used to send the form's contents to the server. It can be either `GET` or `POST`, and defaults to `GET`.
`enctype`	This determines the mechanism used to encode the form's contents. It defaults to `application/x-www-form-urlencoded`.

Further details on handling forms are given in RFC 1867.

HR—Horizontal Rules

Horizontal rules may be used to indicate a change in topic. In a speech-based user agent, the rule could be rendered as a pause.


```
<!ELEMENT HR    - O EMPTY>
<!ATTLIST HR
        align (left¦right¦center) #IMPLIED
        noshade (noshade) #IMPLIED
        size  %Pixels #IMPLIED
        width %Length #IMPLIED
        >
```

`HR` elements are not containers so the end tag is forbidden. The attributes are `ALIGN`, `NOSHADE`, `SIZE`, and `WIDTH`.

`align`	This determines whether the rule is placed at the left, center or right of the space between the current left and right margins for `align=left`, `align=center`, or `align=right`, respectively. By default, the rule is centered.
`noshade`	This attribute requests the user agent to render the rule in a solid color rather than as the traditional two color "groove."
`size`	This can be used to set the height of the rule in pixels.
`width`	This can be used to set the width of the rule in pixels (for example, `width=100`) or as the percentage between the current left and right margins (for example, `width="50%"`). The default is 100%.

Tables

HTML 3.2 includes a widely deployed subset of the specification given in RFC 1942 and can be used to mark up tabular material or for layout purposes. Note that the latter role typically causes problems when rendering to speech or to text-only user agents.

```
<!-- horizontal placement of table relative to window -->
<!ENTITY % Where "(left¦center¦right)">

<!-- horizontal alignment attributes for cell contents -->
<!ENTITY % cell.halign
        "align  (left¦center¦right) #IMPLIED"
        >

<!-- vertical alignment attributes for cell contents -->
<!ENTITY % cell.valign
        "valign  (top¦middle¦bottom)  #IMPLIED"
        >

<!ELEMENT table - - (caption?, tr+)>
<!ELEMENT tr - O (th¦td)*>
<!ELEMENT (th¦td) - O %body.content>

<!ATTLIST table                         -- table element --
        align     %Where;   #IMPLIED  -- table position relative to window --
        width     %Length   #IMPLIED  -- table width relative to window --
        border    %Pixels   #IMPLIED  -- controls frame width around table --
        cellspacing %Pixels #IMPLIED  -- spacing between cells --
        cellpadding %Pixels #IMPLIED  -- spacing within cells --
        >
```

```
<!ELEMENT CAPTION - - (%text;)* -- table or figure caption -->
<!ATTLIST CAPTION
        align (top|bottom) #IMPLIED
        >

<!ATTLIST tr                        -- table row --
        %cell.halign;               -- horizontal alignment in cells --
        %cell.valign;               -- vertical alignment in cells --
        >

<!ATTLIST (th|td)                   -- header or data cell --
        nowrap (nowrap)  #IMPLIED   -- suppress word wrap --
        rowspan NUMBER   1          -- number of rows spanned by cell --
        colspan NUMBER   1          -- number of cols spanned by cell --
        %cell.halign;               -- horizontal alignment in cells --
        %cell.valign;               -- vertical alignment in cells --
        width   %Pixels  #IMPLIED   -- suggested width for cell --
        height  %Pixels  #IMPLIED   -- suggested height for cell --
        >
```

Tables take the general form:

```
<TABLE BORDER=3 CELLSPACING=2 CELLPADDING=2 WIDTH="80%">
<CAPTION> ... table caption ... </CAPTION>
<TR><TD> first cell <TD> second cell
<TR> ...
...
</TABLE>
```

The attributes on TABLE are all optional. By default, the table is rendered without a surrounding border. The table is generally sized automatically to fit the contents, but you can also set the table width using the WIDTH attribute. BORDER, CELLSPACING, and CELLPADDING provide further control over the table's appearance. Captions are rendered at the top or bottom of the table depending on the ALIGN attribute.

Each table row is contained in a TR element, although the end tag can always be omitted. Table cells are defined by TD elements for data and TH elements for headers. Like TR, these are containers and can be given without trailing end tags. TH and TD support several attributes: ALIGN and VALIGN for aligning cell content, ROWSPAN and COLSPAN for cells which span more than one row or column. A cell can contain a wide variety of other block- and text-level elements, including form fields and other tables.

The TABLE element always requires both start and end tags. It supports the following attributes:

align — This takes one of the case insensitive values: LEFT, CENTER, or RIGHT. It specifies the horizontal placement of the table relative to the current left and right margins. It defaults to left alignment, but this can be overridden by an enclosing DIV or CENTER element.

width — In the absence of this attribute, the table width is automatically determined from the table contents. You can use the WIDTH attribute to set the table width to a fixed value in pixels (for example WIDTH=212) or as a percentage of the space between the current left and right margins (for example WIDTH="80%").

border — This attribute can be used to specify the width of the outer border around the table to a given number of pixels (for example BORDER=4). The value can be set to zero to suppress the border altogether. In the absence of this attribute, the border should be suppressed. Note that some browsers also accept <TABLE BORDER> with the same semantics as BORDER=1.

cellspacing — In traditional desktop publishing software, adjacent table cells share a common border. This is not the case in HTML. Each cell is given its own border which is separated from the borders around neighboring cells. This separation can be set in pixels using the CELLSPACING attribute, (for example CELLSPACING=10). The same value also determines the separation between the table border and the borders of the outermost cells.

cellpadding — This sets the padding in pixels between the border around each cell and the cell's contents.

The CAPTION element has one attribute, ALIGN, which can be either ALIGN=TOP or ALIGN=BOTTOM. This can be used to force the caption to be placed above the top or below the bottom of the table, respectively. Most user agents default to placing the caption above the table. CAPTION always requires both start and end tags. Captions are limited to plain text and text-level elements as defined by the %text entity. Block-level elements are not permitted.

The TR or table row element requires a start tag, but the end tag can always be left out. TR acts as a container for table cells. It has two attributes:

align — Sets the default horizontal alignment of cell contents. It takes one of the case insensitive values—LEFT, CENTER, or RIGHT—and plays the same role as the ALIGN attribute on paragraph elements.

valign — This can be used to set the default vertical alignment of cell contents within each cell. It takes one of the case-insensitive values—TOP, MIDDLE, or BOTTOM—to position the cell contents at the top, middle, or bottom of the cell, respectively.

There are two elements for defining table cells. TH is used for header cells and TD for data cells. This distinction allows user agents to render header and data cells in different fonts, and enables speech based browsers to do a better job. The start tags for TH and TD are always needed, but the end tags can be left out. Table cells can have the following attributes:

nowrap	The presence of this attribute disables automatic word wrap within the contents of this cell (for example <TD NOWRAP>). This is equivalent to using the entity for nonbreaking spaces within the content of the cell.
rowspan	This takes a positive integer value specifying the number of rows spanned by this cell. It defaults to one.
colspan	This takes a positive integer value specifying the number of columns spanned by this cell. It defaults to one.
align	Specifies the default horizontal alignment of cell contents, and overrides the ALIGN attribute on the table row. It takes the same values: LEFT, CENTER, and RIGHT. If you don't specify an ALIGN attribute value on the cell, the default is left alignment for <td> and center alignment for <th>, although you can override this with an ALIGN attribute on the TR element.
valign	Specifies the default vertical alignment of cell contents, overriding the VALIGN attribute on the table row. It takes the same values: TOP, MIDDLE, and BOTTOM. If you don't specify a VALIGN attribute value on the cell, the default is middle although you can override this with a VALIGN attribute on the TR element.
width	Specifies the suggested width for a cell content in pixels excluding the cell padding. This value will normally be used except when it conflicts with the width requirements for other cells in the same column.
height	Specifies the suggested height for a cell content in pixels excluding the cell padding. This value will normally be used except when it conflicts with the height requirements for other cells in the same row.

Tables are commonly rendered in bas-relief, raised up with the outer border as a bevel, and individual cells inset into this raised surface. Borders around individual cells are only drawn if the cell has explicit content. White space doesn't count for this purpose with the exception of .

The algorithms used to automatically size tables should take into account the minimum and maximum width requirements for each cell. This is used to determine the minimum and maximum width requirements for each column and hence for the table itself.

Cells spanning more than one column contribute to the widths of each of the columns spanned. One approach is to evenly apportion the cell's minimum and maximum width between these columns; another is to weight the apportioning according to the contributions from cells that don't span multiple columns.

For some user agents it may be necessary or desirable to break text lines within words. In such cases a visual indication that this has occurred is advised.

The minimum and maximum width of nested tables contribute to the minimum and maximum width of the cell in which they occur. Once the width requirements are known for the top-level table, the column widths for that table can be assigned. This allows the widths of nested tables to be assigned and hence, in turn, the column widths of such tables. If practical, all columns should be assigned at least their minimum widths. It is suggested that any surplus space is then shared out proportional to the difference between the minimum and maximum width requirements of each column.

Note that pixel values for width and height refer to screen pixels, and should be multiplied by an appropriate factor when rendering to very high resolution devices such as laser printers. For instance, if a user agent has a display with 75 pixels per inch and is rendering to a laser printer with 600 dots per inch, then the pixel values given in HTML attributes should be multiplied by a factor of 8.

Text-Level Elements

These don't cause paragraph breaks. Text-level elements that define character styles can generally be nested. They can contain other text-level elements but not block-level elements.

- Font style elements
- Phrase elements
- Form fields
- The A (anchor) element
- IMG—inline images
- APPLET *(Java Applets)*
- FONT elements
- BASEFONT elements
- BR—line breaks
- MAP—client-side image maps

Font Style Elements

These all require start and end tags, for example:

```
This has some <B>bold text.</B>
```

Text-level elements must be properly nested; the following is in error:

```
This has some <B>bold and <I></B>italic text</I>.
```

User agents should do their best to respect nested emphasis, for example:

```
This has some <B>bold and <I>italic text</I></B>.
```

Where the available fonts are restricted or for speech output, alternative means should be used for rendering differences in emphasis.

TT	Teletype or monospaced text
I	Italic text style
B	Bold text style
U	Underlined text style
STRIKE	Strike-through text style
BIG	Places text in a large font
SMALL	Places text in a small font
SUB	Places text in subscript style
SUP	Places text in superscript style

Note: future revisions to HTML may phase out `STRIKE` *in favor of the more concise* `S` *tag from HTML 3.0.*

Phrase Elements

These all require start and end tags, for example:

```
This has some <EM>emphasized text</EM>.
```

EM	Basic emphasis typically rendered in an italic font
STRONG	Strong emphasis typically rendered in a bold font
DFN	Defining instance of the enclosed term
CODE	Used for extracts from program code
SAMP	Used for sample output from programs, scripts, etc.
KBD	Used for text to be typed by the user
VAR	Used for variables or arguments to commands
CITE	Used for citations or references to other sources

Form Fields

`INPUT`, `SELECT`, and `TEXTAREA` are only allowed within `FORM` elements. `INPUT` can be used for a variety of form fields, including single-line text fields, password fields, checkboxes, radio buttons, submit and reset buttons, hidden fields, file upload, and image buttons. `SELECT` elements are used for single or multiple choice menus. `TEXTAREA` elements are used to define multiline text fields. The content of the element is used to initialize the field.

INPUT Text Fields, Radio Buttons, Checkboxes...

`INPUT` elements are not containers, so the end tag is forbidden.

```
<!ENTITY % IAlign "(top|middle|bottom|left|right)">

<!ENTITY % InputType
        "(TEXT | PASSWORD | CHECKBOX | RADIO | SUBMIT
            | RESET | FILE | HIDDEN | IMAGE)">

<!ELEMENT INPUT - O EMPTY>
<!ATTLIST INPUT
        type %InputType TEXT      -- what kind of widget is needed --
        name  CDATA #IMPLIED      -- required for all but submit and reset --
        value CDATA #IMPLIED      -- required for radio and checkboxes --
        checked (checked) #IMPLIED -- for radio buttons and check boxes --
        size CDATA  #IMPLIED      -- specific to each type of field --
        maxlength NUMBER #IMPLIED
        src   %URL  #IMPLIED      -- for fields with background images --
        align %IAlign #IMPLIED    -- vertical or horizontal alignment --
        >
```

type

Used to set the type of input field, as follows:

type=text (the default)

A single-line text field whose visible size can be set using the size attribute, for example `size=40` for a 40 character wide field. Users should be able to type more than this limit, though, with the text scrolling through the field to keep the input cursor in view. You can enforce an upper limit on the number of characters that can be entered with the `MAXLENGTH` attribute. The `NAME` attribute is used to name the field, while the `VALUE` attribute can be used to initialize the text string shown in the field when the document is first loaded.

```
<input type=text size=40 name=user value="your name">
```

type=password

This is like `type=text`, but echoes characters using a character like `*` to hide the text from prying eyes when entering passwords. You can use `SIZE` and `MAXLENGTH` attributes to control the visible and maximum length exactly as per regular text fields.

```
<input type=password size=12 name=pw>
```

type=checkbox

Used for simple Boolean attributes, or for attributes that can take multiple values at the same time. The latter is represented by several checkbox fields with the same name

and a different value attribute. Each checked checkbox generates a separate name/value pair in the submitted data, even if this results in duplicate names. Use the checked attribute to initialize the checkbox to its checked state.

```
<input type=checkbox checked name=uscitizen value=yes>
```

type=radio

Used for attributes which can take a single value from a set of alternatives. Each radio button field in the group should be given the same name. Radio buttons require an explicit value attribute. Only the checked radio button in the group generates a name/value pair in the submitted data. One radio button in each group should be initially checked using the checked attribute.

```
<input type=radio name=age value="0-12">
<input type=radio name=age value="13-17">
<input type=radio name=age value="18-25">
<input type=radio name=age value="26-35" checked>
<input type=radio name=age value="36-">
```

type=submit

This defines a button that users can click to submit the form's contents to the server. The button's label is set from the VALUE attribute. If the NAME attribute is given, then the submit button's name/value pair will be included in the submitted data. You can include several submit buttons in the form. See type=image for graphical submit buttons.

```
<input type=submit value="Party on ...">
```

type=image

This is used for graphical submit buttons rendered by an image rather than a text string. The URL for the image is specified with the SRC attribute. The image alignment can be specified with the ALIGN attribute. In this respect, graphical submit buttons are treated identically to IMG elements, so you can set align to left, right, top, middle, or bottom. The x and y values of the location clicked are passed to the server: In the submitted data, image fields are included as two name/value pairs. The names are derived by taking the name of the field and appending ".x" for the x value, and ".y" for the y value.

```
<p>Now choose a point on the map:

    <input type=image name=point src="map.gif">
```

Note: image fields typically cause problems for text-only and speech-based user agents!

type=reset

This defines a button that users can click to reset form fields to their initial state when the document was first loaded. You can set the label by providing a value attribute. Reset buttons are never sent as part of the form's contents.

```
<input type=reset value="Start over ...">
```


type=file

This provides a means for users to attach a file to the form's contents. It is generally rendered by text field and an associated button which when clicked invokes a file browser to select a filename. The filename can also be entered directly in the text field. Just like type=text, you can use the SIZE attribute to set the visible width of this field in average character widths. You can set an upper limit to the length of filenames using the MAXLENGTH attribute. Some user agents support the ability to restrict the kinds of files to those matching a comma-separated list of MIME content types given with the ACCEPT attribute, for example accept="image/*" restricts files to images. Further information can be found in RFC 1867.

```
<input type=file name=photo size=20 accept="image/*">
```

type=hidden

These fields should not be rendered and provide a means for servers to store state information with a form. This will be passed back to the server when the form is submitted, using the name/value pair defined by the corresponding attributes. This is a work around for the statefulness of HTTP. Another approach is to use HTTP "Cookies."

```
<input type=hidden name=customerid value="c2415-345-8563">
```

name

Used to define the property name that will be used to identify this field's content when it is submitted to the server.

value

Used to initialize the field, or to provide a textual label for submit and reset buttons.

checked

The presence of this attribute is used to initialize checkboxes and radio buttons to their checked state.

size

Used to set the visible size of text fields to a given number of average character widths, for example size=20.

maxlength

Sets the maximum number of characters permitted in a text field.

src

Specifies a URL for the image to use with a graphical submit button.

align

Used to specify image alignment for graphical submit buttons. It is defined just like the IMG align attribute and takes one of the values top, middle, bottom, left or right, defaulting to bottom.

SELECT Menus

```
<!ELEMENT SELECT - - (OPTION+)>
<!ATTLIST SELECT
        name CDATA #REQUIRED
        size NUMBER #IMPLIED
        multiple (multiple) #IMPLIED
        >

<!ELEMENT OPTION - O (#PCDATA)*>
<!ATTLIST OPTION
        selected (selected) #IMPLIED
        value  CDATA  #IMPLIED -- defaults to element content --
        >
```

SELECT is used to define select one from many or many from many menus. SELECT elements require start and end tags and contain one or more OPTION elements that define menu items. One from many menus are generally rendered as drop-down menus while many from many menus are generally shown as list boxes.

Example:

```
<SELECT NAME="flavor">
<OPTION VALUE=a>Vanilla
<OPTION VALUE=b>Strawberry
<OPTION VALUE=c>Rum and Raisin
<OPTION VALUE=d>Peach and Orange
</SELECT>
```

SELECT attributes:

name	This specifies a property name that is used to identify the menu choice when the form is submitted to the server. Each selected option results in a property name/value pair being included as part of the form's contents.
size	This sets the number of visible choices for many menus.
multiple	The presence of this attribute signifies that the users can make multiple selections. By default only one selection is allowed.

OPTION attributes:

selected	When this attribute is present, the option is selected when the document is initially loaded. It is an error for more than one option to be so selected for one from many menus.
value	Specifies the property value to be used when submitting the form's content. This is combined with the property name as given by the NAME attribute of the parent SELECT element.

TEXTAREA Multiline Text Fields

```
<!-- Multi-line text input field. -->

<!ELEMENT TEXTAREA - - (#PCDATA)*>
<!ATTLIST TEXTAREA
        name CDATA #REQUIRED
        rows NUMBER #REQUIRED
        cols NUMBER #REQUIRED
        >
```

TEXTAREA elements require start and end tags. The content of the element is restricted to text and character entities. It is used to initialize the text that is shown when the document is first loaded.

Example:

```
<TEXTAREA NAME=address ROWS=4 COLS=40>
Your address here ...
</TEXTAREA>
```

It is recommended that user agents canonize line endings to CR, LF (ASCII decimal 13, 10) when submitting the field's contents. The character set for submitted data should be ISO Latin-1, unless the server has previously indicated that it can support alternative character sets.

name	This specifies a property name that is used to identify the textarea field when the form is submitted to the server.
rows	Specifies the number of visible text lines. Users should be able to enter more lines that this, so user agents should provide some means to scroll through the contents of the textarea field when the contents extend beyond the visible area.
cols	Specifies the visible width in average character widths. Users should be able to enter longer lines that this, so user agents should provide some means to scroll through the contents of the textarea field when the contents extend beyond the visible area. User agents may wrap visible text lines to keep long lines visible without the need for scrolling.

Special Text-Level Elements

A (anchor), IMG, APPLET, FONT, BASEFONT, BR, and MAP.

The A (anchor) Element

```
<!ELEMENT A - - (%text)* -(A)>
<!ATTLIST A
        name    CDATA   #IMPLIED    -- named link end --
        href    %URL    #IMPLIED    -- URL for linked resource --
        rel     CDATA   #IMPLIED    -- forward link types --
        rev     CDATA   #IMPLIED    -- reverse link types --
        title   CDATA   #IMPLIED    -- advisory title string --
        >
```

Anchors can't be nested and always require start and end tags. They are used to define hypertext links and also to define named locations for use as targets for hypertext links, for example:

```
The way to <a href="hands-on.html">happiness</a>.
```

They are also used to define named locations for use as targets for hypertext links, for example:

```
<h2><a name=mit>545 Tech Square - Hacker's Paradise</a></h2>
```

`name` This should be a string defining unique name for the scope of the current HTML document. `NAME` is used to associate a name with this part of a document for use with URLs that target a named section of a document.

`href` Specifies a URL acting as a network address for the linked resource. This could be another HTML document, a PDF file, an image, and so on.

`rel` The forward relationship also known as the "link type". It can be used to determine to how deal with the linked resource when printing out a collection of linked resources.

`rev` This defines a reverse relationship. A link from document A to document B with `REV=relation` expresses the same relationship as a link from B to A with `REL=relation`. `REV=made` is sometimes used to identify the document author, either the author's e-mail address with a mailto URL, or a link to the author's home page.

`title` An advisory title for the linked resource.

IMG—Inline Images

```
<!ENTITY % IAlign "(top|middle|bottom|left|right)">

<!ELEMENT IMG     - O EMPTY --  Embedded image -->
<!ATTLIST IMG
        src     %URL     #REQUIRED  -- URL of image to embed --
        alt     CDATA    #IMPLIED   -- for display in place of image --
        align   %IAlign  #IMPLIED   -- vertical or horizontal alignment --
        height  %Pixels  #IMPLIED   -- suggested height in pixels --
        width   %Pixels  #IMPLIED   -- suggested width in pixels --
        border  %Pixels  #IMPLIED   -- suggested link border width --
        hspace  %Pixels  #IMPLIED   -- suggested horizontal gutter --
        vspace  %Pixels  #IMPLIED   -- suggested vertical gutter --
        usemap  %URL     #IMPLIED   -- use client-side image map --
        ismap   (ismap)  #IMPLIED   -- use server image map --
        >
```


Used to insert images. `IMG` is an empty element and so the end tag is forbidden. Images can be positioned vertically relative to the current textline or floated to the left or right. See `BR` with the `CLEAR` attribute for control over textflow.

```
e.g. <IMG SRC="canyon.gif" ALT="Grand Canyon">
```

`IMG` elements support the following attributes:

`src` This attribute is required for every `IMG` element. It specifies a URL for the image resource, for instance a GIF, JPEG or PNG image file.

`alt` This is used to provide a text description of the image and is vital for interoperability with speech-based and text only user agents.

`align` This specifies how the image is positioned relative to the current textline in which it occurs:

`align=top` positions the top of the image with the top of the current text line. User agents vary in how they interpret this. Some only take into account what has occurred on the text line prior to the `IMG` element and ignore what happens after it.

`align=middle` aligns the middle of the image with the baseline for the current textline.

`align=bottom` is the default and aligns the bottom of the image with the baseline.

`align=left` floats the image to the current left margin, temporarily changing this margin, so that subsequent text is flowed along the image's righthand side. The rendering depends on whether there is any left aligned text or images that appear earlier than the current image in the markup. Such text (but not images) generally forces left-aligned images to wrap to a new line, with the subsequent text continuing on the former line.

`align=right` floats the image to the current right margin, temporarily changing this margin, so that subsequent text is flowed along the image's lefthand side. The rendering depends on whether there is any right aligned text or images that appear earlier than the current image in the markup. Such text (but not images) generally forces right-aligned images to wrap to a new line, with the subsequent text continuing on the former line.

Note that some browsers introduce spurious spacing with multiple left- right-aligned images. As a result, authors can't depend on this being the same for browsers from different vendors. See BR for ways to control text flow.

width	Specifies the intended width of the image in pixels. When given together with the height, this allows user agents to reserve screen space for the image before the image data has arrived over the network.
height	Specifies the intended height of the image in pixels. When given together with the width, this allows user agents to reserve screen space for the image before the image data has arrived over the network.
border	When the IMG element appears as part of a hypertext link, the user agent will generally indicate this by drawing a colored border (typically blue) around the image. This attribute can be used to set the width of this border in pixels. Use border=0 to suppress the border altogether. User agents are recommended to provide additional cues that the image is clickable, for example by changing the mouse pointer.
hspace	This can be used to provide white space to the immediate left and right of the image. The HSPACE attribute sets the width of this white space in pixels. By default HSPACE is a small, nonzero number.
vspace	This can be used to provide white space above and below the image The VSPACE attribute sets the height of this white space in pixels. By default VSPACE is a small, nonzero number.
usemap	This can be used to give a URL fragment identifier for a client-side image map defined with the MAP element.
ismap	When the IMG element is part of a hypertext link, and the user clicks on the image, the ISMAP attribute causes the location to be passed to the server. This mechanism causes problems for text-only and speech-based user agents. Whenever it's possible to do so, use the MAP element instead.

Here is an example of how you use ISMAP:

```
<a href="/cgibin/navbar.map"><img src=navbar.gif ismap border=0></a>
```

The location clicked is passed to the server as follows. The user agent derives a new URL from the URL specified by the HREF attribute by appending ?, the x coordinate, and the y coordinate of the location in pixels. The link is then followed using the new URL. For instance, if the user clicked at the location x=10, y=27 then the derived URL will be "/cgibin/navbar.map?10,27". It is generally a good idea to suppress the border and use graphical idioms to indicate that the image is clickable.

Note that pixel values refer to screen pixels, and should be multiplied by an appropriate factor when rendering to very high resolution devices such as laser printers. For instance, if a user agent has a display with 75 pixels per inch and is rendering to a laser printer with 600 dots per inch, then the pixel values given in HTML attributes should be multiplied by a factor of 8.

APPLET (Java Applets)

```
<!ELEMENT APPLET - - (PARAM ¦ %text)*>
<!ATTLIST APPLET
        codebase %URL      #IMPLIED   -- code base --
        code     CDATA     #REQUIRED  -- class file --
        alt      CDATA     #IMPLIED   -- for display in place of applet --
        name     CDATA     #IMPLIED   -- applet name --
        width    %Pixels   #REQUIRED  -- suggested width in pixels --
        height   %Pixels   #REQUIRED  -- suggested height in pixels --
        align    %IAlign   #IMPLIED   -- vertical or horizontal alignment --
        hspace   %Pixels   #IMPLIED   -- suggested horizontal gutter --
        vspace   %Pixels   #IMPLIED   -- suggested vertical gutter --
        >

<!ELEMENT PARAM - O EMPTY>
<!ATTLIST PARAM
    name    NMTOKEN   #REQUIRED  -- The name of the parameter --
    value   CDATA     #IMPLIED   -- The value of the parameter --
    >
```

Requires start and end tags. This element is supported by all Java-enabled browsers. It allows you to embed a Java applet into HTML documents. APPLET uses associated PARAM elements to pass parameters to the applet. Following the PARAM elements, the content of APPLET elements should be used to provide an alternative to the applet for user agents that don't support Java. It is restricted to text-level markup as defined by the %text entity in the DTD. Java-compatible browsers ignore this extra HTML code. You can use it to show a snapshot of the applet running, with text explaining what the applet does. Other possibilities for this area are a link to a page that is more useful for the Java-ignorant browser, or text that taunts the user for not having a Java-compatible browser.

Here is a simple example of a Java applet:

```
<applet code="Bubbles.class" width=500 height=500>
Java applet that draws animated bubbles.
</applet>
```

Here is another one using a PARAM element:

```
<applet code="AudioItem" width=15 height=15>
<param name=snd value="Hello.au¦Welcome.au">
Java applet that plays a welcoming sound.
</applet>
```

`codebase = codebaseURL`	This optional attribute specifies the base URL of the applet—the directory or folder that contains the applet's code. If this attribute is not specified, then the document's URL is used.
`code = appletFile`	This required attribute gives the name of the file that contains the applet's compiled Applet subclass. This file is relative to the base URL of the applet. It cannot be absolute.
`alt = alternateText`	This optional attribute specifies any text that should be displayed if the browser understands the `<APPLET>` tag but can't run Java applets.
`name = appletInstanceName`	This optional attribute specifies a name for the applet instance, which makes it possible for applets on the same page to find (and communicate with) each other.
`width = pixels` `height = pixels`	These required attributes give the initial width and height (in pixels) of the applet display area, not counting any windows or dialogs that the applet brings up.
`align = alignment`	This attribute specifies the alignment of the applet. This attribute is defined in exactly the same way as the `IMG` element. The permitted values are top, middle, bottom, left, and right. The default is bottom.
`vspace = pixels` `hspace = pixels`	These optional attributes specify the number of pixels above and below the applet (`VSPACE`) and on each side of the applet (`HSPACE`). They're treated the same way as the IMG element's `VSPACE` and `HSPACE` attributes.

The `PARAM` element is used to pass named parameters to applet:

```
<PARAM NAME = appletParameter VALUE = value>
```

`PARAM` elements are the only way to specify applet-specific parameters. Applets read user-specified values for parameters with the `getParameter()` method.

`name = applet parameter name`

`value = parameter value`

SGML character entities such `é`, `"` and `¹` are expanded before the parameter value is passed to the applet. To include an `&` character, use `&`.

Note: `PARAM` elements should be placed at the start of the content for the `APPLET` element. This is not specified as part of the DTD due to technicalities with SGML mixed content models.

FONT

```
<!ELEMENT FONT - - (%text)*      -- local change to font -->
<!ATTLIST FONT
    size    CDATA   #IMPLIED    -- [+]nn e.g. size="+1", size=4 --
    color   CDATA   #IMPLIED    -- #RRGGBB in hex, e.g. red: color="#FF0000" --
    >
```

Requires start and end tags. This allows you to change the font size and/or color for the enclosed text.

The attributes are `SIZE` and `COLOR`. Font sizes are given in terms of a scalar range defined by the user agent with no direct mapping to point sizes etc. The `FONT` element may be phased out in future revisions to HTML.

`size`	This sets the font size for the contents of the font element. You can set size to an integer ranging from 1 to 7 for an absolute font size, or specify a relative font size with a signed integer value, for example `size="+1"` or `size="-2"`. This is mapped to an absolute font size by adding the current base font size as set by the `BASEFONT` element (see following).
`color`	Used to set the color to stroke the text. Colors are given as RGB in hexadecimal notation or as one of 16 widely understood color names defined as per the `BGCOLOR` attribute on the `BODY` element.

Some user agents also support a `FACE` attribute which accepts a comma separated list of font names in order of preference. This is used to search for an installed font with the corresponding name. `FACE` is not part of HTML 3.2.

The following shows the effects of setting font to absolute sizes:

size=1 size=2 size=3 size=4 size=5 size=6 size=7

The following shows the effect of relative font sizes using a base font size of 3:

size=-4 size=-3 size=-2 size=-1 size=+1 size=+2 size=+3 size=+4

The same thing with a base font size of 6:

size=-4 size=-3 size=-2 size=-1 size=+1 size=+2 size=+3 size=+4

BASEFONT

```
<!ELEMENT BASEFONT - O EMPTY     -- base font size (1 to 7) -->
<!ATTLIST BASEFONT
    size     CDATA   #IMPLIED    -- e.g. size=4, defaults to 3 --
    >
```

Used to set the base font size. BASEFONT is an empty element, so the end tag is forbidden. The SIZE attribute is an integer value ranging from 1 to 7. The base font size applies to the normal and preformatted text but not to headings, except where these are modified using the FONT element with a relative font size.

BR

Used to force a line break. This is an empty element, so the end tag is forbidden. The CLEAR attribute can be used to move down past floating images on either margin. <BR CLEAR=LEFT> moves down past floating images on the left margin, <BR CLEAR=RIGHT> does the same for floating images on the right margin, while <BR CLEAR=ALL> does the same for such images on both left and right margins.

MAP

The MAP element provides a mechanism for client-side image maps. These can be placed in the same document or grouped in a separate document, although this isn't yet widely supported. The MAP element requires start and end tags. It contains one or more AREA elements that specify hotzones on the associated image and bind these hotzones to URLs.

```
<!ENTITY % SHAPE "(rect|circle|poly)">
<!ENTITY % COORDS "CDATA" -- comma separated list of numbers -->

<!ELEMENT MAP - - (AREA)+>
<!ATTLIST MAP
    name     CDATA   #REQUIRED
    >

<!ELEMENT AREA - O EMPTY>
<!ATTLIST AREA
    shape    %SHAPE  rect
    coords   %COORDS #IMPLIED  -- defines coordinates for shape --
    href     %URL    #IMPLIED  -- this region acts as hypertext link --
    nohref (nohref) #IMPLIED  -- this region has no action --
    alt      CDATA   #REQUIRED -- needed for non-graphical user agents --
    >
```

Here is a simple example for a graphical navigational toolbar:

```
<img src="navbar.gif" border=0 usemap="#map1">

<map name="map1">
 <area href=guide.html alt="Access Guide" shape=rect coords="0,0,118,28">
 <area href=search.html alt="Search" shape=rect coords="184,0,276,28">
 <area href=shortcut.html alt="Go" shape=rect coords="118,0,184,28">
 <area href=top10.html alt="Top Ten" shape=rect coords="276,0,373,28">
</map>
```


The MAP element has one attribute, NAME, which is used to associate a name with a map. This is then used by the USEMAP attribute on the IMG element to reference the map via a URL fragment identifier. Note that the value of the NAME attribute is case sensitive.

The AREA element is an empty element, and so the end tag is forbidden. It takes the following attributes: SHAPE, COORDS, HREF, NOHREF, and ALT. The SHAPE and COORDS attributes define a region on the image. If the SHAPE attribute is omitted, SHAPE="RECT" is assumed.

```
shape=rect coords="left-x, top-y, right-x, bottom-y"
shape=circle coords="center-x, center-y, radius"
shape=poly coords="x1,y1,x2,y2,x3,y3, ..."
```

Where x and y are measured in pixels from the left/top of the associated image. If x and y values are given with a percent sign as a suffix, the values should be interpreted as percentages of the image's width and height, respectively. For example:

```
SHAPE=RECT COORDS="0, 0, 50%, 100%"
```

The HREF attribute gives a URL for the target of the hypertext link. The NOHREF attribute is used when you want to define a region that doesn't act as a hotzone. This is useful when you want to cut a hole in an underlying region acting as a hotzone.

If two or more regions overlap, the region defined first in the map definition takes precedence over subsequent regions. This means that AREA elements with NOHREF should generally be placed before ones with the HREF attribute.

The ALT attribute is used to provide text labels which can be displayed in the status line as the mouse or other pointing device is moved over hotzones, or for constructing a textual menu for nongraphical user agents. Authors are *strongly recommended* to provide meaningful ALT attributes to support interoperability with speech-based or text-only user agents.

Sample SGML Open Catalog for HTML 3.2

This can be used with an SGML parser like nsgmls to verify that files conform to the HTML 3.2 DTD. It assumes that the DTD has been saved as the file HTML32.dtd and that the Latin-1 entities are in the file ISOlat1.ent.

```
-- html32.soc: catalog for parsing HTML 3.2 documents --
SGMLDECL "HTML32.dcl"
PUBLIC "-//W3C//DTD HTML 3.2 Final//EN" HTML32.dtd
PUBLIC "-//W3C//DTD HTML 3.2 Draft//EN" HTML32.dtd
PUBLIC "-//W3C//DTD HTML 3.2//EN" HTML32.dtd
PUBLIC "ISO 8879-1986//ENTITIES Added Latin 1//EN//HTML" ISOlat1.ent
```

SGML Declaration for HTML 3.2

This uses the eight-bit ISO Latin-1 character set. The size limits on properties like literals and tag names have been considerably increased from their HTML 2.0 values, but it is recommended that user agents avoid imposing arbitrary length limits.

```
<!SGML  "ISO 8879:1986"
   --
        SGML Declaration for HyperText Markup Language version 3.2

        With support for ISO Latin-1 and increased limits
        for tag and literal lengths etc.
   --

   CHARSET
         BASESET  "ISO 646:1983//CHARSET
                   International Reference Version
                   (IRV)//ESC 2/5 4/0"
         DESCSET  0   9   UNUSED
                  9   2   9
                  11  2   UNUSED
                  13  1   13
                  14  18  UNUSED
                  32  95  32
                  127 1   UNUSED
         BASESET  "ISO Registration Number 100//CHARSET
                   ECMA-94 Right Part of
                   Latin Alphabet Nr. 1//ESC 2/13 4/1"
         DESCSET  128  32   UNUSED
                  160  96    32

   CAPACITY   SGMLREF
              TOTALCAP         200000
              GRPCAP           150000
              ENTCAP           150000

   SCOPE    DOCUMENT
   SYNTAX
      SHUNCHAR CONTROLS 0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16
               17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 127
      BASESET  "ISO 646:1983//CHARSET
                International Reference Version
                (IRV)//ESC 2/5 4/0"
      DESCSET  0 128 0

      FUNCTION
               RE             13
               RS             10
               SPACE          32
               TAB SEPCHAR     9

      NAMING   LCNMSTRT ""
               UCNMSTRT ""
               LCNMCHAR ".-"
               UCNMCHAR ".-"
               NAMECASE GENERAL YES
                        ENTITY  NO
      DELIM    GENERAL  SGMLREF
               SHORTREF SGMLREF
      NAMES    SGMLREF
```



```
      QUANTITY SGMLREF
               ATTSPLEN 65536
               LITLEN   65536
               NAMELEN  65536
               PILEN    65536
               TAGLVL   100
               TAGLEN   65536
               GRPGTCNT 150
               GRPCNT   64

   FEATURES
     MINIMIZE
       DATATAG  NO
       OMITTAG  YES
       RANK     NO
       SHORTTAG YES
     LINK
       SIMPLE   NO
       IMPLICIT NO
       EXPLICIT NO
     OTHER
       CONCUR   NO
       SUBDOC   NO
       FORMAL   YES
   APPINFO      NONE
>
```

HTML 3.2 Document Type Definition

```
<!--
        W3C Document Type Definition for the HyperText Markup Language
        version 3.2 as ratified by a vote of W3C member companies.
        For more information on W3C look at  URL http://www.w3.org/

        Date: Tuesday January 14th 1997

        Author: Dave Raggett <dsr@w3.org>

        HTML 3.2 aims to capture recommended practice as of early '96
        and as such to be used as a replacement for HTML 2.0 (RFC 1866).
        Widely deployed rendering attributes are included where they
        have been shown to be interoperable. SCRIPT and STYLE are
        included to smooth the introduction of client-side scripts
        and style sheets. Browsers must avoid showing the contents
        of these element Otherwise support for them is not required.
        ID, CLASS and STYLE attributes are not included in this version
        of HTML.
-->

<!ENTITY % HTML.Version
        "-//W3C//DTD HTML 3.2 Final//EN"

        -- Typical usage:
```

```
            <!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 3.2 Final//EN">
            <html>
            ...
            </html>
        --
        >

<!--================== Deprecated Features Switch ============================-->

<!ENTITY % HTML.Deprecated "INCLUDE">

<!--================== Imported Names ========================================-->

<!ENTITY % Content-Type "CDATA"
        -- meaning a MIME content type, as per RFC1521
        -->

<!ENTITY % HTTP-Method "GET | POST"
        -- as per HTTP specification
        -->

<!ENTITY % URL "CDATA"
        -- The term URL means a CDATA attribute
           whose value is a Uniform Resource Locator,
           See RFC1808 (June 95) and RFC1738 (Dec 94).
        -->

<!-- Parameter Entities -->

<!ENTITY % head.misc "SCRIPT|STYLE|META|LINK" -- repeatable head elements -->

<!ENTITY % heading "H1|H2|H3|H4|H5|H6">

<!ENTITY % list "UL | OL |  DIR | MENU">

<![ %HTML.Deprecated [
    <!ENTITY % preformatted "PRE | XMP | LISTING">
]]>

<!ENTITY % preformatted "PRE">

<!--================ Character mnemonic entities =============================-->

<!ENTITY % ISOlat1 PUBLIC
      "ISO 8879-1986//ENTITIES Added Latin 1//EN//HTML">
%ISOlat1;

<!--================ Entities for special symbols ============================-->
<!-- &trade and &cbsp are not widely deployed and so not included here -->

<!ENTITY amp    CDATA "&#38;"   -- ampersand          -->
<!ENTITY gt     CDATA "&#62;"   -- greater than       -->
<!ENTITY lt     CDATA "&#60;"   -- less than          -->
```



```
<!--================== Text Markup ===========================================-->

<!ENTITY % font "TT | I | B  | U | STRIKE | BIG | SMALL | SUB | SUP">

<!ENTITY % phrase "EM | STRONG | DFN | CODE | SAMP | KBD | VAR | CITE">

<!ENTITY % special "A | IMG | APPLET | FONT | BASEFONT | BR | SCRIPT | MAP">

<!ENTITY % form "INPUT | SELECT | TEXTAREA">

<!ENTITY % text "#PCDATA | %font | %phrase | %special | %form">

<!ELEMENT (%font|%phrase) - - (%text)*>

<!-- there are also 16 widely known color names although
  the resulting colors are implementation dependent:

   aqua, black, blue, fuchsia, gray, green, lime, maroon,
   navy, olive, purple, red, silver, teal, white, and yellow

 These colors were originally picked as being the standard
 16 colors supported with the Windows VGA palette.
 -->

<!ELEMENT FONT - - (%text)*     -- local change to font -->
<!ATTLIST FONT
    size    CDATA   #IMPLIED    -- [+]nn e.g. size="+1", size=4 --
    color   CDATA   #IMPLIED    -- #RRGGBB in hex, e.g. red: color="#FF0000" --
    >

<!ELEMENT BASEFONT - O EMPTY    -- base font size (1 to 7)-->
<!ATTLIST BASEFONT
    size    CDATA   #IMPLIED    -- e.g. size=3 --
    >

<!ELEMENT BR    - O EMPTY    -- forced line break -->
<!ATTLIST BR
        clear (left|all|right|none) none -- control of text flow --
        >

<!--================== HTML content models ===================================-->
<!--
    HTML has three basic content models:

        %text       character level elements and text strings
        %flow       block-like elements e.g. paragraphs and lists
        %bodytext   as %flow plus headers H1-H6 and ADDRESS
-->

<!ENTITY % block
     "P | %list | %preformatted | DL | DIV | CENTER |
      BLOCKQUOTE | FORM | ISINDEX | HR | TABLE">

<!-- %flow is used for DD and LI -->

<!ENTITY % flow "(%text | %block)*">
```

```
<!--==================== Document Body ====================================-->

<!ENTITY % body.content "(%heading ¦ %text ¦ %block ¦ ADDRESS)*">

<!ENTITY % color "CDATA" -- a color specification: #HHHHHH @@ details? -->

<!ENTITY % body-color-attrs "
        bgcolor %color #IMPLIED
        text %color #IMPLIED
        link %color #IMPLIED
        vlink %color #IMPLIED
        alink %color #IMPLIED
        ">

<!ELEMENT BODY O O  %body.content>
<!ATTLIST BODY
        background %URL #IMPLIED  -- texture tile for document background --
        %body-color-attrs;  -- bgcolor, text, link, vlink, alink --
        >

<!ENTITY % address.content "((%text;) ¦ P)*">

<!ELEMENT ADDRESS - - %address.content>

<!ELEMENT DIV - - %body.content>
<!ATTLIST DIV
        align   (left¦center¦right) #IMPLIED -- alignment of following text --
        >

<!-- CENTER is a shorthand for DIV with ALIGN=CENTER -->
<!ELEMENT center - - %body.content>

<!--==================== The Anchor Element ================================-->

<!ELEMENT A - - (%text)* -(A)>
<!ATTLIST A
        name    CDATA   #IMPLIED    -- named link end --
        href    %URL    #IMPLIED    -- URL for linked resource --
        rel     CDATA   #IMPLIED    -- forward link types --
        rev     CDATA   #IMPLIED    -- reverse link types --
        title   CDATA   #IMPLIED    -- advisory title string --
        >

<!--==================== Client-side image maps ============================-->

<!-- These can be placed in the same document or grouped in a
     separate document although this isn't yet widely supported -->

<!ENTITY % SHAPE "(rect¦circle¦poly)">
<!ENTITY % COORDS "CDATA" -- comma separated list of numbers -->

<!ELEMENT MAP - - (AREA)*>
<!ATTLIST MAP
    name    CDATA   #IMPLIED
    >
```



```
<!ELEMENT AREA - O EMPTY>
<!ATTLIST AREA
    shape   %SHAPE  rect
    coords  %COORDS #IMPLIED  -- defines coordinates for shape --
    href    %URL    #IMPLIED  -- this region acts as hypertext link --
    nohref (nohref) #IMPLIED  -- this region has no action --
    alt     CDATA   #REQUIRED -- needed for non-graphical user agents --
    >

<!—==================== The LINK Element ================================—>

<!ENTITY % Types "CDATA"
        -- See Internet Draft: draft-ietf-html-relrev-00.txt
           LINK has been part of HTML since the early days
           although few browsers as yet take advantage of it.

           Relationship values can be used in principle:

                a) for document specific toolbars/menus when used
                   with the LINK element in the document head:
                b) to link to a separate style sheet
                c) to make a link to a script
                d) by stylesheets to control how collections of
                   html nodes are rendered into printed documents
                e) to make a link to a printable version of this document
                   e.g. a postscript or pdf version
-->

<!ELEMENT LINK - O EMPTY>
<!ATTLIST LINK
        href    %URL    #IMPLIED    -- URL for linked resource --
        rel     %Types  #IMPLIED    -- forward link types --
        rev     %Types  #IMPLIED    -- reverse link types --
        title   CDATA   #IMPLIED    -- advisory title string --
        >

<!—==================== Images ============================================—>

<!ENTITY % Length "CDATA"   -- nn for pixels or nn% for percentage length -->
<!ENTITY % Pixels "NUMBER"  -- integer representing length in pixels -->

<!-- Suggested widths are used for negotiating image size
     with the module responsible for painting the image.
     align=left or right cause image to float to margin
     and for subsequent text to wrap around image -->

<!ENTITY % IAlign "(top¦middle¦bottom¦left¦right)">

<!ELEMENT IMG    - O EMPTY --  Embedded image -->
<!ATTLIST IMG
        src     %URL     #REQUIRED  -- URL of image to embed --
        alt     CDATA    #IMPLIED   -- for display in place of image --
        align   %IAlign  #IMPLIED   -- vertical or horizontal alignment --
```

```
        height  %Pixels  #IMPLIED   -- suggested height in pixels --
        width   %Pixels  #IMPLIED   -- suggested width in pixels --
        border  %Pixels  #IMPLIED   -- suggested link border width --
        hspace  %Pixels  #IMPLIED   -- suggested horizontal gutter --
        vspace  %Pixels  #IMPLIED   -- suggested vertical gutter --
        usemap  %URL     #IMPLIED   -- use client-side image map --
        ismap   (ismap)  #IMPLIED   -- use server image map --
        >

<!-- USEMAP points to a MAP element which may be in this document
  or an external document, although the latter is not widely supported -->

<!--================== Java APPLET tag ===================================-->
<!--
  This tag is supported by all Java enabled browsers. Applet resources
  (including their classes) are normally loaded relative to the document
  URL (or <BASE> element if it is defined). The CODEBASE attribute is used
  to change this default behavior. If the CODEBASE attribute is defined then
  it specifies a different location to find applet resources. The value
  can be an absolute URL or a relative URL. The absolute URL is used as is
  without modification and is not effected by the documents <BASE> element.
  When the codebase attribute is relative, then it is relative to the
  document URL (or <BASE> tag if defined).
-->
<!ELEMENT APPLET - - (PARAM | %text)*>
<!ATTLIST APPLET
        codebase %URL     #IMPLIED   -- code base --
        code     CDATA    #REQUIRED  -- class file --
        alt      CDATA    #IMPLIED   -- for display in place of applet --
        name     CDATA    #IMPLIED   -- applet name --
        width    %Pixels  #REQUIRED  -- suggested width in pixels --
        height   %Pixels  #REQUIRED  -- suggested height in pixels --
        align    %IAlign  #IMPLIED   -- vertical or horizontal alignment --
        hspace   %Pixels  #IMPLIED   -- suggested horizontal gutter --
        vspace   %Pixels  #IMPLIED   -- suggested vertical gutter --
        >

<!ELEMENT PARAM - O EMPTY>
<!ATTLIST PARAM
        name    NMTOKEN   #REQUIRED  -- The name of the parameter --
        value   CDATA     #IMPLIED   -- The value of the parameter --
        >

<!--
Here is an example:

    <applet codebase="applets/NervousText"
        code=NervousText.class
        width=300
        height=50>
    <param name=text value="Java is Cool!">
    <img src=sorry.gif alt="This looks better with Java support">
    </applet>
-->
```



```
<!--================== Horizontal Rule ===================================-->

<!ELEMENT HR    - O EMPTY>
<!ATTLIST HR
        align (left|right|center) #IMPLIED
        noshade (noshade) #IMPLIED
        size  %Pixels #IMPLIED
        width %Length #IMPLIED
        >
<!--================== Paragraphs========================================-->

<!ELEMENT P     - O (%text)*>
<!ATTLIST P
        align  (left|center|right) #IMPLIED
        >

<!--================== Headings ==========================================-->

<!--
  There are six levels of headers from H1 (the most important)
  to H6 (the least important).
-->

<!ELEMENT ( %heading )  - -  (%text;)*>
<!ATTLIST ( %heading )
        align  (left|center|right) #IMPLIED
        >

<!--================== Preformatted Text =================================-->

<!-- excludes images and changes in font size -->

<!ENTITY % pre.exclusion "IMG|BIG|SMALL|SUB|SUP|FONT">

<!ELEMENT PRE - - (%text)* -(%pre.exclusion)>
<!ATTLIST PRE
        width NUMBER #implied -- is this widely supported? --
        >

<![ %HTML.Deprecated [

<!ENTITY % literal "CDATA"
        -- historical, non-conforming parsing mode where
           the only markup signal is the end tag
           in full
        -->

<!ELEMENT (XMP|LISTING) - -  %literal>
<!ELEMENT PLAINTEXT - O %literal>

]]>

<!--================== Block-like Quotes =================================-->

<!ELEMENT BLOCKQUOTE - - %body.content>
```

```
<!--================== Lists ============================================-->

<!--
    HTML 3.2 allows you to control the sequence number for ordered lists.
    You can set the sequence number with the START and VALUE attributes.
    The TYPE attribute may be used to specify the rendering of ordered
    and unordered lists.
-->

<!-- definition lists - DT for term, DD for its definition -->

<!ELEMENT DL    - -  (DT|DD)+>
<!ATTLIST DL
        compact (compact) #IMPLIED -- more compact style --
        >

<!ELEMENT DT - O  (%text)*>
<!ELEMENT DD - O  %flow;>

<!-- Ordered lists OL, and unordered lists UL -->
<!ELEMENT (OL|UL) - -  (LI)+>

<!--
      Numbering style
    1   arabic numbers     1, 2, 3, ...
    a   lower alpha        a, b, c, ...
    A   upper alpha        A, B, C, ...
    i   lower roman        i, ii, iii, ...
    I   upper roman        I, II, III, ...

    The style is applied to the sequence number which by default
    is reset to 1 for the first list item in an ordered list.

    This can't be expressed directly in SGML due to case folding.
-->

<!ENTITY % OLStyle "CDATA" -- constrained to: [1|a|A|i|I] -->

<!ATTLIST OL -- ordered lists --
        type      %OLStyle   #IMPLIED   -- numbering style --
        start     NUMBER     #IMPLIED   -- starting sequence number --
        compact  (compact)   #IMPLIED   -- reduced interitem spacing --
        >

<!-- bullet styles -->

<!ENTITY % ULStyle "disc|square|circle">

<!ATTLIST UL -- unordered lists --
        type    (%ULStyle)   #IMPLIED   -- bullet style --
        compact (compact)    #IMPLIED   -- reduced interitem spacing --
        >

<!ELEMENT (DIR|MENU) - -  (LI)+ -(%block)>
<!ATTLIST DIR
        compact (compact) #IMPLIED
        >
```



```
<!ATTLIST MENU
        compact (compact) #IMPLIED
        >

<!-- <DIR>              Directory list                -->
<!-- <DIR COMPACT>      Compact list style            -->
<!-- <MENU>             Menu list                     -->
<!-- <MENU COMPACT>     Compact list style            -->

<!-- The type attribute can be used to change the bullet style
     in unordered lists and the numbering style in ordered lists -->

<!ENTITY % LIStyle "CDATA" -- constrained to: "(%ULStyle|%OLStyle)" -->

<!ELEMENT LI - O %flow -- list item -->
<!ATTLIST LI
        type    %LIStyle     #IMPLIED   -- list item style --
        value    NUMBER      #IMPLIED   -- reset sequence number --
        >

<!--================ Forms ===============================================-->

<!ELEMENT FORM - - %body.content -(FORM)>
<!ATTLIST FORM
        action %URL #IMPLIED  -- server-side form handler --
        method (%HTTP-Method) GET -- see HTTP specification --
        enctype %Content-Type; "application/x-www-form-urlencoded"
        >

<!ENTITY % InputType
        "(TEXT | PASSWORD | CHECKBOX | RADIO | SUBMIT
            | RESET | FILE | HIDDEN | IMAGE)">

<!ELEMENT INPUT - O EMPTY>
<!ATTLIST INPUT
        type %InputType TEXT     -- what kind of widget is needed --
        name  CDATA   #IMPLIED   -- required for all but submit and reset --
        value CDATA   #IMPLIED   -- required for radio and checkboxes --
        checked (checked) #IMPLIED -- for radio buttons and check boxes --
        size CDATA    #IMPLIED   -- specific to each type of field --
        maxlength NUMBER #IMPLIED -- max chars allowed in text fields --
        src   %URL    #IMPLIED   -- for fields with background images --
        align %IAlign #IMPLIED   -- vertical or horizontal alignment --
        >

<!ELEMENT SELECT - - (OPTION+)>
<!ATTLIST SELECT
        name CDATA #REQUIRED
        size NUMBER #IMPLIED
        multiple (multiple) #IMPLIED
        >

<!ELEMENT OPTION - O (#PCDATA)*>
<!ATTLIST OPTION
        selected (selected) #IMPLIED
        value  CDATA  #IMPLIED -- defaults to element content --
        >
```

```
<!-- Multi-line text input field. -->

<!ELEMENT TEXTAREA - - (#PCDATA)*>
<!ATTLIST TEXTAREA
        name CDATA #REQUIRED
        rows NUMBER #REQUIRED
        cols NUMBER #REQUIRED
        >

<!--======================= Tables ===================================-->

<!-- Widely deployed subset of the full table standard, see RFC 1942
     e.g. at http://www.ics.uci.edu/pub/ietf/html/rfc1942.txt -->

<!-- horizontal placement of table relative to window -->
<!ENTITY % Where "(left|center|right)">

<!-- horizontal alignment attributes for cell contents -->
<!ENTITY % cell.halign
        "align  (left|center|right) #IMPLIED"
        >

<!-- vertical alignment attributes for cell contents -->
<!ENTITY % cell.valign
        "valign  (top|middle|bottom)  #IMPLIED"
        >

<!ELEMENT table - - (caption?, tr+)>
<!ELEMENT tr - O (th|td)*>
<!ELEMENT (th|td) - O %body.content>

<!ATTLIST table                         -- table element --
        align     %Where;   #IMPLIED    -- table position relative to window --
        width     %Length   #IMPLIED    -- table width relative to window --
        border    %Pixels   #IMPLIED    -- controls frame width around table --
        cellspacing %Pixels #IMPLIED    -- spacing between cells --
        cellpadding %Pixels #IMPLIED    -- spacing within cells --
        >

<!ELEMENT CAPTION - - (%text;)* -- table or figure caption -->
<!ATTLIST CAPTION
        align (top|bottom) #IMPLIED
        >

<!ATTLIST tr                            -- table row --
        %cell.halign;                   -- horizontal alignment in cells --
        %cell.valign;                   -- vertical alignment in cells --
        >

<!ATTLIST (th|td)                       -- header or data cell --
        nowrap (nowrap)  #IMPLIED       -- suppress word wrap --
        rowspan NUMBER   1              -- number of rows spanned by cell --
        colspan NUMBER   1              -- number of cols spanned by cell --
        %cell.halign;                   -- horizontal alignment in cell --
        %cell.valign;                   -- vertical alignment in cell --
        width   %Pixels  #IMPLIED       -- suggested width for cell --
        height  %Pixels  #IMPLIED       -- suggested height for cell --
        >
```



```
<!--================= Document Head ======================================-->

<!-- %head.misc defined earlier on as "SCRIPT|STYLE|META|LINK" -->

<!ENTITY % head.content "TITLE & ISINDEX? & BASE?">

<!ELEMENT HEAD O O  (%head.content) +(%head.misc)>

<!ELEMENT TITLE - -  (#PCDATA)* -(%head.misc)
          -- The TITLE element is not considered part of the flow of text.
             It should be displayed, for example as the page header or
             window title.
          -->

<!ELEMENT ISINDEX - O EMPTY>
<!ATTLIST ISINDEX
       prompt CDATA #IMPLIED -- prompt message -->

<!--
    The BASE element gives an absolute URL for dereferencing relative
    URLs, e.g.

         <BASE href="http://foo.com/index.html">
         ...
         <IMG SRC="images/bar.gif">

    The image is deferenced to

         http://foo.com/images/bar.gif

   In the absence of a BASE element the document URL should be used.
   Note that this is not necessarily the same as the URL used to
   request the document, as the base URL may be overridden by an HTTP
   header accompanying the document.
-->

<!ELEMENT BASE - O EMPTY>
<!ATTLIST BASE
        href %URL  #REQUIRED
        >

<!ELEMENT META - O EMPTY -- Generic Metainformation -->
<!ATTLIST META
        http-equiv  NAME    #IMPLIED  -- HTTP response header name  --
        name        NAME    #IMPLIED  -- metainformation name       --
        content     CDATA   #REQUIRED -- associated information     --
        >

<!-- SCRIPT/STYLE are place holders for transition to next version of HTML -->

<!ELEMENT STYLE  - - CDATA -- placeholder for style info -->
<!ELEMENT SCRIPT - - CDATA -- placeholder for script statements -->

<!-- ELEMENT STYLE  - - (#PCDATA)*  -(%head.misc) -- style info -->
<!-- ELEMENT SCRIPT - - (#PCDATA)*  -(%head.misc) -- script statements -->
```

```
<!--================ Document Structure ==================================-->

<!ENTITY % version.attr "VERSION CDATA #FIXED '%HTML.Version;'">

<![ %HTML.Deprecated [
    <!ENTITY % html.content "HEAD, BODY, PLAINTEXT?">
]]>

<!ENTITY % html.content "HEAD, BODY">

<!ELEMENT HTML O O  (%html.content)>
<!ATTLIST HTML
        %version.attr;
        >
```

Character Entities for ISO Latin-1

```
<!-- (C) International Organization for Standardization 1986
     Permission to copy in any form is granted for use with
     conforming SGML systems and applications as defined in
     ISO 8879, provided this notice is included in all copies.
     This has been extended for use with HTML to cover the full
     set of codes in the range 160-255 decimal.
-->
<!-- Character entity set. Typical invocation:
     <!ENTITY % ISOlat1 PUBLIC
       "ISO 8879-1986//ENTITIES Added Latin 1//EN//HTML">
     %ISOlat1;
-->
    <!ENTITY nbsp   CDATA " " -- no-break space -->
    <!ENTITY iexcl  CDATA "&#161;" -- inverted exclamation mark -->
    <!ENTITY cent   CDATA "&#162;" -- cent sign -->
    <!ENTITY pound  CDATA "&#163;" -- pound sterling sign -->
    <!ENTITY curren CDATA "&#164;" -- general currency sign -->
    <!ENTITY yen    CDATA "&#165;" -- yen sign -->
    <!ENTITY brvbar CDATA "&#166;" -- broken (vertical) bar -->
    <!ENTITY sect   CDATA "&#167;" -- section sign -->
    <!ENTITY uml    CDATA "&#168;" -- umlaut (dieresis) -->
    <!ENTITY copy   CDATA "&#169;" -- copyright sign -->
    <!ENTITY ordf   CDATA "&#170;" -- ordinal indicator, feminine -->
    <!ENTITY laquo  CDATA "&#171;" -- angle quotation mark, left -->
    <!ENTITY not    CDATA "&#172;" -- not sign -->
    <!ENTITY shy    CDATA "&#173;" -- soft hyphen -->
    <!ENTITY reg    CDATA "&#174;" -- registered sign -->
    <!ENTITY macr   CDATA "&#175;" -- macron -->
    <!ENTITY deg    CDATA "&#176;" -- degree sign -->
    <!ENTITY plusmn CDATA "&#177;" -- plus-or-minus sign -->
    <!ENTITY sup2   CDATA "&#178;" -- superscript two -->
    <!ENTITY sup3   CDATA "&#179;" -- superscript three -->
    <!ENTITY acute  CDATA "&#180;" -- acute accent -->
    <!ENTITY micro  CDATA "&#181;" -- micro sign -->
    <!ENTITY para   CDATA "&#182;" -- pilcrow (paragraph sign) -->
    <!ENTITY middot CDATA "&#183;" -- middle dot -->
    <!ENTITY ccedil CDATA "&#184;" -- small c, cedilla -->
    <!ENTITY sup1   CDATA "&#185;" -- superscript one -->
    <!ENTITY ordm   CDATA "&#186;" -- ordinal indicator, masculine -->
    <!ENTITY raquo  CDATA "&#187;" -- angle quotation mark, right -->
    <!ENTITY frac14 CDATA "&#188;" -- fraction one-quarter -->
    <!ENTITY frac12 CDATA "&#189;" -- fraction one-half -->
```



```
<!ENTITY frac34 CDATA "&#190;" -- fraction three-quarters -->
<!ENTITY iquest CDATA "&#191;" -- inverted question mark -->
<!ENTITY Agrave CDATA "&#192;" -- capital A, grave accent -->
<!ENTITY Aacute CDATA "&#193;" -- capital A, acute accent -->
<!ENTITY Acirc  CDATA "&#194;" -- capital A, circumflex accent -->
<!ENTITY Atilde CDATA "&#195;" -- capital A, tilde -->
<!ENTITY Auml   CDATA "&#196;" -- capital A, dieresis or umlaut mark -->
<!ENTITY Aring  CDATA "&#197;" -- capital A, ring -->
<!ENTITY AElig  CDATA "&#198;" -- capital AE diphthong (ligature) -->
<!ENTITY Ccedil CDATA "&#199;" -- capital C, cedilla -->
<!ENTITY Egrave CDATA "&#200;" -- capital E, grave accent -->
<!ENTITY Eacute CDATA "&#201;" -- capital E, acute accent -->
<!ENTITY Ecirc  CDATA "&#202;" -- capital E, circumflex accent -->
<!ENTITY Euml   CDATA "&#203;" -- capital E, dieresis or umlaut mark -->
<!ENTITY Igrave CDATA "&#204;" -- capital I, grave accent -->
<!ENTITY Iacute CDATA "&#205;" -- capital I, acute accent -->
<!ENTITY Icirc  CDATA "&#206;" -- capital I, circumflex accent -->
<!ENTITY Iuml   CDATA "&#207;" -- capital I, dieresis or umlaut mark -->
<!ENTITY ETH    CDATA "&#208;" -- capital Eth, Icelandic -->
<!ENTITY Ntilde CDATA "&#209;" -- capital N, tilde -->
<!ENTITY Ograve CDATA "&#210;" -- capital O, grave accent -->
<!ENTITY Oacute CDATA "&#211;" -- capital O, acute accent -->
<!ENTITY Ocirc  CDATA "&#212;" -- capital O, circumflex accent -->
<!ENTITY Otilde CDATA "&#213;" -- capital O, tilde -->
<!ENTITY Ouml   CDATA "&#214;" -- capital O, dieresis or umlaut mark -->
<!ENTITY times  CDATA "&#215;" -- multiply sign -->
<!ENTITY Oslash CDATA "&#216;" -- capital O, slash -->
<!ENTITY Ugrave CDATA "&#217;" -- capital U, grave accent -->
<!ENTITY Uacute CDATA "&#218;" -- capital U, acute accent -->
<!ENTITY Ucirc  CDATA "&#219;" -- capital U, circumflex accent -->
<!ENTITY Uuml   CDATA "&#220;" -- capital U, dieresis or umlaut mark -->
<!ENTITY Yacute CDATA "&#221;" -- capital Y, acute accent -->
<!ENTITY THORN  CDATA "&#222;" -- capital THORN, Icelandic -->
<!ENTITY szlig  CDATA "&#223;" -- small sharp s, German (sz ligature) -->
<!ENTITY agrave CDATA "&#224;" -- small a, grave accent -->
<!ENTITY aacute CDATA "&#225;" -- small a, acute accent -->
<!ENTITY acirc  CDATA "&#226;" -- small a, circumflex accent -->
<!ENTITY atilde CDATA "&#227;" -- small a, tilde -->
<!ENTITY auml   CDATA "&#228;" -- small a, dieresis or umlaut mark -->
<!ENTITY aring  CDATA "&#229;" -- small a, ring -->
<!ENTITY aelig  CDATA "&#230;" -- small ae diphthong (ligature) -->
<!ENTITY ccedil CDATA "&#231;" -- small c, cedilla -->
<!ENTITY egrave CDATA "&#232;" -- small e, grave accent -->
<!ENTITY eacute CDATA "&#233;" -- small e, acute accent -->
<!ENTITY ecirc  CDATA "&#234;" -- small e, circumflex accent -->
<!ENTITY euml   CDATA "&#235;" -- small e, dieresis or umlaut mark -->
<!ENTITY igrave CDATA "&#236;" -- small i, grave accent -->
<!ENTITY iacute CDATA "&#237;" -- small i, acute accent -->
<!ENTITY icirc  CDATA "&#238;" -- small i, circumflex accent -->
<!ENTITY iuml   CDATA "&#239;" -- small i, dieresis or umlaut mark -->
<!ENTITY eth    CDATA "&#240;" -- small eth, Icelandic -->
<!ENTITY ntilde CDATA "&#241;" -- small n, tilde -->
<!ENTITY ograve CDATA "&#242;" -- small o, grave accent -->
<!ENTITY oacute CDATA "&#243;" -- small o, acute accent -->
<!ENTITY ocirc  CDATA "&#244;" -- small o, circumflex accent -->
<!ENTITY otilde CDATA "&#245;" -- small o, tilde -->
<!ENTITY ouml   CDATA "&#246;" -- small o, dieresis or umlaut mark -->
<!ENTITY divide CDATA "&#247;" -- divide sign -->
<!ENTITY oslash CDATA "&#248;" -- small o, slash -->
<!ENTITY ugrave CDATA "&#249;" -- small u, grave accent -->
```

```
<!ENTITY uacute CDATA "&#250;" -- small u, acute accent -->
<!ENTITY ucirc  CDATA "&#251;" -- small u, circumflex accent -->
<!ENTITY uuml   CDATA "&#252;" -- small u, dieresis or umlaut mark -->
<!ENTITY yacute CDATA "&#253;" -- small y, acute accent -->
<!ENTITY thorn  CDATA "&#254;" -- small thorn, Icelandic -->
<!ENTITY yuml   CDATA "&#255;" -- small y, dieresis or umlaut mark -->
```

Table of Printable Latin-1 Character Codes

0	32		64	@	96	‘	128		160		192	À	224	à
1	33	!	65	A	97	a	129		161	¡	193	Á	225	á
2	34	"	66	B	98	b	130		162	¢	194	Â	226	â
3	35	#	67	C	99	c	131		163	£	195	Ã	227	ã
4	36	$	68	D	100	d	132		164	¤	196	Ä	228	ä
5	37	%	69	E	101	e	133		165	¥	197	Å	229	å
6	38	&	70	F	102	f	134		166	¦	198	Æ	230	æ
7	39	’	71	G	103	g	135		167	§	199	Ç	231	ç
8	40	(	72	H	104	h	136		168	¨	200	È	232	è
9	41	)	73	I	105	i	137		169	©	201	É	233	é
10	42	*	74	J	106	j	138		170	ª	202	Ê	234	ê
11	43	+	75	K	107	k	139		171	«	203	Ë	235	ë
12	44	,	76	L	108	l	140		172	¬	204	Ì	236	ì
13	45	-	77	M	109	m	141		173	-	205	Í	237	í
14	46	.	78	N	110	n	142		174	®	206	Î	238	î
15	47	/	79	O	111	o	143		175	¯	207	Ï	239	ï
16	48	0	80	P	112	p	144		176	°	208	Ð	240	ð
17	49	1	81	Q	113	q	145		177	±	209	Ñ	241	ñ
18	50	2	82	R	114	r	146		178	²	210	Ò	242	ò
19	51	3	83	S	115	s	147		179	³	211	Ó	243	ó
20	52	4	84	T	116	t	148		180	´	212	Ô	244	ô
21	53	5	85	U	117	u	149		181	µ	213	Õ	245	õ
22	54	6	86	V	118	v	150		182	¶	214	Ö	246	ö
23	55	7	87	W	119	w	151		183	·	215	×	247	÷
24	56	8	88	X	120	x	152		184	¸	216	Ø	248	ø
25	57	9	89	Y	121	y	153		185	¹	217	Ù	249	ù
26	58	:	90	Z	122	z	154		186	º	218	Ú	250	ú
27	59	;	91	[	123	{	155		187	»	219	Û	251	û
28	60	<	92	\	124	|	156		188	¼	220	Ü	252	ü
29	61	=	93	]	125	}	157		189	½	221	Ý	253	ý
30	62	>	94	^	126	~	158		190	¾	222	Þ	254	þ
31	63	?	95	_	127		159		191	¿	223	ß	255	ÿ

Acknowledgments

The author would like to thank the members of the W3C HTML Editorial Review Board, members of the W3C staff, and the many other people who have contributed to this specification.

Further Reading

The World Wide Web Consortium

Further information on W3C activities and pointers to the status of work on HTML and HTTP etc. can be found at `http://www.w3.org/`. Further information on HTML in particular can be found at `http://www.w3.org/pub/WWW/MarkUp/`.

HTML 2.0 (RFC1866)

By Tim Berners-Lee and Dan Connolly, November 1995. Defines the Hypertext Markup Language Specification Version 2.0. Available from `ftp://ds.internic.net/rfc/rfc1866.txt`.

Form-Based File Upload in HTML (RFC1867)

By E. Nebel and L. Masinter, November 1995. Describes extensions to HTML 2.0 (RFC1866) to support file upload from HTML forms. Available from `ftp://ds.internic.net/rfc/rfc1867.txt`.

HTML Tables (RFC1942)

By Dave Raggett, May 1996. This defines the HTML table model. It is a superset of the table model defined by HTML 3.2. Available from `ftp://ds.internic.net/rfc/rfc1942.txt`, or as a W3C working draft at `http://www.w3.org/pub/WWW/TR/WD-tables`.

A Lexical Analyzer for HTML and Basic SGML

By Dan Connolly, June 1996. Describes lexical considerations for parsing HTML documents. Available from `http://www.w3.org/pub/WWW/TR/WD-html-lex`.

The Hypertext Transfer Protocol (HTTP)

Further information of HTTP can be found at `http://www.w3.org/pub/WWW/Protocols`.

A Standard Default Color Space for the Internet—sRGB

By Michael Stokes, Mathew Anderson, Srinivasan Chandrasekar, and Ricardo Motta, November 1996. Available from `http://www.w3.org/pub/WWW/Graphics/Color/sRGB.html`. This provides a precise definition for RGB that allows sRGB images to be reproduced accurately on different platforms and media under varying ambient lighting conditions.

APPENDIX C

CSS1 Quick Reference

by Rick Darnell

IN THIS APPENDIX

This appendix provides an overview of the attributes with which you can control the appearance of your HTML documents through style sheets. The World Wide Web Consortium (W3C) set the current standard for style sheets as Cascading Style Sheets 1 (CSS1). W3C's complete recommendation for CSS is located at the W3C Web site at `http://www.w3.org/pub/WWW/TR/REC-CSS1`.

For more information on using style sheets, see Chapters 19 and 20, "Introducing Cascading Style Sheets" and "Cascading Style Sheet Usage."

Basic Syntax

All styles within a style sheet definition follow the same basic syntax. You'll notice that there are a lot of opportunities to add other attributes or members of a group:

```
SELECTOR[.class] [,SELECTOR2[.class2]] ...
{ attribute1: value1  [;
  attribute2: value2] [;
  ... ]             [;
  attributen: valuen] }
```

The `SELECTOR` is how the style is referenced within the rest of the HTML page. It uses one of the existing HTML tags, such as `<CODE>` or `<P>`, along with an optional `class` to create additional sub-styles. A `class` is a subset of a selector, allowing the same element to have a variety of styles. For example, you could color code block quotes to identify sources or speakers.

In addition to the standard HTML tags, you can use two other values for a selector: `first-line` and `first-letter`. The `first-line` value sets the style for the first line of text in a document or several passages within a document, such as paragraph or block quote. The `first-letter` value creates drop caps and other special effects on the first letter in a document or passage.

Groups of selectors and their classes are separated by commas. A member of the group receives the same style as any other member in the group. For example, if you wanted all headings to be displayed in red, you could list H1 through H6 with the attributes to set the color to red. All other tag attributes, such as size, would remain unaffected.

Another option is contextual selectors, which tell the browser what to do with a certain tag when found nested within the parent tag.

```
OUTER_SELECTOR INNER_SELECTOR {attribute:value}
```

This means that when the `INNER_SELECTOR` is used within the `OUTER_SELECTOR`, the style is used. Otherwise, other occurrences of `INNER_SELECTOR` are handled according to browser default.

After making all of the selector and group definitions, use a curly bracket along with a series of attributes and their values. Mate each attribute with its value with a colon and separate each pair from the next pair by a semicolon. The values within a definition, such as the name of a typeface or a color value, are not case-sensitive. For example, for `font-family`, you can have `Garamond`, `garamond`, or `GARAMOND`, and it will all work out the same in the browser.

As with all good syntax, you can place style definitions in three ways within a document: with an embedded style sheet, with a linked style sheet, and with an inline style sheet.

Embedded Style Sheet

The `<STYLE>` tags contain an embedded style sheet. As a matter of structure, the format of an HTML page with an embedded style sheet is as follows:

```
<HTML>
<HEAD>...</HEAD>
<STYLE>...</STYLE>
<BODY>...</BODY>
</HTML>
```

The `<STYLE>` tags contain the list of selectors and styles.

Linked Style Sheet

The linked style sheet is a `.css` file that contains nothing but a set of `<STYLE>` tags and their contents. Identify the style file within an HTML document using the `<LINK>` tag in the head:

```
<HEAD>
<LINK rel=stylesheet href="filename.css" type="text/css">
</HEAD>
```

At runtime, the browser will load the style in the `.css` file and use it to format the document. If the HTML page also includes an embedded style sheet that conflicts with the linked style sheet, the embedded version also takes precedence.

Inline Style Sheet

The last option, inline style sheets, uses style sheet syntax, although it's technically not a style sheet implementation. This option uses the style sheet nomenclature to customize single incidents of tags within the document:

```
<TAG style="attribute1:value1; ...">
```

Essentially, this is a way to customize HTML tags on a case-by-case basis. When you use all three forms of syntax, they occur in a cascading form of precedence. The highest priority is inline, followed by embedded, then linked.

Style Attributes

Several classes of attributes are used within the definition for a selector. The following sections cover each of the attributes within a class.

Fonts

There are no current standards for typefaces and their use on different user machines, so you'll need to choose carefully and include several options to achieve the desired effect for the user.

The font-family Attribute

The font-family attribute lists font families in order of preference, separated by commas. Two types of variables are used: family name and generic family.

```
BODY {font-family: Garamond, Palatino, Serif}
```

A family name is the name of a specific typeface such as Helvetica, Garamond, Palatino, or Optima. Enclose font names with spaces in quotes, such as "Gil Sans". The generic family is one of five choices that classifies the typeface by its style and is recommended as the last option in a font-family list:

- Serif: Fonts with accents at the tips of the lines (for example, Times)
- Sans-serif: Fonts without finishing accents (for example, Helvetica)
- Cursive: Scripts that more closely resemble hand-drawn calligraphy (for example, Zapf Chancery)
- Fancy: Special-use decorative fonts (for example, Comic Book Sans)
- Monospace: Fonts that maintain uniform spacing despite letter width (for example, Courier)

The font-style Attribute

This attribute specifies the type of treatment a font receives and is represented by the values normal, italic, or oblique. The normal value is also referred to as Roman in some typeface references. The oblique value is similar to italic except that it is usually slanted manually by the system rather than by a separate style of the font, like italic.

```
BODY {font-style: italic}
```

The font-variant Attribute

Similar to font-style, this attribute sets small caps. Its two values are normal and small-caps.

```
BODY {font-variant: small-caps}
```

If there is no true small caps version of the typeface, the system will attempt to scale the capital letters to a smaller size for lowercase letters. As a last resort, the text will appear in all capitals.

The font-weight Attribute

A number of values for this attribute set the darkness or lightness of a typeface. The primary values are normal and bold. You can substitute these values with one of a list of values from 100 to 900. If a typeface includes a "medium" weight, it will correspond to 500. Bold is represented by 700.

```
BODY {font-weight: bold}
```

Two additional values are bolder and lighter, which increase the weight from the current parent weight by one level, such as 200 to 300 for bolder or 700 to 600 for lighter.

The font-size Attribute

Four methods can define the size of a font in a style—absolute size, relative size, length, or percentage.

- Absolute size: This method is represented in several ways. The first is with a value that represents its size relative to other sizes within the family (xx-small, x-small, small, medium, large, x-large, xx-large). You can also use a numerical value, such as 12pt (12 points).

  ```
  BODY {font-size: 18pt}
  ```

- Relative size: This method sets the size relative to the parent style. It can be one of two values, smaller or larger, and it adjusts the size up or down the scale of sizes. If a font doesn't include a mapping to size names, a scaling of 1.5 is recommended between sizes. For example, a 10pt font would be scaled larger to 15pt or smaller to 7pt.

  ```
  P {font-size: smaller}
  ```

- Length: This method is another form of relative size that sets the size by the scale factor of the width of an em, such as 1.5em.

  ```
  P {font-size: 2em}
  ```

- Percentage: This method is also a relative specification that multiplies the size of the parent font by the percentage value to achieve the new size, such as 150%.

  ```
  H3 {font-size: 300%}
  ```

The font Attribute

This attribute provides a shorthand for setting all of the previous attributes under one umbrella. The order of the attributes should be font-style, font-variant, font-weight, font-size, line-height, font-family. Place no commas between each of the attribute values, except for listed font families:

```
BODY {font: small-caps bold 14pt garamond, palatino, serif}
```

Color and Background

These elements set the color values for the text (foreground) and the area behind the text (background). In addition to setting a background color, you can also define a background image. All color values are defined using the same methods as the color attribute.

The color Attribute

This attribute defines the color of the text element and is specified using one of the color keywords (such as red). You can also define the color using a hexadecimal triplet, denoting the mix of red, green, and blue (such as rgb(255,0,0)).

```
BLOCKQUOTE {color: rgb(0,255,255)}
```

The background-color Attribute

This attribute sets the background color for a style. You can set this attribute independently of a background color for the document to enable you to highlight text in a different manner.

```
BLOCKQUOTE {background-color: blue}
```

The background-image Attribute

This attribute specifies a background image for a style element. Use it in conjunction with background-color to ensure a substitute effect if the image becomes unavailable. If the image is available, it will display on top of the background color.

```
BLOCKQUOTE {background-image: url(logo.gif)}
```

The background-repeat Attribute

If the background image should be repeated (tiled), use this attribute to define how. Its values include repeat, repeat-x, and repeat-y. The repeat value indicates that the image should be tiled normally. The repeat-x value repeats the image in a single horizontal line, and the repeat-y value repeats the image in a vertical line.

```
BLOCKQUOTE {background-image: url(logo.gif);
            background-repeat: repeat-x}
```

The background-attachment Attribute

This attribute, an extended feature of background images not seen in HTML before, sets whether the background image is attached to the foreground text (scroll) or anchored to the background (fixed). This feature is apparent only when the user scrolls across a selection of text.

```
BLOCKQUOTE {background-image: url(logo.gif);
            background-attachment: repeat-x}
```

The background-position Attribute

When you use a background image through normal HTML, the starting point is always the top left of the screen. With a style sheet, you can specify a starting point anywhere within the box that contains the style content.

You can specify the image's starting position in three ways. The first way is with key word locations. For horizontal placement, your choices are left, center, or right. For vertical placement, your choices are top, center, or bottom. Alternatively, you can represent the position as a percentage of the available area, with 0% 0% being the top left (default) and 100% 100% being the bottom right. The last option is to specify an actual measurement in centimeters or inches.

If only one value for the placement is given, it's used as the horizontal position. If both values are given, the first is evaluated as horizontal and the second as vertical.

```
BLOCKQUOTE {background-image: url(logo.gif);
            Background-repeat: repeat-y;
            background-position: right top; }
```


The background Attribute

This shorthand attribute, similar to font, enables you to define a set of values for the background in one stop. The order is background-color, background-image, background-repeat, background-attachment, and background-position.

```
P { background: black url(logo.gif) repeat-y fixed right top }
```

Text

This set of style attributes covers the values that can affect the appearance of text, but not by directly changing the typeface. This includes values for spacing, underlining, blinking, and strike-through. It also supports some of the positioning attributes, including left and right justification and indents.

The word-spacing Attribute

This attribute indicates an addition to the default amount of space between individual words and is specified in *ems*. An *em* is the space occupied by the letter "m" and is the baseline for determining widths within a font. To return the value to its default, use 0em or normal.

```
BODY { word-spacing: 1em }
```

The letter-spacing Attribute

The letter-spacing attribute is similar to word-spacing, except that letter-spacing adds an extra bit of spacing between individual letters. In addition to the default method the browser uses to determine spacing, additional letter spacing is also affected by text alignment.

```
BODY { letter-spacing: 0.2em }
```

The text-decoration Attribute

This attribute is more closely related to its cousins in the font family. It specifies extra text flourishes, such as underline, strike-through, and blinking. The four values are none, underline, overline, line-through, and blink.

```
STR.blink { text-decoration: underline blink }
```

The vertical-align Attribute

This attribute sets the vertical position of the text either to an absolute reference or in relation to the parent element. It supports a range of values and keywords:

- Baseline: Aligns the baseline of the style with the baseline of the parent element
- Sub: Assigns the style to a subscript relative to the parent element
- Super: Assigns the style to a superscript relative to the parent element
- Text-top: Aligns the top of the text with the top of the parent's text
- Text-bottom: Aligns the bottom of the text with the bottom of the parent's text
- Middle: Aligns the vertical halfway point of the element with the baseline of the parent plus half of the x-height of the parent (x-height is the height of the lowercase x of the font)
- Top: Aligns the top of the element with the tallest element on the current line
- Bottom: Aligns the bottom of the element with the lowest element on the current line
- (Percentage): Using a positive or negative percentage value, raises or lowers the element beyond the baseline of the parent

```
SUB { vertical-align: -10% }
```

The text-transform Attribute

This attribute sets the capitalization of the affected text to one of four choices: capitalize (first letter of every word), uppercase (all letters in capitals), lowercase (all letters in lowercase), and none.

```
STR.caps { text-transform: uppercase }
```

The text-align Attribute

This attribute moves beyond the standard HTML left-right-center alignment to provide full justification (justify left and right). If a browser doesn't support justify, it will typically substitute left.

```
BLOCKQUOTE { text-align: justify }
```

The text-indent Attribute

The text-indent attribute, specified in an absolute value measured in ems or inches, defines the amount of space that is added before the first line.

```
P { text-indent: 5em }
```


The line-height Attribute

This attribute sets the distance between adjacent baselines using a length (in ems), multiplication factor, or percentage. Factors are indicated without any units, such as 1.5. When you use this method, the child inherits the factor, not the resulting value.

```
DIV { line-height: 1.5; font-size: 12pt }
```

In this instance, the line height becomes 18 points and the font size remains at 12 points.

Margins, Padding, and Borders

Each element created in a style sheet is presented in its own "box." All of the styles from the element inside the box are applied, although the box itself can have its own properties that define how it relates to adjoining elements on the page. Length is specified in inches (in), centimeters (cm), ems (em), points (pt), or pixels (px).

Box properties are divided into three basic categories. Margin properties set the border around the outside of the box, padding properties determine how much space to insert between the border and the content, and border properties define graphical lines around an element.

Additional properties of the box include its width, height, and physical position.

The margin-top, margin-bottom, margin-right, and margin-left Attributes

These four attributes set the amount of space between the element and adjoining elements, whether defined by length or percentage of parent text width or handled automatically.

```
BLOCKQUOTE { margin-top: 4em;
             Margin-bottom: auto }
```

The margin Attribute

The margin attribute provides a shorthand method for setting the four margin values.

When you specify the four values, they are applied, in order, to the top, right, bottom, and left. If you provide only one value, it applies to all sides. If you use two or three values, the missing values are copied from the opposite sides.

```
BLOCKQUOTE {margin: 4em 2em}
```

The padding-top, padding-bottom, padding-right, and padding-left Attributes

These attributes set the distance between the boundaries of the box and the elements inside the box. It can use any of the physical measurements or a percentage of the parent's width.

```
BLOCKQUOTE {padding-top: 110%; padding-bottom: 115%}
```

The padding Attribute

The padding attribute provides a shorthand method for setting the four padding values.

When you specify the four values, they are applied, in order, to the top, right, bottom, and left. If you provide only one value, it applies to all sides. If you use two or three values, the missing values are copied from the opposite sides.

```
BLOCKQUOTE {padding: 10pt 12pt}
```

The border-top, border-bottom, border-right, and border-left Attributes

These four attributes set the style and color of each border around an element. Specify styles with one of the border style keywords: none, dotted, dashed, solid, double, groove, ridge, inset, and outset. For more information on these, see the information on border-style later in this chapter.

Specify colors using a color keyword. For more information, see the border-color later in this chapter.

```
BLOCKQUOTE {border-left: solid red}
```

The border-top-width, border-bottom-width, border-right-width, and border-left-width Attributes

These attributes define a physical border around the box, similar to the border used for HTML tables. In addition to defining a specific width in ems, you can also use the keywords thin, medium, and thick. Using a measurement in ems results in a border whose width changes in relation to the size of the current font.

```
STR {border-right-width: 2pt;
     border-left-width: 2pt }
```

The border-width Attribute

The border-width attribute provides a shorthand method for setting the width of the four borders.

When you specify the four values, they are applied, in order, to the top, right, bottom, and left. If you provide only one value, it applies to all sides. If you use two or three values, the missing values are copied from the opposite sides.

```
BLOCKQUOTE {border-width: medium 0pt 0pt thick}
```

The border-color Attribute

This attribute sets the color of all four borders and uses one color keyword as its value. You cannot set the color of each side independently.

```
BLOCKQUOTE {border-color: yellow}
```


The border-style Attribute

The border's appearance can take on several different settings, represented by none, dotted, dashed, solid, double, groove, ridge, inset, and outset. The last four values are represented in 3D, if the browser supports it. Alternatively, the browser also can present all of the variations as a solid line, except none.

Like border-color, the style is applied uniformly to all four sides.

```
BLOCKQUOTE {border-style: groove}
```

The border Attribute

The border attribute provides a shorthand method for setting all of the border variables, including width, style, and color. It sets the values for all four sides at the same time, overriding any individual settings that may have been set previously for the same element.

```
BLOCKQUOTE {border: 1.5pt double black}
```

The height Attribute

This attribute sets the overall height of the bounding box that contains either the text or image element. If the content is text, scrollbars are added as needed so that all of the material is still available to the user. If the content is an image, it's scaled to fit inside the area. You can set a physical value or use auto to let the browser allocate space as needed.

```
BLOCKQUOTE {height: 100px}
```

The width Attribute

Similar to height, the width attribute sets the overall width of the bounding box that contains the element. If the content is text, scrollbars are added as needed so that all of the material is still available for the user. If both elements are used with an image and the value of one element is auto, the aspect ratio for the image is maintained.

```
BLOCKQUOTE {width: auto}
```

The float Attribute

This attribute sets a value similar to the align attribute used in HTML. The three possible values are left, right, and none. The none value allows the element to fall where it may, and the other two values force the element to the left or right of the screen with text wrapping around the opposite side.

```
BLOCKQUOTE {float: right}
```

The clear Attribute

This attribute mimics the clear attribute used with the HTML <BR> tag and uses the same keywords as float. If you use it with right or left, elements will move below any floating

element on that respective side. If you set it to none, floating elements are allowed on both sides.

```
BLOCKQUOTE {clear: left right}
```

Classification

These attributes control the general behavior of other elements more than actually specifying an appearance. In addition, classification includes the attributes for list items, identified in HTML with the <LI> tag.

The display Attribute

This attribute identifies when and if a style element should be used. Four keywords determine its behavior:

- Inline: A new box is created within the same line as adjoining text items and is formatted according to the size and amount of content within its borders, such as an image (IMG) or text (STR).
- Block: A new box is created relative to the surrounding elements. This is common with elements such as H1 and P.
- List-item: Similar to block, only list item markers, which behave more like inline content, are added.
- None: Turns off the display of the element in any situation, including for children of the element.

```
IMG {display: inline}
BLOCKQUOTE {display: block}
```

The white-space Attribute

The name of this attribute is a bit misleading because it relates to how spaces and line breaks are handled. The choices are normal (in which extra spaces are ignored), pre (as in preformatted HTML text), and nowrap (in which lines are broken only with <BR>).

```
BLOCKQUOTE {white-space: pre}
```

The list-style-type Attribute

This element sets the type of markers used for a list. Your choices are disc, circle, square, decimal, lower-roman, upper-roman, lower-alpha, upper-alpha, and none. For more information on how each of these is represented onscreen, see Chapter 9, "Using Lists to Organize Information."

```
LI.outline1 {list-style-type: upper-roman}
LI.outline2 {list-style-type: upper-alpha}
LI.outline3 {list-style-type: decimal}
```


The list-style-image Attribute

In lieu of a text marker for the list item, you can also specify the URL of an image to use. If the image is unavailable, the text marker is used as default.

```
LI.general {list-style-image: url(bullet.jpg)}
```

The list-style-position Attribute

The two values for this attribute, inside and outside, determine the formatting of text following the list item marker. The outside value, the default value, lines up the additional lines of text beyond the first line with the first character in the first line. If you use the inside value, the second and following lines are justified with the list item marker.

```
LI {list-style-position: inside}
```

The list-style Attribute

This attribute is a shorthand element for the list-style-type, list-style-image, and list-style-position attributes.

```
OL {list-style: lower-alpha outside}
UL {list-style: square url(bullet.jpg) inside}
```

APPENDIX D

HTML Resources

by David Mayhew

IN THIS APPENDIX

Because the Web is constantly changing, no book can be 100 percent up-to-date, even on the day it arrives in the bookstores. Thankfully, hundreds of resources available on the Web itself are frequently revised to keep their information current.

In this appendix, you'll find a summary of many of those resources, including their URLs and a brief description. In addition to Web pages about changes in HTML, the latest browser developments, and HTML development tools, at the end of this appendix is a guide to relevant newsgroups where hundreds of messages are posted each day.

Information on HTML

This section gives you the details on several Web sites where you can find current HTML information.

Netscape DevEdge OnLine

`http://developer.netscape.com/index.html`

If you use Netscape technology to develop your Internet site or intranet, you need to visit the DevEdge site frequently. DevEdge's mission is to provide "communications, tools, support, and marketing assistance to speed the planning, development, and deployment of Internet and Intranet solutions." The site is a combination of free and fee-based content. The free access is primarily through a twice-weekly news e-mail with the latest developments in Netscape technology, news of new betas, bug fixes, and so on. Premium services include early access to betas, a quarterly CD full of Netscape tools, a Netscape ONE directory link, and peer newsgroups.

Microsoft Site Builder Network

`http://www.microsoft.com/sitebuilder/`

Here you will find a very deep resource for developing Web sites with Microsoft technology. The Site Builder Network is a multilevel program that delivers technical information, products, technologies, services, and support for the latest Internet technologies, such as new HTML extensions, ActiveX controls, and Java applets. Information is scaled to match your level of membership, but even the content available free of charge makes this site worthy of frequent visits. Of particular interest is the Workshop area, which contains hands-on information on authoring, design, programming, and server technologies.

The World Wide Web Consortium (W3C)

`http://www.w3.org/pub/WWW/`

The W3C was founded in 1994 to develop common protocols for the evolution of the World Wide Web. The Consortium is led by Tim Berners-Lee, Director of the W3C and creator of the World Wide Web; and Jean-François Abramatic, Chairman of the W3C; and is funded by commercial members. The vendor-neutral W3C works with the global community to produce specifications and reference software that is made freely available throughout the world.

Cougar

`http://www.w3.org/pub/WWW/MarkUp/Cougar/`

Cougar is W3C's next version of HTML.

HTML 3.2 Reference Specification

`http://www.w3.org/pub/WWW/TR/REC-html32.html`

This site offers the final specification for HTML 3.2, written by Dave Raggett.

HTML 2.0 Specification

`http://www.w3.org/pub/WWW/MarkUp/html-spec/html-spec_toc.html`

Tim Berners-Lee and Dan Connolly wrote the HTML 2.0 specification, located at this site.

Cascading Style Sheets, Level 1, W3C Recommendation

`http://www.w3.org/pub/WWW/TR/REC-CSS1`

This document specifies level 1 of the Cascading Style Sheet mechanism (CSS1).

Document Object Model (Dynamic HTML)

`http://www.w3.org/pub/WWW/MarkUp/DOM/`

This site offers an overview of materials related to the Document Object Model at W3C and around the Web. As mentioned earlier in this book, the Document Object Model (DOM) is a platform-independent and language-neutral interface that allows programs and scripts to dynamically access and update the structure, content, and style of Web pages.

URL Descriptions

`http://www.w3.org/hypertext/WWW/Addressing/Addressing.html`

This site presents definitions of the various types of URLs as well as discussion of URIs and URNs.

Yahoo!

`http://www.yahoo.com/Computers/World_Wide_Web`

This site exhibits links to more than 3,000 pages regarding HTML. Find links here to sites about Internet technologies such as HTML, CGI, Java, ActiveX, VBScript, VRML, and many others, as well as resources for design, programming, browser information, and server details, to name just a few.

HTML Writers Guild

`http://hwg.org`

The HTML Writers Guild is an international organization of World Wide Web page authors and Internet publishing professionals. The more than 50,000 members enjoy a wealth of HTML resources, plus access to their newsgroups. Basic membership is free and includes access to more information than many sites give you after charging a fee. Several paid membership levels are available as well, and these offer benefits such as mentoring programs, educational classes, access to the job board, and software discounts.

The Web Design Group

`http://www.htmlhelp.com`

This is a highly useful site for beginners and experienced developers alike. Here you will find the widely acclaimed HTML 3.2 Reference, a Cascading Style Sheets Guide, and a Character Set Overview.

The HTML Guru

`http://members.aol.com/htmlguru/index.html`

This site offers a collection of HTML tips, tricks, and hacks covering both document authoring and Web server management, presented in a question and answer format.

Browsers

The HTML that you create doesn't always look the way that you intended on every user's screen. The different ways that browsers interpret your HTML can be frustrating, causing your Web pages to appear flat, boring, or unreadable instead of dynamic and exciting. To counter this, you should visit the following sites frequently to keep up on the latest developments in browser technology.

Browserwatch

`http://browserwatch.iworld.com`

This browser offers breaking news in the browser and plug-ins industry, plus browser usage statistics and a rich library of plug-ins and ActiveX components.

Netscape

`http://home.netscape.com`

Here you will find Netscape Navigator, as well as a very rich source of information on Web-related issues.

Microsoft Internet Explorer

`http://www.microsoft.com/ie`

Get the latest version of Microsoft's Web browser here. This site also maintains an archive of press releases and links to articles relating to Internet Explorer.

Amaya

`http://www.w3.org/pub/WWW/Amaya/`

The follow-up to Arena, this browser was developed by Irène Vatton, Vincent Quint, and Daniel Veillard with the purpose of being a testbed for experimenting and demonstrating new specifications and extensions of Web protocols and standards.

Arena

`http://www.yggdrasil.com/Products/Arena/`

Arena is a graphical Web browser with origins that predate proprietary packages such as Netscape Navigator, Microsoft Internet Explorer, and Mosaic. It is the source of a number of innovations that have since been copied by other Web browsers, such as HTML tables and style sheets.

Cyberdog

`http://www.cyberdog.apple.com/`

This site presents a versatile tool that includes a Web browser, using Apple's OpenDoc technology.

Galahad

`http://www.mcs.com/~jvwater/main.html`

Galahad is an offline Web browser.

Lynx

`http://www.nyu.edu/pages/wsn/subir/lynx/platforms.html`

Lynx is a fully featured World Wide Web browser for users on both UNIX and VMS platforms who are connected to those systems via cursor-addressable, character-cell terminals or emulators. This text-based browser is widely used in universities, libraries, and many other situations in which there is a desire to bring the information of the World Wide Web to as wide an audience as possible.

NCSA Mosaic

`http://www.ncsa.uiuc.edu/SDG/Software/Mosaic/NCSAMosaicHome.html`

This Web browser was developed at the National Center for Supercomputing Applications at the University of Illinois in Urbana-Champaign.

Opera

`http://traviata.nta.no/opera.htm`

You can customize this highly configurable browser to fit the user's needs. As a bonus, you can easily configure this browser for users with a handicap, and it is also ideal for slow machines.

SlipKnot

`http://plaza.interport.net/slipknot/slipknot.html`

This Web browser works without a PPP or SLIP connection.

HTML Editors and Web Authoring Tools

Many great tools are available to make writing HTML easier and more efficient. Several *tag editors* look a lot like the text editor you might be using now, but have useful features that your text editor lacks. However, the trend recently has been toward tools that display your page in a WYSIWYG window while you work. These tools make it easy to get immediate feedback on your site's appearance and allow for manipulation of the HTML at the object level.

HTML Tag Editors

First, take a look at the tag editors. The following headings list several of the more popular tools available.

BBEdit

`http://www.barebones.com/bbedit.html`

BBEdit 4.0.4 by Bare Bones Software is available at this site.

HomeSite

`http://www.dexnet.com/homesite.html`

This site is home to HomeSite 2.5, which Nick Bradbury wrote and now Allaire owns. In addition to creating HTML pages and Web sites, HomeSite also can create Cold Fusion Web applications.

HotDog

`http://www.sausage.com`

The HotDog Web editor by Sausage Software is located at this site.

HoTMetaL Pro

`http://www.sq.com/products/hotmetal/hmp-org.htm`

HoTMetaL by SoftQuad is located at this site.

Webber

`http://www.csdcorp.com/webber.htm`

This editor from Cerebral Systems Development includes a validator program to look for invalid HTML. Also check out Webber/Active at the same URL. This product does everything Webber does, and it also includes support for ActiveX controls.

WYSIWYG Editors

WYSIWYG editors have received more attention and more respect as powerful utilities such as FrontPage, Backstage, and PageMill have appeared. Under the following headings, you find links to a few of the most popular and most powerful products in this category.

Netscape Composer

`http://home.netscape.com/try/comprod/mirror/client_download.html`

Netscape Composer is the WYSIWYG editor in the Netscape Communicator suite.

Microsoft FrontPad

`http://www.microsoft.com/ie/ie40/`

FrontPad is the WYSIWYG Web page editor in Microsoft Internet Explorer 4. If you're not sure you want to take the plunge with FrontPage, FrontPad is a great way to find out for free. Although nobody at Microsoft would say that FrontPad is "FrontPage Lite," FrontPad does appear to be a scaled down version of the best-selling commercial product.

Microsoft FrontPage 97

`http://www.microsoft.com/frontpage`

FrontPage 97 provides a total Web site creation solution, including site structuring, HTML editing, site maintenance, and the FrontPage Web Server.

PageMill

`http://www.adobe.com/prodindex/pagemill/overview.html`

PageMill is a powerful WYSIWYG editor from Adobe, the company behind Portable Document Format (PDF) and PageMaker.

Backstage Internet Studio

`http://www.macromedia.com/software/backstage/`

Backstage is Macromedia's WYSIWYG Web development tool. In addition to easily integrating multimedia components to your Web pages, Backstage offers powerful database integration features.

Usenet Newsgroups

Usenet provides a platform for any person with Net access to express to the whole world his or her views on a topic. As you might expect, this results in a great deal of chaos. But mixed into the chaos are a lot of useful and timely ideas. The following newsgroups promise to be full of lively discussions of the latest developments in HTML, browsers, and other Web-related issues. A word of caution: Before posting your question to a newsgroup, read the group to make sure your question has not been asked and answered already. If you don't, you'll learn the unfriendly definitions of terms like *newbie* and *flame-bait*.

The `comp.infosystems.www.authoring.html` Newsgroup

At this very useful, vibrant newsgroup, you will find information on all aspects of HTML authoring.

The `alt.html` Newsgroup

The `alt.html` newsgroup is a much less structured, more relaxed HTML newsgroup.

The `comp.infosystems.www.authoring.misc` Newsgroup

This newsgroup is a forum for discussion of questions not covered in the more specific authoring newsgroups.

The `comp.infosystems.www.browsers.ms-windows` Newsgroup

This newsgroup hosts discussion of Microsoft Windows browsers.

The `comp.infosystems.www.browsers.mac` Newsgroup

This newsgroup hosts discussion of Macintosh browsers.

The `comp.infosystems.www.browsers.misc` Newsgroup

This newsgroup hosts discussion of all other browsers.

APPENDIX E

What's on the CD-ROM

IN THIS APPENDIX

On the *HTML Unleashed* CD-ROM, you will find the sample files that were presented in this book, along with a wealth of other applications and utilities.

NOTE

Please refer to the `readme.wri` file on the CD-ROM (Windows) or the Guide to the CD-ROM (Macintosh) for the latest listing of software.

Windows Software

ActiveX

- Microsoft ActiveX Control Pad and HTML Layout Control

Explorer

- Microsoft Internet Explorer 3

Graphics, Video, and Sound Applications

- Cell Assembler
- Goldwave sound editor, player, and recorder
- MapThis image map utility
- Paint Shop Pro
- SnagIt
- ThumbsPlus
- Web graphics from The Rocket Shop

HTML Tools

- NetObjects Fusion demo
- W3e HTML Editor
- Hot Dog 32-bit HTML editor
- HTMLed HTML editor

Utilities

- Adobe Acrobat viewer
- WinZip for Windows NT/95
- WinZip Self-Extractor

Macintosh Software

Graphics, Video, and Sound Applications

- Graphic Converter
- GIFConverter
- Fast Player
- Sparkle
- SoundApp
- Web graphics from The Rocket Shop

HTML Tools

- NetObjects Fusion demo
- BBEdit LITE
- HTML Web Weaver
- WebMap
- HTML.edit
- HTML Editor for the Macintosh

Utilities

- ZipIt for Macintosh
- ScrapIt Pro
- Adobe Acrobat

About the Software

Please read all documentation associated with a third-party product (usually contained with files named `readme.txt` or `license.txt`), and follow all guidelines.

INDEX

Symbols

A

B

C

D

E

F

G

H

I

J

K

L

M

N

O

P

Q

R

S

T

U

V

W

X

Y - Z