J. Webster (ed.), *Wiley Encyclopedia of Electrical and Electronics Engineering*

HYPERTEXT MARKUP LANGUAGE

As described in an article on Telecommunication Traffic in this encyclopedia, today's human communications over the global worldwide collection of networked computers, the Internet, include not only text, but also a variety of other material such as tables, color graphics, two- and three-dimensional (*3D*) animation, recorded and real-time video and sound, software, and many other elements. This multimedia material can be classified as a generalized text, or *hypertext*, a word coined by Ted Nelson around 1965 at a conference of the Association of Computing Machinery (*ACM*), and later expanded to his Xanadu project (1). Others have refined the definition of hypertext since (e.g., Ref. 2). What is even more important is that multimedia documents located at different geographical locations can be connected through hypertext links (hyperlinks) by merely clicking on a word or a picture.

Hyperlink Concept. This hyperlink concept was proposed by Vannevar Bush in 1945 (3). He proposed the MEMEX, a machine to rapidly access and allow random (or nonsequential or nonlinear) links between pieces of information. Early rudimentary hyperlink and hypertext implementations were generally limited to interlinking material among different files on individual personal computers (*PCs*), but within the same local file system, often using a central link database to keep track of all the links. The advantage of this approach was its guarantee that a link would never point to a nonexistent location. However, this approach was not upward scalable to the global World Wide Web (*WWW*, or the Web, for short). Instead, the universal resource locator (*URL*) had to be introduced to provide a common naming and addressing space to be shared by multimedia documents distributed on the computers scattered around the world. Thus, the Web is a convenient way to navigate through the worldwide collection of computers containing the multimedia documents, known as Web pages.

Hypertext Transfer Protocol. This hyperlinked multimedia communications concept required a suitable means of connecting and transmitting the diverse material over the Web. One of the suitable means was a set of rules (protocols) defining how to communicate between the diverse (heterogeneous) and ever-changing networks and computers that can be found on the Web. One of the key ingredients in this set is the hypertext transfer protocol (*HTTP*), which is used to transfer Web pages from a server to the client, i.e., the Web browser. In addition, many other protocols had to be developed, such as the file transfer protocol (*FTP*) and email protocol (MAILTO), as described in many Web resources.

Hypertext Markup Language. Another suitable means was the introduction of a language to describe the multimedia documents. A possible choice was a new procedural language like C or Ada, with its advantages of precision, compactness, and even object-oriented hierarchical inheritance, but also its limitations that include the time it takes to learn such a language. Another simpler approach was to use an ordinary ASCII (American Standard Code for Information Interchange) text with embedded codes, called *tags*, to describe the diverse multimedia components, with their structure, attributes, properties, and values. Since the latter approach was thought to be accessible to many people, the HyperText Markup Language (*HTML*) was formulated (4,5,6,7,8,9,10). It was based on an earlier, more general markup language, the Standard Generalized Markup Language (*SGML*), used by publishers to describe diverse documents such as books, news releases, and legal documents (11,12).

2 HYPERTEXT MARKUP LANGUAGE

HTML is a semantic markup language intended to describe the meaning and structure of a Web document, and not the physical presentation of the document on a computer screen or paper. The advantage of this semantic approach is that the language just adds information defining the objects (such as paragraphs, headings, and images), as well as other information specifying the meaning of the objects (such as an address or block quotation). This language model also allows invisible or alternative descriptors of the objects (such as an image that cannot be displayed by a browser). The overall intent of this semantic approach is to present the document similarly on any human-interface device, ranging from a graphical display to speech synthesizers to Braille generators.

Tim Berners-Lee, a consulting software engineer at CERN (Centre Européen pour la Récherche Nucleaire), the European particle-physics laboratory in Geneva, played one of the key roles in the initial stages of *HTML* and the Web (13). He had seen the blossoming of the Internet from a small number of military and research institutions in the late 1980s and early 1990s, to many thousands of computers with much information scattered among them. In the summer of 1991, he made his software available to many users. In 1994, he became the first director of the World Wide Web Consortium (*W3C*), a nonprofit organization with more than 100 member organizations and located at the Laboratory for Computer Science, Massachusetts Institute of Technology (*MIT*), to coordinate the development of Web software and standards.

HTML Extensions. In the official first version of HTML, the language was quite rudimentary. Today, there are hundreds of tags, and many techniques to extend the HTML language in order to include images, sound, video or even interactive programs. HTML 2 (4) has been replaced with HTML 3 (5) and HTML 4 (6). Many other extensions are mentioned in the "Future Trends" section at the end of this article.

Page Editors. The design of early Web pages was done by hand, i.e., by typing the commands manually. Today, there are many visual HTML editors, such as Microsoft Front Page and Netscape Composer, to facilitate the production of Web pages. A page is composed in a way that it will be displayed later by a browser. It is then translated into HTML by the editor. Such editors are called WYSIWYG (what you see is what you get). Although they are fast and convenient, many designers fine-tune the resulting HTML manually to achieve greater flexibility and consistency between various browsers and computers on which the pages are to be displayed. In contrast to various page composers (e.g., Adobe PageMaker or Quark XPress) that are capable of producing a page intended for printing, with the exact size and location of the intended text and graphics, a Web page is much less precise in that it must be displayable by any browser, on any computer, with any screen size and any set of installed fonts.

Page-Designer Perspective. There are numerous books describing the evolution of HTML in great detail. Due to space limitation, this article cannot present all the details of the language. Instead, we shall summarize HTML from the perspective of designing a Web page (14,15,16,17,18,19). Such design includes text, graphics, sound, and links to other locations on either the same site, or other sites, It includes the commonly used HTML tags, special characters, tables, frames, forms, and style sheets. It also includes special HTML extensions implemented on the common Web browsers such as Netscape Navigator, denoted by {*NS*}, and Microsoft Internet Explorer, denoted by {*IE*}, with specific version numbers added when required. A comprehensive list of all the HTML tags has been compiled by Homer et al. (8). This compilation shows which tags were introduced by the major HTML versions 2.0, 3.2, and 4.0, and it also lists which of the tags are implemented on the major {NS} and {IE} browsers versions 2, 3, and 4. We have also compiled Appendix 1, listing all the tags presented in this article.

Page Examples. There are few people today who have not seen a page from the Web, now containing over a billion pages. Although graphical representation of such rendered pages could be very useful to illustrate the different constructs discussed in this chapter, it would increase the size of the article many times, not only because of the large number of tags, but also because each browser renders the tags slightly differently on the screen. For completeness, however, we include an example of the Wiley Encyclopedia of Electrical and Electronics Engineering Web site, as rendered by the Microsoft IE version 5.0 (Fig. 1) and Netscape NS version 4.08 (Fig. 2) on the Macintosh. The figures show very few differences between the IE5.0 and NS4.08 browsers

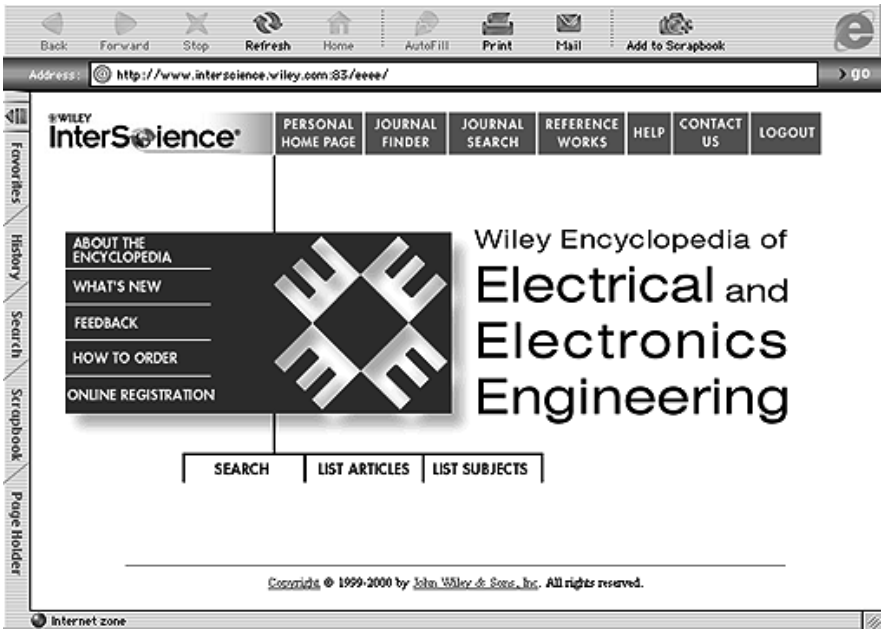

Fig. 1. A Web site as displayed by Microsoft Internet Explorer 5.0.

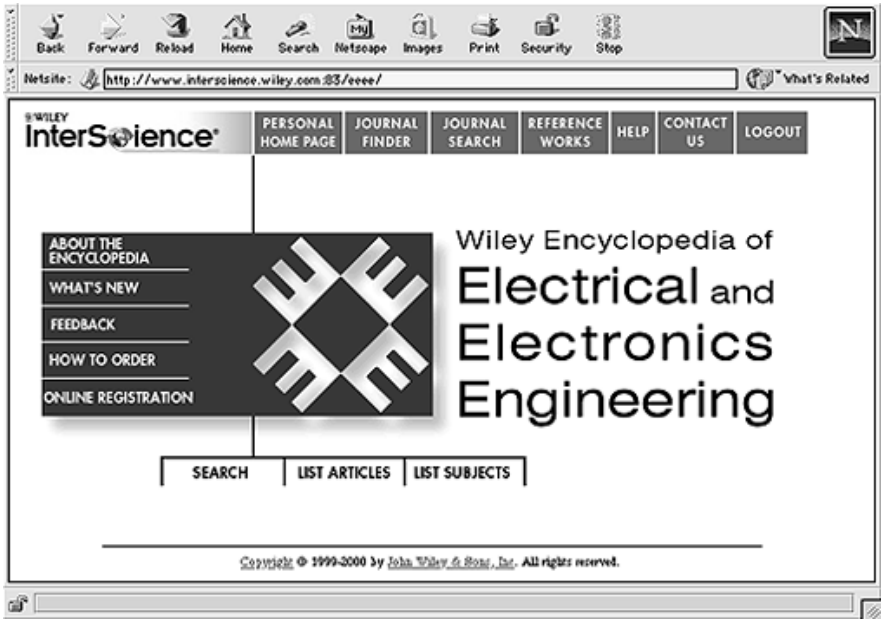

Fig. 2. The Web site of Fig. 1, as displayed by Netscape Navigator 4.08.

when displaying a simple page. We can also see that this home page has many links to the encyclopedia itself and to Wiley-Interscience.

Organization of This Article. The description of the HTML facilities is presented in this article according to the corresponding textual, graphical, table formatting, and other entities used in a page.

The Structure of HTML

Like any markup language, HTML encapsulates various entities—such as paragraphs, images, and addresses of other pages and sites—between an opening tag and a corresponding closing tag. Tags may be nested; e.g. a table may contain another table within one of its cells.

The Structure of Tags. In HTML, a tag is defined as any combination of special keywords between a matching pair of angle parentheses, <>. For example, <HTML> is a legal tag, defining the beginning of a Web page. An opening tag usually requires a closing tag identified by a slash that follows the left angle bracket. For example, </HTML> is the matching closing tag for <HTML>. In addition to the paired tags, there are single tags (*singletons*) to denote such entities as comments.

Some tags require *properties*. For example, the default left-justified paragraph-defining tag, <P>, may be modified to a centered paragraph by <P ALIGN="center">. Notice that the spelling of "center" and many other values is American, and not British.

Transparency of Unknown Tags. It is clear that tags should not be displayed by browsers. However, many new tags have been added since the first definition of HTML. This necessitated a convention in which the new tags not "understood" by many old browsers are ignored, while any text between the ignored tags is displayed. For example, <STRONG>Test</STRONG> displays Test as strongly emphasized characters by browsers that understand this paired style tag, but only as plain Test by those that do not understand the tag.

Conventions Used in This Article. For readability, the tag names and their properties are rendered in uppercase, while their values are in lowercase within quotes. Notice that the HTML is not case-sensitive, treating both upper- and lowercase text in the same fashion, except for special characters such as Ä, which must be displayed as Ä, while ä means ä. Furthermore, although the quotes are not mandatory in the majority of properties, we use them for uniformity.

Optional values are separated by a vertical bar as in "option1|option2", while optional attributes are shown in brackets (square parentheses) as in ["optional_attributes"]. Valued variables are rendered in italics. An ellipsis, . . . , between a starting tag and a closing tag denotes any entity such as text that is omitted for clarity. An ellipsis enclosed in brackets, [. . .], indicates that other values are not included.

Notice that the above convention is in conflict with U.S. conventions. In the latter, for example, if a comma is used as a punctuation mark after a quoted text, the comma must be placed inside the quote (as in "quoted_text,"). However, since the quotation marks are delimiters for values in HTML, the comma cannot be placed inside them, because it would then be used as a part of the value. Thus, this article uses the strict HTML meaning of quotation marks as delimiters, and not as punctuation marks.

Since a complete description of HTML requires an extensive book (e.g., Ref. 6), this article describes a small subset of the tags and attributes to illustrate the main uses of HTML. We have also omitted many tags in both HTML 3.0 and 4.0 that are supported by only a few browsers at this time.

Page Tags

Although a single page could constitute a Web site, the design of a good site involves many pages, organized into several corresponding hierarchical levels in order to reflect the logical structure of its contents, as well

as to load each page quickly even on slow Internet connections. This section describes the structure and the elements of the page construct as the fundamental HTML component in the site design process.

Page Structure Tags. A Web page is defined by the following structure:

```
<!DOCTYPE [...]>
<HTML>
<HEAD> ... </HEAD>
<BODY> ... </BODY>
</HTML>
```

where the first line declares the page type, the <HTML> and </HTML> define the Web-page boundary, and the two sections of the page, <HEAD>...</HEAD> and <BODY>...</BODY>, define the declaration and actual displayable contents of the page, respectively, as described next.

<!– DOCTYPE HTML PUBLIC "..." –>. This comment line is often placed before the formal definition of the Web-page limits in order to describe the *document type declaration* (*DTD*) used within the page. For example, to declare that the content is generic HTML, the DOCTYPE line should be <!– DOCTYPE HTML PUBLIC "<>-//IETF//DTD HTML//EN" –>. However, if the page is written using the HTML 3.0 tags only, the DOCTYPE line should be <!– DOCTYPE HTML PUBLIC "-//IETF//DTD HTML 3.0//EN" –>. Although the DOCTYPE line is not required, many HTML validators (i.e., sites capable of verifying that the Web pages are written correctly) often require it in order to flag errors in the required HTML version.

Notice that anything between the left angle bracket with the exclamation sign, <!–, and the right angle bracket, –>, is ignored by a browser. As in any language, comments are essential for program understanding. In HTML, comments are also used to hide JavaScript and style-sheet codes from older browsers that do not understand the codes and would display them otherwise.

<HTML>...</HTML>. This tag defines the start and end of a Web page.

<HEAD>...</HEAD>. This tag defines the head section of a Web page. It is usually located at the beginning of a page description, following the opening tag <HTML>, as it sets preferences and definitions required by the page, such as its title, default text size, META information, frames, and local style sheets. Since this HEAD section is used by the browser or by Web crawlers, it is not displayed. The tags are described in the next subsection.

<BODY [...]>...</BODY>. This tag defines the contents of the page to be displayed by a browser. Only one pair of BODY tags is allowed in a page.

Tags Defining Properties in the HEAD Section. The nondisplayable HEAD section of a Web page defines many attributes and properties of the displayable BODY section of the page. The major attributes are defined by the following tags. Other attributes are described in the META tag and Frames sections.

Major HEAD Tags.

<TITLE> "text" </TITLE>. This tag sets the title of the Web page to "text", and is usually the name of the displayed window. Notice that Netscape 1.1N, but not later versions, can "animate" the title by inserting more than one TITLE tag.

<BASE [HREF="url"]\|[TARGET="target"]>. This is an optional tag to set the default URL for the page. The URLs used in this page will be relative to the base URL. If the BASE HREF is not used in the page, its base URL is assumed to be the location of the page itself. The BASE TARGET can be used to set the default "target" value to display the window either in the current or new window, as explained in the FRAMES section.

<BASEFONT SIZE="n" [COLOR="color"] [FACE="font"]>.. This optional tag sets the default size of the default font. Optionally, the name of the font can be selected to "font" and its color to "color". Since the FACE and COLOR options are not supported in all the current browsers, they can be set in the BODY section of a page. Notice that the spelling of the word COLOR is American and not British. {IE}

<*BGSOUND SRC="url" [LOOP="n|infinite"]*>.. This tag establishes the source (SRC) of a sound to be played in the background while the page is displayed. Optionally, LOOP defines the number *n* of times the sound should be repeated, with "infinite" causing the sound to be played until the page is closed. The format of the sound must be either audio (.au), or wave (.wav), or MIDI (.mid). {IE, NS3}

<*ISINDEX HREF="url" [PROMPT="text"]*>.. This tag informs the browser that the current Web page is a searchable index. When the page with an ISINDEX is opened, a dialog box appears with a field to enter a keyword. The "url" defines the destination to which the keyword will be sent. The optional "text" is displayed in the dialog box. However, if PROMPT is not used, a default text appears in the dialog box: "You can search this index. Type the keyword(s) to search for."

<*LINK [REL="text"|REV="text"] [TITLE="text"] [HREF="url"]*>.. This tag defines relations between Web pages. A forward relationship is established through REL, a backward relationship through REV. For example, consider the relationship between a chapter in a book and its table of contents (*TOC*) in that volume. The forward relationship between the TOC and say Chapter 7 could be established by REL="Chapter 7" TITLE="Hypertext Markup Language" HREF="chapter7.html". The reverse relationship between the two entities is then established by REV="TOC" TITLE="Table of Contents" HREF="toc.html". Another use of the tag is to include information such as REV="company" TITLE="company_name" HREF="mailto:company_name@domain".

<*STYLE*>...</*STYLE*>.. This paired tag defines global style sheets, as described in the “Style Sheets” section.

<*SCRIPT LANGUAGE="language" [SRC="url"]> "actual code here" </SCRIPT*>.. This paired tag informs a browser that the enclosed text is a code in a language such as JavaScript or VisualBasic, and it should be executed, rather than displayed. If the optional "url" is used, than the code may be located outside the current file. Notice that since older browsers do not recognize the SCRIPT tag, they display the actual code. To suppress the code from being displayed, one can write the actual code as a commented block of lines, <!– actual code here –>.

<*NOSCRIPT*>"*text*"</*NOSCRIPT*>.. This paired tag is related to the SCRIPT tag in that the "text" is displayed in browsers that do not support the SCRIPT tag, thus informing the user that the script has not been executed.

META Tags Related to the HTTP Protocol.. There are many META tags used to provide information about the current page to browsers and search engines. All the META tags are singletons. They can be subdivided into groups. The first group is related to the HTTP protocol, and the others to the page descriptors. A few of the HTTP-related tags are described next.

<*META HTTP-EQUIV="content-type" CONTENT="MIME type [; CHARSET=charset]*">.. This tag defines content using a MIME type other than the server’s default setting, and optionally defines another character set. For example, <META HTTP-EQUIV="content-type" CONTENT="text/html; CHARSET=ISO-8859-1" uses text/html MIME type and the ISO-8859-1 character set.

<*META HTTP-EQUIV="text" CONTENT="text"*>.. This tag is a replacement of the header sent by the server software. A header is a part of the HTTP protocol, and may contain the length of the Web page, the date when the page was last changed, and the server’s software.

<*META HTTP-EQUIV="refresh" CONTENT="n [; URL=url]*">.. This tag requests a reload of the current Web page every *n*th second. The optional URL specifies the location of the page to be opened. This tag is often used in “welcome” pages. However, since not all browsers support this tag, a standard link to the page should also be provided.

<*META HTTP-EQUIV="expires" CONTENT="day, date year time timezone*">.. This tag specifies when the current page should be removed from caches or databases so that old material cannot be searched for. This is applicable to all time-sensitive material, including newspapers and meeting announcements. Some search engines won’t keep the page content in the search database after this time. For example, <META HTTP-EQUIV="expires" CONTENT="Sat, 06 Jan 1990 14:30:00 GMT">.

META Tags Related to Page Properties. As with the HTTP-related, there are many other META tags used to provide information about the current page to browsers and search engines. A few of them are described next.

<META NAME="generator" CONTENT="text">.. This tag is often added by HTML editors or manually to indicate which program has been used to create the Web page. For example, <META NAME="generator" CONTENT="BBEdit 4.03, Macintosh, handcrafted"> specifies BBEdit on the Macintosh as the HTML editor.

<META NAME="description" CONTENT="text">.. This tag provides a search engine with a short description, "text", of the page.

<META NAME="keywords" CONTENT="word1, word2, . . ., wordN">.. This tag provides a search engine with the keywords, "word1, word2, . . ., wordN", describing the page.

<META NAME="distribution" CONTENT="global|local">.. This tag defines the page either as an index page (global) or not (local).

<META NAME="author" CONTENT="author's name">.. Like many other similar tags, this specifies the author of the Web page.

<META NAME="copyright" CONTENT="text">.. This tag provides a search engine with a copyright notice, "text". For example, "©1997, Author".

<META NAME="robots" CONTENT="all|none|nofollow|noindex">.. This tag informs a search engine which pages should be indexed. If "all" is specified, both the current and the linked pages will be indexed. If "nofollow" is specified, the current but not the linked pages will be indexed. If "noindex" is specified, the linked pages but not the current page will be indexed. If "none" is specified, no pages will be indexed at all.

<META NAME="language" CONTENT="language">.. This tag informs a search engine what natural language (e.g., English) the page is written in.

Tags Defining Properties of the BODY Section. There are numerous singleton tags to specify properties of background of the Web page, as well as its text, links, and margins. The tags are placed in the <BODY> structural page tag. For example, <BODY BGCOLOR="#ffffff" BACKGROUND="cpics/logos/backgrnd.jpg" BGPROPERTIES="fixed"TEXT="#000000" LINK="#0000ff" ALINK="#ff0000" VLINK="#9900ee"> specifies white background, texture, the color of the text displayed in the page (usually black), and the text colors of the displayed link (usually blue), activated link (usually red), and visited link (usually magenta), respectively. The individual properties are described next.

BGCOLOR="color".. This tag sets the background of the current Web page to the specified color. The "color" value can be specified as either a hexadecimal number, "#rrggbb", representing the fundamental red, green, and blue (*RGB*) colors (two digits per color), or a reserved word, "colorname". For example, the white color of the background is specified as either BGCOLOR="white" or BGCOLOR="#ffffff".

The default color of a Web page is "gray". Notice that the spelling of "gray" is American, and not British.

BACKGROUND="url".. This optional tag sets the background texture of the current Web page to the specified picture, thus overriding the background color. If the picture is smaller than the page, it will be tiled (repeated in two dimensions) as a seamless pattern over the page.

BGPROPERTIES="fixed".. This optional tag fixes the page background, defined by the BACKGROUND="url" tag as a non scrolling picture. {IE}

TEXT="color".. This tag sets the default color of the text displayed within the page. Black is the usual color, and is either TEXT="black" or TEXT="#000000". Notice that all three colors have 00 intensities.

LINK="color".. This tag sets the default color of the links displayed within the page. Blue is the usual color, and is either LINK="blue" or LINK="#0000ff". Notice that only the blue color has the full ff intensity.

VLINK="color".. This tag sets the color of the already visited links. Netscape usually defaults to magenta (VLINK="#9900ee"), and Explorer to red (VLINK="#ff0000").

ALINK="color".. This tag sets the color of an activated link (i.e., at the moment when a link is clicked on). Red is the usual color, and is either ALINK="red" or ALINK="#ff0000".

LEFTMARGIN="n".. This tag defines the left margin of the current page, where *n* is the number of pixels between the left border of the displayed window and the page content. {IE}

TOPMARGIN="n".. This tag defines the top margin of the page, also in pixels. {IE}

Hyperlinks

As indicated in the introduction to this article, hyperlinks, or *links* for short, are the most fundamental constructs in any hypertext markup language. A special word, or sentence, or picture, or sound may be marked up to provide a link to another entity such as another file, or another picture, or the actual sound within either the same page or another page. The format of a link is discussed next.

<A HREF="[url][#section]" [TARGET="target_name"]>...</A>. This paired tag defines the anchor (A) of a hyperreference (HREF), and optionally a target window in which the HREF should be displayed. The URL defines the address to the link. It may be an address to a file on either the same server hosting the site, or another server. Addressing on the same server may be either absolute (i.e., requiring the full server name, the directory name, and the file name), or relative (requiring the file name only, with address references relative to the current directory). On the other hand, addressing on another server can be absolute only. For example, to link to a file, say "test1.html", on the same server and in the same directory, one can use <A HREF="test1.html"></A>, or optionally <A HREF="./test1.html"></A>, as ./ denotes the current directory. To link to a file, say test2.html, on the same server but in another directory located one level closer to the root directory than the current directory, use <A HREF="../test2.html"></A>. To link to a file, say "test3.html", on the same server but in another directory located two levels closer to the root directory than the current directory, use <A HREF="../../test2.html"></A>. Clearly, this scheme can be expanded to any number of directory levels, up to the root of the server. Notice that the current directory may also be specified by the BASE HREF tag.

The above scheme of relative addressing also applies when the linked file, say "test3.html", is located within the same server, but in another directory, say books. For example, if books is located within the current directory, use either <A HREF="books/test3.html"></A> or <A HREF="./books/test3.html"></A>. However, if books is not in the current directory, but one level above it, than the relative addressing requires <A HREF="../books/test3.html"></A>. Again, this scheme can be expanded to any number of directory levels, up to the root of the server, and to any number of directories. Notice that <A HREF="/books/test4.html"></A> would link to test4.html located in the directory books located at the root level of the server.

Linking to a file, say test7.html, in the directory books on another server, say www.wiley.com, use the absolute addressing in the form <A HREF="http://www.wiley.com/books/test7.html"></A>. Linking to the main (home) page of any site also requires the absolute addressing. For example, if the home page at Wiley is called main.html, then the link is <A HREF="http://www.wiley.com/main.html"></A>. It is also useful to note that the link can be shortened to <A HREF="http://www.wiley.com/"></A> by naming the home page index.html. Also notice that, if required, the "http://url" protocol can be replaced by the "ftp://url" protocol, or by the email protocol in a slightly different form, "mailto:person@domain".

The optional "#section" extends the URL to a section within a linked page. The anchor to the section is set by the following paired tag: <A NAME="text">["text"]</A>. Linking to the anchor, say toc, from the same page, say test10.html, requires <A HREF="#toc"> Return to TOC within this page. </A>, while linking to toc from another page in the same directory requires <A HREF="test10.html#toc">Go to TOC within Test10</A>.

Notice that this relative addressing scheme is very important when the site must be moved from one location to another, as all the references will still be valid at the new location. On the other hand, the absolute addressing scheme is rigid in that any change in the address breaks it permanently.

Since testing for valid links is extremely time-consuming, many tools have been developed to verify if the hyperlinks used in a page or a site are active or broken.

The TARGET parameter can be used to display a linked entity in either the same or a different window or frame (see the section on frames). The values of TARGET are

$$\text{TARGET} = \text{“\_self|\_top|\_parent|\_blank”}$$

where "_self" opens the link into the current window (or frame), "_top" opens the link in the top window (not a frame), and "_parent" opens the link in the frame that contains the FRAMESET tag for the current frame. The value "_blank" opens a new window.

As we have discussed in connection with the BODY section, the links are usually displayed in a color (usually blue) that is different from the ordinary text (usually black), and are underlined. To enhance the readability further, a visited link is displayed in a different color (usually magenta). The underlining of textual links can be changed through *style sheets*. On the other hand, linked pictures have a colorized border, which can be removed using the BORDER property in the IMG tag, as described in the section on images.

Textual Contents

Simple text constitutes much of the content within a typical Web page. Such text is arranged in headers, paragraphs, and lists, all separated by rules, as described next.

Latin Characters, HTML Character Entities, and URL Character Encoding. The Latin alphabet constitutes the basis for the HTML code. It is suitable for the English language, but insufficient for other natural languages that include accents (e.g., é, ü) and even for HTML itself in that one cannot include the angle brackets, <>, in any text, because they are interpreted by a browser as the start and end of a tag. A solution to this problem is to expand the concept of a character to the so-called *entities* that use the standard characters to describe the special characters. According to the *ISO* Latin-1 standard developed by the International Standards Organization (ISO), an entity has two framing characters: a starting ampersand, &, and an ending semicolon, ;. The content of an entity can be either a set of the standard Latin characters or a decimal number representing the extended 8-bit ASCII character. The number is preceded by a number sign, #. For example, both < and < mean the less-than sign, <. The other common special characters are > or > (which means >), & or & (which means &, which is used to start a special character entity itself), and   or   (which means non breaking space, (to insert multiple spaces that are not ignored by a browser). Similarly, ´ and é are both interpreted as é. Notice that if special character entities are not used in the page, <META HTTP-EQUIV="Content-Type" CONTENT="text/html; charset=iso-8859-1"> should be included within the <HEAD> tag. The entire set of ISO Latin-1 (ISO-8859-1) is included in many sources (6,8) and is recompiled in Appendix 2.

There is also a universal character set (Unicode), based on two 8-bit bytes, for international use and for the Internationalized HTML. The current second version of Unicode is equivalent to the basic multilingual plane subset of the multibyte universal character set, ISO 10646, for the characters and symbols used by all the world's languages.

In addition to the character entities, each enclosed between an ampersand and a semicolon and understood within an HTML document, another character encoding scheme is required for URLs to avoid confusion with the structure of the hypertext anchor tag. Thus, the noncurly doublequotes ("), space, tilde (~), percent sign (%), forward slash (/), hash (#), the question mark (?), and the 33 control characters (hex codes 00 to 1F and 7F) are all disallowed, as listed in Appendix 3. If such characters must appear in the URL, they must be encoded as *%xy*, where the percent sign indicates the start of the encoding triplet, and *xy* is the two-digit hexadecimal

code representing the character. For example, if a file has a name "Test File" (as allowed on the Macintosh and Windows 95 and later), the space must be encoded as "Test%20File", which encodes the space.

Text Headers. Headers often precede text that is arranged in paragraphs or other units. Since the headers are frequently used as section or subsection titles, they require different sizes and emphasis in order to distinguish them from the ordinary text. The <Hn> tag satisfies all of the requirements. A further advantage of using the header tag is that it surrounds the header by blank lines automatically. The header tag is written in the following format:

$$< \mathrm{H}n\,[\mathrm{ALIGN} = \text{“left|right|center”}] > \text{“text”} < /\mathrm{H}n >$$

This paired <Hn> (header of size n) tag displays "text" in bold, with the size n ranging from 1 (for the largest size) to 6 (for the smallest size). The optional ALIGN selects the alignment of the header either to the left, or the right, or the center. If ALIGN is not used, the default alignment is the same as that of the previous text. For example, <H3>Textual Contents</H3> would display the header as the standard size 3 text in bold, left justified, if the previous text was left justified.

There are other headers, including list headers <LH>, as described in the subsection on lists.

Paragraphs.

<P [ALIGN="left|right|center|justify"]>...[</P>].. This tag defines a paragraph, separated from another paragraph by a blank line. The optional ALIGN can be used to align the text in the current paragraph either to the left or right margin, or to center the text, respectively. In some browsers, "justify" aligns the text to both left and right margins. Notice that the paragraph tag is now defined as a paired tag, although the current browsers also understand the singleton <P>. Also notice that more than one empty paragraph tag (i.e., a paragraph without any text or another tag) is ignored by browsers. If one wishes to insert more than one line between paragraphs, either the break <BR> tag, or a TABLE, or a transparent spacer picture should be used. Another important feature of HTML is that only single spaces are displayed; multiple spaces are ignored. The optional ALIGN works in the newer browsers only. {IE, NS4}

<DIV ALIGN="left|center|right|justify">...</DIV>.. If more than one paragraph is to be centered, this *division* paired tag can be used. Notice that this tag can be used to justify not only text, but also tables and pictures. However, as with paragraphs, the DIV tag works in the newer browsers only. {IE, NS4}

<CENTER>...</CENTER>.. If more than one paragraph is to be centered, this CENTER paired tag can be used. Notice that this tag can be used to center not only text, but also tables and pictures, in all browsers. Also notice that the spelling of the word CENTER is American and not British.

<BR [CLEAR="left|right|all"]>.. This singleton tag inserts a line break. The optional CLEAR makes the tag even more important in that it can be used to control the text floating around justified pictures. That is, when a picture (or another object) is left-justified, the text may flow on the right side of the picture. Any successive paragraph would continue to flow around the picture. This flow can be interrupted by the use of <BR CLEAR="left">. Any text past this break will be displayed below the picture. For a right-justified picture, <BR CLEAR="right"> should be used. For both left- and right-justified pictures, <BR CLEAR="all"> should be used. The section on image tags contains more on picture justification.

<NOBR>"text"</NOBR>.. This paired tag forces any text to be displayed on a single line. For example, the first and last names of a person should appear on a single line. Similarly, a telephone number should not be wrapped between two lines.

<WBR>.. If placed within a long word, this singleton <WBR> (word break) tag allows the word to be broken between two lines only at that point in it.

<MULTICOL COLS="n" [GUTTER="m"] [WIDTH="k"]>"text"</MULTICOL>.. In some browsers, the text within this paired tag is displayed in columns. The COLS attribute sets the number of columns to n. The optional GUTTER sets the space between the columns to m pixels, usually 10, and WIDTH sets the width of

all the specified columns to *k* pixels. Since not all the browsers understand this tag, multicolumn text can be displayed using tables. {NS3}

Lists. Lists are often useful to display bulleted items that are indented automatically. There are three major different styles of lists: (i) unordered, (ii) ordered, and (iii) definitions. The three lists can be nested, i.e., they can embed other lists. Since the two other kinds of lists, MENU and DIR, are not supported in all browsers, they are not recommended any longer.

Unordered Lists.

<UL [TYPE="disc|circle|square"]>"list of items"</UL>.. This paired <UL> (unordered list) tag displays all the list items contained between the opening and closing tags. The optional TYPE sets the bullet to either a "disc" (large black-filled dot), or a "circle" (unfilled circle), or a "square". If TYPE is not used, the default value is "disc" at the first level, but it is "circle" at the second, and "square" at the third.

<LI [TYPE="disc|circle|square"]>"text"[</LI>].. This <LI> (list item) tag displays a new line containing the "text", starting with the selected bullet. Notice that the <LI> can have a closing </LI> tag. Also notice that although intended for the <UL> tag, it can be used as a standalone tag. The latter use is facilitated by the optional TYPE in <LI>.

Ordered Lists.

<OL [TYPE="1|A|a|I|i"] [START="n"]>"list of items"</OL>.. This <OL> (ordered list) tag displays all the items of the list, similarly to the <UL> tag, except that the items are now put in an order. The optional TYPE sets the order to be determined by either numbers (1), or uppercase letters (A), or lowercase letters (a), or uppercase Roman numbers (I), or lowercase Roman numbers (i). The optional START is the first number of the enumeration, with a default value of 1.

<LI [TYPE="1|A|a|I|i"] [VALUE="n"]>"text"[</LI>].. Similarly to the <LI> tag used in the <UL> tag, this item1 tag displays a new line containing the text, preceded by the selected order. Notice that the <LI> can have a closing </LI> tag. The optional VALUE used to be a Netscape extension, but became part of HTML 3.2. It changes the sequential order of the current and subsequent items. Also notice that although intended for the <OL> tag, it can be used as a standalone tag. The latter use is facilitated by the optional TYPE in <LI>.

Definition Lists.

<DL>"list of items"</DL>.. This paired <DL> (definition list) tag displays all the list items contained between the opening and closing tag. Each item has a <DT> (definition term) and a corresponding indented <DD> (definition data), as described next.

<DT>"text"[</DT>].. This <DT> definition term) tag displays the "text" left-justified. Notice that the opening <DT> tag may have an optional closing </DT> tag.

<DD>"text"[</DD>].. This <DD> (definition data) tag displays the "text" indented with respect to the definition term. Notice that the opening <DD> tag may have an optional closing </DD> tag.

Altering Text Styles and Displayed Behavior. HTML provides many facilities for local modifications to the displayed text, from physical styles, such as the explicit bold or subscript, to logical styles, such implicit emphasis or citation. The actual forms of displaying the logical styles are defined within a browser.

Physical Styles.

<B>"text"</B>.. This paired tag displays the enclosed text in bold.

<I>"text"</I>.. This paired tag displays the enclosed text in italics.

<U>"text"</U>.. This paired tag displays the enclosed text as underlined. Today, underlining of text is not practiced, as it used to be an editorial mark for a typesetter to set the underscored text in italics, and HTML can display text in italics through its <I> tag. Another reason for not using the <U> tag is that the underscored text may be confused with hyperlinks, which are usually underscored. {NS3, IE}

<TT>"text"</TT>.. This paired tag displays the enclosed text in a monospace (i.e., equal character spacing) font, usually Courier.

<*S[TRIKE]*>"*text*"</*S[TRIKE]*>.. This paired tag displays the enclosed text in the strikethrough style (i.e., with a horizontal line through the middle of it). The abbreviated form <S> does not work in older browsers. {NS3, IE}

<*SUP*>"*text*"</*SUP*>.. This paired tag displays the enclosed text as superscript.

<*SUB*>"*text*"</*SUB*>.. This paired tag displays the enclosed text as subscript.

Logical Styles.

<*EM*>"*text*"</*EM*>.. This paired tag displays the enclosed text as emphasized text, usually as italics.

<*STRONG*>"*text*"</*STRONG*>.. This paired tag displays the enclosed text as strongly emphasized text, usually as bold.

<*DFN*>"*text*"</*DFN*>.. This paired <DFN> (definition) tag displays the enclosed text as either italics or bold italics.

<*ADDRESS*>"*text*"</*ADDRESS*>.. This paired tag displays the enclosed text in italics. It is often used at the end of the Web page to display the author's data such as email address. This may be useful for some search engines.

<*CITE*>"*text*"</*CITE*>.. This paired tag displays the enclosed text as indented text (citation), usually rendered in italics.

<*BLOCKQUOTE*>"*text*"</*BLOCKQUOTE*>.. This paired tag displays the enclosed text as indented block text.

<*PRE [WIDTH="n"]*>"*text*"</*PRE*>.. This paired tag displays the enclosed text exactly as it appears in the HTML code, including linebreaks and multiple spaces. The optional WIDTH is the maximum length *n* of a line, expressed in characters (in NS4 only). If this width is exceeded, the text wraps to the next line. Notice that all hyperlinks embedded within the PRE formatted text are both displayed and functional as standard links.

<*CODE*>"*text*"</*CODE*>.. This paired tag displays the enclosed text as a monospace typewriter font such as Courier or Monaco. It can be used to display source code in a computer language.

<*KBD*>"*text*"</*KBD*>.. This paired tag is used to mimic a user typing the text from a keyboard. It is displayed as a typewriter font, sometimes in bold.

<*SAMP*>"*text*"</*SAMP*>.. This paired tag is used to mimic a computer returning a status message. It is displayed as a typewriter font.

<*VAR*>"*text*"</*VAR*>.. This paired tag defines "text" as a variable, usually rendered in italics.

<*SPAN STYLE="css"*>"*text*"</*SPAN*>.. This paired tag defines the cascaded style sheet formatting for the page, as described in the subsection on style sheets.

Modification of Text Attributes. Recall that the global size of the text in the current Web page can be defined by the <BASEFONT> tag in the <HEAD> section of the page. Similarly, the global color of the text displayed in the page can be defined within the <BODY> tag. However, there are many occasions when these text attributes should be modified locally.

<*FONT [SIZE="n|+n|-n"][COLOR="color"][FACE="fonts"]*>"*text*"</*FONT*>.. This paired tag can modify the size, color, and face of "text". The optional SIZE can take absolute values between 1 (for the smallest text size) and 7 (for the largest text). The changes in the text size can also be relative to the default size (i.e., size 3), or as specified by the <BASEFONT> tag. For example, to increase the size by two from the default value, we use SIZE="+2". The obvious advantage of relative sizing is the preservation of the proportions of the text within a page after default-size modification. The optional COLOR specifies the color of "text". The optional FACE is a comma- or space-separated list of font names, with the first name being tried first, followed by the second if the first is not available on the computer running the browser. Since the standard font names differ on different computers, they should be included in the "fonts" list. For example, since Helvetica on the Macintosh computers is similar to Arial on a PC with Windows-based operating systems, we could use <FONT FACE="Helvetica,Arial,sans-serif">. Furthermore, since some fonts display better on the screen, the following specification could be used: <FONT FACE="Geneva, Helvetica, Verdana, Arial, sans-serif">. {NS3, IE}

<BIG>"text"</BIG>.. This paired tag increases the size of "text".

<SMALL>"text"</SMALL>.. This paired tag decreases the size of "text".

Dynamic Text. In some browsers, such the Internet Explorer, text may be displayed as autoscrolling ticker tape to emphasize some important points. The following description is included as an example of such a dynamic text in HTML.

<MARQUEE ["attributes"]>"text"</MARQUEE>.. This paired tag makes "text" autoscrolling. The following optional attributes can be used within the tag {IE}:

WIDTH="*n*|*n*%" HEIGHT="*m*|*m*%" These two attributes set the width to *n* pixels and height to *m* pixels of the marquee. Notice that the width and height may also be specified as a percentage of the current window.

VSPACE="*n*|*n*%" HSPACE="*m*|*m*%" These two attributes set the space between the vertical and horizontal boundaries of the marquee and any other items on the page, to *n* pixels and *m* pixels, respectively.

ALIGN="top|middle|bottom" This attribute sets the alignment of the marquee to either top, middle, or bottom.

BEHAVIOR="scroll|slide|alternate" This attribute defines how the marquee should behave when displayed. The default value, "scroll", moves the text from one side to the other, with the direction specified by the DIRECTION attribute. The "slide" value also ticker-tapes the text, but the movement stops when the text reaches the other side of the field. The "alternate" bounces the text from one end to the other. Notice that the spelling of BEHAVIOR is American, and not British.

DIRECTION="left|right" This attribute sets the side at which the first character is displayed.

BGCOLOR="color" This attribute sets the background color of the marquee.

LOOP="*n*|infinite" This attribute specifies the number of times "text" is to be moved within the marquee. The "infinite" value moves "text" until the page is closed.

SCROLLAMOUNT="*n*" This attribute specifies a space of *n* pixels separating the successive scrolls of "text".

SCROLLDELAY="*n*" This attribute specifies the duration of a pause of *n* milliseconds separating the successive scrolls of "text".

Forms as Interactive Data. In the previous sub-subsection, we have described dynamic text that is provided by the server to the client (browser). This subsection describes data that can be transmitted either from the client to the server ("posting"), or from the server to the client ("getting"). A common method of processing the data requires a common gateway interface (*CGI*) script, residing at the server. However, since not all servers permit CGI scripts, the data may be sent to an email address for processing there, often by specialized programs. Such an interaction between the user and a recipient is accomplished through a form that contains objects such as text field to input or display simple text, radio buttons to select one of many options, check boxes, submit buttons, block-text fields to submit a larger block of text, and popup menus to select a single or multiple items. The following description is included as an example of such an interactive data form in HTML.

The FORM Tag.

<FORM METHOD="post|get" ACTION="url" [ENCTYPE="enctype"] [NAME="text"] [TARGET="target"]>"form objects"</FORM>.. This paired <FORM> tag displays a new form in the current page, as defined by the "form objects". Depending on the value of the attribute METHOD, the data related to the form can be either posted from the client to the server ("post") or obtained from the server and displayed in the form at the client's site. The data are transmitted to the URL as specified by the attribute ACTION. This URL can point to either a CGI script (i.e., a program running on the server, if the server allows it), or an email address, or a Web page. If METHOD="get", the data are sent together with the "url" as a string of up

to 256 characters in the format "script_name?data". If METHOD="post", the data are transmitted either to the server or to an email address. The optional ENCTYPE specifies the encryption type, such as "text/plain" (if email is selected) or "multipart/form-data" (for file transmission). The default value is "application/x-www-form-urlencoded". The optional NAME sets a value that can be used by scripting languages (e.g., JavaScript). The function of the optional TARGET is the same as described in the “Hyperlinks” section.

The Simple Text and Button Form Object.

<INPUT NAME="name" TYPE="set of values" ["attributes"]>.. This singleton <INPUT> tag specifies a form object, such as a field, whose NAME is set to "name" so that the data (value) obtained from the field could be sent as "name=value". The attribute TYPE defines the type of the object, and may have one of the following values:

TYPE="text|password|radio|checkbox|hidden|file|reset|submit|image|button".. The type of the <INPUT> object can be either a simple text field (when "text" is selected), or a password field whose contents are not displayed explicitly, but as large dots ("password"), or a radio button where only one can be selected in a group of radio buttons with the same name ("radio"), or a checkbox field ("checkbox"), a hidden or invisible field that cannot be edited ("hidden"), or a request button to upload a file (which requires special software located at the server and that ENCTYPE in the <FORM> tag be set to "multipart/form-data") ("file"), or a reset button to restore all objects to their default values ("reset"), or a submit button used to send the form data ("submit"), or a clickable image similar in action to the submit button ("image"); or, in some browsers, the value "button" may be selected for use with JavaScript. A form must include at least one <INPUT> object whose TYPE is set to either "submit" or "image" in order to send the data away from the form.

The following optional attributes may be used in this <INPUT> tag.

VALUE="value" If TYPE="text" is set, this attribute specifies the default value, such as "Enter your name", to be displayed in the text field. If TYPE is set to "radio" or "checkbox", the value is sent only if the radio button or checkbox is checked. If TYPE is set to "submit" or "reset", the value is the name of the submit or reset button, respectively.

SIZE="*n*" This attribute defines the width *n* of a field object in characters.

MAXLENGTH="*m*" This attribute defines the maximum length *m* of the field object in characters.

CHECKED This attribute sets the default ("true") highlight of a radio button or a check in checkbox to be displayed when the form opens.

SRC="url" [ALIGN="right|center|left"] In image objects, SRC specifies the URL of the image, while the optional ALIGN functions as in the <IMG> tag.

The Block Text Form Object. The previous form objects are useful to submit short data. For larger data, the <TEXTAREA> tag can be used.

<TEXTAREA NAME="name" COLS="n" ROWS="m" [WRAP="off|soft|hard"]>"text"</TEXTAREA>.. This paired tag creates a scrollable text field with the number of columns (horizontal characters) specified by COLS, and the number of rows (vertical characters) specified by ROWS. The optional WRAP specifies the type of text wrapping. If WRAP="off" (also the default value), horizontal scroll bars appear in the field. If WRAP="soft|hard", the text wraps around to a new line when it reaches the right side of the scrollable text field. More specifically, when WRAP="soft", the text is wrapped in the browser, but is sent as not wrapped. On the other hand, when WRAP="hard", the text is sent wrapped (i.e., with line breaks). Finally, "text" is the default text that is displayed in the field prior to entering any user’s text; for example, "Enter any comment here".

The Popup Menu Form Object.

<SELECT NAME="name" [SIZE="n"] [MULTIPLE]>"options list"</SELECT>.. This paired tag creates either a pop-up menu, or a full list of items (options). The optional SIZE sets the number *n* of displayable

rows in the menu. If SIZE is not used or if SIZE="1", a pop-up menu is created. If SIZE is set to a number greater than 1, the list is created. Optionally, MULTIPLE allows more than one item to be selected by pressing multiple keys, such as the COMMAND/ALT, SHIFT, and CONTROL combinations. The options list includes the items that appear on the popup menu or the list, each defined by the following <OPTION> tag.

<*OPTION [VALUE="value"] [SELECTED]*>"*text*".. This singleton tag defines a single menu item displayed in the popup menu or the list as "text". The optional VALUE specifies the value that is sent if this specific item is selected from the menu or the list. The optional SELECTED makes this preselected when the page opens.

Simple Graphics as Separators and Spacers

Quite often, readability of a Web page may be improved by separating its text or graphics into blocks. The simplest form of a separator is a *horizontal rule*. Another separator is a *stretched pixel*. Still another is a *spacer*.

Horizontal Rules.

<*HR [attributes]*>.. This singleton <HR> (horizontal rule) tag inserts a thin line with a shade. The optional attributes may set the thickness, width, color, alignment, and 3D feature, as described next.

<*HR SIZE="n"*>.. The SIZE attribute sets the height of the line to *n* pixels. If no SIZE is used, the default value is 2.

<*HR WIDTH="n|n%"*>.. The WIDTH attribute sets the width of the line, either to *n* pixels or to *n%* of the width of the screen. The absolute value *n* is used when the size of the window is fixed. For example, a form to be printed in a portrait sheet of paper should not change its size when the window is resized. On the other hand, the percentage is used when one anticipates resizing of the window, and the displayed proportions are important. If no WIDTH is used, the default value is 100%.

<*HR COLOR="color"*>.. The COLOR attribute sets the color of the line, but in some browsers only. The default color is black. {IE}

<*HR ALIGN="left|right|center"*>.. The ALIGN attribute aligns the line. The default value is "center".

<*HR NOSHADE*>.. The NOSHADE attribute changes the appearance of the line from the default 3D (shaded) line to a plain two-dimensional (*2D*) line.

Stretched Pixels. Since colored horizontal rulers are limited to some browsers only, an alternative means of drawing a line of any color and size is to draw a picture using the <IMG> (image) tag, as described in the section on image tags.

Spacers. Since HTML ignores multiple space characters, spacing objects must be accomplished by other means.

<*SPACER TYPE="horizontal|vertical|block" [SIZE="n"] [HEIGHT="n|n%" WIDTH="n|n%" [ALIGN="left|right"]]*>.. In some browsers, this singleton <SPACER> tag inserts an empty space that separates objects. The TYPE attribute sets the required space to either horizontal space, or vertical space, or a 2D block space. The SIZE attribute is either the height in pixels (for "vertical") or width (for "horizontal"), but is not used with "block". When using blocks, the HEIGHT, WIDTH, and ALIGN properties can be selected. This is similar to the IMG tag. {NS3}

Tables as Structural 2D Composition of Objects

Tables are one-dimensional (*1D*) or 2D constructs, containing adjacent cells. An entire table may have some global attributes that affect all its cells. However, each cell is independent in that formatting in one cell does not affect any other cell. The inclusion of tables in HTML provided an important means of 2D assembly of not

only text, but also other objects such as images. Unlike the horizontal rules just described, tables also provide vertical rules. The syntax of the <TABLE> tag is described next.

The Main <TABLE> Tag.

<TABLE ["attributes"]>...</TABLE>.. This paired tag defines a table. The following optional attributes can be used:

ALIGN="left|right|center" This attribute aligns the table to the left, right, or center of the current window.

BORDER="*n*" This attribute sets the size of the border of the table to *n* pixels. Since the border has light and dark shades, the table looks elevated when displayed. When BORDER="0", the border is not displayed.

WIDTH="*n*|*n*%" This attribute sets the width of the table either to a fixed size of *n* pixels, or to a variable size expressed as a percentage *n*% of the displayed Web page. The fixed size is very helpful if one wants to keep the proportions of a carefully designed table intact in windows of various sizes.

HEIGHT="*m*|*m*%" This attribute sets the height of the table either to a fixed size of *m* pixels, or to a variable size expressed as a percentage of the displayed Web page.

CELLSPACING="n" This attribute sets the space between cells to *n* pixels.

CELLPADDING="m" This attribute sets the space between the border and the content of all the cells to *m* pixels.

BGCOLOR="color" This attribute defines the color of the background of the entire table. {IE, NS3}

BACKGROUND="url" This attribute defines a URL of the background image file for the entire table. {IE}

BORDERCOLOR="color" This attribute defines the color of the border of the table. {IE, NS4}

BORDERCOLORLIGHT="color" This attribute defines the color of the light shadow of the table. {IE}

BORDERCOLORDARK="color" This attribute defines the color of the dark shadow of the table. {IE}

FRAME="void|above|below|hsides|vsides|lhs|rhs|box" This attribute defines which sides of the table border should be displayed. The values are selfexplanatory. {IE}

RULES="rows|cols|none|all|groups" This attribute specifies the horizontal ("rows") or vertical ("cols") rules in the table that should be displayed. The value "none" displays the outside border (frame) of the table. The default value "all" displays all the rules. The value "groups" displays lines between the major sections of a table: THEAD, TBODY, and TFOOT. {IE}

The Row and Cell Tags.

<TR [ALIGN="left|center|right"] [VALIGN="top|middle|bottom|baseline"] [BGCOLOR="color"] [BACKGROUND="url"]>"list of cells"</TR>.. This paired <TR> (table row) tag defines a new row within the table and displays any text within all the cells, depending on the optional attributes. The optional ALIGN justifies the text within the cells horizontally, while VALIGN justifies it vertically, with values as discussed for the <IMG> tag. The optional BGCOLOR sets the background color for the entire row, while BACKGROUND sets the URL of an image file for the background. {IE, NS3, NS4}

<TD|TH [WIDTH="n|n%"] [HEIGHT="n|n%"] [ALIGN="left|center|right"] [VALIGN="top|middle|bottom|baseline"] [NOWRAP] [COLSPAN="n"] [ROWSPAN="n"] [BGCOLOR="color"] [BACKGROUND="url"]>"objects"</TD|TH>.. This paired <TD> (table data) or <TH> (table header) tag defines a new cell within the current row and displays "objects" = "text|images|both" within the cell, according to the selected optional attributes, which override the corresponding attributes set in the <TR> tag. The <TH> tag displays text in bold. The optional WIDTH and HEIGHT set the width and height of the cell, respectively. The optional ALIGN justifies objects within the cell. The optional NOWRAP prevents automatic wrapping of objects within the cell. The optional COLSPAN allows the cell to extend over more than one column. If it is not used, the default value is 1. The optional ROWSPAN allows the cell to extend over more than one row. Again, if it is not

used, the default value is 1. The optional BGCOLOR and BACKGROUND set the background color and the URL of the background image of the cell, respectively. {IE, NS3, NS4}

Notice that although, according to HTML 3.2, the closing </TR> and </TD> tags do not have to be used (because a tag starting a new row or a new cell closes the previous row or cell), unambiguous calculation of nested tables still requires it in some browsers.

Other Table Tags.

<CAPTION [ALIGN="top|bottom"]> "text" </CAPTION>.. This paired tag displays "text" as the table header. The optional ALIGN places the caption at either the top or the bottom of the table. The <CAPTION> tag should be placed inside the <TABLE> tag, immediately after the initial <TABLE>, to associate it with the current table.

<THEAD>, <TBODY>, <TFOOT>, <COLGROUP>, <COL>.. These HTML extensions are used mostly with the FRAMES attribute in the <TABLE> tag. {IE}

Displaying Full Graphics

As indicated, many users have been attracted to the Web by its graphics and multimedia—general, its *display*—capabilities. Thus, the tag displaying graphics is quite fundamental to HTML. In this section, we shall summarize image compression formats suitable for the Web, as well as the Web-savvy palette of colors and image handling in HTML, and will also emphasize the ability of any graphic object to become a hyperlink to anything else, such as another site, another graphic object, or to a descriptive text.

Introduction to Web Graphics. Web graphics includes: (i) line graphics, (ii) pictures, and (iii) photographs. All of the graphics forms can be either black and white (*B&W*), gray scale, or multicolor. When the graphics is B&W, the file size may be small; however, when it is gray scale or color, the file sizes may be very large and compression of the files is required. Image compression may be lossless (i.e., no information is lost during the compression and decompression processes) or lossy (some information is lost during the processes). There are many image compression methods and techniques within the method. HTML accepts the image format standards, including *GIF* (Graphics Interchange Format), *JPEG* (Joint Photographic Experts Group), TIFF (Tag Image File Format), and PNG, as described next. Other formats, such as JPEG2000, or WAVELET, or FRACTAL, may require either helper applications or plug-ins to display the images.

Similarly, there are many video compression formats such as Microsoft's *AVI* (Audio/Video Interleave), which is a special case of the *RIFF* (Resource Interchange File Format); Apple's *MOV* (QuickTime format for audio, images, and video); and *MPEG* (Motion Picture Experts Group).

Image and Video Formats (20). The GIF image format was developed by CompuServe, and includes a lossless compression technique called *LZW* (Lempel–Ziv–Welch), patented by Unisys. Since GIF was intended for accurate line graphics and simple color images, it is limited to 256 different colors only. Two current versions, GIF87 and GIF89a, can be displayed by the graphics-oriented browsers. The modified GIF89a standard can be: (a) interlaced (i.e., the image is displayed progressively during its loading), (b) transparent (i.e., one of 256 colors can be made invisible to show the page background color, and thus blend with the page), and (c) animated (i.e., a series of images can be displayed in succession without any external helper application or plug-in).

The JPEG standard image format is a lossy compression technique, with a user-adjustable image quality from low to high. While the low-level quality setting produces small files, block regions appear in the image. At standard- or high-quality levels, the images appear quite good. Since JPEG allows more than 256 colors, the displayed quality of images may be better than the quality of the lossless GIF with 256 colors only. This makes it suitable for photographs and other multicolored material. In addition to JPEG, there is Progressive JPEG, capable of revealing the image gradually from very rough to detailed during its loading.

The TIFF standard is one of the most versatile bitmap formats, often used for storing and interchanging images either without compression or with numerous compression schemes, including the lossless LZW compression and lossy JPEG compression.

There are many commercial and shareware programs to translate images from various formats into the Web-oriented formats. For example, Adobe Photoshop is a sophisticated image editing program (21), Graphic Converter is a good graphics format converter and a simple editor (22), and GIF Builder can be used to produce animated GIF files (23).

Web-Savvy Palettes. As we have seen, a color can be specified in HTML either by its hexadecimal RGB value, "#rrggbb", or by a name, "colorname". Sixteen Windows VGA-palette color names are supported in most browsers. Some browsers support additional 123 other color names, as specified for the cascading stylesheets. All the color names will be displayed properly, without any dithering, on any browser. However, there is a problem when the color is specified by a hex number in that a PC may display it differently from a Macintosh computer because some colors are reserved. Since the full range of "#rrggbb" values is from #FFFFFF (for "white") to #000000 (for "black"), there are $256 \times 256 \times 256 = 16{,}777{,}216$ ("millions of") different colors in the palette. To eliminate the reserved colors and display them on all computers, the 256-color palette must be reduced to 216 "web-savvy" colors. The Web-savvy palette is formed by limiting the available colors to RGB triplets consisting of only 00, 33, 66, 99, CC, or FF. This results in the smaller number of $6 \times 6 \times 6 = 216$ colors. Many commercial and shareware programs can convert from the general to web-savvy graphics, including Photoshop (21) and Graphic Converter (22).

The Image Tag.

<IMG SRC="url" ["attributes"]>.. This singleton <IMG> (image) tag inserts an image into the current Web page, at a location specified by its occurrence with respect to other objects already displayed. The value "url" of the SRC (source) parameter specifies the address to the file containing the image. As usual, the URL may be either local (within the site) or global (anywhere on the Web). The local addressing may use either relative or absolute referencing. Notice that some browsers (e.g., Netscape Navigator) may display their internal icons referenced by their internal names; for example, <IMG SRC="internal-gopher-telnet"> will display the gopher icon.

The following optional attributes can be used within the <IMG> tag:

ALT="text" This attribute displays "text" instead of the image on the older text-only browsers. In graphics-oriented browsers, it displays "text" when the image file cannot be located. Furthermore, depending on the browser used, it displays "text" either during the image loading process, or when the cursor is moved over the displayed image.

WIDTH="*n*|*n*%" HEIGHT="*m*|*m*%" These two attributes set the width and the height of the picture to *n* and *m* pixels, respectively. The values can also be given as a percentage of the current window. Loading of the page appears to be faster when WIDTH and HEIGHT are specified.

VSPACE="*n*" HSPACE="*m*" These two attributes set VSPACE (vertical space) to *n* pixels and HSPACE (horizontal space) to *m* pixels between the image and its surrounding objects, usually text.

ALIGN="left|right|top|middle|bottom|absbottom|texttop|absmiddle|baseline" This attribute aligns the picture on the current page. The first two values position the image with respect to the current window, while the remaining values position the image with respect to the current line where the picture appears. Thus, the values "left" and "right" align the picture to the left or right side of the displayed page, respectively. The value "top" justifies the top side of the image to the top of any largest object in the current line, while the value "absbottom" justifies the bottom of the image to the bottom of any largest object on the line. The value "texttop" justifies the top of the image with the highest character on the line. The values "middle" and "absmiddle" justify the picture either to the center of the baseline of the text line, or to the center of the midline of the text, respectively. The values "baseline" and "bottom" justify the bottom of the image

on the same level as the baseline of the current text line, respectively. The last four values apply to the Netscape Navigator only.

BORDER="n" This attribute sets the width of the border around the image to n pixels. The usual values range from 1 to 3. BORDER="0" eliminates the border altogether. If the image is a hyperlink and if $n > 0$, the border appears in the colors specified for text hyperlinks.

TARGET="text" This attribute operates the same as in the anchor <A> described in the "Hyperlinks" section above.

LOWSRC="url" This attribute displays a preview of the image located at "url" that is smaller and may load faster than the full image. It is displayed before the full image loads.

ISMAP This attribute states that the image is a server-side imagemap, as described in the "Imagemaps" subsection below.

USEMAP="#anchor" This attribute states that the image located at "#anchor" is a client-side imagemap, rather than a server-side imagemap. Usually, USEMAP is used together with ISMAP so that if the browser does not supports client-side maps, ISMAP is then used instead.

DYNSRC="url" This *dynamic source* attribute assigns the "url" to a sound or a *VRML* (virtual reality modelling language) file representing panoramic or other 3D images. If the browser does not support dynamic images, the standard SRC property will be used. A sound must be in either the .au, or .wav, or .mid format. {IE}

CONTROLS This attribute tells whether a DYNSRC object should display controls or not. {IE}

LOOP="n|infinite" This attribute sets the number n of times that the DYNSRC object is to be repeated. {IE}

START="[fileopen] [; mouseover]" This attribute specifies when the DYNSRC object, such as a video, should start. The value "fileopen" starts it after the page is loaded; "mouseover" starts it when the cursor is over the object. Either a single or both values can be used. {IE}

Imagemaps. In the previous subsection, we have pointed out that any image can become a hyperlink. Imagemaps expand the idea from "a single image equals single hyperlink" to "a single image equals multiple hyperlinks" by subdividing the image into subregions, each of which is a separate hyperlink.

An older server-side type of imagemaps consists of two files: (i) a GIF image, and (ii) the corresponding map definition, written in either the CERN format or NCSA (National Center for Supercomputing Applications, located at the Urbana–Champaign campus of the University of Illinois) format, or both, and residing on the server. Since the map definition is not written in HTML, a special program (such as a CGI script) must be run on the server to interpret the map coordinates.

A newer type of imagemaps are called client-side imagemaps. Since such imagemaps are written in HTML and therefore do not require a CGI-based interpreter running on the server, they are easier to handle and are more popular. However, the browser must now support the client-side imagemaps.

Since manual generation of the coordinates of the subregion in the imagemap is simple but very tedious and error-prone, various automated programs [e.g., Mapper (24)] have been developed to accomplish the task easily.

The NCSA Imagemap Format.. The NCSA format defines various clickable, or "hot", objects, such as rectangles and circles, by defining their coordinates within an imagemap. Each object has its transfer URL, and is separated from the next by a RETURN character. The list of all the objects is ordered from the front to the back. The list is terminated by a default URL. The map is stored at the server site. The syntax of the definitions is described next.

/# "text".. A number sign at the beginning of a line makes the remaining text a comment.

rect url x_1, y_1 x_2, y_2.. This declaration defines a rectangle, whose upper left corner is located at x_1, y_1 and lower right corner at x_2, y_2.

circle url x_1, y_1 x_2, y_2.. This declaration defines a circle whose center is located at x_1, y_1 and circumference includes x_2, y_2.

oval url x_1, y_1, x_2, y_2.. This declaration defines an oval whose x_1, y_1 and x_2, y_2 coordinates describe the rectangle with the inscribed oval.

poly url x_1, y_1 ... x_N, y_N.. This declaration defines an N-sided polygon whose corners are represented by the x, y coordinates.

point url x, y.. This declaration defines a point at x, y.

default url.. This declaration defines the default URL, and is a required closing line in the definition list.

The CERN Imagemap Format. The CERN format defines various clickable, or "hot," objects, such as rectangles and circles, by defining their coordinates within an imagemap. Each object has its transfer URL, and is separated from the next by a RETURN character. The list of all the objects is ordered from the back to the front. The list starts with a default URL. Again, the map is stored at the server site. The syntax of the definitions is described next.

"*text*".. As in the NCSA format, the number sign at the beginning of a line makes the remaining text a comment.

default url.. This declaration defines the default URL, and is a required starting line in the definition list.

rect (x_1,y_1) (x_2,y_2) *url*.. The definition of a rectangle is the same as in the NCSA format, except for the enclosing parentheses and the position of the URL.

circle (x,y) rad url.. This declaration of a circle is different from the NCSA format in that the center located at (x,y) is followed by a radius, rad.

poly (x,y)...(x,y) url. The definition of a polygon is similar to the NCSA format.

The Client-Side Imagemap Format.. The client-side imagemap format defines various "hot" objects by defining their coordinates within an imagemap. The entire definition follows the position-free syntax of HTML in that the entire map is enclosed in a paired <MAP> tag, while each hot object is defined by an <AREA> tag, whose attributes can be arranged in any order, as they are defined by individual keywords. Each object has its transfer URL. The list of all the objects is not ordered. Unlike the other two formats, this map is stored in the HEAD section of the current Web page. The syntax of the definitions is described next.

<MAP NAME="mapname">...</MAP>.. This paired <MAP> tag encapsulates the imagemap definition. The anchor NAME is used by the <IMG> tag to locate the imagemap.

<AREA [SHAPE="rect|circ[le]|poly[gon]|default"] COORDS="coordinates" [NOHREF] [HREF="url"] [ALT="text1"] [TARGET="text2"]>.. This singleton tag defines an imagemap object. If SHAPE is not used, the value "rect" is assumed, and "coordinates" = "x_1,y_1,x_2,y_2" with the definition of corners as in the previous two NCSA and CERN formats. Otherwise, if the value "circ" is used, "coordinates" = "center_x,center_y,radius", and if the value "poly" is used, "coordinates" = "x_1,y_1,x_2,y_2,...". The last value "default" can be used not to set a hot object, but to set the default URL of the map through the HREF. As in the <IMG> tag, HREF sets the URL to which this <AREA> links. On the other hand, NOHREF indicates that no action should be taken, thus defining a "dead" area in the imagemap. The attributes ALT and TARGET behave the same as in the <IMG> tag.

Other HTML Facilities

The preceding six sections describe the basic elements of the HTML page construct. This section (i) summarizes several newer constructs that expand the capabilities of a single page, and (ii) addresses some issues related to both the design of the entire site and the appearance of each page within the site.

Frames. Standard Web pages are displayed within the entire window. Very often, one would like to preserve a part of the page, such as a table of contents or a banner, when other pages are displayed. This

idea has lead to the FRAMES construct. Frames are parts of the displayed window, each of which can be handled independently. This requires that the subdivision of the window be done first through a special tag (<FRAMESET>), encapsulating the definitions of the frames through a special tag (<FRAME>), each fitting into the specific subdivision in the window, as described next. The <FRAMESET> is usually followed by the <NOFRAMES> tag in order to inform users with browsers that do not understand frames that the frame was ignored. Frames were initially introduced in Netscape Navigator 2.0, but their popularity resulted in incorporating them into the HTML standard.

Regular Frames.

<FRAMESET COLS="sizes"|ROWS="sizes" [attributes]>"list_of_frames"</FRAMESET>.. This paired <FRAMESET> tag defines the composition of frames (i.e., "list_of_frames") within the screen window. The attributes COLS and ROWS specify how the window should be split into either columns or rows. One <FRAMESET> tag cannot have both ROWS and COLS, although more than one <FRAMESET> tag can be used in a Web page. The COLS and ROWS have values, "sizes", expressed as "sizes"="size1,size2,. . .,size*N*", which is a comma-delimited list of widths (for COLS) or heights (for ROWS) of the individual *N* frames that should appear in those columns or rows. Each size can be either absolute in pixels, or relative in percent of the page (*n*%), or automatic (∗), which fills the remaining part of the displayed window. For example, if the window is 11 inches, then COLS="144, 288,∗" sets three columns of sizes 2, 4, and 5 inches. Notice that since the <FRAMESET> tag provides a structural metadefinition only, it should be placed within the HEAD tag. In fact, there should be no <BODY> in the file containing the <FRAMESET>.

The optional attributes include:

[BORDER="*n*"] [FRAMESPACING="*n*"] These attributes set the border or frame spacing to *n* pixels.

[BORDERCOLOR="color"] This attribute sets the color of the border.

[FRAMEBORDER="yes|no"] This attributes either displays or hides the frame dividers.

The "list_of_frames" consists of the required number of frames, each defined by the following <FRAME> tag.

<FRAME NAME="name" SRC="url" [attributes]>.. This singleton tag defines a frame within the <FRAMESET> tag. The NAME attributes sets the name of the frame. If, for example, NAME="toc" and a hyperlink in another frame contains TARGET="toc", the hyperlinked page is loaded into the "toc" frame. Since this frame-setting file does not have a BODY section, the URL is required to establish a link to the actual page that has the contents to be displayed in this frame.

SCROLLING="yes|no|auto".. This attribute sets scroll bars in the frame ("yes"). If “no” is used, the contents must be guaranteed to be displayed within the frame. If "auto" is selected, scroll bars are displayed only if they are needed, or parts of the content will be hidden outside the displayed area.

MARGINWIDTH="n" MARGINHEIGHT="m".. These two margin attributes set the space between the frame border and the page content to *n* and *m* pixels.

NORESIZE.. This attribute inhibits resizing the current frame by moving its borders. This is useful to preserve the intended proportions of the frame.

BORDER="n" FRAMESPACING="m" BORDERCOLOR="color" FRAMEBORDER="yes|no".. These attributes have the same meaning as in the <FRAMESET> tag, except they now apply to the current frame only.

TARGET="frame_name".. As described in the hyperlink section, this attribute directs a linked page to the specified target frame (either the same, or different, or new).

<NOFRAMES>"text_and_images"</NOFRAMES>.. This paired tag is required because not all browsers display frames. Any text and images contained within this tag are displayed in order to inform the user about the problem.

Floating Frames. Defined in HTML 4, a floating frame can be placed anywhere in the current window. Such frames are often used for messages, advertising, and other similar purposes.

<IFRAME NAME="text" SRC="url" WIDTH="n" HEIGHT="m" [attributes]>"text"</IFRAME>.. This paired tag defines a floating frame. The meaning of NAME and URL is the same as in the regular <FRAME>. The attributes WIDTH and HEIGHT set the width and height of the frame to n and m pixels, respectively. In browsers that do not support floating frames, "text" is displayed. The optional attributes include:

SCROLLING="yes|no|auto" The meaning of this attribute is the same as in the <FRAME> tag.

HSPACE="n" VSPACE="n" These attributes set the horizontal and vertical space to objects within the floating frame as in the <IMG> tag.

ALIGN="left|right|top|middle|bottom|texttop|absmiddle|baseline|absbottom" The meaning of this attribute is the same as in the <IMG> tag.

BORDER="n" BORDERCOLOR="color" FRAMEBORDER="yes|no" The meaning of these attributes is the same as those in the <FRAMESET> tag.

Problems with Frames. There are two groups of problems with frames: one is related to the way frames operate, and the other is related to the design of frames.

A major problem is that not all browsers understand how to display frames. Developing both frame-based and non-frame-based sites is expensive.

Another problem is with retracing the visited frames, as they do not appear in the history window. Explicit internal links, or a multiple use of the return icon, solves the problem.

Printing of a given frame requires that it be selected first. Otherwise, another frame is printed.

In practice, browsers run on machines with different screen sizes and different resolutions. An incorrectly designed frame set may be displayed as intended only on the screen of its designer, and be incomprehensible on any other screen. This may occur if the sizes of the columns and rows are expressed in percentage of the screen. Consequently, a frame with size-critical contents (e.g., table of contents) should have a fixed size, while other frames can use variable widths (either "$*$" or "$n\%$").

A selected frame is displayed with a colored border on Netscape Navigator 3 or earlier on the Macintosh. In version 4 or later, this border can be removed by using borderless frames. However, borderless frames in Netscape Navigator have visible borders in Internet Explorer, and vice versa. Clearly, a solution to this problem is to remove all the borders, if that is acceptable.

Java Applets. Applets are small programs, written in the platform-independent Java programming language, capable of performing special tasks on a Java-enabled browsers that could not be performed by HTML alone, such as visualization of dynamic and interactive data. One or more applets can be embedded into a Web page. An applet is usually small in size and fast in execution because it is loaded from the server to the client and then executed by the client during the current Web-page processing. As a mini-application, an applet is different from an application in that it does not have to have an open window, and that it cannot perform input/output (I/O) operations on the client's machine, for security reasons.

This subsection is a short summary of some HTML tags related to Java applets. There are many resources on Java (25) and Java applets (26).

Applet Definition.

<APPLET CODE="url" HEIGHT="n|n%" WIDTH="n|n%" [attributes]> "parameter_list" </APPLET>.. Within the current page, this paired <APPLET> tag invokes an applet located at "url" and passes "parameter_list" to the applet for execution. The CODE attribute gives a URL to the code of the actual applet. The required HEIGHT and WIDTH define the height and width of the applet embedded within the corresponding page window.

The following optional attributes can be used within the tag:

[CODEBASE="url"] This attribute specifies the URL of the supporting class libraries required by the applet. It can be omitted if the libraries reside in the same directory as the applet itself.

[ALIGN="left|right|top|middle|bottom"] This attribute defines the alignment of the applet within the window. The first two values place the applet at the respective margins of the window, and float text around it. The remaining values align the applet with respect to the surrounding text.

[HSPACE="n" VSPACE="n"] Similarly to the <IMG> tag, these attributes define the horizontal and vertical extra padding space in pixels to the left and right (HSPACE) and at the top and bottom (VSPACE) of the applet window. {IE}

<PARAM NAME="name" [VALUE="value"]> This singleton tag assigns a value to a variable "name" which is sent to the current applet. The value can be either a number n, or a list of N space-separated numbers "n_1 $n_2 \ldots n_N$", or an URL to an image, or a string. More than one <PARAM> can be used within "parameter_list".

In the future, the <APPLET> tag may be replaced by the more flexible <OBJECT> element, which can embed not only applets, but also many other elements such as images and simple data files.

Embedding Objects into a Page. In addition to the GIF and JPEG images <IMG> and applets (<APPLET>) just discussed, many other objects can be embedded into the Web page, including images with arbitrary formats [e.g., Adobe PostScript files, or JPEG2000 files, or computer-aided design (*CAD*) files], spreadsheets (e.g., Microsoft's Excel), sound (MIDI music sequences), and video (Apple QuickTime, or Microsoft AVI movies, or VRML world scenes), using the <EMBED> and <OBJECT> tags. Clearly, since handling of such objects may not be understood by the browser, special platform- and browser-specific application software that is plugged directly into the browser (also known as plug-ins) may be required. The type of file is determined by the browser from the *MIME* (multipurpose Internet mail extensions) content-type header transmitted by the server before the actual file is delivered, using the HTML protocol. The content-type header has the following format: content-type: type/subtype, where the type indicates the type of the file transmitted, such as image, audio, text, video, application, multipart, message, and extension token, while the subtype gives the specifics of the data contained in the file, such as zip or x-pdf. For example, a standard HTML file with extensions .html or .htm would have content-type: text/html (4, Appendix B1; 6, Web site).

The EMBED Element.

<EMBED SRC="url" | TYPE="mime-type" WIDTH="n" HEIGHT="n" [attributes] [. . .]>.. This singleton tag includes special nonstandard objects into the current page. The required SRC or TYPE specifies either the URL to the object to be embedded into the page (which provides the MIME type to the browser by the HTTP protocol), or the MIME type for the embedded object directly, so that the browser can select the required plug-in for its processing. The required WIDTH and HEIGHT attributes specify the size of the embedded object in pixels.

The optional attributes include:

NAME="name" This attribute specifies a symbolic name for the embedded object in order to reference it from other objects, scripts, or applets.

ALIGN="left|right|absbottom|absmiddle|baseline|top|middle|bottom" This attribute specifies how the embedded object is to be displayed, with values similar to the <IMG> tag.

HIDDEN="true|false" This attribute specifies the visibility of the embedded object. If it is "true", the object is not visible. If it is "false", it is displayed, and WIDTH and HEIGHT must be used.

<NOEMBED>"text"</NOEMBED> If a browser does not support plug-ins, "text" is displayed. A good idea would be to put a link to the embedded file here.

Notice that, since arbitrary attribute names are not allowed in the syntax of SGML, the <EMBED> tag is not part of the HTML standard. Consequently, as with the <APPLET> tag, <EMBED> may be replaced by the more versatile <OBJECT> element.

The OBJECT Element {IE4, NS4}.. This new element is designed to replace the other embedding elements: IMG, IFRAME, EMBED, and APPLET. Parameters can be specified by the PARAM element and passed to the object during its run time. In addition, and <OBJECT> can also contain the older EMBED and APPLET embedding elements so that browsers that do not understand the <OBJECT> tag can execute these older tags. If the browser understands the OBJECT embedding, the older tags are ignored.

<OBJECT DATA|CLASSID="url" TYPE="mime-type" WIDTH="n" HEIGHT="n" [attributes]>.. Unlike the <EMBED> element, both DATA (or CLASSID) and TYPE attributes are required to provide the browser with more flexibility. For example, if the browser cannot process the MIME type, it will not retrieve the data file at "url", thus speeding up its responses. CLASSID also specifies the URL of the program code (object). If CLASSID is not used, TYPE selects an appropriate data handler. As before, WIDTH and HEIGHT define the size of the displayed object. There are many attributes that the <OBJECT> tag can have, depending on the type of object handled (6).

<PARAM NAME="name" [VALUE="string" VALUETYPE="data|object|ref" TYPE="mime-type"]>.. As in the <APPLET> tag, this singleton tag is used to send parameters to the current object. The new VALUETYPE specifies the three types of value: either "data", which treats "string" as input data for the object, or "object", which treat "string" as the name of another object, or "ref", which treats "string" as a URL.

There are many <OBJECT> implementation bugs in the current browsers.

Layers. Recall that floating images or applets forced the surrounding text to flow around them. In contrast, *layers* appear to float above (overlap) the text or other objects on the displayed page window. Layers may be nested, thus overlapping the lower layers. Layers can be manipulated with scripting languages such as JavaScript, thus making the layers visible or invisible, or movable from one position to another in the x–y plane, or movable in the z direction. Similarly to <NOFRAMES>, the <NOLAYERS> tag was also introduced for browsers that do not understand layers. Originally, the idea was implemented in Netscape Navigator 4. Although this element is being replaced by style sheets, the basic ideas can be explained easily through this tag.

<LAYER WIDTH="n|n%" HEIGHT="n|n%" [attributes]>...</LAYER>.. This paired tag defines a new layer in the z direction. The preferred width and height of the layer, in pixels or in percent of the parent layer, is set by WIDTH and HEIGHT, respectively.

The following optional attributes can be used: {NS4}

NAME="layer_ID" This attribute gives a name to the layer, so that external languages such as JavaScript can manipulate it (hiding, revealing, moving).

TOP="*n*|*n*%" LEFT="*m*|*m*%" | PAGEY="*n*" PAGEX="*m*" These attributes set the top left position of the layer, in pixels or percentage, counted from the top left of either the displayed window, or the parent layer, if nested. If <TOP> and <LEFT> are not used, the layer appears after the previous object. An alternative pair of positioning coordinates is PAGEY and PAGEX, expressed in pixels only, and always measured from the parent page window.

Z-INDEX="*n*" | ABOVE="Layer_ID" | BELOW="layer_ID" These attributes define the absolute stacking order of the layers. Only one of these attributes can be selected. Z-INDEX sets the position of the layer in the z direction, with the lowest number specifying the lowest layer, while the largest number specifies the top layer. The other two attributes, ABOVE and BELOW, define the stacking order relative to an already created layer whose name is "layer_ID".

VISIBILITY="show|hidden|inherit" This attribute tells whether the layer content is visible ("show") and hides the layers below it, or is invisible ("hidden"), or takes ("inherit") the visibility of the parent layer. By default, it is set to "show".

BGCOLOR="color" This attribute defines the background color of the layer, which make the layer opaque (i.e., the underlying layer is invisible).

BACKGROUND="url" This attribute defines the URL of an image that forms the background of the layer, thus overriding "color" as set by BGCOLOR. The background image can be transparent.

CLIP="[x_1,y_1,]x_2,y_2" This attribute sets the border of a rectangular clipping box within the current layer. The area inside the clipping box obscures the layer below it, while the area outside the clipping box is transparent. The values x and y are given in pixels, measured from the left and top edges of the current layer, respectively, with x_1 defining the left edge of the clipping box, y_1 its top edge, x_2 the right edge, and y_2 the bottom edge. If x_1 and y_1 are not provided, they are assumed to be 0.

<NOLAYER>"text" </NOLAYER> If the browser does not support layers, "text" is displayed.

Style Sheets. The proprietary specification of layers has now been generalized to the so-called style-sheet approach, and is now considered the recommended mechanism for element positioning and z- indexing above the layer of the displayed Web page. The concept of style sheets is very important, because it separates the advantageous semantic nature of HTML (i.e., the markup in HTML is intended to describe the meaning and structure of a Web document) from the physical presentation of the document on a computer screen or on a printed page. A style sheet defines how the document is displayed, in a compact form, usually in a single separate file, possibly affecting all the Web pages within the Web site. This is in contrast with the limited capabilities of HTML (e.g., FONT, ALIGN, BGCOLOR) to control individual details locally in each page separately. That makes the pages much larger than necessary, and very difficult to modify.

A simple style-sheet language, called Cascading Style Sheets (*CSS*), has been developed to implement the style-sheet concept. The name "cascading" originates from the passing on (cascading) of the style "inheritance" from one object to another object nested within it. Its first version, CSS1, was introduced by the W3C in December 1996. This standard has been implemented either partially [in the current versions of Netscape and Internet Explorer] or fully [in other browsers such as Opera (27) and Mozilla (28)]. An extension of CSS1 was recently introduced as CSS2 in order to provide much greater control over the positioning of displayed objects, production of sound, and new displays. Another extension is being discussed as CSS3. The CSS language and its use have been described in many sources (e.g., Refs. 29,30,31).

There are three schemes to describe the style of elements within a page: (i) local to a page, (ii) global, linked to from any page, and (iii) local, affecting formatting of an element within a page, as summarized next.

Page Style Sheets.

<STYLE TYPE="mime/type"><!– style sheet –></STYLE>.. This paired tag specifies the formatting of the selected elements within the current page. It must appear within the HEAD section of the page. The TYPE attribute is the MIME type of the style, and must be "text/css" for the CSS. When other style-sheet languages are developed, they can be selected by TYPE accordingly.

The actual style sheet is encapsulated not only between the opening and closing <STYLE> tags, but also between the delimiters of a comment block so that browsers that do not understand style sheets can ignore them. The syntax of the style sheet is

$$<!--\ \text{tagname}[.\text{ext}]\{\text{attributes}\}\ -->$$

The tagname is the name of the tag or tags within the current page that will use the style-sheet definitions associated with the tag, such as BODY, FONT, A:link. The scope of the influence of this definition can be reduced to a class of objects, if the optional name extension, .ext, is appended to the tagname, and when the CLASS is added to the tags themselves. The {attributes} have the following format: {attrib1: value1[; attrib2: value2 [. . .]]}, where each attrib acquires a corresponding value. An attribute is separated from its value by a

colon, while different attributes are separated by semicolons. The following attributes in CSS1 are designed to extend the flexibility of styles, far beyond that provided by FONT:

font-family: font_list [, serif|sans-serif|cursive|fantasy|monoscape] This attribute defines a font family, scanned from left to right for the first availability to the browser. If none of the fonts in the family is available, the optional generic font is used.

font-size:*n* pt|px|pc|ex|em|in|mm|cm|%|xx-small|x-small|small|medium|large|x-large|xx-large The font size is given either by a number, *n*—expressed in units (such as points, pixels, picas, x-spaces, m-lengths, inches, millimeters, and centimeters) or as a percentage of the 12 pt size—or by a variable. The units are related by 12 pt = 12 px = 1 pc = 2 ex = 1 em = 0.17 in = 6.3 mm = 0.63 cm = 100% = medium.

text-align: left|center|right|justify This attribute defines the alignment of the text.

font-weight: *n*|demi-light|extra-light|lighter|light|medium|bold|bolder|extra-bold|demi-bold This attribute defines the weight (boldness) of the displayed font, given either by a number (in the range of 100 to 900) or by a variable.

font-style: normal|italic|oblique This attribute defines the style of the text.

text-decoration: [underline] [,overline] [,line-through] [,blink] This attribute specifies a comma-separated list of text styles.

line-height: *n* pt|px|pc|em|in|mm|cm|% This attribute defines the space between two text baselines.

text-indent: *n* pt|px|pc|em|in|mm|cm|% This attribute defines the indentation of the first character.

color: value|colorname This attribute sets the color of the text. There are 139 Web-savvy legal CSS colornames (30) that should appear the same on all the platforms, while colors specified by their RGB values may appear differently on different platforms.

background: url(url_value)|color This attribute sets the background, either to an image retrieved from its url_value or to a specified color.

margin-top: *n* pt|px|pc|em|in|mm|cm|% This attribute sets the top margin.

margin-left: *n* pt|px|pc|em|in|mm|cm|% This attribute sets the left margin.

margin-bottom: *n* pt|px|pc|em|in|mm|cm|% This attribute sets the bottom margin.

margin-right: *n* pt|px|pc|em|in|mm|cm|% This attribute sets the right margin.

Global Linked Style Sheets.

<LINK REL="stylesheet" TYPE="mime/type" HREF="url">.. This singleton tag must appear within the HEAD section of a page, and replaces the explicit <STYLE> tag, as described in the previous sub-subsection. An external file, located at "url", is linked to provide a specific style sheet description for the page. The URL has a .css suffix to distinguish it from HTML files. TYPE is the mime/type and must be "text/css" for CSS. Since this single style sheet can be linked from many pages, this approach reduces the size of the HTML pages, and allows global changes to the entire site by changing the style sheet alone.

Local Object Style Sheets.

STYLE="attrib1: value1 [;attrib2: value2 [. . .]]".. This is the simplest way to provide a style to an object defined by almost any paired tag. For example, <P STYLE="font-weight: bold; color: "red">"text"</P> displays "text" in red and bold. This local style is often used to override any previous setting for this object.

Scripting in HTML. We have seen various attempts to make HTML not only more flexible in terms of page styles, but also dynamic and interactive (e.g., FORMS). Such interactivity (passing of parameters either from the Web page to the program for processing, or from the program to the Web page for display) can be accomplished by interfacing the standard HTML documents to special programs. The interfacing can be done

either by embedding the program within the current page, or linking the page to an external program in a manner similar to embedding style sheets.

Netscape introduced the concept of scripted HTML in their Navigator 2 by developing a scripting language, JavaScript (32), which is derived from object-oriented languages like Java and C++. Its object model is related to both the component structure of the browser (such as URLs, windows, and status bars) and the elements of HTML documents (such as applets, embedded objects, and forms). Microsoft also implemented a similar JScript in their Internet Explorer 3, as well as VBScript related to their Visual Basic (33). The emergence of the next generation of browsers from the two groups produced convergence in their functionality, but also divergence because of many new incompatible extensions, commonly known as Dynamic-HTML or *DHTML* (34). A fuller coverage of scripting would require another chapter.

Future Trends

In the previous section, we have looked at some shortcomings of the original HTML and interesting solutions to overcome them, including dynamic and interactive functionality. This section looks at further shortcomings of HTML and current attempts to solve them incrementally or radically.

HTML Mathematics: Problem and Solutions. One of the major shortcomings of HTML is its inability to handle mathematical symbols, variables, expressions, equations, their numbering, and I/O interaction. A very cumbersome scheme to represent mathematical constructs is to produce images of the mathematical constructs and embed them in the text.

Another solution is to develop a mathematics-oriented document using a typesetting–document-formatting language such as $\mathrm{L^AT_EX}$ or the American Mathematical Society (AMS) enhancement, called $\mathrm{A_MS}$-$\mathrm{L^AT_EX}$, then translate it into HTML, using a program such as latex2html, which produces all the GIF images automatically.

A third solution is to use a browser plug-in (such as IBM's techexplorer) that understands $\mathrm{L^AT_EX}$ documents and displays them directly within the browser's window.

The above three approaches to solving the mathematics limitations in HTML were poor fixes to the problem. A more radical solution is to abandon HTML and develop a new mathematically parsable markup language that understands the meaning of mathematical expressions so that they can be not only displayed, but used in other mathematics-oriented programs such as Maple, Mathematica, and MatLab. Much work is being directed towards developing a mathematical markup language (MathML) (35).

HTML Page Banners: Problem and Solutions. Nonscrolling headers, footers, and sidebars can be implemented using FRAMES or other facilities, but they are difficult to manage and print. In HTML 3, several attempts were made to ease this problem, but the proposed changes have not been implemented in HTML 4. This appears to be urgent, as e-commerce uses such constructs extensively.

HTML Inflexibility and Portability: Problem and Solutions. As with many standards, approval and implementation of any changes in HTML requires years to complete. As we have seen with style sheets and scripting, the problem is even more exacerbated by the inconsistent implementation of such changes in various browsers. Furthermore, by the year 2002 as much as 75% of Internet access may be carried out on non-PC platforms, such as palm computers, televisions, refrigerators, automobiles, and telephones. Usually, those machines do not have the computing power of a desktop computer, and will not be able to cope with the portability of a badly implemented or proprietary HTML.

Again, a radical solution is to abandon the fixed tag set HTML, and develop a new customizable markup language instead. In fact, a simple Extensible Markup Language (*XML*) has been developed with no fixed set of tags, but a scheme to develop custom elements and their attributes to define any desirable document structure (36,37,38,39). The XML document can be processed on any software with an XML parser. There are many dialects of XML, including the already mentioned MathML, the Extensible HTML (*XHTML*) (40),

and Synchronized Multimedia Integration Language (*SMIL*) (15). XHTML is of particular interest in that it is a reformulation of HTML 4.0 as an application of XML 1.0. So those familiar with HTML 4 will be able to program in XHTML 1.0.

New related concepts have also been developed, including the Simple *API* (Application Program Interface) for XML (*SAX*), which is a standard interface for event-based XML parsing, and Document Object Model (*DOM*) programming interfaces, using the XSL Transforming (*XSLT*) language, which transforms XML documents into other XML documents. More precisely, the XML Stylesheet Language (*XSL*) specifies the styling of an XML document by using XSLT to describe how the document is transformed into another XML document. SAX promotes modular design of XML systems, since it places no requirement on parsers to provide particular data structures (such as the DOM) representing XML documents. Since parsers plug into the API, using a driver style API, different parsers can be used for different application environments. An event-based API reports parsing events (such as the start and end of elements) directly to the application through callbacks, and does not usually build an internal tree. Consequently, such an event-based API provides simpler, lower-level access to an XML document, thus allowing parsing documents that are much larger than the available system memory, as well as constructing custom data structures, using custom callback event handlers. SAX and DOM are the two most common interfaces for programming XML.

Closing Remarks. The Web has been evolving in its technologies, protocols, and data formats from stylistic formatting to structural markup to semantic markup. In the last domain, we have moved from PostScript (opaque, operational, formatting) to troff (readable, operational, formatting) to Rich Text Format (*RTF*) (readable, extensible, formatting) to HTML (readable, declarative, limited descriptive semantics like <ADDRESS>). We have recently advanced to XML and its permutations, and are moving on to intelligent metadata formats such as the Platform for Internet Content Selection (*PICS*) labels. The PICS specification enables labels (metadata) to be associated with Internet content. It was originally designed to help parents and teachers control what children access on the Internet, but it also facilitates other uses for labels, including code signing and privacy.

The Web itself is becoming a kind of cyborg (human and machine) intelligence, harnessed together to generate and manipulate information. As indicated by MIT Laboratory for Computer Science Director, Michael Dertouzos (41), if automatability is to be a human right, then machine assistance must eliminate the drudge work involved in exchanging and manipulating knowledge. As described by Douglas Adams (42), the shift from structural HTML markup to semantic XML markup is a critical phase in the struggle to transform the Web from a global information space into a universal knowledge network.

Appendix 1. HTML Tags by Categories

This appendix lists all the tags and their functionalities, as grouped and discussed in this chapter. The list can serve as a quick overview of the available main functions in the design of a site. For brevity, the corresponding attributes are not described in detail. The following ten major groups of tags are often used in the design of pages for Web sites: (1) page structure, (2) hyperlinks, (3) textual contents, (4) forms as interactive data, (5) simple graphics as separators and spacers, (6) tables as structural 2D composition of objects, (7) displaying full graphics, (8) frames, (9) applets and objects, and finally (10) layers and style sheets. If a group is very large, it is split into several subgroups for clarity.

Page Structure.

Fundamentals.

<!– "text" –> A comment, ignored by the HTML parser in any browser.

<!DOCTYPE [. . .]> Declares the type and content format of a Web page. The [. . .] construct signifies optional attributes needed by the declaration.

<HTML> The beginning of the page contents. Requires a matching closing tag </HTML>.

<HEAD>...</HEAD> Delineates the description of the page; not directly displayed in a browser.
<BODY>...</BODY> Delineates the description of the page; usually displayed by a browser.
</HTML> Defines the end of the page contents. Requires a matching starting tag <HTML>.

Header Properties.

<TITLE>"text"</TITLE> Sets the title of the Web page to "text".
<BASE HREF="url"> An optional tag to set the default URL for the page, useful for relative addressing.
<BASE TARGET="target"> Changes the default current-page target display window to another new display window.
<BASEFONT SIZE="*n*" [COLOR="color"] [FACE="font"]> This optional tag sets the default size of the default font. It can also include default color and face.
<BGSOUND SRC="url" [LOOP="*n*|infinite"]> Points to the source of a sound to be played in the background and the number of repeats desired.
<ISINDEX HREF="url" [PROMPT="text"]> Informs the browser that the current Web page is a searchable index.
<LINK [REL="text" [REV="text"] [TITLE="text"] [HREF="url"]> Defines forward/backward relations between Web pages.
<STYLE>...</STYLE> Defines global style sheets.
<SCRIPT LANGUAGE="language" [SRC="url"]> "actual code here"
</SCRIPT> Informs a browser that the enclosed text is a code to be executed.
<NOSCRIPT>"text"</NOSCRIPT> Displays "text" in browsers that do not support the SCRIPT tag.

META Tags Related to the HTTP Protocol.

<META HTTP-EQUIV="content-type" CONTENT="mime-type [; CHARSET=charset]"> Defines content using a MIME type other than the server's default
<META HTTP-EQUIV="text" CONTENT="text"> A replacement of the header sent by the server software.
<META HTTP-EQUIV="refresh" CONTENT="*n* [; URL=url]"> Requests a reload of the current Web page every *n*th second.
<META HTTP-EQUIV="expires" CONTENT="day, date year time timezone"> Specifies when the current page should be removed from caches.

META Tags Related to Page Properties.

<META NAME="generator" CONTENT="text"> Indicates which program has been used to create the Web page.
<META NAME="description" CONTENT="text"> Provides a search engine with a short description.
<META NAME="keywords" CONTENT="word1, word2, ..., word*N*"> Provides a search engine with the keywords.
<META NAME="distribution" CONTENT="global|local"> Defines the page either as an index page (global) or not (local).
<META NAME="author" CONTENT="author's name"> Specifies the author of the Web page.
<META NAME="copyright" CONTENT="text"> Provides a search engine with a copyright notice.

<META NAME="robots" CONTENT="all|none|nofollow|noindex"> Informs a search engine which pages should be indexed.

<META NAME="language" CONTENT="language"> Informs a search engine what natural language is used.

Tags Defining Properties of the BODY Section.

<BODY [. . .]> Defines the attributes of a page such as background and background texture, the color of the text, and the sizes of margins.

Hyperlinks.

<A HREF="[url][#section]" [TARGET="target_name"]>. . .</A> Defines the anchor (A) of a hyperreference (HREF), and optionally a target window code for display where TARGET="_self|_top|_parent|_blank".

Textual Contents.

Paragraph Management.

<H*n* [ALIGN="left|right|center"]>"text"</H*n*> Displays "text" in bold, with the size *n* ranging from 1 to 6

<P [ALIGN="left|right|center|justify"]>. . .[</P>] Defines a paragraph, separated from another paragraph by a blank.

<DIV ALIGN="left|center|right|justify">. . .</DIV> Aligns more than one paragraph.

<CENTER>. . ..</CENTER> Centers more than one paragraph.

<BR [CLEAR="left|right|all"]> Inserts a line break.

<NOBR>"text"</NOBR> Forces "text" to be displayed on a single line.

<WBR> Marks where a long word may be broken between two lines.

<MULTICOL COLS="*n*" [GUTTER="*m*"] [WIDTH="*k*"]>"text"</MULTICOL> Displays "text" in columns.

List Management.

<UL [TYPE="disc|circle|square"]>"list of items"</UL> Displays all the list items with unordered bullets.

<LI [TYPE="disc|circle|square"]>"text"[</LI>] Displays "text", starting with the selected unordered bullet.

<OL [TYPE="1|A|a|I|i"] [START="*n*"]>"list of items"</OL> Displays list items in an ordered arrangement.

<LI [TYPE="1|A|a|I|i"] [VALUE="*n*"]>"text"[</LI>] Displays "text", preceded by the selected order.

<DL>"list of items"</DL> Displays all the list items as definitions.

<DT>"text"[</DT>] Displays "text" as a left-justified definition title.

<DD>"text"[</DD>] Displays "text" as a left justified definition data.

Altering Text Styles and Displayed Behavior.

Physical Styles.

<B>"text"</B> Displays the enclosed "text" in bold.

<I>"text"</I> Displays the enclosed "text" in italics.

<U>"text"</U> Displays the enclosed "text" as underlined.

<TT>"text"</TT> Displays the enclosed "text" in a monotype.

<S[TRIKE]>"text"</S[TRIKE]> Displays the enclosed "text" in the strikethrough style
^{"text"} Displays the enclosed "text" as superscript.
_{"text"} Displays the enclosed "text" as subscript.

Logical Styles.

<EM>"text"</EM> Displays the enclosed "text" as emphasized text, usually as italics.
<STRONG>"text"</STRONG> Displays "text" as strongly emphasized (bold) text.
<DFN>"text"</DFN> Displays "text" as either italics or bold italics.
<ADDRESS>"text"</ADDRESS> Displays "text" in italics.
<CITE>"text"</CITE> Displays "text" as indented text (citation).
<BLOCKQUOTE>"text"</BLOCKQUOTE> Displays the enclosed "text" as indented block text.
<PRE [WIDTH="*n*"]>"text"</PRE> Displays "text" exactly as it appears in the HTML code.
<CODE>"text"</CODE> Displays "text" as a monospace typewriter font
<KBD>"text"</KBD> Displays "text" as a typewriter font to mimic keyboard entry.
<SAMP>"text"</SAMP> Displays "text" as a typewriter font to mimic a computer output.
<VAR>"text"</VAR> Defines "text" as a variable, usually rendered in italics.
<SPAN STYLE="css">"text"</SPAN> Defines the cascaded style-sheet formatting for the page.

Modification of Text Attributes.

<FONT [SIZE="*n*|+*n*|−*n*"] [COLOR="color"] [FACE="fonts"]>"text"</FONT> Modifies the size, color, and face of "text".
<BIG>"text"</BIG> Increases the size of "text".
<SMALL>"text"</SMALL> Decreases the size of "text".
<MARQUEE ["attributes"]>"text"</MARQUEE> Makes "text" autoscrolling with various attributes.

Forms as Interactive Data.

<FORM METHOD="post|get" ACTION="url" [ENCTYPE="enctype"] [NAME="text"] [TARGET="target"]>"form objects"</FORM> Displays a new form in the current page, as defined by the "form objects".
<INPUT NAME="name" TYPE="set of values" ["attributes"]> Specifies a form object, such as a field.
<TEXTAREA NAME="name" COLS="*n*" ROWS="*m*" [WRAP="off|soft|hard"]>"text"</TEXTAREA> Creates a scrollable text field with columns and rows.
<SELECT NAME="name" [SIZE="*n*"] [MULTIPLE]>"options list"</SELECT> Creates either a pop-up menu, or a full list of items (options).
<OPTION [VALUE="value"] [SELECTED]>"text" Defines a single menu item displayed in the popup menu.

Simple Graphics as Separators and Spacers.

<HR [attributes]> This horizontal-rule tag inserts a thin line with a shade.
<HR SIZE="*n*"> Sets the height of the line to *n* pixels.
<HR WIDTH="*n*|*n*%"> Sets the width of the line in pixels or to a percentage of the width of the screen.

<HR COLOR="color"> Sets the color of the line.

<HR ALIGN="left|right|center"> Aligns the line.

<HR NOSHADE> Changes the appearance of the line from the default 3D to a 2D line.

<SPACER TYPE="horizontal|vertical|block" [SIZE="*n*"] [HEIGHT="*n*|*n*%" WIDTH="*n*|*n*%" [ALIGN="left|right"]]> Inserts an empty space that separates objects.

Tables as Structural 2D Composition of Objects.

<TABLE ["attributes"]>...</TABLE> Defines a table with many optional attributes.

<TR [ALIGN="left|center|right"] [VALIGN="top|middle|bottom|baseline"] [BGCOLOR="color"] [BACKGROUND="url"]>"list of cells"</TR> Defines a new row within the table and displays any text within all the cells, depending on the optional attributes.

<TD|TH [WIDTH="*n*|*n*%"] [HEIGHT="*n*|*n*%"] [ALIGN="left|center|right"] [VALIGN="top|middle|bottom|baseline"] [NOWRAP] [COLSPAN="*n*"] [ROWSPAN="*n*"] [BGCOLOR="color"] [BACKGROUND="url"]>"objects"</TD|TH> This table data or table header tag defines a new cell within the current row.

<CAPTION [ALIGN="top|bottom"]>"text"</CAPTION> Displays "text" as the table header.

<THEAD>, <TBODY>, <TFOOT>, <COLGROUP>, <COL> Used mostly with FRAMES.

Displaying Full Graphics.

<IMG SRC="url" ["attributes"]> Insert an image into the current Web page, at a location specified by its occurrence with respect to other objects already displayed. The image may become a hyperlink.

Imagemaps Imagemaps expand the idea from "a single image equals a single hyperlink" to "a single image equals multiple hyperlinks" by subdividing the image into subregions, each of which is a separate hyperlink. There are NCSA and CERN Imagemap formats.

<MAP NAME="mapname">...</MAP> Encapsulates a client-side imagemap definition.

<AREA [SHAPE="rect|circ[le]|poly [gon]|default"] COORDS="coordinates" [NOHREF] [HREF="url"] [ALT="text1"] [TARGET="text2"]> Defines an imagemap object.

Frames.

<FRAMESET COLS="sizes"|ROWS="sizes" [attributes]>"list_of_frames"</FRAMESET> Defines the composition of frames (i.e., "list_of_frames") within a screen window, as controlled by many attributes.

<IFRAME NAME="text" SRC="url" WIDTH="*n*" HEIGHT="*m*" [attributes]>"text"</IFRAME> Defines a floating frame.

Applets and Objects.

<APPLET CODE="url" HEIGHT="*n*|*n*%" WIDTH="*n*|*n*%" [attributes]> "parameter_list"</APPLET> Invokes an applet located at "url" and passes "parameter_list" to the applet for execution.

<PARAM NAME="name" [VALUE="value"]> Assigns a value to a variable "name", which is sent to the current applet.

<EMBED SRC="url"|TYPE="mime-type" WIDTH="*n*" HEIGHT="*n*" [attributes] [. . .]> Includes special non standard objects into the current page, such as Adobe PostScript files, or JPEG2000 files, or CAD files, in addition to the standard objects such as GIF and JPEG images and applets.

<NOEMBED>"text"</NOEMBED> Displays "text" if a browser does not support plug-ins.

<OBJECT DATA|CLASSID="url" TYPE="mime-type" WIDTH="*n*" HEIGHT="*n*" [attributes]> Generalizes the <EMBED> element.

<PARAM NAME="name" [VALUE="string" VALUETYPE="data|object|ref" TYPE="mime-type"]> Used to send parameters to the current object.

Layers and Style Sheets.

<LAYER WIDTH="*n*|*n*%" HEIGHT="*n*|*n*%" [attributes]>. . .</LAYER> Defines a new layer in the *z* direction.

<NOLAYER>"text"</NOLAYER> If the browser does not support layers, "text" is displayed.

<STYLE TYPE="mime/type"><!– style sheet –></STYLE> Specifies the formatting within the current Web page.

<!– tagname[.ext] {attributes} –> The syntax of the style sheet.

<LINK REL="stylesheet" TYPE="mime/type" HREF="url"> Links to an external style sheet description for the page.

STYLE="attrib1: value1 [;attrib2: value2 [. . .]]" Specifies a local style to an object to override previous settings.

Appendix 2. Complete List of Special Characters

Character	HTML Decimal Coding	URL Hex Coding	Character Entity	Description
	&#32;	%20	—	Space
!	&#33;	%21	—	Exclamation mark
"	&#34;	%22	&quote;	Quotation mark
#	&#35;	%23	&num;	Hash sign; number sign
$	&#36;	%24	&dollar;	Dollar sign
%	&#37;	%25	&percnt;	Percent sign
&	&#38;	%26	&amp;	Ampersand
’	&#39;	%27	&apos;	Apostrophe
(	&#40;	%28	&lpar;	Left parenthesis
)	&#41;	%29	&rpar;	Right Parenthesis
*	&#42;	%2A	&ast;	Asterisk
+	&#43;	%2B	—	Plus sign
,	&#44;	%2C	&comma;	Comma
-	&#45;	%2D	&hyphen;	Hyphen; soft hyphen
.	&#46;	%2E	&period;	Period
/	&#47;	%2F	—	Slash; forward slash

Character	HTML Decimal Coding	URL Hex Coding	Character Entity	Description
0	&#48;	%30	—	Zero
1	&#49;	%31	—	One
2	&#50;	%32	—	Two
3	&#51;	%33	—	Three
4	&#52;	%34	—	Four
5	&#53;	%35	—	Five
6	&#54;	%36	—	Six
7	&#55;	%37	—	Seven
8	&#56;	%38	—	Eight
9	&#57;	%39	—	Nine
:	&#58;	%3A	&colon;	Colon
;	&#59;	%3B	&semi;	Semicolon
<	&#60;	%3C	&lt;	Less than
=	&#61;	%3D	—	Equal sign
>	&#62;	%3E	&gt;	Greater than
?	&#63;	%3F	—	Question mark
@	&#64;	%40	—	At sign
A	&#65;	%41	—	Capital A
Z	&#90;	%5A	—	Capital Z
[	&#91;	%5B	&lsqb;	Left square bracket
\	&#92;	%5C	&bsol;	Backslash
]	&#93;	%5D	&rsqb;	Right square bracket
^	&#94;	%5E	&circ;	Circumflex
_	&#95;	%5F	&lowbar;	Low bar; underscore
`	&#96;	%60	&grave;	Grave sign
a	&#97;	%61	—	Lower case a
z	&#122;	%7A	—	Lower case z
{	&#123;	%7B	&cub;	Left curly bracket, left brace
|	&#124;	%7C	&verbar;	Vertical bar
}	&#125;	%7D	&rcub;	Right curly bracket, right brace
~	&#126;	%7E	&tilde;	Tilde
†	&#134;	%86	&dagger;	Dagger (also †)
‡	&#135;	%87	&Dagger;	Double dagger (also ‡)
‰	&#137;	%89	&permil;	Per-mille sign (also ‰)
Œ	&#140;	%8C	&OElig;	Capital OE, ligature (also Œ)
™	&#153;	%99	&trade;	Trademark
œ	&#156;	%9C	&oelig;	Lower case oe, ligature (also œ)
		%A0		Non breaking space; hard space
¡	&#161;	%A1	&iexcl;	Inverted exclamation mark
¢	&#162;	%A2	&cent;	Cent sign
£	&#163;	%A3	&pouns;	Pound sterling sign

Character	HTML Decimal Coding	URL Hex Coding	Character Entity	Description
¤	&#164;	%A4	&curren;	General currency sign
¥	&#165;	%A5	&yen;	Yen sign, Japanese
ƒ	&#166;	%A6	&brvbar;	Broken vertical bar
§	&#167;	%A7	&sect;	Section sign
¨	&#168;	%A8	&uml;	Umlaut; dieresis
©	&#169;	%A9	&copy;	Copyright sign
ª	&#170;	%AA	&ordf;	Feminine ordinal
«	&#171;	%AB	&laquo;	Left guillemet
/	&#172;	%AC	&not;	Logical NOT sign
-	&#173;	%AD	&shy;	Soft hyphen; breaking hyphen
®	&#174;	%AE	&reg;	Registered mark
¯	&#175;	%AF	&macr;	Macron accent
°	&#176;	%B0	&deg;	Degree sign
±	&#177;	%B1	&plusmn;	Plus/minus sign
²	&#178;	%B2	&sup2;	Superscript two
³	&#179;	%B3	&sup3;	Superscript three
´	&#180;	%B4	&acute;	Acute accent
µ	&#181;	%B5	&micro;	Micro sign; Greek mu
¶	&#182;	%B6	&para;	Paragraph sign; pilcrow
·	&#183;	%B7	&middot;	Middle dot
¸	&#184;	%B8	&cedil;	Cedilla
¹	&#185;	%B9	&sup1;	Superscript one
º	&#186;	%BA	&ordm;	Masculine ordinal
»	&#187;	%BB	&raquo;	Right guillement
¼	&#188;	%BC	&fract14;	Fraction one-quarter
½	&#189;	%BD	&fract12;	Fraction one-half
¾	&#190;	%BE	&fract34;	Fraction three quarters
¿	&#191;	%BF	&iquest;	Inverted question mark
À	&#192;	%C0	&Agrave;	Capital A, grave accent
Á	&#193;	%C1	&Aacute;	Capital A, acute accent
Â	&#194;	%C2	&Acirc;	Capital A, circumflex
Ã	&#195;	%C3	&Atilde;	Capital A, tilde
Ä	&#196;	%C4	&Auml;	Capital A, umlaut
Å	&#197;	%C5	&Aring;	Capital A, ring
Æ	&#198;	%C6	&AElig;	Capital AE, ligature
Ç	&#199;	%C7	&Ccedil;	Capital C, cedilla
È	&#200;	%C8	&Egrave;	Capital E, grave accent
É	&#201;	%C9	&Eacute;	Capital E, acute accent
Ê	&#202;	%CA	&Ecirc;	Capital E, circumflex
Ë	&#203;	%CB	&Euml;	Capital E, dieresis
Ì	&#204;	%CC	&Igrave;	Capital I, grave accent
Í	&#205;	%CD	&Iacute;	Capital I, acute accent

Character	HTML Decimal Coding	URL Hex Coding	Character Entity	Description
Î	&#206;	%CE	&Icirc;	Capital I, circumflex
Ï	&#207;	%CF	&Iuml;	Capital I, dieresis
Ñ	&#209;	%D1	&Ntilde;	Capital N, tilde
Ò	&#210;	%D2	&Ograve;	Capital O, grave accent
Ó	&#211;	%D3	&Oacute;	Capital O, acute accent
Ô	&#212;	%D4	&Ocirc;	Capital O, circumflex
Õ	&#213;	%D5	&Otilde;	Capital O, tilde
Ö	&#214;	%D6	&Ouml;	Capital O, umlaut
×	&#215;	%D7	&times;	Cross multiplication sign
Ø	&#216;	%D8	&Oslash;	Capital O, slash
Ù	&#217;	%D9	&Ugrave;	Capital U, grave accent
Ú	&#218;	%DA	&Uacute;	Capital U, acute accent
Û	&#219;	%DB	&Ucirc;	Capital U, circumflex
Ü	&#220;	%DC	&Uuml;	Capital U, umlaut
Ý	&#221;	%DD	&Yacute;	Capital Y, acute accent
ß	&#223;	%DF	&szlig;	German ss (eszet)
à	&#224;	%E0	&agrave;	Lowercase a, grave accent
á	&#225;	%E1	&aacute;	Lowercase a, acute accent
â	&#226;	%E2	&acirc;	Lowercase a, circumflex
ã	&#227;	%E3	&atilde;	Lowercase a, tilde
ä	&#228;	%E4	&auml;	Lowercase a, umlaut
å	&#229;	%E5	&aring;	Lowercase a, umlaut
æ	&#230;	%E6	&aelig;	Lowercase ae, ligature
ç	&#231;	%E7	&ccedil;	Lowercase c, cedilla
è	&#232;	%E8	&egrave;	Lowercase e, grave accent
é	&#233;	%E9	&eacute;	Lowercase e, acute accent
ê	&#234;	%EA	&ecirc;	Lowercase e, circumflex
ë	&#235;	%EB	&euml;	Lowercase e, dieresis
ì	&#236;	%EC	&igrave;	Lowercase i, grave accent
í	&#237;	%ED	&iacute;	Lowercase i, acute accent
î	&#238;	%EE	&icirc;	Lowercase i, circumflex
ï	&#239;	%EF	&iuml;	Lowercase i, dieresis
ñ	&#241;	%F1	&ntilde;	Lowercase n, tilde
ò	&#242;	%F2	&ograve;	Lowercase o, grave accent
ó	&#243;	%F3	&oacute;	Lowercase o, acute accent
ô	&#244;	%F4	&ocirc;	Lowercase o, circumflex
õ	&#245;	%F5	&otilde;	Lowercase o, tilde
ö	&#246;	%F6	&ouml;	Lowercase o, umlaut
÷	&#247;	%F7	&divide;	Division sign
ø	&#248;	%F8	&oslash;	Lowercase o, slash
ù	&#249;	%F9	&ugrave;	Lowercase u, grave accent
ú	&#250;	%FA	&uacute;	Lowercase u, acute accent
û	&#251;	%FB	&ucirc;	Lowercase u, circumflex
ü	&#252;	%FC	&uuml;	Lowercase u, umlaut
ý	&#253;	%FD	&yacute;	Lowercase y, acute accent
ÿ	&#255;	%FF	&yuml;	Lowercase y, dieresis

Appendix 3. ASCII Characters that Must be Coded in URLs

Character	Decimal Value	URL Hex Coding	Description
	09	%09	Tab character
	32	%20	Space
”	34	%22	Quotation mark
<	60	%3C	Less than
>	62	%3E	Greater than
[	91	%5B	Left square bracket
\	92	%5C	Back slash
]	93	%5D	Right square bracket
^	94	%5E	Circumflex
`	96	%60	Grave accent sign
{	123	%7B	Left curly bracket, left brace
|	124	%7C	Vertical bar
}	125	%7D	Right curly bracket, right brace
~	126	%7E	Tilde

BIBLIOGRAPHY

1. T. H. Nelson *Dream Machines: New Freedoms Through Computer Screens—A Minority Report*, South Bend, IN: The Distributors, 1978.
2. J. Nielsen *Multimedia and Hypertext: The Internet and Beyond*, Cambridge, MA: Academic Press, 1995.
3. V. Bush As we may think, *Atlantic Monthly*, July 1945.
4. I. S. Graham *HTML Sourcebook*, New York: Wiley, 1995.
5. I. S. Graham *HTML 3.2 Sourcebook*, New York: Wiley, 1997.
6. I. S. Graham *HTML 4.0 Sourcebook*, New York, NY: Wiley, 1998. Information and software also available from http://www.wiley.com/compbooks/graham/.
7. L. Aronson *HTML Manual of Style*, Emeryville, CA: Ziff-Davis Press, 1994.
8. A. Homer C. Ullman S. Wright *Instant HTML: HTML 4.0 Edition*, Birmingham, UK: Wrox Press, 1997.
9. L. Lemay *Teach Yourself Web Publishing with HTML in a Week*, Indianapolis: Sams, 1995.
10. D. Scharf *HTML Visual Quick Reference*, Indianapolis: Que Corp., 1995.
11. *Standard Generalized Markup Language (SGML): ISO 8879: Information Processing: Text and Office Systems*, International Standards Organization, 1986.
12. R. Cover SGML Page: Caveats, work in progress, 1997. Information also available at http://www.sil.org/sgml/caveats.html.
13. T. Berners-Lee Keynote address, Seybold San Francisco, February 1996 [Online]. Available www: http://www.w3.org/Talks/9602seybold/slide6.htm.
14. D. M. Chandler *Running a Perfect Web Site*, Indianapolis: Que Corp., 1995.
15. I. S. Graham *The XHTML 1.0 Web Development Sourcebook: Building Better Sites and Applications*. New York: Wiley, 2000. Information also available from http://www.wiley.com/compbooks/graham/ and http://www.utoronto.ca.
16. B. Le Vitus J. Evans *Webmaster Macintosh*, Boston: AP Professional, 1995.
17. M. E. S. Morris *HTML for Fun and Profit*, Mountain View, CA: SunSoft Press (Prentice-Hall), 1995.
18. D. Taylor *Creating Cool Web Pages with HTML*, 2nd ed., Foster City, CA: IDG Books Worldwide, 1995.
19. L. Weinman W. Weinman *Creative HTML Design*, Indianapolis: New Riders Publishing, 1998.
20. J. D. Murray W. vanRyper *Encyclopedia of Graphics File Formats*, Sebastopol, CA: O'Reilly, 1994.
21. Adobe Photoshop. Information available at http://www.adobe.com.

22. T. Lemke Graphic Converter. Information available at http://www.lemkesoft.de.
23. Y. Piquet Gif Builder. Information available at http://iawww.epfl.ch/staff/yves.piguet/clip2GIF-home/GIFbuilder.html.
24. C. Bäckström Mapper. Information available at http://www.calles.pp.se/nisseb/mapper.html.
25. Java Site Milpitas, CA: Sun Microsystems, 2000. Information available from http://java.sun.com.
26. The Java SIG Team, *Java-SIG's 100 Best Applets*, New York: Wiley, 1997. Information also available from http://www.yahoo.com
27. Opera Browser. Opera Software, 2000. Information available at http://www.opera.com.
28. Mozilla Browser, Netscape Corporation, 2000. Information available at http://www.mozilla.org/.
29. I. S. Graham *HTML Stylesheet Sourcebook*, New York: Wiley, 1997. Information and software also available from http://www.utoronto.ca/ian/books/style/.
30. K. Schengili-Roberts *Core CSS*, Saddle River, NJ: Prentice Hall PTR, 2000.
31. Style Sheet Information, World Wide Web Consortium, 2000. Information available at http://www.w3c.org/Style/CSS/.
32. JavaScript, Netscape Corp., 2000. Information available from http://developer.netscape.com/library/documentation/communicator/jsref/, http://home.netscape.com/eng/mozilla/3.0/handbook/javascript/.
33. JScript and VBScript, Microsoft Corp., 2000. Information available from http://www.microsoft.com/vbscript/default.htm.
34. Dynamic HTML, Netscape Corporation and Microsoft Corporation, 2000. Information available from http://developer.netscape.com/library/documentation/communicator/dynhtml/, http://www.microsoft.com/workshop/author/dhtml/.
35. Mathematical Markup Language, MathML, WWW Corp., 2000. Information available at http://www.w3.org/Math/ and http://www.w3.org/TR/WD-Mmath/.
36. eXtensible Markup Language, World Wide Web Consortium, 2000. Information available at http://www.w3c.org/XML/.
37. I. S. Graham L. Quin *XML Specification Guide*, New York: Wiley, 1999. Information also available from http://www.wiley.com/compbooks/graham/.
38. O'Reilly XML Site [Online] 2001. Available http://www.xml.com/.
39. R. Cover The XML Language, [Online], 2001. Available http://www.oasis-open.org/cover/xml.html.
40. I. S. Graham XHTML 1.0 Language and Design Sourcebook: The Next Generation HTML, New York: Wiley, 2000. Information also available from http://www.wiley.com/compbooks/graham/.
41. M. Dertouzous *What Will Be: How the New World of Information Will Change Our Lives*, San Francisco: HarperEdge, 1997.
42. D. Adams *The Hitchhiker's Guide to the Galaxy*, New York: Ballantine, 1979.

W. KINSNER
University of Manitoba
M. KINSNER
McMaster University