HTML+ (Hypertext markup format)

Status of this memo

This document is an Internet-Draft. Internet-Drafts are working documents of the Internet Engineering Task Force (IETF), its areas, and its working groups. Note that other groups may also distribute working documents as Internet-Drafts.

Internet-Drafts are draft documents valid for a maximum of six months. Internet-Drafts may be updated, replaced, or obsoleted by other documents at any time. It is not appropriate to use Internet-Drafts as reference material or to cite them other than as a “working draft” or “work in progress”.

To learn the current status of any Internet-Draft, please check the `1id-abstracts.txt` listing contained in the Internet-Drafts Shadow directories on `ds.internic.txt`, `nic.nordu.net`, `ftp.nisc.sri.com` or `munnari.oz.au`. Distribution of this document is unlimited. Please mail comments to the author at
`dsr@hplb.hpl.hp.com` or to the discussion list: `www-talk@nxoc01.cern.ch`

This draft is valid until May 1st, 1994. It is available in the file `draft-raggett-www-html-00.ps` and `draft-raggett-www-html-00.txt`.The Postscript file contains figures and formatting examples which are missing from the plain text version.

Abstract

This draft presents a proposal for a light weight delivery format for browsing and querying information in a web of globally distributed hypertext, accessible over the Internet. HTML+ embodies a pageless model making it suitable for efficient rendering on a wide range of display types, for example, VT100 terminals, X11 Workstations, Windows 3.x and the Macintosh. HTML+ is based upon SGML, and represents document elements at a logical level, e.g. headers, paragraphs, lists, tables, and figures. Authors can choose to create HTML+ documents directly, or to use filters to convert from other formats such as LaTeX, Framemaker, and Word for Windows.

HTML+ has grown out of several years experience with the HTML document format in the World Wide Web community. Browser writers are experimenting with extensions to HTML and it is now appropriate to draw these ideas together into a revised document format.. The new format is designed to allow a gradual roll over from HTML, adding features like tables, captioned figures and fill-out forms for querying remote databases or mailing questionnaires. Large documents can be split into a number of smaller nodes for reduced latency, with explicit or implicit navigation links. This draft also includes a proposal to add direct support for mathematical formulae. Authors can include limited presentation hints, and further control may eventually be possible via associated style sheets.

Table of Contents

HTML+ (Hypertext markup format)

1 HTML+ Discussion Document

Following the WWW workshop in July 1993 and subsequent discussions on www-talk, it seems an opportune moment to try and reach a consensus before writing a more formal draft as an informational RFC.

1.1 Introduction

The World Wide Web is a wide-area client-server architecture for retrieving hypermedia documents over the Internet. It also supports a means of searching remote information sources, for example bibliographies, phone directories and instruction manuals. There are three main ingredients: naming schemes for retrievable objects, protocols and interchange formats.

- Universal naming scheme for documents. The Universal Resource Location (URL) syntax specifies documents in terms of the protocol to be used to retrieve them, their Internet Host and path name. A format for location independent lifetime identifiers is currently being defined by a working group of the IETF. A network protocol will allow Universal Resource Numbers (URNs) to be resolved to the URL for the nearest available copy. A URN may specify a number of variants of a document, but the URL will always specify a single copy.
- Use of *de facto* protocols for retrieving documents over the Internet including FTP, NNTP, WAIS, Gopher and HTTP. The latter being designed specifically for the World Wide Web, and uses the MIME message format for document exchange.
- A document format supporting hypertext links based on URLs and URNs which can be rendered on a wide variety of display types. HTML+ is intended in this role as a successor to the existing HTML format.

HTML+ documents offer a means for providing hypertext links to a variety of media including images, sound sequences, MPEG movies, Postscript files and other formats. These links allow a global web of information sources to be established as new servers and document names are announced. Registers of information sources can also be made available via the web, using its ability to let users search for information via keywords. It is hoped that HTML+ will be useful for information exchange via email and network news as well as HTTP.

1.2 Positioning of HTML+

HTML+ is designed for use in the World Wide Web as a non-proprietary delivery format for wide-area hypertext. It embodies a pageless model making it suitable for efficient rendering on a wide range of display types including VT100 terminals, X11, Windows 3.1 and the Macintosh. HTML+ is based upon SGML and represents document elements at a logical level. Authors may choose to create HTML+ documents directly or to use filters to convert from other formats such as LaTeX, Framemaker, and Word for Windows.

1.3 HTML+ and HTML

HTML+ is a superset of HTML and designed to allow a gradual roll over from the earlier format, with features like tables, captioned figures and fill-out forms for querying remote databases or mailing questionnaires. Large documents can be split into a number of smaller nodes for reduced latency, with explicit or implicit navigation links. This draft also includes a proposal to add support for mathematical formulae. Authors can include limited presentation hints, and further control may eventually be possible via associated style sheets.

1.4 HTML+ and SGML

HTML+ is based on the Standard Generalized Markup Language which is an international standard for document markup that is becoming increasingly important. The term markup derives from the way proof-readers have traditionally pencilled in marks that indicate how a document is to be revised.

SGML grew out of a decade of work addressing the need for capturing the logical elements of documents as opposed to the processing functions to be performed on those elements. SGML is essentially an extensible document description language, based on a notation for embedding tags into the body of a document' s text. It is defined by the international standard ISO 8879. The markup structure permitted for each class of documents is defined by an SGML Document Type Definition, usually abbreviated to DTD.

A lot of work is underway to produce DTDs for a range of purposes. These include ISO 12083 for books and ISO 10744 which defines the HyTime architectural forms for hypermedia/time-based documents. The Text Encoding Initiative (TEI) is an international research project for SGML-based document exchange in the humanities. Publishers are cooperating to produce common DTDs for computer manuals, e.g. the DocBook DTD. The CALS programme of the US Department of Defence defines SGML DTDs for documentation for defence procurement contracts.

So what sets HTML+ apart from these efforts? It is impractical to design a DTD to meet the needs of all possible users. Instead, the markup has to be tailored to the needs of a specific community. HTML+ is aimed at fulfilling the dream of a web of information freely accessible over the Internet with links between documents spanning continents. The need to support a very wide range of display types and to keep browser software as simple as possible limits the complexity that can be handled. Similarly the disparate needs of authors has led to the inclusion of limited rendering hints. The features supported arise from several years experience with the World Web and the existing HTML format.

2 An Overview of HTML+

HTML+ documents consists of headers, paragraphs, lists, tables and figures. A simple example of an HTML+ document is:

```
<title>A simple HTML+ Document</title>

<h1 id="a1">This is a level one header</h1>

<p>This is some normal text which will wrap at the window margin.
You can emphasise <em>parts of the text</em> if you wish.

<p>This is a new paragraph. Note that unlike title and header tags
the matching end tag is not needed.
```

The text of the document includes tags which are enclosed in <angle brackets>. Many tags require matching end tags for which the tag name is preceded by the “/” character. The tags are used to markup the document’s logical elements, for example the title, headers and paragraphs. Tags may also be accompanied by attributes, e.g. the *id* attribute in the header tag which can be used to name destinations for hypertext links.

Unlike most document formats, HTML+ leaves out the processing instructions that determine the precise appearance, for instance the font names and point size, the margins, tab settings and how much white space to leave before and after different elements. The rendering software makes these choices for itself (perhaps guided by user preferences). This ensures that browsers can avoid problems with different page sizes or missing fonts. Logical markup also preserves essential distinctions that are often lost by lower level procedural formats, making it easier to carry out operations like indexing and conversion into other document formats.

Note that the tag and attribute names are case insensitive. HTML+ parsers are expected to ignore unrecognised tags and attributes, and to process their contents as if the start/end tags weren't present. SGML minimisation is not supported - this avoids any possibility of confusion with unrecognised tags.

2.1 Document Structure

An HTML+ document consists of some optional declarations followed by one or more elements from the following:

- Headers
- Paragraphs
- Lists
- Figures
- Tables
- Forms
- Literal or Preformatted text
- Mathematical formulae

2.2 Large Documents

Keeping a large document such as a book in one node will increase the time it takes to retrieve the node over the network. It is generally better to split large documents into a number of smaller nodes. Many documents are written with the expectation that the reader will start at the beginning and read through until the end. If the document is split into a number of nodes, the intended sequence is known as a *path*. HTML+ provides a means for authors to specify such paths either explicitly via declarations at the beginning of the node or implicitly according to the context in which a given node is reached. Another possibility is for servers to send such information independently, e.g. as MIME message headers.

You can provide navigation links for readers which appear as buttons on a toolbar or as entries in a navigation menu. For works using a lot of technical terms or perhaps in an unfamiliar language, you can provide glossaries offering further explanation. Readers can invoke this by double clicking on words, or by drag selection and clicking the Glossary menu item. You can also provide a search field that is always present (and can't be scrolled away), in which readers can enter one or more keywords to search an index. These facilities can be specified explicitly using the `LINK` element. Implicit links allow you to define the table of contents (toc) as an

HTML+ document without needing to place links to the toc in every subdocument. The link back to the toc is implied when you follow hypertext link from the toc to its subdocuments.

3 Headers

The tags `H1`, `H2`, ... `H6` are used to represent headers. `H1` is the most significant and rendered in a large font[1] (preferably centered). `H2` to `H6` are progressively less significant and usually rendered flush left in smaller fonts. A common convention is to begin the body of a document with an `H1` header, e.g.

```
<h1>Introduction to HTML+</h1>
```

Header names should be appropriate to the following section of the document, while the title should cover the document as a whole. There are no restrictions on the sequence of headers, e.g. you could use a level three header following a level one header.Header and section elements can take an identifier, unique to the current document, for use as named destinations of hypertext links. This is specified with the `ID` attribute, e.g.

```
<h1 id=”intro”>Introduction to HTML+</h1>
```

This allows authors to create hypertext links to particular sections of documents. It is a good idea to use something obvious when creating an identifier, to help jog your memory at a later date. WYSIWYG editors should automatically generate identifiers. In this case, they should provide a point and click mechanism for defining links so that authors don't need to deal explicitly with identifier names. Automatic generation of `IDs` for headers, paragraphs and other major elements is important as it makes it easier for other people to create links to your document, by ensuring that there are plenty of `ID` attributes present as potential destinations.

Should we support headers for which the level is implicitly defined by nestable section elements?[2] We could also support autonumbering of headers. Unfortunately, on further investigation these ideas proved trickier than thought at first, and so have been dropped from this draft.

4 Paragraphs and <P>

Normal text is automatically wrapped by the browser at the current window margin and adapts to changes in window size. The text is generally shown in a proportional font:

```
<P ID=”p1”>The P element acts as container for the text between the start
tag &lt;P&gt;;[3] and end tag &lt;/P&gt;. You don’t need to give the end tag
as it is implied by the context, e.g. the following &lt;P&gt; tag.
<P ID=”p2”>If you wish, you may think of the &lt;P&gt; tag as a paragraph
separator. This works since HTML+ formally doesn’t require you to wrap
text up as paragraphs.
```

This would be rendered as:

> The P element acts as a container for the text between the start tag <P> and the end tag </P>. You don't need to give the end tag as it is implied by the context, e.g. the following <P> tag.

---

1. Headers are often rendered in a sans serif font with paragraph text in a serif font.
2. For example with <H> for headers and <SECTION> for nestable sections.
3. < and > are SGML entity definitions for the < and > characters, see section 5.1.

If you wish, you may think of the <P> tag as a paragraph separator. This works since HTML+ formally doesn't require you to wrap text up as paragraphs.

The following samples of HTML+ all produce exactly the same results when displayed:

```
<H1>Different ways of using the P element</H1>
<P>The first piece of text</P><P>The second piece</P>
```

```
<H1>Different ways of using the P element</H1>
<P>The first piece of text<P>The second piece
```

```
<H1>Different ways of using the P element</H1>
The first piece of text<P>The second piece
```

They all produce:

Different ways of using the P element

The first piece of text

The second piece

In some situations you will want to preserve the original line breaks and spacing, for this you should use the `LIT` or `PRE` elements, these are described in a later section. You can force line breaks in normal paragraph text with the `<BR>` element, but the browser may wrap lines arbitrarily at window margins prior to reaching the `<BR>` element.

The `ALIGN` attribute can be used to center a paragraph, e.g. `<P ALIGN=center>`. Other possibilities are `ALIGN=left` (the default), `ALIGN=right`, `ALIGN=justify` and `ALIGN=indent`. This attribute is a hint and may be ignored by some browsers. Note that when using explicit line breaks (see section 5.12) you may wish to switch off word wrap with `WRAP=OFF`.

Browsers, when parsing paragraphs, can choose to simply treat the `<P>` tag as denoting a paragraph break. If the paragraph style includes a blank line between paragraphs, then additional care is needed after headers and other major elements[1] to avoid inserting an unwanted blank line, e.g. when a `<P>` tag directly follows a header. This ability to perceive `<P>` as a paragraph break provides for continuity with HTML, and allows authors to graduate to treating it as a container in their own time.

5 Normal Text

Paragraphs can include the following:

- Character entity names for unusual characters such as ó which are included using SGML entity definitions: &*name*; as in "the dream of óengus" which is displayed as: "the dream of óengus". The full list of standard entity names recognised by most browsers is given in Appendix II
- Character emphasis using logical and presentational markup. The set of logical character emphasis can be extended, and HTML+ provides the means for browsers to determine how to render such extensions
- Simple footnotes or margin notes, which can be rendered as pop-up overlays
- Images which act as single characters and which can be vertically aligned relative to the text line in which they are embedded

1. These are: H, H1 to H6, NOTE, DL, UL, OL, LIST, FIG, TABLE, FORM and MATH

- Hypertext Links based on the URL or URN notations
- Markup signifying the start and end of change bars. You can also mark text as being removed or added, as is common in legal documents
- Conditional text which appears only on-line or only when printed
- Input fields when the paragraph is part of a form
- Explicit line breaks

5.1 Character Sets and Entity Definitions

By default, HTML+ documents are made up of 8-bit characters from the ISO 8859 Latin-1 character set. The network protocol used to retrieve documents may translate the character set into a locally acceptable form, e.g. EBCDIC. The HTTP protocol uses the MIME standard (RFC 1341) to specify the document type and character set. ISO SGML entity definitions are used to include characters which are missing from the character set or which would otherwise be confused with markup elements, e.g:

&	ampersand &
<	less than sign <
>	greater than sign >
"	the double quote sign “

Appendix II lists a broad range of characters and symbols, relating their ISO names to the corresponding character codes in common character sets. They allow authors to include accented characters in 7-bit ASCII documents. Some other useful entity definitions are:

–	en dash (half the width of an em unit)
—	em dash (equal to width of an “m” character)
	en space
	em space
	non breaking space
	soft hyphen (normally invisible)
©	copyright sign ©
™	trade mark sign ™
®	registered sign ®

There are a large number of entities defined by the ISO, covering most languages and symbols for publishing and mathematics. Requiring all browsers to support these would be impractical, e.g. how should a dumb terminal show such symbols. In some cases there will be accepted ways of mapping them to normal characters, e.g. **æ** as **ae** and **è** as **e**. Perhaps the safest recommendation is that where authors need to use a specialised character or symbol, they should use ISO entity names rather than inventing their own. Browsers should leave unrecognised entity names untranslated.

In some cases it is useful to specify the language used in a given element, with the LANG attribute. The ISO defines abbreviations for most languages, e.g. FR for french as in: <Q LANG="FR">Je m'aveugle.</Q>. This attribute permits language dependent layout and hyphenation decisions, e.g. Hebrew uses right to left word order.

To allow SGML parsers to recognise entity names, authors should declare them before use, for example:

```
<!ENTITY % ISOcyr1 PUBLIC "ISO 8879-1986//ENTITIES Russian Cyrillic/EN">
%ISOcyr1;
```

This introduces ISOcyr1 as a local name for the ISO public identifier for the cyrillic alphabet and then includes the associated set of entity definitions as part of the current document. This declaration is unnecessary for entities defined within the HTML+ DTD.

5.2 Hypertext Links

HTML+ allows authors to embed hypertext links into the document text. In a browser this might look to the reader like:

Links are defined with the *A tag*. HTML+ supports a number of *different link types*.

Clicking on a link will normally cause the browser to retrieve the linked document and display it in place of the current one. This example is represented by the following piece of HTML+

```
Links are defined with the <a href="#z1">A tag</a>. HTML+ supports a
number of <a href="links.html">different link types.
```

The first link is to an anchor named “z1” in the current document (using an `ID` attribute on some element). The second is to a file named “links.html” in the same directory as the current document. The link caption is the text between the start and end tags. The `HREF` attribute defines the link destination using the URL or URN notations. This may be abbreviated in certain circumstances using relative URLs. The link should be rendered in a clearly distinguishable way, e.g. as a raised button, or with underlined text in a particular color or emphasis. For displays without pointing devices, it is suggested that the link is indicated with a reference number in square brackets after the caption, which the reader enters to follow the link. Note that it is illegal for anchors to include headers, paragraphs, lists etc. The anchor text is restricted to normal text with emphasis and inline images.

The A element has several optional attributes:

`ID` This can be used to define a unique identifier for the text contained by the A element. Another document can then make a reference to this by putting the identifier after the URL for this document, separated by a hash sign. The `ID` attribute replaces the `NAME` attribute in HTML.

`HREF` This specifies a URL or URN of a linked document which will be retrieved when the user clicks on the anchor’s label. `HREF=#`*`id`* can be used for links to other parts of the same document.

`REL` The relationship the linked document has to this one. `REL=Subdocument` is used to break long documents into smaller ones. This importance of this particular attribute value is explained in section 14.

`REV` The reverse relationship type, and the inverse of `REL`.

`EFFECT` This determines how the browser displays the linked document when following the link. `EFFECT=Replace` causes the browser to replace the current document with the linked one; `EFFECT=NEW` results in the linked document being shown in a new window (if practical); and `EFFECT=OVERLAY` causes the linked document to be shown in a pop-up window, as used by the Microsoft Windows Help system.

PRINT — This attribute makes it easy for users to print off the current document and relevant parts. PRINT=REFERENCE (the default) treats the link as a reference, i.e. the URL is given as a footnote; PRINT=FOOTNOTE prints the linked document as a footnote; PRINT=SIDEBAR prints the linked document as a sidebar; and PRINT=SECTION prints the linked document as a follow on section. Use PRINT=SILENT when you don't want the link referenced or printed out[1].

TITLE — Defines the title to use when the linked document is otherwise untitled.

TYPE — The MIME content type of the linked document - for use in providing presentation cues only, as it could easily become out of date.

SIZE — The size in bytes for the linked document. This should only be used as a guide to progress in retrieving documents, as it is likely to get out of step with changes to the target document.

METHODS — This is a comma separated list of HTTP methods supported by the linked object. The browser might choose a different way of rendering the link for say searchable objects.

SHAPE — This is used to define shaped buttons on top of images or figures, and is explained later on.

You can also use the LINK element at the start of the document to define document-wide relationships with other documents, e.g. a link to a table of contents. This is described later on.

5.3 Character Emphasis

There has been considerable discussion on how to represent character emphasis. The previous draft of HTML+ used a single element to handle all forms with a role attribute for the logical role, and other attributes for providing hints as to how to render the emphasis. This mechanism was seen as being overloaded and prompted the use of separate elements in the current draft.

5.4 Presentation Only Tags

In many cases it is convenient to indicate directly how the text is to be rendered, e.g. as italic, bold, underline or strike-through:

<I>italic text</I>	*italic text*
<B>bold text</B>	**bold text**
<U>underlined text</U>	underlined text
<S>strike through</S>	~~strike through~~
^{superscript}	superscript
_{subscript}	subscript
<TT>fixed pitch</TT>	fixed pitch (TT for Teletype)

These tags may be nested to combine effects, e.g. bold-italic-fixed-pitch text, and should be considered as hints rather than as binding obligations on the browser, e.g.

```
Some <B><I><TT>bold italic fixed pitch text</TT></I></B>.
```

which is rendered as: Some ***bold italic fixed pitch text***.

1. Browsers might want to count the number of links on a page and switch to the silent mode if there are too many to reference in footnotes.

5.5 Generic Emphasis

These are some tags for indicating a level of emphasis without committing oneself to how they should be rendered:

`<EM>normal emphasis</EM>`	typically italic
`<STRONG>strong emphasis</STRONG>`	typically bold

5.6 Logical Emphasis

These tags indicate the role of the marked text, e.g. bibliographic references. By using a standard way of marking up text, it becomes possible to automatically index such references. There are a potentially huge number of different distinctions that could be made, and the set given below is intentionally minimalistic. Discussion is welcomed on just which elements should be included in HTML+ given its intended role as a delivery format for hypertext documents:

`q`	a short quotation which can be included inline, e.g. <q>to be or not to be, that is the question</q> use <q> and </q> in place of double quote marks.
`cite`	citation, e.g. <cite>Festinger, L.(1957), <I>A Theory of Cognitive Dissonance</I>, Stanford.</cite>
`person`	proper names, e.g. <person>Albert Einstein</person>
`acronym`	acronyms e.g <acronym>NATO</acronym>
`abbrev`	abbreviations, e.g. <abbrev>v. aux</abbrev>
`cmd`	command name, e.g. chmod in Unix
`arg`	command argument, e.g. <arg>-s</arg>
`kbd`	something the use would have to type
`var`	named place holder, e.g. <var>filename</var>
`dfn`	defining instance of a term
`code`	code example - usually shown in fixed pitch font
`samp`	sequence of literal characters usually in variable pitch font

All these tags require a matching closing tag like the other emphasis elements, e.g.

<cmd>cmp</cmd> [<arg>-l</arg>] [<arg>-s</arg>] <var>file1</var> <var>file2</var>

5.7 Extending the Set of Logical Roles

When translating from other SGML-based formats, documents may include non-standard elements e.g. `PROPNAME` for proper names. HTML+ browsers will normally ignore such markup and process their contents as if these tags weren't present. The `RENDER` element can be used to tell the browser how to render such tags, e.g.

```
<RENDER TAG="PROPNAME" STYLE="I">
```

The `STYLE` attribute is a comma separated list of presentation tag names, i.e. one or more names from the list: `I`, `B`, `U`, `S`, `SUP`, `SUB`, `TT`. Include `P` in the list of styles if the element needs a paragraph break. Keeping non-standard markup in HTML+ documents may be useful for indexing purposes. Note that the `RENDER` element isn't meant to apply retrospectively.

5.8 Annotations

Authors can include annotations which act to draw the readers attention or which provide some additional comment on the main text of the paragraph. There are two types:

`footnote`	for additional information on some point
`margin`	attention getter for this paragraph

When printed out, these annotations appear as footnotes or margin notes as their name implies, e.g. <footnote>This is an example of a footnote</footnote>[1]. For on-line use, browsers may show the annotation by a hypertext button, e.g. a superscripted dingbat symbol or icon, which when clicked reveals the annotation in a pop-up window. Footnotes and margins can contain text with emphasis and images, but not other markup such as paragraphs, lists or tables. The `PANEL` element in the previous draft has been dropped. You can however, indicate that a hypertext link should be rendered as a sidebar when printed out.

5.9 Images

Images can be included as character like elements with text flowing around the image, e.g.

Before coming to CERN, Tim worked on, among other things, document production and text processing. He developed his first hypertext system, "Enquire", in 1980 for his own use (although unaware of the existence of the term HyperText). With a background in text processing, real-time software and communications, Tim decided that high energy physics needed a networked hypertext system and CERN was an ideal site for the development of wide-area hypertext ideas.Tim started the WorldWideWeb project at CERN in 1989. He wrote the application on the NeXT along with most of the communications software.

This example is produced by the following piece of HTML+

```
<p><img align=top src="http://spoof.cern.ch/people/tbl.gif"> Before com-
ing to CERN, Tim worked on, among other things, document production and
text processing. He developed his first hypertext system, "Enquire", in
1980 for his own use (although unaware of the existence of the term
HyperText). With a background in text processing, real-time software and
communications, Tim decided that high energy physics needed a networked
hypertext system and CERN was an ideal site for the development of wide-
area hypertext ideas. Tim started the WorldWideWeb project at CERN in
1989. He wrote the application on the NeXT along with most of the commu-
nications software.
```

The `IMG` element specifies an image via a `URL`. The `ALIGN=TOP` attribute ensures that the top of the image is level with the top of the current text line. You can also use `ALIGN=MIDDLE` to align the center of the image with that of the current text line, and `ALIGN=BOTTOM` to align the bottom of the image with the bottom of the current text line. Browsers are not expected to apply text flow retrospectively, so using `ALIGN=MIDDLE` and `ALIGN=BOTTOM` may overwrite previous lines of text. If the `ALIGN` attribute is missing then `ALIGN=TOP` is assumed.

1. This is an example of a footnote (while not strictly accurate, this gives the general idea!).

Not all display types can show images. The `IMAGE` element behaves in the same way as `IMG`[1] but allows you to include descriptive text, which can be shown on text-only displays:

```
<image align=top src="http://spoof.cern.ch/people/tbl.gif">A photo of
Tim Berners-Lee</image> Before coming to CERN, Tim worked on, among other
things, document production and text processing. etc.
```

On text-only displays, the text within the `IMAGE` element can be shown in place of the image:

```
[A photo of Tim Berners-Lee] Before coming to CERN, Tim worked on, among
other things, document production and text processing. etc.
```

The `SEETHRU` attribute can be used to designate a chromakey so that the image background matches the document background. This is an experimental feature and the format of the attribute's value has yet to be defined - *suggestions are welcomed.*

Images can be made active in one of three ways

- The whole image can be made into a hypertext link
- Mouse/Pen clicks on the image can be passed to a WWW server
- Shaped hypertext buttons can be overlayed on the image

Making the entire image into an iconic hypertext button is simple:

```
<a href="bigpic.giff"><image src="smallpic.gif">Our house</image></a>
```

In this example, readers can click on a small picture embedded in the document to see a larger version, which would take significantly longer to retrieve. When using images as hypertext links, don't forget to include a textual description. This is needed for the link caption for people using text-only displays.

In some cases, servers can handle mouse clicks or drags on the image. This capability is signalled in the header information returned along with the image data. You can also use the `ISMAP` attribute. This mechanism and the ability to add shaped buttons are defined in detail in the description of figures.

The delay in connecting to the server for each image in turn can be reduced by asking HTTP servers to include images with the HTML+ document as a MIME multipart message (include `multipart/mixed` with the `Accept`: header in the request message).

5.10 Change Bars and Document Amendments

Change bars are shown for parts of the document designated with the `CHANGED` element. This can appear anywhere that normal text is allowed (as shown by the `%text;` entity reference in the DTD):

```
<changed id=z34>text including some changes<changed idref=z34>
```

The same element[2] is used to designate the start and end of changes, using matched `ID` and `IDREF` attribute values. This mechanism avoids syntactic problems that would arise from using a conventional start and end tag pair, as changes to a document can span different levels of the

1. You can also use `IMG`'s `ALT` attribute for textual descriptions, but authors are recommended to use `IMAGE` in all cases. Such descriptions should always be given as it is bad practice to rely on users having graphical browsers. The descriptions are also useful when the images take a long time to retrieve.
2. Maybe we should use distinct tag names- if so what names?

document's formal structure. Additional attributes may be used with the CHANGED element to hold related details, e.g. BY, WHEN, WHY, WHAT.

In legal documents and amendments to proposed legislation, there is often the need to show parts of the text as being removed or added to the document. This is commonly shown using strike-through and underlining respectively. The REMOVED and ADDED tags are provided for this purpose:

```
<P>This bill would require the Legislative Counsel, with the advice of the
Joint Rules Committee of the Senate and Assembly, to make available to the
public by means of access by way of <removed>computer modem</removed>
<added>the largest nonproprietary, nonprofit cooperative public computer
network,</added> specified information concerning bills, the proceedings
of the houses and committees of the Legislature, statutory enactments, and
the California Constitution.
```

Which might be displayed as:

> This bill would require the Legislative Counsel, with the advice of the Joint Rules Committee of the Senate and Assembly, to make available to the public by means of access by way of ~~computer modem~~ <u>the largest nonproprietary, nonprofit cooperative public computer network</u>, specified information concerning bills, the proceedings of the houses and committees of the Legislature, statutory enactments, and the California Constitution.

Color enhancements may be used to further distinguish the amendments, e.g. red lines for strike-through. This mechanism is not intended for representing revision histories, which are better served by traditional change control mechanisms.

5.11 Conditional Text

It is often quite difficult to phrase the captions for hypertext buttons so that they make sense when printed out. The ONLINE and PRINTED elements can be used to define text which is for use only when read on-line or on the printed page respectively:

```
<online>click <a href="info.html">here</a>for more information.</online>
<printed>Further information can be found in [Higgins 84b].</printed>
```

In many cases, you can find a way of phrasing the reference so that it makes sense both ways. Browsers can help by referencing hypertext links as footnotes when printed out. See the earlier description of the PRINT attribute for the A tag.

5.12 Explicit Line Breaks

You can make individual lines explicit with the <L> element, which contains the text of the line in the same way that <P> contains the text of the paragraph.

```
<P>
<L>22 The Avenue,
<L>Harrow,
<L>London, NW1 5ER
```

An alternative is the
 element which acts as a forced line break.

```
<P>22 The Avenue<BR>
Harrow,<BR>London, NW1 5ER
```

The <L> element is useful when you want to name each line, e.g. <L ID="L23">. You may also want to disable word wrap for the current paragraph, as in <P WRAP=OFF>.

6 Different Paragraph Styles

To avoid all text appearing in the same style, HTML+ provides distinct styles for quotes, abstracts, bylines and admonishments. All these elements can contain multiple paragraphs:

6.1 Longer Quotations

When you want to include a quotation that extends over more that one paragraph, you should use the `QUOTE`[1] element. Quoted text should preferably be indented, and rendered using a distinctive font, e.g.

```
<P>The following is a quotation from the forward by Yuri Rubinsky to "The
SGML Handbook" by Charles F. Goldfarb, published by the Clarendon Press,
Oxford, 1990.

<QUOTE>The next five years will see a revolution in computing. Users will
no longer have to work at every computer task as if they had no need or
ability to share data with all their other computer tasks, they will not
need to act as if the computer is simply a replacement for paper, nor
will they have to appease computers or software programs that seem to be
at war with one another.</QUOTE>
```

which might be rendered as:

> The following is a quotation from the forward by Yuri Rubinsky to "The SGML Handbook" by Charles F. Goldfarb, published by the Clarendon Press, Oxford, 1990.
>
> > The next five years will see a revolution in computing. Users will no longer have to work at every computer task as if they had no need or ability to share data with all their other computer tasks, they will not need to act as if the computer is simply a replacement for paper, nor will they have to appease computers or software programs that seem to be at war with one another.

6.2 Abstracts

The `ABSTRACT` element can be used to give an overview of a document and typically follows a level one heading. It should be rendered in an easily read font, distinct from normal text, and preferably indented. An example is given in the next section.

6.3 Bylines

The `BYLINE` element[2] is similar to `QUOTE` and is used for information about the author, e.g. contact details and release date. A common convention is to include a hypertext link to a node with more information about the author. Bylines can occur at the beginning or end of a document, e.g:

```
<H1>HTML+ (Hypertext markup format</H1>

<ABSTRACT>A proposed standard for a light weight delivery format for
   browsing and querying information in a web of globally distributed
   hypertext accessible over the Internet
</ABSTRACT>
```

---

1. The equivalent `BLOCKQUOTE` element is provided for backwards compatibility with HTML. Perhaps we should extend `QUOTE` to allow quote by *name*, whereby, the quoted material is defined by a hypertext link to the original material, and automatically retrieved and inserted into the current document?

2. You can also use the ADDRESS element (provided for backwards compatibility with HTML).

```
<BYLINE>Editor: Dave Raggett dsr@hplb.hpl.hp.com</BYLINE>
```

6.4 Notes and admonishments

The NOTE element is used when you want to draw the readers attention to some point or other. For example:

```
<note role="NOTE" src="info.gif">
The "partial-window-name" parameter must exactly match the beginning
characters of the window name (as it appears on the title bar), including
proper case (capital or lower letters) and any punctuation.
</note>
```

This is typically rendered as:

NOTE: The "partial-window-name" parameter must exactly match the beginning characters of the window name (as it appears on the title bar), including proper case (capital or lower letters) and any punctuation.

The text of the ROLE attribute (if given) is inserted at the start of the note in a bold font and followed by a colon. Typical roles are TIP, NOTE, WARNING and ERROR. The SRC attribute may be used to name a URL or URN as an icon which is displayed in the left margin at the start of the note. An upright hand icon is often used for tips; a warning road sign for warnings and a stop sign for errors[1]. Horizontal rules are drawn automatically to help readers distinguish the note from the surrounding text. You can place horizontal rules in other parts of your document using the <HR> element which can appear anywhere a <P> element is allowed.

7 Lists

There are three kinds of lists, which can be freely nested within one another:

- Ordered lists - the list items are automatically numbered
- Unordered lists - bulleted or plain styles, in single or multiple columns
- Definition lists of terms and associated definitions

7.1 Ordered Lists

The OL element is used with LI for each item to represent ordered lists:

```
<OL>
   <LI>Wake up
   <LI>Get dressed
   <LI>Have breakfast
   <LI>Drive to work
</OL>
```

which is usually rendered as:

1) Wake up
2) Get dressed

1. Browsers should provide these particular icons by default when the SRC attribute is missing.

3) Have breakfast
4) Drive to work

The `COMPACT` attribute when present e.g. `<OL COMPACT>` has the effect of reducing inter-item spacing. The numbering style is the responsibility of the browser. Other styles use roman numerals or letters from the alphabet in upper or lower case. One issue for browsers, is how to render ordered lists, nested within a list of the same type. List item text can't include headers, see the DTD in Appendix I for details.

7.2 Bulleted Lists

Bulleted lists are represented with the `UL` and `LI` elements:

```
<UL>
   <LI>Wake up
   <LI>Get dressed
   <LI>Have breakfast
   <LI>Drive to work
</UL>
```

which is usually rendered as:

- Wake up
- Get dressed
- Have breakfast
- Drive to work

The `COMPACT` attribute when present e.g. `<UL COMPACT>` has the effect of reducing inter-item spacing. The bullet style is the responsibility of the browser, and normally an unordered list nested within a list of the same type is given a different style (bullet, dash, box or check). Authors can instead use the `SRC` attribute for the `LI` element to specify an icon with a URL or URN, e.g. `<LI SRC="folder.gif">`. List item text can't include headers, see the HTML+ DTD in Appendix I for details.

7.3 Plain Lists

Plain lists without bullets are represented by the `UL` element together with the `PLAIN` attribute, e.g. `<UL PLAIN>`. The `WRAP` attribute is used for multi-column lists and should be `WRAP=HORIZ` for horizontally wrapping of list items or `WRAP=VERT` for vertical wrapping of list items, e.g.

```
<UL PLAIN>
   <LI>icons1/
   <LI>icons2/
   <LI>icons3/
   <LI>src/
   <LI>xpm-3-paper.ps
   <LI>xpm-3.2-to-3.2a.patch
</UL>
```

without the `WRAP` attribute, this is rendered as:

icons1/
icons2/
icons3/

src/
xpm-3-paper.ps
xpm-3.2-to-3.2a.patch

with `<UL PLAIN WRAP=VERT>` this would appear like:

icons1/	icons3/	xpm-3-paper.ps
icons2/	src/	xpm-3.2-to-3.2a.patch

with `WRAP=HORIZ` it would appear like:

icons1/	icons2/	icons3/
src/	xpm-3-paper.ps	xpm-3.2-to-3.2a.patch

Everyday familiarity with printed lists leads us to expect lists to be organized into columns which are read top to bottom; horizontally wrapped lists are seldom seen. Browsers are free to choose the number of columns to match the current window size and item widths. If there are N items and M columns then the longest column will have (N+M-1)/M rows. This requires a prepass through the list to count the items (and optionally their maximum width). However, this information can be cached to avoid speed penalties when resizing the window or refreshing the screen. You can use the `SRC` attribute for the `LI` element to specify an icon for each item in the list, e.g. to show the type of each document in a directory listing.

For convenience, the `<MENU>` and `<DIR>` elements can be used in place of `<UL PLAIN>` and `<UL PLAIN WRAP=VERT>` respectively.

7.4 Definition Lists

These consist of **pairs** of terms and definitions, but can also be used for plays as in:

```
<DL>
   <DT>King Henry
   <DD>I myself heard the King say he would not be ransomed.
   <DT>Williams
   <DD>Ay, he said so, to make us fight cheerfully: but when our
      throats are cut he may be ransomed, and we none the wiser.
   <DT>King Henry
   <DD>If I live to see it, I will never trust his word after.
   <DT>Williams
   <DD>You pay him then! That's a perilous shot out of an elder-gun,
      that a poor and a private displeasure can do against a monarch!
      You may as well go about to turn the sun to ice, with fanning in
      his face with a peacock's feather. You'll never trust his word
      after! Come 'tis a foolish saying.
</DL>
```

This could be rendered as:

King Henry: I myself heard the King say he would not be ransomed.

Williams: Ay, he said so, to make us fight cheerfully: but when our throats are cut he may be ransomed, and we none the wiser.

King Henry: If I live to see it, I will never trust his word after.

Williams: You pay him then! That's a perilous shot out of an elder-gun, that a poor and private displeasure can do against a monarch! You may as well go about to turn the sun

> to ice, with fanning his face with a peacock's feather. You'll never trust his word after! Come 'tis a foolish saying.

or as:

King Henry	I myself heard the King say he would not be ransomed.
Williams	Ay, he said so, to make us fight cheerfully: but when our throats are cut he may be ransomed, and we none the wiser.
King Henry	If I live to see it, I will never trust his word after.
Williams	You pay him then! That's a perilous shot out of an elder-gun, that a poor and private displeasure can do against a monarch! You may as well go about to turn the sun to ice, with fanning his face with a peacock's feather. You'll never trust his word after! Come 'tis a foolish saying.

Browsers should make allowance for the infrequent case when the term text (`DT`) is longer than the definition text (`DD`) and wraps onto subsequent lines. Note that you are allowed to have several consecutive `DT` elements followed by a `DD` element, but you can't have `DD` without an associated `DT` element, The `COMPACT` attribute as in `<DL COMPACT>` forces the browser to use the former more compact style.

8 Figures

The FIG element is similar to the `IMAGE` element, but acts as a paragraph. The `ALIGN` attribute can be one of `LEFT` (the default), `CENTER`, `RIGHT` or `FLOAT`. This determines whether the figure is flush left, centered or flush right. If `ALIGN=FLOAT` the figure may float to another more convenient location (and possibly zoomed or reduced in the process). A caption can be defined with the `CAPTION` element and followed by text describing the figure for readers using text only displays[1]:

```
<FIG ALIGN=FLOAT SRC="cat.gif">
   <CAPTION>"Not curried fish again!"<CAPTION>
   A cartoon of a scrawny cat with its tongue out saying ACK!
</FIG>

<P>The text in the following paragraphs will flow around the figure
if there is enough room. The browser is free to position the caption at
the top, bottom or sides of the figure.
```

1. The caption must be placed before this description, and can be used as a header for the figure description for text-only displays. This text may also be usefully shown while the browser is retrieving the image data for GUI displays.

which is rendered as:

“Not curried fish again!

The text in the following paragraphs will flow around the figure if there is enough room. The browser is free to position the caption at the top, bottom or sides of the figure.

Note that browsers can only support a limited range of image types. Currently these are GIF and XBM (X bitmap format). This list will evolve over time.

8.1 Active Areas

The upper left of the image is designated as x,y = (0, 0), with x increasing across the page and y down the page. This choice was made for continuity with the `IMG` element in HTML, to ensure a simple migration path to HTML+. If points are given in real numbers, the lower right corner of the image is taken as being (1.0, 1.0), otherwise, with integer values the coordinates are assumed to be in pixels. A simple test to distinguish the two schemes is to check if a “.” character occurs anywhere in the list of points. Using scaled coordinates is much safer as the pixel extent of an image may alter, e.g. as a result of format negotiation with the server.

For some images, HTTP servers will be able to handle mouse/pen clicks or drags on the image. This is signalled in the header information returned along with the image data. Alternatively, the `ISMAP` attribute can be used to signal this capability. The mouse click is sent to the server indicated by the URL in the `SRC` attribute, using the same URL plus the suffix “?x=*X*&y=*Y*”[1] where *X* and *Y* are the coordinates of the click event. Mouse drags can be used to designate a rectangular region of the image. In this case the suffix takes the form: “?x=*X*&y=*Y*&w=*W*&h=*H*” where *(X, Y)* is the upper left of the rectangle, and *(W, H)* define its width and height. The `ISMAP` mechanism is useful when the active regions in the image change their boundaries with time, e.g.

```
<fig ismap src="weather.gif">
   <caption>Click on your area for a local forecast</caption>
   Todays weather map for the US.
</fig>
```

8.2 Placing Hypertext Buttons on Images

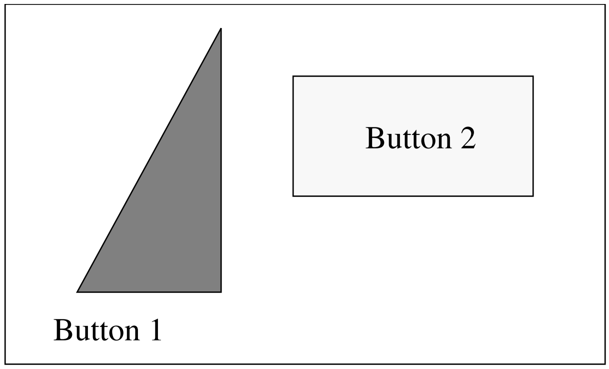
The `A` element[2] can be used to define shaped buttons on top of images. The shape is defined by an arbitrary polygon and specified via the `SHAPE` attribute, e.g.

```
<FIG SRC="test.gif">
   <CAPTION>Click on the triangle or the rectangle</CAPTION>
   A line drawing with a
   <A SHAPE="0.35,0.1&0.1,0.8&0.35,0.8"HREF="button1.html">
   triangle</A>and a
```

1. Changed from “?X,Y” in the previous draft for consistency with the `FORM` mechanism.
2. This replaces the `FIGA` element in the previous draft, and gives readers on text-only displays the ability to click shaped buttons even though the image itself can’t be shown.

```
    <A SHAPE="0.5,0.25&0.5,0.5&0.8,0.5&0.8,0.25" HREF="button2.html">
    rectangle</A>
</FIG>
```

Which could be rendered as:



Click on the triangle or the rectangle

The example uses scaled coordinates, and shows how you give the vertices of the polygon defining the shape of the button. Like the ISMAP mechanism, you can use pixel-based coordinates by using integer numbers throughout. Note that clicks on shaped buttons take precedence over the ISMAP mechanism for sending events to the server. An efficient algorithm for testing if a mouse/pen click lies inside a polygon is given as a C routine in Appendix III.

8.3 Possible extensions

In future, HTML+ may be extended to support simple drawings with embedded hypertext links. One idea would be a line drawing primitive using the SHAPE attribute. A better approach is to extend existing drawing formats such as the ANSI Computer Graphics Metafile format (CGM) or Adobe's PDF to include URL based hypertext links. This extended format could then be used for figures within HTML+ documents.

The use of the MIME multipart message format would also help to speed up the display of figures by sending image data at the same time as the HTML+ document. Another possibility would be to allow image data to be embedded in the document using an EMBED element in place of the SRC attribute. Binary data could be represented using the MIME character encoding or the more compact ASCII base 85 encoding, as used in Adobe's PDF. The drawback with this approach is the inability to use format negotiation. As a result, the EMBED element has been dropped from the current draft.

9 Tables

Tables are specified using the TABLE element. This allows you to define a caption and to differentiate header and data cells. Cells may contain, text, multiple paragraphs, lists and headers.

Adjacent cells can be merged, e.g. to define a header which spans two columns. A simple table could look like:

Table 1: A simple table

Year	Month	Day
1972	June	23rd
1982	October	7th

This is defined by the markup:

```
<table border>
   <caption>A simple table</caption>
   <th>Year <th>Month <th>Day <tr>
   <td>1972 <td>June <td>23rd <tr>
   <td>1982 <td>October <td>7th
</table>
```

The BORDER attribute acts as a hint to the browser to draw lines enclosing each cell. The TH element precedes header cell text and the TD element precedes data cell text. The TR element is used to separate table rows. By default text is centered in each cell. Header text should be shown emphasised, e.g. the browser could use a bold sans serif font for headers and a serif font for the data cells. The next example shows how cells can be merged with their neighbors:

Table 2: A more complex table

	average		other category
	height	**weight**	
males	height	0.003	yyy
females	1.9	0.002	xxx

This table is defined by the markup:

```
<table border>
  <caption>A more complex table</caption>
  <th rowspan=2><th colspan=2>average<th rowspan=2>other<br>category<tr>
  <th>height <th>weight <tr>
  <th align=left>males <td>1.9 <td>0.003 <td>yyy <tr>
  <th align=left>females <td>1.7 <td>0.002 <td>xxx
</table>
```

The first cell (a header cell) is merged with the cell below it: <th rowspan=2>. Note that this merged cell is empty - the definition of the next column for the first row starts immediately. Looking again at the first row, the second column is merged with the third: <th colspan=2>. The definition for the third column is skipped as it was covered by the merged cell. The fourth column/first row is also merged, this time with the next row: <th rowspan=2>. The
 element has been used here to force a line break between other and category. The <TR> element signifies the end of the first row and the beginning of the second. Note that empty cells at the end of a row can be omitted as the `<TR>` element unambiguously marks the end of the row.

The second row only contains definitions for the second and third columns since the others were merged with cells on the preceding row. The general rule is to avoid defining any cell twice. The last two rows start with headers and the `align=left` attribute ensures that the browser will align these headers to the left of their cells. The `ALIGN` attribute can be one of `LEFT`, `CENTER` or `RIGHT`, with `CENTER` as the default. It can be used with both `TH` and `TD`.

9.1 Implementation Issues for Tables

Browsers need a prepass through the table markup to count the number of columns and determine their widths. A simple algorithm that takes merged cells into account will suffice. Text fields wrap to fit their columns, which should be sized to best match current window width. This information should be cached to avoid speed penalties during subsequent screen refresh/window resize operations. Browsers can ignore alignment hints if required, and using a fixed pitch font may speed up the sizing step.

The number of columns is given by the row with the largest number of `<TH>` and `<TD>` elements, remembering to add in merged cells. The widths of columns are evaluated by finding the minimum and maximum widths needed for each cell, and hence the minimum and maximum width for the column as a whole. All this can be done during a single pass through the `<TABLE>` element. Caching these min/max values for each column then permits the browser to instantly adjust the table when the window is resized.

10 Fill-out Forms and Input fields

Forms are composed by placing input fields within paragraphs, preformatted/literal text, lists and tables. This gives considerable scope in designing the layout of forms. Each field is defined by an `INPUT` element and must have an `NAME` attribute which uniquely names the field in the document. Additional optional attributes can be used to specify the type of the field (defaults to free text), its size/precision, its initial value and whether the field is currently disabled or in error:

```
<FORM ACTION="mailto:www_admin@info.cern.ch">
   <MH HIDDEN>Subject: WWW Questionaire</MH>

   Please help up to improve the World Wide Web by filling in the
   following questionaire:
   <P>Your organization? <INPUT NAME="org" SIZE="48">
   <P>Commercial? <INPUT NAME="commerce" TYPE=checkbox>
      How many users? <INPUT NAME="users" TYPE=int>
   <P>Which browsers do you use?
   <OL>
      <LI>X Mosaic <INPUT NAME="browsers" TYPE=checkbox VALUE="xmosaic">
      <LI>Cello <INPUT NAME="browsers" TYPE=checkbox VALUE="cello">
      <LI>Others <TEXTAREA NAME="others" COLS=48 ROWS=4></TEXTAREA>
   </OL>
   A contact point for your site: <INPUT NAME="contact" SIZE="42">
   <P>Many thanks on behalf of the WWW central support team.

   <P ALIGN=CENTER><INPUT TYPE=submit> <INPUT TYPE=reset>
</FORM>
```

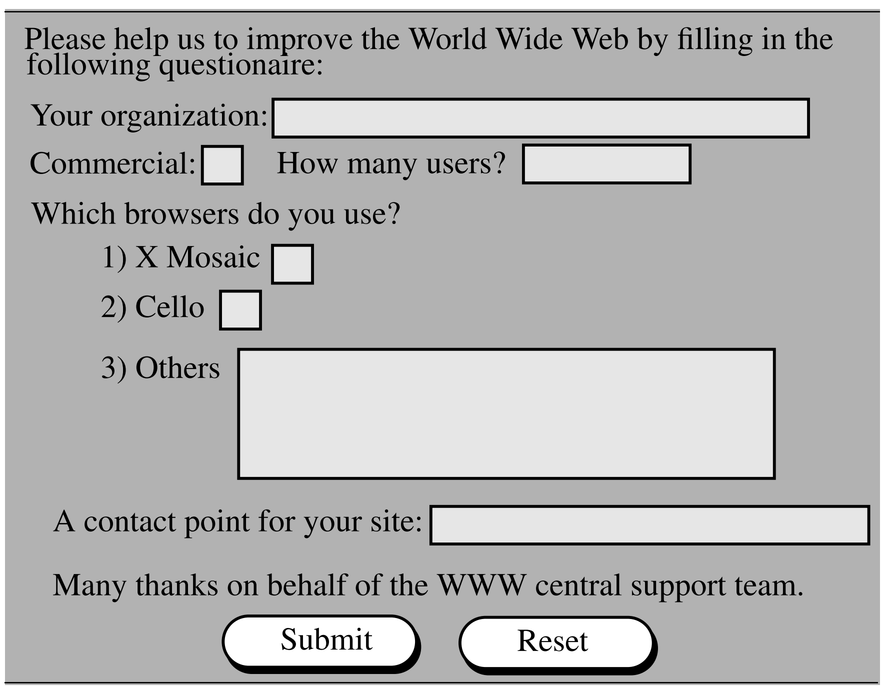
This fictitious example is a questionnaire that will be emailed to `www_admin@info.cern.ch`. The `FORM` element is used to delimit the form. There can be several forms in a single document, but the `FORM` element can't be nested. The `ACTION` attribute specifies a URL that designates an HTTP server or an email address. If missing, the URL for the document itself will be assumed. The effect of the action can be modified by including a method prefix, e.g. `ACTION="POST http://...."`. This prefix is used to select the HTTP method when sending the form's contents to an HTTP server. Would it be cleaner to use a separate attribute, e.g. `METHOD`?

Servers can disable forms by sending an appropriate header or by an attribute on the optional `HTMLPLUS` element at the very start of the document, e.g. `<htmlplus forms=off>`.

The `MH` element can be used to specify RFC 822 mail headers that are included when sending the form's content either as an email message or as a HTTP request, e.g.

```
<MH HIDDEN>
Subject: WWW Questionnaire
Priority: Low
</MH>
```

The `MH` element can contain several headers separated by line breaks. The text within the MH element is transcribed directly[1], preserving spaces, tabs and line breaks, with each line terminated by a `CR LF` pair as per the RFC 822 guidelines. The preceding example of a form might be rendered as:



Here, the `<P>` and `<OL>` elements have been used to lay out the text (and input fields. The browser has changed the background color within the `FORM` element to distinguish the form from other parts of the document. The browser is responsible for handling the input focus, i.e. which field will currently get keyboard input.

1. Except for a initial line break which is ignored.

For many platforms there will be existing conventions for forms, e.g. *tab* and shift-*tab* keys to move the keyboard focus forwards and backwards between fields, while an *Enter* key submits the form. In the example, the *Submit* and *Reset* buttons are specified explicitly with special purpose fields. The *Submit* button is used to email the form or send its contents to the server as specified by the `ACTION` attribute, while the *Reset* button resets the fields to their initial values. When the form consists of a single text field, it may be appropriate to leave such buttons out and rely on the *Enter* key.

The `INPUT` element has the following attributes:

`NAME`	Symbolic name used when transferring the form's contents. This attribute is always needed and should uniquely identify this field.
`TYPE`	Defines the type of data the field accepts. Defaults to free text.
`SIZE`	Specifies the size or precision of the field according to its type.
`MAXLENGTH`	The maximum number of characters that will be accepted as input. This can be greater that specified by `SIZE`, in which case the field will scroll appropriately. The default is unlimited.
`VALUE`	The initial value for the field, or the value when checked for checkboxes and radio buttons. This attribute is required for radio buttons.
`CHECKED`	When present indicates that a checkbox or radio button is selected.
`DISABLED`	When present indicates that this field is temporarily disabled. Browsers should show this by "greying it" out in some manner.
`ERROR`	When present indicates that the field's initial value is in error in some way, e.g. because it is inconsistent with the values of other fields. Servers should include an explanatory error message with the form's text.
`SRC`	A URL or URN specifying an image - for use only with `TYPE=IMAGEMAP`.
`ALIGN`	Vertical alignment of the image - for use only with `TYPE=IMAGEMAP`.

The following types of fields can be defined with the `TYPE` attribute *(upper or lower case)*:

`TEXT`	Single line text entry fields. Use the `SIZE` attribute to specify the visible width in characters, e.g. `SIZE="24"` for a 24 character field. The `MAX` attribute can be used to specify an upper limit to the number of characters that can be entered into a text field, e.g. `MAX=72`. Use the `TEXTAREA` element for text fields which can accept multiple lines (see below).
`INT`	For entering integer numbers, the maximum number of digits can be specified with the `SIZE` attribute (excluding the sign character), e.g. `size=3` for a three digit number.
`FLOAT`	For fields which can accept floating point numbers.
`DATE`	Fields which can accept a recognized date format.
`URL`	For fields which expect document references as URLs or URNs.
`CHECKBOX`	Used for simple Boolean attributes, or for attributes which can take multiple values at the same time. The latter is represented by a number of checkbox fields each of which has the same `NAME`.

`RADIO` For attributes which can take a single value from a set of alternatives. Each radio button field in the group should be given the same `NAME`.

`RANGE` This allows you to specify an integer range with the `MIN` and `MAX` attributes, e.g. `MIN=1 MAX=100`. Users can select any value in this range.

`IMAGE` This allows you to specify an image field upon which you can click with a pointing device. The `SRC` and `ALIGN` attributes are exactly the same as for the `IMG` and `IMAGE` elements. The symbolic names for the x and y coordinates of the click event are specified with *name*.x and *name*.y for the *name* given with the `NAME` attribute. The `VALUE` attribute is ignored.

`SCRIBBLE` A field upon which you can write with a pen or mouse. The size of the field in millimeters is given as `SIZE=`*`width`*`,`*`height`*. The units are absolute as they relate to the dimensions of the human hand, rather than pixels of varying resolution. The scribble may involve time and pressure data in addition to the basic ink data. You can use scribble for signatures or sketches. The field can be initialised by setting the `SRC` attribute to a URL which contains the ink[1]. The `VALUE` attribute is ignored.

`AUDIO` This provides a way of entering spoken messages into a form. Browsers might show an icon which when clicked pops-up a set of tape controls that you can use to record and replay messages. The initial message can be set by specifying a URL with the `SRC` attribute. The `VALUE` attribute is ignored.

`SUBMIT` This is a button that when pressed submits the form. It offers authors control over the location of this button. You can use an image as a submit button by specifying a URL with the `SRC` attribute.

`RESET` This is a button that when pressed resets the form's fields to their initial values as specified by the `VALUE` attribute. You can use an image as a reeset button by specifying a URL with the `SRC` attribute.

When you need to let users enter more than one line of text, you should use the TEXTAREA element, e.g.

```
<TEXTAREA NAME="address" ROWS=64 COLS=6>
   Hewlett Packard Laboratories
   1501 Page Mill Road
   Palo Alto, California 94304-1126
</TEXTAREA>
```

The text up to the end tag is used to initialize the field's value. This end tag is always required even if the field is initially blank. The `ROWS` and `COLS` attributes determine the visible dimension of the field in characters. Browsers are recommended to allow text to grow beyond these limits by scrolling as needed. In the initial design for forms, multi-line text fields were supported by the `INPUT` element with `TYPE=TEXT`. Unfortunately, this causes problems for fields with long text values as SGML limits the length of attribute literals. The HTML+ DTD allows for up to 1024 characters (the SGML default is only 240 characters!).

The `RADIO` and `CHECKBOX` fields can be used to specify multiple choice forms in which every

1. Ink is commonly transferred in the JOT format defined by Slate, Lotus, GO, Microsoft, Apple, General Magic and others. Phone Slate Technical Support on +1 (602) 991-6844 for more information.

alternative is visible as part of the form. An alternative is to use the SELECT element which is generally rendered in a more compact fashion as a pull down combo list. Every alternative is represented by the OPTION element, e.g.

```
<SELECT NAME="flavor">
   <OPTION>Vanilla
   <OPTION>Strawberry
   <OPTION>Rum and Raisin
   <OPTION>Peach and Orange
</SELECT>
```

The SEVERAL attribute is needed when users are allowed to make several selections, e.g. `<SELECT SEVERAL>`. The ERROR attribute can be used to indicate that the initial selection is in error in some way, e.g. because it is inconsistent with the values of other fields.

The OPTION element can take the following attributes:

SELECTED Indicates that this option is initially selected.

DISABLED When present indicates that this option is temporarily disabled. Browsers should show this by "greying it" out in some manner.

10.1 Sending form data to an HTTP server

The form contents are expressed as a property list of attribute names and values. Radio buttons and checkboxes are left out of the list when unchecked. This ensures that only the selected radio button contributes a *name=value* pair. Omitting the VALUE attribute for a checkbox field causes the field when checked to appear as a *name* without a value (this is appropriate for Boolean attributes). Currently, there are two ways of transferring form contents to an HTTP server:

- As a suffix on the URL given by the ACTION attribute
- As a multipart MIME message

In the first approach, the property list is encoded as a sequence of *name=value* elements separated by the "&" character. The values for the form's fields are sent as a search string, e.g.

```
URL?org=Acme%20Foods&commerce&users=42
```

Note that "=", "&" and space characters in attribute names and values should be escaped by "%" followed by the hexadecimal code for the character in question, e.g. "%20" should be used in place of the space character. IMAGE fields are only included in the list when clicked, and give rise to something matching:

```
URL?name.x=23&name.y=29
```

They can be used as iconic controls for other images or data. The *object-name*.*property-name* notation paves the way for more complex input controls in the future. In the near future, format negotiation will not change the number of pixels in an image, so using pixel based coordinates is okay. In the longer term, scaled coordinates in the range 0 to 1.0 may prove safer.

Multipart MIME messages are necessary if the form contains scribble or audio fields. Form data can be sent in the same *name=value* representation as described above. For scribble and audio fields, the *value* identifies a subsequent part in the multipart message, as specified by the `Content-ID:` header for each part. Another approach is to send just the SGML elements used to define form fields, i.e. the INPUT, TEXTAREA and SELECT elements. The `Content-ID:` headers are used to define dummy URLs. It may be possible for servers to use this approach to update forms being viewed by a browser without having to send the entire document.

10.2 Sending a form via Electronic Mail

In this case, the form needs to be viewable on ordinary mail readers. The form should be converted to ASCII and mailed as a plain text message, preceded by the headers as specified by the `MH` element. Each `INPUT` field is copied as text and delimited by the "[" and "]" characters. An longer term alternative is to send the HTML+ document as a MIME message, along with the current values for each field.

11 Literal and Preformatted Text

Preformatted text started off in HTML with a simple mechanism for showing computer output, for which the spaces and line breaks were significant in determining the layout. The desire to offer Unix manual pages as hypertext forced a rethink. The next version supported character emphasis and embedded hypertext buttons. HTML+ adds the capability to use variable pitch fonts and to set up tab stops.

The LIT element is rendered in a proportional font, e.g.

```
<LIT>
From Oberon in fairyland,
   The king of ghosts and shadows there,
Mad Robin I, at his command,
   Am sent to view the night sports here.
      What revel rout
      is kept about,
   In every corner where I go,
      I will o'ersee
      And merry be
   And make good sport, with ho, ho, ho!
</LIT>
```

This is rendered literally as:

From Oberon in fairyland,
    The king of ghosts and shadows there,
Mad Robin I, at his command,
    Am sent to view the night sports here.
        What revel rout
        Is kept about,
    In every corner where I go,
        I will o'ersee
        And merry be
    And make good sport, with a ho, ho, ho!

The ability to set tab stops in `LITeral` text makes it much easier to write filters that convert documents written on word processors into HTML+. Tab stops can be set by the `TAB` element and apply for the scope of the `LIT` element, e.g.

```
<tab at=40 align=right>
```

The `AT` attribute specifies the position of the tab stop, as measured from the left margin in terms of the width of a capital M.[1] The `ALIGN` attribute can be `LEFT`, `CENTER` or `RIGHT`, defaulting to `LEFT`. These have the conventional meaning as used on most word processors. If greater control over fonts and layout is needed then authors should make a hypertext link to a document written in a page description format like Adobe's PDF.

For computer output or plain text files, you should use the PRE element which is rendered in a fixed pitch font. The following is part of the man page for the Unix ls command:

```
<PRE>
 The next 9 characters are interpreted as three sets of three
 bits each which identify access permissions for owner,
 group, and others as follows:

 +----------------- 0400 read by owner (<B>r</B> or <B>-</B>)
 | +--------------- 0200 write by owner (<B>w</B> or <B>-</B>)
 | | +------------- 0100 execute (search directory) by owner
 | | |                   (<B>x</B>, <B>s</B>, <B>S</N>, or <B>-</B>)
 | | | +----------- 0040 read by group (<B>r</B> or <B>-</B>)
 | | | | +--------- 0020 write by group (<B>w</B> or <B>-</B>)
 | | | | | +------- 0010 execute/search by group
 | | | | | |             (<B>x</B>, <B>s</B>, <B>S</B>, or <B>-</B>)
 | | | | | | +----- 0004 read by others (<B>r</B> or <B>-</B>)
 | | | | | | | +--- 0002 write by others (<B>w</B> or <B>-</B>)
 | | | | | | | | +- 0001 execute/search by others
 | | | | | | | | |       (<B>x</B>, <B>t</B>, <B>T</B>, or <B>-</B>)
 | | | | | | | | |
 r w x r w x r w x

</PRE>
```

This is rendered as

```
 The next 9 characters are interpreted as three sets of three
 bits each which identify access permissions for owner,
 group, and others as follows:

 +----------------- 0400 read by owner (r or -)
 | +--------------- 0200 write by owner (w or -)
 | | +------------- 0100 execute (search directory) by owner
 | | |                   (x, s, S, or -)
 | | | +----------- 0040 read by group (r or -)
 | | | | +--------- 0020 write by group (w or -)
 | | | | | +------- 0010 execute/search by group
 | | | | | |             (x, s, S, or -)
 | | | | | | +----- 0004 read by others (r or -)
 | | | | | | | +--- 0002 write by others (w or -)
 | | | | | | | | +- 0001 execute/search by others
 | | | | | | | | |       (x, t, T, or -)
 | | | | | | | | |
 r w x r w x r w x
```

1. Chosen as one of the widest in most proportional fonts and should produce good results in most cases.

The SEE ALSO section of the Unix manual pages can be processed to make references to other manual pages into hypertext buttons using the <A> element (see section 5.2). The example shows how character emphasis can be added to literal or preformatted text.

12 Mathematical Equations

Currently, the best way of including equations in HTML documents is to first write the document in *LaTeX* and then use the *latex2html* filter to create the corresponding HTML document, together with the equations as a number of bitmap files[1]. The previous draft of the HTML+ specification described a way of embedding LaTeX equations in HTML+ documents. Unfortunately, it now seems too cumbersome to form a practical solution, and has been dropped.

The following is a preliminary proposal for representing equations directly as HTML+ using an SGML-based notation, inspired by the approach taken by LaTeX. It is intended to cover the majority of users needs, rather than aiming for complete coverage. This makes it practical to use a simplified notation compared with richer notations, e.g. the ISO 12083 Maths DTD. An experimental browser supporting the MATH element is being developed at CERN.

Consider the equation:

$$H(s) = \int_0^\infty e^{-st} h(t)\, dt$$

This can be represented as:

```
<math>
   H(s) = &int;<sub>0</sub><sup>&infin;</sup> e<sup>-st</sup> h(t) dt
</math>
```

The mathematical symbols are given with their standard ISO entity names. SUB and SUP are used to specify subscripts and superscripts. For integral signs and related operators, the subscript/superscript text is centered over the symbol, otherwise it appears to the right as shown in the preceding example. The BOX and OVER elements allow you to define more complex equations, as in:

$$C \frac{dV_{out}}{dt} = I_b \tanh\left(\frac{\kappa(V_{in} - V_{out})}{2}\right)$$

which is represented by:

```
<math>
   C <box>dV<sub>out</sub><over>dt</box> = I<sub>b</sub>
   &tanh;(<box>&kappa;(V<sub>in</sub>-V<sub>out</sub>)<over>2</box>)
</math>
```

1. When including formulae as inline images, it is worth giving textual equivalents of the formulae for clients with text only displays, i.e. the *LaTeX* expressions, or the *Mathematica* equivalent.

The `BOX` element can be used to generally group items and can be thought of as non-printing parentheses. The `OVER` element is optional and divides the box into numerator and denominator. The `ARRAY` element is used to specify arrays for expressions like:

$$\left(\begin{array}{c} \left|\begin{array}{cc} x_{11} & x_{12} \\ x_{21} & x_{22} \end{array}\right| \\ y \\ z \end{array}\right)$$

The `ARRAY` element has a single attribute `ALIGN` which specifies the number of columns and the alignment of items within the columns. For each column there is a single letter that specifies how items in that column should be positioned: `c` for centered, `l` for flush left or `r` for flush right. Each item in the array must follow an `<ITEM>` element.

The preceding example is represented by:

```
<math>
   (<array align="c"> <item>
      &ldet;<array align="cc">
         <item>x<sub>11</sub>
         <item>x<sub>12</sub>
         <item>x<sub>21</sub>
         <item>x<sub>22</sub>
      </array><rd>&rdet;
      <item> y <item> z
   </array>)
</math>
```

The browser is responsible for working out the vertical and horizontal spacing required for the array. Parentheses[1] are stretched to match the size of the array. Arrays can be used only in the context of the `MATH` element. The `TABLE` element should be used for other contexts.

Spaces are significant within the `MATH` element, and used for disambiguation, as can be seen in the following two examples:

$$\int\limits_{i}^{j} x$$

```
&int;<sub>i</sub><sup>j</sup> x
```

$$\int_{i}^{j} x$$

```
&int; <sub>i</sub><sup>j></sup>x
```

Authors can adjust the default horizontal spacing with the ISO entities: ` ` for thin space (1/6 em) and ` ` for hair space. Horizontal, diagonal and vertical ellipsis are possible with `…` `&dellip;` and `⋮` respectively. Common functions like `sin`, `log` and `tanh` should be rendered in a non-italic font. These functions are defined by their entity namesakes. Additional elements are needed to represent roots and for over and under lining.

1. These are { } [ ] ( ) and symbols from “ISO 8879-1986//ENTITIES Added Math Symbols: Delimiters//EN”. Note that `&ldet;` and `&rdet;` are used when the “|” symbol is used for parentheses.

An open question is how to render maths on dumb terminals. One approach is to translate into an existing convention such as *Mathematica*. Another is to write equations as they would be spoken aloud. For GUI displays, browsers need to be able to show characters in at least two point sizes as well as being able to stretch parentheses and integral signs etc. to various sizes. The processing time needed to size and position symbols suggests that caching may be useful to speed up subsequent scrolling and refresh operations.

Comments from mathematicians are welcomed. Widespread support for formulae is likely to be delayed until most platforms support the relevant symbols fonts (or Unicode).

13 Indexing

A good index plays an important role in helping you find your way to the material you need. It allows you to type in one or more keywords to see a list of matching topics. The ability to view an index directly allows you to gain a feeling for what is covered, and lets you dip in and out of the associated document. Full text indexes like WAIS are easy to create, but don't give you this flexibility since the index itself cannot be viewed directly.

Generating a conventional index for a document is a skilled task, and HTML+ allows authors to include directives for automatically creating hypertext indexes. These directives can be included in many HTML+ elements, such as headers, paragraphs and character emphasis using the `INDEX` attribute. This allows each such element to be referenced in the index under primary or secondary keys, e.g.

```
<h3 id="z23" index="Radiation damage/shielding from as difficult">Radiation shielding</h3>
```

This can be used to generate an index like:

Radiation damage
- classical target theory
- dominance of
- in molecular mills
- shielding from as difficult
- simple lifetime model
- track-structure lifetime model

Radicals
- *and so on ...*

In many cases, a given key will be associated with more than one part of the document. In this case you can either use secondary keys to disambiguate the references, as shown above, or allow the indexing program to generate its own names for each reference, e.g. (a), (b), (c), ...

The indexing program creates an HTML+ file that can then be linked to the documents it was produced from. The program may also generate a list of references from occurrences of the `CITE` element. These can be simply ordered alphabetically. Sophisticated bibliographic references are beyond the scope of HTML+ as they require a much richer system of markup.

14 Document declarations

It is recommended that HTML+ documents start with the following external identifier[1], indicating that the document conforms to the HTML+ DTD. This will ensure that other SGML

1. The official identifier will be defined when HTML+ reaches the status of an official standard.

parsers can process HTML+ documents, without needing to include the DTD with each document.

```
<!DOCTYPE htmlplus PUBLIC "-//Internet/RFC xxxx//EN">
```

There are several elements that can only occur at the start of the document before any headers or text elements:

14.1 HTMLPLUS

This element if present must follow immediately after the `DOCTYPE` declaration. It can be used to disable form filling:

```
<htmplus forms=off>
```

Another idea is to provide a `VERSION` attribute for specifying the version number of HTML+ in used by this document. This would provide an alternative to including the version number in the public name given with the `DOCTYPE` element.

14.2 The HEAD and BODY elements

These may be used to delimit the document declarations and document body with the HEAD and BODY elements respectively, e.g.

```
<HEAD>
   <ISINDEX>
   <LINK REL="Next" HREF="...">
   etc.
</HEAD>

<BODY>
   body elements go here
</BODY>
```

14.3 TITLE

This element is used to define the title of the current document. and is often used as the window banner for window-based displays. There may be only one title in any node, and it should identify the content of the node in a fairly wide context. No markup is permitted within title text, although character entity references may be used for accented characters etc.

14.4 ISINDEX

The `ISINDEX` element specifies that the URL given with the `HREF` attribute is *searchable*, e.g. `<ISINDEX HREF="glossary.html">`. If the `HREF` attribute is missing, the URL for this document is assumed. Servers may also indicate that the current document is *searchable* via the HTTP headers returned with the document. Browsers should allow users to enter a search string of one or more keywords. When the user presses the *Return* key, the browser maps any spaces to "+", and appends this string to the designated URL and sends it to the server, e.g.

```
URL?word+word+word
```

This mechanism has to a large extent been superseded by the `FORM` element. There are still good reasons for keeping it in HTML+. In particular, when reading a long document, having the search field always visible, makes it much easier for people to enter search strings, than if they first had to scroll to the part of the document which included a search form.

14.5 NEXTID

The `NEXTID` element is used by browsers that automatically generate identifiers for anchor points. It specifies the next identifier to use, to avoid possible confusion with older (deleted) values, e.g. `<nextid n="id56">`. The identifier should take the form of zero or more letters followed by one or more digits. The numeric suffix should be incremented to generate successive identifiers.

14.6 BASE

The `HREF` attribute of the `BASE` element gives the full URL of the document, and is added by the browser when the user makes a local copy. Keeping the full URL is essential when subsequently viewing the copied document as it allows relative URLs to be resolved to their original references, e.g. `<BASE HREF=`*`URL`*`>`.

14.7 LINK

This provides a means of describing the relationship between this document and other documents, and has the same attributes as the `<A>` element (see section 5.2). A document can have multiple `LINK` elements. Typical uses are to indicate authorship, related indexes and glossaries, older or more recent versions etc. Another use is to indicate a stylesheet that contains the author's layout preferences, e.g. for headers and multi-columns displays. Links can also be used to indicate a static tree structure of documents with relationships such as "parent", "next" and "previous", e.g. `<LINK HREF=`*`URL`*` REL="next">`

The standard values for the `REL` attribute are *(case insensitive):*

`UseIndex` The linked document can be used as an index for this document. There may be several such indexes. The `TITLE` attribute should be used to name each index, e.g. in menus and dialog boxes. This relationship implies the document is *searchable*, and the browser should provide a means for users to type in one or more keywords. The index may be a full text WAIS index or a conventional hypertext-based index.

`UseGlossary` The linked document can be used to answer glossary queries for this document. Typically invoked by a double click on a word[1], or by drag selection, followed by clicking a menu item. There may be several such indexes. The `TITLE` attribute should be used to name each index, e.g. in menus and dialog boxes.

`Contents` The linked document acts as a contents page for a number of related documents. The browser should make this available as a button on a toolbar or as an entry in a navigation menu. The TITLE attribute can be used to override the default "Contents" name.

`Next` The linked document is next on a path of documents. Browsers should make this available as a button on a toolbar or as an entry in a navigation menu.

1. This may cause problems when the user double clicks a word which is part of a hypertext link.

`Previous` The linked document is the previous one to the current document on a path of documents. Browsers should make this available as a button on a toolbar or as an entry in a navigation menu.

`Parent` The linked document is at the next level up in a hierarchy of documents. Browsers should make this available as a button on a toolbar or as an entry in a navigation menu. There may be several such parents. The `TITLE` attribute should be used to name each such document.

`Bookmark` The URL specified by the `HREF` attribute is a bookmark, which is named by the `TITLE` attribute. Browsers should make these available as buttons on a toolbar or as entries in a navigation menu.

`Made` Defines who is the “maker” of this document. The HREF attribute should give an appropriate URL e.g. “mailto:dsr@hplb.hpl.hp.com”. Browsers can use this to allow people to mail or post comments to the author of the document.

`Help` Associates a help document with this node.

Sometimes it may be useful to specify a hypertext link separately from the text associated with the start of the link. For example a sidebar could be associated with a given paragraph as follows:

```
<LINK IDREF=”z36” REL=”Sidebar” HREF=”sidebar.html”>
```

The `IDREF` attribute localizes the link to an element in the current document with a specific identifier (as defined with the `ID` attribute). In the absence of the `IDREF` attribute, the link is associated with the current document as a whole.

Other suggestions for `LINK` currently lie outside this proposal, pending further work. In some cases, `LINK` elements may be implied by the context in which this document was reached. This is explained in section 15.

15 Dealing with Large Documents

Many classic works are available over the Internet, now that their copyright has expired. Downloading these as large documents is time consuming, and a better strategy is to split them up into smaller pieces. Other people have lots of paper documents and wish to make them available electronically. While it is easy to scan these documents in, the size of the images makes them tedious to transfer over the network. Once again, time can be saved by avoiding the need to download the whole document at once. HTML+ makes it easy to do this with explicit or implicit links between the pieces that make up the complete document.

A book might have the following pieces:

- ‘Cover page’
- About the author
- Copyright and publishing details
- Table of contents
- Foreword
- Preface

- Acknowledgement
- One or more chapters
- One or more appendices
- Bibliography
- Glossary
- Index

Each of these could be held as separate HTML+ subdocuments. The table of contents should obviously include hypertext links to other parts of the book rather than page numbers. You can define a linear sequence through each of these subdocuments by including `LINK` elements with `REL=NEXT` and `REL=PREVIOUS`. This will allow readers to read through each part of the book in turn. You should also include LINKs to the table of contents (`REL=CONTENTS`) and other key parts (using `REL=BOOKMARK`).

Generating a hypertext version of the index may prove time consuming, and it may be simpler to offer a full text search facility instead. The `INDEX` attribute can be used with many HTML+ elements to facilitate automatic generation of a conventional looking index, see section 13.

Implicit links are useful when you want to reuse a given subdocument in another independent book, and for non-HTML+ formats such as scanned page images. To define implicit links, you need to first create a HTML+ document such as a table of contents, and to make each entry into a hypertext link using the `<A>` element with the attribute `REL="SUBDOCUMENT"`. When the user follows one of these links, the browser scans the current document to locate the next `<A>` element with the subdocument relationship. If it reaches the end of the document it looks for a `LINK` element with `REL=NEXT`. This procedure is used to imply a `LINK` element in the retrieved subdocument. A similar process is used to imply a `LINK` element with `REL=PREVIOUS`. The other links for the current document are simply inherited, i.e. any bookmarks, glossary or index links that hold for the table of contents, also hold for the subdocument.

The browser then retrieves the subdocument and merges the implied LINKs with any that are given explicitly[1]. If the user now presses the "Next" button on the toolbar (or menu), the browser follows the implicit link to the next subdocument. The browser needs to look again at the parent document to find the new *next* subdocument. This mechanism is difficult to explain, but simple to write documents for. All that authors need to do, is to remember to include the subdocument relationship when defining hypertext links.

For a hundred page scanned document where each page is held as a separate file, the "table of contents" is going to be pretty dull, and there is little point creating it as an HTML+ node. Instead, you should use an HTTP server which passes the missing `LINK` elements as header fields for each page image. The suggested representation for these header fields uses the same attributes and syntax as the `LINK` element:

```
WWW-Link: REL="Next" HREF="http://info.cern.ch/...."
```

There could be several `WWW-Link:` headers, one for each implied `LINK`. This idea puts the burden on the server to supply such links as appropriate to each requested document.

1. An explicit `LINK` for the *next* or *previous* document overrides an implicit one.

16 Acknowledgements

I would like to thank the many people on the *www-talk* mailing list who have contributed to the design of HTML+ and to the management of HP Labs for their support during this work.

David Raggett, Hewlett Packard Laboratories, October 1993.

17 References

"*Hypertext Markup Language (HTML)*", Tim Berners-Lee, January 1993.
URL=ftp://info.cern.ch/pub/www/doc/html-spec.ps
or http://info.cern.ch/hypertext/WWW/MarkUp/MarkUp.html

"*Uniform Resource Locators*", Tim Berners-Lee, January 1992.
URL=ftp://info.cern.ch/pub/www/doc/url7a.ps
or http://info.cern.ch/hypertext/WWW/Addressing/Addressing.html

"*Protocol for the Retrieval and Manipulation of Texual and Hypermedia Information*",
Tim Berners-Lee, 1993.
URL=ftp://info.cern.ch/pub/www/doc/http-spec.ps
or http://info.cern.ch/hypertext/WWW/Protocols/HTTP/HTTP2.html

"*The SGML Handbook*", Charles F. GoldFarb, pub. 1990 by the Clarendon Press, Oxford.

Appendix I

The HTML+ Document Type Definition (DTD). The preliminaries are taken from the HTML DTD and declares the character set as Latin-1, disables markup minimisation and sets limits for tag/attribute names to 34 characters, and attribute values to a maximum of 1024 characters.

```
<!SGML  "ISO 8879:1986"
--
Document Type Definition for the HyperText Markup Language Plus
for use with the World Wide Web application (HTML+ DTD). These
initial settings are take from the HTML DTD.

        NOTE: This is a definition of HTML with respect to
        SGML, and assumes an understanding of SGML terms.
--
CHARSET
        BASESET "ISO 646:1983//CHARSET
                 International Reference Version (IRV)//ESC 2/5 4/0"
        DESCSET 0    9   UNUSED
                9    2   9
                11   2   UNUSED
                13   1   13
                14  18   UNUSED
                32  95   32
               127   1   UNUSED
        BASESET "ISO Registration Number 100//CHARSET
                 ECMA-94 Right Part of Latin Alphabet Nr. 1//ESC 2/13 4/1"
        DESCSET 128   32  UNUSED
                160   95  32
                255    1  UNUSED

CAPACITY        SGMLREF
                TOTALCAP        150000
                GRPCAP          150000

SCOPE   DOCUMENT
SYNTAX
        SHUNCHAR CONTROLS  0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18
                           19 20 21 22 23 24 25 26 27 28 29 30 31 127 255
        BASESET "ISO 646:1983//CHARSET
                 International Reference Version (IRV)//ESC 2/5 4/0"
        DESCSET 0 128 0
        FUNCTION RE         13
                 RS         10
                 SPACE      32
                 TAB SEPCHAR 9
        NAMING   LCNMSTRT ""
                 UCNMSTRT ""
                 LCNMCHAR ".-"
                 UCNMCHAR ".-"
                 NAMECASE GENERAL YES
                          ENTITY  NO
```

```
        DELIM    GENERAL  SGMLREF
                 SHORTREF SGMLREF
        NAMES    SGMLREF
        QUANTITY SGMLREF
                 NAMELEN  34
                 TAGLVL   100
                 LITLEN   1024
                 GRPGTCNT 150
                 GRPCNT   64

FEATURES
  MINIMIZE
    DATATAG  NO
    OMITTAG  NO
    RANK     NO
    SHORTTAG NO
  LINK
    SIMPLE   NO
    IMPLICIT NO
    EXPLICIT NO
  OTHER
    CONCUR  NO
    SUBDOC  NO
    FORMAL  YES
  APPINFO NONE
>
<!DOCTYPE HTMLPLUS [
<!-- DTD for HTML+
Markup minimisation should be avoided, otherwise the default <!SGML> decla-
ration is fine. Browsers should be forgiving of markup errors.

Common Attributes:

id      the id attribute allows authors to name elements such as headers
        and paragraphs as potential destinations for links. Note that
        links don't specify points, but rather extended objects.
index   allows authors to specify how given headers etc should be
        indexed as primary or secondary keys, where "/" separates primary
        from secondary keys, ";" separates multiple entries
-->
<!-- ENTITY DECLARATIONS
 <!ENTITY % foo "X | Y | Z"> is a macro definition for parameters and in
 subsequent statements, the string "%foo;" is expanded to "X | Y | Z"

 Various classes of SGML text types:

  #CDATA  text which doesn't include markup or entity references
  #RCDATA text with entity references but no markup
  #PCDATA text occurring in a context in which markup and entity references
     may occur.
-->
<!ENTITY % URL "CDATA" -- a URL or URN designating a hypertext node -->
<!ENTITY % emph1 "I|B|U|S|SUP|SUB|TT">
<!ENTITY % emph2 "Q|CITE|PERSON|ACRONYM|ABBREV|EM|STRONG">
<!ENTITY % emph3 "CMD|ARG|KBD|VAR|DFN|CODE|SAMP|REMOVED|ADDED">
```

```
<!ENTITY % emph "%emph1;|%emph2;|%emph3;">
<!ENTITY % misc "RENDER|FOOTNOTE|MARGIN|INPUT|TEXTAREA|SELECT">
<!ENTITY % text "#PCDATA|A|IMG|IMAGE|%emph;|%misc;|BR|CHANGED">
<!ENTITY % paras "P|PRE|LIT|FIG">
<!ENTITY % lists "UL|OL|DL|MENU|DIR">
<!ENTITY % block "TABLE|FORM|MATH|NOTE|QUOTE|ABSTRACT|BYLINE|HR">
<!ENTITY % heading "H1|H2|H3|H4|H5|H6">
<!ENTITY % table "%text;|P|%heading;|%lists;">
<!ENTITY % math "BOX|%text;">
<!ENTITY % main "%heading;|%block;|%lists;|%paras;|%text;">
<!ENTITY % setup "(TITLE? & ISINDEX? & NEXTID? & LINK* & BASE?)">

<!-- Basic types of elements:
  <!ELEMENT tagname - - CONTENT> elements needing end tags
  <!ELEMENT tagname - O CONTENT> elements with optional end tags
  <!ELEMENT tagname - O EMPTY> elements without content or end tags

The content definition is:
       -  an entity definition as defined above
       -  a tagname
       -  (brackets enclosing the above)
These may be combined with the operators:
  A*      A occurs zero or more times
  A+      A occurs one or more times
  A|B     implies either A or B
  A?      A occurs zero or one times
  A,B     implies first A then B
  A&B     either or both A and B (in either order)
-->

<!ELEMENT HTMLPLUS O O ((HEAD, BODY) | ((%setup;), (%main;)*))>
<!ATTLIST HTMLPLUS
        version CDATA #IMPLIED -- the HTML+ version number --
        forms   (on|off) on    -- used to disable form filling -->

<!ELEMENT HEAD - - (%setup;) -- delimits document wide properties -->
<!ELEMENT BODY - - (%main;)* -- delimits the document's body -->

<!-- Document title -->
<!ELEMENT TITLE - - (#PCDATA | %emph;)+>
<!ATTLIST TITLE
        id      ID      #IMPLIED -- link destination --
        lang    CDATA   #IMPLIED -- ISO language abbreviation --
        index   CDATA   #IMPLIED -- entries for index compilation -->

<!-- Document headers -->
<!ELEMENT (%heading;) - - (#PCDATA | %emph;)+>
<!ATTLIST (%heading;)
        id      ID      #IMPLIED -- defines link destination --
        lang    CDATA   #IMPLIED -- ISO language abbreviation --
        index   CDATA   #IMPLIED -- entries for index compilation -->

<!-- character emphasis -->
<!ELEMENT (%emph;) - - (%text;)*>
<!ATTLIST (%emph;)
```

```
        id      ID      #IMPLIED -- link destination --
        lang    CDATA   #IMPLIED -- ISO language abbreviation --
        index   CDATA   #IMPLIED -- entries for index compilation -->

<!ELEMENT (FOOTNOTE|MARGIN) - - (%text;)* -(FOOTNOTE|MARGIN)>
<!ATTLIST (FOOTNOTE|MARGIN)
        id      ID      #IMPLIED -- link destination --
        lang    CDATA   #IMPLIED -- ISO language abbreviation --
        index   CDATA   #IMPLIED -- entries for index compilation -->

<!ELEMENT RENDER -O (EMPTY) -- how to render unknown elements -->
<!ATTLIST RENDER
        tag     CDATA   #IMPLIED -- tag name --
        style   CDATA   #IMPLIED -- comma separated list of styles -->

<!-- Paragraphs which act as containers for the following text -->
<!ELEMENT P - O (L|%text;)+>
<!ATTLIST P
        id      ID      #IMPLIED -- link destination --
        align   (left|indent|center|right|justify) left
        lang    CDATA   #IMPLIED -- ISO language abbreviation --
        index   CDATA   #IMPLIED -- entries for index compilation -->
<!ELEMENT L - O (%text;)+>
<!ATTLIST L
        id      ID      #IMPLIED -- link destination --
        align   (left|indent|center|right|justify) left
        lang    CDATA   #IMPLIED -- ISO language abbreviation --
        index   CDATA   #IMPLIED -- entries for index compilation -->

<!ELEMENT HR - O EMPTY -- Horizontal Rule -->
<!ELEMENT BR - O EMPTY -- line break in normal text-->

<!ELEMENT PRE - - (TAB|%text;)+ -- preformatted fixed pitch text -->
<!ATTLIST PRE
        id      ID      #IMPLIED -- link destination --
        lang    CDATA   #IMPLIED -- ISO language abbreviation --
        index   CDATA   #IMPLIED -- entries for index compilation -->

<!ELEMENT LIT - - (TAB|%text;)+ -- literal variable pitch text -->
<!ATTLIST LIT
        id      ID      #IMPLIED -- link destination --
        lang    CDATA   #IMPLIED -- ISO language abbreviation --
        index   CDATA   #IMPLIED -- entries for index compilation -->

<!ELEMENT TAB - O EMPTY -- tabs for imported text -->
<!ATTLIST TAB
        at      NUMBER  #IMPLIED -- measured in widths of a capital M --
        align   (left|center|right|decimal) left -- tab alignment -->

<!ELEMENT QUOTE - - (P|%text;)* -- block quote -->
<!ATTLIST QUOTE
        id      ID      #IMPLIED -- link destination --
        lang    CDATA   #IMPLIED -- ISO language abbreviation --
        index   CDATA   #IMPLIED -- entries for index compilation -->
```

```
<!ELEMENT ABSTRACT - - (P|%text;)* -- document summary -->
<!ATTLIST ABSTRACT
        id      ID      #IMPLIED -- link destination --
        lang    CDATA   #IMPLIED -- ISO language abbreviation --
        index   CDATA   #IMPLIED -- entries for index compilation -->
<!ELEMENT BYLINE - - (P|%text;)* -- info on author -->
<!ATTLIST BYLINE
        id      ID      #IMPLIED -- link destination --
        lang    CDATA   #IMPLIED -- ISO language abbreviation --
        index   CDATA   #IMPLIED -- entries for index compilation -->
<!ELEMENT NOTE - - (P|%text;)* -- admonishment -->
<!ATTLIST NOTE
        id      ID      #IMPLIED -- link destination --
        role    CDATA   #IMPLIED -- e.g. Tip, Note, Warning, or Error --
        lang    CDATA   #IMPLIED -- ISO language abbreviation --
        index   CDATA   #IMPLIED -- entries for index compilation -->

<!ELEMENT (ADDRESS|BLOCKQUOTE) - - (%text;|P)* --compatibility with HTML-->

<!-- Lists which can be nested -->
<!ELEMENT OL - - (LI | UL | OL)+ -- ordered list -->
<!ATTLIST OL
        id      ID      #IMPLIED
        compact (compact)       #IMPLIED
        lang    CDATA   #IMPLIED -- ISO language abbreviation --
        index   CDATA   #IMPLIED -- entries for index compilation -->
<!ELEMENT UL - - (LI | UL | OL)+ -- unordered list -->
<!ATTLIST UL
        id      ID      #IMPLIED    -- link destination --
        compact (compact) #IMPLIED  -- reduced interitem spacing --
        plain   (plain) #IMPLIED -- suppress bullets --
        wrap    (vert|horiz) vert -- multicolumn list wrap style --
        lang    CDATA   #IMPLIED -- ISO language abbreviation --
        index   CDATA   #IMPLIED -- entries for index compilation -->

<!-- List items for UL and OL lists -->
<!ELEMENT LI - O (DL|P|%text;)+>
<!ATTLIST LI
        id      ID      #IMPLIED
        src     %URL;   #IMPLIED -- icon for use in place of bullet --
        lang    CDATA   #IMPLIED -- ISO language abbreviation --
        index   CDATA   #IMPLIED -- entries for index compilation -->

<!ELEMENT MENU - - (LI)* -- plain single column list -->
<!ATTLIST MENU
        id      ID      #IMPLIED
        lang    CDATA   #IMPLIED -- ISO language abbreviation --
        index   CDATA   #IMPLIED -- entries for index compilation -->
<!ELEMENT DIR - - (LI)* -- plain multi column list -->
<!ATTLIST DIR
        id      ID      #IMPLIED
        lang    CDATA   #IMPLIED -- ISO language abbreviation --
        index   CDATA   #IMPLIED -- entries for index compilation -->

<!-- Definition Lists (terms + definitions) -->
```

```
<!ELEMENT DL - - (DT+,DD)+>
<!ATTLIST DL
        id      ID      #IMPLIED
        compact (compact)       #IMPLIED
        lang    CDATA   #IMPLIED -- ISO language abbreviation --
        index   CDATA   #IMPLIED -- entries for index compilation -->

<!ELEMENT DT - O (%text;)+ -- term text -- >
<!ELEMENT DD - O (P|UL|OL|DL|%text;)+ -- definition text -- >
<!ATTLIST (DT|DD)
        id      ID      #IMPLIED
        lang    CDATA   #IMPLIED -- ISO language abbreviation --
        index   CDATA   #IMPLIED -- entries for index compilation -->

<!ELEMENT CAPTION - - (%text;)+ -- table or figure caption -->
<!ATTLIST CAPTION
        id      ID      #IMPLIED
        lang    CDATA   #IMPLIED -- ISO language abbreviation --
        index   CDATA   #IMPLIED -- entries for index compilation -->

<!ELEMENT TABLE - - (CAPTION?, (TH|TD|TR|TB)*) -- mixed headers and data -->
<!ATTLIST TABLE
        id      ID      #IMPLIED
        border (border) #IMPLIED -- draw borders --
        lang    CDATA   #IMPLIED -- ISO language abbreviation --
        index   CDATA   #IMPLIED -- entries for index compilation -->

<!ELEMENT TH - O (%table;)* -- a header cell -->
<!ATTLIST TH
        colspan NUMBER    1     -- columns spanned --
        rowspan NUMBER    1     --. rows spanned --
        align   (left|center|right) center -- alignment in cell --
        lang    CDATA   #IMPLIED -- ISO language abbreviation -->
<!ELEMENT TD - O (%table;)* -- a data cell -->
<!ATTLIST TD
        colspan NUMBER    1     -- columns spanned --
        rowspan NUMBER    1     --. rows spanned --
        align   (left|center|right) center -- alignment in cell --
        lang    CDATA   #IMPLIED -- ISO language abbreviation -->
<!ELEMENT TR - O (EMPTY) -- row separator -->
<!ATTLIST TR id ID #IMPLIED>

<!ELEMENT FORM - - (MH, (%main;)*) -(FORM) -- forms can't be nested -->
<!ATTLIST FORM
        id      ID      #IMPLIED
        action  %URL;   #IMPLIED -- defaults for URL for current doc --
        method  CDATA   #IMPLIED -- PUT, POST, DELETE etc. --
        lang    CDATA   #IMPLIED -- ISO language abbreviation --
        index   CDATA   #IMPLIED -- entries for index compilation -->

<!ELEMENT MH - - CDATA -- one or more RFC 822 header fields -->
<!ATTLIST MH hidden (hidden) #IMPLIED -- hide the mail headers from view -->

<!ELEMENT INPUT - O EMPTY>
<!ATTLIST INPUT
```

```
        name    CDATA   #IMPLIED -- attribute name (may not be unique) --
        type    CDATA   #IMPLIED -- a wide variety of field types --
        size    CDATA   #IMPLIED -- visible size of text fields --
        min     NUMBER  #IMPLIED -- for range controls --
        max     NUMBER  #IMPLIED -- for range controls or text fields --
        value   CDATA   #IMPLIED -- attribute value (altered by user) --
        checked (checked) #IMPLIED -- for check boxes and radio buttons --
        disabled (disabled) #IMPLIED -- if grayed out --
        error   (error) #IMPLIED -- if in error --
        src      %URL;  #IMPLIED -- for certain fields e.g. IMAGEs --
        align (top|middle|bottom) top -- for IMAGE fields only --
        lang    CDATA   #IMPLIED -- ISO language abbreviation -->

<!ELEMENT TEXTAREA - - CDATA -- multi-line text fields -->
<!ATTLIST TEXTAREA
        name    CDATA   #IMPLIED -- attribute name (may not be unique) --
        cols    NUMBER  #IMPLIED -- visible width in characters --
        rows    NUMBER  #IMPLIED -- visible height in characters --
        disabled (disabled) #IMPLIED -- if grayed out --
        error   (error) #IMPLIED -- if in error --
        lang    CDATA   #IMPLIED -- ISO language abbreviation -->

<!ELEMENT SELECT - - (OPTION+) -- combo style selection lists -->
<!ATTLIST SELECT
        name    CDATA   #IMPLIED -- attribute name (may not be unique) --
        several (several) #IMPLIED -- permits multiple selections --
        error   (error) #IMPLIED -- if in error --
        lang    CDATA   #IMPLIED -- ISO language abbreviation -->

<!ELEMENT OPTION - - CDATA>
<!ATTLIST OPTION
        selected (selected) #IMPLIED -- if initially selected --
        disabled (disabled) #IMPLIED -- if grayed out --
        lang    CDATA   #IMPLIED -- ISO language abbreviation -->

<!ELEMENT FIG - - (CAPTION?,(%text;)*)>
<!ATTLIST FIG
        id      ID      #IMPLIED
        align   (left|center|right|float) float -- position --
        ismap   (ismap) #IMPLIED -- server can handle mouse clicks/drags --
        src     %URL;   #IMPLIED -- link to image data --
        index   CDATA   #IMPLIED -- entries for index compilation --
        lang    CDATA   #IMPLIED -- ISO language abbreviation -->
<!ELEMENT IMG - O EMPTY>
<!ATTLIST IMG
        src     %URL;   #REQUIRED -- where to get image data --
        align   (top|middle|bottom) top  -- top, middle or bottom --
        seethru CDATA   #IMPLIED  -- for chromakey --
        alt     CDATA   #IMPLIED -- description for text-only displays --
        ismap   (ismap) #IMPLIED  -- send mouse clicks/drags to server -->
<!ELEMENT IMAGE - - (A|%text;)*>
<!ATTLIST IMAGE
        src     %URL;   #REQUIRED -- where to get image data --
        align   (top|middle|bottom) top  -- top, middle or bottom --
        seethru CDATA   #IMPLIED  -- for transparency --
```

```
        ismap   (ismap) #IMPLIED  -- send mouse clicks/drags to server --
        lang    CDATA   #IMPLIED -- ISO language abbreviation -->

<!-- proposal for representing formulae -->
<!ELEMENT MATH - - (%math;)*>
<!ATTLIST MATH id ID #IMPLIED>
<!ELEMENT BOX - - ((%math;)*, OVER?, (%math;)*)>
<!ELEMENT OVER - O EMPTY>

<!ELEMENT CHANGED - O EMPTY -- for change bars -->
<!ATTLIST CHANGED -- one of id and idref is always required --
        id      ID      #IMPLIED -- signals start of changes --
        idref   IDREF   #IMPLIED -- signals end of changes -->

<!-- Hypertext Links from points within document nodes -->
<!ELEMENT A - - (#PCDATA | IMG | EM | EMBED)*>
<!ATTLIST A
        id      ID      #IMPLIED -- as target of link --
        name    CDATA   #IMPLIED -- for backwards compatibility with HTML--
        shape   CDATA   #IMPLIED -- list of points for shaped buttons --
        href    %URL;   #IMPLIED -- destination node --
        rel     CDATA   #IMPLIED -- forward relationship type --
        rev     CDATA   #IMPLIED -- reverse relationship type --
        methods CDATA   #IMPLIED -- supported public methods --
        effect  CDATA   #IMPLIED -- replace/new/overlay/embed --
        print   CDATA   #IMPLIED -- reference/footnote/section --
        title   CDATA   #IMPLIED -- when otherwise unavailable --
        type    CDATA   #IMPLIED -- for presentation cues --
        size    NAMES   #IMPLIED -- for progress cues --
        lang    CDATA   #IMPLIED -- ISO language abbreviation -->
<!-- Other kinds of relationships between documents -->
<!ELEMENT LINK - O EMPTY>

<!ATTLIST LINK
        idref  IDREF    #IMPLIED -- starting point --
        href    %URL;   #IMPLIED -- destination node --
        rel    CDATA    #IMPLIED -- forward relationship type --
        rev    CDATA    #IMPLIED -- reverse relationship type --
        methods CDATA   #IMPLIED -- supported public methods -->

<!-- Original document URL for resolving relative URLs  -->
<!ELEMENT BASE - O EMPTY>
<!ATTLIST BASE HREF %URL; #IMPLIED>
<!-- Signifies the document’s URL accepts queries -->
<!ELEMENT ISINDEX - O (EMPTY)>
<!ATTLIST ISINDEX href %URL; #IMPLIED -- defaults to document’s URL -->
<!-- For use with autonumbering editors - don’t reuse ids, allocate next one
starting from this one -->
<!ELEMENT NEXTID - O (EMPTY)>
<!ATTLIST NEXTID N NAME #REQUIRED>

<!-- Mnemonic character entities for 8 bit ANSI Latin-1
     ignore when using other character sets -->
<!ENTITY iexcl “&#161;” -- inverted exclamation mark -->
<!ENTITY cent “&#161;” -- cent sign -->
```

```
<!ENTITY pound “&#163;” -- pound sign -->
<!ENTITY yen “&#165;” -- yen sign -->
<!ENTITY brvbar “&#166;” -- broken vertical bar -->
<!ENTITY sect “&#167;” -- section sign -->
<!ENTITY copy “&#169;” -- copyright sign -->
<!ENTITY laquo “&#171;” -- angle quotation mark, left -->
<!ENTITY raquo “&#187;” -- angle quotation mark, right -->
<!ENTITY not “&#172;” -- negation sign -->
<!ENTITY reg “&#174;” -- circled R registered sign -->
<!ENTITY deg “&#176;” -- degree sign -->
<!ENTITY plusmn “&#177;” -- plus or minus sign -->
<!ENTITY sup2 “&#178;” -- superscript 2 -->
<!ENTITY sup3 “&#179;” -- superscript 3 -->
<!ENTITY micro “&#181;” -- micro sign -->
<!ENTITY para “&#182;” -- paragraph sign -->
<!ENTITY sup1 “&#185;” -- superscript 1 -->
<!ENTITY middot “&#183;” -- center dot -->
<!ENTITY frac14 “&#188;” -- fraction 1/4 -->
<!ENTITY frac12 “&#189;” -- fraction 1/2 -->
<!ENTITY iquest “&#191;” -- inverted question mark -->
<!ENTITY frac34 “&#190;” -- fraction 3/4 -->
<!ENTITY AElig “&#198;” -- capital AE diphthong (ligature) -->
<!ENTITY Aacute “&#193;” -- capital A, acute accent -->
<!ENTITY Acirc “&#194;” -- capital A, circumflex accent -->
<!ENTITY Agrave “&#192;” -- capital A, grave accent -->
<!ENTITY Aring “&#197;” -- capital A, ring -->
<!ENTITY Atilde “&#195;” -- capital A, tilde -->
<!ENTITY Auml “&#196;” -- capital A, dieresis or umlaut mark -->
<!ENTITY Ccedil “&#199;” -- capital C, cedilla -->
<!ENTITY ETH “&#208;” -- capital Eth, Icelandic -->
<!ENTITY Eacute “&#201;” -- capital E, acute accent -->
<!ENTITY Ecirc “&#202;” -- capital E, circumflex accent -->
<!ENTITY Egrave “&#200;” -- capital E, grave accent -->
<!ENTITY Euml “&#203;” -- capital E, dieresis or umlaut mark -->
<!ENTITY Iacute “&#205;” -- capital I, acute accent -->
<!ENTITY Icirc “&#206;” -- capital I, circumflex accent -->
<!ENTITY Igrave “&#204;” -- capital I, grave accent -->
<!ENTITY Iuml “&#207;” -- capital I, dieresis or umlaut mark -->
<!ENTITY Ntilde “&#209;” -- capital N, tilde -->
<!ENTITY Oacute “&#211;” -- capital O, acute accent -->
<!ENTITY Ocirc “&#212;” -- capital O, circumflex accent -->
<!ENTITY Ograve “&#210;” -- capital O, grave accent -->
<!ENTITY Oslash “&#216;” -- capital O, slash -->
<!ENTITY Otilde “&#213;” -- capital O, tilde -->
<!ENTITY Ouml “&#214;” -- capital O, dieresis or umlaut mark -->
<!ENTITY THORN “&#222;” -- capital THORN, Icelandic -->
<!ENTITY Uacute “&#218;” -- capital U, acute accent -->
<!ENTITY Ucirc “&#219;” -- capital U, circumflex accent -->
<!ENTITY Ugrave “&#217;” -- capital U, grave accent -->
<!ENTITY Uuml “&#220;” -- capital U, dieresis or umlaut mark -->
<!ENTITY Yacute “&#221;” -- capital Y, acute accent -->
<!ENTITY aacute “&#225;” -- small a, acute accent -->
<!ENTITY acirc “&#226;” -- small a, circumflex accent -->
<!ENTITY aelig “&#230;” -- small ae diphthong (ligature) -->
<!ENTITY agrave “&#224;” -- small a, grave accent -->
```

```
<!ENTITY amp "&amp;" -- ampersand -->
<!ENTITY aring "&#229;" -- small a, ring -->
<!ENTITY atilde "&#227;" -- small a, tilde -->
<!ENTITY auml "&#228;" -- small a, dieresis or umlaut mark -->
<!ENTITY ccedil "&#231;" -- small c, cedilla -->
<!ENTITY eacute "&#233;" -- small e, acute accent -->
<!ENTITY ecirc "&#234;" -- small e, circumflex accent -->
<!ENTITY egrave "&#232;" -- small e, grave accent -->
<!ENTITY eth "&#240;" -- small eth, Icelandic -->
<!ENTITY euml "&#235;" -- small e, dieresis or umlaut mark -->
<!ENTITY gt "&#62;" -- greater than -->
<!ENTITY iacute "&#237;" -- small i, acute accent -->
<!ENTITY icirc "&#238;" -- small i, circumflex accent -->
<!ENTITY igrave "&#236;" -- small i, grave accent -->
<!ENTITY iuml "&#239;" -- small i, dieresis or umlaut mark -->
<!ENTITY lt "&lt;" -- less than -->
<!ENTITY ntilde "&#241;" -- small n, tilde -->
<!ENTITY oacute "&#243;" -- small o, acute accent -->
<!ENTITY ocirc "&#244;" -- small o, circumflex accent -->
<!ENTITY ograve "&#242;" -- small o, grave accent -->
<!ENTITY oslash "&#248;" -- small o, slash -->
<!ENTITY otilde "&#245;" -- small o, tilde -->
<!ENTITY ouml "&#246;" -- small o, dieresis or umlaut mark -->
<!ENTITY szlig "&#223;" -- small sharp s, German (sz ligature) -->
<!ENTITY thorn "&#254;" -- small thorn, Icelandic -->
<!ENTITY uacute "&#250;" -- small u, acute accent -->
<!ENTITY ucirc "&#251;" -- small u, circumflex accent -->
<!ENTITY ugrave "&#249;" -- small u, grave accent -->
<!ENTITY uuml "&#252;" -- small u, dieresis or umlaut mark -->
<!ENTITY yacute "&#253;" -- small y, acute accent -->
<!ENTITY yuml "&#255;" -- small y, dieresis or umlaut mark -->

<!-- The END -->
]>
```

Appendix II

The proposed character entities for HTML+ and their corresponding character codes for Unicode and the Adobe Latin-1 & Symbol character sets.

Symbol	ISO 8879	Adobe name	Adobe Hex	Unicode	Unicode name
ISO Latin 1 Entities					
"	quot	quotedbl	22	0022	QUOTATION MARK
&	amp	ampersand	26	0026	AMPERSAND
<	lt	less	3C	003C	LESS-THAN SIGN
>	gt	greater	3E	003E	GREATER-THAN SIGN
	nbsp		A0	00A0	NON-BREAKING SPACE
¡	iexcl	exclamdown	A1	00A1	INVERTED EXCLAMATION MARK
¢	cent	cent	A2	00A2	CENT SIGN
£	pound	sterling	A3	00A3	POUND SIGN
¤	curren	currency	A4	00A4	CURRENCY SIGN
¥	yen	yen	A5	00A5	YEN SIGN
	brvbar	brokenbar	A6	00A6	BROKEN VERTICAL BAR
§	sect	section	A7	00A7	SECTION SIGN
¨	die	dieresis	A8	00A8	SPACING DIAERESIS
©	copy	copyright	A9	00A9	COPYRIGHT SIGN
ª	ordf	ordfeminine	AA	00AA	FEMININE ORDINAL INDICATOR
«	laquo	guillemotleft	AB	00AB	LEFT POINTING GUILLEMET
¬	not	logicalnot	AC	00AC	NOT SIGN
-	shy	hyphen	AD	00AD	SOFT HYPHEN
®	reg	registered	AE	00AE	REGISTERED TRADE MARK SIGN
¯	macron	macron	AF	00AF	SPACING MACRON
°	degree	degree	B0	00B0	DEGREE SIGN

Symbol	ISO 8879	Adobe name	Adobe Hex	Unicode	Unicode name
±	plusmn	plusminus	B1	00B1	PLUS-OR-MINUS SIGN
²	sup2	twosuperior	B2	00B2	SUPERSCRIPT DIGIT TWO
³	sup3	threesuperior	B3	00B3	SUPERSCRIPT DIGIT THREE
´	acute	acute	B4	00B4	SPACING ACUTE
µ	micro	mu	B5	00B5	MICRO SIGN
¶	para	paragraph	B6	00B6	PARAGRAPH SIGN
·	middot	periodcentered	B7	00B7	MIDDLE DOT
¸	Cedilla	cedilla	B8	00B8	SPACING CEDILLA
¹	sup1	onesuperior	B9	00B9	SUPERSCRIPT DIGIT ONE
º	ordm	ordmasculine	BA	00BA	MASCULINE ORDINAL INDICATOR
»	raquo	guillemotright	BB	00BB	RIGHT POINTING GUILLEMET
1/4	frac14	onequarter	BC	00BC	FRACTION ONE QUARTER
1/2	frac12	onehalf	BD	00BD	FRACTION ONE HALF
3/4	frac34	threequarters	BE	00BE	FRACTION THREE QUARTERS
¿	iquest	questiondown	BF	00BF	INVERTED QUESTION MARK
À	Agrave	Agrave	C0	00C0	LATIN CAPITAL LETTER A GRAVE
Á	Aacute	Aacute	C1	00C1	LATIN CAPITAL LETTER A ACUTE
Â	Acirc	Acircumflex	C2	00C2	LATIN CAPITAL LETTER A CIRCUMFLEX
Ã	Atilde	Atilde	C3	00C3	LATIN CAPITAL LETTER A TILDE
Ä	Auml	Adieresis	C4	00C4	LATIN CAPITAL LETTER A DIAERESIS
Å	Aring	Aring	C5	00C5	LATIN CAPITAL LETTER A RING
Æ	AElig	AE	C6	00C6	LATIN CAPITAL LETTER A E
Ç	Ccedil	CCedilla	C7	00C7	LATIN CAPITAL LETTER C CEDILLA
È	Egrave	Egrave	C8	00C8	LATIN CAPITAL LETTER E GRAVE
É	Eacute	Eacute	C9	00C9	LATIN CAPITAL LETTER E ACUTE
Ê	Ecirc	Ecircumflex	CA	00CA	LATIN CAPITAL LETTER E CIRCUMFLEX
Ë	Euml	Edieresis	CB	00CB	LATIN CAPITAL LETTER E DIAERESIS

Symbol	ISO 8879	Adobe name	Adobe Hex	Unicode	Unicode name
Ì	Igrave	Igrave	CC	00CC	LATIN CAPITAL LETTER I GRAVE
Í	Iacute	Iacute	CD	00CD	LATIN CAPITAL LETTER I ACUTE
Î	Icirc	Icircumflex	CE	00CE	LATIN CAPITAL LETTER I CIRCUMFLEX
Ï	Iuml	Idieresis	CF	00CF	LATIN CAPITAL LETTER I DIAERESIS
	ETH	Eth	D0	00D0	LATIN CAPITAL LETTER ETH
Ñ	Ntilde	Ntilde	D1	00D1	LATIN CAPITAL LETTER N TILDE
Ò	Ograve	Ograve	D2	00D2	LATIN CAPITAL LETTER O GRAVE
Ó	Oacute	Oacute	D3	00D3	LATIN CAPITAL LETTER O ACUTE
Ô	Ocirc	Ocircumflex	D4	00D4	LATIN CAPITAL LETTER O CIRCUMFLEX
Õ	Otilde	Otilde	D5	00D5	LATIN CAPITAL LETTER O TILDE
Ö	Ouml	Odieresis	D6	00D6	LATIN CAPITAL LETTER O DIAERESIS
×	times	multiply	D7	00D7	MULTIPLICATION SIGN
Ø	Oslash	Oslash	D8	00D8	LATIN CAPITAL LEETER O SLASH
Ù	Ugrave	Ugrave	D9	00D9	LATIN CAPITAL LETTER U GRAVE
Ú	Uacute	Uacute	DA	00DA	LATIN CAPITAL LETTER U ACUTE
Û	Ucirc	Ucircumflex	DB	00DB	LATIN CAPITAL LETTER U CIRCUMFLEX
Ü	Uuml	Udieresis	DC	00DC	LATIN CAPITAL LETTER U DIAERESIS
	Yacute	Yacute	DD	00DD	LATIN CAPITAL LETTER Y ACUTE
	THORN	Thorn	DE	00DE	LATIN CAPITAL LETTER THORN
ß	szlig	germandbls	DF	00DF	LATIN SMALL LETTER SHARP S
à	agrave	agrave	E0	00E0	LATIN SMALL LETTER A GRAVE
á	aacute	aacute	E1	00E1	LATIN SMALL LETTER A ACUTE
â	acirc	acircumflex	E2	00E2	LATIN SMALL LETTER A CIRCUMFLEX
ã	atilde	atilde	E3	00E3	LATIN SMALL LETTER A TILDE
ä	auml	adieresis	E4	00E4	LATIN SMALL LETTER A DIAERESIS
å	aring	aring	E5	00E5	LATIN SMALL LETTER A RING
æ	aelig	ae	E6	00E6	LATIN SMALL LETTER A E

Symbol	ISO 8879	Adobe name	Adobe Hex	Unicode	Unicode name
ç	ccedil	ccedilla	E7	00E7	LATIN SMALL LETTER C CEDILLA
è	egrave	egrave	E8	00E8	LATIN SMALL LETTER E GRAVE
é	eacute	eacute	E9	00E9	LATIN SMALL LETTER E ACUTE
ê	ecirc	ecircumflex	EA	00EA	LATIN SMALL LETTER E CIRCUMFLEX
ë	euml	edieresis	EB	00EB	LATIN SMALL LETTER E DIAERESIS
ì	igrave	igrave	EC	00EC	LATIN SMALL LETTER I GRAVE
í	iacute	iacute	ED	00ED	LATIN SMALL LETTER I ACUTE
î	icirc	icircumflex	EE	00EE	LATIN SMALL LETTER I CIRCUMFLEX
ï	iuml	idieresis	EF	00EF	LATIN SMALL LETTER I DIAERESIS
	eth	eth	F0	00F0	LATIN SMALL LETTER ETH
ñ	ntilde	ntilde	F1	00F1	LATIN SMALL LETTER N TILDE
ò	ograve	ograve	F2	00F2	LATIN SMALL LETTER O GRAVE
ó	oacute	oacute	F3	00F3	LATIN SMALL LETTER O ACUTE
ô	ocirc	ocircumflex	F4	00F4	LATIN SMALL LETTER O CIRCUMFLEX
õ	otilde	otilde	F5	00F5	LATIN SMALL LETTER O TILDE
ö	ouml	odieresis	F6	00F6	LATIN SMALL LETTER O DIAERESIS
÷	divide	divide	F7	00F7	DIVISION SIGN
ø	oslash	oslash	F8	00F8	LATIN SMALL LETTER O SLASH
ù	ugrave	ugrave	F9	00F9	LATIN SMALL LETTER U GRAVE
ú	uacute	uacute	FA	00FA	LATIN SMALL LETTER U ACUTE
û	ucirc	ucircumflex	FB	00FB	LATIN SMALL LETTER U CIRCUMFLEX
ü	uuml	udieresis	FC	00FC	LATIN SMALL LETTER U DIAERESIS
	yacute	yacute	FD	00FD	LATIN SMALL LETTER Y ACUTE
	thorn	thorn	FE	00FE	LATIN SMALL LETTER THORN
ÿ	yuml	ydieresis	FF	00FF	LATIN SMALL LETTER Y DIAERESIS
Adobe Symbol Font Entities					
∀	forall	universal	22	2200	FOR ALL

Symbol	ISO 8879	Adobe name	Adobe Hex	Unicode	Unicode name
∃	exist	existential	24	2203	THERE EXISTS
∍	ni	suchthat	27	220B	CONTAINS AS MEMBER
∗	lowast	asteriskmath	2A	2217	ASTERISK OPERATOR
−	minus	minus	2D	2212	MINUS SIGN
≅	cong	congruent	40	2245	APPROXIMATELY EQUAL TO
Α	Agr	Alpha	41	0391	GREEK CAPITAL LETTER ALPHA
Β	Bgr	Beta	42	0392	GREEK CAPITAL LETTER BETA
Χ	KHgr	Chi	43	03A7	GREEK CAPITAL LETTER CHI
Δ	Delta	Delta	44	0394	GREEK CAPITAL LETTER DELTA
Ε	Egr	Epsilon	45	0395	GREEK CAPITAL LETTER EPSILON
Φ	PHgr	Phi	46	03A6	GREEK CAPITAL LETTER PHI
Γ	Gamma	Gamma	47	0393	GREEK CAPITAL LETTER GAMMA
Η	EEgr	Eta	48	0397	GREEK CAPITAL LETTER ETA
Ι	Igr	Iota	49	0399	GREEK CAPITAL LETTER IOTA
ϑ	thetav	theta1	4A	03D1	GREEK SMALL LETTER SCRIPT THETA
Κ	Kgr	Kappa	4B	039A	GREEK CAPITAL LETTER KAPPA
Λ	Lambda	Lambda	4C	039B	GREEK CAPITAL LETTER LAMBDA
Μ	Mgr	Mu	4D	039C	GREEK CAPITAL LETTER MU
Ν	Ngr	Nu	4E	039D	GREEK CAPITAL LETTER NU
Ο	Ogr	Omicron	4F	039F	GREEK CAPITAL LETTER OMICRON
Π	Pi	Pi	50	03A0	GREEK CAPITAL LETTER PI
Θ	Theta	Theta	51	0398	GREEK CAPITAL LETTER THETA
Ρ	Rgr	Rho	52	03A1	GREEK CAPITAL LETTER RHO
Σ	Sigma	Sigma	53	03A3	GREEK CAPITAL LETTER SIGMA
Τ	Tgr	Tau	54	03A4	GREEK CAPITAL LETTER TAU
Υ	Upsi	Upsilon	55	03A5	GREEK CAPITAL LETTER UPSILON
ς	sfgr	sigma1	56	03C2	GREEK SMALL LETTER FINAL SIGMA

Symbol	ISO 8879	Adobe name	Adobe Hex	Unicode	Unicode name
Ω	Omega	Omega	57	03A9	GREEK CAPITAL LETTER OMEGA
Ξ	Xi	Xi	58	039E	GREEK CAPITAL LETTER XI
Ψ	Psi	Psi	59	03A8	GREEK CAPITAL LETTER PSI
Ζ	Zgr	Zeta	5A	0396	GREEK CAPITAL LETTER ZETA
∴	there4	therefore	5C	2234	THEREFORE
⊥	perp	perpendicular	5E	22A5	UP TACK
‾		radicalex	60	203E	SPACING OVERSCORE
α	alpha	alpha	61	03B1	GREEK SMALL LETTER ALPHA
β	beta	beta	62	03B2	GREEK SMALL LETTER BETA
χ	chi	chi	63	03C7	GREEK SMALL LETTER CHI
δ	delta	delta	64	03B4	GREEK SMALL LETTER DELTA
ε	epsi	epsilon	65	03B5	GREEK SMALL LETTER EPSILON
ϕ	phis	phi	66	03C6	GREEK SMALL LETTER PHI
γ	gamma	gamma	67	03B3	GREEK SMALL LETTER GAMMA
η	eta	eta	68	03B7	GREEK SMALL LETTER ETA
ι	iota	iota	69	03B9	GREEK SMALL LETTER IOTA
φ	phiv	phi1	6A	03D5	GREEK SMALL LETTER SCRIPT PHI
κ	kappa	kappa	6B	03BA	GREEK SMALL LETTER KAPPA
λ	lambda	lambda	6C	03BB	GREEK SMALL LETTER LAMBDA
μ	mu	mu	6D	03BC	GREEK SMALL LETTER MU
ν	nu	nu	6E	03BD	GREEK SMALL LETTER NU
ο	ogr	omicron	6F	03BF	GREEK SMALL LETTER OMICRON
π	pi	pi	70	03C0	GREEK SMALL LETTER PI
θ	thetas	theta	71	03B8	GREEK SMALL LETTER THETA
ρ	rho	rho	72	03C1	GREEK SMALL LETTER RHO
σ	sigma	sigma	73	03C3	GREEK SMALL LETTER SIGMA
τ	tau	tau	74	03C4	GREEK SMALL LETTER TAU

Symbol	ISO 8879	Adobe name	Adobe Hex	Unicode	Unicode name
υ	upsi	upsilon	75	03C5	GREEK SMALL LETTER UPSILON
ϖ	piv	omega1	76	03D6	GREEK SMALL LETTER OMEGA PI
ω	omega	omega	77	03C9	GREEK SMALL LETTER OMEGA
ξ	xi	xi	78	03BE	GREEK SMALL LETTER XI
ψ	psi	psi	79	03C8	GREEK SMALL LETTER PSI
ζ	zeta	zeta	7A	03B6	GREEK SMALL LETTER ZETA
~	sim	similar	7E	223C	TILDE OPERATOR
ϒ		Upsilon1	A1	03D2	GREEK CAPITAL LETTER UPSILON HOOK
′	vprime	minute	A2	2032	PRIME
≤	le	lessequal	A3	2264	LESS THAN OR EQUAL TO
⁄		fraction	A4	2044	FRACTION SLASH
∞	infin	infinity	A5	221E	INFINITY
ƒ	fnof	florin	A6	0192	LATIN SMALL LETTER SCRIPT F
♣	clubs	club	A7	2663	BLACK CLUB SUIT
♦	diams	diamond	A8	2666	BLACK DIAMOND SUIT
♥	hearts	heart	A9	2665	BLACK HEART SUIT
♠	spades	spade	AA	2660	BLACK SPADE SUIT
↔	harr	arrowboth	AB	2194	LEFT RIGHT ARROW
←	larr	arrowleft	AC	2190	LEFT ARROW
↑	uarr	arrowup	AD	2191	UP ARROW
→	rarr	arrowright	AE	2192	RIGHT ARROW
↓	darr	arrowdown	AF	2193	DOWN ARROW
″	Prime	second	B2	2033	DOUBLE PRIME
≥	ge	greaterequal	B3	2265	GREATER THAN OR EQUAL TO
∝	prop	proportional	B5	221D	PROPORTIONAL TO
∂	part	partialdiff	B6	2202	PARTIAL DIFFERENTIAL
•	bull	bullet	B7	2022	BULLET

Symbol	ISO 8879	Adobe name	Adobe Hex	Unicode	Unicode name
≠	ne	notequal	B9	2260	NOT EQUAL TO
≡	equiv	equivalence	BA	2261	IDENTICAL TO
≈	ap	approxequal	BB	2248	ALMOST EQUAL TO
…	hellip	ellipsis	BC	2026	HORIZONTAL ELLIPSIS
|		arrowvertex	BD		
—		arrowhorizex	BE		
↵		carriagereturn	BF	21B5	DOWN ARROW WITH CORNER LEFT
ℵ	aleph	aleph	C0	2135	FIRST TRANSFINITE CARDINAL
ℑ	image	Ifraktur	C1	2111	BLACK-LETTER I
ℜ	real	Rfraktur	C2	211C	BLACK-LETTER R
℘	weierp	weierstrass	C3	2118	SCRIPT P
⊗	otimes	circlemultiply	C4	2297	CIRCLED TIMES
⊕	oplus	circleplus	C5	2295	CIRCLED PLUS
∅	empty	emptyset	C6	2205	EMPTY SET
∩	cap	intersection	C7	2229	INTERSECTION
∪	cup	union	C8	222A	UNION
⊃	sup	propersuperset	C9	2283	SUPERSET OF
⊇	supe	reflexsuperset	CA	2287	SUPERSET OF OR EQUAL TO
⊄	nsub	notsubset	CB	2284	NOT A SUBSET OF
⊂	sub	propersubset	CC	2282	SUBSET OF
⊆	sube	reflexsubset	CD	2286	SUBSET OF OR EQUAL TO
∈	isin	element	CE	2208	ELEMENT OF
∉	notin	notelement	CF	2209	NOT AN ELEMENT OF
∠	ang	angle	D0	2220	ANGLE
∇	nabla	gradient	D1	2207	NABLA
™	trade	trademarkserif	D4	2122	TRADEMARK
∏	prod	product	D5	220F	N-ARY PRODUCT

Symbol	ISO 8879	Adobe name	Adobe Hex	Unicode	Unicode name
√	radic	radical	D6	221A	SQUARE ROOT
·	sdot	dotmath	D7	22C5	DOT OPERATOR
∧	and	logicaland	D9	2227	LOGICAL AND
∨	or	logicalor	DA	2228	LOGICAL OR
⇔	hArr	arrowdblboth	DB	21D4	LEFT RIGHT DOUBLE ARROW
⇐	lArr	arrowdblleft	DC	21D0	LEFT DOUBLE ARROW
⇑	uArr	arrowdblup	DD	21D1	UP DOUBLE ARROW
⇒	rArr	arrowdblright	DE	21D2	RIGHT DOUBLE ARROW
⇓	dArr	arrowdbldown	DF	21D3	DOWN DOUBLE ARROW
◊	loz	lozenge	E0	25CA	LOZENGE
〈	lang	angleleft	E1	2329	BRA
∑	sum	summation	E5	2211	N-ARY SUMMATION
⎛		parenlefttp	E6		
⎜		parenleftex	E7		
⎝		parenleftbt	E8		
⌈	lceil	bracketlefttp	E9	2308	LEFT CEILING
⎢		bracketleftex	EA		
⌊	lfloor	bracketleftbt	EB	230A	LEFT FLOOR
⎧		bracelefttp	EC		
⎨		braceleftmid	ED		
⎩		braceleftbt	EE		
⎪		braceex	EF		
〉	rang	angleright	F1	232A	KET
∫	int	integral	F2	222B	INTEGRAL
⌠		integraltp	F3	2320	TOP HALF INTEGRAL
⎮		integralex	F4		
⌡		integralbt	F5	2321	BOTTOM HALF INTEGRAL
⎞		parenrighttp	F6		

Symbol	ISO 8879	Adobe name	Adobe Hex	Unicode	Unicode name
⎟		parenrightex	F7		
⎠		parenrightbt	F8		
⎤	rceil	bracketrighttp	F9	2309	RIGHT CEILING
⎥		bracketrightex	FA		
⎦	rfloor	bracketrightbt	FB	230B	RIGHT FLOOR
⎫		bracerighttp	FC		
⎬		bracerightmid	FD		
⎭		bracerightbt	FE		

Many thanks to Bob Stayton for volunteering his time and effort to prepare this table.

Appendix III

ANSI C Code for testing if a point lies within a polygon. This may be freely used provided the original copyright message is retained in full.

```
/*
** Reproduced with thanks from mapper 1.2
** 7/26/93 Kevin Hughes, kevinh@pulua.hcc.hawaii.edu
** polygon code copyright 1992 by Eric Haines, erich@eye.com
*/

#include <stdio.h>

#define MAXLINE 500
#define MAXVERTS 100
#define X 0
#define Y 1

int pointinpoly(double point[2], double pgon[MAXVERTS][2])
{
    int i, numverts, inside_flag, xflag0;
    int crossings;
    double *p, *stop;
    double tx, ty, y;

    for (i = 0; pgon[i][X] != -1 && i < MAXVERTS; i++)
            ;
    numverts = i;
    crossings = 0;

    tx = point[X];
    ty = point[Y];
    y = pgon[numverts - 1][Y];

    p = (double *) pgon + 1;
    if ((y >= ty) != (*p >= ty))
    {
        if ((xflag0 = (pgon[numverts - 1][X] >= tx)) ==
            (*(double *) pgon >= tx))
        {
            if (xflag0)
                crossings++;
        }
         else
        {
            crossings += (pgon[numverts - 1][X] - (y - ty) *
                        (*(double *) pgon - pgon[numverts - 1][X]) /
                        (*p - y)) >= tx;
        }
    }

    stop = pgon[numverts];
```

```
    for (y = *p, p += 2; p < stop; y = *p, p += 2)
    {
        if (y >= ty)
        {
            while ((p < stop) && (*p >= ty))
                p += 2;
            if (p >= stop)
                break;
            if ((xflag0 = (*(p - 3) >= tx)) == (*(p - 1) >= tx))
            {
                if (xflag0)
                    crossings++;
            }
            else
            {
                crossings += (*(p - 3) - (*(p - 2) - ty) *
                             (*(p - 1) - *(p - 3)) / (*p - *(p - 2))) >= tx;
            }
        }
        else
        {
            while ((p < stop) && (*p < ty))
                p += 2;
            if (p >= stop)
                break;
            if ((xflag0 = (*(p - 3) >= tx)) == (*(p - 1) >= tx))
            {
                if (xflag0)
                    crossings++;
            }
            else
            {
                crossings += (*(p - 3) - (*(p - 2) - ty) *
                             (*(p - 1) - *(p - 3)) / (*p - *(p - 2))) >= tx;
            }
        }
    }
    inside_flag = crossings & 0x01;
    return (inside_flag);
}
```

Appendix IV

Alphabetically sorted list of HTML+ tags and their associated attribute names.

A	id, name, shape, href, rel, rev, methods, effect, print, title, type, size, lang
ABBREV	id, lang, index
ABSTRACT	id, lang, index
ACRONYM	id, lang, index
ADDED	id, lang, index
ADDRESS	
ARG	id, lang, index
B	id, lang, index
BASE	href
BLOCKQUOTE	
BODY	
BOX	
BR	
BYLINE	id, lang, index
CAPTION	id, lang, index
CHANGED	id, idref
CITE	id, lang, index
CMD	id, lang, index
CODE	id, lang, index
DD	id, lang, index
DFN	id, lang, index
DIR	id, lang, index
DL	id, compact, lang, index
DT	id, lang, index
EM	id, lang, index
FIG	id, align, ismap, src, index, lang
FOOTNOTE	id, lang, index
FORM	id, action, method, lang, index
H1	id, lang, index
H2	id, lang, index
H3	id, lang, index
H4	id, lang, index
H5	id, lang, index
H6	id, lang, index
HEAD	
HTMLPLUS	version, forms
I	id, lang, index
IMAGE	src, align, seethru, ismap, lang
IMG	src, align, seethru, alt, ismap

INPUT	name, type, size, min, max, value, checked, disabled, error, src, align, lang
ISINDEX	href
KBD	id, lang, index
L	id, align, lang, index
LI	id, src, lang, index
LINK	idref, href, rel, rev, methods
LIT	id, lang, index
MARGIN	id, lang, index
MATH	id
MENU	id, lang, index
NEXTID	n
NOTE	id, role, lang, index
OL	id, compact, lang, index
OPTION	selected, disabled, lang
OVER	
P	id, align, lang, index
PERSON	id, lang, index
PRE	id, lang, index
Q	id, lang, index
QUOTE	id, lang, index
RENDER	tag, style
REMOVED	id, lang, index
S	id, lang, index
SAMP	id, lang, index
SELECT	name, several, error, lang
STRONG	id, lang, index
SUB	id, lang, index
SUP	id, lang, index
TAB	at, align
TABLE	id, border, lang, index
TD	colspan, rowspan, align, lang
TEXTAREA	name, cols, rows, disabled, error, lang
TH	colspan, rowspan, align, lang
TITLE	id, lang, index
TR	id
TT	id, lang, index
U	id, lang, index
UL	id, compact, plain, wrap, lang, index
VAR	id, lang, index