Request for Comments: XXXX
David Raggett (Hewlett Packard)
12th July 1993

HTML+ (Hypertext markup language)

A proposed standard for a light weight presentation
independent delivery format for browsing and
querying information across the Internet

Status of this Memo

This document is a proposal for an Internet Draft, and specifies the HTML+ wide-area hypertext document format, with a view to requesting discussion[1] and suggestions for improvements. Distribution of this memo is unlimited.

Abstract

HTML+ is a simple SGML based format for wide-area hypertext documents, for use within the World Wide Web. Unlike desktop publishing formats, HTML+ captures the logical intent of authors. This simplifies the task of writing documents, and permits them to be effectively rendered on a wide range of display types as well as the printed page.

HTML+ represents a substantial improvement over the existing format: HTML, offering nested lists, figures, embedded data in foreign formats for equations etc, tables with support for titles and column headings, change bars, entry forms for querying and updating information sources and for use as questionaires for mailing. This document specifies the HTML+ format with guidelines on how it should be rendered by browsers.

Introduction

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

a) Universal naming scheme for documents. The universal resource location 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 working groups of the IETF. A network protocol will allow universal resource numbers (URNs) to be resolved to the URL for the nearest available copy.

b) Use of available protocols for retrieving documents over the network, including FTP, NNTP, WAIS, Gopher, and HTTP. The latter is designed specifically for use with the World Wide Web, and combines efficiency with an ability to flexibly exchange information between clients and servers.

c) A document format supporting hypertext links based on URLs and URNs which can specify documents anywhere in the Internet. HTML+ is designed for rendering on a wide variety of different display types and platforms.

Information browsers can display information in a wide variety of formats, e.g. plain text, rich text in the HTML+ format, images in the GIF and JPEG formats, MPEG movies, and MIME documents. The hypertext format has a special significance as it allows users to navigate from one document to the next at the click of a button. It provides the basis for menus, cross references, either within a document or to other documents,

---

[1]Please mail comments to the author `dsr@hplb.hpl.hp.com`, or to the WWW discussion group: `www-talk@nxoc01.cern.ch`

perhaps on the other side of the world. It also provides a means of building larger scale collections of documents that act as journals, books or encyclopedias. The format is also intended to act as a building block for creating wide area groupware applications.

HTML+ follows on from an earlier standard - HTML, see [Berners-Lee 93a], which has been widely used as the basis for hypertext documents in the World Wide Web. The new format grew out of experience with HTML, culminating in the desire to add new features, e.g. inline images, tables, and form fields for greater flexibility in querying remote information sources. This document specifies the HTML+ format and suggests ways in which browsers can choose to render it on a variety of different display types.

2. HTML+ and SGML

HTML+ itself is based on the Standardised General Markup Language (SGML), 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 the document should 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 Data Type Definition, usually abbreviated to DTD.

Working groups in ISO have recently produced a range of SGML DTDs for documents, e.g. ISO 12083 defines DTDs for books and ISO 10744, which defines the HyTime standard for hypermedia/time-based documents. These standards are large and complex, and perhaps best suited as interchange standards that facilitate conversion between proprietary document formats. By contrast, HTML+ provides a lightweight delivery format that can be rendered by relatively simple browsers, and which has grown out of two years practical experience with wide-area hypertext information systems in the Internet community.

HTML+ and HyTime

The HyTime standard provides a rich range of architectural forms, but is not aimed at run-time efficiency. Suggestions have been made as to how the HTML DTD could be adapted to comply with HyTime's clink architectural form [Kimber 93]. This would necessitate documents declaring links as external entities and the use of local names in link definitions, but in the absence of any immediate benefit, there has been little enthusiasm for this within the World Wide Web community. Instead, it is believed that a straightforward filter program should be used to map HTML and HTML+ documents into a format which is strictly compliant with HyTime, when this becomes appropriate.

A simple example of HTML+

The following is a simple example of an HTML+ document,which illustrates the basic ideas involved in SGML.

```
<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. Notice that unlike title and header tags,
there is no matching end tag.
```

The text of the document includes tags which are enclosed in <angle brackets>. Many tags have 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 parameters, e.g. the "id" attribute in the header tag, which is used to define potential destinations for hypertext jumps.

Unlike most document formats, SGML leaves out the processing instructions that determine the precise appearence of the document, for example the font name 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), and so 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.

Practical experience has shown that people often make mistakes when they have to type in the markup for themselves. As a result, most browsers are tolerant of bad markup. This problem is being minimised by keeping the format as simple as possible and encouraging the development of WYSIWYG editors.

The HTML+ Document Format

The following sections go through the various features of the format with suggestions as to how browsers should render them. The DTD for HTML+ is given in Appendix I.

Parsing HTML+ Documents

By default, HTML+ documents are made up of 8-bit characters in the ISO 8859 Latin-1 character set. In future, 16 bit character sets may be used to cover a wider range of languages. The HTTP network protocol uses the MIME standard (RFC 1341) to specify the document type and the character set. It is assumed that the chosen character set includes the printable 7 bit US ASCII characters as a subset.

The DTD specifies the syntax of the document structure, in particular, which tags are permitted in any given context. Certain tags are only permitted at the start of the document. Tags and attribute names are case insensitive, thus <TITLE> is equivalent to <title>. End tags may be minimised to </> instead of say </title>.

In general, SGML entity definitions are used to represent characters which would otherwise be confused with markup elements:

`&`	is represented by	`&`
`<`	is represented by	`<`
`>`	is represented by	`>`

Such entity definitions should be used in all places except within attribute values for tags (tag names and attribute names cannot contain these particular characters). Entity definitions can also be used for special characters, e.g. "é" for a small e with an accute accent. The full list is given in Appendix II. Additional entities may be defined within documents using the SGML entity declaration tag !ENTITY, e.g.

```
<!ENTITY % sgml "Standardised General Markup Language">
```

The browser will then insert the full form whenever it comes across "&sgml;".

Repeated white space characters such as space, tab, carriage return, line feed and form feed are ignored except within preformatted text, i.e. it doesn't matter which white space characters you use or how many of them you put between words, or before or after markup elements, the effect is the same as a single space character.

It is recommended that HTML+ documents start with the following external identifier, indicating that the document conforms to the HTML+ DTD. This will ensure that other SGML parsers can process HTML+ documents, without needing to include the DTD with each document.

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

HTML+ departs slightly from pure presentation independence by allowing authors to specify rendering hints, e.g. to use a bold font for a given type of emphasis. This step was taken to give authors greater control over the final appearence, and is based upon practical experience with the earlier HTML format. In addition, attribute values are used to distinguish different subcategories of markup, rather than adding extra tags. New logical categories of emphasis etc. can be added in future without needing to change existing browsers. These decisions have made it practical to restrict HTML+ to a very small set of tags.

Backwards Compatibility with HTML

The format is designed to be largely compatible with the earlier format HTML, and HTML+ browsers will be able to display documents in the HTML format with little extra cost. Suggestions on how to map HTML elements to HTML+ are given in Appendix III.

Notes for Implementors

Please ensure that browsers can tolerate bad markup. In practice, this is quite straightforward to achieve, provided a naive top-down SGML parser is avoided. A forgiving parser should be able to cope with tags in unexpected positions, e.g. the <A> tag bracketing a header[2]. Unknown tags should be simply ignored.

Implementors should endevour to make sure that documents can be scrolled efficiently regardless of their length. Always parsing from the start of the document leads to jerky performance. Two strategies for efficiently scrolling through documents are:

a) Establish regular landmarks throughout the document for which the state of the parse is known. The browser can then work forward from the nearest landmark, when it needs to refresh the screen after a scroll operation. The landmarks need updating when users make changes, while using a WYSIWYG editor.

b) When scrolling up, parse backwards to work out the state at earlier points in the document. This can be done via a combination of skipping back, looking for markup which causes a line break etc. and then parsing forward until the current position, to find the change of state. This can be repeated until the parser reaches a point prior to the new top of the window.

Practical experience has shown the importance of providing cues to users on progress in retrieving documents over the network. These will depend on the protocol, but should show how much data has been received at any point. The network connections shouldn't block, and an abort button is essential[3]. It is generally better to avoid displaying the retrieved document in a new window, unless explicitly requested by the user, e.g. by holding down the shift key when clicking the hypertext link.

Normal Text

This is generally shown with a serif font and wraps on the right window margin. It can include:

- Entity references, e.g. `">" and "é"`
- Significant Line breaks (the BR tag)
- Non-breaking spaces - the SP tag
- Hypertext links - the A tag
- Inlined graphics or icons - the ICON tag
- Various styles of logical emphasis - the EM tag
- Embedded data in an external format, e.g. TeX equations - the EMBED tag
- Input fields for forms - the INPUT tag

Line breaks and

This tag causes the renderer to start a new line at the current left margin setting. There is no corresponding end tag. The BR tag is *empty*, that is to say, it doesn't act as a container around other text or markup.

---

[2]Headers typically cause a line break and leave a vertical gap. If the hypertext link definition is parsed prior to the beginning of the header, the starting position for the button will be in the wrong place - browsers should therefore adjust this position to the beginning of the text.

[3]For X11 on Unix systems, the *select* system call can be used with non-blocking I/O to poll the event queue at regular intervals. The *XtAddInput* call acts as a wrapper around select for this very purpose. Users can then continue to view the current document as well as being able to click an abort button (which sets a global variable, polled by the comms software). Be careful to disable unsafe actions, e.g. trying to get a second document while still waiting to get the first (a race hazard).

Non-breaking spaces and <SP>

This allows authors to be certain that browsers won't break a line at an inappropriate place. Authors may also use SP at the start of a line to indent text. This is deprecated.

Some authors like to have a slightly longer space after punctuation at the end of sentances. While this is a stylistic issue, browsers need to be able to distinguish periods which denote the end of sentances from those used in abbreviations. One way of doing this in HTML+ is to use the EM tag to delimit abbreviations.

Hypertext Links

When the user clicks on a hypertext link in the document, the current document is replaced by the one referenced by the link. Links can be made to a wide range of document types, based on the URL[4] and URN[5] notations. Some document types permit links to be made to specific sections within a document[6]. The syntax for links within the same document or to documents in the same directory is particularly simple:

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

In a browser this might look like:

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

The first link is to an anchor named "z1" in the current document. The second is to a file named "links.html" in the same directory as the current document. The caption for the link is the text between the start and end tags. The value for the HREF attribute defines the destination point, and can be abbreviated in certain cases. If practical, word the caption in such a way that continues to make sense when the document is printed out. The link should be shown in a clearly recognisable way, e.g. as a raised button, or with underlined text in a particular color. For displays without pointing devices, it is suggested that a reference number is given in square brackets, which can then be typed by the user.

A more general discussion of hypertext links and their treatment in HTML+ is presented in a later section.

Inlined Graphics or Icons

These are treated like characters and inserted as part of the text, e.g.

```
This line has a egyptian hieroglyph at the end of the
line. <img src="ankh.tiff">
```

The URL notation is used to name the source of the graphics data. The *align* attribute can be used to control the vertical position of the image relative to the current text line in which the IMG element is placed. Use a value of "top", "middle" or "bottom" to align the top, middle or bottom of the image with the current text line. The *seethru* attribute allows authors to include a chromakey, i.e. a colour that designates portions of the image to be left unpainted so that the background shows through. The format for this attribute's value is dependent on the type of graphics data, and has yet to be defined.

Note that you can create simple iconic buttons, e.g.

```
<a href="bigpic.gif"><img src="littlepic.tiff"></a>
```

If the user clicks anywhere on the image, this will cause the browser to retrieve its bigger version. This approach allows users to preview images which may take significant time to download. Note that there is little additional penalty for displaying the same image at multiple points in the document. The *ismap* attribute is provided for backwards compatibility with HTML. When present the browser will send all mouse clicks and drags on the image, to the server.This mechanism is explained in more detail for the FIG tag.

---

[4]The notation for universal resource locators is defined in [Berners-Lee 93b].

[5]The notation for universal resource numbers and the protocol for resolving them to the nearest available copy is currently under study by the IETF URN working group.

[6]At the time this document was written, such links were restricted to named anchors within HTML and HTML+ documents

Sophistocated HTML+ editors should allow authors to modify images using an external editor. Larger images should be specified with the FIG tag.which provides support for flowing text around figures, along with captions, overlays and active areas.

Various Styles of Emphasis[7]

This allows you to emphasise a portion of the text. The simplest approach is:

```
<em>default emphasis, usually shown in an italic font</em>
```

The logical role of emphasis denotes the semantic significance, e.g. a citation, or text to be input by a user for a computer program.The physical style of emphasis controls its appearence. Note that EM elements can include inlined graphics.

Logical Role of Emphasis

It is strongly recommended that the logical role of the emphasis is given with the *role* attribute, e.g.

```
<em role="cite">a citation</em>
```

Providing a logical role allows browsers to apply differing rendering styles according to the role, but more importantly, it allows indexes to be constructed automatically, e.g. the list of bibliographic references in a technical report. These can be used for searching through collections of documents according to semantic keys giving better focussed searches compared with full text indexes .

The list of recommended roles are as follows:

For references to other works:

CITE	a reference to a related work
PUB	a publication containing a referenced work
AUTHOR	an author of a referenced work
EDITOR	an editor of a referenced work
CREDITS	e.g. the rights owner of a photograph
COPYRIGHT	the holder of the copyright
ISBN	for ISBN numbers
ACRONYM	for acronyms like "NATO" and "US"
ABBREV	for abbreviations

For annotations:

FOOTNOTE	shown as footnote or pop-up
MARGIN	shown as margin note or pop-up

For computer instruction manuals:

DFN	defining instance of a term
KBD	something a user would have to type
CMD	command name, e.g. "chmod"
ARG	command arguments, e.g. "-I"
VAR	named place holder, e.g. "filename"
INS	an instanced of a named printer, directory or file etc.
OPT	an option of some kind
CODE	an example of code (shown with a fixed pitch font)
SAMP	a sequence of literal characters

On dumb terminals annotations should be shown in round brackets. Margin notes should be right aligned, and may include graphics via the IMG tag. The set of recommended roles will be kept by the HTML+ registration authority.

---

[7]The name EM was chosen in preference to EMPH because it allows existing HTML browsers to show all HTML+ emphasis in italics. It also allows HTML+ browsers to correctly process the common case for emphasis in HTML documents.

Physical Styles

The appearence can be modified by adding optional rendering hints from the list:

```
<em b>              bold text
<em i>              italic text
<em u>              underlined text
<em sup>            superscript text
<em sub>            subscript text
<em tt>             type writer font (courier)
<em hv>             sans serif font (helvetica)
<em tr>             serif font (times roman)
```

These hints can be combined, e.g.

```
<em b i> for bold italic text </em>
```

Note that these are only hints and may be ignored by browsers. Indeed, arbitrary combinations will present difficulties for most browsers. If the display is limited to a single font, colour or underlining can be used, but should be clearly differentiated from hypertext links and headers. Dumb terminals can use email conventions, e.g. switching to all capitals, or delimiting with the * or _ characters. Subscript and superscript text should be shown in a smaller point size, vertically offset as appropriate.

Browsers may choose to simplify or ignore hints, but should aim to do so in a consistent manner. At the simplest level, browsers can ignore the attributes and render all emphasis in the same style.

Nested Emphasis

Emphasis can be nested as in:

```
<em b>bold text, and <em i>bold italic text</em></em>
```

Nested emphasis is better suited for grouping logical roles together, for instance, you could use the EM to separately tag author, title, and publication, and then wrap these up as a citation. Without this, indexing programs will have difficulty in grouping markup into the correct references.

Embedded data in an external format

The EMBED tag provides a simple form of object level embedding. This is very convenient for mathematical equations and simple drawings. It allows authors to continue to use familiar standards, such as *TeX* and *eqn*. Images and complex drawings are better specified using the FIG or IMG elements. The *type* attribute specifies a MIME content type and is used by the browser to identify the appropriate shared library or external filter to use to render the embedded data, e.g. by returning a pixmap. It should be possible to add support for new formats without having to change the browser's code, e.g. through using a common calling mechanism and name binding scheme. Sophistocated browsers can link to external editors for creating or revising embedded data. Arbitrary 8-bit data is allowed, but &, < and > must be replaced by their SGML entity definitions.

Input Fields for Forms

Input fields can be arranged with considerable freedom, as part of normal paragraphs, preformatted text, lists or tables. Examples of how to do this are given later on in the section describing the FORM tag. The INPUT tag has the following attributes:

name Used to name this input field, e.g. `name="phone number"` (required attribute).

type Defines the type of data the field accepts (the type name is insensitive to upper/lower case). If missing, the field is assumed to a a free text field.

size Specifies the size/precision of the input field according to its type (optional).

value The initial value for the field, or the value when checked for checkboxes and radio buttons (optional, except for radio buttons).

checked When present, this attribute indicates that a checkbox or radio button is selected.

disabled When present, this attribute indicates that this field is temporarily disabled. Browsers should show this by greying out or via a similar visual clue. Users are unable to set the focus to disabled fields, or change their values.

error	When present, this attribute indicates that the current value for this field is in error in some way, e.g. because it violates some consistency constraints. Browsers should indicate this by a change to the shape and colour (red) of the field's border. This should be accompanied by an error message and a beep.

The following types of field should be supported:

TEXT	Single or multi-line text entry fields. Use the *size* attribute to specify the width and height in characters, e.g. size="24" or size="32x4".
URL/URN	For fields which expect document references.
INT	For entering integer numbers, the maximum number of digits may be given with the *size* attribute, e.g. size=3 for a 3 digit number[8].
FLOAT	For fields restricted to floating point numbers.
DATE	Restricted to a recognised date format.
CHECKBOX	Use these for simple boolean attributes, or for attributes which can take multiple values at the same time from some set of alternatives.
RADIO	Use these for attributes which can take a single value from a set of alternatives (groups input fields with the same *name*).

For the purposes of sending the contents of a form to a server, as part of a query, the input fields are mapped to a list of properties. In most cases the *name* and current *value* are used to define a property/value pair for each field. Radio buttons and check boxes are left out if they are unselected. This ensures that only the selected radio button yields a property/value pair. By missing out the *value* attribute for check boxes, these fields will map to a simple (value-less) property. The representation of property lists is defined as part of the HTTP protocol.

Browsers can choose to notify the server whenever a field is changed (i.e. when a field looses the focus and its contents have changed) or wait until the form is completed. This choice will depend on network latency.

Headers and Titles

The title tag is generally used to define the window banner when viewing a particular document, e.g.

```
<title>Reference Guide to HTML+</title>
```

This element should appear at the start of the document. There are six levels of headers, H1 to H6, with H1 the most important, and H6 the least. A common convention is to begin the body of the document with a level one header. e.g.

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

Header names should be appropriate to the following section of the document, while the document 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. Browsers should render headers with a line break before and after the header text. A common convention for headers is to use a sans serif font, e.g. Helvetica, with a smaller point sizes for less significant headers, and a serif font, e.g. Times Roman, for normal text.

Headers can include an identifier, unique to the current document, for use as destinations of hypertext links, e.g.

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

This allows authors to make 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 may automatically generate the identifiers. In this case, they should also provide a point and click mechanism for defining links, so that authors don't need to deal explicitly with the identifiers.

---

[8]Perhaps the syntax should permit integer ranges, e.g. size="1 to 6", in which case a more appropriate name for the attribute than *size* would be desirable.

The attribute "margin" when present acts as a hint to the browser to insert the header into the margin and causes the following text to be vertically aligned with the start of the margin header. By convention, margin headers are left justified, e.g.

```
<h4 margin> Deleting the Curve </h4>

The Delete command allows you to delete any selected symbol or text
block.
```

Note that headers don't act as containers for the subsequent text. You can group the header and text with the GROUP tag, see later for details.

Indexing

A good index plays an important role in helping users find their way to the material they need. It allows users to type in one or more keywords to see a meaningful list of matching topics. Alternatively they can browse through the index and take advantage of serendipity, and gain a feeling for the limits of what is covered in the associated document. The two approaches can be combined, when the characters typed act dynamically to control the viewing position within the index. Typically each keyword entry in the index is associated with one or more topics. This notion of guiding the user is absent from full text indexes like WAIS, where users are given very little help in choosing the keywords to search on.

Generating a conventional index for a document is a skilled task, and HTML+ allows authors to include annotations for creating an index. These directives can be included with document titles, headers and emphasis etc. using the *index* attribute. This allows each such element to be included in one or more entries in the index, under primary or secondary keys, e.g.

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

This resulting index looks like:[9]

- 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.

Where each entry is a hypertext link to the associated anchor. The *index* attribute can specify multiple entries, each separated with the ";" character. The optional secondary key (*shielding from as difficult*) is introduced by the "/" character. Secondary keys are useful when the primary key occurs more than once. To allow for future extension, primary keys should not start with the "#" character. This prefix is being reserved to designate indirect index entries. Use "\/", "\;", "\#" and "\\" to escape "/", ";", "#" and "\" respectively.

Paragraphs and Preformatted Text

HTML+ includes support for paragraphs and preformattted or verbatim text.

Defining Paragraphs with <P>

The <P> tag splits normal text into paragraphs. Unlike headers, there is no corresponding end tag, so don't use </P>. The following optional attributes can be used:

id An identifier, unique to this document, which can be used as a destination in a hypertext link. Note that the paragraph tag acts as a container for the paragraph.

[9]Taken from K. Eric Drexlers's "Nanosystems, Molecular Macherinery, Manufacturing and Computation".

role The role of the paragraph, see the following list for supported types.

align A rendering hint to the browser to justify lines. The supported values should be: `align="left"`, `align="center"` and `align="right"`. This is useful for single line paragraphs or when the lines are made explicit with the
 tag.

indent When present, this hint suggests that the left and right margins are indented by an amount dependent on the browser, e.g. about 4 character widths.

The *role* attribute is used to indicate the logical role of the paragraph, e.g. a stanza in a poem or a cautionary note in a computer manual. Browsers may apply particular rendering styles to certain roles. The role name is case insensitive. The following roles are recommended:

quote A paragraph quoted directly from some other work. Browsers could indent the paragraph and maybe use a different font.

byline Information about the author of the document, e.g contact details. This could be displayed in a different font, and perhaps right aligned.

note Advisory note in an instruction manual. The browser could display a hand icon in the margin.

caution Cautionary note. The browser could display an warning road sign in the margin.

error A note describing error conditions. The browser could indicate the importance of the note by displaying a stop sign in the margin.

An example of a paragraph element:

```
<p role="note"> If you accidentally delete a symbol other than the red
circle, immediately press ALT+BKSP to choose the undo command, and
then select the red circle and delete it again.
```

Paragraphs can be rendered by indenting the first line, or by leaving a vertical gap equal to half the current line spacing. When using the latter style, browsers should take care to avoid this vertical gap when the paragraph element immediately follows a header. This rule ensures that authors can tag paragraphs directly following a header without causing unwanted extra space before the start of the text.

Ordered, Unordered and Definition Lists

There are three kinds of lists: ordered or numbered lists, unordered lists and definition lists. Ordered and unordered lists can be nested arbitrarily, and browsers should progressively inset the left margin for each level of nesting.

Ordered Lists with <OL>

The list items are automatically numbered, e.g.

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

Is displayed as:

1) Wake up
2) Get dressed
3) Have breakfast
4) Drive to work

The *compact* attribute when present has the effect of reducing interitem spacing, e.g. <ol compact>. Authors can also make both the OL tag and the LI tag potential destinations for hypertext links with the *id* attribute. List item text can include normal text and paragraph elements, but not headers.

Unordered Lists with <UL>

These are bulleted lists, e.g.

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

Is displayed as a bulleted list:

- Wakeup
- Get Dressed
- Have breakfast
- Drive to work

The *compact* attribute when present has the effect of suppressing bullets and reducing interitem spacing, e.g. <ol compact>. Multicolumn lists can be requested with the *narrow* attribute, e.g. <ul narrow>. This causes the browser to try to lay out the list as a number of columns, depending on the window width. This attribute should only be used when all the items are less than 20 characters long. Authors can also make both the UL tag and the LI tag potential destinations for hypertext links with the *id* attribute. List item text can include normal text and paragraph elements, but not headers. For nested unordered lists, browsers may use different bullet symbols for different levels, in addition to progressively insetting the left margin. The *src* attribute on the LI tag can be used to specify an icon for use in place of the standard bullet symbols.

Definition Lists with <DL>

These consists of pairs of terms <DT> and definitions <DD>. The following example is part of a french dictionary:

```
<DL>
  <DT>endetter
  <DD>Engager dans des dettes

  <DT>endeuiller
  <DD>Plonger dans le deuil, remplir de tristesse

  <DT>endiablé, ée
  <DD>D'une vivacité extrême
</DL>
```

Is commonly displayed as:

endetter	Engager dans des dettes
endeuiller	Plonger dans le deuil, remplir de tristesse
endiablé, ée	D'une vivacité extrême

With the *compact* attribute, e.g. <dl compact>, this is altered to:

endetter Engager dans des dettes

endeuiller Plonger dans le deuil, remplir de tristesse

endiablé, ée D'une vivacité extrême

In this style, the term and definition appear in the same paragraph, with the term text emphasised in a bold font. The definition text follows on, and wraps to a left margin a little further inset than the term text. This style is common place in dictionaries.

Term text following the <DT> is restricted to normal text. The definition text after the <DD> tag can additionally include paragraph elements and ordered/unordered lists. Headers are not allowed in either case. Authors can make the DL, DT and DD tags potential destinations for hypertext links with the *id* attribute.

Authors are reminded to check that DT and DD are paired up. Common misunderstandings lead to people repeating DD tags to separate paragraphs (use <P> instead), or leaving out the DT tag altogether to indent text (use <p indent> or <group indent>). The ability of browsers to cope with bad markup seems to encourage such problems, which will hopefull fade away as wysiwyg editors become commonplace

Figures

Figures provide great flexibility:

- ❏ linked or embedded graphics
- ❏ control of picture alignment and text flow
- ❏ Figure description for when the image can't be shown
- ❏ caption placement
- ❏ scaled or pixel-based coordinates
- ❏ hypertext links with active areas
- ❏ text and image overlays

The following simple example will set the scene for the description of the various features:

```
<fig align="right" src="map.gif"> How to get to my house </fig>
```

Here, the image is defined by a link to an external document. The caption "How to get to my house" will appear at the bottom of the image. The *align* attribute directs the browser to display the figure at the right of the widow, and to flow subsequent text around the left of the image.

Using embedded graphics data

Instead of the *src* attribute, you can include an EMBED element immediately following the <fig> tag. This is useful for graphs etc. defined in an external format.

Figure Description

The FIGD tag allows you to give a textual description which can be shown when the figure itself can't be shown, e.g. for browsers working on dumb terminals, e.g.

```
<FIGD> This is an aerial photograph of central London, showing
Buckingham Palace and the Houses of Parliament. On the left you can see
Hyde Park and in front the Albert Hall and the Natural History
Museum.</FIGD>
```

Alignment and Text Flow

The *align* attribute controls the horizontal position of the figure: "left", "right", or "center". The default is "left". Browsers may flow text when there is sufficient room, unless the figure is center aligned or the *noflow* attribute is present.

Caption Placement

The *cap* attribute allows you to ask the browser to position the caption text to the "left", "right", "top" or "bottom". The default is to place the caption at the bottom of the figure. Text flow will occur around the figure and caption, leaving a suitable gully. The browser will ignore this attribute if there is insufficient room for the requested placement.

Pixel-base or Scaled Coordinates

The upper left of the figure is designated as x,y = (0, 0), with x increasing across the page, and y down the page. If points are given in real numbers, the lower right is taken as being (1.0, 1.0), otherwise with integer values, the coordinates are assumed to be in pixels[10]. Note that using scaled coordinates is much safer, especially for graphics! The extent of the image in pixels may change, e.g. as a result of format negotiation with the server, and by retrieving images with lower resolution when network performance is poor.

Active areas

The *ismap* attribute causes the browser to send mouse clicks on the figure, back to the server using the selected coordinate scheme. The mouse button-up event is sent with the URL formed by adding "?x,y" as a suffix to the URL for the current document. You can also designate rectangular regions of interest in the picture by holding the mouse button down while dragging the mouse. The browser should show a rubber band outline for the rectangle defined by the current location of the mouse pointer and the point at which the mouse button was pressed. The region is named by taking the current URL and adding the suffix: "?x1,y1;x2,y2", where (x1, y1) and (x2, y2) define the points at which the mouse button went down and came up, respectively. The *ismap* mechanism is relatively slow, but makes sense when the active regions change their boundaries over time, e.g.

```
<fig ismap src="weather.gif">Click on your area for todays weather</fig>
```

You can also designate arbitrary areas of the figure as hypertext links. Mouse clicks are handled locally, and the browser can provide visual clues that the pointer is over an active area, for example, by changing the pointer from an arrow to a hand symbol, or highlighting the area in some way.

Active areas are defined with the FIGA tag. This has two attributes:

href A URL specifying the link to traverse when clicked (required)

area Defines a polygonal[11] area as a list of points: "x1, y1; x2, y2; ..." (optional)

The *area* attribute lists a sequence of points defining a polygon. Closure is ensured by joining the last point in the list to the first (i.e. a triangular area is defined with a list of 3 points).When the *area* attribute is missing, the whole of the picture is assumed. Polygons may be non-convex or even intersect themselves, thereby complicating the definition of what is enclosed by the polygon. Holes should be excluded. Note that active areas defined with FIGA take precedence over the *map* mechanism.

Overlays

The FIGT tag allows you to position text and image overlays on top of the figure, e.g.

```
<fig src="map.giff">
  <figt at="0.2, 0.3" framed>A text overlay</figt>
  The figure caption
</fig>
```

The overlay can contain a wide variety of elements including text, images (IMG), lists and tables. Figures shouldn't be nested. Any hypertext links in the overlay text will take precedence over the *href* attribute in FIGT. The following attributes are permitted:

at The upper left of the overlay, relative to the figure.

width As a fraction of the figure, e.g. width="0.3". This allows you to limit the lengths of wrapped text lines. The vertical extent is then determined automatically.

framed Directs the browser to draw a frame around the overlay and to colour in the background in some way.

href Allows you to make the overlay into a hypertext button.

---

[10]This mechanism was designed to be backwards compatible with the *ismap* feature as used with IMG in HTML, and as a consequence forces the choice of y increasing down rather than up the page. A simple test to distinguish the two schemes is to check if the "." character occurs anywhere in the list of points.

[11]The code for hit testing polygons is tricky, but quite fast. A public domain version of the code would be helpful.

Tables

Tables are defined with the TBL tag. Cells are designated as being headers or data. You can join adjacent cells, e.g. to define a header spanning two columns.

An Example of a Table

	average		other
	height	weight	category
males	1.9	0.003	yyy
females	1.7	0.002	xxx

This is defined by the markup:

```
<tbl border>
   <tt top> An Example of a Table
   <th rowspan=2> <th colspan="2"> average <th> other <tr>
   <th> height <th> weight <th> category <tr>
   <th align=left> males <td> 1.9 <td> .003 <td> yyy <tr>
   <th align=left> females <td> 1.7 <td> .002 <td> xxx
</tbl>
```

The *border* attribute for TBL directs the browser to draw borders. The *compact* attribute is used when you want the table to appear in a smaller size.

The optional <tt> tag defines a title. By default (i.e. when *top* is missing) this should be positioned below the table. The <th> and <td> tags define header or data cells respectively. The <tr> tag acts as a separator between rows. In the example, you can see that the first header in each of the first two rows is void.

TH, and TD all have the same permitted attributes:

colspan Columns spanned by this cell, see example

rowpan[12] Rows spanned by this cell, see example

align=left Left justify the cell's content

align=center Center justify the cell's content

align=right Right justify the cell's content

By default, headers are centered, while other cells are left justified. If practical, browsers should be smarter than this, e.g. if all the cells in a column are shorter than the column header, then indent the cells to make them appear under the middle of the header.

Browsers need to carry out a pre-parse (e.g. when sizing the vertical scroll bar) in order to determine the number of columns and their widths. The following guidelines may be useful:

- There is no need to declare empty cells at the end of a row, so the number of columns for the table is given by the row with the most columns.
- Restricting text to a fixed pitch font may simplify matters.
- If a column only contains numbers or empty cells then align on units and set width to maxium precision needed (before and after decimal point, allowing for an exponent). This rule also applies when currency symbols are used.
- Otherwise set column width to the minimum of a threshold width and the maximum text length for all cells in the column. Text is left aligned and wrapped if it exceeds the chosen column width.

[12]This is tricky to handle. The parser should carry a spanned cell over to the next row, the definition of which should miss out the spanned cell, i.e. the next row will have one fewer explicit cell definitions.

The threshold column width can be set according to the number of columns and the width of the display window. It is also necessary to take the column headers into account in this process. Header text wraps to the next line if the column is too narrow. Browsers will by default center the header in the column.

A complication occurs when a header or data cell spans more than one column, as specified by the ***s*** attribute. This can be used to give complex headers which share a header between columns followed by individual headers on the next line.

Vertical gaps can be introduced with the <tb> element - this inserts 1/2 line space into the next row. Header and Data rows can be intermixed. Authors can use alternate header and data rows when the rows alternate between text and numbers. The vertical alignment of numbers only applies to data fields.

Tables which don't fit into this model should be defined as figures using an external format, e.g. Postscript, Tex or Computer Graphics Metafile.

Forms

A document can include one or more forms. Each form is defined by a FORM element, which contains a number of input fields laid out with normal and preformatted text, lists and tables. The browser should manage the input focus, e.g. with the tab key and mouse clicks. The Return key can be used to mean that the user has filled in the form and wants the appropriate action to be taken. Browsers may also display "Accept" and "Cancel" buttons as part of the document (or perhaps on another part of the browser). Note that forms shouldn't be nested.

The action to be taken is specified by the *action* attribute of the FORM tag. If missing the URL for the current document is assumed. This attribute uses a URL to specify a server to query, or an email address to send the form to. When sending the form to a server as a query, the form's contents are encoded as a property list (see definition of the INPUT tag). The precise encoding is dependent on the HTTP protocol and defined in [Berners-Lee 93c][13]. When the form is to be mailed, it is first converted into plain text, closely resembling the appearence on the screen. You can include multiple RFC 822 mail headers with the MH tag. The *hidden* attribute may be used to hide the headers when browsing the document. The following is an example of a simple questionaire:

```
<form action="mailto:www_admin@info.cern.ch">

<mh hidden>
  Subject: WWW questionaire
</mh>

Please help us to improve the World Wide Web by filling in the
following questionaire:

<p>
Your organisation? <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 compact>

<li> X Mosaic <input name="browsers" type="checkbox" value="xmosaic">

<li> Cello <input name="browsers" type="checkbox" value="cello">

<li> Viola <input name="browsers" type="checkbox" value="viola">

<li> Others? <input name="other browsers" size="48x4">

</ol>

A contact point for your site: <input name="contact" size="48">

<p>Many thanks on behalf of the WWW central support team.

</form>
```

[13]This and the *ismap* feature rely on the forthcoming definition of HTTP as an official Internet standard.

Floating Panels

The PANEL tag can be used to define panels or boxes which are free to float with respect to the standard flow of text. These are often used in magazine articles for asides on background material. The panel is typically shown with a distinctive background colour and border. The lay out software positions the panel to coincide with the page boundaries in printed media. For on-line use, panels can be rendered as pop-up windows. The body of the panel can be defined by a link to a separate document or included in the current document.

The following optional attributes are permitted with the <panel> tag:

id An identifier, unique to this document, which can be used as a destination in a hypertext link.

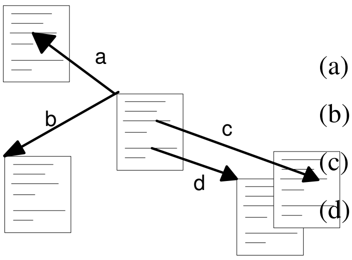
at An identifier elsewhere in this document. The panel mustn't be placed before this point. (Defaults to the current position if the *at* attribute is missing).

href This attribute allows authors to fill the panel from a separate document, as specified by a URL. Note that the matching end tag: </panel> is always needed.

The text contained by the panel element can include any of the markup elements and looks like a separate document (panels themselves can't be nested). If the *href* attribute is used the text delimited by <panel> ... </panel> may be used as the caption for a pop-up. The *at* attribute allows you to include the panel definition at a convenient point in the HTML+ document, rather than interrupting the main flow of the document.

More on Links

Before describing the details of how links are represented in HTML+ it is worth looking more generally at the nature of hypertext links. First a terminological point: a *node* is the atomic unit for information retrieval, while *documents* may consist of one or more nodes, perhaps arranged as a hierarchy. A node may even be shared between several documents. Hypertext links start and end on nodes or anchor points within nodes.



The diagram illustrates the basic possibilities:

(a) Link from a node to an anchor

(b) Link from a node to a node

(c) Link from an anchor to another anchor

(d) Link from an anchor to a node

Links in HTML+ are represented with the LINK and A tags. The LINK tag is used for cases (a) and (b), while the A tag is used for cases (c) and (d). These links are held in the source node only, so there is a risk that the destination may have disappeared. Organisations can manage this risk by continued support for a few well published nodes (servers can use redirection to hide name changes). Links to other subsidiary nodes are at higher risk. This structured approach allows people to become familiar with the major routes through the web, without needing to worry about the minor routes.

In most cases URLs and URNs explicitly specify a node/anchor. The nodes may be explicit files or generated as the result of some process invoked by the server, e.g. a hypertext listing of a directory or a list of matches for a given search string. The search string can be explicitly encoded as part of a link, or dynamically defined by the user (see the ISINDEX tag, as described later on).

Links may be held separately from the source and destination nodes[14]. This is particularly appropriate for annotations and discussion groups. For example, consider making an annotation on a document held by a server located far away in another organisation. You could take a local copy and directly annotate it, but this is only appropriate for private use. The remote server might even support a protocol to add your annotations in place. More likely though, you will have to use an annotation server. This mechanism can be used to obtain a

[14]These correspond to HyTime's *ilink* architectural form.

copy of the document with the annotations inserted as hypertext links and shown as pop-ups or separate documents.

Context Dependent Links

For discussion groups, responses are made asynchronously, and include one or more references to other articles. In this situation, context dependent links are appropriate. The resolution to an explicit node can be carried out by either the client or server. The former approach is often appropriate, but requires special support, e.g. for network news and nntp.

Context dependent links are also useful for links to the table of contents for documents consisting of multiple nodes, when some of the nodes also appear in other documents. The appropriate table of contents for a given node will depend on which document is currently being viewed. In this case, the context will depend on how the current node was reached. This is quite simple to track if the links from the table of contents are differentiated from cross reference links.[15]

Hypertext paths are recommended routes through a set of nodes, and generally shown by next and previous buttons on a toolbar. Paths can be defined using explicit links in a node, or held separately in another node. The latter case once again, depends on the context. Paths and tables of contents all fall under the general category of navigating around a hierarchy of nodes forming a document too large or unwieldy to be held in a single node.

Types of Links

There are several motivations for differentiating between types of links:

how it is viewed The potential to show different cues depending on the type and size of the node to be retrieved. If this information is explicitly stated as part of the link, there is the risk that it will become out of step with the linked node.

what happens Whether the linked document replaces the current one, or appears in a new window, or as a pop-up overlay on top of the current one.

printed appearance Whether links are treated as references, footnotes or as separate sections

effect on context After traversing the link, will there be implicit values for the table of contents, and hypertext path etc?

Link Attributes

The A tag has the following attributes:

id An identifier unique to this document which can act as a hypertext anchor

name The same as *id* and included for backwards compatibility with HTML. New documents should use the *id* attribute for consistency with the other tags.

href The URL or URN identifying the destination of the link.

role A string giving the role of the link, e.g. `role="partof" or "annotation"`

effect A string defining how the linked node is shown: "replace", "new", "overlay", with the default effect of replacing the current document.

print How should the link be printed: "reference", "footnote" and "section", defaulting to "reference" (i.e. a footnote stating the link's URL).

title The title to show when otherwise undefined for the node.

type The MIME content type for the linked node for use with presentation cues.

size The size in bytes for the linked node. This allows the browser to show a gauge indicating progress in retrieving long documents or images etc.

The LINK tag has only the *href* and *role* attributes.

---

[15]This is more general than deriving the role of the link from that of the node alone.

The *role* attribute is appropriate when context dependent properties such as table of contents (toc) are implied for the linked node, e.g. if the current node is a toc (as defined by the html or group tags) and the link has the role "partof", then the current node should act as the *toc* for the linked node. This property propagates down "partof" links, but not normal links. The *next* and *prev* properties are given by the sequence of "partof" links in the parent node. The *parent* property is only defined if the current node was reached via a "partof" link.

The LINK tag is used to express these properties in an explicit form, e.g.

```
<LINK href="toc.html" role="toc">
```

The recommended property names are:

toc Table of contents for current node.

next The next node in a hypertext path.

prev The previous node in a hypertext path.

parent The next level up in the hierarchy.

style The style sheet appropriate to this node.

Style sheets provide a way for authors to express their detailed preferences for fonts, and layout, whether for the screen or when the node is printed out. A possible format is given in [Raisch 93].

The *effect* attribute is a hint and may be disregarded by browsers. It allows you to click on an image and to see a linked movie as an overlay at the same position. The browser tries to position the overlay at the same origin as the link. In some cases, the linked node is a description of the current node. By including `effect="new"`, the linked node will appear in a new window so that users can see both nodes at the same time. This hint should be used sparingly!

The *print* attribute makes it practical to print nodes along with relevant linked nodes. By default each link appears as a footnote stating the link's URL. Short nodes can be included in their entirety as footnotes, and longer ones as sections in their own right. This approach could be extended in future, to reorder the sequence of nodes from that defined by the position of the links in the source node, and to control the level that nodes appear as, e.g. chapter, section or subsection.

The *title* attribute is useful for nodes without titles of their own, e.g. Gopher menus. The *type* attribute can be used to show cues for the node type, e.g. iconic decorations[16]. The size attribute allows browsers to show a gauge on how much of a document has been retrieved at a any time. These attributes are liable to get out of step with the target node, and should be treated as hints only.

Groups

The GROUP tag allows you to define arbitrary groups, e.g. books, chapters, and sections. The *role* attribute is used to name the logical role of the group. You can use most markup elements inside a group element, including group itself. The *inset* attribute is a rendering hint to inset the left margin. Using the A tag with role="partof" allows you to designate a node as being included within the group, allowing hierarchies of groups which cross multiple nodes. See previous discussion of how properties are propagated.

Groups offer opportunities for presenting and searching documents at different levels of abstraction. For example, you might first describe a book by its title, author, publisher and ISBN number. The next level down could add a cover illustration together with a summary of the book's contents, some comments by reviewers and a short biography of the author. A number of books to be presented in an iconic form using a miniature version of the "cover page". Publishers could include copyright and other details in a standard place.

[16]The appropriate cue might also depend on the role of the link, e.g. for annotations browsers could show an icon of a drawing pin (as in attaching a note to a pin board). The colour of the pin could then vary according to the media type of the annotation.

Change Bars

Authors can indicate a part of a document has been changed using the CHANGED tag. This may appear anywhere that normal text is allowed (as designated by the entity reference `%text;` in the DTD). This tag signals the beginning or end of changes, which should be rendered by a vertical bar in the left margin. The tag can have one (but not both) of the following attributes:

id An identifier unique to the current document, which can also be used as a a destination for hypertext links. This signals the beginning of changes.

idref This must be an identifier matching the preceding changed element. It signals the end of changes. Note that you mustn't have both *id* and *idref* together.

Miscellaneous Tags

The remaining tags must appear at the start of the node like TITLE and LINK. They describe properties which apply to the node as a whole.

The HTML tag.

This is intended to provide short informal classifications for use in cataloging documents held by HTTP servers. The *role* attribute identifies the purpose of the node, for example `<html role="home page">`. Another common role is "toc" for table of contents. See previous discussion of link attributes.

The ISINDEX tag

This specifies that the URL designated with the *href* attribute is searchable (defaults to this document's URL). Browsers should allow users to enter a search string of one or more keywords. When the Return key is pressed the search string is appended to the designated URL, after a "?" character and sent to the server specified by the URL. Certain characters should be escaped as specified by the standard URL syntax, for example, the space character is mapped to "+". The newer HTTP protocol offers an alternative means for specifying that documents are searchable. In this case, the search string is sent as part of an RFC 822 style header. See [Berners-Lee 93c] for details.

The NEXTID tag

This is used by browsers that automatically generate identifiers for anchor points. It specifies the next identifier to use, to avoid confusionwith old (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.

The BASE tag

The *href* attribute gives the full URL of the document, and is added by the browser when the user makes a local copy. Keeping the original URL in a local copy is essential when subsequently viewing the copy as it allows relative URLs in the document to be resolved to their original references.

Note that one motivation for using relative URLs is to allow a group of documents to be copied without the need to alter any links between them. In this case, the BASE tag is inappropriate, since it would cause links to be intepreted as being to the original documents rather than their copies.

The HEAD and BODY tags

The HEAD tag can be used to delimit properties which apply to the document as a whole, and if used, must be present at the start of the document, followed by the BODY tag which then delimits the rest of the document.

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, July 1993.

Email: `dsr@hplb.hpl.hp.com`, *Phone*: `+44 272 228046`

Appendix I - The HTML+ DTD

```
<!DOCTYPE HTMLPLUS [

<!-- DTD for HTML+ It assumes the default <!SGML> declaration

Markup minimisation should be avoided with the exception of </>
for the endtag. 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 % text "#PCDATA|A|IMG|EM|EMBED|INPUT|SP|BR|CHANGED">

<!ENTITY % paras "P|PRE|FIG">

<!ENTITY % lists "UL|OL|DL">

<!ENTITY % misc "TBL|FORM|PANEL|GROUP">

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

<!ENTITY % table "%text;|P|%heading;|%lists;">

<!ENTITY % main "%heading;|%misc;|%lists;|%paras;|%text;">

<!ENTITY % setup "(TITLE? & HTML? &ISINDEX? & NEXTID? & LINK* & BASE?)">

<!--

<!ELEMENT tagname - - CONTENT> elements needing closing tags

<!ELEMENT tagname - O CONTENT> elements without closing tags

<!ELEMENT tagname - O EMPTY> elements without content or closing tags

   The content definition is:

      a)    an entity definition as defined above

      b)    a tagname

      c)    (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
-->
<!ELEMENT HTMLPLUS O O ((HEAD, BODY) | ((%setup;), (%main;)*))>
<!ELEMENT HEAD - - (%setup;)>
<!ELEMENT BODY - - (%main;)*>
<!-- Document title -->
<!ELEMENT TITLE - - (#PCDATA | EM)+>
<!ATTLIST TITLE
      id       ID     #IMPLIED -- link destination --
      index    CDATA  #IMPLIED -- entries for index compilation -->
<!-- Document role for cataloging documents held by servers -->
<!ELEMENT HTML - O (EMPTY)>
<!ATTLIST HTML role CDATA #IMPLIED -- home page, index, ... -->
<!-- Floating panel which can be moved around relative to the normal text
flow. Often rendered with a different background and possibly framed. The
panel can be anchored to a named point in the document as specified by
the AT attribute. The panel may be placed at that point or after, but not
before.
-->
<!ELEMENT PANEL - - (TITLE?, (%main;)*)>
<!ATTLIST PANEL
      id       ID     #IMPLIED -- defines link destination --
      at       IDREF  #IMPLIED -- anchor point --
      index    CDATA  #IMPLIED -- entries for index compilation -->
<!-- Document headers -->
<!ELEMENT (%heading;) - - (#PCDATA | EM)+>
<!ATTLIST (%heading;)
      id       ID     #IMPLIED -- defines link destination --
      index    CDATA  #IMPLIED -- entries for index compilation -->
<!-- logical emphasis with optional style hints -->
<!ELEMENT EM - - (%text;)*>
<!ATTLIST EM
   role   CDATA  #IMPLIED -- semantic category e.g. CITE --
   b      (b)    #IMPLIED -- render in bold font --
   i      (i)    #IMPLIED -- render in italic font --
   u      (u)    #IMPLIED -- underline text --
   tt     (tt)   #IMPLIED -- render in typewriter font --
   tr     (tr)   #IMPLIED -- render in serif (Times Roman) font --
   hv     (hv)   #IMPLIED -- render in sans serif (Helvetica) font --
   sup    (sup)  #IMPLIED -- superscript --
   sub    (sub)  #IMPLIED -- subscript --
   index  CDATA  #IMPLIED -- entries for index compilation -->
<!-- Paragraphs with different roles and optional style hints -->
<!ELEMENT P - O (%text;)+>
<!ATTLIST P
   id       ID           #IMPLIED -- link destination --
   role     CDATA        #IMPLIED -- semantic role --
   align    CDATA        #IMPLIED -- left, center or right --
   indent   (indent)     #IMPLIED -- indented margins --
   index    CDATA        #IMPLIED -- entries for index compilation -->
<!ELEMENT BR - O EMPTY -- line break -->
```

```
<!ELEMENT SP - O EMPTY -- unbreakable space -->

<!-- Preformatted text with fixed pitch font, respecting original spacing
and newlines. Authors can also request proportional fonts. Further
control is possible with EM. -->

<!ELEMENT PRE - - (%text;)+>

<!ATTLIST PRE
   id       ID       #IMPLIED -- link destination --
   style    CDATA    #IMPLIED -- various styles --
   tr       (tr)     #IMPLIED -- serif (Times Roman) font --
   hv       (hv)     #IMPLIED -- sans serif (Helvetica) font --
   width    NUMBER   #IMPLIED -- e.g. 40, 80, 132 --
   index    CDATA    #IMPLIED -- entries for index compilation -->

<!-- Lists which can be nested -->

<!ELEMENT OL - - (LI | UL | OL)+ -- ordered list -->

<!ATTLIST OL
   id       ID         #IMPLIED
   compact  (compact)  #IMPLIED
   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 --
   narrow   (narrow)   #IMPLIED     -- narrow perhaps multi columns --
   index    CDATA      #IMPLIED -- entries for index compilation -->

<!-- List items for UL and OL lists -->

<!ELEMENT LI - O (P|%text;)+>

<!ATTLIST LI
   id       ID         #IMPLIED
   src      %URL;      #IMPLIED -- icon for use in place of bullet --
   index    CDATA      #IMPLIED -- entries for index compilation -->

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

<!ELEMENT DL - - (DT,DD)+ -- DT and DD *MUST* be paired -- > <!ATTLIST DL
   id       ID         #IMPLIED
   compact  (compact)  #IMPLIED
   index    CDATA      #IMPLIED -- entries for index compilation -->

<!ELEMENT DT - O (%text;)+ -- term text -- >

<!ELEMENT DD - O (P|QUOTE|UL|OL|%text;)+ -- definition text -- >

<!ATTLIST (DT|DD)
   id        ID        #IMPLIED
   index     CDATA     #IMPLIED -- entries for index compilation -->

<!-- Tables with titles and column headers, e.g.

   <tbl border>
      <tt> An Example of a Table
      <th> <th s="2"> average <th> other <tr>
      <th> <th> height <th> weight <th> category <tr>
      <td> males <td> 1.9 <td> .003 <td> yyy <tr>
      <td> females <td> 1.7 <td> .002 <td> xxx
   </tbl>

-->

<!ELEMENT TBL - - (TT?, (TH|TD|TR|TB)*) -- mixed headers and data -->

<!ATTLIST TBL
   id       ID         #IMPLIED
   compact  (compact)  #IMPLIED     -- if present use compact style --
   border   (border)   #IMPLIED     -- if present draw borders --
   index    CDATA      #IMPLIED      -- entries for index compilation -->

<!ELEMENT TT - O (%text;)+ -- table title -->
```

```
<!ATTLIST TT top (top) #IMPLIED -- place title above table -->

<!ELEMENT TH - O (%table;)* -- a header cell -->

<!ATTLIST TH
   colspan  NUMBER     1         -- columns spanned --
   rowspan  NUMBER     1         --. rows spanned --
   align    CDATA    #IMPLIED    -- left, center or right -->

<!ELEMENT TD - O (%table;)* -- a data cell -->

<!ATTLIST TD
   colspan  NUMBER     1         -- columns spanned --
   rowspan  NUMBER     1         --. rows spanned --
   align    CDATA    #IMPLIED    -- left, center or right -->

<!ELEMENT TR - O EMPTY -- row separator -->

<!ELEMENT TB - O EMPTY -- vertical break of 1/2 line spacing -->


<!-- Forms composed from input fields and selection menus

These elements define fields which users can type into or select with
mouse clicks. The browser should manage the input focus e.g. with the
tab/shift tab keys and mouse clicks.

The enter/return key is then taken to mean the use has filled in the form
and wants the apppropriate action taken:
   -  send as query/update to WWW server
   -  email/fax to designated person

The action is specified as a URL, e.g. "mailto:dsr@hplb.hpl.hp.com You
can specify additional mail headers with the MH tag:

<MH>Subject: Please add me to tennis tournament</MH>

Each FORM should include one or more INPUT elements which can be layed
out with normal and preformatted text, lists and tables.
 -->

<!ELEMENT FORM - - (MH, (%main;)*)>

<!ATTLIST FORM
   id       ID         #IMPLIED
   action   %URL;      #IMPLIED
   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 -->

<!-- INPUT elements should be defined within a FORM element.

Users can alter the value of the INPUT element by typing or clicking with
the mouse. Use radio buttons for selecting one attribute value from a set
of alternatives. In this case there will be several INPUT elements with
the same name. Attributes which can take multiple values at the same time
should be defined with checkboxes: define each allowed value in a
separate INPUT element but with the same attribute name. For checkboxes
and radio buttons, the value doesn't change, instead the state of the
button shown by the presence or absence of the checked attribute in each
element.

The size attribute specifies the size of the input field as appropriate
to each type. For text this gives the width in characters and height in
lines (separated by an "x"). For numbers this gives the maximum
precision.

-->
```

```
<!ELEMENT INPUT - O EMPTY>

<!ATTLIST INPUT
   name    CDATA     #IMPLIED -- attribute name (may not be unique) --
   type    CDATA     #IMPLIED --TEXT,URL,INT,FLOAT,DATE,CHECKBOX,RADIO--
   size    CDATA     #IMPLIED -- e.g."32x4" for multiline text --
   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 -->

<!-- Embedded Data

You can embed information in a foreign format into the HTML+ document.
This is very convenient for mathematical equations and simple drawings.
Images and complex drawings are better specified as linked documents
using the FIG or IMG elements.

Arbitrary 8 bit data is allowed but any occurrences of the following
chars must be escaped as shown:

      "&"     by      "&amp;"
      "<"     by      "&lt;"
      ">"     by      "&gt;"

The browser can pipe such data thru filters to generate the corresponding
pixmap The data format is specified as a MIME content type, e.g.
"text/eqn"
-->

<!ELEMENT EMBED - - (RCDATA)>

<!ATTLIST EMBED
   id       ID         #IMPLIED
   type     CDATA      #IMPLIED  -- mime content type --
   index    CDATA      #IMPLIED -- entries for index compilation -->

<!-- Figures

The image/drawing is specified by a URL or as embedded data for simple
drawings. The element's text serves as the caption. Use the emphasis with
style = "credits" to record photo credits etc.

-->

<!ELEMENT FIG - - (EMBED?, FIGD?, (FIGA|FIGT)*, (%text;)*)>

<!ATTLIST FIG
   id       ID         #IMPLIED
   align    CDATA      #IMPLIED -- position: left, right or center --
   cap      CDATA      #IMPLIED -- caption at left, right, top, bottom --
   noflow   (noflow)   #IMPLIED -- disables text flow --
   ismap    (ismap)    #IMPLIED -- server can handle mouse clicks/drags --
   src      %URL;      #IMPLIED -- link to image data --
   index    CDATA      #IMPLIED -- entries for index compilation -->

<!ELEMENT FIGD - - (%table;) -- figure description -->

<!-- Figure anchors designate polygonal areas on the figure which can be
clicked with the mouse. The default area is the whole of the figure. This
mechanism interprets mouse clicks locally, and browsers can choose to
highlight the designated area (or change the mouse sprite) when the mouse
is moved over the area.

Note that polygons may be non-convex or even intersect themselves,
thereby complicating the definition of what is enclosed by the polygon.
Holes are excluded.
-->

<!ELEMENT FIGA - O EMPTY>
```

```
<!ATTLIST FIGA
   href   %URL;    #REQUIRED -- link to traverse when clicked --
   area   NUMBERS  #IMPLIED -- x1,y1,x2,y2,x3,y3,... -->

<!-- FIGT Text on top of an figure background, or in a colored background
box which sits arbitrarily on top of an figure background. The text can
include headers, lists and tables etc. The width attribute allows you to
limit the width of the text box. The height is then determined
automatically by the browser.

FIGT can also be used to position a graphic on top of a picture using an
IMG element within FIGT. In this case the chromakey attribute may allow
parts of the underlying image to show through.

You can make the whole of the box into a hypertext link. This will act as
if it is underneath any hypertext links specified by the overlay markup
itself.
-->

<!ELEMENT FIGT - - (%main;)>

<!ATTLIST FIGT
   at     NUMBERS #IMPLIED -- upper left origin for text --
   width  NUMBER  #IMPLIED -- given as fraction of picture --
   framed (framed)#IMPLIED -- framed with coloured background --
   href   %URL;   #IMPLIED -- link to traverse when clicked -->

<!-- inline icons/small graphics
The align attribute defines whether the top middle or bottom of the
graphic and current text line should be aligned vertically

The SEETHRU attribute is intended as a chromakey to allow a given colour
to be designated as "transparent". Pixels with this value should not be
painted. The exact format of this attribute's value has yet to be
defined.


Use the FIG tag for captioned figures with active areas etc.
-->

<!ELEMENT IMG - O EMPTY>

<!ATTLIST IMG
   src     %URL;   #REQUIRED -- where to get image data --
   align   CDATA   #IMPLIED  -- top, middle or bottom --
   seethru CDATA   #IMPLIED  -- for transparency --
   ismap   (ismap) #IMPLIED  -- send mouse clicks/drags to server -->

<!-- Hierarchical groups for books, chapters, sections etc. -->

<!ELEMENT GROUP - - ((TITLE|LINK*), (%main;)*)>

<!ATTLIST GROUP
   id    ID       #IMPLIED
   role  CDATA    #IMPLIED -- book, chapter, section etc. --
   inset (inset)  #IMPLIED -- rendering hint: indent margins -->

<!-- change bars defined by a matched pair of CHANGED elements:

      <changed id=z34> changed text <changed idref=z34>

This tag can't act as a container, since changes don't respect
the nesting implied by paragraphs, headers, lists etc.
-->

<!ELEMENT CHANGED - O EMPTY>

<!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    ID       #IMPLIED -- backwards compatibility --
   href    %URL;    #IMPLIED -- destination node --
   role    CDATA    #IMPLIED -- role of link, e.g. "partof" --
   effect  CDATA    #IMPLIED -- replace/new/overlay --
   print   CDATA    #IMPLIED -- reference/footnote/section --
   title   CDATA    #IMPLIED -- when otherwise unavailable --
   type    CDATA    #IMPLIED -- for presentation cues --
   size    NAMES    #IMPLIED -- for progress cues -->

<!-- Other kinds of relationships between documents -->

<!ELEMENT LINK - O EMPTY>

<!ATTLIST LINK
   href    %URL;    #IMPLIED -- destination node --
   role    CDATA    #IMPLIED -- role played, e.g. "toc" -->

<!-- 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. -->

<!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 -->

<!-- dash entities -->

<!ENTITY endash "--"  -- En dash -->
<!ENTITY emdash "---" -- Em dash -->

<!-- The END -->

]>
```

Appendix II - Entity Definitions

ISO Latin 1 character entities in HTML+ derived from "ISO 8879:1986//ENTITIES Added Latin 1//EN". The corresponding 8-bit character codes are given in the DTD.

Æ	capital AE diphthong (ligature)
Á	capital A, acute accent
Â	capital A, circumflex accent
À	capital A, grave accent
Å	capital A, ring
Ã	capital A, tilde
Ä	capital A, dieresis or umlaut mark
Ç	capital C, cedilla
Ð	capital Eth, Icelandic
É	capital E, acute accent
Ê	capital E, circumflex accent
È	capital E, grave accent
Ë	capital E, dieresis or umlaut mark
Í	capital I, acute accent
Î	capital I, circumflex accent
Ì	capital I, grave accent
Ï	capital I, dieresis or umlaut mark
Ñ	capital N, tilde
Ó	capital O, acute accent
Ô	capital O, circumflex accent
Ò	capital O, grave accent
Ø	capital O, slash
Õ	capital O, tilde
Ö	capital O, dieresis or umlaut mark
Þ	capital THORN, Icelandic
Ú	capital U, acute accent
Û	capital U, circumflex accent
Ù	capital U, grave accent
Ü	capital U, dieresis or umlaut mark
Ý	capital Y, acute accent
á	small a, acute accent
â	small a, circumflex accent
æ	small ae diphthong (ligature)
à	small a, grave accent
å	small a, ring
ã	small a, tilde
ä	small a, dieresis or umlaut mark
ç	small c, cedilla
é	small e, acute accent
ê	small e, circumflex accent
è	small e, grave accent
ð	small eth, Icelandic
ë	small e, dieresis or umlaut mark

í	small i, acute accent
î	small i, circumflex accent
ì	small i, grave accent
ï	small i, dieresis or umlaut mark
ñ	small n, tilde
ó	small o, acute accent
ô	small o, circumflex accent
ò	small o, grave accent
ø	small o, slash
õ	small o, tilde
ö	small o, dieresis or umlaut mark
ß	small sharp s, German (sz ligature)
þ	small thorn, Icelandic
ú	small u, acute accent
û	small u, circumflex accent
ù	small u, grave accent
ü	small u, dieresis or umlaut mark
ý	small y, acute accent
ÿ	small y, dieresis or umlaut mark

In addition, there are two entity definitions for horizontal dashes longer than the "-" character.

&endash;	En sized horizontal dash (--)
&emdash;	Em sized horizontal dash (---)

Appendex III - Compatibility with HTML

HTML documents can be easily converted into the HTML+ format, and only a few changes are needed. Most documents won't need any changes at all. HTML+ browsers should be able to view HTML documents with very little effort. Older browsers will be able to view HTML+ documents which don't contain, tables or forms.

Lists

<menu>	becomes	<ul compact>
<dir>	becomes	<ul narrow>

Emphasis

HTML+ replaces the various tags used by HTML with a single tag. It may be worth changing the name for the emphasis tag in HTML+ from EM to EM, to gain compatibility with this common form. However, using EM might be confused with the typographical term *em* as in em dash (you also get en dash). EM has the merit of being unambigous. **I would like to get peoples views on this.**

<em>	becomes	<em>
<tt>	becomes	<em tt>
<strong>	becomes	<em b>
<b>	becomes	<em b>
<i>	becomes	<em i>
<u>	becomes	<em u>
<code>	becomes	<em role="code">
<samp>	becomes	<em role="samp">
<kbd>	becomes	<em role="kbd">
<var>	becomes	<em role="var">
<dfn>	becomes	<em role="dfn">
<cite>	becomes	<em role="cite">

Miscellaneous

Some tags which are deprecated in HTML are now obsolete, and should be mapped to preformatted text:

<plaintext>	becomes	<pre>
<xmp>	becomes	<pre>
<listing>	becomes	<pre>

The following two tags have been absorbed into the standard mechanism for paragraphs:

<address>	becomes	<p role="byline" align="right">
<blockquote>	becomes	<p role="quote">

References

This is missing the appropriate references to work on the syntax and name service for URNs. The HTTP definition needs updating to cover the encoding of form data (and *ismap* ?).

[Berners-Lee 93a] "*Hypertext Markup Language (HTML)*", Tim Berners-Lee, March 1993. URL=`ftp://info.cern.ch/pub/www/doc/http-spec.ps`

[Berners-Lee 93b] "*Uniform Resource Locators*", Tim Berners-Lee, January 1992. URL=`ftp://info.cern.ch/pub/ietf/url4.ps`

[Berners-Lee 93c] "*Protocol for the Retrieval and Manipulation of Textual and Hypermedia Information*", Tim Berners-Lee, 1993. URL=`ftp://info.cern.ch/pub/www/doc/html-spec.ps`

[Raisch 93] "*Style sheets for HTML*", Robert Raisch, June 1993, O'Reilly & Associates email: `raisch.ora.com`

[Kimber 93] Article in comp.text.sgml newsgroup, 24th May 1993 by Elliot Kimber (drmacro@vnet.almaden.ibm.com), URL=`news:19930524.152345.29@almaden.ibm.com`