A Review of the HTML+ Document Format

David Raggett, Hewlett Packard Laboratories

(email: dsr@hplb.hpl.hp.com)

HTML+ is a set of modular extensions to the hypertext markup language (HTML), which is in widespread use in the World Wide Web. The use of SGML to specify HTML+ allows authors to create documents in a variety of ways: with text editors, SGML authoring tools and filters from common word processing formats like Framemaker, Microsoft Word and LaTeX. The paper finishes by looking at extensions under current consideration and encouraging wider debate on what is needed to fuel the next stage in the development of the Web.

1 Introduction

In 1450 Johannes Gutenberg is credited with bringing together two ideas: the use of metal dies to make moveable type and a press for obtaining sharp impressions on sheets of paper. The invention of printing at the dawn of the age of great discoveries accelerated economic, social and ideological changes that were to usher in the modern world.

Traditional media like books, newspapers, radio, cinema and television are published by the few for the many, while telephones are limited to one to one conversations. The World Wide Web is a radically new media, bringing together hypertext, logical markup and global networking. The Web makes it very easy for anyone to publish information to people located anywhere in the world. The Web is growing at a very rapid rate - doubling in traffic every two to three months. At the time of writing some 400 Gigabytes was being carried per month.

There is now a great body of experience in laying out the printed word. That experience brings with it a fixation with the printed page. The development of computers has perpetuated that fixation: Word processing software has been seen principally as a means of preparing printed documents. Global networks now make it easy to distribute information electronically and the time has come to shake off the limitations of the printed page.

Today, many of the formats for electronic distribution of information embody the restrictions of the printed page - documents are broken into a sequence of fixed sized pages and pages are described in terms of marks to be made at specified positions, usually involving named fonts. This has prevented true interchangeability of information. Many of you will have seen this with Postscript - for some reason or other a problem often occurs when trying to print out that paper you just retrieved over the net!

The Standard Generalized Markup Language (SGML) was invented as a solution to these problems. By focussing on the logical elements in a document, the recipient of information is freed from the often inappropriate choices of the originator. Users can resize their windows to make optimal use of their screens and printing software can layout documents to match local paper sizes. Other benefits from using SGML include the ability to protect the investment of information providers in the face of short lived proprietary formats, and the availability of tools for authoring and format conversion. SGML is a metaformat that allows one to specify a wide range of document formats. The hypertext markup language HTML was developed as a simple non-proprietary delivery format for global hypertext. HTML+ is a set of modular extensions to HTML and has been developed in response to a growing understanding of the needs of information providers. These extensions include text flow around floating figures, fill-out forms, tables and mathematical equations.

Browsers are programs for viewing HTML documents and are now available for most platforms. The best known are the Mosaic family of browsers from the U.S. National Center for Supercomputing Applications, with browsers for X11, Windows, and the Macintosh. Non-graphical browsers are available for VT100 terminals etc, for example Lynx and Emacs-W3. The appearence of a document will vary from one browser to the next according to the capabilities of each system and the user’s preferences.

Currently, most documents are created with conventional text editors. The effort needed to type the markup elements can be reduced by using post-processing tools to ensure that the resulting document complies with the HTML document type definition (DTD). Authors can also use common word processing packages and apply filters to convert to HTML. Filters are available for FrameMaker, LaTeX and Microsoft Word. Other people are using standard SGML tools for document authoring and conversion from proprietary SGML based document formats.

2 An Overview of HTML

HTML includes markup elements for:

- Headers
- Paragraphs
- Various types of character highlighting
- Character-like in-line images
- Hypertext links
- Lists
- Preformatted text
- Simple search facility

Here is a simple example of an HTML document:

```
<title>A sample HTML document</title>
<h1>An example of Structure</h1>
Here’s a typical <i>paragraph</i><p>
<ul>
 <li>Item one has an <a name="anchor">anchor</a>
 <li>The next item has a <a href="link.html">hypertext link</a>.
</ul>
```

The `TITLE` element is used to name the document as a whole. There are six levels of headers of decreasing importance ranging from `H1` to `H6`. The `P` element indicates a paragraph break. HTML supports four styles of character highlighting: fixed-width teletype font `TT`, bold face `B`, italic `I` and underlined `U`. Browsers can choose to render them in alternative ways or even to ignore them altogether. There are in addition several logical styles.

Hypertext links are represented with the anchor tag, and restricted to unidirectional binary links. The text between the start tag `<a>` and the end tag `</a>` is used as the caption. Links are usually made to a document as a whole but can also be made to anchors within a document. The example shows an anchor being used with the `NAME` attribute to declare a potential destination for such a link. The `HREF` attribute is used to identify the linked document via a universal resource locator (URL). This is a naming scheme for referencing objects located anywhere in the network. URLs start with a prefix that describe the access method or protocol for retrieving the referenced object, and is usually followed by an Internet host name and path. Relative references provide a convenient method of abbreviating URLs and are expanded in the context of the full URL for the current document. A reference to the specification of URLs can be found in section 5 at the end of this document.

There are several kinds of lists: definition lists `DL`, ordered (numbered) lists `OL`, bulleted lists `UL` and plain lists (`MENU` and `DIR`). Lists cannot be nested, although in practice, most browsers relax this restriction. Preformatted text is represented with the `PRE` tag, and can include character highlighting and hypertext links. It was introduced to make it easy to import the output of existing computer programs, e.g. Unix manual pages. HTML includes a simple search facility (the `<ISINDEX>` tag) where users can type in a string of characters and send them to a server for interpretation.

The `HEAD` and `BODY` elements can be used to separate the front matter from the document `body`. The document’s head is used for information pertaining to the document as a whole, such as the title, whether it supports searches: `ISINDEX`, and relationships with other documents, expressed with the `LINK` element.

3 A Comparison between HTML+ and HTML

HTML+ adds new features like figures, tables and forms. It also generalizes the structures present in HTML to reflect practical experience with writing browsers and a desire to make it easier to convert between HTML+ and other formats. These changes are reviewed below before presenting major new features. This part of the paper assumes some prior familiarization with the HTML document format.

Most browsers now support nested lists. HTML+ formalizes this, allowing arbitrary nesting of the various kinds of lists. Lists items can now include the horizontal rule element `HR`. A further extension under consideration is to allow preformatted text in list items using the `PRE` element. The `DIR` and `MENU` elements have been replaced by a more powerful version of the `UL` element for unordered lists. The `PLAIN` attribute suppresses bullets, while the `WRAP` attribute specifies the wrap style (vertical or horizontal) for multicolumn lists. Bullets can also be replaced by icons, using the `SRC` attribute to specify a URL for an image.

A number of elements have been made into containers. These include paragraphs `P`, list items `LI`, `DT` and `DD`. This change has little effect on authors, but helps with document conversion, and brings HTML into accord with other SGML formats. It also makes it practical to associate attributes with the contained text, e.g. character sets and paragraph alignment. Most elements now accept an `ID` attribute, which provides a name unique within that document, and can be used as the destination of hypertext links. This is a common technique in SGML documents. The new `CHARSET` attribute can be used to specify the character set on an element by element basis, e.g. ISO-2022-JP for japanese.

Strike through `S`, is a new style for character highlighting and particularly useful for legal documents. Highlighting elements can now be nested, e.g. for bold-italic text. Other innovations include subscripts `SUB` and superscripts `SUP`, in-line quotes `Q`, and support for change bars `CHANGED`. The element for longer quotes has been abbreviated to `QUOTE`. A forced line break is now possible with the `BR` element. LaTeX style tab stops can be defined with the `TAB` element. The range of character entities has been considerably expanded to cover a wide range of symbols, for example: *&nbsp;* for a non-breaking space and *&copy;* for a copyright sign. Entity names are also defined for standard icons for use as annotations in directory listings, e.g. *&folder;* and other applications.

Character-like images are supported using the existing `IMG` element. The `ALT` attribute can be used to give a textual description for non-graphical browsers. The vertical position of the image relative to the baseline can be specified as before with the `ALIGN` attribute (*top*, *middle* or *bottom*), but greater precision is possible with the new `BASELINE` attribute which adjusts the image so that the baseline occurs at the specified number of pixels above the bottom of the image. The `ISMAP` attribute allows users to pass clicks on the image through to the server as an offset from the top-left corner of the image. This is useful for graphical menus or active maps.

3.1 Some new elements

The `FOOTNOTE` element is intended for additional information and can be implemented as an internal jump to the end of the document or as a pop-up panel. The `MARGIN` element is useful for attracting people’s attention with a snappy phrase placed in the margin or rendered as a separate paragraph in a largish font, perhaps marked off by horizontal rules. The `ABSTRACT` element is used as its name suggests, typically being placed just after a major heading. The `NOTE` element is used for admonishments. The `ROLE` attribute is one of *simple*, *tip*, *note*, *warning* or *error*. This element is usually rendered with an icon in the left margin according to the `ROLE` attribute with the text inset accordingly. The default icons are typically: none for *simple*, an upright hand with an outstretched finger for *tip*, an information sign for *note*, a traffic warning sign for *warning* and a stop sign for *error*. These icons can be overridden by using the `SRC` attribute to specify a URL for the image.

3.2 Figures, text flow and local processing of events

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.

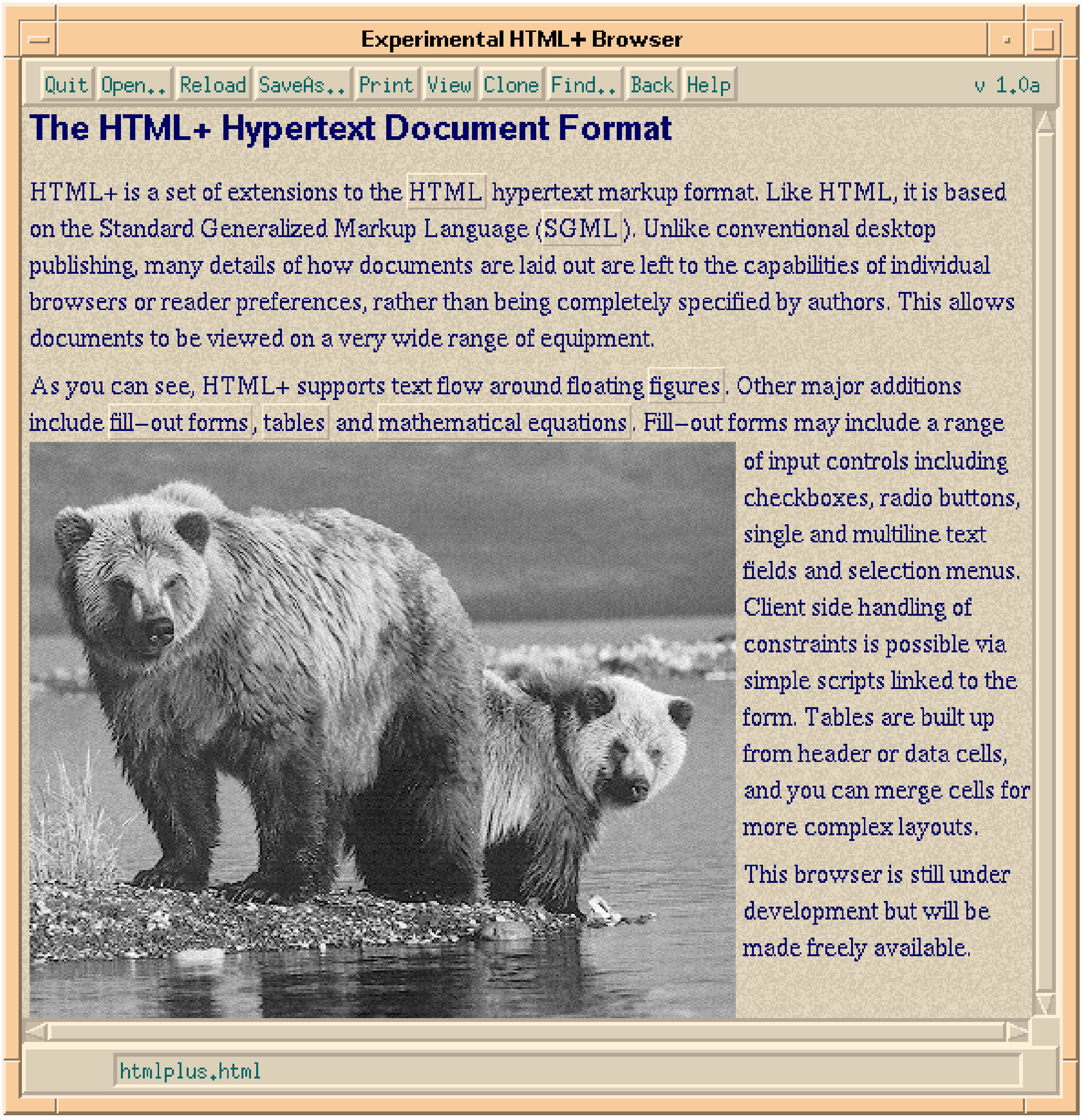
This example is produced by the following piece of HTML+

```
<p><fig align=left src=”people/tbl.gif”>Photo of Tim Berners-Lee</fig>
Before coming to CERN, Tim worked on, among other things, ...
```

The `FIG` element can be placed anywhere in a paragraph, and uses the `SRC` attribute to specify an image as a URL. If it is placed after the beginning, the figure will float until the next line break. Text is then flowed around the left or right of the figure provided there is sufficient room. The whitespace around the figure is adjusted to compensate for the figure being a non-integral number of text lines in height. You can control the horizontal positioning of the figure with the `ALIGN` attribute, e.g. *`left`*, *`center`* or *`right`*. Text flow is disabled for centered figures. The `CAPTION` element may be used to specify a caption for the figure, and can be placed above or below the figure according to the `ALIGN` attribute (*`top`* or *`bottom`*). The text between the start and end tags for the figure is used to describe the image content and is only shown on non-graphical displays:

```
<fig align=left src="people/tbl.gif">
<caption>Tim Berners-Lee</caption>This photograph was taken
at the World Wide Web Wizards Workshop in July 1993.</fig>
```

The following illustration shows another example of textflow around figures, taken from a snapshot of an HTML+ browser being developed by the author:



The HTML+ Hypertext Document Format

HTML+ is a set of extensions to the HTML hypertext markup format. Like HTML, it is based on the Standard Generalized Markup Language (SGML). Unlike conventional desktop publishing, many details of how documents are laid out are left to the capabilities of individual browsers or reader preferences, rather than being completely specified by authors. This allows documents to be viewed on a very wide range of equipment.

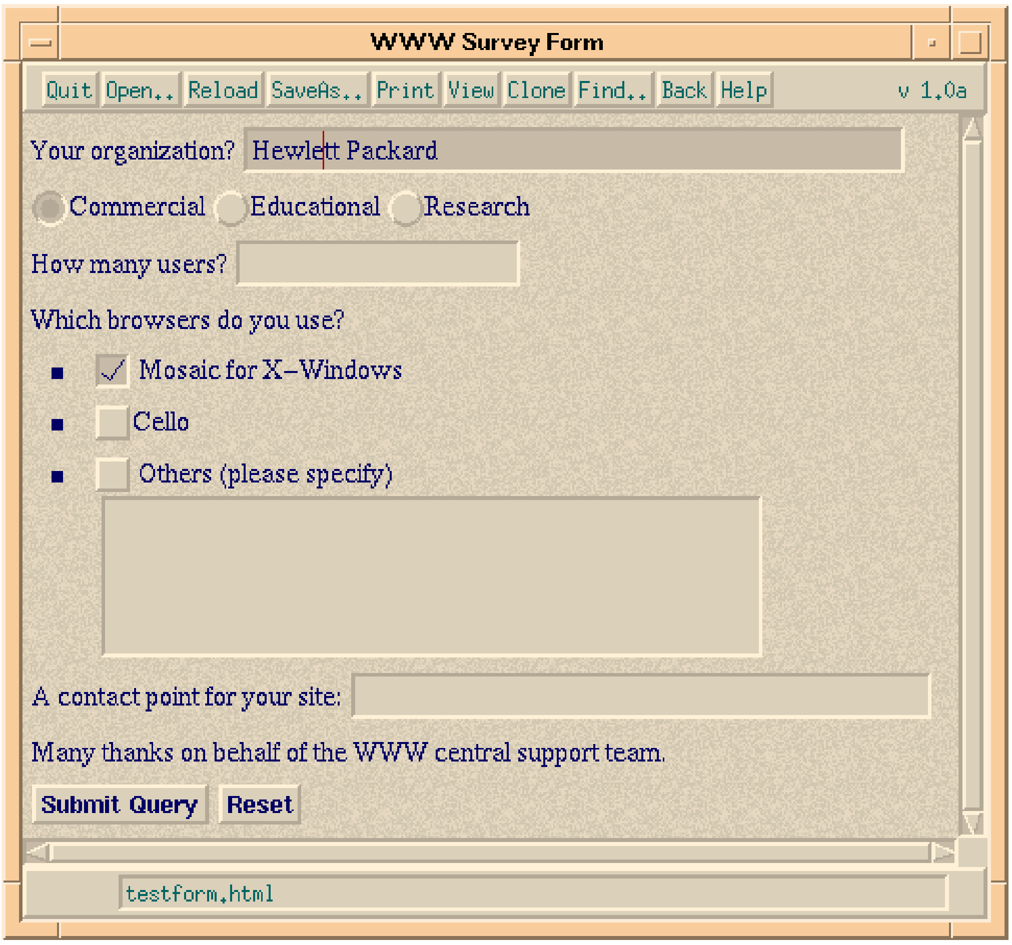
As you can see, HTML+ supports text flow around floating figures. Other major additions include fill-out forms, tables and mathematical equations. Fill-out forms may include a range of input controls including checkboxes, radio buttons, single and multiline text fields and selection menus. Client side handling of constraints is possible via simple scripts linked to the form. Tables are built up from header or data cells, and you can merge cells for more complex layouts.

This browser is still under development but will be made freely available.

Like `IMG`, you can use the `ISMAP` attribute to have mouse clicks on the figure passed to the server. A drawback of this approach is that you often have to wait an appreciable time to know if you successfully clicked on an active region or whether you just missed! The solution to this problem is to define shaped hypertext anchors in the figure itself. This is done by including anchors with the `A` element and using the `SHAPE` attribute to define the outline as a closed polygon. These anchors are placed in the figure description and hence are available to users with non-graphical displays (unlike the `ISMAP` mechanism). It is recommended that the mouse cursor actively change shape to indicate when it is currently over an active region of the figure.

3.3 Fill-out forms

HTML allowed users to enter a character string to search remote databases. HTML+ fill-out forms provides a much richer interface with a variety of field types:



Your organization? Hewlett Packard

[✓ Commercial] Educational Research

How many users?

Which browsers do you use?

- [✓] Mosaic for X-Windows
- [ ] Cello
- [ ] Others (please specify)

A contact point for your site:

Many thanks on behalf of the WWW central support team.

testform.html

This example shows single and multi-line text input fields, radio buttons, and checkboxes. The buttons at the lower left are used to submit the form to the server, or to reset the fields to their initial values. Other kinds of fields include password fields, option lists and hidden fields for state information. Scripts at the server simplify handling the form's contents and interfacing with databases or other software.

The above form was generated by the following piece of HTML+

```
<FORM ACTION="http://www.com/survey.html">
  <P>Your organization? <INPUT NAME="org" SIZE="48">
  <P><INPUT NAME="commerce"TYPE=radio VALUE="commerce">Commercial
  <INPUT NAME="commerce" TYPE=radio VALUE="education">Educational
  <INPUT NAME="commerce" TYPE=radio VALUE="research">Research
  <P>How many users? <INPUT NAME="users" TYPE=int>
  <P>Which browsers do you use?
  <UL>
    <LI><INPUT NAME="browsers" TYPE=checkbox VALUE="xmosaic">
```

```
      Mosaic for X-Windows
     <LI><INPUT NAME="browsers" TYPE=checkbox VALUE="cello">Cello
     <LI><INPUT NAME="browsers" TYPE=checkbox VALUE="others">
      Others (please specify)<BR>
      <TEXTAREA NAME="others" COLS=48 ROWS=4></TEXTAREA>
   </UL>
   <P>A contact point for your site: <INPUT NAME="contact" SIZE="42">
   <P>Many thanks on behalf of the WWW central support team.
   <P><INPUT TYPE=submit> <INPUT TYPE=reset>
 </FORM>
```

When the form is sent to the server, the field values are converted into a property list of names and values. As the example shows, several fields can share the same NAME. Radio buttons and checkboxes are only included in the property list when checked. Currently the property list is encoded as a sequence of *name=value* elements separated by the “&” character, and appended to the URL as a search string. The Hypertext Transfer Protocol (HTTP) is used to send the form to the server. HTTP uses the Multipurpose Internet Mail Extensions (MIME) for encapsulating requests and responses, and MIME multipart/related messages are expected to take over from the URL based notation, although the precise details are still being worked on. The ENCTYPE attribute in the FORM element is used to specify which encoding method to apply. Multipart/related messages have the advantage of declaring the content type for each message part, and will allow forms to use more efficient encodings such as JOT for pen-based input.[1]

Forms can include multiple submit buttons. The NAME attribute allows the server to determine which button was pressed. You can also use an image in place of a submit button via the SRC attribute. In this case, the location clicked on the image is sent to the server along with the contents of the other fields. New field types under consideration include: scribble fields which allow you to scribble with a "pen" on top of a picture, voice input, a tabular widget for data entry, and a field in which users can type a file name for passing a local file to the server.

Sometimes, you will want to create forms where clicking on one field effects the values shown by others. You may want to restrict the values input into a given field, or to place constraints on interdependent fields. Handling this remotely is too slow, and work is underway to allow forms to be associated with scripts that are executed by the browser in response to typing at the keyboard or clicking with the mouse. This work is focussing on defining a standard API (applications programming interface) that will be used by the browser to call the script interpreter. This will allow the scripting language to be specified separately. The API acts as a firewall around the script and provides a guarantee that hostile scripts can’t mess up client systems. The SCRIPT attribute in the FORM element specifies the script via a URL. This will enable browsers to use HTTP format negotiation to declare which languages they support.

3.4 Tables

Tables are specified using the TABLE element. This allows you to define a caption (above or below the table) and to differentiate header and data cells. Cells may contain, text, images, 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

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.

This is defined by the markup:

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

Each row is contained within a `TR` element, although the end tag may be omitted. 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. By default text is centered in each cell. Header text should be shown emphasized, 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	1.9	0.003	yyy
females	1.7	0.002	xxx

This table is defined by the markup:

```
<table border>
  <caption>A more complex table</caption>
  <tr><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 `<BR>` element has been used here to force a line break between `other` and `category`. Note that empty cells at the end of a row can be omitted as the next `<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`*. It can be used with both `TH` and `TD`.

Browsers use a prepass to size each table to match the current window size, although sometimes the table is just too wide. Text which is too long to fit on one line is wrapped within each cell in the same way that text normally wraps within the window.

3.5 Mathematical Equations

HTML+ provides a relatively simple way of representing equations, intended to cover the majority of users needs, rather than aiming for complete coverage. An experimental browser supporting the `MATH` element has been 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 8879-1986` 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>
```

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. Other math elements include `ARRAY` for matrices and `ROOT` for expressions involving roots.

3.6 Document declarations

It is good practice to start an HTML+ document with the `DOCTYPE` declaration, e.g.

```
<!DOCTYPE htmlplus [
  <!ENTITY % HTML.tables "INCLUDE">
  <!ENTITY % HTML.forms "INCLUDE">
]>
```

This preamble identifies the document type definition and informs the browser which extensions to HTML will be used in the document. The extensions are divided into the following categories:

- HTML.emph
- HTML.forms
- HTML.figures
- HTML.tables
- HTML.math
- HTML.obsolete

`HTML.emph` switches on the new character emphasis types such as strike-through and in-line quotes, together with footnotes and related elements. `HTML.obsolete` switches on obsoleted elements such as `DIR` and `MENU`.

The HTML+ DTD can be dynamically extended to add novel tags for use in the current document by redefining the special entity names: `cextra` for elements which occur in the middle of text, and `pextra` for block-like elements. To add a new kind of emphasis tag for proper names, you would write:

```
<!DOCTYPE htmlplus [
 <!ENTITY % cextra "|PROPNAME">
 <!ELEMENT PROPNAME - - CDATA>
 <RENDER tag="PROPNAME" style="I">
]>
```

The `RENDER` element specifies how the new element should be rendered in terms of a space separated list of existing tag names. These are limited to the character highlighting tags: `I` for italic, `B` for bold, `U` for underlined, `S` for strike-through and `TT` for a fixed-pitch font. The new element would be used as in:

```
Many people down the ages have wondered about the smile
on the face of the <propname>Sphinx</propname>.
```

If a browser doesn’t recognize a start or end tag it is ignored, and the contents processed as if the tags weren’t present. Likewise, unknown attributes are simply ignored. It is good practice, however, to formally declare all new elements as this will facilitate the use of SGML document processing tools.

3.7 The Document HEAD and BODY

The simplest possible HTML+ document looks like:

```
<!DOCTYPE HTMLPLUS SYSTEM "HTMLPLUS.DTD">
<HEAD>
<BODY>
```

The `HEAD` and `BODY` elements are used to mark off the document front matter from the document body. The `HEAD` contains all elements pertaining to the document as a whole. These include `TITLE`, `ISINDEX`, `BASE`, `META` and `LINK`. The `BASE` element is inserted by the browser when making a local copy of a document, and records the original URL. It is subsequently used to resolve relative URLs when following hypertext links from the local copy. Servers should read the document head to generate HTTP headers such as "`Expires:`". These are initialized from `META` elements such as:

```
<meta name="Expires" value="Tue, 04 Dec 1993 21:29:02 GMT">
```

Other likely names are "`Keywords`", "`Created`", "`Owner`" (a name) and "`Reply-To`" (an email address). This information is used for maintaing local caches of HTML documents.

3.8 Hypertext paths

Authors can create guided tours through the web. These are defined using HTML+ documents in which links specify a sequence of documents to present to the user. The `NODE` attribute in the anchor element is used to signify that a particular link should form part of the path, e.g.

```
<P>This is a sample path:
<UL>
  <LI><P>This is the <A HREF=node1.html" NODE>first node
  <LI><P>and this is the <A HREF=node2.html" NODE>second node
</UL>
```

You can compose paths by using the `PATH` attribute to indicate that a link references another path, e.g.

```
<A HREF="path.html" PATH>inserts the linked path into the current one</A>
```

Users can move along a path via next and previous buttons on the browser’s toolbar (or navigation menu). Additional buttons can be defined using the `LINK` element to specify links to the table of contents and author defined bookmarks. The specification allows for `LINK` elements to be dynamically added to documents via the HTTP protocol. This is used by the path mechanism to imply link elements for the active path, and previous/next nodes in that path. These richer navigation mechanisms are intended to reduce the chances that users will feel "lost in hyperspace" when browsing through complex material.

4 Where Next

Most browsers now support simple fill-out forms, following the pioneering work by NCSA on X Mosaic. Support for tables and figures is currently thin on the ground. Viola provides limited support for tables and NCSA are working on adding these features to Mosaic. The author’s X11 HTML+ browser should help to speed up this process. The Internet Draft for HTML+ lapsed at the beginning of May, and is being replaced, and is now available in hypertext form via the Web.

Information providers are interested in making their documents appear in a particular style which differentiates them from other information providers. Work is under way to see how HTML+ could support style information without limiting platform independence. Style hints could be expressed as part of the document head and cover aspects such as font families, text color and size, and the use of whitespace around elements. The use of images, and the opportunity to set the color and texture of the background offer further ways of creating a unique style. Another idea is to define a document specific toolbar using `LINK` elements. Custom images for links will be possible using the `SRC` attribute. In-line images are currently limited to the GIF format. A natural extension will be to support JPEG and even MPEG. The Web is also missing the capability to compose images from multiple sources. It would be desirable to be able to overlay line graphics on top of bitmapped images or say a small video inset on a static image. Much work remains to be done on how to compose and coordinate multimedia sources. The ISO HyTime standard is likely to be of relevance to this work.

The possibilities for extending the World Wide Web to include virtual reality are being covered in another paper in this conference. The author is also involved with work on adapting the World Wide Web to work on ATM networks, which offer dramatic improvements in speed over current network technologies, and rich opportunities for novel multimedia applications, for example integration with the telephone service for voice mail, and video-telephone calls. The Web looks like providing an excellent way for users to browse through multicast "radio" and "television" channels. HTML+ is expected to play an important role as the hypertext glue that bridges the gaps between such services, e.g. by providing a means for users to browse through what’s-on guides and follow links to particular channels or videos-on-demand.

The development of commercial services using the web will be greatly stimulated by global deployment of public key cryptography. The waiving of licence fees for freeware implementations of browsers and servers is expected to speed widespread adoption of these technique. The development of effective directory services is also going to play an increasingly important role. The deployment of commercial services will in turn provide the resources to develop improved browsers and document conversion tools, and feed through into the continuing evolution of hypermedia document formats.

5 Further Reading

“*HTML+ (Hypertext markup format)*”, Dave Raggett, November 1993.
URL=ftp://15.254.100.100/pub/draft-raggett-www-htmp-00.ps

“*HTML+ Document Type Definition*”, Dave Raggett, April 1994.
URL=ftp://15.254.100.100/pub/htmlplus.dtd.txt

“*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 HyTime Hypermedia/Time-based Document Structuring Language*”,
Steven R. Newcomb, Neill A. Kipp, and Victoria T. Newcomb.
Communications of the ACM/November ’91 / Vol.34 No.11 pp 67-83

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