INTERNET DRAFT Dave Raggett, W3C

Expires in six months email: <dsr@w3.org>

HyperText Markup Language Specification Version 3.0

<draft-ietf-html-specv3-00.txt>

Status of this Memo

be found at URL: http://www.cnri.reston.va.us/

Abstract

The HyperText Markup Language (HTML) is a simple markup language used to create hypertext documents that are portable from one platform to another. HTML documents are SGML documents with generic semantics that are appropriate for representing information from a wide range of applications. HTML markup can represent hypertext news, mail, documentation, and hypermedia; menus of options; database query results; simple structured documents with inlined graphics and hypertext views of existing bodies of information.

This specification defines the capabilities of HTML version 3.0 and provides additional capabilities over previous versions such as tables, text flow around figures and math. It is backwards compatible with HTML 2.0.

[Link to Table of Contents]--

Table of Contents

Introduction to HTML 3.0

HyperText Markup Language (HTML) is a simple markup system used to create hypertext documents that are portable from one platform to another. HTML documents are SGML documents with generic semantics that are appropriate for representing information from a wide range of applications. HTML markup can represent hypertext news, mail, documentation, and hypermedia; menus of options; database query results; simple structured documents with in-lined graphics; and hypertext views of existing bodies of information.

HTML has been in use by the World-Wide Web (WWW) global information initiative since 1990. The HTML 3.0 specification provides a number of new features, and is broadly backwards compatible with HTML 2.0. It is defined as an application of International Standard ISO ISO8879:1986 Standard Generalized Markup Language (SGML). This specificiation will be proposed as the Internet Media Type (RFC 1590) and MIME Content Type (RFC 1521) called "text/html; version=3.0".

How to participate in refining HTML 3.0

The process of refining HTML 3.0 into a formal standard will be carried out by the IETF HTML working group. The World Wide Web

Organization is continuing to develop a freeware testbed browser for HTML 3.0 ("Arena") to encourage people to try out the proposed features. The discussion list for HTML 3.0 is www-html with html-wg reserved for use by the IETF working group for detailed matters relating to the formal specification. The process for developing HTML 3.0 is open, and anyone who is interested and able to contribute to this effort is welcome to join in.

--Note: make mailing list names into hypertext links to their archives and add info on how to join these lists--

HTML 3.0 Overview

HTML 3.0 builds upon HTML 2.0 and provides full backwards compatibility. Tables have been one of the most requested features, with text flow around figures and math as runners up. Traditional SGML table models, e.g. the CALS table model, are really complex. The HTML 3.0 proposal for tables uses a lightweight style of markup suitable for rendering on a very wide range of output devices, including braille and speech synthesizers.

HTML 3.0 introduces a new element: FIG for inline figures. This provides for client-side handling of hotzones while cleanly catering for non-graphical browsers. Text can be flowed around figures and you can control when to break the flow to begin a new element.

Including support for equations and formulae in HTML 3.0 adds relatively little complexity to a browser. The proposed format is strongly influenced by TeX. Like tables, the format uses a lightweight style of markup - simple enough to type in by hand, although it will in most cases be easier to use a filter from a word processing format or a direct HTML 3.0 wysiwyg editor. The level of support is compatible with most word processing software, and avoids the drawbacks from having to convert math to inline images.

The Web has acted as a huge exercise in user testing, and we have been able to glean lots of information from the ways people abuse HTML in trying to get a particular effect; as well as from explicit demand for new features. HTML 3.0, as a result, includes support for customised lists; fine positioning control with entities like &emspace; horizontal tabs and horizontal alignment of headers and paragraph text.

Additional features include a static banner area for corporate logos, disclaimers and customized navigation/search controls. The LINK element can be used to provide standard toolbar/menu items for navigation, such as previous and next buttons. The NOTE element is used for admonishments such as notes, cautions or warnings, and also used for footnotes.

Forms have been extended to support graphical selection menus with client-side handling of events similar to FIG. Other new form field types include range controls, scribble on image, file upload and audio input fields. Client-side scripting of forms is envisaged with the script attribute of the FORM element. Forms and tables make for a powerful combination offering rich opportunities for laying out custom interfaces to remote information systems.

To counter the temptation to add yet more presentation features, HTML 3.0 is designed (but doesn't require) to be used together with style sheets which give rich control over document rendering, and can take into account the user's preferences, the window size and other resource limitations, such as which fonts are actually available. This work will eventually lead to smart layout under the author's control, with rich magazine style layouts for full screen viewing, switching to simpler layouts when the window is shrunk.

The SGML Open consortium is promoting use of DSSSL Lite by James Clark. This is a simplified subset of DSSSL - the document style semantics specification language. DSSSL is a ISO standard for representing presentation semantics for SGML documents, but is much too complex in its entirety to be well suited to the World Wide Web. HÕkon Lie maintains a list of pointers to work on style sheets.

Transition Strategy from HTML 2.0

The use of the MIME content type: "text/html; version=3.0" is recommended to prevent existing HTML 2.0 user agents screwing up by attempting to show 3.0 documents. Tests have shown that the suggested content type will safely cause existing user agents to display the save to file dialog rather than incorrectly displaying the document as if it were HTML 2.0.

To make it easy for servers to distinguish 3.0 documents from 2.0 documents, it is suggested that 3.0 files are saved with the extension ".html3" (or ".ht3" for PCs). Servers can also exploit the accept headers in HTTP requests from HTML user agents, to distinguish whether each client can or cannot support HTML 3.0. This makes it practical for information providers to start providing HTML 3.0 versions of existing documents for newer user agents, without impacting older user agents. It is envisaged that programs will be made available for automatic down conversion of 3.0 to 2.0 documents. This conversion could be carried out in batch mode, or on the fly (with caching for greater efficiency).

Design Guidelines

The HTML 3.0 draft specification has been written to the following guidelines.

Lingua Franca for the Web

HTML is intended as a common medium for tying together information from widely different sources. A means to rise above the interoperability problems with existing document formats, and a means to provide a truly open interface to proprietary information systems.

Simplicity

The first version of HTML was designed to be extremely simple, both to author and to write browsers for. This has played a major role in the incredibly rapid growth of the World Wide Web. HTML 3.0 provides a clean superset of HTML 2.0 adding high value features such as tables, text flow around figures and math, while still remaining a simple document format. The pressures to adopt the complexities of traditional SGML applications has been resisted, for example the Department of Defense's CALS table model or the ISO 12083 math DTD.

Scaleability

As time goes by, people's expectations change, and more will be demanded of HTML. One manifestation of this is the pressure to add yet more tags. HTML 3.0 introduces a means for subclassing elements in an open-ended way. This can be used to distinguish the role of a paragraph element as being a couplet in a stansa, or a mathematical term as being a tensor. This ability to make fresh distinctions can be exploited to impart distinct rendering styles or to support richer search mechanisms, without further complicating the HTML document format itself. Scaleability is also achieved via URI based links for embedding information in other formats. Initially limited to a few image formats, inline support is expected to rapidly evolve to cover drawing formats, video, distributed virtual reality and a general means for embedding other applications.

Platform Independence

HTML is designed to allow rendering on a very wide range of devices, from clunky teletypes, to terminals, DOS, Windows, Macs and high end Workstations, as well as non-visual media such as speech and braille. In this, it allows users to exploit the legacy of older equipment as well as the latest and best of new machines. HTML 3.0 provides for improved support for non-graphical clients, allowing for rich markup in place of the figures shown on graphical clients. HTML can be rendered on a wide variety of screen sizes, using a scrolling or paged model. The fonts and presentation can be adjusted to suit the resources available in the host machine and the user's preferences.

Content --not-- Presentation Markup

Information providers are used to tight control over the final appearence of documents. The need for platform independence weighs against this, but there is still a strong pressure to find appropriate means for information providers to express their intentions. The experience with proprietary document formats has shown the dangers of mixing presentation markup with content (or structural) markup. It becomes difficult to apply different presentation styles. It becomes painful to incorporate material from different sources (with different presentation styles). It becomes difficult to be truly platform independent. As a result, HTML 3.0 is designed for use with linked style information that defines the intended presentation style for each element. Style sheets can be expressed in a platform independent fashion or used to provide more detailed control for particular classes of clients or output media.

Support for Cascaded Style Sheets

For the Web, it is valuable to allow for a cascading of style preferences. The client has certain built-in preferences; the publisher may require a particular house style, e.g. for brand distinction; the author may feel the need to override the house style for special cases; the end-user may feel strongly about certain things, e.g. large fonts for easier visibility or avoiding certain colors due to an inability to distinguish between them. HTML 3.0 supports style sheets via the use of the LINK element to reference a style sheet with a URI. Authors can place overrides in separate style sheets or include them in the document head within the STYLE element. The effectiveness of caching mechanisms for speeding up the retrieval of style sheets is enhanced by the separation of style information into generic commonly used style sheets, and overrides specific to this document.

Support for Non-Visual Media

HTML 3.0 is designed to cater for the needs of the visually impaired. Markup for inline figures includes support for rich descriptions, along with hypertext links that double up as defining geometric hotzones for graphical browsers, simplifying the author's job in catering for the different groups of users. Table markup includes provision for abbreviated row and column names for each cell, which are essential for conversion to speech or braille. Math markup treats formulae and equations as hierarchies of expressions. This allows disambiguating pauses to be inserted in appropriate places during conversion to speech.

Support for different ways of creating HTML

HTML 3.0 has been designed to be created in a variety of different ways. It is deliberately simple enough to type in by hand. It can be authored using wysiwyg editors for HTML, or it can be generated via export filters from common word processing formats, or other SGML applications.

Understanding HTML and MIME

--I have dropped the differentiation of HTML into a sequence of conformance levels. Many people confused levels with versions. The different levels also encourage interoperability problems! Lets encourage full conformance with HTML 2.0 or HTML 3.0 rather than perpetuating intermediate levels of support.--

HTML as an Internet Media Type

This (and upward compatible specifications) define the Internet Media Type (RFC 1590) and MIME Content Type (RFC 1521) called "text/html". The type "text/html" accepts the following parameters:

Version

To help avoid future compatibility problems, the version parameter may be used to give the version number of the specification to which the document conforms. The version number appears at the front of this document and within the public identifier for the SGML DTD. This specification defines version 3.0.

Character sets

The charset parameter (as defined in section 7.1.1 of RFC 1521)

may be used with the text/html content type to specify the
encoding used to represent the HTML document as a sequence of
bytes. Normally, text/* media types specify a default of
US-ASCII for the charset parameter. However, for text/html, if
the byte stream contains data that is not in the 7-bit US-ASCII
set, the HTML interpreting agent should assume a default charset
of ISO-8859-1.

When an HTML document is encoded using US-ASCII, the mechanisms
of numeric character references and character entity references
may be used to encode additional characters from ISO-8859-1.
Character entity references are needed for symbols such as math
and greek characters from other unspecified character sets.

Other values for the charset parameter are not defined in this
specification, but may be specified in future versions of HTML.
It is envisioned that HTML will use the charset parameter to
allow support for non-Latin characters such as Arabic, Hebrew,
Cyrillic and Japanese, rather than relying on any SGML mechanism
for doing so.

--What about Unicode and its assorted encodings? This section
would benefit from an explanation of the issues underlying
support for multiple character sets and the problems arising
from bidirectionality.--

Understanding HTML and SGML

HTML is an application conforming to International Standard ISO 8879 -- Standard Generalized Markup Language (SGML). SGML is a system for defining structured document types, and markup languages to represent instances of those document types. The SGML declaration for HTML is given in SGML Declaration for HTML. It is implicit among WWW implementations.

In the event of any apparent conflict between HTML and SGML standards, the SGML standard is definitive.

Every SGML document has three parts:

SGML declaration

Binds SGML processing quantities and syntax token names to specific values. For example, the SGML declaration in the HTML DTD specifies that the string that opens an end tag is </ and the maximum length of a name is 72 characters.

Prologue

Includes one or more document type declarations (DTDs), which specify the element types, element relationships and attributes. The HTML 3.0 DTD provides a definitive specification of the allowed syntax for HTML 3.0 documents.

References

Can be represented by markup. An instance, which contains the data and markup of the document.

HTML refers to the document type as well as the markup language for representing instances of that document type.

Understanding HTML Elements

In HTML documents, tags define the start and end of headings, paragraphs, lists, character highlighting and links. Most HTML elements are identified in a document as a start tag, which gives the element name and attributes, followed by the content, followed by the end tag. Start tags are delimited by < and >, and end tags are delimited by </ and >. For example:

```
<H1>This is a Heading</H1>
<P>This is a paragraph.
```

Some elements appear as just a start tag. For example, to create a line break, you use <BR>. Additionally, the end tags of some other elements (e.g. P, LI, DT, DD) can be omitted as the position of the end tag is clearly implied by the context.

The content of an element is a sequence of characters and nested elements. Some elements, such as anchors, cannot be nested. Anchors and character highlighting may be put inside other constructs. The content model for a tag defines the syntax permitted for the content.

Note: The SGML declaration for HTML specifies SHORTTAG YES, which means that there are other valid syntaxes for tags, such as NET tags, <EM/.../; empty start tags, <>; and empty end tags, </>. Until support for these idioms is widely deployed, their use is strongly discouraged.

Names

The element name immediately follows the tag open delimiter. An element name consist of a letter followed by up to 72 letters, digits, periods, or hyphens. Names are not case sensitive. For example, H1 is equivalent to h1. This limit of 72 characters is set by the NAMELEN parameter in the SGML declaration for HTML 3.0.

Attributes

In a start tag, white space and attributes are allowed between the element name and the closing delimiter. An attribute typically consists of an attribute name, an equal sign, and a value (although some attributes may be just a value). White space is allowed around the equal sign.

The value of the attribute may be either:

1. A string literal, delimited by single quotes or double quotes

2. A name token (a sequence of letters, digits, periods, or hyphens)

In this example, a is the element name, href is the attribute name, and http://host/dir/file.html is the attribute value:

```
<A HREF="http://host/dir/file.html">
```

Some implementations consider any occurrence of the > character to signal the end of a tag. For compatibility with such implementations, when > appears in an attribute value, you may want to represent it with an entity or numeric character reference, such as:

```
<IMG SRC="eq1.ps" alt="a &#62; b">
```

To put quotes inside of quotes, you can use single quotes if the outer quotes are double or vice versa, as in:

```
<IMG SRC="image.ps" alt="First 'real' example">
```

Alternatively, you use the character representation &quot; as in:

```
<IMG SRC="image.ps" alt="First &quot;real&quot; example">
```

The length of an attribute value (after replacing entity and numeric character references) is limited to 1024 characters. This number is defined by the LITLEN parameter in the SGML declaration for HTML 3.0.

Note: Some implementations allow any character except space or > in a name token. Attributes values must be quoted only if they don't satisfy the syntax for a name token.

Attributes with a declared value of NAME (e.g. ISMAP, COMPACT) may be written using a minimized syntax. The markup:

```
<UL COMPACT="compact">
```

can be written as:

```
<UL COMPACT>
```

Note: Unless you use the minimized syntax, some implementations won't understand.

Undefined Tag and Attribute Names

It is an accepted networking principle to be conservative in that which one produces, and liberal in that which one accepts. HTML parsers should be liberal except when verifying code. HTML generators should generate strictly conforming HTML. It is suggested that where ever practical, parsers should at least flag the presence of markup errors, as this will help to avoid bad markup being produced inadvertently.

The behavior of WWW applications reading HTML documents and discovering tag or attribute names which they do not understand should be to behave as though, in the case of a tag, the whole tag had not been there but its content had, or in the case of an attribute, that the attribute had not been present.

Special Characters

The characters between the tags represent text in the ISO-Latin-1 character set, which is a superset of ASCII. Because certain characters will be interpreted as markup, they should be represented by markup -- entity or numeric character references, for instance the character "&" must be represented by the entity &amp;. See the

Special Characters section of this specification for more information.

Comments

To include comments in an HTML document that will be ignored by the parser, surround them with <!-- and -->. After the comment delimiter, all text up to the next occurrence of --> is ignored. Hence comments cannot be nested. White space is allowed between the closing -- and >, but not between the opening <! and --.

For example:

```
<HEAD>
<TITLE>HTML Guide: Recommended Usage</TITLE>
<!-- Id: Text.html,v 1.6 1994/04/25 17:33:48 connolly Exp -->
</HEAD>
```

Note: Some historical implementations incorrectly consider a > sign to terminate a comment.

Formal Variants of HTML 3.0

The HTML 3.0 document type definition includes two flags for controlling how prescriptive or how lax the language is. This makes use of SGML marked sections in the DTD to enable or disable certain features.

HTML.Recommended

Certain features of the language are necessary for compatibility with widespread usage, but they may compromise the structural integrity of a document. The HTML.Recommended entity should be defined as INCLUDE in the DTD subset to enable a more prescriptive version of HTML 3.0 that eliminates the above features. For example:

```
<!DOCTYPE HTML PUBLIC "-//IETF//DTD HTML 3.0//EN"
[ <!ENTITY % HTML.Recommended "INCLUDE"> ] >
```

In particular, this prevents text from appearing except within block elements.

HTML.Deprecated

By default, for backwards compatibility, the %HTML.Deprecated entity is defined as INCLUDE, enabling certain features which are now deprecated. These features can be eliminated by defining this entity as IGNORE in the DTD subset. For example:

```
   <!DOCTYPE HTML PUBLIC "-//IETF//DTD HTML 3.0//EN" [
<!ENTITY % HTML.Deprecated "IGNORE"> ] >
```

Note: defining %HTML.Recommended as INCLUDE automatically sets

%HTML.Deprecated to IGNORE.

In the spirit of being liberal in what you accept and strict in what you generate, HTML user agents are recommended to accept syntax corresponding to the specification with %HTML.Deprecated turned on, while HTML user agents generating HTML are recommended to generate documents that conform to the specification with %HTML.Recommended turned on.

The Structure of HTML 3.0 Documents

In HTML documents, tags define the start and end of headings, paragraphs, lists, character highlighting and links etc. Most HTML elements are identified in a document as a start tag, which gives the element name and attributes, followed by the content, followed by the end tag. Start tags are delimited by < and >, while end tags are delimited by </ and >. For example:

```
<title>This is a Title</title>
<h1>This is a Heading</h1>
<P>This is a paragraph.
```

Every HTML document as a minimum must have a title. To identify the document as being HTML 3.0, it is recommended that documents start with the prologue:

```
<!doctype HTML public "-//W3O//DTD W3 HTML 3.0//EN">
```

When absent, this prologue is implied by the MIME content type for HTML 3.0 together with the associated version parameter.

Document Structure

HTML 3.0 documents formally have the following structure:

```
<HTML>
<HEAD> --head elements ...--
<BODY> --body elements ...--
</HTML>
```

In most cases, the HTML, HEAD and BODY tags can be safely omitted. Note that the formal syntax of HTML 3.0 is defined by the document type definition, which is included as an appendix of this specification. The details of the HEAD and BODY elements will be described in subsequent sections.

The permitted syntax of HTML 3.0 compliant documents is specified by the DTD. This includes the content model for each element, defining what markup is permitted within each element. The DTD uses SGML entities in content models to express regular features of HTML 3.0, for example %body.content defines what markup is permitted within the BODY element. A number of other elements also share this content model, e.g. BQ, DIV, FORM, TH and TD.

The description of each tag includes the --content model-- and the --permitted context-- (which elements can contain this tag). Where practical, these properties are given with the same entity names as used in the DTD, and should help the newcomer to get to grips with understanding the DTD itself. For example, the description of the

NOTE element starts with:

The NOTE element

Permitted context: %block

Content model: %flow

This says that the NOTE element (used for admonishments such as notes, cautions and errors) can occur in any element which includes %block in its content model. Similarly, any element with %flow as part of its permitted context can occur within a NOTE element.

The HTML element

This has three attributes:

VERSION

This is fixed by the DTD as the string "-//W3O//DTD W3 HTML 3.0//EN"

URN

The universal resource name for the document (optional)

ROLE

An optional space separated list of SGML NAME tokens that define the role this document plays, e.g. table of contents. The conventions for these names are outside the scope of this specification. --wouldn't it be better to leave this to a link to a URC?--

Note that both the start and end tag for the HTML element can be omitted.

The Head Element and Related Elements

HEAD

The HEAD element has no attributes and the start and end tag can always be safely omitted as they can be readily inferred by the parser. Information in the HEAD element corresponds to the top part of a memo or mail message. It describes properties of the document such as the title, the document toolbar, and additional meta-information. There is no intended significance to the order of elements in the document head. Note that the TITLE element is always required. In fact, the minimal HTML 3.0 document consists of the TITLE element alone!

Within the HEAD element, only certain elements are allowed. Information in the HEAD element may include the following elements (arranged alphabetically):

BASE

The BASE element allows the URL of the document itself to be recorded in situations in which the document may be read out of context. URLs within the document may be in a "partial" form

relative to this base address. The default base address is the URL
used to retrieve the document.

For example:

```
<base href="http://acme.com/docs/mydoc.html">
...
<img src="images/me.gif">
```

which resolves to "http://acme.com/docs/images/me.gif".

ISINDEX

The ISINDEX element informs the HTML user agent that the document is
an index document. As well as reading it, the reader may use a
keyword search.

The document can be queried with a keyword search by adding a
question mark to the end of the document address, followed by a list
of keywords separated by plus signs. See the network address format
for more information.

Note: A server normally generates the ISINDEX tag automatically. If
added by hand to an HTML document, the browser assumes that the
server can handle a search on the document. Obviously the server

must have this capability for it to work: simply adding <ISINDEX> in the document is not enough to make searches happen if the server does not have a search engine!

Example:

```
<ISINDEX>
```

The URL used for processing queries can be overridden with the HREF attribute. You can also use the PROMPT attribute to change the default prompt supplied by the browser, e.g.

```
<ISINDEX HREF="phone.db" PROMPT="Enter Surname:">
```

LINK

The LINK element indicates a relationship between the document and some other object. A document may have any number of LINK elements. The LINK element is empty (does not have a closing tag), but takes the same attributes as the anchor element. The important attributes are:

REL
    This defines the relationship defined by the link.

REV

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

HREF

This names an object using the URI notation.

Using LINK to define document specific toolbars

An important use of the LINK element is to define a toolbar of navigation buttons or an equivalent mechanism such as menu items.

LINK relationship values reserved for toolbars are:

REL=Home

The link references a home page or the top of some hierarchy.

REL=ToC

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

REL=Index

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

REL=Glossary

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

REL=Copyright

The link references a copyright statement for the current document.

REL=Up

When the document forms part of a hierarchy, this link references the immediate parent of the current document.

REL=Next

The link references the next document to visit in a guided tour.

REL=Previous

The link references the previous document in a guided tour.

REL=Help

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

REL=Bookmark

Bookmarks are used to provide direct links to key entry points into an extended document. The TITLE attribute may be used to label the bookmark. Several bookmarks may be defined in each document, and provide a means for orienting users in extended documents.

An example of toolbar LINK elements:

```
<LINK REL=Previous HREF=doc31.html>
<LINK REL=Next HREF=doc33.html>
<LINK REL=Bookmark TITLE="Order Form" HREF=doc56.html>
```

Using LINK to include a Document Banner

The LINK element can be used with REL=Banner to reference another document to be used as banner for this document. This is typically used for corporate logos, navigation aids, and other information which shouldn't be scrolled with the rest of the document. For example:

```
<LINK REL=Banner HREF=banner.html>
```

The use of a LINK element in this way, allows a banner to be shared between several documents, with the benefit of being able to separately cache the banner. Rather than using a linked banner, you can also include the banner in the document itself, using the BANNER element.

Link to an associated Style Sheet

The LINK element can be used with REL=StyleSheet to reference a style sheet to be used to control the way the current document is rendered. For example:

```
<LINK REL=StyleSheet HREF=housestyle.dsssl>
```

Other uses of the LINK element

Additional relationship names have been proposed, but do not form part of this specification. Servers may also allow links to be added by those who do not have the right to alter the body of a document.

META

The META element is used within the HEAD element to embed document meta-information not defined by other HTML elements. Such information can be extracted by servers/clients for use in identifying, indexing and cataloging specialized document meta-information.

Although it is generally preferable to used named elements that have well defined semantics for each type of meta-information, such as title, this element is provided for situations where strict SGML parsing is necessary and the local DTD is not extensible.

In addition, HTTP servers can read the contents of the document head to generate response headers corresponding to any elements defining a value for the attribute HTTP-EQUIV. This provides document authors with a mechanism (not necessarily the preferred one) for identifying information that should be included in the response headers of an HTTP request.

The META element has three attributes:

NAME

Used to name a property such as author, publication date etc. If absent, the name can be assumed to be the same as the value of HTTP-EQUIV.

CONTENT

Used to supply a value for a named property.

HTTP-EQUIV

This attribute binds the element to an HTTP response header. If the semantics of the HTTP response header named by this attribute is known, then the contents can be processed based on a well defined syntactic mapping, whether or not the DTD includes anything about it. HTTP header names are not case sensitive. If absent, the NAME attribute should be used to identify this meta-information and it should not be used within

an HTPP response header.

Examples:

If the document contains:

```
<META HTTP-EQUIV=Expires CONTENT="Tue, 04 Dec 1993 21:29:02 GMT">
<META HTTP-EQUIV="Keywords" CONTENT="Nanotechnology, Biochemistry">
<META HTTP-EQUIV="Reply-to" CONTENT="dsr@w3.org (Dave Raggett)">
```

The server will include the following response headers:

```
Expires: Tue, 04 Dec 1993 21:29:02 GMT
Keywords: Nanotechnology, Biochemistry
Reply-to: dsr@w3.org (Dave Raggett)
```

When the HTTP-EQUIV attribute is absent, the server should not generate an HTTP response header for this meta-information, e.g.

```
<META NAME="IndexType" CONTENT="Service">
```

Do not use the META element to define information that should be associated with an existing HTML element.

Example of an inappropriate use of the META element:

```
<META NAME="Title" CONTENT="The Etymology of Dunsel">
```

Do not name an HTTP-EQUIV attribute the same as a response header that should typically only be generated by the HTTP server. Some inappropriate names are "Server", "Date", and "Last-Modified". Whether a name is inappropriate depends on the particular server implementation. It is recommended that servers ignore any META elements that specify HTTP equivalents (case insensitively) to their own reserved response headers.

NEXTID

The NEXTID is a parameter read and generated by text editing software to generate unique identifiers. This tag takes a single attribute which is the the next document-wide alpha-numeric identifier to be allocated of the form z123.

When modifying a document, existing anchor identifiers should not be reused, as these identifiers may be referenced by other documents. Human writers of HTML usually use mnemonic alphabetical identifiers.

Example:

```
<NEXTID N=Z27>
```

HTML user agents may ignore the NEXTID element. Support for NEXTID does not impact HTML user agents in any way.

--I want to get rid of NEXTID, or at least deprecate it!--

RANGE

The RANGE element is used to mark a range of the document, for example for highlighting regions of the document matching some search criteria, or which are the subject of an annotation etc.

```
<RANGE CLASS=Search FROM=spot01 UNTIL=spot02>
```

The FROM and UNTIL attributes specify positions in the document using SGML identifiers. Most elements in the document body can define such identifiers using ID attributes. The SPOT element is useful in this regard, as it allows search software etc. to insert IDs at random places:

```
<SPOT ID=spot01> ... <SPOT ID=spot02>
```

The RANGE element supports the following attributes:

ID

An SGML identifer used to name the range element.

CLASS

A character string used to subclass the range element.

FROM

References an SGML identifier for an element in the document body. It identifies the start of the marked range.

UNTIL

References an SGML identifier for an element in the document body. It identifies the end of the marked range.

STYLE

The STYLE element provides a means for including rendering information using a specified style notation. Information in the STYLE element overrides client defaults and that of linked style sheets. It allows authors to specify overrides, while for the most part using a generic style sheet, and as such improves the effectiveness of caching schemes for linked style sheets. There is one attribute - NOTATATION - which specifies an entity identifying an SGML notation in the HTML 3.0 DTD, for example:

```
<style notation=dsssl-lite>
  --some dsssl-lite stuff ...--
```

```
</style>
```

Stylistic rules will in general match tag names and attribute values for elements in the document body. Context sensitive rules may be used for such purposes as rendering drop down capitals for the initial letter in the first paragraph following a header.

TITLE

Every HTML document must contain a TITLE element. The title should identify the contents of the document in a global context, and may be used in a history lists and as a label for the window displaying the document. Unlike headings, titles are not normally displayed in the text of a document itself.

The TITLE element must occur within the head of the document, and may not contain anchors, paragraph tags, or highlighting. There may only be one TITLE in any document.

The length of titles is unlimited, however, long titles may be truncated in some applications. To minimize this possibility, keep titles to fewer than 64 characters. Also keep in mind that a short title, such as Introduction, may be meaningless out of context. An example of a meaningful title might be:

```
<Title>Recent Advances in Nanotechnology</Title>
```

The Body Element and Related Elements

The BODY element

Permitted Context: HTML

Content Model: %Body.Content

Within the BODY element, you can structure text into paragraphs, and lists, as well as highlighting phrases and creating links, amongst other things. The BODY element has the following attributes, all of which are optional:

Note that the ID, LANG and CLASS attributes can be used with virtually all of the elements permitted in the document body.

ID

An SGML identifier used as the target for hypertext links or for naming particular elements in associated style sheets. Identifiers are NAME tokens and must be unique within the scope of the current document.

LANG

This is one of the ISO standard language abbreviations, e.g. "en.uk" for the variation of English spoken in the United

Kingdom. It can be used by parsers to select language specific choices for quotation marks, ligatures and hypenation rules etc. The language attribute is composed from the two letter language code from ISO 639, optionally followed by a period and a two letter country code from ISO 3166.

CLASS

This a space separated list of SGML NAME tokens and is used to subclass tag names. For instance, <P CLASS=STANZA.COUPLET> defines a paragraph that acts as a couplet in a stanza. By convention, the class names are interpreted hierarchically, with the most general class on the left and the most specific on the right, where classes are separated by a period. The CLASS attribute is most commonly used to attach a different style to some element, but it is recommended that where practical class names should be picked on the basis of the element's semantics, as this will permit other uses, such as restricting search through documents by matching on element class names. The conventions for choosing class names are outside the scope of this specification.

BACKGROUND

This can be used to specify a URI for an image tile to cover the document background. This provides a way of giving a group of documents a distinctive appearence. Clients may ignore this attribute. It is included here for the benefit of clients that don't support style sheets. Note that the text color may need to be adjusted to show an adequate contrast with the background.

Note that you don't need to include a BODY tag unless you want to specify one of the above attributes.

Body Structure

The document body is composed from zero or more of the following elements:

* DIV - used for hierarchical containers and static banners

* Headings (H1, to H6) - a set of headers of varying levels of importance

* Block elements - paragraphs, lists, forms, tables, figures and other elements

* Horizontal rules, and the ADDRESS element

* Text and character level markup including emphasis, images, math, hypertext links and miscellaneous elements.

Note that text and character level markup are only permitted at this level for backwards compatibility with legacy documents. The HTML.Recommended flag enforces a more structured approach to authoring HTML documents.

Banners

Permitted Context: the start of the BODY element

Content Model: %Body.Content

The BANNER element is used for corporate logos, navigation aids, disclaimers and other information which shouldn't be scrolled with the rest of the document. It provides an alternative to using the LINKelement in the document head to reference an externally defined banner.

Permitted Attributes

ID

An SGML identifier used as the target for hypertext links or for naming particular elements in associated style sheets. Identifiers are NAME tokens and must be unique within the scope of the current document.

LANG

This is one of the ISO standard language abbreviations, e.g. "en.uk" for the variation of English spoken in the United Kingdom. It can be used by parsers to select language specific choices for quotation marks, ligatures and hypenation rules etc.

The language attribute is composed from the two letter language code from ISO 639, optionally followed by a period and a two letter country code from ISO 3166.

CLASS

This a space separated list of SGML NAME tokens and is used to subclass tag names. By convention, the class names are interpreted hierarchically, with the most general class on the left and the most specific on the right, where classes are separated by a period. The CLASS attribute is most commonly used to attach a different style to some element, but it is recommended that where practical class names should be picked on the basis of the element's semantics, as this will permit other uses, such as restricting search through documents by matching on element class names. The conventions for choosing class names are outside the scope of this specification.

Divisions

Permitted Context: %Body.Content

Content Model: %Body.Content

The DIV element is used with the CLASS attribute to represent different kinds of containers, e.g. chapter, section, abstract, or appendix. For example:

```
<DIV CLASS=Abstract>
<P>TheChieftain product range is the white hot hope for the
coming year. This report sets out how to position Chieftain
against competing products.
</DIV>
```

Permitted Attributes

ID

An SGML identifier used as the target for hypertext links or for naming particular elements in associated style sheets. Identifiers are NAME tokens and must be unique within the scope of the current document.

LANG

This is one of the ISO standard language abbreviations, e.g. "en.uk" for the variation of English spoken in the United Kingdom. It can be used by parsers to select language specific choices for quotation marks, ligatures and hypenation rules etc. The language attribute is composed from the two letter language code from ISO 639, optionally followed by a period and a two letter country code from ISO 3166.

CLASS

This a space separated list of SGML NAME tokens and is used to subclass tag names. For instance, <DIV CLASS=APPENDIX> defines a division that acts as an appendix. By convention, the class names are interpreted hierarchically, with the most general class on the left and the most specific on the right, where classes are separated by a period. The CLASS attribute is most commonly used to attach a different style to some element, but it is recommended that where practical class names should be picked on the basis of the element's semantics, as this will permit other uses, such as restricting search through documents by matching on element class names. The conventions for choosing class names are outside the scope of this specification.

ALIGN

The ALIGN attribute can be used to explicitly specify the horizontal alignment of paragraphs within a division:

align=left

Paragraphs are rendered flush left (the default).

align=center

Paragraphs are centered.

align=right

Paragraphs are rendered flush right.

align=justify

Text lines are justified where practical, otherwise this gives the same effect as the default align=left setting.

NOWRAP

The NOWRAP attribute is used when you don't want the browser to automatically wrap lines. You can then explicitly specify line breaks in paragrphs using the BR element.

CLEAR

This attribute is common to all block-like elements. When text flows around a figure or table in the margin, you sometimes want to start the division below the figure rather than alongside it. The CLEAR attribute allows you to move down unconditionally:

clear=left

move down until left margin is clear

clear=right

move down until right margin is clear

clear=all

move down until both margins are clear

Alternatively, you can decide to place the element alongside the figure just so long as there is enough room. The minimum width needed is specified as:

clear="40 en"

move down until there is at least 40 en units free

clear="100 pixels"

move down until there is at least 100 pixels free

The style sheet (or browser defaults) may provide default minimum widths for each class of block-like elements.

Headings

Permitted Context: %Body.Content

Content Model: %text

HTML defines six levels of headings. A heading element implies all the font changes, paragraph breaks before and after, and any white space necessary to render the heading. The heading elements are H1, H2, H3, H4, H5, and H6 with H1 being the highest (or most important) level and H6 the least. For example:

```
<H1>This is a top level heading</H1> Here is some text.
<H2>Second level heading</H2> Here is some more text.
```

Use the DIV element together with header elements when you want to make the hierarchical structure of a document explicit. This is needed as header elements themselves only contain the text of the header, and do not imply any structural division of documents into sections. Header elements have the same content model as paragraphs, that is text and character level markup, such as character emphasis, inline images, form fields and math.

Headers play a related role to lists in structuring documents, and it is common to number headers or to include a graphic that acts like a bullet in lists. HTML 3.0 recognizes this with attributes that assist with numbering headers and allow authors to specify a custom graphic.

The numbering style is controlled by the style sheet, e.g.

1. The style sheet specifies whether headers are numbered, and which style is used to render the current sequence number, e.g. arabic, upper alpha, lower alpha, upper roman, lower roman or a numbering scheme appropriate to the current language.

2. Whether the parent numbering is inherited, e.g. "5.1.d" where 5 is the current sequence number for H1 headers, 1 is the number for H2 headers and 4 for H3 headers.

The seqnum and skip attributes can be used to override the default treatment of header sequence numbers, and provide for a continuity with numbered lists.

The dingbat or src attribute may be used to specify a bullet-like graphic to be placed adjacent to the header. The positioning of this graphic is controlled by the style sheet. The graphic is for decorative purposes only and silently ignored on non-graphical HTML user agents.

Word Wrapping

User agents are free to wrap lines at whitespace characters so as to ensure lines fit within the current window size. Use the &nbsp;

entity for the non-breaking space character, when you want to make sure that a line isn't broken! Alternatively, use the NOWRAP attribute to disable word wrapping and the <BR> element to force line breaks where desired.

--Netscape includes two tags: <NOBR>...</NOBR>, and <WBR>. The former turns off wordwrapping between the start and end NOBR tag, while WBR is for the rare case when you want to specify where to break the line if needed. Should HTML 3.0 provide an equivalent mechanism to WBR, (either a tag or an entity)?--

Permitted Attributes

ID

An SGML identifier used as the target for hypertext links or for naming particular elements in associated style sheets. Identifiers are NAME tokens and must be unique within the scope of the current document.

LANG

This is one of the ISO standard language abbreviations, e.g. "en.uk" for the variation of English spoken in the United Kingdom. It can be used by parsers to select language specific choices for quotation marks, ligatures and hypenation rules etc.

The language attribute is composed from the two letter language code from ISO 639, optionally followed by a period and a two letter country code from ISO 3166.

CLASS

This a space separated list of SGML NAME tokens and is used to subclass tag names. For instance, <H2 CLASS=Section> defines a level 2 header that acts as a section header. By convention, the class names are interpreted hierarchically, with the most general class on the left and the most specific on the right, where classes are separated by a period. The CLASS attribute is most commonly used to attach a different style to some element, but it is recommended that where practical class names should be picked on the basis of the element's semantics, as this will permit other uses, such as restricting search through documents by matching on element class names. The conventions for choosing class names are outside the scope of this specification.

ALIGN

Headings are usually rendered flush left. The ALIGN attribute can be used to explicitly specify the horizontal alignment:

align=left

The heading is rendered flush left (the default).

align=center

The heading is centered.

align=right

The heading is rendered flush right.

align=justify

Heading lines are justified where practical, otherwise this gives the same effect as the default align=left setting.

For example:

```
<h1 align=center>This is a centered heading</H1>
Here is some text. <H2 align=right>and this is a flush right
heading</H2> Here is some more text.
```

CLEAR

This attribute is common to all block-like elements. When text flows around a figure or table in the margin, you sometimes want to start an element like a header, paragraph or list below the figure rather than alongside it. The CLEAR attribute allows you to move down unconditionally:

clear=left

move down until left margin is clear

clear=right

move down until right margin is clear

clear=all

move down until both margins are clear

Alternatively, you can decide to place the element alongside the figure just so long as there is enough room. The minimum width needed is specified as:

clear="40 en"

move down until there is at least 40 en units free

clear="100 pixels"

move down until there is at least 100 pixels free

The style sheet (or browser defaults) may provide default minimum widths for each class of block-like elements.

SEQNUM

A sequence number is associated with each level of header from the top level (H1) to the bottom level (H6). This attribute is used to set the sequence number associated with the header level of the current element to a given number, e.g. SEQNUM=10. Normally, the sequence number is initialized to 1 at the beginning of the document and incremented after each header element. It is reset to 1 by any header element of a higher level, e.g. an H1 header resets the sequence numbers for H2 to H6. The style of header numbering is controlled by the style sheet.

SKIP

Increments the sequence number before rendering the element. It is used when headers have been left out of the sequence. For instance, SKIP=3 advances the sequence number past 3 omitted items.

DINGBAT

Specifies an iconic image to appear preceding the header. The icon is specified as an entity name. A list of standard icon entity names for HTML 3.0 is given in an appendix of this specification.

SRC

Specifies an image to appear preceding the header. The image is specified as a URI. This attribute may appear together with the MD attribute.

MD

Specifies a message digest or cryptographic checksum for the associated graphic specified by the SRC attribute. It is used when you want to be sure that a linked object is indeed the same one that the author intended, and hasn't been modified in any way. For instance, MD="md5:jV2OfH+nnXHU8bnkPAad/mSQlTDZ", which specifies an MD5 checksum encoded as a base64 character string. The MD attribute is generally allowed for all elements which support URI based links.

NOWRAP

The NOWRAP attribute is used when you don't want the browser to automatically wrap lines. You can then explicitly specify line breaks in headings using the BR element. For example:

```
<h1 nowrap>This heading has wordwrap turned off<br>
and the BR element is used for explicit line breaks</H1>
```

Paragraphs

Permitted Context: %Body.Content, %flow, %block

Content Model: %text

The <P> element is used to define a paragraph. The exact rendering (indentation, leading etc.) is not defined and may be a function of other tags, style sheets, etc. The ALIGN attribute can be used to explicitly specify the horizontal alignment. Paragraph elements have the same content model as headers, that is text and character level markup, such as character emphasis, inline images, form fields and math.

Example:

```
<H1>The heading precedes the first paragraph</H1>
<P>Here is the text of the first paragraph. <P>and this is
the text of the second paragraph.
```

The text up to the next <p> element is treated as being part of the current paragraph. This is an example of how SGML allows certain end tags like </p> to be left out where they can be inferred from the context.

Word Wrapping

User agents are free to wrap lines at whitespace characters so as to ensure lines fit within the current window size. Use the &nbsp; entity for the non-breaking space character, when you want to make sure that a line isn't broken! Alternatively, use the NOWRAP attribute to disable word wrapping and the <BR> element to force line breaks where desired.

--Netscape includes two tags: <NOBR>...</NOBR>, and <WBR>. The former turns off wordwrapping between the start and end NOBR tag, while WBR is for the rare case when you want to specify where to break the line if needed. Should HTML 3.0 provide an equivalent mechanism to WBR, (either a tag or an entity)?--

Note: Do not use empty paragraphs to add white space around headings, lists or other elements. White space is added by the rendering software.

Permitted Attributes

ID

An SGML identifier used as the target for hypertext links or for naming particular elements in associated style sheets. Identifiers are NAME tokens and must be unique within the scope of the current document.

LANG

This is one of the ISO standard language abbreviations, e.g.

"en.uk" for the variation of English spoken in the United Kingdom. It can be used by parsers to select language specific choices for quotation marks, ligatures and hypenation rules etc. The language attribute is composed from the two letter language code from ISO 639, optionally followed by a period and a two letter country code from ISO 3166.

CLASS

This a space separated list of SGML NAME tokens and is used to subclass tag names. For instance, <P CLASS=abstract> defines a paragraph that acts as an abstract. By convention, the class names are interpreted hierarchically, with the most general class on the left and the most specific on the right, where classes are separated by a period. The CLASS attribute is most commonly used to attach a different style to some element, but it is recommended that where practical class names should be picked on the basis of the element's semantics, as this will permit other uses, such as restricting search through documents by matching on element class names. The conventions for choosing class names are outside the scope of this specification.

ALIGN

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

align=left

The paragraph is rendered flush left (the default).

align=center

The paragraph is centered.

align=right

The paragraph is rendered flush right.

align=justify

Text lines are justified where practical, otherwise this gives the same effect as the default align=left setting.

For example:

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

CLEAR

This attribute is common to all block-like elements. When text flows around a figure or table in the margin, you sometimes want to start an element like a header, paragraph or list below the figure rather than alongside it. The CLEAR attribute allows you to move down unconditionally:

clear=left

move down until left margin is clear

clear=right

move down until right margin is clear

clear=all

move down until both margins are clear

Alternatively, you can decide to place the element alongside the figure just so long as there is enough room. The minimum width needed is specified as:

clear="40 en"

move down until there is at least 40 en units free

clear="100 pixels"

move down until there is at least 100 pixels free

The style sheet (or browser defaults) may provide default minimum widths for each class of block-like elements.

NOWRAP

The NOWRAP attribute is used when you don't want the browser to automatically wrap lines. You can then explicitly specify line breaks in paragraphs using the BR element. For example:

```
<p nowrap>This paragraph has wordwrap turned off<br>
and the BR element is used for explicit line breaks
```

Line Breaks

Permitted Context: %text

Content Model: Empty!

Line break and tab elements can be used when you need a little more control over how the browser renders the text. The <BR> element is used to force a line break.

For example:

```
This is the first line<br>
and this is the second<br>
and this the third
```

--Shouldn't we have a conditional line break element like Netscape's WBR thats indicates where to break lines when needed and when wordwrap is disabled? Rather than an element, shouldn't this be an entity - is there one already defined for this purpose?--

Permitted Attributes

ID

An SGML identifier used as the target for hypertext links or for

naming particular elements in associated style sheets.
Identifiers are NAME tokens and must be unique within the scope
of the current document.

LANG

This is one of the ISO standard language abbreviations, e.g.
"en.uk" for the variation of English spoken in the United
Kingdom. It can be used by parsers to select language specific
choices for quotation marks, ligatures and hypenation rules etc.
The language attribute is composed from the two letter language
code from ISO 639, optionally followed by a period and a two
letter country code from ISO 3166.

CLASS

This a space separated list of SGML NAME tokens and is used to
subclass tag names. By convention, the class names are
interpreted hierarchically, with the most general class on the
left and the most specific on the right, where classes are
separated by a period. The CLASS attribute is most commonly used
to attach a different style to some element, but it is
recommended that where practical class names should be picked on
the basis of the element's semantics, as this will permit other
uses, such as restricting search through documents by matching
on element class names. The conventions for choosing class names
are outside the scope of this specification.

CLEAR

When text flows around a figure or table in the margin, you sometimes want to start the next line below the figure rather than alongside it. The CLEAR attribute allows you to move down unconditionally:

clear=left
: move down until left margin is clear

clear=right
: move down until right margin is clear

clear=all
: move down until both margins are clear

Alternatively, you can decide to place the element alongside the figure just so long as there is enough room. The minimum width needed is specified as:

clear="40 en"
: move down until there is at least 40 en units free

clear="100 pixels"
: move down until there is at least 100 pixels free

The style sheet (or browser defaults) may provide default minimum widths for each class of block-like elements.

Horizontal Tabs

Permitted Context: %text

Content Model: Empty!

The TAB element can be used when you want fine control over the horizontal positioning. The TAB element is used with the <tab id=--name--> attribute to define named tab stops. Subsequently, you can use the TAB element with the <tab to=--name--> attribute to move to the previously defined tab stop. This approach avoids the need to know the font metrics in advance. The TAB element, together with style sheets, allows conversion software to preserve layout information when importing documents created with conventional word processing software.

For example:

```
<p><b>noct<tab id=t1>ambulant</b> - walking at night<br>
<tab to=t1>(from Latin: <i>nox noctis</i> night + <i>ambulare</i> walk)
```

which is rendered as:

noctambulant - walking at night

    (from Latin: --nox noctis-- night + --ambulare-- walk)

The tab stop name (--t1-- in the example) should be unique within the current document and composed from an initial letter followed by letters, digits or hyphens.

Sometimes, you want to make the remainder of the line flush right while leaving the earlier words unmoved. This is possible with the --align-- attribute. For example:

Left part of line<tab align=right>and right part of line.

which is rendered as:

Left part of line   and right part of line.

Permitted Attributes

ID

An SGML identifier used to name a new tab stop at the current position. The scope of the tab stop is the rest of the document.

INDENT

Specifies the number of en units before the tab stop. The en is a typographical unit equal to half the point size. It allows authors to control the leading indent before text, e.g. in poetry, one might use: <TAB INDENT=6> to indent six en units at the start of a line. The INDENT attribute is not meaningful when combined with the TO attribute.

TO

Specifies a previously defined tab stop (see ID attribute).

ALIGN

Lines are usually rendered according to the alignment option for the enclosing paragraph element. The ALIGN attribute can be used to explicitly specify the horizontal alignment:

align=left

Following text starts immediately after the designated tab stop (the default).

align=center

Following text up to next tab or line break is centered on the designated tab stop. If the TO attribute is missing, it centers the text between the current left and right margins.

align=right

Following text up to the next tab or line break is rendered flush right to the designated tab stop. If the TO attribute is missing, it renders the text flush right against the current right margin.

align=decimal

The following text is searched for the first occurrence of the character representing the decimal point. The text up to the next tab or line break is then aligned such that the decimal point starts at the designated tab stop. If the TO attribute is missing, the tab element is treated as a single space character.

DP

This specifies the character to be used for the decimal point with the ALIGN attribute, e.g. dp="." (the default) or dp=",". The default may be altered by the language context, as set by the LANG attribute on enclosing elements.

Note: if the specified alignment and tab stop would cause text to overlap preceding text, then the tab element may be treated as a single space character.

--How should the above be rewritten to work with languages which are rendered from right to left? What about lines with mixed directions?--

Hypertext Links

Permitted Context: %text

Content Model: %text, but no nested anchors

The anchor <A> element is used to define the start and/or destination of a hypertext link. In previous versions of HTML it provided the only means for defining destination anchors within documents, but you can now use any ID attribute as a destination anchor so that links can now be made to divisions, paragraphs and most other elements.

Example:

```
The <A HREF="http://www.w3.org/">World Wide Web Organization</A>
provides information on Web related standards, mailing lists
and freeware tools.
```

The text between the start and end tag defines the label for the link. Selecting the link takes the reader to the document specified by the HREF attribute, in this case, the W3O home page. The label can include graphics defined with IMG elements.

For FIG elements, the anchor element serves a dual role.

Non-graphical user agents interpret it as a conventional text-based hypertext link, while graphical user agents interpret the anchor's SHAPE attribute as a graphical hotzone on the figure.

Permitted Attributes

ID

An SGML identifier used as the target for hypertext links or for naming particular elements in associated style sheets. Identifiers are NAME tokens and must be unique within the scope of the current document. This attribute supercedes the "NAME" attribute, see below.

For example, the following paragraph is defined as an anchor named "potomac":

```
<P ID="potomac">The Potomac river flows into Boston harbour,
and played an important role in opening up the hinterland
to early settlers...
```

Elsewhere, you can define a link to this paragraph, as follows:

```
<A HREF="#potomac">Boston</A> is a historic city and
a thriving center of commerce and higher education.
```

The reader can select the link labelled "Boston" to see further information on the Boston area.

LANG

This is one of the ISO standard language abbreviations, e.g. "en.uk" for the variation of English spoken in the United Kingdom. It can be used by parsers to select language specific choices for quotation marks, ligatures and hypenation rules etc. The language attribute is composed from the two letter language code from ISO 639, optionally followed by a period and a two letter country code from ISO 3166.

CLASS

This a space separated list of SGML NAME tokens and is used to subclass tag names. By convention, the class names are interpreted hierarchically, with the most general class on the left and the most specific on the right, where classes are separated by a period. The CLASS attribute is most commonly used to attach a different style to some element, but it is recommended that where practical class names should be picked on the basis of the element's semantics, as this will permit other uses, such as restricting search through documents by matching on element class names. The conventions for choosing class names are outside the scope of this specification.

HREF

The HREF attribute implies that the anchor acts as the start of a hypertext link. The destination is designated by the value of

the HREF attribute, which is expressed in the Universal Resource
Identifier (URI) notation.

MD

Specifies a message digest or cryptographic checksum for the
linked document designated by the HREF attribute. It is used
when you want to be sure that a linked object is indeed the same
one that the author intended, and hasn't been modified in any
way. For instance, MD="md5:jV2OfH+nnXHU8bnkPAad/mSQlTDZ", which
specifies an MD5 checksum encoded as a base64 character string.
The MD attribute is generally allowed for all elements which
support URI based links.

NAME

This attribute is used to define a named anchor for use as the
destination of hypertext links. For example, the following
defines an anchor than can be used as the destination of a jump
into a description of the Boston area.

The <A NAME="potomac"&gtPotomac river</A> flows into Boston
harbour.

Note: the NAME attribute has been superceded by the ID
attribute. User agents should include support for NAME to ensure
backwards compatibility with legacy documents produced using
previous versions of HTML.

SHAPE

This attribute is used within figures to define shaped hotzones for graphical hypertext links. Full details of how to use this feature will be given with the description of the figure element. The attribute value is a string taking one of the following forms:

"default"

Used to define a default link for the figure background.

"circle x, y, r"

Where x and y define the center and r specifies the radius.

"rect x, y, w, h"

Where x, y define the upper left corner and w, h define the width and height respectively

"polygon x1, y1, x2, y2, ..."

Given n pairs of x, y coordinates, the polygon is closed by a line linking the n'th point to the first. Intersecting polygons use the non-zero winding number rule to determine if a point lies inside the polygon.

If a pointer event occurs in a region where two or more shapes overlap, the distance from the point to the center of gravity of each of the overlapping shapes is computed and the closest one chosen. This feature is useful when you want lots of closely spaced hotzones, for example over points on a map, as it allows you to use simple shapes without worrying about overlaps.

Note: The x coordinate increases to the right, and the y coordinate increases downwards in the same way as IMG and image maps. If both numbers are integers, the coordinates are interpreted as pixel offsets from the upper left corner of the figure. Otherwise, the coordinates are interpreted as scaled values in the range 0.0 to 1.0 across the figure. Note the syntax is tolerant of repeated white space characters between tokens.

TITLE

This is informational only and describes the object specified with the HREF attribute. It can be used for object types that don't possess titles, such as graphics, plain text and Gopher menus.

REL

Used to describe the relationship of the linked object specified with the HREF attribute. The set of relationship names is not part of this specification, although "Path" and "Node" are reserved for future use with hypertext paths or guided tours.

The REL attribute can be used to support search for links serving particular relationships.

REV

This defines a reverse relationship. A link from document A to document B with REV=--relation-- expresses the same relationship as a link from B to A with REL=--relation--. REV=made is sometimes used to identify the document author, either the author's email address with a --mailto-- URI, or a link to the author's home page. Tables of contents can use anchors with REV="ToC" to allow software to insert page numbers when printing hypertext documents. The plain text version of this specification was generated in this way!

Overview of Character-Level Elements

Permitted Context: %text

Content Model: %text

Character level elements are used to specify either the structural meaning or the physical appearence of marked text without causing a paragraph break. Like most other elements, character level elements include both start and end tags. Only the characters between the tags are effected. For example:

```
This is <EM>emphasized</EM> text.
```

Highlighting elements are allowed within the content of other highlighting elements, but implementations are not required to render these nested highlighting elements distinctly from non-nested elements. For example, implementations may render the following two cases identically:

```
plain <B>bold <I>italic</I></B>

plain <B>bold </B><I>italic</I>
```

Some character highlighting styles are more explicit than others about how they should be physically represented. Designate the information type rather than the character format wherever possible, unless for example, it is necessary to refer to the text as in "The italic parts are mandatory".

Permitted Attributes

ID

An SGML identifier used as the target for hypertext links or for naming particular elements in associated style sheets. Identifiers are NAME tokens and must be unique within the scope of the current document.

LANG

This is one of the ISO standard language abbreviations, e.g. "en.uk" for the variation of English spoken in the United Kingdom. It can be used by parsers to select language specific choices for quotation marks, ligatures and hypenation rules etc. The language attribute is composed from the two letter language code from ISO 639, optionally followed by a period and a two letter country code from ISO 3166.

CLASS

This a space separated list of SGML NAME tokens and is used to subclass tag names. By convention, the class names are interpreted hierarchically, with the most general class on the left and the most specific on the right, where classes are

separated by a period. The CLASS attribute is most commonly used

to attach a different style to some element, but it is recommended that where practical class names should be picked on the basis of the element's semantics, as this will permit other uses, such as restricting search through documents by matching on element class names. The conventions for choosing class names are outside the scope of this specification.

Information Type Elements

EM

The <EM> element provides typographic emphasis, typically italics. While <EM> and <I> often give the same effect, use <EM> except where it is necessary in the text to refer to the formatting, as in "The italic parts are mandatory". This will help to ensure consistency between documents from various sources if (for example) the reader prefers to use color in place of italics for emphasis.

CITE

The <CITE> element specifies a citation. Sections tagged with the CITE element are typically rendered in italics.

STRONG

The <STRONG> element provides strong typographic emphasis, typically bold.

CODE

The <CODE> element indicates an example of code; typically rendered in a mono-spaced font. Do not confuse with PRE.

SAMP

The <SAMP> element indicates a sequence of literal characters.

KBD

The <KBD> element indicates text typed (keyboarded) by the user. It might typically be used in an instruction manual.

VAR

The <VAR> element indicates a variable name, and might typically be used in an instruction manual.

DFN

The <DFN> element indicates the defining instance of a term. --New in 3.0--.

Q

The <Q> element is used for a short quotation. It is typically shown enclosed in quotation marks as appropriate to the language context. For English these would be matching double or single quotation marks, alternating for nested quotes. The language context is set by the LANG attribute. --New in 3.0--.

LANG

The <LANG> element is used to alter the language context when it is inappropriate to do this with other character-level elements. --New in 3.0--.

AU

The <AU> element indicates the name of an author. --New in 3.0--.

PERSON

The <PERSON> element is used for names of people to allow these to be extracted automatically by indexing programs. --New in 3.0--.

ACRONYM

The <ACRONYM> element is used to markup acronyms. --New in 3.0--.

ABBREV

The <ABBREV> element is used to markup abbreviations. --New in 3.0--.

INS

The <INS> element is used for inserted text, for instance in legal documents. --New in 3.0--.

DEL

The <DEL> is used for deleted text, for instance in legal documents. --New in 3.0--.

An example:

This text contains an <em>emphasized</em> word.

<strong>Don't assume</strong> that it will be italic!

It was made with the <code>EM</code> element. A cite is

often italic and has no formally required structure:

<cite>Moby Dick</cite> is a book title.

Font Style Elements

These elements may be nested within one another. Browsers should, where practical, aim to combine different types of highlighting as required.

B (Boldface)

The <B> element specifies that the enclosed text should be displayed in a boldface. If this is not practical, an alternative mapping is allowed.

I (Italic)

The <I> element specifies that the enclosed text should be displayed, if practical, in an italic font (or slanted).

TT (TeleType)

The <TT> element specifies that the enclosed text should be displayed, if practical, in a fixed-pitch typewriter font.

U (Underline)

The <U> element specifies that the enclosed text should be displayed, if practical, as underlined. --Not widely supported--

S (Strike through)

The <S> element specifies that the enclosed text should be displayed with a horizontal line striking through the text. If this is not practical, an alternative mapping is allowed. --New in 3.0--.

BIG (Big print)

The <BIG> element specifies that the enclosed text should be displayed, if practical, using a big font (compared with the current font). --New in 3.0--.

SMALL (Small print)

The <SMALL> element specifies that the enclosed text should be displayed, if practical, using a small font (compared with normal text). --New in 3.0--.

SUB (Subscript)

The <SUB> element specifies that the enclosed text should be displayed as a subscript, and if practical, using a smaller font (compared with normal text). The ALIGN attribute for SUB is only meaningful within the MATH element. --New in 3.0--.

SUP (Superscript)

The <SUP> element specifies that the enclosed text should be displayed as a superscript, and if practical, using a smaller font (compared with normal text). The ALIGN attribute for SUP is only applicable within the MATH element. --New in 3.0--.

An example:

```
This text contains some <b><i>bold italic</i></b> text, some

<S>struck through</S> text and some <SMALL>small print</SMALL>.
```

The IMG (Image) Element

Permitted Context: %text

Content Model: Empty!

The <IMG> tag is used to incorporate in-line graphics (typically icons or small graphics) into an HTML document. This element is NOT intended for embedding other HTML text. For large figures with captions and text flow see FIG element.

Example:

```
<IMG SRC="tajmahal.gif" ALT="The Taj Mahal">
```

Browsers that cannot display in-line images ignore the IMG element unless it contains the ALT attribute. Note that some browsers can display (or print) linked graphics but not in-line graphics. If the graphic is essential, you may want to create a link to it rather than to put it in-line. If the graphic is essentially decorative, then IMG is appropriate.

Permitted Attributes

ID

An SGML identifier used as the target for hypertext links or for naming particular elements in associated style sheets. Identifiers are NAME tokens and must be unique within the scope of the current document.

LANG

This is one of the ISO standard language abbreviations, e.g. "en.uk" for the variation of English spoken in the United Kingdom. It can be used by parsers to select language specific choices for quotation marks, ligatures and hypenation rules etc. The language attribute is composed from the two letter language code from ISO 639, optionally followed by a period and a two letter country code from ISO 3166.

CLASS

This a space separated list of SGML NAME tokens and is used to subclass tag names. By convention, the class names are interpreted hierarchically, with the most general class on the left and the most specific on the right, where classes are separated by a period. The CLASS attribute is most commonly used to attach a different style to some element, but it is recommended that where practical class names should be picked on the basis of the element's semantics, as this will permit other uses, such as restricting search through documents by matching on element class names. The conventions for choosing class names are outside the scope of this specification.

SRC (Source)

The SRC attribute specifies the URI for the image to be embedded. Its syntax is the same as that of the HREF attribute of the <A> tag. SRC is mandatory.

MD

Specifies a message digest or cryptographic checksum for the associated graphic specified by the SRC attribute. It is used when you want to be sure that the image is indeed the same one that the author intended, and hasn't been modified in any way. For instance, MD="md5:jV2OfH+nnXHU8bnkPAad/mSQlTDZ", which specifies an MD5 checksum encoded as a base64 character string. The MD attribute is generally allowed for all elements which support URI based links.

WIDTH

Optional suggested width for the image. By default, this is given in pixels.

HEIGHT

Optional suggested height for the image. By default, this is given in pixels.

UNITS

This optional attribute specifies the units for the width and height attributes. It is one of: units=pixels (the default) or units=en (half the point size).

ALIGN

Take values TOP or MIDDLE or BOTTOM, defining whether the top or middle or bottom of the graphic should be aligned with the baseline for the text line in which the IMG element appears.

With ALIGN=LEFT, the graphic will float down and over to the current left margin, and subsequent text will wrap around the right hand side of the graphic. Likewise for ALIGN=RIGHT, the graphic aligns with the current right margin and, and text wraps around the left. It is inappropriate to use this feature for larger graphics as these are best represented with the FIG element.

ALT (Alternate text)

Optional alternative text as an alternative to the graphics for display in text-only environments. The alt text can contain entities e.g. for accented characters or special symbols, but it can't contain markup. The latter is possible, however, with the FIG element.

ISMAP

An image map is a graphical map by which users can navigate transparently from one information resource to another. The ISMAP attribute identifies an image as an image map. The IMG element can then be used as part of the label for a hypertext

link (see the anchor element). When the user clicks on the image

the location clicked is sent to the server designated by the hypertext link.

For example:

```
<A HREF="http://machine/htbin/imagemap/sample">
<IMG SRC="sample.gif" ISMAP></A>
```

Note: There are drawbacks from having the server process clicks on images: the delay in getting feedback and the inability to change the pointer cursor on the fly as it moves over hotzones. Client-side processing of events is possible if one of the following applies:

* The server may allow the image map to be downloaded and processed locally. This should work with legacy documents produced using earlier versions of HTML.

* Using an image format that includes image hotzones as part of the file format.

* The FIG element provides for client-side image maps as a unified part of the figure description. It offers a number of advantages over IMG, including captions, markup in alt text and text flow around figures.

UL (Unordered List)

Permitted Context: %Body.Content, %flow, %block

Content Model: Optional list header (LH), followed by one or more list items(LI)

An unordered list typically is a bulleted list of items. HTML 3.0 gives you the ability to customise the bullets, to do without bullets and to wrap list items horizontally or vertically for multicolumn lists.

The opening list tag must be <UL>. It is followed by an optional list header (<LH>caption</LH>) and then by the first list item (<LI>). For example:

```
<UL>
  <LH>Table Fruit</LH>
  <LI>apples
  <LI>oranges
  <LI>bananas
</UL>
```

which could be rendered as:

Table Fruit

   *   apples

   *   oranges

   *   bananas

Note: Some legacy documents may include headers or plain text before the first LI element. Implementors of HTML 3.0 user agents are advised to cater for this possibility in order to handle badly formed legacy documents.

MENU and DIR elements

These elements are superceded by extensions to the UL element. User agents are advised to continue to support them for the sake of legacy documents. Both MENU and DIR consist of one or more LI elements, similar to UL. MENU lists are typically rendered without bullets in a more compact style than UL. You can get the same effect with <UL PLAIN>. DIR lists are used to present lists of items containing up to 20 characters each. Items in a DIR list are arranged in columns. You can get the same effect with <UL PLAIN WRAP=HORIZ>.

Permitted Attributes for the UL Element

ID

An SGML identifier used as the target for hypertext links or for

naming particular elements in associated style sheets. Identifiers are NAME tokens and must be unique within the scope of the current document.

LANG

This is one of the ISO standard language abbreviations, e.g. "en.uk" for the variation of English spoken in the United Kingdom. It can be used by parsers to select language specific choices for quotation marks, ligatures and hypenation rules etc. The language attribute is composed from the two letter language code from ISO 639, optionally followed by a period and a two letter country code from ISO 3166.

CLASS

This a space separated list of SGML NAME tokens and is used to subclass tag names. By convention, the class names are interpreted hierarchically, with the most general class on the left and the most specific on the right, where classes are separated by a period. The CLASS attribute is most commonly used to attach a different style to some element, but it is recommended that where practical class names should be picked on the basis of the element's semantics, as this will permit other uses, such as restricting search through documents by matching on element class names. The conventions for choosing class names

are outside the scope of this specification.

CLEAR

This attribute is common to all block-like elements. When text flows around a figure or table in the margin, you sometimes want to start an element like a header, paragraph or list below the figure rather than alongside it. The CLEAR attribute allows you to move down unconditionally:

clear=left
: move down until left margin is clear

clear=right
: move down until right margin is clear

clear=all
: move down until both margins are clear

Alternatively, you can decide to place the element alongside the figure just so long as there is enough room. The minimum width needed is specified as:

clear="40 en"

move down until there is at least 40 en units free clear="100 pixels"

move down until there is at least 100 pixels free

The style sheet (or browser defaults) may provide default minimum widths for each class of block-like elements.

PLAIN

The presence of this attribute suppresses the display of bullets, e.g. <UL PLAIN>.

SRC

Specifies an image for use as a bullet. The image is specified as a URI. This attribute may appear together with the MD attribute.

MD

Specifies a message digest or cryptographic checksum for the associated graphic specified by the SRC attribute. It is used when you want to be sure that a linked object is indeed the same one that the author intended, and hasn't been modified in any way. For instance, MD="md5:jV2OfH+nnXHU8bnkPAad/mSQlTDZ", which specifies an MD5 checksum encoded as a base64 character string. The MD attribute is generally allowed for all elements which support URI based links.

DINGBAT

Specifies an iconic image for use as a bullet. The icon is specified as an entity name. A list of standard icon entity names for HTML 3.0 is given in an appendix of this specification, e.g. folder is the entity name for an icon denoting a directory or folder.

WRAP

The WRAP attribute is used for multicolumn lists. Use wrap=vert if you want to arrange the list items down the page before wrapping to the next column. Use wrap=horiz if you want to arrange the items across the page (less useful). The user agent is responsible for determining how many columns are appropriate.

COMPACT

The presence of this attribute indicates the user agent should use reduced interitem spacing. In practice, there are several ways to increase the compactness of lists: reduced vertical interitem spacing, smaller font size, or even to avoid line breaks between items. This is best handled through associated style sheets and the class attribute.

LH (List Header)

Permitted Context: Immediately following UL, OL or DL

Content Model: %text

The LH or list header element is used to provide a title for a list. User agents can use this in place of the full list when a mechanism is provided to fold and unfold nested lists.

Permitted Attributes for the LH Element

ID

An SGML identifier used as the target for hypertext links or for naming particular elements in associated style sheets. Identifiers are NAME tokens and must be unique within the scope of the current document.

LANG

This is one of the ISO standard language abbreviations, e.g. "en.uk" for the variation of English spoken in the United Kingdom. It can be used by parsers to select language specific choices for quotation marks, ligatures and hypenation rules etc. The language attribute is composed from the two letter language code from ISO 639, optionally followed by a period and a two letter country code from ISO 3166.

CLASS

This a space separated list of SGML NAME tokens and is used to subclass tag names. By convention, the class names are interpreted hierarchically, with the most general class on the left and the most specific on the right, where classes are separated by a period. The CLASS attribute is most commonly used to attach a different style to some element, but it is recommended that where practical class names should be picked on the basis of the element's semantics, as this will permit other uses, such as restricting search through documents by matching on element class names. The conventions for choosing class names are outside the scope of this specification.

LI (List Item)

Permitted Context: UL or OL

Content Model: %flow

The LI or list item element is used for items in both ordered and unordered lists.

Note: The content model for list items is quite broad, including paragraphs, lists, performatted text, forms, tables, figures and admonishments. Headers are not permitted, although implementors of HTML 3.0 user agents are advised to cater for this possibility in order to handle badly formed legacy documents. If %html.recommended is active, the HTML 3.0 DTD expects you to enclose plain text in a block element such as <P>

Permitted Attributes for the LI Element

ID

An SGML identifier used as the target for hypertext links or for naming particular elements in associated style sheets. Identifiers are NAME tokens and must be unique within the scope of the current document.

LANG

This is one of the ISO standard language abbreviations, e.g. "en.uk" for the variation of English spoken in the United Kingdom. It can be used by parsers to select language specific choices for quotation marks, ligatures and hypenation rules etc. The language attribute is composed from the two letter language code from ISO 639, optionally followed by a period and a two letter country code from ISO 3166.

CLASS

This a space separated list of SGML NAME tokens and is used to subclass tag names. By convention, the class names are interpreted hierarchically, with the most general class on the left and the most specific on the right, where classes are separated by a period. The CLASS attribute is most commonly used to attach a different style to some element, but it is recommended that where practical class names should be picked on the basis of the element's semantics, as this will permit other uses, such as restricting search through documents by matching on element class names. The conventions for choosing class names are outside the scope of this specification.

CLEAR

This attribute is common to all block-like elements. When text flows around a figure or table in the margin, you sometimes want to start the list item below the figure rather than alongside it. The CLEAR attribute allows you to move down unconditionally:

clear=left

move down until left margin is clear

clear=right

move down until right margin is clear

clear=all

move down until both margins are clear

Alternatively, you can decide to place the element alongside the figure just so long as there is enough room. The minimum width needed is specified as:

clear="40 en"

move down until there is at least 40 en units free

clear="100 pixels"

move down until there is at least 100 pixels free

The style sheet (or browser defaults) may provide default minimum widths for each class of block-like elements.

SRC

Specifies an image for use as a bullet. The image is specified as a URI. This attribute may appear together with the MD attribute.

MD

Specifies a message digest or cryptographic checksum for the associated graphic specified by the SRC attribute. It is used when you want to be sure that a linked object is indeed the same one that the author intended, and hasn't been modified in any way. For instance, MD="md5:jV2OfH+nnXHU8bnkPAad/mSQlTDZ", which specifies an MD5 checksum encoded as a base64 character string. The MD attribute is generally allowed for all elements which support URI based links.

DINGBAT

Specifies an iconic image for use as a bullet. The icon is specified as an entity name. A list of standard icon entity names for HTML 3.0 is given in an appendix of this specification, e.g. folder is the entity name for an icon denoting a directory or folder.

SKIP

Increments the sequence number before rendering the element. It is used when headers have been left out of the sequence. For instance, SKIP=3 advances the sequence number past 3 omitted

items.

OL (Ordered List)

Permitted Context: %Body.Content, %flow, %block

Content Model: Optional list header (LH), followed by one or more list items(LI)

An ordered list typically is a numbered list of items. HTML 3.0 gives you the ability to control the sequence number - to continue where the previous list left off, or to start at a particular number. The numbering style is left to associated style sheets, e.g. whether nested lists contribute to a compound item number, e.g. "3.1.5", or whether numbers are rendered as arabic, upper or lower case roman numerals or using the numbering scheme appropriate to the language context.

The opening list tag must be <OL>. It is followed by an optional list header (<LH>caption</LH>) and then by the first list item (<LI>). For example:

```
<OL>
  <LH>Meeting Agenda</LH>
  <LI>Minutes of the last meeting
  <LI>Do we need yet more meetings?
  <LI>Any other business
```

```
    </OL>
```

which could be rendered as:

Meeting Agenda

1. Minutes of the last meeting

2. Do we need yet more meetings?

3. Any other business

Note: Some legacy documents may include headers or plain text before the first LI element. Implementors of HTML 3.0 user agents are advised to cater for this possibility in order to handle badly formed legacy documents.

Permitted Attributes for the OL Element

ID
: An SGML identifier used as the target for hypertext links or for naming particular elements in associated style sheets. Identifiers are NAME tokens and must be unique within the scope of the current document.

LANG

This is one of the ISO standard language abbreviations, e.g. "en.uk" for the variation of English spoken in the United Kingdom. It can be used by parsers to select language specific choices for quotation marks, ligatures and hypenation rules etc. The language attribute is composed from the two letter language code from ISO 639, optionally followed by a period and a two letter country code from ISO 3166.

CLASS

This a space separated list of SGML NAME tokens and is used to subclass tag names. By convention, the class names are interpreted hierarchically, with the most general class on the left and the most specific on the right, where classes are separated by a period. The CLASS attribute is most commonly used to attach a different style to some element, but it is recommended that where practical class names should be picked on the basis of the element's semantics, as this will permit other uses, such as restricting search through documents by matching on element class names. The conventions for choosing class names are outside the scope of this specification.

CLEAR

This attribute is common to all block-like elements. When text flows around a figure or table in the margin, you sometimes want to start an element like a header, paragraph or list below the figure rather than alongside it. The CLEAR attribute allows you to move down unconditionally:

clear=left

move down until left margin is clear

clear=right

move down until right margin is clear

clear=all

move down until both margins are clear

Alternatively, you can decide to place the element alongside the figure just so long as there is enough room. The minimum width needed is specified as:

clear="40 en"

move down until there is at least 40 en units free

clear="100 pixels"

move down until there is at least 100 pixels free

The style sheet (or browser defaults) may provide default minimum widths for each class of block-like elements.

CONTINUE

Don't restart the sequence number, i.e. continue where previous list left off, e.g. <OL CONTINUE>

SEQNUM

Set the starting sequence number for the first item, e.g. <OL SEQNUM=23>

COMPACT

The presence of this attribute indicates the user agent should use reduced interitem spacing. In practice, there are several ways to increase the compactness of lists: reduced vertical interitem spacing, smaller font size, or even to avoid line breaks between items. This is best handled through associated style sheets and the class attribute.

DL - Definition Lists

Permitted Context: %Body.Content, %flow, %block

Content Model: Optional list header(LH), followed by one or more terms(DT) and definitions(DD).

A definition list is a list of terms and corresponding definitions. Definition lists are typically formatted with the term on the left with the definition following on the right or on the next line. The definition text is typically indented with respect to the term.

An alternative format places the term left aligned in a wide margin and the definition on one or more lines to the right of the term. If the DT term does not fit in the DT column (one third of the display area), it may be extended across the page with the DD section moved to the next line, or it may be wrapped onto successive lines of the left hand column.

The opening list tag must be <DL>. It is followed by an optional list header (<LH>caption</LH>) and then by term names (<DT>) and definitions (<DD>). For example:

```
<DL>
<LH>List Header</LH>
```

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

which could be rendered as:

List Header

Term 1
    This is the definition of the first term.

Term 2
    This is the definition of the second term.

The definition list element can take the COMPACT attribute, which suggests that a compact rendering be used, and is appropriate if the list elements are small and/or the entire list is large.

Note: Use the NOTE element when you want to have an indented note. The practice of using <DD> elements without corresponding <DT> elements is deprecated.

Permitted Attributes for the DL Element

ID
    An SGML identifier used as the target for hypertext links or for

naming particular elements in associated style sheets.

Identifiers are NAME tokens and must be unique within the scope of the current document.

LANG

This is one of the ISO standard language abbreviations, e.g. "en.uk" for the variation of English spoken in the United Kingdom. It can be used by parsers to select language specific choices for quotation marks, ligatures and hypenation rules etc. The language attribute is composed from the two letter language code from ISO 639, optionally followed by a period and a two letter country code from ISO 3166.

CLASS

This a space separated list of SGML NAME tokens and is used to subclass tag names. By convention, the class names are interpreted hierarchically, with the most general class on the left and the most specific on the right, where classes are separated by a period. The CLASS attribute is most commonly used to attach a different style to some element, but it is recommended that where practical class names should be picked on the basis of the element's semantics, as this will permit other uses, such as restricting search through documents by matching on element class names. The conventions for choosing class names are outside the scope of this specification.

CLEAR

This attribute is common to all block-like elements. When text flows around a figure or table in the margin, you sometimes want to start an element like a header, paragraph or list below the figure rather than alongside it. The CLEAR attribute allows you to move down unconditionally:

clear=left
: move down until left margin is clear

clear=right
: move down until right margin is clear

clear=all
: move down until both margins are clear

Alternatively, you can decide to place the element alongside the figure just so long as there is enough room. The minimum width needed is specified as:

clear="40 en"
: move down until there is at least 40 en units free

clear="100 pixels"

move down until there is at least 100 pixels free

The style sheet (or browser defaults) may provide default minimum widths for each class of block-like elements.

COMPACT

The presence of this attribute indicates the user agent should use reduced interitem spacing. The COMPACT attribute may also reduce the width of the left-hand (DT) column.

In practice, there are several ways to increase the compactness of lists: reduced vertical interitem spacing, smaller font size, or even to avoid line breaks between items. This is best handled through associated style sheets and the class attribute.

The opening list tag must be DL COMPACT. It must be immediately followed by the first term (DT). For example:

```
<DL compact>
<DT>Term<DD>This is the first definition in compact format.
<DT>Term<DD>This is the second definition in compact format.
</DL>
```

DT - Term Name

Permitted Context: DL

Content Model: %text

The DT tag element specifies a term name, and you can have several terms per DD element.

Note: Term names are restricted to character level markup only, including epmhasis, inline images and footnotes. Paragraph tags and other block-like element such as headers are not permitted, although implementors of HTML 3.0 user agents are advised to cater for this possibility in order to handle badly formed legacy documents.

Permitted Attributes for the DT Element

ID

An SGML identifier used as the target for hypertext links or for naming particular elements in associated style sheets. Identifiers are NAME tokens and must be unique within the scope of the current document.

LANG

This is one of the ISO standard language abbreviations, e.g.

"en.uk" for the variation of English spoken in the United Kingdom. It can be used by parsers to select language specific choices for quotation marks, ligatures and hypenation rules etc. The language attribute is composed from the two letter language code from ISO 639, optionally followed by a period and a two letter country code from ISO 3166.

CLASS

This a space separated list of SGML NAME tokens and is used to subclass tag names. By convention, the class names are interpreted hierarchically, with the most general class on the left and the most specific on the right, where classes are separated by a period. The CLASS attribute is most commonly used to attach a different style to some element, but it is recommended that where practical class names should be picked on the basis of the element's semantics, as this will permit other uses, such as restricting search through documents by matching on element class names. The conventions for choosing class names are outside the scope of this specification.

CLEAR

When text flows around a figure or table in the margin, you sometimes want to start the term name below the figure rather than alongside it. The CLEAR attribute allows you to move down unconditionally:

clear=left

move down until left margin is clear

clear=right

move down until right margin is clear

clear=all

move down until both margins are clear

Alternatively, you can decide to place the element alongside the figure just so long as there is enough room. The minimum width needed is specified as:

clear="40 en"

move down until there is at least 40 en units free

clear="100 pixels"

move down until there is at least 100 pixels free

The style sheet (or browser defaults) may provide default minimum widths for each class of block-like elements.

DD - Term Definition

Permitted Context: DL

Content Model: %flow

The DD tag element specifies a term definition, and follows one or more DT elements.

Note: The content model for term definitions is quite broad, including paragraphs, lists, performatted text, forms, tables, figures and admonishments. Headers are not permitted, although implementors of HTML 3.0 user agents are advised to cater for this possibility in order to handle badly formed legacy documents. If %html.recommended is active, the HTML 3.0 DTD expects you to enclose plain text in a block element such as <P>

Permitted Attributes for the DD Element

ID

An SGML identifier used as the target for hypertext links or for naming particular elements in associated style sheets. Identifiers are NAME tokens and must be unique within the scope of the current document.

LANG

This is one of the ISO standard language abbreviations, e.g. "en.uk" for the variation of English spoken in the United Kingdom. It can be used by parsers to select language specific choices for quotation marks, ligatures and hypenation rules etc. The language attribute is composed from the two letter language code from ISO 639, optionally followed by a period and a two letter country code from ISO 3166.

CLASS

This a space separated list of SGML NAME tokens and is used to subclass tag names. By convention, the class names are interpreted hierarchically, with the most general class on the left and the most specific on the right, where classes are separated by a period. The CLASS attribute is most commonly used to attach a different style to some element, but it is recommended that where practical class names should be picked on the basis of the element's semantics, as this will permit other uses, such as restricting search through documents by matching on element class names. The conventions for choosing class names are outside the scope of this specification.

CLEAR

When text flows around a figure or table in the margin, you sometimes want to start term definition below the figure rather than alongside it. The CLEAR attribute allows you to move down unconditionally:

clear=left

move down until left margin is clear

clear=right

move down until right margin is clear

clear=all

move down until both margins are clear

Alternatively, you can decide to place the element alongside the figure just so long as there is enough room. The minimum width needed is specified as:

clear="40 en"

move down until there is at least 40 en units free

clear="100 pixels"

move down until there is at least 100 pixels free

The style sheet (or browser defaults) may provide default minimum widths for each class of block-like elements.

Figures

Permitted Context: %body.content, %flow, %block

Content Model: Optional OVERLAYs followed by an optional CAPTION, then %body.content and finally an optional CREDIT

The FIG element is used for figures. Subsequent elements will be flowed around the figure if there is sufficient room. This behaviour is disabled when the align attribute is --center-- (the default) or --justify--.

Figure overlays provide for more effective use of caching as small changes to a figure in a subsequent document incur only the penalty of downloading the overlays and not the larger base figure, as the latter is already in the cache.

The figure description text is intended to convey the content of the figure for people with non-graphical user agents, while the figure caption and credit are rendered on both graphical and non-graphical user agents. The FIG element improves on the IMG element by allowing authors to use markup for the description text. The content model allows authors to include headers, which is appropriate when the headers are part of the image data. It also allows graphical hypertext links to be specified in the markup and interpreted by the user agent rather than the server.

The anchor elements in the figure description text play a dual role: Non-graphical user agents show conventional hypertext links, while for graphical user agents, the same anchor elements specify graphical hypertext links, with the SHAPE attribute designating the hotzones. This is designed to simplify the task of authors writing for both audiences. Hopefully, the FIG element will help to combat the tendency for authors to forget about people limited to terminal access or the visually impaired relying on text to speech, as the new element forces you to write description text to define the graphical hypertext links.

For some applications the hotzones are dynamically defined by programs running on the server. HTML 3.0 allows clicks and drags to be passed to the server with the IMAGEMAP attribute. Hotzones may also be specified as part of the graphics data format e.g. as in VRML. Hotzones in the FIG element take precedence over hotzones in the graphics data, which in turn take precedence over passing events to a server imagemap program.

Hotzones in overlay graphics data take precedence over hotzones in figure data. Similarly, the imagemap attribute in overlays takes precedence over the imagemap attribute for the figure. For a group of overlapping overlays the precedence is determined by the order the OVERLAY elements appear within the FIG element. Later overlays take precedence over earlier ones.

Examples

Photographic image with caption and credits:

```
<FIG SRC="nicodamus.jpeg">
  <CAPTION>Ground dweller: <I>Nicodamus bicolor</I>
  builds silk snares</CAPTION>
  <P>A small hairy spider light fleshy red in color with a brown abdomen.
  <CREDIT>J. A. L. Cooke/OSF</CREDIT>
</FIG>
```

Company home page:

```
<FIG SRC="mainmenu.gif">
 <H1>Access HP from Hewlett Packard</H1>
 <P>Select between:
 <UL>
  <LI><A HREF="guide.html" SHAPE="rect 30,200,60,16">Access Guide</A>
  <LI><A HREF="about.html" SHAPE="rect 100,200,50,16">About HP</A>
  <LI><A HREF="guide.html" SHAPE="rect 160,200,30,16">News</A>
  <LI><A HREF="guide.html" SHAPE="rect 200,200,50,16">Products</A>
  <LI><A HREF="guide.html" SHAPE="rect 260,200,80,16">Worldwide Contacts</A>
 </UL>
</FIG>
```

Aerial photograph with map overlay:

```
<FIG SRC="newyork.jpeg">
  <OVERLAY SRC="map.gif">
  <P>New York from the air!
</FIG>
```

Permitted Attributes

ID

An SGML identifier used as the target for hypertext links or for naming particular elements in associated style sheets. Identifiers are NAME tokens and must be unique within the scope of the current document.

LANG

This is one of the ISO standard language abbreviations, e.g. "en.uk" for the variation of English spoken in the United Kingdom. It can be used by parsers to select language specific choices for quotation marks, ligatures and hypenation rules etc. The language attribute is composed from the two letter language code from ISO 639, optionally followed by a period and a two letter country code from ISO 3166.

CLASS

This a space separated list of SGML NAME tokens and is used to

subclass tag names. By convention, the class names are interpreted hierarchically, with the most general class on the left and the most specific on the right, where classes are separated by a period. The CLASS attribute is most commonly used to attach a different style to some element, but it is recommended that where practical class names should be picked on the basis of the element's semantics, as this will permit other uses, such as restricting search through documents by matching on element class names. The conventions for choosing class names are outside the scope of this specification.

CLEAR

When there is already a figure or table in the margin, you sometimes want to position another figure below the figure in the margin rather than alongside it. The CLEAR attribute allows you to move down unconditionally:

clear=left

move down until left margin is clear

clear=right

move down until right margin is clear

clear=all

move down until both margins are clear

Alternatively, you can decide to place the figure alongside the figure in the margin just so long as there is enough room. The minimum width needed is specified as:

clear="40 en"

move down until there is at least 40 en units free

clear="100 pixels"

move down until there is at least 100 pixels free

The style sheet (or browser defaults) may provide default minimum widths for each class of block-like elements.

NOFLOW

The presence of this attribute disables text flow around the figure. It avoids the need to use the CLEAR or NEEDS attributes on the following element.

SRC

Specifies the figure's graphical content. The image is specified as a URI. This attribute may appear together with the MD attribute.

MD

Specifies a message digest or cryptographic checksum for the associated graphic specified by the SRC attribute. It is used when you want to be sure that a linked object is indeed the same one that the author intended, and hasn't been modified in any way. For instance, MD="md5:jV2OfH+nnXHU8bnkPAad/mSQlTDZ", which specifies an MD5 checksum encoded as a base64 character string. The MD attribute is generally allowed for all elements which support URI based links.

ALIGN

Specifies horizontal alignment of the figure:

BLEEDLEFT

Flush left with the left (window) border.

LEFT

Flush left with the left text margin.

CENTER

The figure is centered between the text margins and text flow around the figure is disabled. This is the default setting for ALIGN.

RIGHT

Flush right with the right text margin.

BLEEDRIGHT

Flush right with the right (window) border

JUSTIFY

When applicable the figure should be magnified or reduced to fill the space between the left and right text margins. Text flow around the figure is disabled for align=justify.

WIDTH

Specifies the desired width in pixels or en units (according to the value of the UNITS attribute). User agents may scale the figure image to match this width.

HEIGHT

Specifies the desired height in pixels or en units (according to the value of the UNITS attribute). User agents may scale the figure image to match this height.

UNITS

Specifies the choice of units for width and height. units=pixels (the default) specifies pixels, while units=en specifies en units. The en unit is a typographical unit equal to half the point size.

IMAGEMAP

Specifies a URI for processing image clicks and drags.

Figure Overlays

Permitted Context: start of FIG element

Content Model: Empty!

The OVERLAY element is used to overlay images on top of a base figure. Figure overlays provide for more effective use of caching as small changes to a figure in a subsequent document incur only the penalty of downloading the overlays and not the larger base figure, as the latter is already in the cache. The overlay can be offset from the top left corner of the base image.

Permitted Attributes

SRC

Specifies the overlay image as a URI. This attribute may appear together with the MD attribute.

MD

Specifies a message digest or cryptographic checksum for the associated graphic specified by the SRC attribute. It is used when you want to be sure that a linked object is indeed the same one that the author intended, and hasn't been modified in any way. For instance, MD="md5:jV2OfH+nnXHU8bnkPAad/mSQlTDZ", which specifies an MD5 checksum encoded as a base64 character string. The MD attribute is generally allowed for all elements which support URI based links.

UNITS

Specifies the choice of units for width and height: units=pixels (the default) specifies pixels, while units=en specifies en units (a typographical unit equal to half the point size).

X

The X offset from the top left corner of the base image. X increases to the right, and is given in pixels or en units (according to the value of the UNITS attribute).

Y

The Y offset from the top left corner of the base image. Y increases downwards, and is given in pixels or en units (according to the value of the UNITS attribute).

WIDTH

Specifies the desired width in pixels or en units (according to the value of the UNITS attribute). User agents may scale the figure image to match this width.

HEIGHT

Specifies the desired height in pixels or en units (according to the value of the UNITS attribute). User agents may scale the figure image to match this height.

IMAGEMAP

Specifies a URI for processing image clicks and drags.

Captions

Permitted Context: TABLE or FIG

Content Model: %text

The CAPTION element is used to label a table or figure. Use the align attribute to specify the position of the caption relative to the table/figure. For example:

```
<CAPTION ALIGN=LEFT>The Niagara Falls</CAPTION>
```

--Should we provide separate align and valign attributes for controlling the horizontal and vertical positioning respectively?--

Permitted Attributes

ID

An SGML identifier used as the target for hypertext links or for naming particular elements in associated style sheets. Identifiers are NAME tokens and must be unique within the scope of the current document.

LANG

This is one of the ISO standard language abbreviations, e.g.

"en.uk" for the variation of English spoken in the United Kingdom. It can be used by parsers to select language specific choices for quotation marks, ligatures and hypenation rules etc. The language attribute is composed from the two letter language code from ISO 639, optionally followed by a period and a two letter country code from ISO 3166.

CLASS

This a space separated list of SGML NAME tokens and is used to subclass tag names. By convention, the class names are interpreted hierarchically, with the most general class on the left and the most specific on the right, where classes are separated by a period. The CLASS attribute is most commonly used to attach a different style to some element, but it is recommended that where practical class names should be picked on the basis of the element's semantics, as this will permit other uses, such as restricting search through documents by matching on element class names. The conventions for choosing class names are outside the scope of this specification.

ALIGN

Positioning of the caption relative to the table or figure it labels. The permitted values are: TOP, BOTTOM, LEFT or RIGHT.

Credits

Permitted Context: BQ or FIG

Content Model: %text

The CREDIT element is used to name the source of a block quotation or figure. For example:

```
<CREDIT>The Writer by Richard Wilbur</CREDIT>
```

Permitted Attributes for the CREDIT Element

ID

An SGML identifier used as the target for hypertext links or for naming particular elements in associated style sheets. Identifiers are NAME tokens and must be unique within the scope of the current document.

LANG

This is one of the ISO standard language abbreviations, e.g. "en.uk" for the variation of English spoken in the United Kingdom. It can be used by parsers to select language specific choices for quotation marks, ligatures and hypenation rules etc. The language attribute is composed from the two letter language code from ISO 639, optionally followed by a period and a two letter country code from ISO 3166.

CLASS

This a space separated list of SGML NAME tokens and is used to subclass tag names. By convention, the class names are interpreted hierarchically, with the most general class on the left and the most specific on the right, where classes are separated by a period. The CLASS attribute is most commonly used to attach a different style to some element, but it is recommended that where practical class names should be picked on the basis of the element's semantics, as this will permit other uses, such as restricting search through documents by matching on element class names. The conventions for choosing class names are outside the scope of this specification.

Tables

Permitted Context: %body.content, %flow, %block

Content Model: Optional CAPTION, then one or more table rows (TR)

The HTML table model has been chosen for its simplicity and flexibility. By default the table is automatically sized according to the cell contents and the current window size. The COLSPEC attribute can be used when needed to exert control over column widths, either by setting explicit widths or by specifying relative widths. You can also specify the table width explicitly or as a fraction of the current margins (see WIDTH attribute).

Table start with an optional caption followed one or more rows. Each row is formed by one or more cells, which are differentiated into header and data cells. Cells can be merged across rows and columns, and include attributes assisting rendering to speech and braille, or for exporting table data into databases. The model provides little direct support for control over appearence, for example border styles and margins, as these can be handled via subclassing and associated style sheets.

Tables can contain a wide range of content, such as headers, lists, paragraphs, forms, figures, preformatted text and even nested tables. When the table is flush left or right, subsequent elements will be flowed around the table if there is sufficient room. This behaviour is disabled when the --noflow-- attribute is given or the table align attribute is --center-- (the default), or --justify--.

Example

```
  <TABLE BORDER>
    <CAPTION>A test table with merged cells</CAPTION>
    <TR><TH ROWSPAN=2><TH COLSPAN=2>Average
        <TH ROWSPAN=2>other<BR>category<TH>Misc
    <TR><TH>height<TH>weight
    <TR><TH ALIGN=LEFT>males<TD>1.9<TD>0.003
    <TR><TH ALIGN=LEFT ROWSPAN=2>females<TD>1.7<TD>0.002
  </TABLE>
```

This would be rendered something like:

```
              A test table with merged cells
    /--------------------------------------------------\
    |          |      Average      |  other   |  Misc  |
    |          |-------------------| category |--------|
    |          |  height |  weight |          |        |
    |-----------------------------------------|--------|
    | males    |   1.9   |  0.003  |          |        |
    |-----------------------------------------|--------|
    | females |   1.7   |  0.002  |          |        |

    \---------------------------------------------------/
```

There are several points to note:

* By default, header cells are centered while data cells are flush left. This can be overriden by the ALIGN attribute for the cell; the COLSPEC attribute for the TABLE element; or the ALIGN attribute on the enclosing row's TR element (from the most specific to the least).

* Cells may be empty.

* Cells spanning rows contribute to the column count on each of the spanned rows, but only appear in the markup once (in the first row spanned).

* If the column count for the table is greater than the number of cells for a given row (after including cells for spanned rows), the missing cells are treated as occurring on the right handside of the table, and rendered as empty cells.

* The row count is determined by the TR elements - any rows implied by cells spanning rows beyond this should be ignored.

* The user agent should be able to recover from a missing <TR> tag

prior to the first row as the TH and TC elements can only occur
within the TR element.

* It is invalid to have cells overlap, see below for an example.
In such cases, the rendering is implementation dependent.

An example of an invalid table:

```
<table border>
<tr><tdrowspan=2>1<td>2<td>3<td>4<td>5
<tr><td rowspan=2>6
<tr><td colspan=2>7<td>8
</table>
```

which looks something like:

```
/-------------------\
| 1 | 2 | 3 | 4 | 5 |
|   |---------------|
|   | 6 |   |   |   |    The cells labelled 6 and 7 overlap!
|---|...|-----------|
| 7 :   | 8 |   |   |
\-------------------/
```

Borderless tables are useful for layout purposes as well as their
traditional role for tabular data, for instance with fill-out forms:

name: [John Smith   ]

card number: [4619 693523 20851]

```
    expires: [03] / [97]
  telephone: [212 873 2739      ]
```

This can be represented as a table with one row and two columns. The first column is right aligned, while the second is left aligned.

This example could be marked up as:

```
<table&gt
  <tr valign=baseline>
  <td align=right>
    name:<br>
    card number:<br>
    expires:<br>
    telephone:
  <td align=left>
    <input name="name" size=18><br>
    <input name="cardnum" size=18><br>
    <input name="expires-month" size=2> /
    <input name="expires-year" size=2><br>
    <input name="phone" size=18><br>
</table>
```

The use of such techniques is one of the motivations for using nested tables, where borderless tables are used to layout cell contents for an enclosing table

Hint: You can achieve a similar effect to the above by using decimal alignment and using the DP attribute to set the alignment character to a convenient character, for example:

```
<table>
  <tr align=decimal dp=":">
  <td>
    name: <input name="name" size=18><br>
    card number: <input name="cardnum" size=18><br>
    expires: <input name="expires-month" size=2> /
    <input name="expires-year" size=2><br>
    telephone:<input name="phone" size=18><br>
</table>
```

Each line in the table is then indented so that all the colons are positioned under one another.

Table Sizing Algorithm

The default sizing algorithm requires two passes through the table data. In the first pass, word wrapping is disabled, and the user agent keeps track of the minimum and maximum width of each cell. The maximum width is given by the widest line. As word wrap has been disabled, paragraphs are treated as long lines unless broken by <BR> elements. The minimum width is given by the widest word or image etc. taking into account leading indents and list bullets etc. In other words, if you were to format the cell's content in a window of its own, determine the minimum width you could make the window before things begin to be clipped.

The minimum and maximum cell widths are then used to determine the corresponding minimum and maximum widths for the columns. These in turn, are used to find the minimum and maximum width for the table. Note that cells can contain nested tables, but this doesn't complicate the code significantly. The next step is to assign column widths according to the current window size (more accurately - the width between the left and right margins).

The table borders and intercell margins need to be included in the assignment step. There are three cases:

1. The minimum table width is equal to or wider than the available space. In this case, assign the minimum widths and allow the user to scroll horizontally. For conversion to braille, it will be necessary to replace the cells by references to notes containing their full content. By convention these appear before the table.

2. The maximum table width fits within the available space. In this

case, set the columns to their maximum widths.

3. The maximum width of the table is greater than the available space, but the minimum table width is smaller. In this case, find the difference between the available space and the minimum table width, lets call it --W--. Lets also call --D-- the difference between maximum and minimum width of the table.

   For each column, let --d-- be the the difference between maximum and minimum width of that column. Now set the column's width to the minimum width plus --d-- times --W-- over --D--. This makes columns with lots of text wider than columns with smaller amounts.

This assignment step is then repeated for nested tables. In this case, the width of the enclosing table's cell plays the role of the current window size in the above description. This process is repeated recursively for all nested tables.

If the COLSPEC attribute specifies the column widths explicitly, the user agent can attempt to use these values. If subsequently, one of the cells overflows its column width, the two pass mechanism may be invoked to redraw the table with more appropriate widths. If the attribute specifies relative widths, then the two pass model is always needed.

The column width assignment algorithm is then modified:

* Explicit widths from the COLSPEC attribute should be used when

given, provided they are greater than the minimum column width, otherwise the latter should be used.

* For relative widths, the surplus space --W--, as defined above, is divided up between the columns appropriately, ensuring that each column is given at least its minimum width. If --W-- is zero or negative, column widths should be increased over the minimum width to meet the relative width requirements.

If the table width is specified with the WIDTH attribute, the user agent attempts to set column widths to match. The WIDTH attribute should be disregarded if this results in columns having less than their minimum widths.

Permitted Attributes

ID

An SGML identifier used as the target for hypertext links or for naming particular elements in associated style sheets. Identifiers are NAME tokens and must be unique within the scope of the current document.

LANG

This is one of the ISO standard language abbreviations, e.g.

"en.uk" for the variation of English spoken in the United Kingdom. It can be used by parsers to select language specific choices for quotation marks, ligatures and hypenation rules etc. The language attribute is composed from the two letter language code from ISO 639, optionally followed by a period and a two letter country code from ISO 3166.

CLASS

This a space separated list of SGML NAME tokens and is used to subclass tag names. By convention, the class names are interpreted hierarchically, with the most general class on the left and the most specific on the right, where classes are separated by a period. The CLASS attribute is most commonly used to attach a different style to some element, but it is recommended that where practical class names should be picked on the basis of the element's semantics, as this will permit other uses, such as restricting search through documents by matching on element class names. The conventions for choosing class names are outside the scope of this specification.

CLEAR

When there is a figure or another table in the margin, you sometimes want to start another table below the figure rather than alongside it. The CLEAR attribute allows you to move down unconditionally:

clear=left

move down until left margin is clear

clear=right

move down until right margin is clear

clear=all

move down until both margins are clear

Alternatively, you can decide to place the table alongside the figure just so long as there is enough room. The minimum width needed is specified as:

clear="40 en"

move down until there is at least 40 en units free

clear="100 pixels"

move down until there is at least 100 pixels free

The style sheet (or browser defaults) may provide default minimum widths for each class of block-like elements.

NOFLOW

The presence of this attribute disables text flow around the table. It avoids the need to use the CLEAR or NEEDS attributes on the following element.

ALIGN

Specifies horizontal alignment of the table (--not-- its contents):

BLEEDLEFT

Flush left with the left (window) border.

LEFT

Flush left with the left text margin.

CENTER

The table is centered between the text margins and text flow around the table is disabled. This is the default setting for ALIGN.

RIGHT

Flush right with the right text margin.

BLEEDRIGHT

Flush right with the right (window) border

JUSTIFY

When applicable the table should be sized to fill the space between the left and right text margins. Text flow around the table is disabled for align=justify.

UNITS

Specifies the choice of units for the COLSPEC attribute:

units=en

Specifies en units (a typographical unit equalt to half the point size). This is the default setting and allows user agents to render the table a row at a time without waiting until all of the table's data has been received.

units=relative

Used to set the relative width of columns. The user agent sums the values to determine the proportional width of each column.

units=pixels

The least useful!

A design issue for user agents is how to handle cases where cell contents won't fit into the specified column widths. One approach is to clip the contents to the given column width, another is to resize the columns to fit the contents regardless of the COLSPEC attribute (its best to wait until all of the table's data has been processed before resizing).

COLSPEC

The colspec attribute is a list of column widths and alignment specifications. The columns are listed from left to right with a capital letter followed by a number, e.g. COLSPEC="L20 C8 L40". The letter is L for left, C for center, R for right alignment of cell contents. J is for justification, when feasible, otherwise this is treated in the same way as L for left alignment. D is for decimal alignment, see DP attribute.

Capital letters are required to avoid a particularly common error when a lower case L is confused with a one. Column entries are delimited by one or more space characters.

The number specifies the width in en's, pixels or as a fractional value of the table width, as according to the associated units attribute. This approach is more compact than used with most SGML table models and chosen to simplify hand entry. The width attribute allows you to specify the width of the table in pixels, em units or as a percentage of the space between the current left and right margins.

DP

This specifies the character to be used for the decimal point with the COLSPEC attribute, e.g. dp="." (the default) or dp=",".

The default may be altered by the language context, as set by

the LANG attribute on enclosing elements.

WIDTH

This specifies the width of the table according to the UNITS attribute. If units=relative, the width is taken as a percentage of the width between the current left and right margins. The user agent should disregard this attribute if it would result in columns having less than their minimum widths.

BORDER

This presence of this attribute instructs the user agent to render borders around tables. For instance: <TABLE BORDER>. The precise appearence, along with the size of margins around cells, can be controlled by associated style sheets, or via information in the STYLE element in the document head. Subclassing tables, rows and cells is particularly useful in this regard.

NOWRAP

The NOWRAP attribute is used when you don't want the browser to automatically wrap lines. You can then explicitly specify line breaks in paragrphs using the BR element.

Table Rows

Permitted Context: TABLE

Content Model: Table Cells (TH or TD)

The TR element acts as a container for a row of table cells defined with the TH or TD elements. You can set default horizontal and vertical alignment of cell contents for the row. You also have the ability to disable word wrap for the row, and thereafter use the <BR> element to determine line breaks and hence cell widths.

To assist with formatting tables to paged media, authors can differentiate leading and trailing rows that are to be duplicated when splitting tables across page boundaries. The recommended approach is to subclass rows using the CLASS attribute For example:

```
<TABLE BORDER COLSPEC= --...-->
  <TR CLASS=Header> --header cells ...--
  <TR CLASS=Body>   --body cells ...--
  <TR CLASS=Footer> --footer cells ...--
</TABLE>
```

Paged browsers when splitting a table across a page boundary, can then insert footer rows at the bottom of the current page and header rows at the top of the next page, followed by the remaining body rows, and the footer rows. This is repeated as necessary until all of the body rows have been rendered. Refinements of this scheme can be devised by further subclassing the rows together with an appropriate style sheet.

Permitted Attributes for the TR Element

ID

An SGML identifier used as the target for hypertext links or for naming particular elements in associated style sheets. Identifiers are NAME tokens and must be unique within the scope of the current document.

LANG

This is one of the ISO standard language abbreviations, e.g. "en.uk" for the variation of English spoken in the United Kingdom. It can be used by parsers to select language specific choices for quotation marks, ligatures and hypenation rules etc. The language attribute is composed from the two letter language code from ISO 639, optionally followed by a period and a two letter country code from ISO 3166.

CLASS

This a space separated list of SGML NAME tokens and is used to

subclass tag names. By convention, the class names are

interpreted hierarchically, with the most general class on the

left and the most specific on the right, where classes are separated by a period. The CLASS attribute is most commonly used to attach a different style to some element, but it is recommended that where practical class names should be picked on the basis of the element's semantics, as this will permit other uses, such as restricting search through documents by matching on element class names. The conventions for choosing class names are outside the scope of this specification.

ALIGN

The ALIGN attribute can be used to explicitly specify the horizontal alignment of paragraphs within a table row:

align=left

Paragraphs are rendered flush left.

align=center

Paragraphs are centered.

align=right

Paragraphs are rendered flush right.

align=justify

Text lines are justified where practical, otherwise this gives the same effect as the align=left setting.

align=decimal

Text lines are indented such that the first occurrence of a decimal point on each line are aligned vertically. If a line doesn't contain a decimal point, the line is rendered flush left for data cells and centered for header cells.

Note: By default, header cells are centered while data cells are flush left. This attribute can be used to alter these defaults on a row by row basis. If you are specifying column alignments with the TABLE's COLSPEC attribute, there is no point in also including an ALIGN attribute with the TR element, as the latter will be ignored.

DP

This specifies the character to be used for the decimal point with the ALIGN attribute, e.g. dp="." (the default) or dp=",". The default may be altered by the language context, as set by the LANG attribute on enclosing elements.

VALIGN

The VALIGN attribute can be used to explicitly specify the vertical alignment of material within a table row. It is overridden by the VALIGN attribute on individual cells:

valign=top

The cell contents appear at the top of each cell (the default).

valign=middle

Cell contents are centered vertically in each cell.

valign=bottom

The cell contents appear at the bottom of each cell.

valign=baseline

This is used when you want to ensure that all cells in the row share the same baseline. This constraint only applies to the first text line for each cell.

NOWRAP

The NOWRAP attribute is used when you don't want the browser to automatically wrap lines. You can then explicitly specify line breaks in paragrphs using the BR element.

Table Cells (TH and TD)

Permitted Context: TR

Content Model: %body.content

The TH and TD elements are used for table cells. TH is used for table header cells while TD is used for table data cells. This distinction gives user agents a means to render such cells distinctly, for instance by using a larger or heavier font for header cells. It is also needed when rendering to speech. The CLASS attribute can be used to further differentiate cells, for instance into heads and subheads. This can be used together with style sheets to control the cell border style, and fill color etc.

The horizontal and vertical alignment of cell contents are determined by the ALIGN and VALIGN attributes respectively. In their absence, the alignment will be inherited from the TR element for the row. The COLSPEC attribute of the enclosing TABLE element provides a convenient way of specifying the default horizontal alignment for columns.

The AXIS and AXES attributes can be used when rendering to speech to provide abbreviated names for each cell's headers. Another application is when you want to be able to later process table contents to enter them into a database. Theses attributes are then used to give database field names. The table's class attribute should be used to let the software recognise which tables can be treated in this way.

Note: Disabling word wrap and using the <BR> element in order to control cell widths is discouraged in favor of using the table COLSPEC and WIDTH attributes.

Permitted Attributes for the TH/TD Element

ID

An SGML identifier used as the target for hypertext links or for naming particular elements in associated style sheets. Identifiers are NAME tokens and must be unique within the scope of the current document.

LANG

This is one of the ISO standard language abbreviations, e.g. "en.uk" for the variation of English spoken in the United Kingdom. It can be used by parsers to select language specific choices for quotation marks, ligatures and hypenation rules etc. The language attribute is composed from the two letter language code from ISO 639, optionally followed by a period and a two letter country code from ISO 3166.

CLASS

This a space separated list of SGML NAME tokens and is used to subclass tag names. By convention, the class names are interpreted hierarchically, with the most general class on the left and the most specific on the right, where classes are separated by a period. The CLASS attribute is most commonly used to attach a different style to some element, but it is recommended that where practical class names should be picked on the basis of the element's semantics, as this will permit other uses, such as restricting search through documents by matching on element class names. The conventions for choosing class names are outside the scope of this specification.

COLSPAN

The number of columns spanned by this cell. This allows you to merge cells across columns. It defaults to 1 (one).

ROWSPAN

The number of rows spanned by this cell. This allows you to merge cells across rows. It defaults to 1 (one).

ALIGN

The ALIGN attribute can be used to explicitly specify the horizontal alignment of paragraphs within a table row:

align=left

Paragraphs are rendered flush left. This is the default for data cells (TD).

align=center

Paragraphs are centered. This is the default for header cells (TH).

align=right

Paragraphs are rendered flush right.

align=justify

Text lines are justified where practical, otherwise this gives the same effect as the align=left setting.

align=decimal

Text lines are indented such that the first occurrence of a decimal point on each line are aligned vertically. If a line doesn't contain a decimal point, the line is rendered flush left for data cells and centered for header cells.

Note: In the absence of the ALIGN attribute, the default is overridden by the presence of an ALIGN attribute on the parent TR element, or by the COLSPEC attribute on the TABLE element. The COLSPEC attribute takes precedence over the TR element though!

DP

This specifies the character to be used for the decimal point with the ALIGN attribute, e.g. dp="." (the default) or dp=",". The default may be altered by the language context, as set by the LANG attribute on enclosing elements.

VALIGN

The VALIGN attribute can be used to explicitly specify the vertical alignment of material within a table cell:

valign=top

The cell contents appear at the top of each cell (the default).

valign=middle

Cell contents are centered vertically in each cell.

valign=bottom

The cell contents appear at the bottom of each cell.

valign=baseline

This is used when you want to ensure that all cells in the row with valign=baseline share the same baseline. This constraint only applies to the first text line for each cell.

Note: In the absence of the VALIGN attribute, the default can be
overridden by the presence of a VALIGN attribute on the parent
TR element.

NOWRAP

The NOWRAP attribute is used when you don't want the browser to
automatically wrap lines. You can then explicitly specify line
breaks in paragraphs using the BR element.

AXIS

This defines an abbreviated name for a header cell, which can be
used when rendering to speech. It defaults to the cell's
content.

AXES

This is a comma separated list of axis names which together
identify the row and column headers that pertain to this cell.
It is used when rendering to speech to identify the cell's
position in the table. If missing the user agent can try to
follow up columns and left along rows (right for some languages)
to find the corresponding header cells.

Note: a subheader cell may include both attributes - using AXIS
to name itself and AXES to name the parent header cell. When

data cells refer to header cells with both attributes, the parent header cells are found by following back the head-subhead relationships.

HTML Math

Permitted Context: %text

Content Model: %math

The <MATH> element is used to include math expressions in the current line. HTML math is powerful enough to describe the range of math expressions you can create in common word processing packages, as well as being suitable for rendering to speech. When rendering to fixed pitch text-only media, simple text graphics can be used for math symbols such as the integration sign, while other symbols can be rendered using their entity names. The SGML SHORTREF capability is used to provide abbreviations for hidden brackets, subscripts and superscripts.

The design of HTML math owes a lot to LaTeX's math mode, which has been found to be effective for a wide variety of mathematical typesetting. Where practical, HTML math uses tag names matching LaTeX commands, e.g. ATOP, CHOOSE and SQRT act in the same way as their LaTeX namesakes. Of course, SGML and LaTeX have quite different syntactical conventions. As a result, HTML math uses the ISO entity names for symbols rather than the TeX names. In LaTeX, the character command ^ sets the next character as an exponent, while the character command _ sets it as an index. If the exponent or index contains more than one character then the group of characters must be enclosed in curly brackets { }. This syntax is inappropriate for SGML, so HTML math instead treats _ and ^ as shortref characters for the SUB and SUP elements which are used for indices and exponents, respectively.

--I can't find the ISO entity names for the _ and ^ chararacters!--

HTML math has been designed to be both concise and comparatively easy to read. In practice, formulae will be a little longer than in LaTeX, but much shorter than with other math proposals for SGML, for instance EuroMath or ISO 12083. This simplification has been achieved through the power of the BOX element, which replaces many elements in other proposals, as well as the simple conventions for binding the SUB and SUP elements and their use as generic raising and lowering operators. HTML math differentiates terms e.g. binary operators, variables, constants, integral signs, delimiters and so on. This simplifies rendering and reflects the assumptions adopted by LaTeX. It further allows the same raising and lowering operators to be used for many different roles according to the term they apply to. HTML math doesn't provide direct support for multi-line equations, as this can be effectively handled by combining math with the TABLE element.

Example - the integral from a to b of f(x) over 1+x

```
<MATH>&int;_a_^b^{f(x)<over>1+x} dx</MATH>
```

which can be rendered on a fixed pitch text-only medium as:

```
b
/   f(x)
| ------- dx
/  1 + x
a
```

The example uses { and } as shortrefs for <BOX> and </BOX> respectively. This is used for invisible brackets, stretchy delimiters and integral signs, and placing one thing over another. The shortref characters "_" and "^" are used for subscripts and superscripts respectively.

HTML math follows general practice in mathematical typesetting by rendering functions, numbers and other constants in an upright font, while variables are rendered in an italic font. You can set particular terms in a bold face, and for chemical formulae, you can force the use of an upright font. Limits for symbols like the integral and summation signs are placed directly above (below) the symbol or to the immediate right depending on the symbol.

Spacing between constants, variables and operators is determined automatically. Additional spacing can be inserted with entities such as &thinsp; &sp; and &quadsp;. White space in the markup is used only to delimit adjacent variables or constants. You don't need spaces before or after binary operators or other special symbols, as these are recognised by the HTML math tokeniser. White space can be useful, though, for increased legibility while authoring.

--I need to check on the ISO entity names for spacing!--

Math Markup

The following elements are permitted within MATH elements:

BOX

Used for hidden brackets, stretchy delimiters, and placing one expression over another (e.g. numerators and denominators).

SUB, SUP

Subscripts and superscripts. Also used for limits.

ABOVE

Used to draw an arrow, line or symbol above an expression.

BELOW

Used to draw an arrow, line or symbol below an expression.

VEC, BAR, DOT, DDOT, HAT, TILDE

These are convenience tags for common accents as an alternative to using ABOVE.

SQRT, ROOT

For square roots and other roots of an expression.

ARRAY

For matrices and other kinds of arrays.

TEXT

Used to include a short piece of text within a math element, and often combined with SUB or SUP elements.

B, T, BT

These elements are used override the default rendering. B renders the enclosed expression in an bold face. T designates a term to be rendered in an upright font, while BT designates a term to be rendered in a bold upright font. The class attribute can be used to describe the kind of term, e.g. vector, tensor, or matrix.

HTML Math Entities

* Functions

* Operators

* Continuation dots

* Greek letters

* Relations

* Accents, arrows and pointers

* Delimiters

* Other symbols.

* Spacing entities.

Rendering HTML Math

The expression is rendered in three steps:

1. The first step recursively parses expressions building up a matching hierarchy of data structures (with bounding boxes) corresponding to sequences of nested expressions. The math tokeniser needs to be able to distinguish constants, variables, functions, operators, delimiters, and special symbols such as integrals, which can take limits and may be stretchy.

2. The next step sets the size of the innermost expressions based on the size of available fonts. If possible subscript and superscript expressions should be set in a smaller font. The

size and relative positioning of neighboring and enclosing expressions is then propagated up the hierarchy from the innermost outwards, as the procedure stack formed in step (1) unwinds.

3. The final step is to render the hierarchy of expressions to the output medium. This is now straight forward as all the positioning and sizes of special symbols and text strings are now fixed.

Note: In practice, only a limited range of font sizes are suitable, as a result, deeply nested expressions like continued fractions can't use ever smaller fonts. This is simply handled by a parameter to the --ParseExpression-- routine that sets the font size to be used for that expression. ParseExpression is called recursively for nested expressions and uses the next smaller font until it bottoms out with the smallest font available. The size parameter corresponds to an enumeration of the available font sizes.

Permitted Attributes

ID

An SGML identifier used as the target for hypertext links or for naming particular elements in associated style sheets. Identifiers are NAME tokens and must be unique within the scope of the current document.

CLASS

This a space separated list of SGML NAME tokens and is used to subclass tag names. By convention, the class names are interpreted hierarchically, with the most general class on the left and the most specific on the right, where classes are separated by a period.

For the MATH element, CLASS can be used to describe the kind of math expression involved. This can be used to alter the way formulae are rendered, and to support exporting the expression to symbolic math software. The class "chem" is useful for chemical formulae which use an upright font for variables rather than the default italic font. For example:

```
<math class=chem> Fe_2_^2+^Cr_2_O_4_</math>
```

which is rendered as $\mathrm{Fe}_2^{2+}\mathrm{Cr}_2\mathrm{O}_4$

Otherwise, the conventions for choosing class names are outside the scope of this specification.

BOX

The presence of this attribute causes the user agent to draw a rectangular box around the formulae.

The BOX element

Permitted Context: %math

Content Model: %math?, LEFT?, expression, RIGHT?, %math?

where expression is %math or %math --tag-- %math, see below.

The BOX element is used for a variety of purposes:

* As invisible brackets for disambiguating expressions.

* For placing numerators over denominators, with or without dividing lines, corresponding to LaTeX's --frac, atop-- and --choose-- commands.

* For placing delimiters before, and/or after an expression, which grow to match the height of the expression. This corresponds to LaTeX's --left-- and --right-- commands.

* For stretchy integral and related signs that grow to match the height of the integrand. This goes beyond LaTeX and avoids the need for different symbols for different sizes of integral signs etc.

The SHORTREF characters { and } are used as abbreviations for the start and end tags for BOX elements. Use the SGML entities &lcub; and &rcub; respectively when you need to use these characters literally.

Short fractions, are best represented using the slash character / as in <math>(n + m)/2</math> for (n + m)/2. For more complicated fractions you can use the BOX element with the OVER tag, as in:

```
     1
   -----         {1<over>x + y}
   x + y


   x - y
 ---------       {x - y<over>1 + {a - b<over>a + b}}
     a - b
 1 + -----
     a + b
```

Use the ATOP tag when you want to place one thing above another, but without the dividing line. With the CHOOSE tag, the expressions are also enclosed within round brackets, for instance:

```
   a - b
                      {a - b<atop>a + b}
   a + b
```

(n + 1)   (n)   (n)   {n+1<choose>k}

```
 (     ) = (   ) + (     )          = {n<choose>k}
 (  k  )   ( k )   (k - 1)            + {n<choose>k-1}
```

The BOX element can be used with the LEFT and RIGHT tags for delimiters that stretch to match the size of the enclosed expression. You are free to give only one or both of the delimiters, and you can use different symbols for the left and right. Left delimiters should be given before the LEFT tag, while right delimiters should be given following the RIGHT tag, for instance:

```
           ( 1 + x )       f(x) = {(<left>
      f(x) = ( ----- )               1+x<over> sin x
           ( sin x )                <right>)}
```

For curly brackets you will need to use the &lcub; and &rcub; entities for { and } respectively. The SIZE attribute can be used to get oversized delimiters, for instance:

```
        2 |        |
   omega  |        |      {&omega;^2^<over>c^2^}
   -----  |<j| mu |i>|      <box size=huge> | <left>
      2   |     z   |           &lt;j | &mu;_z_ | i&gt;
     c    |        |        <right> | </box>
```

The full tag name for the BOX element is needed above in order to specify a value for the SIZE attribute. The BOX element is also useful for stretching integral signs to match the height of the integrand. The integral and its limits appear in the position of the left delimiter for the BOX element. You can also include multiple integrals, for instance:

```
 inf inf
  /  /  f(x, y)          {&int;_0_^&inf;^ &int;_0_^&inf;^<left>
  |  | --------- dx dy        f(x,y)<over>x^2^ + y^2^} dx dy
  |  |   2     2
  /  /  x  +  y
  0   0
```

Permitted Attributes for the BOX element

SIZE

This used to obtain oversized delimiters. The permitted values are --normal-- (the default), --medium--, --large-- and --huge--.

Note: there are no attributes for the LEFT, RIGHT, OVER, ATOP and CHOOSE tags.

SUB and SUP

Permitted Context: %math

Content Model: %math

The SUB and SUP elements are used for subscripts and superscripts respectively, and are preferably rendered in reduced font. For integral and related signs the SUB and SUP elements are used for the lower and upper limits, for instance:

```
  &sum;<sub>n = 0</sub><sup>&inf;</sup>
```

which renders to (within limits of ascii art):

```
 infinity
   ---
   \
   /
   ---
  n = 0
```

The SGML SHORTREF characters "_" and "^" can be used as abbreviations. The above example is abbreviated to:

```
    &sum;_n = 0_^&inf;^
```

--Need to give entity names for when you need the _ and ^ characters!--

You have to revert to the full form of the tags when superscripting a superscript or subscripting a subscript, as in:

```
    X<sub>a<sub>i</sub></sub>  which is  X
                                          a
                                           i
```

You can also use superscripts to place expressions or words over binary operators, for instance in the following chemical reaction:

```
    4LiH + AlCl_3_ &rarr;^ether^ LiAlH_4_ + 3LiCl
```

which renders as:

```
                  ether
     4LiH + AlCl  ------>  LiAlH  +  3LiCl
         3                      4
```

The superscript is centered over the operator, and when feasible the operator stretched to match the width of the superscript.

Note: Dont' use the shortrefs for SUB and SUP in normal text - they

only work within MATH elements!

Placement of subscripts and superscripts

Some times you will want to place superscripts to the left of a term. The simplest way to do this is with shortrefs, e.g.

```
 _4_^9^Be + _2^4^He &rarr; _6_^12^C + _0_^1^n
```

which renders to (within limits of ascii art):

```
  9    4      12   1
   Be +  He  -->  C +  n
  4    2       6   0
```

Subscripts and superscripts immediately before a term are rendered to the left of the term, where as subscripts and superscripts immediately after a term are rendered on its right. Opportunities for confusion arise when scripts for a preceding term are followed by scripts for another term. The simplest solution is to use whitespace to disambiguate the situation. For instance:

```
               b                   b
 X_a_ ^b^Y  is  X   Y   while  X_a_^b^Y   is  X   Y
              a                    a
```

but both X^a^ ^b^Y and X^a^^b^Y are $X^{a}\;{}^{b}Y$

The last case is disambiguated because the two superscript elements can't both apply to the preceding term, as they can't both be drawn in the same position. The same would apply to two subscript elements. If in any doubt, its wise to insert a space to make your meaning clear! If you prefer, you can use curly brackets, as in: {X_a_}{^b^Y}. These brackets are shortrefs for the BOX element and don't appear when the expression is finally rendered.

The same rules apply when you use the full form of the SUB and SUP elements. The ALIGN attribute can be used to override the default position to render the script, which is normally taken from whether the script element precedes or follows the term to which it applies. For instance:

```
X<sub align=right>a</sub> <sup align=left>b</sup>Y
X<sub align=right>a</sub>Y<sup align=left>b</sup>
```

both render to $X_{a}\;{}^{b}Y$

while X<sub align=right>a</sub><sup align=left>b</sup>Y b

renders to X Y

a

In the last case, the SUP element act as a prefix superscript to X as there isn't a delimiter between it and the preceding SUB element. If in doubt, you should use brackets or whitespace to disambiguate the binding.

Permitted Attributes

The ID, LANG and CLASS attributes for SUB and SUP are not used within MATH elements.

ALIGN

Subscripts and superscripts are normally placed to the right of the term to which they apply, while limits are normally placed above (or below) the symbol they apply to. The ALIGN attribute can be used to override the default positioning.

align=left

The script (or limit) is placed to the left of the term.

align=center

The script is centered on the term and placed below it for subscripts, and above it (for superscipts).

align=right

The scripts are placed to the right of the term.

For example, you can force limits on integral signs to appear on the right rather than centered on the integral sign:

```
&int;<sub align=right>0</sub><sup align=right>&inf;</sup>
```

Note: The ALIGN attribute should be ignored by user agents for the SUB and SUP elements except within MATH elements.

The ABOVE element

Permitted Context: %math

Content Model: %math

The <ABOVE> element is used to draw a line, arrow, curly bracket, or accent --above-- the expression enclosed by this element. Stretchy symbols should be stretched to match the width of the enclosed expression. For example:

<above>X + Y</above> giving $\overline{X + Y}$

<above sym=equals>X + Y</above> giving $\overset{=====}{X + Y}$

You can also place an expression centered above the line or arrow with the SUP element or its shortref form, for example:

```
<above sym=cub>n(n - 1)(n - 2)&dots;(n - m + 1)</above>
<sup><text>total of m factors</text></sup>
```

which would be rendered as (within limits of ascii art):

```
      total of m factors
 /---------------^-------------\
 n(n - 1)(n - 2) ... (n - m + 1)
```

Permitted Attributes

SYM

An entity name for a symbol, e.g. --cub-- for a curly bracket (brace). Defaults to --line--. The other choices are: --larr-- (left arrow), --rarr-- (right arrow), --hat-- and --tilde--.

Note: Don't include the & prefix, so <above sym="&rarr;"> is wrong!

The BELOW element

Permitted Context: %math

Content Model: %math

The <BELOW> element is used to draw a line, arrow, or curly bracket --below-- the expression enclosed by this element. For example:

```
    <below>X + Y</below>    giving   X + Y
                                     ─────

    <below sym=rarr>X + Y</below>   giving   X + Y
                                             ---->
```

You can also place an expression centered below the line or arrow with the SUP element or its shortref form, for example:

```
<above sym=cub>n(n - 1)(n - 2)&dots;(n - m + 1)</above>
<sup><text>total of m factors</text></sup>
```

which would be rendered as (within limits of ascii art):

```
    n(n - 1)(n - 2) ... (n - m + 1)

    \---------------v-------------/

           total of m factors
```

--I can't find the ISO entity names for under/over curly brackets!--

Permitted Attributes

SYM

An entity name for a stretchy symbol, e.g. --cub-- for a curly bracket (brace). Defaults to --line--. The other choices are: --larr-- (left arrow), --rarr-- (right arrow), --hat-- and --tilde--.

Note: Don't include the & prefix, so <below sym="&rarr;"> is wrong!

VEC, BAR, DOT, DDOT, HAT and TILDE

Permitted Context: %math

Content Model: %math

These elements place an accent above the term enclosed by the element. --VEC-- draws a right arrow above the term; --BAR-- draws a line; --DOT-- and --DDOT-- draw a single and double dot respectively; --HAT-- and --TILDE-- draw the corresponding character above the term, for example:

```
                 ^
   <HAT>X</HAT>   giving   X

                     ~
   <TILDE>X</TILDE>   giving   X
```

Note: there are no attributes for these elements.

SQRT

Permitted Context: %math

Content Model: %math

This draws a square root sign around the contents, for example the square root of 1 + x is expressed as:

```
<SQRT>1 + x</SQRT>
```

and possibly rendered as: --(this is the best I can do with ascii art!)--

```
   ----------
  / 1 + x
 v
```

The SQRT element has no attributes.

ROOT

Permitted Context: %math

Content Model: %math, OF, %math

This allows you to specify arbitary roots of an expression. The radix comes first, and is separated from the radicand by the <OF> tag. For example the cube root of 1 + x is expressed as:

```
    <ROOT>3<OF>1 + x</ROOT>
```

and possibly rendered as:

```
      --------
    3/ 1 + x
    v
```

The ROOT element has no attributes.

The ARRAY element

Permitted Context: %math

Content Model: one or more ROWs, each containing one or more ITEMs

The <ARRAY> element is used for LaTeX-like arrays. It can only be used within MATH elements. For example:

$$\begin{array}{cccc} a_{11} & a_{12} & \ldots & a_{1n} \\ : & : & \ldots & : \\ a_{n1} & a_{n2} & \ldots & a_{nn} \end{array}$$

This is represented in HTML math by:

```
<array>
 <row><item>a_11_<item>a_12_<item>&cdots;<item>a_1n_
 <row><item>&vdots;<item>&vdots;<item>&ddots;<item>&vdots;
 <row><item>a_n1_<item>a_n2_<item>&cdots;<item>a_nn_
</array>
```

You can specify the "+", "-" or "=" characters as column separators:

$$a_{11}x_1 + a_{12}x_2 + ... + a_{1n}x_n = b_1$$

$$a_{22}x_1 + a_{22}x_2 + ... + a_{2n}x_n = b_2$$

....

$$a_{n2}x_1 + a_{n2}x_2 + ... + a_{nn}x_n = b_n$$

This is represented by:

```
<array coldef="C+C+C+C=C">
 <row><item>a_11_x_1_<item>a_12_x_2_<item>&cdots;<item>a_1n_x_n_<item>b_1_
 <row><item>a_21_x_1_<item>a_22_x_2_<item>&cdots;<item>a_2n_x_n_<item>b_2_
 <row><item colspan=5>&dotfill;
 <row><item>a_n1_x_1_<item>a_n2_x_2_<item>&cdots;<item>a_nn_x_n_<item>b_n_
</array>
```

The number of rows and colums is left to the parser to work out for itself by counting the number of ROW elements, and the number of

ITEMs on each row. This is slightly complicated by the possibility that some items may span several rows and columns. If a row has less than the expected number of items after accounting for merged items, the missing items are assumed to be on the righthandside, and should be treated as empty. The row count should be based on the ROW elements, and additional rows, implied by items spanning rows beyond the end of the array, should be ignored.

Permitted Attributes

ALIGN

By default, arrays are vertically positioned so that preceding and following expressions are aligned with the mid point of the array. The ALIGN attribute can take on of the following values:

TOP

The top row of the array is aligned with the same baseline as the preceding or following expression.

MIDDLE

The middle row of the array is aligned with the same baseline as the preceding or following expression. This is the default. If there are an even number of rows, the midpoint of the array is used instead.

BOTTOM

The bottom row of the array is aligned with the same baseline as the preceding or following expression.

COLDEF

By default the columns are centered. This attribute can be used to specify the horizontal alignment for each column with character string formed by one capital letter per column, with L for left, C for center and R to right alignment, e.g. "LLCR" for a 4 column table.

The COLDEF attribute can also be used to place a "+", "-" or "=" between the columns, e.g. coldef="C+C+C+C=C". Whitespace within the COLDEF attribute is ignored.

LDELIM

An entity or character for the left delimiter, e.g. ldelim="[" for a left square bracket or ldelim="|". The default is no delimiter.

RDELIM

An entity name or character for the right delimiter, e.g. rdelim="}" or rdelim="|". The default is no delimiter.

LABELS

The presence of this attribute has the same effect as TeX's bordermatrix command. The first row and column are separated from the rest of the array.

Example of a labelled array:

```
     i    j    k


 a  (X     X     X  )
    ( 11    21    31)    <array ldelim="(" rdelim=")" labels>
    (               )    <row><item>dummy<item>i<item>j<item>k
 b  (X     X     X  )     <row><item>a<item>X_11_<item>X_21_<item>X_31_
    ( 12    22    32)     <row><item>b<item>X_12_<item>X_22_<item>X_32_
    (               )    <row><item>c<item>X_31_<item>X_32_<item>X_33_
 c  (X     X     X  )    </array>
    ( 13    23    33)
```

Note: An item is always required for the first item of the first

row, although its contents will be ignored for labelled arrays. This

has been emphasized above with the value "dummy".

The ROW element

Permitted Context: ARRAY

Content Model: one or more ITEMs

The <ROW> element is used for rows of items within the ARRAY element. It has no permitted attributes. The end tag </ROW> can always be left out.

The ITEM element

Permitted Context: ROW

Content Model: %math

The <ITEM> element is used for items within a row of the ARRAY element. The end tag </ITEM> can always be left out.

Permitted Attributes for ITEM

ALIGN

By default expressions in items are centered horizontally. This can be altered by the COLDEF attribute on the parent ARRAY element, and overridden on a per item basis with the ALIGN attribute. The permitted values are: LEFT, CENTER or RIGHT.

COLSPAN

This can be used to provide a single item which spans several columns. The attribute value is a positive integer and defaults to one.

ROWSPAN

This can be used to provide a single item which spans several rows. The attribute value is a positive integer and defaults to one. It can be used together with the COLSPAN attribute.

TEXT

Permitted Context: %math

Content Model: PCDATA

This element is used to include a few words of text within MATH elements. It avoids the need to separate words with spacing entities such as &sp; that are otherwise needed to provide adequate spacing. The text is rendered literally, and may include entities for accented characters etc.

There are no attributes for this element.

B, T and BT

Permitted Context: %math

Content Model: %math

These elements are used, when feasible, to alter the default fonts used to render variables and constants. Numbers, operators, delimiters and other symbols are unaffected. B renders its contents in bold, while T renders its contents in an upright font rather than an italic font. These can be nested to combine the effects, for a bold upright font. The BT element allows you to write <BT>--term--</BT> rather from having to write: <B><T>--term--</T></B>.

Permitted Attributes

CLASS

This a space separated list of SGML NAME tokens and is used to subclass tag names. By convention, the class names are interpreted hierarchically, with the most general class on the left and the most specific on the right, where classes are separated by a period.

By using CLASS to describe the term as a --vector--, --tensor--

or --matrix-- etc. user agents can do a better job when rendering to non-visual media. Further conventions for term class names are not part of this specification.

Note: Don't use B for chemical formulae. These should be handled by subclassing the MATH element with class=chem, for instance: <MATH CLASS=CHEM>

Horizontal Rules

Permitted Context: %Body.Content

Content Model: Empty!

The <HR> element is used for horizontal rules that act as dividers between sections. The SRC attribute can be used to designate a custom graphic, otherwise subclass HR with the CLASS attribute and specify the appropriate rendering with an associated style sheet.

Permitted Attributes

ID

An SGML identifier used as the target for hypertext links or for naming particular elements in associated style sheets.
Identifiers are NAME tokens and must be unique within the scope of the current document.

CLASS

This a space separated list of SGML NAME tokens and is used to subclass tag names. For instance, <H2 CLASS=Section> defines a level 2 header that acts as a section header. By convention, the class names are interpreted hierarchically, with the most
general class on the left and the most specific on the right,

where classes are separated by a period. The CLASS attribute is most commonly used to attach a different style to some element, but it is recommended that where practical class names should be picked on the basis of the element's semantics, as this will permit other uses, such as restricting search through documents by matching on element class names. The conventions for choosing class names are outside the scope of this specification.

CLEAR

This attribute is common to all block-like elements. When text flows around a figure or table in the margin, you sometimes want to position the rule below the figure rather than alongside it. The CLEAR attribute allows you to move down unconditionally:

clear=left

move down until left margin is clear

clear=right

move down until right margin is clear

clear=all

move down until both margins are clear

Alternatively, you can decide to place the rule alongside the figure just so long as there is enough room. The minimum width needed is specified as:

clear="40 en"

move down until there is at least 40 en units free

clear="100 pixels"

move down until there is at least 100 pixels free

The style sheet (or browser defaults) may provide default minimum widths for each class of block-like elements.

SRC

Specifies a custom image for the rule. The image is specified as a URI. This attribute may appear together with the MD attribute.

MD

Specifies a message digest or cryptographic checksum for the associated graphic specified by the SRC attribute. It is used when you want to be sure that a linked object is indeed the same one that the author intended, and hasn't been modified in any way. For instance, MD="md5:jV2OfH+nnXHU8bnkPAad/mSQlTDZ", which specifies an MD5 checksum encoded as a base64 character string. The MD attribute is generally allowed for all elements which support URI based links.

Preformatted Text (PRE)

Permitted Context: %Body.Content, %flow, %block

Content Model: subset of %text (see DTD for details)

Preformatted text between the start and end PRE tag is rendered using a fixed with font, in addition whitespace characters are treated literally. The spacing and line breaks are rendered directly, unlike other elements, for which repeated whitespace chararacters are collapsed to a single space character and line breaks introduced automatically.

* Line breaks within the text are rendered as a move to the beginning of the next line. The exceptions are line breaks immediately following the starting PRE tag or immediately preceding the ending PRE tag, which should be ignored.

* The <P> tag should be avoided, but for robustness, user agents are recommended to treat these tags as line breaks.

* Anchor elements, and character highlighting elements may be used.

* FORM elements may be included, and the fixed width font

exploited to control layout (the TAB or TABLE elements give
similar control for normal text though).

* Block-like elements such as headers, lists, FIG and TABLES
should be avoided.

* The horizontal tab character (encoded in US ASCII and ISO 8859-1
as decimal 9) should be interpreted as the smallest nonzero
number of spaces which will leave the number of characters so
far on the line as a multiple of 8. Its use is deprecated!

For example, a verse from Shelley (To a Skylark):

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

which is rendered as:

```
       Higher still and higher
         From the earth thou springest
       Like a cloud of fire;
         The blue deep thou wingest,
```

And singing still dost soar, and soaring ever singest.

Permitted Attributes

ID

An SGML identifier used as the target for hypertext links or for naming particular elements in associated style sheets. Identifiers are NAME tokens and must be unique within the scope of the current document.

LANG

This is one of the ISO standard language abbreviations, e.g. "en.uk" for the variation of English spoken in the United Kingdom. It can be used by parsers to select language specific choices for quotation marks, ligatures and hypenation rules etc. The language attribute is composed from the two letter language code from ISO 639, optionally followed by a period and a two letter country code from ISO 3166.

CLASS

This a space separated list of SGML NAME tokens and is used to subclass tag names. By convention, the class names are interpreted hierarchically, with the most general class on the left and the most specific on the right, where classes are separated by a period. The CLASS attribute is most commonly used to attach a different style to some element, but it is recommended that where practical class names should be picked on

the basis of the element's semantics, as this will permit other
uses, such as restricting search through documents by matching
on element class names. The conventions for choosing class names
are outside the scope of this specification.

CLEAR

This attribute is common to all block-like elements. When text
flows around a figure or table in the margin, you sometimes want
to start the preformatted text below the figure rather than
alongside it. The CLEAR attribute allows you to move down unconditionally:

clear=left	move down until left margin is clear
clear=right	move down until right margin is clear
clear=all	move down until both margins are clear

Alternatively, you can decide to place the element alongside the
figure just so long as there is enough room. The minimum width
needed is specified as:

clear="40 en"

move down until there is at least 40 en units free

clear="100 pixels"

move down until there is at least 100 pixels free

The style sheet (or browser defaults) may provide default minimum widths for each class of block-like elements.

WIDTH

This is optionally used to specify a width as a number of characters to try and display within the current window width. The user agent can exploit this suggestion to select an appropriate font size. The default is a width of 80 characters. Where the WIDTH attribute is supported, widths of 40, 80 and 132 characters should be presented optimally, with other widths being rounded up.

--Can't we get rid of this obsolete nonsense? How many browsers support the WIDTH attribute anyway? --

Admonishments

Permitted Context: %body.content, %flow, %block

Content Model: %body.content

The NOTE element is designed for use as admonishments such as notes, cautions or warnings, as commonly used in technical documentation. The CLASS attribute specifies the type of the element and is typically associated with different graphics such as a road traffic warning sign. The graphic can be customized with the SRC attribute.

Example:

```
<NOTE CLASS=WARNING>Please check with the local weather
service before starting your climb. The mountain weather
is subject to rapid deterioration. It is essential to
carry a good map and compass.</NOTE>
```

The class names: NOTE, CAUTION and WARNING are recommended for standard admonishments. In the absence of the CLASS attribute, a NOTE element is typically rendered indented, without an accompanying graphic.

Permitted Attributes

ID

An SGML identifier used as the target for hypertext links or for naming particular elements in associated style sheets. Identifiers are NAME tokens and must be unique within the scope of the current document.

LANG

This is one of the ISO standard language abbreviations, e.g. "en.uk" for the variation of English spoken in the United Kingdom. It can be used by parsers to select language specific choices for quotation marks, ligatures and hypenation rules etc. The language attribute is composed from the two letter language code from ISO 639, optionally followed by a period and a two letter country code from ISO 3166.

CLASS

This a space separated list of SGML NAME tokens and is used to subclass tag names. By convention, the class names are interpreted hierarchically, with the most general class on the left and the most specific on the right, where classes are separated by a period. The CLASS attribute is most commonly used to attach a different style to some element, but it is recommended that where practical class names should be picked on the basis of the element's semantics, as this will permit other uses, such as restricting search through documents by matching on element class names. Apart from the values suggested above, the conventions for choosing class names are outside the scope of this specification.

CLEAR

This attribute is common to all block-like elements. When text flows around a figure or table in the margin, you sometimes want to start the note below the figure rather than alongside it. The CLEAR attribute allows you to move down unconditionally:

clear=left

move down until left margin is clear

clear=right

move down until right margin is clear

clear=all

move down until both margins are clear

Alternatively, you can decide to place the note alongside the figure just so long as there is enough room. The minimum width needed is specified as:

clear="40 en"

move down until there is at least 40 en units free

clear="100 pixels"

move down until there is at least 100 pixels free

The style sheet (or browser defaults) may provide default minimum widths for each class of block-like elements.

SRC

Specifies an image to appear preceding the note. The image is specified as a URI. This attribute may appear together with the MD attribute.

MD

Specifies a message digest or cryptographic checksum for the associated graphic specified by the SRC attribute. It is used when you want to be sure that a linked object is indeed the same one that the author intended, and hasn't been modified in any way. For instance, MD="md5:jV2OfH+nnXHU8bnkPAad/mSQlTDZ", which specifies an MD5 checksum encoded as a base64 character string. The MD attribute is generally allowed for all elements which support URI based links.

Footnotes

Permitted Context: %body.content, %flow, %block

Content Model: %body.content

The FN element is designed for footnotes, and when practical, rendered as pop-up notes.

Example:

```
<DL>
<DT>Hamlet: <DD>You should not have believed me, for virtue cannot
so <a href="#fn1">inoculate</a> our old stock but we shall <a
href="#fn2">relish of it</a>. I loved you not.

<DT>Ophelia: <DD> I was the more deceived.

<DT>Hamlet: <DD>Get thee to a nunnery. Why wouldst thou be a breeder
of sinners? I am myself <a href="#fn2">indifferent honest</a> ...
</DL>

<fn id=fn1><i>inoculate</i> - graft</fn>
<fn id=fn2><i>relish of it</i> - smack of it (our old sinful nature)</fn>
<fn id=fn3><i>indifferent honest</i> - moderately virtuous</fn>
```

Note: If %html.recommended is active, the HTML 3.0 DTD expects you
to enclose plain text in a block element such as <P> e.g.

```
<FN ID=fn23><P>A simple footnote</FN>
```

Permitted Attributes

ID

An SGML identifier used as the target for hypertext links or for naming particular elements in associated style sheets. Identifiers are NAME tokens and must be unique within the scope of the current document.

LANG

This is one of the ISO standard language abbreviations, e.g. "en.uk" for the variation of English spoken in the United Kingdom. It can be used by parsers to select language specific choices for quotation marks, ligatures and hypenation rules etc. The language attribute is composed from the two letter language code from ISO 639, optionally followed by a period and a two letter country code from ISO 3166.

CLASS

This a space separated list of SGML NAME tokens and is used to subclass tag names. By convention, the class names are interpreted hierarchically, with the most general class on the

left and the most specific on the right, where classes are

separated by a period. The CLASS attribute is most commonly used to attach a different style to some element, but it is recommended that where practical class names should be picked on the basis of the element's semantics, as this will permit other uses, such as restricting search through documents by matching on element class names. The conventions for choosing class names are outside the scope of this specification.

Block Quotes

Permitted Context: %Body.Content, %flow, %block

Content Model: %Body.Content followed by optional CREDIT element

The BQ element is used for extended quotations. The tag name has been abbreviated from HTML 2.0's BLOCKQUOTE to the more convenient BQ, and the content model extended to allow the source of the quotation to be credited.

Example:

```
<BQ>
<P>But now I shall shortly proffer him the strength and the
courage of the Geats in combat. He who has the right to it shall
go once more to the mead-drinking with confident heart, after
the morning light of another day, the sun clothed in ethereal
radiance, shines from the south upon the children of men.
<CREDIT>Beowulf replying to Unferth, from the Anglo-Saxon poem
"Beowolf", Cotton Vitellus A xv manuscript</CREDIT>
</BQ>
```

Note: If %html.recommended is active, the HTML 3.0 DTD expects you to enclose plain text in a block element such as <P>

Permitted Attributes

ID

An SGML identifier used as the target for hypertext links or for naming particular elements in associated style sheets. Identifiers are NAME tokens and must be unique within the scope of the current document.

LANG

This is one of the ISO standard language abbreviations, e.g. "en.uk" for the variation of English spoken in the United Kingdom. It can be used by parsers to select language specific choices for quotation marks, ligatures and hypenation rules etc. The language attribute is composed from the two letter language code from ISO 639, optionally followed by a period and a two letter country code from ISO 3166.

CLASS

This a space separated list of SGML NAME tokens and is used to subclass tag names. By convention, the class names are interpreted hierarchically, with the most general class on the left and the most specific on the right, where classes are separated by a period. The CLASS attribute is most commonly used to attach a different style to some element, but it is recommended that where practical class names should be picked on the basis of the element's semantics, as this will permit other uses, such as restricting search through documents by matching on element class names. The conventions for choosing class names

are outside the scope of this specification.

CLEAR

This attribute is common to all block-like elements. When text flows around a figure or table in the margin, you sometimes want to start the block quote below the figure rather than alongside it. The CLEAR attribute allows you to move down unconditionally:

clear=left

move down until left margin is clear

clear=right

move down until right margin is clear

clear=all

move down until both margins are clear

Alternatively, you can decide to place the quote alongside the figure just so long as there is enough room. The minimum width needed is specified as:

clear="40 en"

move down until there is at least 40 en units free

clear="100 pixels"

move down until there is at least 100 pixels free

The style sheet (or browser defaults) may provide default minimum widths for each class of block-like elements.

NOWRAP

The NOWRAP attribute is used when you don't want the browser to automatically wrap lines. You can then explicitly specify line breaks using the BR element.

The ADDRESS element

Permitted Context: %Body.Content

Content Model: P or %text

The ADDRESS element specifies such information as address, signature and authorship for the current document, and typically placed at the top or bottom of the document. When used with %text, the element acts similar to a paragraph with breaks before and after.

Example:

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

Permitted Attributes

ID

An SGML identifier used as the target for hypertext links or for naming particular elements in associated style sheets.

Identifiers are NAME tokens and must be unique within the scope of the current document.

LANG

This is one of the ISO standard language abbreviations, e.g. "en.uk" for the variation of English spoken in the United Kingdom. It can be used by parsers to select language specific choices for quotation marks, ligatures and hypenation rules etc. The language attribute is composed from the two letter language code from ISO 639, optionally followed by a period and a two letter country code from ISO 3166.

CLASS

This a space separated list of SGML NAME tokens and is used to subclass tag names. By convention, the class names are interpreted hierarchically, with the most general class on the left and the most specific on the right, where classes are separated by a period. The CLASS attribute is most commonly used to attach a different style to some element, but it is recommended that where practical class names should be picked on the basis of the element's semantics, as this will permit other uses, such as restricting search through documents by matching on element class names. The conventions for choosing class names are outside the scope of this specification.

CLEAR

This attribute is common to all block-like elements. When text flows around a figure or table in the margin, you sometimes want to start the address element below the figure rather than alongside it. The CLEAR attribute allows you to move down unconditionally:

clear=left
: move down until left margin is clear

clear=right
: move down until right margin is clear

clear=all
: move down until both margins are clear

Alternatively, you can decide to place the address alongside the figure just so long as there is enough room. The minimum width needed is specified as:

clear="40 en"
: move down until there is at least 40 en units free

clear="100 pixels"
: move down until there is at least 100 pixels free

The style sheet (or browser defaults) may provide default minimum widths for each class of block-like elements.

NOWRAP

The NOWRAP attribute is used when you don't want the browser to automatically wrap lines. You can then explicitly specify line breaks using the BR element.

The FORM element

Permitted Context: %Body.Content

Content Model: %Body.Content --plus-- INPUT, TEXTAREA, and SELECT

Note you are not allowed to nest FORM elements!

HTML fill-out forms can be used for questionaires, hotel reservations, order forms, data entry and a wide variety of other applications. The form is specified as part of an HTML document. The user fills in the form and then --submits-- it. The user agent then sends the form's contents as designated by the FORM element. Typically, this is to an HTTP server, but you can also email form contents for asynchronous processing.

Forms are created by placing input fields within paragraphs, preformatted text, lists and tables. This gives considerable flexibility in designing the layout of forms.

HTML 3.0 supports the following kinds of fields:

* Simple text fields

* Multi-line text fields

*   Radio buttons

*   Checkboxes

*   Range controls (sliders, or knobs)

*   Single/multiple choice menus

*   Scribble on image

*   File widgets for attaching files to forms.

*   Submit buttons for sending form contents

*   Reset buttons for resetting fields to their initial values

*   Hidden fields for book keeping information

It is expected that future revisions to HTML will add support for audio fields, multi-row entry of database tables, and extending multi-line text fields to support a range of other data types, in addition to plain text. Client-side scripts will provide the means to constrain field values and to add new field types.

Example of a form

This fictitious example is a questionnaire. It uses the INPUT element for simple text fields, radio buttons, checkboxes, and the submit and reset buttons. The TEXTAREA field is used for a multi-line text entry field. The form fields are laid out with several paragraph elements and an unordered list. Notice the use of the NAME attribute to name each field:

```
<TITLE>Sample Questionaire</TITLE>
<H1>Sample Questionaire</H1>

<P>Please fill out this questionaire:

<FORM METHOD=post ACTION="http://www.hal.com/sample">
<P>Your name: <input name="name" size="48">
<P><input name="male" type=radio> Male
<P><input name="female" type=radio>Female
  Number in family: <input name="family" type=int>

<P>Cities in which you maintain a residence:

<UL PLAIN>
<LI><input name="city" type=checkbox value="kent"> Kent
<LI><input name="city" type=checkbox value="miami"> Miami
<LI>Others <textarea name="other" cols=48 rows=4></textarea>
</UL>
```

```
<P>Nickname: <INPUT NAME="nickname" size ="42">

<P>Thank you for responding to this questionaire.
<P><INPUT TYPE=SUBMIT> <INPUT TYPE=RESET>
</FORM>
```

Every form must be enclosed within a FORM element. There can be several forms in a single document, but the FORM element can't be nested. The browser is responsible for handling the input focus, i.e. which field will currently get keyboard input. Many platforms have existing conventions for forms, for example, using Tab and Shift-Tab to move the keyboard focus forwards and backwards between fields, and using the Enter (aka Return) key to submit the form.

This standard defines and requires support for the HTTP access protocol only. Under any protocol, the submitted contents of the form logically consist of a list of name/value pairs where the names are given by the NAME attributes of the various fields in the FORM. Each field will normally be given a distinct name. Several radio buttons can share the same name, as this is how you specify that they belong to the same control group - at any time, only one button in the group can be selected.

Note: The contents list of name/value pairs excludes unselected radio buttons and checkboxes. In general, any field with a null value can be omitted from the contents list.

HTML 3.0 doesn't provide direct support for constraining the values entered into text fields, or for derived fields whose values are calculated from the values of other fields. Rather than extending the markup to support these features, HTML 3.0 provides a means for associating the form with a script. Support for scripts is not required, however, and the HTML 3.0 specification doesn't cover the scripting languages or the details of their interface with the user agent

The SCRIPT attribute of the FORM element specifies the script via a URI. The user agent down-loads the script and interprets it locally. Scripts handle a variety of messages for individual fields and the form as a whole. These messages correspond to events such as:

* Enter/Leave Form (for initialization and clean up)

* When a field gains or loses the input focus

* Mouse clicks and drags over a field

* Keyboard events

Scripts can examine and set properties of fields. They can also examine a small set of standard properties of the user agent, for instance the user's name, the time of day, the type of user agent, and so on.

Scripts can't do anything that might jeopardize the user or the host machine. Scripts can't send messages over the network, or read or write files. The library calls that are allowed are restricted to a very small and well defined set. These precautions are necessary for untrusted scripts. It is envisaged that script interpreters will offer a much wider application programming interface to trusted scripts, as determined on the basis of a digital signature by a trusted third party.

Permitted Attributes for FORM

ACTION

The ACTION attribute is a URL specifying the location to which the contents of the form is submitted to elicit a response. If the ACTION is missing, the URL for the document itself is assumed. The way data is submitted varies with the access protocol of the URL, and with the values of the METHOD and ENCTYPE attributes.

METHOD

This specifies variations in the protocol used to send the form contents. It is currently restricted to GET (the default) or

POST. The attribute was introduced to inform user agents which

HTTP methods the server supports.

ENCTYPE

This attribute specifies the MIME content type to be used to encode the form contents. It defaults to the string: "application/x-www-form-urlencoded"

SCRIPT

This can be used to give a URI for a script. The scripting language and the interface with the user agent is not part of the HTML 3.0 specification.

The INPUT fields

Permitted Context: %Body.Content

Content Model: --Empty!--

The INPUT element is used for a wide variety of different kinds of entry fields within HTML fill-out forms. The TYPE attribute determines the type of field.

Single-line text fields --(type=text)--

These are used for entering short text strings, like peoples names, numbers and dates. The visible width of the field in characters can be set with the SIZE attribute. When using a variable pitch font, the SIZE attribute sets the width in en units (half the point size). The user should be able to enter more than this, with the contents of the field scrolling horizontally as needed. The MAXLENGTH attribute can be used to specify the maximum number of characters permitted for the string.

If the TYPE attribute is missing, the INPUT element is assumed to be a single-line text field. The NAME attribute is used to identify the field when the form's contents are converted to the name/value list. The VALUE field can be used to initialize the text string. Character entities can be used include accented characters in this string.

Note: Use the TEXTAREA element for multi-line text entry fields.

Password fields --(type=password)--

This is the same as single-line text fields except that each
character typed is echoed by a shadow character, e.g. an asterisk or
the space character. The user can see how many characters that have
been typed but not what was typed.

Checkbox fields --(type=checkbox)--

A checkbox field has two states: selected and unselected.Its
name/value pair only appear in the submitted data when selected.
Checkboxes are used for boolean attributes. They can also be used
for attributes which can take multiple values at the same time. This
is represented by a checkbox for each optional value, with the same
name for each of the checkboxes. Unselected checkboxes don't appear
in the submitted data. Both NAME and VALUE are required for
checkboxes. To initialize the checkbox to its selected state,
include the CHECKED attribute. Checkboxes provide an alternative to
using the SELECT element for multiple-choice menus.

Radio Buttons --(type=radio)--

Suitable for attributes which can take a single value from a set of alternatives. All radio buttons in the same group should be given the same NAME. Only the selected radio button in the group generates a name/value pair in the submitted data. Both NAME and VALUE are required for radio buttons. To initialize the radio button to its selected state, include the CHECKED attribute. Radio buttons offer an alternative to using the SELECT element for single choice menus.

Range fields --(type=range)--

These allow the user to pick a numeric value in between a lower and an upper bound. The range is specified with the MIN and MAX attributes, as in:

```
<input name=rating type=range min=1 max=10>
```

If either the lower or upper bound is a real number, then the range is real valued, otherwise it is restricted to integer values only. The VALUE attribute can be used to initialize the range field. It an error for the value to lie outside the specified range. The default value is midway between the lower and upper limits.

Scribble on Image --(type=scribble)--

These fields allow the user to scribble with a pointing device (such as a mouse or pen) on top of a predefined image. The image is specified as a URI with the SRC attribute. If the user agent can't display images, or can't provide a means for users to scribble on the image, then the field should be treated as a text field. The VALUE attribute can be used to initialize the text field for these users. It is ignored when the user agent provides scribble on image support.

File Attachments --(type=file)--

This allows users to attach one or more files to be submitted with the form's contents. The ACCEPT attribute can be used to specify a comma separated list of MIME content types. These are used to restrict the kinds of files that can be attached to the form. For instance:

```
<input name=pictures type=file accept="image/*">
```

This example restricts files to match "image/*", i.e. to registered MIME image types. For windows based user agents, it is suggested that file fields display the name of the last file attached, with the ability to open a file dialog box to view the complete list of files attached so far. The accept attribute then acts to specify the filter on the list of candidate files.

Hidden fields --(type=hidden)--

No field is presented to the user, but the content of the field is sent with the submitted form. This value may be used to transmit state information about client/server interaction, for instance a transaction identifier. These fields are needed because HTTP servers don't preserve state information from one request to the next.

Submit buttons --(type=submit)--

These are buttons that when pressed submit the form's data. You can use the VALUE attribute to provide a non-editable label to be displayed on the button. The default label is application-specific. A graphic can be specified for the submit button using the SRC attribute.

The submit button normally makes no contribution to the submitted data. The exception is when the field includes a NAME attribute, in which case, the name and value attributes are included with the submitted data. This can be used to distinguish which submit button the user pressed.

Image fields --(type=image)--

These act like submit buttons but include the location where the user clicked on the image. The image is specified with the SRC attribute.

--Should we phase these out, in favor of using SUBMIT? For this, we would need to ensure that the submit button included the location clicked when a graphic was specified with SRC.--

Reset buttons --(type=reset)--

When a reset button is pressed, the form's fields are reset to their specified initial values. The label to be displayed on the button may be specified just as for the SUBMIT button. Likewise, the SRC attribute can be used to specify a graphic.

Permitted Attributes for the INPUT element

ID

An SGML identifier used as the target for hypertext links or for naming particular elements in associated style sheets. Identifiers are NAME tokens and must be unique within the scope of the current document.

LANG

This is one of the ISO standard language abbreviations, e.g. "en.uk" for the variation of English spoken in the United Kingdom. It can be used by parsers to select language specific choices for quotation marks, ligatures and hypenation rules etc. The language attribute is composed from the two letter language code from ISO 639, optionally followed by a period and a two letter country code from ISO 3166.

CLASS

This a space separated list of SGML NAME tokens and is used to

subclass tag names. By convention, the class names are interpreted hierarchically, with the most general class on the left and the most specific on the right, where classes are separated by a period. The CLASS attribute is most commonly used to attach a different style to some element, but it is recommended that where practical class names should be picked on the basis of the element's semantics, as this will permit other uses, such as restricting search through documents by matching on element class names. The conventions for choosing class names are outside the scope of this specification.

TYPE

Defines the type of the field as one of: TEXT, PASSWORD, CHECKBOX, RADIO, RANGE, FILE, SCRIBBLE, HIDDEN, SUBMIT, IMAGE or RESET. It defaults to TEXT. The attribute value is an SGML name token and and as such is case insensitive.

NAME

This provides a character string used to name the field when submitting the form's data. Several fields may share the same name, for instance a group of radio buttons or checkboxes. The name is case insensitive.

VALUE

This is a character string or number which is used to initialize text, range and hidden fields.

DISABLED

When present, the field should be rendered as normal, but can't be modified by the user. Where practical the rendering should provide a cue that the field is disabled e.g. by graying out the text, changing the color of the background or similar.

ERROR

This attribute specifies an error message explaining why the field's current value is incorrect. When this attribute is missing, the field can be assumed to be ok. User agents are recommended to provide a cue to indicate that the field is in error.

CHECKED

The presence of this attribute indicates that a radio button or checbox should be initialized to its selected state.

SIZE

This specifies the visible width of a text or password field. For fixed pitch fonts, the size attribute specifies the maximum number of characters visible, while for variable pitch fonts, the attribute specifies the width in en units (half the point size).

MAXLENGTH

Specifies the maximum number of characters permitted for text and password fields.

MIN

This is an integer or real number and specifies the lower bound for a range field.

MAX

This is an integer or real number and specifies the upper bound for a range field.

ACCEPT

A comma separated list of MIME content types for use in restricting the types of files that can be attached to a form with a file field.

SRC (Source)

The SRC attribute specifies the URI for an image for use as the background of a SCRIBBLE, IMAGE, SUBMIT or RESET field. Its syntax is the same as that of the HREF attribute of the <A> tag.

MD

Specifies a message digest or cryptographic checksum for the associated image specified by the SRC attribute. It is used when you want to be sure that the image is indeed the same one that

the author intended, and hasn't been modified in any way. For
instance, MD="md5:jV2OfH+nnXHU8bnkPAad/mSQlTDZ", which specifies
an MD5 checksum encoded as a base64 character string. The MD
attribute is generally allowed for all elements which support
URI based links.

ALIGN

This attribute only applies to fields with background images,
i.e. SCRIBBLE, IMAGE, SUBMIT or RESET fields. It is intended to
provide the same positional control as for the IMG element. The
ALIGN attribute takes the values TOP or MIDDLE or BOTTOM,
defining whether the top or middle or bottom of the field should
be aligned with the baseline for the text line in which the
INPUT element appears.

With ALIGN=LEFT, the field will float down and over to the
current left margin, and subsequent text will wrap around the
right hand side of the field. Likewise for ALIGN=RIGHT, the
field aligns with the current right margin and, and text wraps
around the left.

TEXTAREA

Permitted Context: %Body.Content

Content Model: PCDATA (i.e. text and entities)

TEXTAREA

To let users enter more than one line of text, use the TEXTAREA element. For example:

```
<TEXTAREA NAME="address" ROWS=64 COLS=6>
HaL Computer Systems
1315 Dell Avenue
Campbell, California 95008
</TEXTAREA>
```

The text up to the end tag is used to initialize the field's value. The initialization text can contain SGML entities, e.g. for accented characters, but is otherwise treated as literal text. This end tag is always required even if the field is initially blank. When submitting a form, the line terminators are implementation dependent. Servers should be capable of recognizing a CR immediately followed by an LF, or separate CRs and LFs as all signifying the ends of lines. User agents should tolerate the same range of line terminators within the initialization text.

In a typical rendering, the ROWS and COLS attributes determine the visible dimension of the field in characters. The field is rendered in a fixed-width font. User agents should allow text to grow beyond these limits by scrolling as needed. The user agent is recommended to wrap words as they are entered, to fit within the textarea field. It is further recommended that a means is provided for users to turn this feature off and on.

Note: In the initial design for forms, multi-line text fields were supported by the INPUT element with TYPE=TEXT. Unfortunately, this causes problems for fields with long text values as SGML limits the length of attribute literals. The HTML 2.0 DTD allows for up to 1024 characters (the SGML default is only 240 characters).

Permitted Attributes

ID

An SGML identifier used as the target for hypertext links or for naming particular elements in associated style sheets. Identifiers are NAME tokens and must be unique within the scope of the current document.

LANG

This is one of the ISO standard language abbreviations, e.g. "en.uk" for the variation of English spoken in the United Kingdom. It can be used by parsers to select language specific choices for quotation marks, ligatures and hypenation rules etc. The language attribute is composed from the two letter language code from ISO 639, optionally followed by a period and a two letter country code from ISO 3166.

CLASS

This a space separated list of SGML NAME tokens and is used to subclass tag names. By convention, the class names are interpreted hierarchically, with the most general class on the left and the most specific on the right, where classes are separated by a period. The CLASS attribute is most commonly used to attach a different style to some element, but it is recommended that where practical class names should be picked on the basis of the element's semantics, as this will permit other uses, such as restricting search through documents by matching on element class names. The conventions for choosing class names are outside the scope of this specification.

NAME

The formal name of the field which is used in the form's contents list.

ROWS

This gives the visible number of text lines shown by the field.

User agents should allow text to grow beyond these limits by scrolling as needed.

COLS

The visible number of characters across the field. User agents should allow text to grow beyond these limits by scrolling as needed.

DISABLED

When present, the field should be rendered as normal, but can't be modified by the user. Where practical the rendering should provide a cue that the field is disabled e.g. by graying out the text, changing the color of the background or similar.

ERROR

This attribute specifies an error message explaining why the field's current value is incorrect. When this attribute is missing, the field can be assumed to be ok. User agents are recommended to provide a cue to indicate that the field is in error.

ALIGN

Take values TOP or MIDDLE or BOTTOM, defining whether the top or middle or bottom row of the field should be aligned with the baseline for the text line in which the TEXTAREA element appears. The default is align=top.

With ALIGN=LEFT, the field will float down and over to the current left margin, and subsequent text will wrap around the right hand side of the field. Likewise for ALIGN=RIGHT, the field aligns with the current right margin and, and text wraps around the left.

The SELECT element

Permitted Context: %Body.Content --but-- must be within FORM

Content Model: one ore more OPTION elements

The SELECT element is used for single and multiple choice menus. It is generally rendered as a drop-down or pop-up menu, and offers a more compact alternative to using radio buttons for single choice menus, or checkboxes for multiple choice menus.

Example:

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

This is a single choice menu. When you want a multiple choice menu, you need to include the MULTIPLE attribute with the SELECT element, e.g. <SELECT MULTIPLE NAME="flavor">.

The NAME attribute is used when creating the name/value list describing the form's contents. A name/value pair is contributed for each selected option. The value is taken from the OPTION's VALUE attribute, and defaults to the content of the OPTION when the VALUE attribute is missing.

For single choice menus, if no option is initially marked as selected, then the first item listed is selected. This is inappropriate for multiple choice menus, though.

Graphical Menus

HTML 3.0 extends the SELECT element to support graphical menus. This is allows you to specify an image for the SELECT element, and hotzones for each of the OPTION elements. In this way the same menu can be rendered as a conventional text-based menu for non-graphical user agents and a graphical menu for graphical user agents.

The image is specified in the same way as for IMG elements. This means you can specify suggested values for the width and height. You can also float the image to the left or right margins and flow other elements around it. The hotzones for OPTION elements are specified using the SHAPE attribute in the same way as for anchor elements.

Permitted Attributes

ID

An SGML identifier used as the target for hypertext links or for naming particular elements in associated style sheets.

Identifiers are NAME tokens and must be unique within the scope of the current document.

LANG

This is one of the ISO standard language abbreviations, e.g. "en.uk" for the variation of English spoken in the United Kingdom. It can be used by parsers to select language specific choices for quotation marks, ligatures and hypenation rules etc. The language attribute is composed from the two letter language code from ISO 639, optionally followed by a period and a two letter country code from ISO 3166.

CLASS

This a space separated list of SGML NAME tokens and is used to subclass tag names. By convention, the class names are interpreted hierarchically, with the most general class on the left and the most specific on the right, where classes are separated by a period. The CLASS attribute is most commonly used to attach a different style to some element, but it is recommended that where practical class names should be picked on the basis of the element's semantics, as this will permit other uses, such as restricting search through documents by matching on element class names. The conventions for choosing class names are outside the scope of this specification.

NAME

The formal name of the menu which is used in the form's contents list.

MULTIPLE

The presence of this attribute denotes that the SELECT element defines a multiple choice menu. In its absence, the element defines a single choice menu.

DISABLED

When present, the menu should be rendered as normal, but can't be modified by the user. Where practical the rendering should provide a cue that the menu is disabled e.g. by graying out the text, changing the color of the background or similar.

ERROR

This attribute specifies an error message explaining why the menu's current selections are incorrect. Further error messages can be attached to individual options. When this attribute is missing, the menu can be assumed to be ok. User agents are recommended to provide a cue to indicate that the menu is in error.

SRC (Source)

The SRC attribute is used for graphical menus to specify the URI for the image. Its syntax is the same as that of the HREF attribute of the <A> tag.

MD

Specifies a message digest or cryptographic checksum for the associated image specified by the SRC attribute. It is used when you want to be sure that the image is indeed the same one that the author intended, and hasn't been modified in any way. For instance, MD="md5:jV2OfH+nnXHU8bnkPAad/mSQlTDZ", which specifies an MD5 checksum encoded as a base64 character string. The MD attribute is generally allowed for all elements which support URI based links.

WIDTH

Optional suggested width for the image. By default, this is given in pixels.

HEIGHT

Optional suggested height for the image. By default, this is given in pixels.

UNITS

This optional attribute specifies the units for the width and height attributes. It is one of: units=pixels (the default) or units=em (the width of the letter "m") which scales with the font size.

ALIGN

Take values TOP or MIDDLE or BOTTOM, defining whether the top or middle or bottom of the graphic should be aligned with the baseline for the text line in which the IMG element appears.

With ALIGN=LEFT, the graphic will float down and over to the current left margin, and subsequent text will wrap around the right hand side of the graphic. Likewise for ALIGN=RIGHT, the graphic aligns with the current right margin and, and text wraps around the left.

Menu OPTIONs

Permitted Context: SELECT

Content Model: PCDATA

The OPTION element can only occur within a SELECT element. It represents a possible choice. It can only contain text, together with SGML entities for accented characters etc.

When the form is submitted, the NAME of the enclosing SELECT element is paired with the OPTION's VALUE attribute to contribute a name/value pair for the selection. Unselected options don't contribute to the form's submitted data. You can initialize the option to its selected state by including the SELECT attribute.

The SHAPE attribute is used for graphical menus to specify the region of the background image to be associated with this option. It uses the same definition as for the anchor element.

Permitted Attributes

ID

An SGML identifier used as the target for hypertext links or for naming particular elements in associated style sheets.

Identifiers are NAME tokens and must be unique within the scope of the current document.

LANG

This is one of the ISO standard language abbreviations, e.g. "en.uk" for the variation of English spoken in the United Kingdom. It can be used by parsers to select language specific choices for quotation marks, ligatures and hypenation rules etc. The language attribute is composed from the two letter language code from ISO 639, optionally followed by a period and a two letter country code from ISO 3166.

CLASS

This a space separated list of SGML NAME tokens and is used to subclass tag names. By convention, the class names are interpreted hierarchically, with the most general class on the left and the most specific on the right, where classes are separated by a period. The CLASS attribute is most commonly used to attach a different style to some element, but it is recommended that where practical class names should be picked on the basis of the element's semantics, as this will permit other uses, such as restricting search through documents by matching on element class names. The conventions for choosing class names are outside the scope of this specification.

DISABLED

When present, the option should be rendered as normal, but can't be modified by the user. Where practical the rendering should provide a cue that the option is disabled e.g. by graying out the text, changing the color of the background or similar.

ERROR

This attribute specifies an error message explaining why the option is inappropriate. When this attribute is missing, the option can be assumed to be ok. User agents are recommended to provide a cue to indicate that the option is in error.

VALUE

The string to be used together with the name attribute of the enclosing select element, when submitting the form. It defaults to the content of the OPTION element.

SELECTED

When present, this attribute signifies that the option should be initialized in its selected state. It is an error for more than one option to be selected for single choice menus.

SHAPE

This attribute is used within menus to define shaped hotzones associated with this option's value. The attribute value is a string taking one of the following forms:

"default"

Used to define a default menu choice for the menu background.

"circle x, y, r"

Where x and y define the center and r specifies the radius.

"rect x, y, w, h"

Where x, y define the upper left corner and w, h define the width and height respectively

"polygon x1, y1, x2, y2, ..."

Given n pairs of x, y coordinates, the polygon is closed by a line linking the n'th point to the first. Intersecting polygons use the non-zero winding number rule to determine if a point lies inside the polygon.

If a pointer event occurs in a region where two or more shapes overlap, the distance from the point to the center of gravity of each of the overlapping shapes is computed and the closest one chosen. This feature is useful when you want lots of closely spaced hotzones, for example over points on a map, as it allows you to use simple shapes without worrying about overlaps.

Note: The x coordinate increases to the right, and the y coordinate increases downwards in the same way as IMG and image maps. If both numbers are integers, the coordinates are interpreted as pixel offsets from the upper left corner of the image. Otherwise, the coordinates are interpreted as scaled values in the range 0.0 to 1.0 across the image. Note the syntax is tolerant of repeated white space characters between tokens.

Special Characters

This section contains information of how user agents should treat control characters and other special characters.

Character Data

The characters between the tags represent text encoded according to ISO 8859/1 8-bit single-byte coded graphic character set known as Latin Alphabet No. 1, or simply Latin-1. There are 256 character positions in the Latin-1 encoding. Latin-1 includes characters from most Western European languages. It consists of the space character, 186 characters that form a subset of the graphic characters in ISO 6937/2 (1983), and four additional characters that are intended for inclusion in ISO 6937/2. For more information, see Character Sets

The lower 128 character positions include a space, 33 control characters, the 26 upper- and lowercase letters of the english alphabet, 10 numerals and 32 other printing characters This subset, functionally identical to ASCII, is defined by ISO 646 7-bit coded character set for information interchange, also known as the International Reference Version. ISO 646 is identical in most respect to the ANSI standard for ASCII (American Standard Code for Information Interchange). The only significant difference between

ISO 646 and ASCII is the specific names assigned to the control characters which occupy positions 00-31 and 127

The upper 128 positions include a non-breaking space, a soft hyphen indicator, 93 graphical characters, 8 unassigned characters, and 25 control characters. The non-breaking space and soft hyphen indicator are not recognized and interpreted by all HTML browsers, and their use is discouraged

There are 58 character positions which are occupied by control characters. See the discussion for details on the interpretation of control characters. Because certain special characters are subject to interpretation and special processing, information providers and browser implementors should follow these guidelines

Certain characters may not be accessible from your keyboard, or some part of your system (i.e. translation software) may not be equipped to deal with 8-bit character codes. HTML and many WWW browsers provide character entity references and numerical character references to facilitate the entry and interpretation of characters by name and by numerical position.

Because certain characters will be interpreted as markup, they should be"escaped"; that is, represented by markup -- numeric character or entity references.

Special Characters

Certain characters are taken to have special meaning within the context of an HTML document. There are two printing characters which may be interpreted by the browser to have an effect of the format of the text:

Space

* Interpreted as a word space in all contexts except <PRE>.

* Interpreted as a no-break space within <PRE>.

The character entities &ensp; and &emsp; denote an en space and an em space respectively, where an en space is half the point size and an em space is equal to the point size of the current font. For fixed pitch fonts, the user agent can treat the en space as being equivalent to a single space character, and the em space as being equuivalent to two space characters.

Non-breaking Space (&nbsp;)

This should be treated in the same way as the space character (ASCII character code 32 decimal), except that the user agent should never break lines at this point. It is useful when you want to ensure that neigbouring words always stay together and don't get split across lines.

Hyphen

* Interpreted as a hyphen glyph in all contexts.

* Interpreted as a potential word space by hyphenation engine.

The character entities &endash; and &emdash; denote dash marks with the same widths as the &ensp; and &emsp; entities respectively.

Control Characters

Control characters are non-printable characters that are typically used for communication and device control, as format effectors, and as information separators.

In SGML applications, the use of control characters is limited in order to maximize the chance of sucessful interchange over heterogenous networks and operating systems. In HTML, there are only three control characters which are used. The remaining 55 control characters are shunned and should not appear in an HTML document. The valid control characters and their interpretation are:

Horizontal Tab (HT - 9 dec)

* Interpreted as a word space in all contexts except <PRE>.

* Within <PRE>, the tab should be interpreted to shift the horizontal column position to the next position which is a multiple of 8 on the same line; that is, col := (col+8) mod 8.

Line Feed (LF - 10 dec)

* Interpreted as a word space in all contexts except <PRE>.

* Within <PRE>, the tab should be interpreted as a shift to the start of a new line; that is, col := 0; row := row+1

Carriage Return (CR - 13 dec)

* Interpreted as a word space in all contexts except <PRE>.

* Within <PRE>, the tab should be interpreted as a shift to the start of the line; that is, col := 0;

Numeric Character References

Any printing character within the 8-bit character encoding of ISO 8859/1 (256 character positions) or the 7-bit character encoding of ISO 646 (128 character positions) may be represented within the text of an HTML document by a numeric character reference, e.g. &#233; is a small e with an acute accent. It is recommended that character entity references such as &eacute; are used in preference to numberic character references.

Security Considerations

Anchors, embedded images, and all other elements which contain URIs as parameters may cause the URI to be dereferenced. In this case, the security considerations of the URI specification apply.

Documents may be constructed whose visible contents mislead the reader to follow a link to unsuitable or offensive material.

The MD attribute is useful when authors are concerned that a linked object may be subsequently changed to something other than intended. This attribute is used to specify a cryptographic checksum for the linked object to provide a check on its integrity.

```
<!SGML  "ISO 8879:1986"

--

        SGML Declaration for HyperText Markup Language (HTML).


--


CHARSET

         BASESET  "ISO 646:1983//CHARSET

                   International Reference Version

                   (IRV)//ESC 2/5 4/0"

         DESCSET  0   9   UNUSED

                  9   2   9

                  11  2   UNUSED

                  13  1   13

                  14  18  UNUSED

                  32  95  32

                  127 1   UNUSED

     BASESET   "ISO Registration Number 100//CHARSET

                ECMA-94 Right Part of

                Latin Alphabet Nr. 1//ESC 2/13 4/1"


         DESCSET  128  32   UNUSED

                  160  96    32
```

```
CAPACITY       SGMLREF
          TOTALCAP       200000
          GRPCAP         150000


SCOPE   DOCUMENT
SYNTAX
      SHUNCHAR CONTROLS 0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16
           17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 127
      BASESET  "ISO 646:1983//CHARSET
             International Reference Version
             (IRV)//ESC 2/5 4/0"
      DESCSET  0 128 0
      FUNCTION
            RE         13
            RS         10
            SPACE      32
            TAB SEPCHAR  9



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

```
        SHORTREF SGMLREF
    NAMES    SGMLREF
    QUANTITY SGMLREF
```

```
         ATTSPLEN 2100
         LITLEN   1024
         NAMELEN  72    -- somewhat arbitrary; taken from
                internet line length conventions --
         PILEN    1024
         TAGLEN   2100
         GRPGTCNT 150
         GRPCNT   64

FEATURES
  MINIMIZE
    DATATAG  NO
    OMITTAG  YES
    RANK     NO
    SHORTTAG YES
  LINK
    SIMPLE   NO
    IMPLICIT NO
    EXPLICIT NO
  OTHER
    CONCUR   NO
    SUBDOC   NO
    FORMAL   YES
  APPINFO    "SDA"  -- conforming SGML Document Access application
```

```
                    --
>
<!--
        $Id: html.decl,v 1.1 1995/03/07 05:50:34 connolly Exp $

        Author: Daniel W. Connolly <connolly@hal.com>

        See also: http://www.hal.com/%7Econnolly/html-spec
          http://info.cern.ch/hypertext/WWW/MarkUp/MarkUp.html
 -->
```

Character Entity Set(s)

This section is undergoing revision ...

--In particular, we need to add a more complete list of character entities, e.g. for the characters below decimal 128 and missing codes such as currency signs.

The following entity names are used in HTML, always prefixed by ampersand (&) and followed by a semicolon as shown.

They represent particular graphic characters which have special meanings in places in the markup, or may not be part of the character set available to the writer.

Numeric and Special Graphic Entities

The following table lists each of the supported characters specified in the Numeric and Special Graphic entity set, along with its name, syntax for use, and description.

This list is derived from "ISO 8879:1986//ENTITIES Numeric and Special Graphic//EN" however HTML does not provide support for the entire entity set. Only the entities listed below are supported.

Name	Syntax	Description
lt	&lt;	Less than sign
gt	&gt;	Greater than sign
amp	&amp;	Ampersand
quot	&quot;	Double quote sign

ISO Latin 1 Character Entities

The following table lists each of the characters specified in the Added Latin 1 entity set, along with its name, syntax for use, and description.

This list is derived from "ISO 8879:1986//ENTITIES Added Latin 1//EN", and HTML does provide support for the entire entity set.

Name	Syntax	Description
Aacute	&Aacute;	Capital A, acute accent
Agrave	&Agrave;	Capital A, grave accent
Acirc	&Acirc;	Capital A, circumflex accent
Atilde	&Atilde;	Capital A, tilde

Aring	&Aring;	Capital A, ring
Auml	&Auml;	Capital A, dieresis or umlaut mark
AElig	&AElig;	Capital AE dipthong (ligature)

Ccedil	&Ccedil;	Capital C, cedilla
Eacute	&Eacute;	Capital E, acute accent
Egrave	&Egrave;	Capital E, grave accent
Ecirc	&Ecirc;	Capital E, circumflex accent
Euml	&Euml;	Capital E, dieresis or umlaut mark
Iacute	&Iacute;	Capital I, acute accent
Igrave	&Igrave;	Capital I, grave accent
Icirc	&Icirc;	Capital I, circumflex accent
Iuml	&Iuml;	Capital I, dieresis or umlaut mark
ETH	&ETH;	Capital Eth, Icelandic
Ntilde	&Ntilde;	Capital N, tilde
Oacute	&Oacute;	Capital O, acute accent
Ograve	&Ograve;	Capital O, grave accent
Ocirc	&Ocirc;	Capital O, circumflex accent
Otilde	&Otilde;	Capital O, tilde
Ouml	&Ouml;	Capital O, dieresis or umlaut mark
Oslash	&Oslash;	Capital O, slash
Uacute	&Uacute;	Capital U, acute accent
Ugrave	&Ugrave;	Capital U, grave accent
Ucirc	&Ucirc;	Capital U, circumflex accent
Uuml	&Uuml;	Capital U, dieresis or umlaut mark
Yacute	&Yacute;	Capital Y, acute accent
THORN	&THORN;	Capital THORN, Icelandic

szlig	&szlig;	Small sharp s, German (sz ligature)
aacute	&aacute;	Small a, acute accent
agrave	&agrave;	Small a, grave accent
acirc	&acirc;	Small a, circumflex accent
atilde	&atilde;	Small a, tilde
atilde	&atilde;	Small a, tilde
auml	&auml;	Small a, dieresis or umlaut mark
aelig	&aelig;	Small ae dipthong (ligature)
ccedil	&ccedil;	Small c, cedilla
eacute	&eacute;	Small e, acute accent
egrave	&egrave;	Small e, grave accent
ecirc	&ecirc;	Small e, circumflex accent
euml	&euml;	Small e, dieresis or umlaut mark
iacute	&iacute;	Small i, acute accent
igrave	&igrave;	Small i, grave accent
icirc	&icirc;	Small i, circumflex accent
iuml	&iuml;	Small i, dieresis or umlaut mark
eth	&eth;	Small eth, Icelandic
ntilde	&ntilde;	Small n, tilde
oacute	&oacute;	Small o, acute accent
ograve	&ograve;	Small o, grave accent
ocirc	&ocirc;	Small o, circumflex accent
otilde	&otilde;	Small o, tilde
ouml	&ouml;	Small o, dieresis or umlaut mark
oslash	&oslash;	Small o, slash

uacute	&uacute;	Small u, acute accent
ugrave	&ugrave;	Small u, grave accent
ucirc	&ucirc;	Small u, circumflex accent

uuml	&uuml;	Small u, dieresis or umlaut mark
yacute	&yacute;	Small y, acute accent
thorn	&thorn;	Small thorn, Icelandic
yuml	&yuml;	Small y, dieresis or umlaut mark

Numerical Character References

This list, sorted numerically, is derived from the ISO 8859/1 8-bit single-byte coded graphic character set:

Reference	Description
&#00;-&#08;	Unused
&#09;	Horizontal tab
&#10;	Line feed
&#11;-&#31;	Unused
&32;	Space
&33;	Exclamation mark
&34;	Quotation mark
&35;	Number sign
&36;	Dollar sign

&37;	Percent sign
&38;	Ampersand
&39;	Apostrophe
&40;	Left parenthesis
&41;	Right parenthesis
&42;	Asterisk
&43;	Plus sign
&44;	Comma
&45;	Hyphen
&46;	Period (fullstop)
&47;	Solidus (slash)
&#48; - &#57;	Digits 0-9
&58;	Colon
&59;	Semi-colon
&60;	Less than
&61;	Equals aign
&62;	Greater than
&63;	Question mark
&64;	Commercial at
&#65;-&#90;	Letters A-Z
&91;	Left square bracket
&92;	Reverse solidus (backslash)

&93;	Right square bracket
&95;	Horizontal bar
&96;	Acute accent

&#97;-&#122;	Letters a-z
&123;	Left curly brace
&124;	Vertical bar
&125;	Right curly brace
&126;	Tilde
&#127;- &#160;	Unused
&161;	Inverted exclamation
&162;	Cent sign
&163;	Pound sterling
&164;	General currency sign
&165;	Yen sign
&166;	Broken vertical bar
&167;	Section sign
&168;	Umlaut (dieresis)
&169;	Copyright
&170;	Feminine ordinal
&171;	Left angle quote, guillemotleft
&172;	Not sign
&173;	Soft hyphen
&174;	Registered trademark

&175;	Macron accent
&176;	Degree sign
&177;	Plus or minus
&178;	Superscript two
&179;	Superscript three
&180;	Acute accent
&181;	Micro sign
&182;	Paragraph sign
&183;	Middle dot
&184;	Cedilla
&185;	Superscript one
&186;	Masculine ordinal
&187;	Right angle quote, guillemotright
&188;	Fraction one-fourth
&189;	Fraction one-half
&190;	Fraction three-fourths
&191;	Inverted question mark
&192;	Capital A, acute accent
&193;	Capital A, grave accent
&194;	Capital A, circumflex accent
&195;	Capital A, tilde
&196;	Capital A, ring
&197;	Capital A, dieresis or umlaut mark
&198;	Capital AE dipthong (ligature)
&199;	Capital C, cedilla

&200;	Capital E, acute accent
&201;	Capital E, grave accent
&202;	Capital E, circumflex accent

&203;	Capital E, dieresis or umlaut mark
&204;	Capital I, acute accent
&205;	Capital I, grave accent
&206;	Capital I, circumflex accent
&207;	Capital I, dieresis or umlaut mark
&208;	Capital Eth, Icelandic
&209;	Capital N, tilde
&210;	Capital O, acute accent
&211;	Capital O, grave accent
&212;	Capital O, circumflex accent
&213;	Capital O, tilde
&214;	Capital O, dieresis or umlaut mark
&215;	Multiply sign
&216;	Capital O, slash
&217;	Capital U, acute accent
&218;	Capital U, grave accent
&219;	Capital U, circumflex accent
&220;	Capital U, dieresis or umlaut mark
&221;	Capital Y, acute accent
&222;	Capital THORN, Icelandic
&223;	Small sharp s, German (sz ligature)

&224;	Small a, acute accent
&225;	Small a, grave accent
&226;	Small a, circumflex accent
&227;	Small a, tilde
&228;	Small a, tilde
&229;	Small a, dieresis or umlaut mark
&230;	Small ae dipthong (ligature)
&231;	Small c, cedilla
&232;	Small e, acute accent
&233;	Small e, grave accent
&234;	Small e, circumflex accent
&235;	Small e, dieresis or umlaut mark
&236;	Small i, acute accent
&237;	Small i, grave accent
&238;	Small i, circumflex accent
&239;	Small i, dieresis or umlaut mark
&240;	Small eth, Icelandic
&241;	Small n, tilde
&242;	Small o, acute accent
&243;	Small o, grave accent
&244;	Small o, circumflex accent
&245;	Small o, tilde
&246;	Small o, dieresis or umlaut mark
&247;	Division sign

&248;	Small o, slash
&249;	Small u, acute accent

&250;	Small u, grave accent
&251;	Small u, circumflex accent
&252;	Small u, dieresis or umlaut mark
&253;	Small y, acute accent
&254;	Small thorn, Icelandic
&255;	Small y, dieresis or umlaut mark

Math Entities

This list is in a very preliminary stage ...

--I hope to use ISO names where practical, and want to ensure that names are meaningful, rather than cryptic. The character codes for common fonts will be included, although which fonts to include is still under review.--

The following sets out the range of math symbols supported by HTML math, giving the HTML entity name, the corresponding LaTeX command name and a short description. Character codes are given in hexadecimal when available for the Postscript symbol set and HP's math-8 symbol set.

Continuation dots - ellipsis

&ldots;	\ldots	three dots on the baseline
&cdots;	\cdots	three dots on same level as a minus sign
&vdots;	\vdots	three vertical dots
&ddots;	\ddots	diagonal dots (top left to bottom right)
&dotfill;	\dotfill	like cdots but fills column in an array

Added Spacing

&thinsp;	\,	thin space
&sp;	\:	medium space
&emsp;	\;	thick space
&quad;	\quad	huge space

Lower case Greek Letters

			PS-Symbol	Math-8
&alpha;	\alpha	alpha	61	61
&beta;	\beta	beta	62	62
&gamma;	\gamma	gamma	67	63
&delta;	\delta	delta	64	64
&epsilon;	\epsilon	epsilon	--	65
&vepsilon;	\varepsilon	var epsilon	65	3B
&zeta;	\zeta	zeta	7A	66
&eta;	\eta	eta	68	67
&theta;	\theta	theta	71	68
&vtheta;	\vartheta	var theta	--	79
&iota;	\iota	iota	69	69
&kappa;	\kappa	kappa	6B	6A
&lambda;	\lambda	lambda	6C	6B

&mu;	\mu	mu	6D	6C
&nu;	\nu	nu	6E	6D
&xi;	\xi	xi	78	6E

&omicron;	....	omicron	6F	6F
&pi;	\pi	pi	70	70
&varpi;	\varpi	var pi	76	7B
&rho;	\rho	rho	72	71
&varrho;	\varrho	var rho	--	--
&sigma;	\sigma	sigma	73	72
&vsigma;	\varsigma	var sigma	56	5B
&tau;	\tau	tau	74	73
&upsilon;	\upsilon	upsilon	75	74
&phi;	\phi	phi	66	75
&varphi;	\varphi	var phi	6A	7A
&chi;	\chi	chi	63	76
&psi;	\psi	psi	79	77
&omega;	\omega	omega	77	78

Note: LaTeX uses the latin letter o for omicron.

```
<!-- Standard ISO/WWW icons courtesy of Bert Bos and Kevin Hughes

   These can be used in place of default symbols for list items or as
   part of hypertext links, and save time needed to download images.
   Browsers can define them in terms of library images or as URL/URNs.
-->


<!ENTITY ftp SDATA "ftp" -- ftp server -->
<!ENTITY gopher SDATA "gopher" -- gopher server -->
<!ENTITY telnet SDATA "telnet" -- telnet connection -->
<!ENTITY archive SDATA "archive" -- archive server -->
<!ENTITY filing.cabinet SDATA "filing.cabinet" -- filing cabinet -->
<!ENTITY folder SDATA "folder" -- folder or directory -->
<!ENTITY fixed.disk SDATA "fixed.disk" -- fixed media drive -->
<!ENTITY disk.drive SDATA "disk.drive" -- removeable media drive -->
<!ENTITY document SDATA "document" -- unspecified document type -->
<!ENTITY unknown.document SDATA "unknown.document" -- unrecognised document type -->
<!ENTITY text.document SDATA "text.document" -- text/plain, text.html etc. -->
<!ENTITY binary.document SDATA "binary.document" -- binary data -->
<!ENTITY binhex.document SDATA "binhex.document" -- binhex format -->
<!ENTITY audio SDATA "audio" -- audio sequence -->
<!ENTITY film SDATA "film" -- film or animation, such as an MPEG movie -->
<!ENTITY image SDATA "image" -- photograph, drawing or graphic of any kind -->
```

```
<!ENTITY map SDATA "map" -- geographical or a schematic map -->

<!ENTITY form SDATA "form" -- fill-out form -->

<!ENTITY mail SDATA "mail" -- email messages -->

<!ENTITY parent SDATA "parent" -- parent of current document -->

<!ENTITY next SDATA "next" -- next document in current sequence -->

<!ENTITY previous SDATA "previous" -- previous document in current sequence -->

<!ENTITY home SDATA "home" -- home document -->

<!ENTITY toc SDATA "toc" -- table of contents -->

<!ENTITY glossary SDATA "glossary" -- glossary of terms etc. -->

<!ENTITY index SDATA "index" -- searchable index -->

<!ENTITY summary SDATA "summary" -- summary -->


<!ENTITY calculator SDATA "calculator" -- A calculator -->

<!ENTITY caution SDATA "caution" -- Warnign sign -->

<!ENTITY clock SDATA "clock" -- A clock -->

<!ENTITY compressed.document SDATA "compressed.document">

<!ENTITY diskette SDATA "diskette" -- A diskette -->

<!ENTITY display SDATA "display" -- A computer screen -->

<!ENTITY fax SDATA "fax" -- A fax machine -->

<!ENTITY mail.in SDATA "mail.in" -- mail-in tray -->

<!ENTITY mail.out SDATA "mail.out" -- mail-out tray -->

<!ENTITY mouse SDATA "mouse" -- mouse/pointing device -->

<!ENTITY printer SDATA "printer" -- hardcopy device -->

<!ENTITY tn3270 SDATA "tn3270" --tn3270 terminal session -->

<!ENTITY trash SDATA "trash" -- waste paper basket -->

<!ENTITY uuencoded.document SDATA "uuencoded.document" -- uuencoded data -->
```

```
<!--
    html3.dtd

    Document Type Definition for the HyperText Markup Language (HTML DTD)

    Draft: Fri 24-Mar-95 09:46:33

    Author: Dave Raggett <dsr@hplb.hpl.hp.com>

    W3O is developing a testbed browser to provide practical
    experience with HTML 3.0 before it becomes a standard.
    See:  http://www.w3.org/hypertext/WWW/Arena/

    This is an open process and comments are welcomed on the
    www-html mailing list.

    Please use the following MIME content type:

        Content-Type: text/html; version=3.0

    This will allow clients to distinguish HTML 3.0 from current
    HTML documents. This is most easily achieved by saving
    files with the extension ".html3" or ".ht3" so that servers
    can easily distinguish these files from HTML 2.0 files.
```

The entity HTML.Recommended can be used to give a more rigorous

version of the DTD suitable for use with SGML authoring tools.

The default version of the DTD offers a laxer interpretation,

e.g. allowing authors to omit leading <P> elements. You can

switch on the more rigorous version of the DTD by including

the following at the start of your HTML document.

```
 <!DOCTYPE HTML PUBLIC "-//IETF//DTD HTML 3.0//EN//"
    [ <!ENTITY % HTML.Recommended "INCLUDE"> ] >
```

Design Objectives:

o Backwards compatibility with 2.0

o Tightening up HTML.Recommended and
   moving more things to HTML.Deprecated

o Keep HTML - simple don't compete with CALS

o Make it practical for people to edit HTML 3.0
   documents directly, i.e. avoid long names.

o Tables, figures and math from HTML+
   with tweaks based on recent experience

- Client-side event handling for figures

  and graphical form selection menus

- o Add limited presentational controls with a view to use of linked style sheets (style overrides are supported)

- o Compatibility with ICADD as per Yuri's suggestions

HTML 3.0 relies on linked style info to give authors control over the appearence of documents. Such info is placed in a linked style sheet, or as overrides in the HTML document head, using the STYLE element. The generic CLASS attribute can be used to subclass elements when you want to use a different style from normal, e.g. you might use <h2 class=bigcaps> for headers with enlarged capital letters. Note that the class attribute has a wider scope than just style changes, e.g. browsers could provide the means for searching through documents, restricting search according to element class values.

The DTD contains a small number of attributes for direct control of basic alignment parameters; column widths for tables; support for custom bullets, sequence numbering for lists and headers; and text flow. These attributes offer control over appearence which would be inconvenient to express exclusively via associated style sheets.

The MD attribute for each hypertext or inline link specifies a
message digest such as MD5 for the linked object and is needed
to ensure someone hasn't tampered with a linked document.

History:

24th March '95

Changed ROLE->CLASS for HTML element
Added dummy elements to fix problem with mixed
content models for BODY, BLOCKQUOTE/BQ and FIG
Dropped audio fields from FORMs
Reinstated MIN/MAX for range fields
Reinstated DISABLED and ERROR attributes for fields

22nd March '95

Changed from em to en units. The latter
is a typographical unit = half point size
Merged NEEDS into CLEAR for control of textflow

21st March '95

Added REL=Banner to LINK element

Added BANNER element in place of <DIV CLASS=BANNER>

Added RANGE and SPOT elements

Added FN in place of <NOTE ROLE=FOOTNOTE>

Changed ROLE->CLASS for NOTE element

17th March '95

Fixed bug in PRE content model

Changed external references to omit traillling //

Dropped <!DOCTYPE HTML [...]> wrapper to avoid problems
  with "real" sgml parsers

Added NOFLOW attribute to FIG and TABLE

Fixed typo in IMG ALIGN attribute

Made SELECT match IMG for graphic attributes

Added decimal alignment for tabs and table cells

Added ALIGN attribute to TEXTAREA for parity
  with IMG and INPUT, including ALIGN=LEFT etc.

13th March '95

Dropped MARK tag on advice from SGML Open

Allowed spaces in table colspec attribute

Changed ARRAY element

Added CHOOSE tag to BOX element

Cleaned up PRE content model

Obsoleted tags incompatible with SGML

6th March '95

Added several tags to MATH: %mathface, %mathvec

and improved ROOT, with new SQRT convience tag

1st March '95

Dropped align attribute from BR element

Added indent attribute to TAB

Added optional CREDIT to end of BQ

Changed FIG to %body.content to allow headers

22nd February '95

Added align attribute, and dropped before, after, center and right

  attributes to clean up TAB element

Added INS and DEL for legal documents

Added CREDIT to end of FIG element

Dropped FN in favor of <NOTE ROLE=FootNote>

9th Feburuary '95

Dropped base attribute mechanism for scoping relative URLs

Dropped nofold attribute for disabling whitespace folding

Dropped border width attributes for FIG (-> style sheet)

Dropped delims attribute from math BOX element

Dropped stylistic attributes from OL such as inherit (-> stylesheet)

Added baseline to list of valign attribute values for tables.

Added DIV element for generic container class and static banners

```
    Added MARK element for marked range class

    Added closed set of LINK REL values for toolbars

    Added numbering attributes to headers

    Added bullet attributes to headers

    Added TERM element to math for style sheet control of term rendering

    Changed to imagemap=URI for server-side event handling for FIG/OVERLAY

    Changed delimiter attributes for math arrays

    Changed ROOT element for maths to allow an arbitrary radix

    Simplified numbering attributes for ordered lists

    Simplified STYLE element to leave binding to style language

-->


<!ENTITY % HTML.Version

        "-//IETF//DTD HTML 3.0//EN"


        -- Typical usage:


            <!DOCTYPE HTML PUBLIC "-//IETF//DTD HTML 3.0//EN">

            <html>

            ...

            </html>

        --

        >
```

```
<!--================== Flags for Marked Sections ===========================-->

<!ENTITY % HTML.Recommended "IGNORE"
        -- Certain features of the language are necessary for compatibility
           with widespread usage, but they may compromise the structural
           integrity of a document. This feature test entity enables
           a more prescriptive document type definition that eliminates
           the above features.
        -->

<![ %HTML.Recommended [
        <!ENTITY % HTML.Deprecated "IGNORE">
]]>

<!ENTITY % HTML.Deprecated "INCLUDE"
        -- Certain features of the language are necessary for compatibility
           with earlier versions of the specification, but they tend
           to be used an implemented inconsistently, and their use is
           deprecated. This feature test entity enables a document type
           definition that eliminates these features.
        -->

<!ENTITY % HTML.Obsoleted "IGNORE"
        -- The XMP, LISTING and PLAINTEXT tags are incompatible with SGML
           and derive from very early versions of HTML. They require non-
           standard parsers and will cause problems for processing
```

documents with standard SGML tools.

-->

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


<!ENTITY % Content-Type "CDATA"

        -- meaning a MIME content type, as per RFC1521

        -->


<!ENTITY % HTTP-Method "GET | POST"

        -- as per HTTP specification

        -->


<!ENTITY % URI "CDATA"

        -- The term URI means a CDATA attribute

           whose value is a Uniform Resource Identifier,

           as defined by

        "Uniform Resource Identifiers" by Tim Berners-Lee

        aka http://info.cern.ch/hypertext/WWW/Addressing/URL/URI_Overview.html

        aka RFC 1630


        Note that CDATA attributes are limited by the LITLEN

        capacity (1024 in the current version of html.decl),

        so that URIs in HTML have a bounded length.


        -->
```

```
<!ENTITY % REAL "CDATA" -- real numbers (not in SGML) -->

<!ENTITY % SHAPE "CDATA"
   -- Shape of hotzone in image.

      All coordinates are assumed to be numbers in the range 0 to 1
      and interpreted as fractional width/height and measured from
      the top left corner of the associated image.

      The attribute value is a string taking one of the following forms:

            "default"

      Used to define a default link for the figure background.

         "circle x, y, r"

      (x, y) define the center and r the radius.

         "rect x, y, w, h"

      (x, y) defines upper left, and w and h the width and height.

         "polygon x1, y1, x2, y2, ..."

      Given n pairs of x, y coordinates, the polygon is closed by a
```

line linking the n'th point to the first. Intersecting polygons

use the non-zero winding number rule to determine if a point lies

inside the polygon.I

```
   --
>

<!-- 3.0 Parameter Entities -->

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

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


<![ %HTML.Deprecated [
   <!ENTITY % list "UL | OL | DIR | MENU">
   <!ENTITY % blockquote "BLOCKQUOTE | BQ">
]]>

<!ENTITY % list "UL | OL">

<!ENTITY % blockquote "BQ">

<!ENTITY % preformatted "PRE">
```

```
<!-- The CLASS attribute is used to subclass HTML elements for
     rendering purposes, when used with style sheets, e.g. DSSSL lite -->

<!ENTITY % attrs  -- common attributes for elements --
        'id       ID      #IMPLIED -- as target for hrefs (link ends) --
         lang     CDATA   "en.us"  -- ISO language, country code --
         class    NAMES   #IMPLIED -- for subclassing elements --'>

<!-- SGML standard forces different NAMES for all attribute values
     in the same element, regardless of the attribute name! As a result
     CDATA is used for CLEAR attribute to avoid clash with ALIGN attribute.-->

<!--
When text flows around a figure or table in the margin, you sometimes want
to start an element like a header, paragraph or list below the figure rather
than alongside it. The CLEAR attribute allows you to move down unconditionally:

        clear=left          move down until left margin is clear
        clear=right         move down until right margin is clear
        clear=all           move down until both margins are clear

Alternatively, you can decide to place the element alongside the figure just
so long as there is enough room. The minimum width needed is specified as:

        clear="40 en"       move down until there is at least 40 en units free
        clear="100 pixels"  move down until there is at least 100 pixels free
```

The style sheet (or browser defaults) may provide default minimum widths for each class of block-like elements.

```
-->

<!ENTITY % needs -- Attributes for controlling text flow. Used in headers
                and other elements to guarantee sufficient room --
      'clear  CDATA #IMPLIED'>

<!--
   The following attribute may be included where ever a URL can be given:

   md     message digest e.g. md="md5:jV2OfH+nnXHU8bnkPAad/mSQlTDZ"
          where the digest is base64 encoded and preceded by a prefix
          denoting the algorithm (in this case MD5).
-->

<!ENTITY % url.link -- Attributes associated with URL based links --
      "md    CDATA  #IMPLIED  -- message digest for linked object --">

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

<!-- The HTML list of Latin-1 entities includes the full range
     of characters in widely available Latin-1 fonts, and as such
     is a mixture of ISOlat1 and other ISO publishing symbols -->

<!ENTITY % HTMLlat1 PUBLIC
```

```
  "-//IETF//ENTITIES Added Latin 1 for HTML//EN">
%HTMLlat1;

<!--================ Entities for special symbols =========================-->

<!ENTITY emsp   SDATA "[emsp  ]" -- em space -->
<!ENTITY ensp   SDATA "[ensp  ]" -- en space (1/2-em) -->
<!ENTITY mdash  SDATA "[ndash ]" -- em dash -->
<!ENTITY ndash  SDATA "[ndash ]" -- en dash (1/2-em) -->
<!ENTITY nbsp   SDATA "[nbsp  ]" -- non breaking space -->
<!ENTITY shy    SDATA "[shy   ]" -- soft hyphen -->
<!ENTITY copy   SDATA "[copy  ]" -- copyright sign -->
<!ENTITY trade  SDATA "[trade ]" -- trade mark sign -->
<!ENTITY reg    SDATA "[reg   ]" -- registered sign -->

<!--================ Entities for standard icons ==========================-->

<!-- a range of standard icons such as &folder; for use
     in speeding up display of directory listings etc. -->

<!ENTITY % HTMLicons PUBLIC
  "-//IETF//ENTITIES icons for HTML//EN">
%HTMLicons;

<!--================ Entities for math symbols ============================-->
```

```
<!-- ISO subset chosen for use with the widely available Adobe math font -->

<!ENTITY % HTMLmath PUBLIC
```

```
  "-//IETF//ENTITIES Math and Greek for HTML//EN">

%HTMLmath;


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


<!ENTITY % font " U | S | TT | I | BIG | SMALL">


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


<!ENTITY % misc "Q | LANG | AU | DFN | PERSON | ACRONYM | ABBREV | INS | DEL">


<!ENTITY % special "TAB | MATH | A | IMG | BR">


<!ENTITY % notmath "%font | %phrase | %special | %misc">


<!ENTITY % text "#PCDATA | SUB | SUP | B | %notmath">




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


<!ELEMENT (%font|B|%phrase|%misc) - - (%text)+>

<!ATTLIST (%font|B|%phrase|%misc) %attrs;>


<!-- Subscripts and superscripts. The ALIGN attribute is only used for math -->
```

```
<!ELEMENT (SUB|SUP) - - (%text)+>
<!ATTLIST (SUB|SUP)
        %attrs;
        align (left|center|right) #IMPLIED
        >

<!-- Forced line break -->

<!ELEMENT BR    - O EMPTY>
<!ATTLIST BR
        %attrs;
        %needs; -- for control of text flow --
        >

<!-- Named left, center and right tab stops (independent of '\t' char) -->

<!ELEMENT TAB - O EMPTY>
<!ATTLIST TAB
        id       ID       #IMPLIED  -- defines named tab stop --
        indent   NUMBER   0         -- en units before new tab stop --
        to       IDREF    #IMPLIED  -- jump to named tab stop --
        align   (left|center|right|decimal) left
        dp      CDATA     #IMPLIED  -- decimal point e.g. dp="," --
        >
```

```
<!--================== Link Markup
========================================-->


<!--
```

```
    With HTML 3.0 you can use ID attributes on most elements for named
    link ends. The use of the NAME attribute on anchors is deprecated.

    Do we want to support arbitrary elements for link starts? This would
    involve adding HREF and related attributes to most elements.
-->

<![ %HTML.Deprecated [
   <!ENTITY % linkName "name CDATA #IMPLIED -- named link end --">
]]>

<!ENTITY % linkName "">


<!ENTITY % ToolBar "home|toc|index|glossary|copyright|
                up|previous|next|help|bookmark"

      -- LINK RELationship values which are used to create toolbar
         buttons or menu items for navigation, where toc stands
         for table of contents and bookmark provides for an open
         ended set of links, i.e. you can use multiple bookmarks
         for key entry points. Use the optional TITLE attribute
         to override default names.
      -->
```

```
<!ENTITY % linkType "NAME"
        -- A definitive list will be specified at a later date.

           They are used

                a) by stylesheets to control how collections of
                   html nodes are rendered into printed documents

                b) for document specific toolbars/menus when used
                   with the LINK element in document head:

                    "home|toc|index|glossary|copyright|
                          up|previous|next|help|bookmark"

                   where toc stands for table of contents and
                   bookmark provides for an open ended set of links,
                   i.e. you can use several bookmarks for key entry
                   points. Use the optional TITLE attribute to
                   override default names.

                c) for hypertext paths or guided tours,
                   with REL=NODE and REL=PATH.

                d) to make a link to a style sheet, e.g. rel=stylesheet
                   (used only with the LINK element).
```

e) to make a link to a separate banner, e.g. rel=banner

(used only with the LINK element).

```
    -->



<!ENTITY % linkExtraAttributes -- URN moved to %url.link --
    "rel %linkType #IMPLIED -- forward relationship type --
    rev %linkType #IMPLIED -- reversed relationship type
                    to referent data --
    title   CDATA #IMPLIED -- advisory only --
    methods NAMES #IMPLIED -- supported public methods of the object:
                        TEXTSEARCH, GET, HEAD, ... --
    ">

<![ %HTML.Deprecated [
  <!ENTITY % A.content "(%heading|%text)+">
]]>

<!ENTITY % A.content   "(%text)+">

<!ELEMENT A     - - %A.content -(A)>

<!ATTLIST A
    %attrs;
    href   %URI;  #IMPLIED
    %url.link;    -- standard link attributes --
```

```
        %linkName;     -- name attribute is deprecated; use ID instead --

        shape %SHAPE; #IMPLIED -- for shaped hotzones in FIGs --

        %linkExtraAttributes;

        >


<!--================== Images
===========================================-->


<!-- Desired widths are used for negotiating image size

     with the module responsible for painting the image.

     align=left or right cause image to float to margin

     and for subsequent text to wrap around image -->


<!ELEMENT IMG    - O EMPTY --  Embedded image -->

<!ATTLIST IMG

        %attrs;

        src  %URI;  #REQUIRED  -- URI of image to embed --

        %url.link;             -- standard link attributes --

        alt CDATA   #IMPLIED   -- for display in place of image --

        align  (top|middle|bottom|left|right) top -- relative to baseline

        --        width  NUMBER #IMPLIED -- desired width in en's or pixels --

        height NUMBER #IMPLIED -- desired height in en's or pixels --

        units  (en|pixels) pixels -- units for width and height --

        ismap (ismap) #IMPLIED -- pass clicks to server --

        >


<!--==================
Paragraphs=========================================-->


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

<!ATTLIST P
```

```
        %attrs;
        align  (left|center|right|justify) #IMPLIED
        %needs; -- for control of text flow --
        nowrap (nowrap) #IMPLIED -- disable wordwrap --
        >

<!--================== Headings, Titles, Sections ========================-->

<!ELEMENT HR    - O EMPTY -- customizable horizontal rule -->
<!ATTLIST HR
        %attrs;
        src     %URI;   #IMPLIED -- URI of custom rule graphic --
        %url.link;              -- standard link attributes --
        %needs; -- for control of text flow --
        >

<!--
  Headers can be numbered, although this is a matter for style sheets.
  The style sheet controls the numbering style:

     a) whether the parent numbering is inherited, e.g.  5.i.c
        where 5 is the current sequence number for H1 headers, and
        1 is the number for H2 headers and 3 for H3 headers.
```

```
    b) what style is used for current sequence number

       e.g. arabic, upperalpha, loweralpha, upperroman, lowerroman

       or a numbering scheme appropriate for the current language


  The skip attribute is used to skip over sequence numbers for items

  which have been left out of the list, e.g. skip=3 advances the

  sequence number past 3 omitted items. The seqnum sets the sequence

  number to a specified value. Note that the style sheet may take

  advantage of the sequence number for higher level headers.


  The dingbat or src attributes may be used to specify a bullet like

  image to be placed adjacent to the header. Defining this in the

  header element simplifies the document markup and avoids the need

  to use the clear or needs attribute in the following element to

  prevent it flowing around this image.

-->


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

<!ATTLIST ( %heading )

        %attrs;

        align  (left|center|right|justify) #IMPLIED

        %needs; -- for control of text flow --

        seqnum   NUMBER   #IMPLIED  -- starting sequence number --

        skip     NUMBER   0         -- skip seq nums for missing items --

        dingbat  ENTITY   #IMPLIED  -- dingbat entity from HTMLicons --

        src      (%URI;)  #IMPLIED  -- bullet defined by graphic --
```

```
%url.link;                    -- standard link attributes --
nowrap   (nowrap)  #IMPLIED -- disable wordwrap --
>
```

```
<!ELEMENT TITLE - -  (#PCDATA)

          -- The TITLE element is not considered part of the flow of text.

             It should be displayed, for example as the page header or

             window title.

          -->


<!--=================== Text Flows
=========================================-->


<!ENTITY % block

     "P | %list | DL

        | %preformatted

        | %blockquote

        | FORM | ISINDEX | FN

        | TABLE | FIG | NOTE">


<!--

  ((%block)* | (%text)*) would be much nicer as it would avoid the need

  for a <P> tag when all you want is a few words of text. The problem

  is that it also prevents: "<LI> <P>some text" since it forbids PCDATA

  and hence the white space between the <LI> and the <P>.

-->


<![ %HTML.Recommended [
   <!ENTITY % flow "(%block)*">
]]>

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

<!ELEMENT PRE - - (%text)* -(%pre.exclusion)>

<!ATTLIST PRE
      %attrs;
      width NUMBER #implied
      %needs; -- for control of text flow --
      >

<![ %HTML.Obsoleted [

<!ENTITY % literal "CDATA"
      -- special non-conforming parsing mode where
        the only markup signal is the end tag
        in full. This will cause problems for
        standard SGML tools!
      -->

<!ELEMENT XMP - -  %literal>
<!ELEMENT LISTING - -  %literal>
<!ELEMENT PLAINTEXT - O %literal>
```

```
]]>

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

```
<!ELEMENT DL    - -  (LH?, (DT|DD)+) -- this is perhaps too lax? -->

<!ATTLIST DL

        %attrs;

        %needs; -- for control of text flow --

        compact (compact) #IMPLIED -- more compact style --

        >


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

<!ELEMENT DD    - O  %flow;>

<!ATTLIST (DT|DD)

        %attrs;

        %needs; -- for control of text flow --

        >



<!ELEMENT (OL|UL) - -  (LH?, LI+) -- should we allow a list header ? -->


<!-- style sheet controls numbering style

    a) whether the parent numbering is inherited, e.g.  5.1.c

    b) what style is used for current sequence number

        e.g. arabic, upperalpha, loweralpha, upperroman, lowerroman

        or a numbering scheme for the current language

-->
```

```
<!ATTLIST OL
        %attrs;
        %needs; -- for control of text flow --
        continue (continue)  #IMPLIED   -- don't restart sequence number --
        seqnum    NUMBER     #IMPLIED   -- starting sequence number --
        compact (compact)    #IMPLIED   -- reduced interitem spacing --
        >

<!-- Unordered lists:

    o   single or multicolumn with
        horizontal or vertical wrapping

    o   plain or bulletted list items

    o   bullets can be customised via:
            - entities (dingbats in HTMLicons)
            - external graphic via URL
            - individual attributes on LI tags
-->
<!ATTLIST UL
        %attrs;
        %needs; -- for control of text flow --
        wrap (vert|horiz|none) none -- multicolumn list style --
        plain   (plain)  #IMPLIED   -- suppress bullets --
        dingbat  ENTITY  #IMPLIED   -- dingbat entity from HTMLicons --
        src     (%URI;)  #IMPLIED   -- bullet defined by graphic --

        %url.link;                  -- standard link attributes --

        compact (compact) #IMPLIED  -- reduced interitem spacing --
        >

<!ELEMENT LH - O (%text;)+ -- list header -->
<!ATTLIST LH %attrs;>

<!--
  For unordered lists, you can override the standard bullet with
  a custom graphic specified via a URI e.g. src="splash.gif" or
  a reference to one of the HTMLicons graphics e.g. dingbat=folder

  The skip attribute is used with ordered lists to skip over sequence
  numbers for items which have been left out of the list, e.g. skip=3
  advances the sequence number past 3 omitted items.
-->

<!ELEMENT LI - O %flow; -- list item -->
<!ATTLIST LI
    %attrs;
    %needs; -- for control of text flow --
    dingbat ENTITY #IMPLIED -- dingbat entity from HTMLicons --
    src  (%URI;) #IMPLIED   -- custom bullet graphic --
    %url.link;              -- standard link attributes --
    skip NUMBER 0           -- skip seq nums for missing items --
    >
```

```
<!-- DIR and MENU are now subsumed by UL with type=plain. Use the

     wrap attribute to control wrapping style for multicolumn lists -->


<![ %HTML.Deprecated [

    <!ELEMENT (DIR|MENU) - - (LI)+ -(%block)>

    <!ATTLIST (DIR|MENU)

        compact (compact) #IMPLIED>

]]>


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


<![ %HTML.Recommended [

        <!ENTITY % body.content "(DIV|%heading|%block|HR|ADDRESS)*"

        -- <h1>Heading</h1>

           <p>Text ...

                is preferred to

           <h1>Heading</h1>

           Text ...

        -->

]]>


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


<!ELEMENT BODY O O  (BANNER?, BODYTEXT) +(SPOT)>

<!ATTLIST BODY
```

```
%attrs;
background %URI; #IMPLIED  -- texture tile for document background --
>
```

```
<!ELEMENT BODYTEXT O O %body.content -- dummy element -->

<!--
  The BANNER element is used for a banner section which appears at
  the top of the window and doesn't scroll with window contents.
  This can be used for corporate logos, copyright statements and
  disclaimers, as well as customized navigation/search controls.
-->
<!ELEMENT BANNER - - %body.content>
<!ATTLIST BANNER %attrs; >

<!-- SPOT is used to insert IDs at arbitrary places
     e.g. for end points of a marked range (see RANGE) -->
<!ELEMENT SPOT - O EMPTY>
<!ATTLIST SPOT id ID #REQUIRED>

<!ELEMENT (%blockquote) - - (BODYTEXT, CREDIT?)>
<!ATTLIST (%blockquote)
        %attrs;
        %needs; -- for control of text flow --
        nowrap   (nowrap)  #IMPLIED -- disable wordwrap --
        >
```

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


<!ELEMENT ADDRESS - - %address.content>

<!ATTLIST ADDRESS

        %attrs;

        %needs; -- for control of text flow --

        >



<!--

  DIV can be used with the CLASS attribute to represent different

  kinds of container, e.g. chapter, section, abstract, appendix.

-->


<!ELEMENT DIV - - %body.content>

<!ATTLIST DIV

        %attrs;

        %needs; -- for control of text flow --

        align   (left|center|right) left -- alignment of following text --

        nowrap   (nowrap)  #IMPLIED -- disable wordwrap --

        >


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



<!--

    As HTML 2.0 plus a few extensions:
```

a) A RANGE control which varies between pair of values

specified with the size attribute, e.g. SIZE="1, 10"

```
        b)  FILE widget for uploading one or more files to a server

        c)  SCRIBBLE on image widget that sends the "ink" to the server

        d)  SUBMIT/RESET buttons can now be customised with an image.
            This subsumes the IMAGE type which is now deprecated.

        e)  Graphical SELECTion menus are now supported, using
            the new SHAPE attribute on OPTION elements.

    Further extensions are in the pipeline (e.g. table entry,
    multiple data formats for textarea fields and client-side
    scripts with custom widgets) but will have to wait until
    the backlog of implementation work diminishes.
-->

<!ELEMENT FORM - - %body.content -(FORM) +(INPUT|SELECT|TEXTAREA)>
<!ATTLIST FORM
        action %URI #REQUIRED -- server-side form handler --
        method (%HTTP-Method) GET -- see HTTP specification --
        enctype %Content-Type; "application/x-www-form-urlencoded"
        script %URI #IMPLIED -- link to client-side script --
        >
```

```
<![ %HTML.Deprecated [
   <!ENTITY % InputType "(TEXT | PASSWORD | CHECKBOX | RADIO | SUBMIT
              | RESET | RANGE | FILE | SCRIBBLE | HIDDEN | IMAGE)">
]]>

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

<!ELEMENT INPUT - O EMPTY>
<!ATTLIST INPUT
      %attrs;
      type %InputType TEXT
      name  NAME  #IMPLIED       -- required for all but submit and reset --
      value CDATA #IMPLIED       -- required for radio buttons & checkboxes --
      disabled (disabled) #IMPLIED -- read-only fields --
      error CDATA #IMPLIED       -- why field is in error --
      checked (checked) #IMPLIED -- for radio buttons and check boxes --
      size NUMBER #IMPLIED       -- visible width of TEXT fields --
      maxlength NUMBER #IMPLIED  -- max number of chars for TEXT fields --
      min %REAL #IMPLIED         -- lower limit for RANGE fields --
      max %REAL #IMPLIED         -- upper limit for RANGE fields --
      accept CDATA #IMPLIED      -- list of MIME types for file fields --
      src   %URI  #IMPLIED       -- for fields with background images --
      %url.link;                 -- standard link attributes --
      align  (top|middle|bottom|left|right) top
      >
```

```
<!--
  SRC attribute added for graphical selection menus
  The WIDTH, HEIGHT and UNITS attributes apply to the
  image specified by the SRC attribute.
 -->

<!ELEMENT SELECT - - (OPTION+) -(INPUT|TEXTAREA|SELECT)>
<!ATTLIST SELECT
        %attrs;
        name CDATA #REQUIRED
        multiple (multiple) #IMPLIED
        disabled (disabled) #IMPLIED -- read-only menu --
        error CDATA #IMPLIED   -- why selections are in error --
        src  %URI  #IMPLIED    -- for graphical selection menus --
        %url.link;             -- standard link attributes --
        width  NUMBER #IMPLIED -- desired width of in en's or pixels --
        height NUMBER #IMPLIED -- desired height in en's or pixels --
        units  (en|pixels) pixels -- units for width and height --
        align  (top|middle|bottom|left|right) top
        >

<!ELEMENT OPTION - O (#PCDATA)>
<!ATTLIST OPTION
        %attrs;
        selected (selected) #IMPLIED
        value  CDATA  #IMPLIED -- default to element content --
        shape %SHAPE; #IMPLIED -- for graphical selection menus --
        disabled (disabled) #IMPLIED -- unselectable option --
        error CDATA #IMPLIED   -- why this choice is in error --
        >

<!--
  Multi-line text input field. align=left or right
  causes the field to float to margin and for
  subsequent text to wrap around the field.
-->

<!ELEMENT TEXTAREA - - (#PCDATA) -(INPUT|TEXTAREA|SELECT)>
<!ATTLIST TEXTAREA
        %attrs;
        name CDATA #REQUIRED
        rows NUMBER #REQUIRED
        cols NUMBER #REQUIRED
        disabled (disabled) #IMPLIED -- read-only field --
        error CDATA #IMPLIED         -- why field is in error --
        align  (top|middle|bottom|left|right) top
        >

<!--====================== Captions
========================================-->

<!ELEMENT CAPTION - - (%text;)+ -- table or figure caption -->
<!ATTLIST CAPTION
```

```
%attrs;

align (top|bottom|left|right) #IMPLIED

>
```

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


<!--
   Tables and figures can be aligned in several ways:


   bleedleft   flush left with the left (window) border
   left        flush left with the left text margin
   center      centered (text flow is disabled for this mode)
   right       flush right with the right text margin
   bleedright  flush right with the right (window) border
   justify     when applicable the table/figure should stretch
               to fill space between the text margins


   Note: text will flow around the table or figure if the browser
   judges there is enough room and the alignment is not centered
   or justified. The table or figure may itself be part of the
   text flow around some earlier figure. You can in this case use
   the clear or needs attributes to move the new table or figure
   down the page beyond the obstructing earlier figure. Similarly,
   you can use the clear or needs attributes with other elements
   such as headers and lists to move them further down the page.
-->


<!ENTITY % block.align
```

```
        "align  (bleedleft|left|center|right|bleedright|justify) center">

<!--
   The HTML 3.0 table model has been chosen for its simplicity
   and the ease in writing filters from common DTP packages.

   By default the table is automatically sized according to the
   cell contents and the current window size. Specifying the columns
   widths using the colspec attribute allows browsers to start
   displaying the table without having to wait for last row.

   The colspec attribute is a list of column widths and alignment
   specifications. The columns are listed from left to right with
   a capital letter followed by a number, e.g. COLSPEC="L20 C8 L40".
   The letter is L for left, C for center, R for right alignment of
   cell contents. J is for justification, when feasible, otherwise
   this is treated in the same way as L for left alignment.
   Column entries are delimited by one or more space characters.

   The number specifies the width in en's, pixels or as a
   fractional value of the table width, as according to the
   associated units attribute. This approach is more compact
   than used with most SGML table models and chosen to simplify
   hand entry. The width attribute allows you to specify the
   width of the table in pixels, en units or as a percentage
   of the space between the current left and right margins.
```

To assist with rendering to speech, row and column headers

can be given short names using the AXIS attribute. The AXES

```
   attribute is used to explicitly specify the row and column
   names for use with each cell. Otherwise browsers can follow
   up columns and left along rows (right for some languages)
   to find the corresponding header cells.

   Table content model: Braille limits the width of tables,
   placing severe limits on column widths. User agents need
   to render big cells by moving the content to a note placed
   before the table. The cell is then rendered as a link to
   the corresponding note.

   To assist with formatting tables to paged media, authors
   can differentiate leading and trailing rows that are to
   be duplicated when splitting tables across page boundaries.
   The recommended way is to subclass rows with the CLASS attribute
   For example: <TR CLASS=Header>, <TR CLASS=Footer> are used for
   header and footer rows. Paged browsers insert footer rows at
   the bottom of the current page and header rows at the top of
   the new page, followed by the remaining body rows.
-->

<!ELEMENT TABLE - - (CAPTION?, TR*) -- mixed headers and data -->
<!ATTLIST TABLE
        %attrs;
```

```
    %needs; -- for control of text flow --
    border (border) #IMPLIED -- draw borders --
    colspec CDATA   #IMPLIED -- column widths and alignment --
    units  (en|pixels|relative) en -- units for column widths --
    dp      CDATA   #IMPLIED -- decimal point e.g. dp="," --
    width NUMBER    #IMPLIED -- absolute or percentage width --
    %block.align;  -- horizontal alignment --
    noflow (noflow) #IMPLIED -- noflow around table --
    nowrap (nowrap) #IMPLIED -- don't wrap words --
    >

<!ENTITY % cell "TH | TD">
<!ENTITY % horiz.align "left|center|right|justify">
<!ENTITY % vert.align  "top|middle|bottom|baseline">

<!--
  Browsers should tolerate an omission of the first <TR>
  tag as it is implied by the context. Missing trailing
  <TR>s implied by rowspans should be ignored.

  The alignment attributes act as defaults for rows
  overriding the colspec attribute and being in turn
  overridden by alignment attributes on cell elements.
  Use valign=baseline when you want to ensure that text
  in different cells on the same row is aligned on the
  same baseline regardless of fonts. It only applies
```

when the cells contain a single line of text.

-->

```
<!ELEMENT TR - O (%cell)* -- row container -->

<!ATTLIST TR

        %attrs;

        align  (%horiz.align) #IMPLIED -- horizontal alignment --

        valign (%vert.align)  top  -- vertical alignment --

        dp     CDATA     #IMPLIED  -- decimal point e.g. dp="," --

        nowrap (nowrap)  #IMPLIED  -- don't wrap words --

        >


<!--

    Note that table cells can include nested tables.

    Missing cells are considered to be empty, while

    missing rows should be ignored, i.e. if a cell

    spans a row and there are no further TR elements

    then the implied row should be ignored.

-->


<!ELEMENT (%cell) - O %body.content>

<!ATTLIST (%cell)

        %attrs;

        colspan NUMBER    1      -- columns spanned --

        rowspan NUMBER    1      -- rows spanned --

        align  (%horiz.align) #IMPLIED -- horizontal alignment --

        valign (%vert.align) top -- vertical alignment --
        dp      CDATA    #IMPLIED  -- decimal point e.g. dp="," --

        nowrap (nowrap)  #IMPLIED  -- don't wrap words --

        axis CDATA #IMPLIED -- axis name, defaults to element content --

        axes CDATA #IMPLIED -- comma separated list of axis names --

        >


<!--====================== Figures
=========================================-->


<!--

  The element contains text for use in non-graphical displays. Note that

  you can use the shape attribute in anchors to specify hotzones on images.

  This provides for local processing of pointer clicks and a unified method

  for dealing with graphical and non-graphical displays.


  Text is flowed around figures when the figure is left or right aligned.

  You can request the browser to move down until there is enough room for

  the next element, see the CLEAR and NEED attributes (in %needs)


  Figures offer a path towards embedding arbitrary information formats

  via some kind of OLE/OpenDoc mechanism.

-->


<!ELEMENT FIG - - (OVERLAY*, CAPTION?, FIGTEXT, CREDIT?) -(FIG|IMG)>

<!ATTLIST FIG

        %attrs;

        %needs;                  -- for control of text flow --
        src  %URI;  #REQUIRED    -- URI of document to embed --
        %url.link;               -- standard link attributes --
        %block.align;            -- horizontal alignment --
```

```
    noflow (noflow) #IMPLIED -- noflow around figure --
    width  NUMBER #IMPLIED   -- desired width in units --
    height NUMBER #IMPLIED   -- desired height in units --
    units (en|pixels) pixels -- specifies units as en's or pixels --
    imagemap (%URI) #IMPLIED -- pass background clicks to server --
    >

<!ELEMENT FIGTEXT O O %body.content -- dummy element -->

<!--
  Figure overlays. When combined with local caching, overlays
  provide a cheap way of modifying a larger base image sent as
  part of a previous page.
-->

<!ELEMENT OVERLAY - O EMPTY -- image overlay -->
<!ATTLIST OVERLAY
    src  %URI;  #REQUIRED    -- URI of image overlay --
    %url.link;               -- standard link attributes --
    units (en|pixels) pixels -- specifies units as en's or pixels --
    x     NUMBER   0         -- offset from left in units --
    y     NUMBER   0         -- offset from top in units --
    width  NUMBER #IMPLIED   -- desired width in units --
    height NUMBER #IMPLIED   -- desired height in units --
```

```
        imagemap (%URI) #IMPLIED -- pass background clicks to server --

        >


<!ELEMENT CREDIT - - (%text;)* -- source of image -->

<!ATTLIST CREDIT

        %attrs;

        >


<!--======================== Notes
==========================================-->


<!--

The NOTE element is used for admonishments. The CLASS attribute

is used to differentiate NOTE's, e.g. Note, Caution or Warning.

-->


<!ELEMENT NOTE - - %body.content; -- admonishment -->

<!ATTLIST NOTE

        %attrs;

        src %URI;   #IMPLIED  -- URI of custom graphic --

        %url.link;            -- standard link attributes --

        %needs; -- for control of text flow --

        >


<!--======================== Footnotes
======================================-->
```

```
<!--
Typically rendered as popup note. These elements are referenced
by hypertext links specified with the anchor element.
-->
```

```
<!ELEMENT FN - - %body.content;>

<!ATTLIST FN %attrs;>


<!--======================= Math
=========================================-->


<!-- Use &thinsp; &emsp; etc for greater control of spacing. -->


<!-- Subscripts and Superscripts


  <SUB> and <SUP> are used for subscripts and superscripts.


                                    i j

     X <SUP>i</SUP>Y<SUP>j</SUP>  is   X  Y


  i.e. the space following the X disambiguates the binding.

  The align attribute can be used for horizontal alignment,

  e.g. to explicitly place an index above an element:

                                        i

       X<sup align=center>i</sup>  produces  X


Short references are defined for superscripts, subscripts and boxes

to save typing when manually editing HTML math, e.g.


     x^2^    is mapped to   x<sup>2</sup>
```

```
    y_z_    is mapped to   y<sub>z</sub>

    {a+b}   is mapped to   <box>a + b</box>


Note that these only apply within the MATH element and can't be

used in normal text!

-->

<!ENTITY REF1   STARTTAG   "SUP">

<!ENTITY REF2   ENDTAG     "SUP">

<!ENTITY REF3   STARTTAG   "SUB">

<!ENTITY REF4   ENDTAG     "SUB">

<!ENTITY REF5   STARTTAG   "BOX">

<!ENTITY REF6   ENDTAG     "BOX">


<!USEMAP MAP1   MATH>

<!USEMAP MAP2   SUP>

<!USEMAP MAP3   SUB>

<!USEMAP MAP4   BOX>


<!SHORTREF MAP1 "^" REF1

           "_" REF3

           "{" REF5 >


<!SHORTREF MAP2 "^" REF2

           "_" REF3

           "{" REF5 >
```

```
<!SHORTREF MAP3 "_" REF4

          "^" REF1

          "{" REF5 >
```

```
<!SHORTREF MAP4 "}" REF6

            "^" REF1

            "_" REF3

            "{" REF5 >


<!--

 The inclusion of %math and exclusion of %notmath is used here

 to alter the content model for the B, SUB and SUP elements,

 to limit them to formulae rather than general text elements.

-->


<!ENTITY % mathvec "VEC|BAR|DOT|DDOT|HAT|TILDE" -- common accents -->

<!ENTITY % mathface "B|T|BT" -- control of font face -->

<!ENTITY % math "BOX|ABOVE|BELOW|%mathvec|ROOT|SQRT|ARRAY|SUB|SUP|
%mathface">

<!ENTITY % formula "#PCDATA|%math">


<!ELEMENT MATH - - (#PCDATA)* -(%notmath) +(%math)>

<!ATTLIST MATH

      id     ID     #IMPLIED

      class    NAMES   #IMPLIED -- e.g. class=chem -->


<!-- The BOX element acts as brackets. Delimiters are optional and

   stretch to match the height of the box. The OVER element is used
```

when you want a line between numerator and denominator. This line is suppressed with the alternative ATOP element. CHOOSE acts like ATOP but adds enclosing round brackets as a convenience for binomial coefficients. Note the use of { and } as shorthand for <BOX> and </BOX> respectively:

```
                  1 + X
{1 + X<OVER>Y}  is  _______
                    Y
```

```
                       a + b
{a + b<ATOP>c - d} is
                       c - d
```

The delimiters are represented using the LEFT and RIGHT elements as in:

```
{[<LEFT>x + y<RIGHT>]}   is   [ x + y ]
{(<LEFT>a<RIGHT>]}       is   (a]
{||<LEFT>a<RIGHT>||}     is   || a ||
```

Use &lbrace; and &rbrace; for "{" and "}" respectively as these symbols are used as shorthand for BOX, e.g.

```
{&lbrace;<LEFT>a+b<RIGHT>&rbrace;}  is  {a+b}
```

You can stretch definite integrals to match the integrand, e.g.

```
{&int;<SUB>a</SUB><SUP>b</SUP><LEFT>{f(x)<over>1+x} dx}
```

```
          b
          /  f(x)
          | ----- dx
          / 1 + x
          a

    Note the complex content model for BOX is a work around
    for the absence of support for infix operators in SGML.

    You can get oversize delimiters with the SIZE attribute,
    for example <BOX SIZE=large>(<LEFT>...<RIGHT>)</BOX>

    Note that the names of common functions are recognized
    by the parser without the need to use "&" and ";" around
    them, e.g. int, sum, sin, cos, tan, ...
-->

<!ELEMENT BOX - - ((%formula)*, (LEFT, (%formula)*)?,
                ((OVER|ATOP|CHOOSE), (%formula)*)?,
                (RIGHT, (%formula)*)?)>
<!ATTLIST BOX
       size  (normal|medium|large|huge) normal -- oversize delims -->
```

```
<!ELEMENT (OVER|ATOP|CHOOSE|LEFT|RIGHT) - O EMPTY>


<!-- Horizontal line drawn ABOVE contents

     The symbol attribute allows authors to supply

     an entity name for an accent, arrow symbol etc.

     Generalisation of LaTeX's overline command.


     e.g. <above sym=ssmile>x</above>

     places an upwardly turning curve above the "x"

 -->


<!ELEMENT ABOVE - - (%formula)+>

<!ATTLIST ABOVE sym ENTITY #IMPLIED>


<!-- Horizontal line drawn BELOW contents

     The symbol attribute allows authors to

     supply an entity name for an arrow symbol etc.

     Generalisation of LaTeX's underline command.

 -->


<!ELEMENT BELOW - - (%formula)+>

<!ATTLIST BELOW sym ENTITY #IMPLIED>


<!-- Convenience tags for common accents:

     vec, bar, dot, ddot, hat and tilde

-->
```

```
<!ELEMENT (%mathvec) - - (%formula)+>
```

```
<!--
  T and BT are used to designate terms which should
  be rendered in an upright font (& bold face for BT)
-->

<!ELEMENT (T|BT) - - (%formula)+>
<!ATTLIST (T|BT) class NAMES #IMPLIED>

<!-- Roots  e.g. <ROOT>3<OF>1+x</ROOT> -->

<!ELEMENT ROOT - - ((%formula)+, OF, (%formula)+)>
<!ELEMENT OF - O (%formula)* -- what the root applies to -->

<!ELEMENT SQRT - - (%formula)* -- square root convenience tag -->

<!-- LaTeX like arrays. The COLDEF attribute specifies
     a single capital letter for each column determining
     how the column should be aligned, e.g. coldef="CCC"

        "L"    left
        "C"    center
        "R"    right

     An optional separator letter can occur between columns
```

```
    and should be one of + - or =, e.g. "C+C+C+C=C".
    Whitespace within coldef is ignored. By default, the
    columns are all centered.

    The ALIGN attribute alters the vertical position of the
    array as compared with preceding and following expressions.

    Use LDELIM and RDELIM attributes for delimiter entities.
    When the LABELS attribute is present, the array is
    displayed with the first row and the first column as
    labels displaced from the other elements. In this case,
    the first element of the first row should normally be
    left blank.

    Use &vdots; &cdots; and &ddots; for vertical, horizontal
    and diagonal ellipsis dots. Use &dotfill; to fill an array
    cell with horizontal dots (e.g. for a full row).
    Note &ldots; places the dots on the baseline, while &cdots;
    places them higher up.
-->

<!ELEMENT ARRAY - - (ROW)+>
<!ATTLIST ARRAY
      align (top|middle|bottom) middle -- vertical alignment --
      coldef  CDATA   #IMPLIED  -- column alignment and separator --
      ldelim  CDATA   #IMPLIED  -- stretchy left delimiter --
```

```
rdelim  CDATA   #IMPLIED  -- stretchy right delimiter --

labels (labels) #IMPLIED  -- TeX's \bordermatrix style -->
```

```
<!ELEMENT ROW - O (ITEM)*>

<!ELEMENT ITEM - O (%formula)*>

<!ATTLIST ITEM

        align   CDATA  #IMPLIED  -- override coldef alignment --

        colspan NUMBER 1         -- merge columns as per TABLE --

        rowspan NUMBER 1         -- merge rows as per TABLE -->


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


<![ %HTML.Deprecated [

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

                              & META* & LINK* & RANGE* & NEXTID?">

]]>


<!ENTITY % head.nextid "">


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

                              & META* & LINK* & RANGE*">


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


<!ELEMENT LINK - O EMPTY>

<!ATTLIST LINK

        href %URI #REQUIRED
        %linkExtraAttributes; >


<!ELEMENT RANGE - O EMPTY>
<!ATTLIST RANGE
        id    ID    #IMPLIED  -- for naming marked range --
        class NAMES #IMPLIED  -- for subclassing --
        from  IDREF #REQUIRED -- start of marked range --
        until IDREF #REQUIRED -- end of marked range --
        >


<!ELEMENT ISINDEX - O EMPTY>
<!ATTLIST ISINDEX
        href   %URI  #IMPLIED -- server handling queries --
        prompt CDATA #IMPLIED -- prompt message -->


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


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


    The image is deferenced to


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

```
-->

<!ELEMENT BASE - O EMPTY>
```

```
<!ATTLIST BASE
        id   ID    #IMPLIED
        href %URI; #REQUIRED
        >

<![ %HTML.Deprecated [
    <!ELEMENT NEXTID - O EMPTY>
    <!ATTLIST NEXTID N CDATA #REQUIRED>
]]>

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

<!--
    A style sheet can be associated with the document using the
    LINK element, e.g. <LINK rel=style href="housestyle.dsssl">.
    Style overrides can be placed in the document head using the
    STYLE element, e.g.

        <style notation=dsssl-lite>
```

```
          dsss-lite stuff
        </style>

    Later on in the document you can use:

        <h2 class=bigcaps>Header with bigger than normal capitals</h2>
        <p class=abstract>A paragraph with a unique style of its own
        ...

    Statements in the given style notation

    The tag names, class and id attributes are used in the style sheet
    notation to describe how to render matching elements.
-->

<!ENTITY % style-notations "dsssl-lite | w3c-style">
<!NOTATION dsssl-lite PUBLIC
      "ISO/IEC 10179:1995//NOTATION DSSSL Style Language//EN">

<!NOTATION w3c-style PUBLIC "IETF/RFC nnn/W3C Style Language//EN">

<!ELEMENT STYLE - O (#PCDATA)>
<!ATTLIST STYLE
    notation NOTATION (%style-notations;) #REQUIRED
>
```

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

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

```
<!ELEMENT HTML O O  (%html.content)>

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


<!-- suggested roles are: TOC, DOC, DOCPART, HITLIST, DIALOG -->


<!ATTLIST HTML

        %version.attr;      -- report DTD version to application --

        urn   CDATA #IMPLIED -- universal resource name for this document --

        class NAMES #IMPLIED -- role of this document, eg table of contents --

        >


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

Glossary of Terms

Pasted from HTML 2.0 spec, this is now under revision ...

The HTML specification uses these words with precise meanings:

attribute

A characteristic quality of an element, other than type or content.

browser

A tool used to read electronic books.

document

For the purposes of this standard, an HTML instance.

element

A component of the hierarchical structure defined by the document type definition; it is identified in a document instance by descriptive markup, usually a start-tag and an end-tag.

HTML

HyperText Markup Language.

HTTP

A generic stateless object-oriented protocol, which may be used for many similar tasks by extending the commands, or "methods", used. For example, you might use HTTP for name servers and distributed object-oriented systems, With HTTP, the negotiation of data representation allows systems to be built independent of the development of new representations. For more information see: http://www.w3.org/hypertext/WWW/Protocols/Overview.html

markup

Text added to the data of a document to convey information about it. There are four different kinds of markup: descriptive markup (tags), references, markup declarations, and processing instructions.

MIME

Multipurpose Internet Mail Extensions as defined in Mechanisms for Specifying and Describing the Format of Internet Message Bodies, 09/23/1993. (Pages=81) (Format=.txt, .ps) (Obsoletes RFC1341) (Updated by RFC1590).

representation

The encoding of information for interchange. For example, HTML is a representation of hypertext.

rendering

Formatting and presenting information to human readers.

SGML

Standard Generalized Markup Language as defined in ISO 8879:1986, Information Processing Text and Office Systems.

SGMLS

An SGML parser by James Clark, jjc@jclark.com, derived from the ARCSGML parser materials which were written by Charles F. Goldfarb. The source is available at ftp.ifi.uio.no/pub/SGML/SGMLS.

tag

Descriptive markup. There are two kinds of tags; start-tags and end-tags.

URI

Universal Resource Identifiers. Available by anonymous FTP as Postscript (www.w3.org/pub/www/doc/url.ps) or text (www.w3.org/pub/www/doc/url.txt)

W3

The World-Wide Web, a global information initiative. For bootstrap information, telnet www.w3.org or find documents at ftp://www.w3.org/pub/www/doc

References

Under revision ..

The HTML specification cites these works:

HTTP

HTTP: A Protocol for Networked Information. This document is available at http://www.w3.org/WWW/Protocols/HTTP/HTTP2.html.

MIME

N. Borenstein, N. Freed, MIME (Multipurpose Internet Mail Extensions) Part One: Mechanisms for Specifying and Describing the Format of Internet Message Bodies, 09/23/1993. (Pages=81) (Format=.txt, .ps) (Obsoletes RFC1341) (Updated by RFC1590).

SGML

ISO 8879:1986, Information Processing Text and Office Systems Standard Generalized Markup Language (SGML).

SGMLS

An SGML parser by James Clark, jjc@jclark.com, derived from the ARCSGML parser materials which were written by Charles F. Goldfarb. The source is available at ftp.ifi.uio.no/pub/SGML/SGMLS.

URI

Universal Resource Identifiers. RFCxxx. Available by anonymous FTP as Postscript (info.cern.ch/pub/www/doc/url.ps) or text (info.cern.ch/pub/www/doc/url.txt)

W3

The World-Wide Web , a global information initiative. For bootstrap information, telnet info.cern.ch or find documents by ftp://info.cern.ch/pub/www/doc.

Acknowledgments

Pasted from HTML 2.0 spec, this section is under revision ...

The HTML document type was designed by Tim Berners-Lee at CERN as part of the 1990 World-Wide Web project. In 1992, Dan Connolly wrote the HTML Document Type Definition (DTD) and a brief HTML specification.

Since 1993, a wide variety of Internet participants have contributed to the evolution of HTML. NCSA Mosaic played a particularly important role in establishing HTML. Mosaic pioneered the addition of in-line images, image maps, nested lists and fill-out forms (derived from work on HTML+). Minor variations in the way extensions were supported by different browsers eventually led to the setting up of the HTML working group. The HTML 2.0 specification sets out a definitive standard for HTML, formalizing the de facto situation during 1994.

HTML+ was the result of my work on possible directions for extending HTML to meet the needs of information providers, e.g. to support forms, tables, text flow around figures and math. This work has now culminated in the current HTML 3.0 specification, which adds a range of important new features to HTML while preserving simplicity and backwards compatibility with existing documents.

I would like to express my special thanks to members of the Internet community on the www-talk, www-html and html-wg mailing lists; to people who have written to me in person, and to members of the SGML-Open who have been very supportive of the Web initiative. Thanks also to Hewlett Packard for funding my work on HTML.

Particular thanks are due to:

* Terry Allen; O'Reilly & Associates; terry@ora.com

* Marc Andreessen; Netscape Communications Corp; marca@netscape.com

* Eric Bina; Netscape Communications Corp; ebina@netscape.com

* Paul Burchard; The Geometry Center, University of Minnesota; burchard@geom.umn.edu

* James Clark; jjc@jclark.com

* Daniel W. Connolly; HaL Computer Systems; connolly@hal.com

* Stephen DeRose; EBT; ??? steve@ebt.com

* Roy Fielding; University of California, Irvine;

fielding@ics.uci.edu

* Jay Glicksman; Enterprise Integration Technology; jay@eit.com

* Eduardo Gutentag; Sun Microsystems; eduardo@Eng.Sun.com

* Bill Hefley; Software Engineering Institute, Carnegie Mellon University; weh@sei.cmu.edu

* Chung-Jen Ho; Xerox Corporation; cho@xsoft.xerox.com

* Mike Knezovich; Spyglass, Inc.; mike@spyglass.com

* Tim Berners-Lee; CERN; timbl@info.cern.ch

* Tom Magliery; NCSA; mag@ncsa.uiuc.edu

* Murray Maloney; SCO Canada; murray@sco.com

* Larry Masinter; Xerox Palo Alto Research Center; masinter@parc.xerox.com

* Karen Olson Muldrow; HaL Computer Systems; karen@hal.com

* Bill Perry, Spry, Inc., wmperry@spry.com

* E. Corprew Reed; Cold Spring Harbor Laboratory; corp@cshl.org

* Yuri Rubinsky; SoftQuad, Inc.; yuri@sq.com

* Eric Schieler; Spyglass, Inc.; eschieler@spyglass.com

* Eric Severson; Avalanche, Inc.; ??? severson@avalanche.com

* Eric W. Sink; Spyglass, Inc.; eric@spyglass.com

* Stuart Weibel; OCLC Office of Research; weibel@oclc.org

* Chris Wilson; Spry, Inc.; cwilson@spry.com

Dave Raggett <dsr@w3.org>, February 1995.