HTML Working Group
INTERNET-DRAFT
<draft-ietf-html-spec-02.txt>
Expires: In six months

T. Berners-Lee
MIT/W3C
D. Connolly
May 6, 1995

Hypertext Markup Language - 2.0

CONTENTS

Status of this Memo

ABSTRACT

The Hypertext Markup Language (HTML) is a simple markup language used to create hypertext documents that are platform independent. HTML documents are SGML documents with generic semantics that are appropriate for representing information from a wide range of domains. 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. This specification roughly corresponds to the capabilities of HTML in common use prior to June 1994. HTML is an application of ISO Standard 8879:1986 Information Processing Text and Office Systems; Standard Generalized Markup Language (SGML).

The `"text/html; version=2.0"' Internet Media Type (RFC 1590) and MIME Content Type (RFC 1521) is defined by this specification.

1. Introduction

The HyperText Markup Language (HTML) is a simple data format 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 domains.

1.1. Scope

HTML has been in use by the World-Wide Web (WWW) global information initiative since 1990. This specification corresponds to the capabilities of HTML in common use prior to June 1994 and referred to as ``HTML 2.0''.

HTML is an application of ISO Standard 8879:1986 _Information Processing Text and Office Systems; Standard Generalized Markup Language_ (SGML). The HTML Document Type Definition (DTD) is a formal definition of the HTML syntax in terms of SGML.

This specification also defines HTML as an Internet Media Type[IMEDIA] and MIME Content Type[MIME] called `text/html', or `text/html; version=2.0'. As such, it defines the semantics of the HTML syntax and how that syntax should be

interpreted by user agents.

1.2. Conformance

This specification governs the syntax of HTML documents and the behaviour of HTML user agents.

1.2.1. Documents

A document is a conforming HTML document only if:

* It is a conforming SGML document, and it conforms to the HTML DTD (see 11.1, "HTML DTD")
* It conforms to the application conventions in this specification. For example, the value of the `HREF' attribute of the <A> element must conform to the URI syntax.
* Its document character set includes ISO-8859-1 and agrees with ISO10646; that is, each code position listed in 14.1, "The ISO-8859-1 Coded Character Set" is included, and each code position in the document character set is mapped to the same character as ISO10646 designates for that code position.

NOTE - The document character set is somewhat independent of the character encoding scheme used to represent a document. For example, the ISO-2022-JP character encoding scheme can be used for HTML documents, since its repertoire is a subset of the ISO10646 repertoire. The crititcal distinction is that numeric character references agree with ISO10646 regardless of how the document is encoded.

NOTE - There are a number of syntactic idioms that are not supported or are supported inconsistently in some historical user agent implementations. These idioms are called out in notes like this throughout this specification.

HTML documents should not contain these idioms, at least until such time as support for them is widely deployed.

The HTML DTD defines a standard HTML document type and several variations, based on feature test entities:

HTML.Recommended
: 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 those features.

For example, in order to preserve the structure of a document, an editing user agent may translate HTML documents to the recommended subset, or it may require that the documents be in the recommended subset for import.

HTML.Deprecated

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.

Documents generated by tranlation software or editing software should not contain these idioms.

1.2.2. User Agents

An HTML user agent conforms to this specification if:

* It parses the characters of an HTML document into data characters and markup as per [SGML].
* It supports the ISO-8859-1 character encoding scheme, and processes each character in the ISO Latin Alphabet Nr. 1 as specified in 5.1, "The ISO Latin 1 Character Repertoire".
NOTE - To support non-western writing systems, HTML user agents should support the Unicode-1-1-UTF-8 and Unicode-1-1-UCS-2 encodings and as much of the character repertoire of ISO10646 as is possible as well.
* It behaves identically for documents whose parsed token sequences are identical.
For example, comments and the whitespace in tags disappear during tokenization, and hence they do not influence the behaviour of conforming user agents.
* It allows the user to traverse (or at least attempt to traverse, resources permitting) all hyperlinks in an HTML document.
* It allows the user to express all form field values specified in an HTML document and to (attempt to) submit the values as requests to information services.

NOTE - In the interest of robustness and extensibility,

there are a number of widely deployed conventions for handling non-conforming documents. See 3.2.1, "Undeclared Markup Error Handling" for details.

2. HTML as an Application of SGML

HTML is an application of ISO Standard 8879:1986 - Standard Generalized Markup Language (SGML). SGML is a system for defining structured document types and markup languages to represent instances of those document types[SGML]. The public text -- DTD and SGML declaration -- of the HTML document type definition are provided in 11, "HTML Public Text".

The term _HTML_ refers to both the document type defined here and the markup language for representing instances of this document type.

2.1. SGML Documents

An HTML document is an SGML document; that is, a sequence of characters organized physically into a set of entities, and logically as a hierarchy of elements.

The first production of the SGML grammar separates an SGML document into three parts: an SGML declaration, a prologue, and an instance. For the purposes of this specification, the prologue is a DTD. This DTD describes another grammar: the start symbol is given in the doctype declaration; the terminals are data characters and tags, and the productions are determined by the element declarations. The instance must conform to the DTD, that is, it must be in the language defined by this grammar.

The SGML declaration determines the lexicon of the grammar. It specifies the document character set, which determines a character repertoire that contains all characters that occur in all text entities in the document, and the code positions associated with those characters.

The SGML declaration also specifies the syntax-reference character set of the document, and a few other parameters that bind the abstract syntax of SGML to a concrete syntax. This concrete syntax determines how the sequence of characters of the document is mapped to a sequence of terminals in the grammar of the prologue.

For example, consider the following document:

```
<!DOCTYPE html PUBLIC "-//IETF//DTD HTML 2.0//EN">
```

```
<title>Parsing Example</title>
<p>Some text. <em>&#42;wow&#42;</em></p>
```

An HTML user agent should use the SGML declaration is given in 11.2, "SGML Declaration for HTML". According to the document character set there,`*' refers to an asterisk character.

The instance above is regarded as the following sequence of terminals:

1. TITLE start-tag
2. data characters: ``Parsing Example''
3. TITLE end-tag
4. P start-tag
5. data characters ``Some text. ''
6. EM start-tag
7. ``*wow*''
8. EM end-tag

The start symbol of the DTD grammar is HTML, and the productions are given in the public text identified by `-//IETF//DTD HTML 2.0//EN' (11.1, "HTML DTD"). Hence the terminals above parse as:

```
   HTML
    |
    \-HEAD
    |  |
    |  \-TITLE
    |      |
    |      \-<TITLE>
    |      |
    |      \-"Parsing Example"
    |      |
    |      \-</TITLE>
    |
    \-BODY
      |
      \-P
        |
        \-<P>
        |
        \-"Some text. "
        |
        \-EM
        |  |
        |  \-<EM>
        |  |
        |  \-"*wow*"
        |  |
```

```
|  \-</EM>
|
\-</P>
```

2.2. HTML Lexical Syntax

SGML specifies an abstract syntax and a reference concrete syntax. Aside from certain quantities and capacities (e.g. the limit on the length of a name), all HTML documents use the reference concrete syntax. In particular, all markup characters are in the ISO-646-IRV character repertoire. Data characters are drawn from the document character set (see 5, "Character Content").

A complete discussion of SGML parsing, e.g. the mapping of a sequence of characters to a sequence of tags and data is left to the SGML standard[SGML]. This section is only a summary.

2.2.1. Data Characters

Any sequence of characters that do not constitute markup (see 9.6 ``Delimiter Recognition'' of [SGML]) are mapped directly to strings of data characters. Some markup also maps to data character strings. Numeric character references also map to single-character strings, via the document character set. Each reference to one of the general entities defined in the HTML DTD also maps to a single-character string.

For example,

```
abc&lt;def    => "abc","<","def"
abc&#60;def   => "abc","<","def"
```

Note that the terminating semicolon is only necessary when the character following the reference would otherwise be recognized as markup:

```
abc &lt def     => "abc ","<"," def"
abc &#60 def    => "abc ","<"," def"
```

And note that an ampersand is only recognized as markup when it is followed by a letter or digit:

```
abc & lt def    => "abc & lt def"
abc & 60 def    => "abc & 60 def"
```

A useful technique for translating plain text to HTML is to replace each '<', '&', and '>' by an entity reference or

numeric character reference as follows:

CHARACTER	ENTITY REFERENCE	NUMERIC CHAR REF	CHARACTER DESCRIPTION
&	&amp;	&#38;	Ampersand
<	&lt;	&#60;	Less than
>	&gt;	&#62;	Greater than

NOTE - There are SGML mechanisms, CDATA and RCDATA, to allow most `<', `>', and `&' characters to be entered without the use of entity references. Because these features tend to be used and implemented inconsistently, and because they conflict with techinques for reducing HTML to 7 bit ASCII for transport, they are not used in this version of the HTML DTD.

2.2.2. Tags

Tags delimit elements such as 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 `>'; end tags are delimited by `</' and `>'. An example is:

<H1>This is a Heading</H1>

Some elements only have a start-tag without an end-tag. For example, to create a line break, you use the `<BR>' tag. Additionally, the end tags of some other elements, such as Paragraph (`</P>'), List Item (`</LI>'), Definition Term (`</DT>'), and Definition Description (`<DD>') elements, may be omitted.

The content of an element is a sequence of data character strings and nested elements. Some elements, such as anchors, cannot be nested. Anchors and character highlighting may be put inside other constructs. See the HTML DTD, 11.1, "HTML DTD" for full details.

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.

2.2.3. Names

A name consists of a letter followed by up to 71 letters, digits, periods, or hyphens. Element names are not case sensitive, but entity names are. For example, `<BLOCKQUOTE>', `<BlockQuote>', and `<blockquote>' are equivalent, whereas `&' is different from `&'.

In a start-tag, the element name must immediately follow the tag open delimiter `<'.

2.2.4. 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, though some attributes may be just a value. White space is allowed around the equal sign.

The value of the attribute may be either:

* A string literal, delimited by single quotes or double quotes and not containing any occurrences of the delimiting character.
* A name token (a sequence of letters, digits, periods, or hyphens)

In this example, img is the element name, `src' is the attribute name, and `http://host/dir/file.gif' is the attribute value:

<img src="http://host/dir/file.gif">

NOTE - Some historical implementations consider any occurrence of the `>' character to signal the end of a tag. For ompatibility with such implementations, when `>' appears in an attribute value, it should be represented with a numeric character reference, such as in: `<IMG SRC="eq1.jpg" alt="a>b">'.

A useful technique for computing an attribute value literal for a given string is to replace each quote and space character by an entity reference or numeric character reference as follows:

CHARACTER	ENTITY REFERENCE	NUMERIC CHAR REF	CHARACTER DESCRIPTION
TAB			Tab
LF	
Line Feed
CR	
Carriage Return
			Space
"	"	"	Quotation mark
&	&	&	Ampersand

For example:

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

> NOTE - Some historical 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.

Note that the SGML declaration in section 13.3 limits the length of an attribute value to 1024 characters.

Attributes such as ISMAP and COMPACT, may be written using a minimized syntax. The markup:

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

can be written using a minimized syntax:

```
<UL COMPACT>
```

> NOTE - Some historical implementations only understand the minimized syntax.

2.2.5. Comments

To include comments in an HTML document that will be eliminated in the mapping to terminals, 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: html-sgml.sgm,v 1.4 1995/05/06 01:44:46 connolly Exp $ -->
</HEAD>
```

> NOTE - Some historical HTML implementations incorrectly consider any `>' character to be the termination of a comment.

2.2.6. Example HTML Document

```
<!DOCTYPE HTML PUBLIC "-//IETF//DTD HTML//EN">
<HTML>
<!-- Here's a good place to put a comment. -->
```

```
<HEAD>
<TITLE>Structural Example</TITLE>
</HEAD><BODY>
<H1>First Header</H1>
<P>This is a paragraph in the example HTML file. Keep in mind
that the title does not appear in the document text, but that
the header (defined by H1) does.</P>
<OL>
<LI>First item in an ordered list.
<LI>Second item in an ordered list.
  <UL COMPACT>
  <LI> Note that lists can be nested;
  <LI> Whitespace may be used to assist in reading the
       HTML source.
  </UL>
<LI>Third item in an ordered list.
</OL>
<P>This is an additional paragraph. Technically, end tags are
not required for paragraphs, although they are allowed. You can
include character highlighting in a paragraph. <EM>This sentence
of the paragraph is emphasized.</EM> Note that the &lt;/P&gt;
end tag has been omitted.
<P>
<IMG SRC ="triangle.xbm" alt="Warning: ">
Be sure to read these <b>bold instructions</b>.
</BODY></HTML>
```

3. HTML as an Internet Media Type

An HTML user agent allows users to interact with resources which have HTML representations. At a minimum, it must allow users to examine and navigate the content of HTML documents. HTML user agents should be able to preserve all formatting distinctions represented in an HTML document, and be able to simultaneously present resources referred to by IMG elements. (they may ignore some formatting distinctions or IMG resources at the request of the user). Conforming HTML user agents should support form entry and submission.

3.1. text/html media type

This specification defines the Internet Media Type[IMEDIA] (formerly referred to as the Content Type[MIME]) called `text/html'. The following is to be registered with [IANA].

Media Type name

text

Media subtype

name
html

Required
parameters
none

Optional
parameters
version, charset

Encoding
considerations
any encoding is allowed

Security
considerations
see 3.3, "Security Considerations"

The optional parameters are defined as follows:

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 of the HTML DTD. This specification defines version 2.0. There is no default.

Charset
The charset parameter (as defined in section 7.1.1 of RFC 1521[MIME]) may be given to specify the character encoding scheme used to represent the HTML document as a sequence of octets. The default value is outside the scope of this specification; but for example, the default is US-ASCII in the context of MIME mail, and ISO-8859-1 in the context of HTTP.

3.2. HTML Document Representation

A message entity with a content type of `text/html' represents an HTML document, consisting of a single text entity. The `charset' parameter (whether implicit or explicit) identifies a character encoding scheme. The text entity consists of the characters determined by this character encoding scheme and the octets of the body of the message entity.

3.2.1. Undeclared Markup Error Handling

To facilitate experimentation and interoperability between implementations of various versions of HTML, the installed base of HTML user agents supports a superset of the HTML 2.0 language by reducing it to HTML 2.0: markup in the form of a start-tag or end-tag whose generic identifier is not declared is mapped to nothing during tokenization. Undeclared attributes are treated similarly. The entire attribute specification of an unknown attribute (i.e., the unknown attribute and its value, if any) should be ignored. On the other hand, references to undeclared entities should be treated as data characters.

For example:

```
<div class=chapter><h1>foo</h1><p>...</div>
  => <H1>,"foo",</H1>,<P>,"..."
xxx <P ID=z23> yyy
  => "xxx ",<P>," yyy
Let &alpha; and &beta; be finite sets.
  => "Let &alpha; and &beta; be finite sets."
```

Support for notifying the user of such errors is encouraged.

Information providers are warned that this convention is not binding: unspecified behavior may result, as such markup is not conforming to this specification.

3.2.2. Conventional Representation of Newlines

SGML specifies that a text entity is a sequence of records, each beginning with a record start character and ending with a record end character (code positions 10 and 13 respectively). (section 7.6.1, ``Record Boundaries'' in [SGML])

[MIME] specifies that a body of type `text/*' is a sequence of lines, each terminated by CRLF, that is octets 10, 13.

In practice, HTML documents are frequently represented and transmitted using an end of line convention that depends on the conventions of the source of the document; frequently, that representation consists of CR only, LF only, or CR LF combination. Hence the decoding of the octets will often result in a text entity with some missing record start and record end characters.

Since there is no ambiguity, HTML user agents are encouraged to infer the missing record start and end characters.

An HTML user agent should treat end of line in any of its variations as a word space in all contexts except preformatted text. Within preformatted text, an HTML user agent should expect to treat any of the three common representations of end-of-line as starting a new line.

3.3. Security Considerations

Anchors, embedded images, and all other elements which contain URIs as parameters may cause the URI to be dereferenced in response to user input. In this case, the security considerations of the URI specification apply.

The widely deployed methods for submitting forms requests -- HTTP and SMTP -- provide little assurance of confidentiality. Information providers who request sensitive information via forms -- especially by way of the `PASSWORD' type input field -- should be aware and make their users aware of the lack of confidentiality.

>

4. Document Structure Elements

To identify information as an HTML document conforming to this specification, each document should start with the prologue:

```
<!DOCTYPE HTML PUBLIC "-//IETF//DTD HTML 2.0//EN">
```

> NOTE - If the body of a text/html body part does not begin with a document type declaration, an HTML user agent should infer the above document type declaration.

HTML user agents are required to support the above document type declaration, the following document type declarations, and no others.

```
<!DOCTYPE HTML PUBLIC "-//IETF//DTD HTML 2.0 Strict//EN">
<!DOCTYPE HTML PUBLIC "-//IETF//DTD HTML Strict//EN">
```

In particular, they may support other formal public identifiers, or document types altogether. They may support an internal declaration subset with supplemental entity, element, and other markup declarations, or they may not.

4.1. HTML Document Element

<HTML> ... </HTML> Level 0

The HTML document element is organized as a head and a body, much like a memo or a mail message. Within the head, you can specify the title and other information about the document. Within the body, you can structure text into paragraphs and lists, as well as highlight phrases and create links, using HTML elements.

NOTE - The start and end tags for HTML, Head, and Body elements are omissible; however, this is not recommended since the head/body structure allows an implementation to determine certain properties of a document, such as the title, without parsing the entire document.

<

4.2. Head

<HEAD> ... </HEAD> Level 0

The head of an HTML document is an unordered collection of information about the document. The Title element is required.

```
<HEAD>
<TITLE>Introduction to HTML</TITLE>
</HEAD>
```

4.3. Body

<BODY> ... </BODY> Level 0

The Body element identifies the body component of an HTML document. Specifically, the body of a document may contain links, text, and formatting information within <BODY> and </BODY> tags.

4.4. Title

<TITLE> ... </TITLE> Level 0

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 history lists and as a label for the window displaying the document. Unlike headings, titles are not rendered in the text of a document itself.

The Title element must occur within the head of the

document, and must not contain anchors, paragraph tags, or highlighting. Only one title is allowed in a document.

> NOTE - The length of a title is not limited; however, long titles may be truncated in some applications. To minimize this possibility, titles should be 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 ``Introduction to HTML Elements.''

4.5. Base

<BASE> Level 0

The Base element allows the URI of the document itself to be recorded in situations in which the document may be read out of context. URIs within the document may be in a ``partial'' form relative to this base address[RELURL].

The Base element has one attribute, HREF, which identifies the absolute base URI.

4.6. Isindex

<ISINDEX> Level 0

The Isindex element tells the interpreter that the document is an index. This means that the reader may request a keyword search on the resource by adding a question mark to the end of the document address, followed by a list of keywords separated by plus signs.

The Isindex element is usually generated by the network server from which the document was obtained via a URI. The server must have a search engine that supports this feature for the resource. If the document URI is unknown to the interpreter, <isindex> must be ignored.

4.7. Link

<LINK> Level 0

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.

Typical uses are to indicate authorship, related indexes and glossaries, older or more recent versions, etc. Links can indicate a static tree structure in which the document was authored by pointing to a ``parent'' and ``next'' and ``previous'' document, for example.

Servers may also allow links to be added by those who do not have the right to alter the body of a document.

4.8. Meta

<META> Level 0

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

Although it is generally preferable to use named elements which have well-defined semantics for each type of metainformation (e.g. TITLE), the META element is provided for situations where strict SGML parsing is necessary and the local DTD is not extensible. HTML interpreters may use the META element's content if they recognize and understand the semantics identified by the NAME or HTTP-EQUIV attributes, and may treat the content as metainformation (and not render it) even when they do not recognize the name.

In addition, HTTP servers may wish to read the content of the document HEAD to generate header fields corresponding to any elements defining a value for the attribute HTTP-EQUIV. Note, however, that the method by which the server extracts document metainformation is not part of this specification, nor can it be assumed by authors that any given server will be capable of extracting it. The META element only provides an extensible mechanism for identifying and embedding document metainformation - how it may be used is up to the individual server implementation and the HTML interpreter.

Attributes of the META element:

HTTP-EQUIV
: This attribute binds the element to an HTTP header field. It means that if you know the semantics of the HTTP header field named by this attribute, then you can process the contents based on a well-defined syntactic mapping, whether or not your DTD tells you anything about it. HTTP header

field names are not case sensitive. If not present, the attribute NAME should be used to identify this metainformation and the content should not be used within an HTTP response header.

NAME
Metainformation name. If the NAME attribute is not present, the name can be assumed to be equal to the value of HTTP-EQUIV.

CONTENT
The metainformation content to be associated with the given name. If multiple META elements are provided with the same name, their combined contents-concatenated as a comma-separated list-is the value associated with that name.

Examples

If the document contains:

<META HTTP-EQUIV="Expires"
      CONTENT="Tue, 04 Dec 1993 21:29:02 GMT">
<meta http-equiv="Keywords" CONTENT="Fred, Barney">
<META HTTP-EQUIV="Reply-to"
      content="fielding@ics.uci.edu (Roy Fielding)">

then the server (if so configured) may include the following headers:

Expires: Tue, 04 Dec 1993 21:29:02 GMT
Keywords: Fred, Barney
Reply-to: fielding@ics.uci.edu (Roy Fielding)

as part of the HTTP response to a GET or HEAD request for that document.

When the HTTP-EQUIV attribute is not present, the server should not generate an HTTP response header for the metainformation; e.g.,

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

would never generate an HTTP response header, but would still allow HTML interpreters to identify and make use of that metainformation.

The Meta element should never be used to define information that should be associated with an existing HTML element. An example of an inappropriate use of the Meta element is:

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

Do not name an HTTP-EQUIV equal to a response header that should normally only be generated by the HTTP server. Example names that are inappropriate include ``Server'', ``Date'', and ``Last-modified'' - the exact list of inappropriate names is dependent on the particular server implementation. We recommend that servers ignore any META elements which specify HTTP-equivalents which are equal (case-insensitively) to their own reserved response headers.

4.9. Nextid

<NEXTID> Level 0

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

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

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.

HTML interpreters may ignore the Nextid element. Support for the Nextid element does not impact HTML interpreters in any way.

5. Character Content

An HTML user agent should present the body of an HTML document as a collection of typeset paragraphs and preformatted text. Except for the <PRE> element, each block structuring element is regarded as a paragraph by taking the data characters in its content and the content of its descendant elements, concatenating them, and splitting the result into words, separated by space, tab, or record end characters (and perhaps hyphen characters). The sequence of words is typeset as a paragraph by breaking it into lines.

5.1. The ISO Latin 1 Character Repertoire

The minimum character repertoire supported by all conforming HTML user agents is Latin Alphabet Nr. 1, or simply Latin-1.

Latin-1 includes characters from most Western European languages, as well as a number of control characters. Latin-1 also includes a non-breaking space, a soft hyphen indicator, 93 graphical characters, 8 unassigned characters, and 25 control characters.

> NOTE - Use the non-breaking space and soft hyphen indicator characters is discouraged because support for them is not widely deployed.

In SGML applications, the use of control characters is limited in order to maximize the chance of successful interchange over heterogeneous networks and operating systems. In HTML, only three control characters are allowed: Horizontal Tab (HT, encoded as 9 decimal in US-ASCII and ISO-8859-1), Carriage Return, and Line Feed.

The HTML DTD references the Added Latin 1 entity set, to allow mnemonic representation of Latin 1 characters using only the widely supported ASCII character repertoire. For example:

Kurt Gödel was a famous logician and mathematician.

See 11.4.2, "ISO Latin 1 Character Entity Set" for a table of the ``Added Latin 1'' entities, and 14.1, "The ISO-8859-1 Coded Character Set" for a table of the code positions of ISO-8859-1.

6. Data Elements

6.1. Line Break

 Level 0

The Line Break element specifies that a new line must be started at the given point. A new line indents the same as that of line-wrapped text.

Example of use:

```
<P> Pease porridge hot<BR>
Pease porridge cold<BR>
Pease porridge in the pot<BR>
Nine days old.
```

6.2. Horizontal Rule

<HR> Level 0

A Horizontal Rule element is a divider between sections of text such as a full width horizontal rule or equivalent graphic.

Example of use:

```
<HR>
<ADDRESS>February 8, 1995, CERN</ADDRESS>
</BODY>
```

6.3. Image

<IMG> Level 0

The Image element is used to incorporate in-line graphics (typically icons or small graphics) into an HTML document. This element cannot be used for embedding other HTML text.

HTML interpreters that cannot render in-line images ignore the Image element unless it contains the ALT attribute. Note that some HTML interpreters can render linked graphics but not in-line graphics. If a graphic is essential, you may want to create a link to it rather than to put it in-line. If the graphic is not essential, then the Image element is appropriate.

The Image element, which is empty (no closing tag), has these attributes:

ALIGN
: The ALIGN attribute accepts the values TOP or MIDDLE or BOTTOM, which specifies if the following line of text is aligned with the top, middle, or bottom of the graphic.

ALT
: Optional text as an alternative to the graphic for rendering in non-graphical environments. Alternate text should be provided whenever the graphic is not rendered. Alternate text is mandatory for Level 0 documents. Example of use:

```
<IMG SRC="triangle.xbm" ALT="Warning:"> Be sure
to read these instructions.
```

ISMAP
: The ISMAP (is map) attribute identifies an image as an image map. Image maps are graphics in which certain regions are mapped to URIs. By clicking on different regions, different resources can be

accessed from the same graphic. Example of use:

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

SRC

The value of the SRC attribute is the URI of the document to be embedded; only images can be embedded, not HTML text. Its syntax is the same as that of the HREF attribute of the `<A>' tag. SRC is mandatory. Image elements are allowed within anchors.

Example of use:

```
<IMG SRC="triangle.xbm">Be sure to read these
instructions.
```

7. Character Format Elements

Character format elements are used to specify either the logical meaning or the physical appearance of marked text without causing a paragraph break. Like most other elements, character format elements include both opening and closing tags. Only the characters between the tags are affected:

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

Character format tags may be ignored by minimal HTML applications.

Character format tags are interpreted from left to right as they appear in the flow of text. Level 1 interpreters must render highlighted text distinctly from plain text. Additionally, EM content must be rendered as distinct from STRONG content, and B content must rendered as distinct from I content.

Character format elements may be nested within the content of other character format elements; however, HTML interpreters are not required to render nested character format elements distinctly from non-nested elements:

```
plain <B>bold <I>italic</I></B> may the rendered
the same as plain <B>bold </B><I>italic</I>
```

7.1. Semantic Format Elements

Note that typical renderings for semantic format elements

vary between applications. If a specific rendering is necessary - for example, when referring to a specific text attribute as in ``The italic parts are mandatory'' - a physical formating element can be used to ensure that the intended rendered is used where possible.

Note that different sematic elements may be rendered in the same way.

7.1.1. Citation

<CITE>...</CITE> Level 1

The Citation element specifies a citation, typically rendered as italics.

7.1.2. Code

<CODE> ... </CODE> Level 1

The Code element indicates an example of code, typically rendered in a monospaced font. This should not be confused with the Preformatted Text element.

7.1.3. Emphasis

<EM> ... </EM> Level 1

The Emphasis element indicates typographic emphasis, typically rendered as italics.

7.1.4. Keyboard

<KBD> ... </KBD> Level 1

The Keyboard element indicates text typed by a user, typically rendered in a monospaced font. This is commonly used in instruction manuals.

7.1.5. Sample

<SAMP> ... </SAMP> Level 1

The Sample element indicates a sequence of literal characters, typically rendered in a monospaced font.

7.1.6. Strong

<STRONG> ... </STRONG> Level 1

The Strong element indicates strong typographic emphasis, typically rendered in bold.

7.1.7. Variable

<VAR> ... </VAR> Level 1

The Variable element indicates a variable name, typically rendered as italic.

7.2. Physical Format Elements

Physical format elements are used to specify the format of marked text.

7.2.1. Bold

<B> ... </B> Level 1

The Bold element specifies that the text should be rendered in boldface, where available. Otherwise, an alternative mapping is allowed.

7.2.2. Italic

<I> ... </I> Level 1

The Italic element specifies that the text should be rendered in an italic font, where available. Otherwise, an alternative mapping is allowed.

7.2.3. Teletype

<TT> ... </TT> Level 1

The Teletype element specifies that the text should be rendered in a fixed-width typewriter font.

8. Hyperlink Elements

8.1. Anchor

<A> ... </A> Level 0

An anchor is a marked section of text that is the start and/or destination of a hypertext link. Anchor elements are defined by the `<A>' tag. The `<A>' tag accepts several attributes; at least one of the NAME and HREF attributes is required.

Attributes of the `<A>' tag:

8.1.1. HREF

If the HREF attribute is present, the text between the opening and closing anchor tags becomes hypertext. If this hypertext is selected by readers, they are moved to another document, or to a different location in the current document, whose network address is defined by the value of the HREF attribute.

Example:

```
See <A HREF="http://www.hal.com/">HaL</A>'s
information for more details.
```

In this example, selecting ``HaL'' takes the reader to a document at http://www.hal.com. The format of the network address is specified in the URI specification for print readers.

With the HREF attribute, the form HREF=``#identifier'' can refer to another anchor in the same document.

Example:

```
The <A HREF="#glossary">glossary</A> defines
terms used in this document.
```

In this example, selecting ``glossary'' takes the reader to another anchor (i.e., <A NAME=``glossary''>Glossary</A>) in the same document. The NAME attribute is described below. If the anchor is in another document, the HREF attribute may be relative to the document's address or the specified base address (see 4.5, "Base").

8.1.2. NAME

If present, the NAME attribute allows the anchor to be the target of a link. The value of the NAME attribute is an identifier for the anchor. Identifiers are arbitrary strings but must be unique within the HTML document.

Example of use:

```
<A NAME="coffee">Coffee</A> is an example of ...
... An example of this is <A HREF="#coffee">coffee</A>.
```

Another document can then make a reference explicitly to this anchor by putting the identifier after the address, separated by a hash sign:

```
<A HREF="drinks.html#coffee">
```

8.1.3. TITLE

The TITLE attribute is informational only. If present, the TITLE attribute should provide the title of the document whose address is given by the HREF attribute. The TITLE attribute is useful for at least two reasons. The HTML interpreter may display the title of the document prior to retrieving it, for example, as a margin note or on a small box while the mouse is over the anchor, or while the document is being loaded. Another reason is that documents that are not marked up text, such as graphics, plain text and Gopher menus, do not have titles. The TITLE attribute can be used to provide a title to such documents. When using the TITLE attribute, the title should be valid and unique for the destination document.

8.1.4. REL

The REL attribute gives the relationship(s) described by the hypertext link from the anchor to the target. The value is a whitespace-separated list of relationship names. Relationship names and their semantics will be registered by the W3 Consortium. The default relationship is void. The REL attribute is only used when the HREF attribute is present.

8.1.5. REV

The REV attribute is the same as the REL attribute, but the semantics of the link type are in the reverse direction. A link from A to B with REL=``X'' expresses the same relationship as a link from B to A with REV=``X''. An anchor may have both REL and REV attributes.

8.1.6. URN

If present, the URN attribute specifies a uniform resource name (URN) for a target document. The format of URNs is

under discussion (1995) by various working groups of the Internet Engineering Task Force.

8.1.7. METHODS

The METHODS attributes of anchors and links provide information about the functions that the user may perform on an object. These are more accurately given by the HTTP protocol when it is used, but it may, for similar reasons as for the TITLE attribute, be useful to include the information in advance in the link. For example, the HTML interpreter may chose a different rendering as a function of the methods allowed; for example, something that is searchable may get a different icon.

The value of the METHODS attribute is a whitespace-separated list of HTTP methods supported by the object for public use.

9. Block Structuring Elements

The following elements may be included in the body of an HTML document:

9.1. Paragraph

<P> ... </P> Level 0

The Paragraph element indicates a paragraph. The exact indentation, leading space, etc. of a paragraph is not defined and may be a function of other tags, style sheets, etc.

Typically, paragraphs are surrounded by a vertical space of one line or half a line. This is typically not the case within the Address element and is never the case within the Preformatted Text element. With some HTML interpreters, the first line in a paragraph is indented.

Example of use:

```
<H1>This Heading Precedes the Paragraph</H1>
<P>This is the text of the first paragraph.
<P>This is the text of the second paragraph. Although you do not
need to start paragraphs on new lines, maintaining this
convention facilitates document maintenance.</P>
<P>This is the text of a third paragraph.</P>
```

9.2. Preformatted Text

<PRE> ... </PRE> Level 0

The Preformatted Text element presents blocks of text in fixed-width font, and so is suitable for text that has been formatted on screen.

The <PRE> tag may be used with the optional WIDTH attribute. The WIDTH attribute specifies the maximum number of characters for a line and allows the HTML interpreter to select a suitable font and indentation. If the WIDTH attribute is not present, a width of 80 characters is assumed. Where the WIDTH attribute is supported, widths of 40, 80 and 132 characters should be presented optimally, with other widths being rounded up.

Within preformatted text:

* Line breaks within the text are rendered as a move to the beginning of the next line.
* Anchor elements and character highlighting elements may be used.
* Elements that define paragraph formatting (headings, address, etc.) must not be used.
* The horizontal tab character (encoded in US-ASCII and ISO-8859-1 as decimal 9) must be interpreted as the smallest positive nonzero number of spaces which will leave the number of characters so far on the line as a multiple of 8. Its use is not recommended however.

NOTE - Som historical documents contain <P> tags in <PRE> elements. User agents are engcouraged to treat this a a line break. A <P> tag followed by a newline character should produce only one line break, not a line break plus a blank line.

NOTE - References to the ``beginning of a new line'' do not imply that the renderer is forbidden from using a constant left indent for rendering preformatted text. The left indent may be constrained by the width required.

Example of use:

```
<PRE WIDTH="80">
This is an example line.
</PRE>
```

NOTE - Within a Preformatted Text element, the constraint that the rendering must be on a fixed horizontal character pitch may limit or prevent the ability of the HTML interpreter to faithfully render

character formatting elements.

9.3. Address

<ADDRESS> ... </ADDRESS> Level 0

The Address element specifies such information as address, signature and authorship, often at the top or bottom of a document.

Typically, an Address is rendered in an italic typeface and may be indented. The Address element implies a paragraph break before and after.

Example of use:

```
<ADDRESS>
Newsletter editor<BR>
J.R. Brown<BR>
JimquickPost News, Jumquick, CT 01234<BR>
Tel (123) 456 7890
</ADDRESS>
```

9.4. Blockquote

<BLOCKQUOTE> ... </BLOCKQUOTE> Level 0

The Blockquote element is used to contain text quoted from another source.

A typical rendering might be a slight extra left and right indent, and/or italic font. The Blockquote element causes a paragraph break, and typically provides space above and below the quote.

Single-font rendition may reflect the quotation style of Internet mail by putting a vertical line of graphic characters, such as the greater than symbol (>), in the left margin.

Example of use:

```
I think the poem ends
<BLOCKQUOTE>
<P>Soft you now, the fair Ophelia. Nymph, in thy orisons, be all
my sins remembered.
</BLOCKQUOTE>
but I am not sure.
```

9.5. Headings

<H1> ... </H1> through <H6> ... </H6> Level 0

HTML defines six levels of heading. A Heading element implies all the font changes, paragraph breaks before and after, and white space necessary to render the heading.

The highest level of headings is H1, followed by H2 ... H6.

Example of use:

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

The rendering of headings is determined by the HTML interpreter, but typical renderings are:

<H1> ... </H1>
: Bold, very-large font, centered. One or two blank lines above and below.

<H2> ... </H2>
: Bold, large font, flush-left. One or two blank lines above and below.

<H3> ... </H3>
: Italic, large font, slightly indented from the left margin. One or two blank lines above and below.

<H4> ... </H4>
: Bold, normal font, indented more than H3. One blank line above and below.

<H5> ... </H5>
: Italic, normal font, indented as H4. One blank line above.

<H6> ... </H6>
: Bold, indented same as normal text, more than H5. One blank line above.

Although heading levels can be skipped (for example, from H1 to H3), this practice is discouraged as skipping heading levels may produce unpredictable results when generating other representations from HTML.

9.6. List Elements

HTML supports several types of lists, all of which may be nested.

9.6.1. Definition List

<DL> ... </DL> Level 0

A definition list is a list of terms and corresponding definitions. Definition lists are typically formatted with the term flush-left and the definition, formatted paragraph style, indented after the term.

Example of use:

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

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.

Single occurrences of a <DT> tag without a subsequent <DD> tag are allowed, and have the same significance as if the <DD> tag had been present with no text.

The opening list tag must be <DL> and must be immediately followed by the first term (<DT>).

The definition list type can take the COMPACT attribute, which suggests that a compact rendering be used, because the list items are small and/or the entire list is large.

Unless you provide the COMPACT attribute, the HTML interpreter may leave white space between successive DT, DD pairs. The COMPACT attribute may also reduce the width of the left-hand (DT) column.

If using the COMPACT attribute, the opening list tag must be <DL COMPACT>, which must be immediately followed by the first <DT> tag:

```
<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>
```

9.6.2. Directory List

<DIR> ... </DIR> Level 0

A Directory List element is used to present a list of items containing up to 20 characters each. Items in a directory list may be arranged in columns, typically 24 characters wide. If the HTML interpreter can optimize the column width as function of the widths of individual elements, so much the better.

A directory list must begin with the <DIR> tag which is immediately followed by a <LI> (list item) tag:

```
<DIR>
<LI>A-H<LI>I-M
<LI>M-R<LI>S-Z
</DIR>
```

9.6.3. Menu List

<MENU> ... </MENU> Level 0

A menu list is a list of items with typically one line per item. The menu list style is more compact than the style of an unordered list.

A menu list must begin with a <MENU> tag which is immediately followed by a <LI> (list item) tag:

```
<MENU>
<LI>First item in the list.
<LI>Second item in the list.
<LI>Third item in the list.
</MENU>
```

9.6.4. Ordered List

<OL> ... </OL> Level 0

The Ordered List element is used to present a numbered list of items, sorted by sequence or order of importance.

An ordered list must begin with the <OL> tag which is immediately followed by a <LI> (list item) tag:

```
<OL>
<LI>Click the Web button to open the Open the URI window.
<LI>Enter the URI number in the text field of the Open URI
window. The Web document you specified is displayed.
```

```
<LI>Click highlighted text to move from one link to another.
</OL>
```

The Ordered List element can take the COMPACT attribute, which suggests that a compact rendering be used.

9.6.5. Unordered List

<UL> ... </UL> Level 0

The Unordered List element is used to present a list of items which is typically separated by white space and/or marked by bullets.

An unordered list must begin with the <UL> tag which is immediately followed by a <LI> (list item) tag:

```
<UL>
<LI>First list item
<LI>Second list item
<LI>Third list item
</UL>
```

10. Form-based Input Elements

Forms are created by placing input fields within paragraphs, preformatted/literal text, and lists. This gives considerable flexibility in designing the layout of forms.

The following elements are used to create forms:

FORM
: A form within a document.

INPUT
: One input field.

OPTION
: One option within a Select element.

SELECT
: A selection from a finite set of options.

TEXTAREA
: A multi-line input field.

Each variable field is defined by an Input, Textarea, or Option element and must have an NAME attribute to identify its value in the data returned when the form is submitted.

Example of use (a questionnaire form):

```
<H1>Sample Questionnaire</H1>
<P>Please fill out this questionnaire:
<FORM METHOD="POST" ACTION="http://www.w3.org/sample">
<P>Your name: <INPUT NAME="name" size="48">
<P>Male <INPUT NAME="gender" TYPE=RADIO VALUE="male">
<P>Female <INPUT NAME="gender" TYPE=RADIO VALUE="female">
<P>Number in family: <INPUT NAME="family" TYPE=text>
<P>Cities in which you maintain a residence:
<UL>
<LI>Kent <INPUT NAME="city" TYPE=checkbox VALUE="kent">
<LI>Miami <INPUT NAME="city" TYPE=checkbox VALUE="miami">
<LI>Other <TEXTAREA NAME="other" cols=48 rows=4></textarea>
</UL>
Nickname: <INPUT NAME="nickname" SIZE="42">
<P>Thank you for responding to this questionnaire.
<P><INPUT TYPE=SUBMIT> <INPUT TYPE=RESET>
</FORM>
```

In the example above, the <P> and <UL> tags have been used to lay out the text and input fields. The HTML interpreter is responsible for handling which field will currently get keyboard input.

Many platforms have existing conventions for forms, for example, using Tab and Shift keys to move the keyboard focus forwards and backwards between fields, and using the Enter key to submit the form. In the example, the SUBMIT and RESET buttons are specified explicitly with special purpose fields. The SUBMIT button is used to e-mail the form or send its contents to the server as specified by the ACTION attribute, while RESET resets the fields to their initial values. When the form consists of a single text field, it may be appropriate to leave such buttons out and rely on the Enter key.

The Input element is used for a large variety of types of input fields.

To let users enter more than one line of text, use the Textarea element.

The radio button and checkbox types of input field can be used to specify multiple choice forms in which every alternative is visible as part of the form. An alternative is to use the Select element which is typically rendered in a more compact fashion as a pull down combo list.

10.1. Form

<FORM> ... </FORM> Level 2

The Form element is used to delimit a data input form. There can be several forms in a single document, but the Form element can't be nested.

The ACTION attribute is a URI specifying the location to which the contents of the form is submitted to elicit a response. If the ACTION attribute is missing, the URI of the document itself is assumed. The way data is submitted varies with the access protocol of the URI, and with the values of the METHOD and ENCTYPE attributes.

In general:

* the METHOD attribute selects variations in the protocol.
* the ENCTYPE attribute specifies the format of the submitted data in case the protocol does not impose a format itself.

When the ACTION attribute is set to an HTTP URL, the METHOD attribute must be set to an HTTP method [HTTP]. The default method is GET, although for many applications the POST method is preferred. With the POST method, the ENCTYPE attribute is a media type specifying the format of the posted data; the default is ``application/x-www-form-urlencoded''.

The submitted contents of the form logically consist of name/value pairs. The names are usually equal to the NAME attributes of the various interactive elements in the form.

NOTE - The names are not guaranteed to be unique keys, nor are the names of form elements required to be distinct. The values encode the user's input to the corresponding interactive elements. Fields with null values may be omitted from the returned list of name/value pairs, whereas those with non-null values should be included (even if the value was not altered by the user). In particular, unselected radio buttons and checkboxes should be excluded from the contents list.

10.2. Input

<INPUT> Level 2

The Input element represents a field whose contents may be edited by the user.

Attributes of the Input element:

ALIGN
: Vertical alignment of the image. For use only with TYPE=IMAGE. The possible values are exactly the same as for the ALIGN attribute of the image element.

CHECKED
: Indicates that a checkbox or radio button is selected. Unselected checkboxes and radio buttons do not return name/value pairs when the form is submitted.

MAXLENGTH
: Indicates the maximum number of characters that can be entered into a text field. This can be greater than specified by the SIZE attribute, in which case the field will scroll appropriately. The default number of characters is unlimited.

NAME
: Symbolic name used when transferring the form's contents. The NAME attribute is required for most input types and is normally used to provide a unique identifier for a field, or for a logically related group of fields.

SIZE
: Specifies the size or precision of the field according to its type. For example, to specify a field with a visible width of 24 characters:

```
INPUT TYPE=text SIZE="24"
```

SRC
: A URI specifying an image. For use only with TYPE=IMAGE.

TYPE
: Defines the type of data the field accepts. Defaults to free text. Several types of fields can be defined with the type attribute:

CHECKBOX
: Used for simple Boolean attributes, or for attributes that can take multiple values at the same time. The latter is represented by a number of checkbox fields each of which has the same name. Each selected checkbox generates a separate name/value pair in the submitted data, even if

this results in duplicate names. The default value for checkboxes is ``on''.

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.

IMAGE

An image field upon which you can click with a pointing device, causing the form to be immediately submitted. The coordinates of the selected point are measured in pixel units from the upper-left corner of the image, and are returned (along with the other contents of the form) in two name/value pairs. The x-coordinate is submitted under the name of the field with ``.x'' appended, and the y-coordinate is submitted under the name of the field with ``.y'' appended. Any VALUE attribute is ignored. The image itself is specified by the SRC attribute, exactly as for the Image element.

NOTE - In a future version of the HTML specification, the IMAGE functionality may be folded into an enhanced SUBMIT field.

PASSWORD

The same as the TEXT attribute, except that text is not displayed as it is entered.

RADIO

Used for attributes that accept a single value from a set of alternatives. Each radio button field in the group should be given the same name. Only the selected radio button in the group generates a name/value pair in the submitted data. Radio buttons require an explicit VALUE attribute.

RESET

A button that when pressed resets the form's fields to their specified initial values. The label to be displayed on the button may be specified just as for the SUBMIT button.

SUBMIT

A button that when pressed submits the form. You can use the VALUE attribute to provide a non-editable label to be displayed on the button. The default label is application-specific. If a

SUBMIT button is pressed in order to submit the form, and that button has a NAME attribute specified, then that button contributes a name/value pair to the submitted data. Otherwise, a SUBMIT button makes no contribution to the submitted data.

TEXT

Used for a single line text entry fields. Use in conjunction with the SIZE and MAXLENGTH attributes. Use the Textarea element for text fields which can accept multiple lines.

VALUE

The initial displayed value of the field, if it displays a textual or numerical value; or the value to be returned when the field is selected, if it displays a Boolean value. This attribute is required for radio buttons.

10.3. Option

<OPTION> Level 2

The Option element can only occur within a Select element. It represents one choice, and can take these attributes:

SELECTED

Indicates that this option is initially selected.

VALUE

When present indicates the value to be returned if this option is chosen. The returned value defaults to the contents of the Option element.

The contents of the Option element is presented to the user to represent the option. It is used as a returned value if the VALUE attribute is not present.

10.4. Select

<SELECT NAME=... > ... </SELECT> Level 2

The Select element allows the user to chose one of a set of alternatives described by textual labels. Every alternative is represented by the Option element. Attributes are:

MULTIPLE

The MULTIPLE attribute is needed when users are allowed to make several selections, e.g. <SELECT

MULTIPLE>.

NAME
Specifies the name that will submitted as a name/value pair.

SIZE
Specifies the number of visible items. If this is greater than one, then the resulting form control will be a list.

The Select element is typically rendered as a pull down or pop-up list. For example:

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

If no option is initially marked as selected, then the first item listed is selected.

10.5. Text Area

<TEXTAREA> ... </TEXTAREA> Level 2

The Textarea element lets users enter more than one line of text. 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 (</TEXTAREA>) is used to initialize the field's value. This end tag is always required even if the field is initially blank. When submitting a form, lines in a TEXTAREA should be terminated using CRLF.

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. HTML interpreters should allow text to extend beyond these limits by scrolling as needed.

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. SGML's default (Reference
Quantity Set) limits the length of attribute literals
to only 240 characters. The HTML 2.0 SGML declaration
increases the limit to 1024 characters.

11. HTML Public Text

11.1. HTML DTD

```
This is the Document Type Definition for the HyperText
Markup Language.

<!--    html.dtd

        Document Type Definition for the HyperText Markup Language
                (HTML DTD)

        $Id: html.dtd,v 1.25 1995/03/29 18:53:13 connolly Exp $

        Author: Daniel W. Connolly <connolly@w3.org>
        See Also: html.decl, html-0.dtd, html-1.dtd
          http://info.cern.ch/hypertext/WWW/MarkUp/MarkUp.html
-->

<!ENTITY % HTML.Version
        "-//IETF//DTD HTML 2.0//EN"

        -- Typical usage:

            <!DOCTYPE HTML PUBLIC "-//IETF//DTD HTML//EN">
            <html>
            ...
            </html>
        --
        >


<!--============ Feature Test Entities ========================-->

<!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
           those 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.Highlighting "INCLUDE"
        -- Use this feature test entity to validate that a
           document uses no highlighting tags, which may be
           ignored on minimal implementations.
        -->

<!ENTITY % HTML.Forms "INCLUDE"
        -- Use this feature test entity to validate that a document
           contains no forms, which may not be supported in minimal
           implementations
        -->

<!--============== Imported Names ==============================-->

<!ENTITY % Content-Type "CDATA"
        -- meaning an internet media type
           (aka MIME content type, as per RFC1521)
        -->

<!ENTITY % HTTP-Method "GET | POST"
        -- as per HTTP specification, in progress
        -->

<!ENTITY % URI "CDATA"
        -- The term URI means a CDATA attribute
           whose value is a Uniform Resource Identifier,
           as defined by
        "Universal Resource Identifiers" by Tim Berners-Lee
        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.

        -->

<!--========= DTD "Macros" ====================-->
```

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

<!ENTITY % list " UL | OL | DIR | MENU " >


<!--======= Character mnemonic entities =================-->

<!ENTITY % ISOlat1 PUBLIC
  "ISO 8879-1986//ENTITIES Added Latin 1//EN//HTML">
%ISOlat1;

<!ENTITY amp CDATA "&#38;"     -- ampersand          -->
<!ENTITY gt CDATA "&#62;"      -- greater than       -->
<!ENTITY lt CDATA "&#60;"      -- less than          -->
<!ENTITY quot CDATA "&#34;"    -- double quote       -->


<!--========== SGML Document Access (SDA) Parameter Entities =====-->

<!-- HTML 2.0 contains SGML Document Access (SDA) fixed attributes
in support of easy transformation to the International Committee
for Accessible Document Design (ICADD) DTD
        "-//EC-USA-CDA/ICADD//DTD ICADD22//EN".
ICADD applications are designed to support usable access to
structured information by print-impaired individuals through
Braille, large print and voice synthesis.  For more information on
SDA & ICADD:
        - ISO 12083:1993, Annex A.8, Facilities for Braille,
          large print and computer voice
        - ICADD ListServ
          <ICADD%ASUACAD.BITNET@ARIZVM1.ccit.arizona.edu>
        - Usenet news group bit.listserv.easi
        - Recording for the Blind, +1 800 221 4792
-->

<!ENTITY % SDAFORM  "SDAFORM  CDATA  #FIXED"
          -- one to one mapping        -->
<!ENTITY % SDARULE  "SDARULE  CDATA  #FIXED"
          -- context-sensitive mapping -->
<!ENTITY % SDAPREF  "SDAPREF  CDATA  #FIXED"
          -- generated text prefix     -->
<!ENTITY % SDASUFF  "SDASUFF  CDATA  #FIXED"
          -- generated text suffix     -->
<!ENTITY % SDASUSP  "SDASUSP  NAME   #FIXED"
          -- suspend transform process -->


<!--========== Text Markup ====================-->

<![ %HTML.Highlighting [

<!ENTITY % font " TT | B | I ">
```

```
<!ENTITY % phrase "EM | STRONG | CODE | SAMP | KBD | VAR | CITE ">

<!ENTITY % text "#PCDATA | A | IMG | BR | %phrase | %font">

<!ELEMENT (%font;|%phrase) - - (%text)*>
<!ATTLIST ( TT | CODE | SAMP | KBD | VAR )
        %SDAFORM; "Lit"
        >
<!ATTLIST ( B | STRONG )
        %SDAFORM; "B"
        >
<!ATTLIST ( I | EM | CITE )
        %SDAFORM; "It"
        >

<!-- <TT>       Typewriter text                          -->
<!-- <B>        Bold text                                -->
<!-- <I>        Italic text                              -->

<!-- <EM>       Emphasized phrase                        -->
<!-- <STRONG>   Strong emphais                           -->
<!-- <CODE>     Source code phrase                       -->
<!-- <SAMP>     Sample text or characters                -->
<!-- <KBD>      Keyboard phrase, e.g. user input         -->
<!-- <VAR>      Variable phrase or substituable          -->
<!-- <CITE>     Name or title of cited work              -->

<!ENTITY % pre.content "#PCDATA | A | HR | BR | %font | %phrase">

]]>

<!ENTITY % text "#PCDATA | A | IMG | BR">

<!ELEMENT BR    - O EMPTY>
<!ATTLIST BR
        %SDAPREF; "&#RE;"
        >

<!-- <BR>       Line break      -->


<!--========= Link Markup ======================-->

<![ %HTML.Recommended [
        <!ENTITY % linkName "ID">
]]>

<!ENTITY % linkName "CDATA">

<!ENTITY % linkType "NAME"
        -- a list of these will be specified at a later date -->
```

```
    <!ENTITY % linkExtraAttributes
195         "REL %linkType #IMPLIED
            REV %linkType #IMPLIED
            URN CDATA #IMPLIED
            TITLE CDATA #IMPLIED
            METHODS NAMES #IMPLIED
200         ">

    <![ %HTML.Recommended [
            <!ENTITY % A.content   "(%text)*"
            -- <H1><a name="xxx">Heading</a></H1>
205                 is preferred to
               <a name="xxx"><H1>Heading</H1></a>
            -->
    ]]>

210 <!ENTITY % A.content   "(%heading|%text)*">

    <!ELEMENT A     - - %A.content -(A)>
    <!ATTLIST A
            HREF %URI #IMPLIED
215         NAME %linkName #IMPLIED
            %linkExtraAttributes;
            %SDAPREF; "<Anchor: #AttList>"
            >
    <!-- <A>                Anchor; source/destination of link      -->
220 <!-- <A NAME="..."> Name of this anchor                        -->
    <!-- <A HREF="..."> Address of link destination               -->
    <!-- <A URN="..."> Permanent address of destination           -->
    <!-- <A REL=...>   Relationship to destination                -->
    <!-- <A REV=...>   Relationship of destination to this        -->
225 <!-- <A TITLE="..."> Title of destination (advisory)          -->
    <!-- <A METHODS="..."> Operations on destination (advisory)   -->


    <!--========== Images ==========================-->
230
    <!ELEMENT IMG    - O EMPTY>
    <!ATTLIST IMG
            SRC %URI;  #REQUIRED
            ALT CDATA #IMPLIED
235         ALIGN (top|middle|bottom) #IMPLIED
            ISMAP (ISMAP) #IMPLIED
            %SDAPREF; "<Fig><?SDATrans Img: #AttList>#AttVal(Alt)</Fig>"
            >

240 <!-- <IMG>              Image; icon, glyph or illustration      -->
    <!-- <IMG SRC="..."> Address of image object                  -->
    <!-- <IMG ALT="..."> Textual alternative                      -->
    <!-- <IMG ALIGN=...> Position relative to text                -->
```

```
    <!-- <IMG ISMAP>        Each pixel can be a link                -->
245
    <!--========== Paragraphs======================-->

    <!ELEMENT P     - O (%text)*>
    <!ATTLIST P
250         %SDAFORM; "Para"
            >

    <!-- <P>        Paragraph       -->

255
    <!--========== Headings, Titles, Sections ===============-->

    <!ELEMENT HR    - O EMPTY>
    <!ATTLIST HR
260         %SDAPREF; "&#RE;&#RE;"
            >

    <!-- <HR>       Horizontal rule -->

265 <!ELEMENT ( %heading )  - -  (%text;)*>
    <!ATTLIST H1
            %SDAFORM; "H1"
            >
    <!ATTLIST H2
270         %SDAFORM; "H2"
            >
    <!ATTLIST H3
            %SDAFORM; "H3"
            >
275 <!ATTLIST H4
            %SDAFORM; "H4"
            >
    <!ATTLIST H5
            %SDAFORM; "H5"
280         >
    <!ATTLIST H6
            %SDAFORM; "H6"
            >

285 <!-- <H1>       Heading, level 1 -->
    <!-- <H2>       Heading, level 2 -->
    <!-- <H3>       Heading, level 3 -->
    <!-- <H4>       Heading, level 4 -->
    <!-- <H5>       Heading, level 5 -->
290 <!-- <H6>       Heading, level 6 -->


    <!--========== Text Flows ======================-->

295 <![ %HTML.Forms [
```

```
        <!ENTITY % block.forms "BLOCKQUOTE | FORM | ISINDEX">
]]>

<!ENTITY % block.forms "BLOCKQUOTE">

<![ %HTML.Deprecated [
        <!ENTITY % preformatted "PRE | XMP | LISTING">
]]>

<!ENTITY % preformatted "PRE">

<!ENTITY % block "P | %list | DL
        | %preformatted
        | %block.forms">

<!ENTITY % flow "(%text|%block)*">

<!ENTITY % pre.content "#PCDATA | A | HR | BR">
<!ELEMENT PRE - - (%pre.content)*>
<!ATTLIST PRE
        WIDTH NUMBER #implied
        %SDAFORM; "Lit"
        >

<!-- <PRE>             Preformatted text              -->
<!-- <PRE WIDTH=...>   Maximum characters per line    -->

<![ %HTML.Deprecated [

<!ENTITY % literal "CDATA"
        -- historical, non-conforming parsing mode where
           the only markup signal is the end tag
           in full
        -->

<!ELEMENT (XMP|LISTING) - -  %literal>
<!ATTLIST XMP
        %SDAFORM; "Lit"
        %SDAPREF; "Example:&#RE;"
        >
<!ATTLIST LISTING
        %SDAFORM; "Lit"
        %SDAPREF; "Listing:&#RE;"
        >

<!-- <XMP>             Example section         -->
<!-- <LISTING>         Computer listing        -->

<!ELEMENT PLAINTEXT - O %literal>
<!-- <PLAINTEXT>       Plain text passage      -->

<!ATTLIST PLAINTEXT
```

```
        %SDAFORM; "Lit"
        >
]]>


<!--========== Lists ==================-->

<!ELEMENT DL    - -  (DT | DD)+>
<!ATTLIST DL
        COMPACT (COMPACT) #IMPLIED
        %SDAFORM; "List"
        %SDAPREF; "Definition List:"
        >

<!ELEMENT DT    - O (%text)*>
<!ATTLIST DT
        %SDAFORM; "Term"
        >

<!ELEMENT DD    - O %flow>
<!ATTLIST DD
        %SDAFORM; "LItem"
        >

<!-- <DL>              Definition list, or glossary    -->
<!-- <DL COMPACT>      Compact style list              -->
<!-- <DT>              Term in definition list         -->
<!-- <DD>              Definition of term              -->

<!ELEMENT (OL|UL) - -  (LI)+>
<!ATTLIST OL
        COMPACT (COMPACT) #IMPLIED
        %SDAFORM; "List"
        >
<!ATTLIST UL
        COMPACT (COMPACT) #IMPLIED
        %SDAFORM; "List"
        >
<!-- <UL>              Unordered list                  -->
<!-- <UL COMPACT>      Compact list style              -->
<!-- <OL>              Ordered, or numbered list       -->
<!-- <OL COMPACT>      Compact list style              -->

<!ELEMENT (DIR|MENU) - -  (LI)+ -(%block)>
<!ATTLIST DIR
        COMPACT (COMPACT) #IMPLIED
        %SDAFORM; "List"
        %SDAPREF; "<LHead>Directory</LHead>"
        >
<!ATTLIST MENU
```

```
          COMPACT (COMPACT) #IMPLIED
          %SDAFORM; "List"
          %SDAPREF; "<LHead>Menu</LHead>"
          >

<!-- <DIR>             Directory list                -->
<!-- <DIR COMPACT>     Compact list style            -->
<!-- <MENU>            Menu list                     -->
<!-- <MENU COMPACT>    Compact list style            -->

<!ELEMENT LI    - O %flow>
<!ATTLIST LI
        %SDAFORM; "LItem"
        >

<!-- <LI>              List item                     -->

<!--========== Document Body ==================-->

<![ %HTML.Recommended [
        <!ENTITY % body.content "(%heading|%block|HR|ADDRESS|IMG)*"
        -- <h1>Heading</h1>
           <p>Text ...
                is preferred to
           <h1>Heading</h1>
           Text ...
        -->
]]>

<!ENTITY % body.content "(%heading | %text | %block |
                          HR | ADDRESS)*">

<!ELEMENT BODY O O  %body.content>

<!-- <BODY>     Document body   -->

<!ELEMENT BLOCKQUOTE - - %body.content>
<!ATTLIST BLOCKQUOTE
        %SDAFORM; "BQ"
        >

<!-- <BLOCKQUOTE>       Quoted passage  -->

<!ELEMENT ADDRESS - - (%text|P)*>
<!ATTLIST  ADDRESS
        %SDAFORM; "Lit"
        %SDAPREF; "Address:&#RE;"
        >

<!-- <ADDRESS>  Address, signature, or byline   -->
```

```
    <!--======= Forms ====================-->

    <![ %HTML.Forms [

455 <!ELEMENT FORM - - %body.content -(FORM) +(INPUT|SELECT|TEXTAREA)>
    <!ATTLIST FORM
            ACTION %URI #IMPLIED
            METHOD (%HTTP-Method) GET
            ENCTYPE %Content-Type; "application/x-www-form-urlencoded"
460         %SDAPREF; "<Para>Form:</Para>"
            %SDASUFF; "<Para>Form End.</Para>"
            >

    <!-- <FORM>                    Fill-out or data-entry form     -->
465 <!-- <FORM ACTION="...">        Address for completed form      -->
    <!-- <FORM METHOD=...>          Method of submitting form       -->
    <!-- <FORM ENCTYPE="...">       Representation of form data     -->

    <!ENTITY % InputType "(TEXT | PASSWORD | CHECKBOX |
470                       RADIO | SUBMIT | RESET |
                          IMAGE | HIDDEN )">
    <!ELEMENT INPUT - O EMPTY>
    <!ATTLIST INPUT
            TYPE %InputType TEXT
475         NAME CDATA #IMPLIED
            VALUE CDATA #IMPLIED
            SRC %URI #IMPLIED
            CHECKED (CHECKED) #IMPLIED
            SIZE CDATA #IMPLIED
480         MAXLENGTH NUMBER #IMPLIED
            ALIGN (top|middle|bottom) #IMPLIED
            %SDAPREF; "Input: "
            >

485 <!-- <INPUT>                    Form input datum                -->
    <!-- <INPUT TYPE=...>           Type of input interaction       -->
    <!-- <INPUT NAME=...>           Name of form datum              -->
    <!-- <INPUT VALUE="...">        Default/initial/selected value  -->
    <!-- <INPUT SRC="...">          Address of image                -->
490 <!-- <INPUT CHECKED>            Initial state is "on"           -->
    <!-- <INPUT SIZE=...>           Field size hint                 -->
    <!-- <INPUT MAXLENGTH=...>      Data length maximum             -->
    <!-- <INPUT ALIGN=...>          Image alignment                 -->

495 <!ELEMENT SELECT - - (OPTION+) -(INPUT|SELECT|TEXTAREA)>
    <!ATTLIST SELECT
            NAME CDATA #REQUIRED
            SIZE NUMBER #IMPLIED
            MULTIPLE (MULTIPLE) #IMPLIED
500         %SDAFORM; "List"
            %SDAPREF;
```

```
        "<LHead>Select #AttVal(Multiple)</LHead>"
        >

<!-- <SELECT>                  Selection of option(s)          -->
<!-- <SELECT NAME=...>         Name of form datum              -->
<!-- <SELECT SIZE=...>         Options displayed at a time     -->
<!-- <SELECT MULTIPLE>         Multiple selections allowed     -->

<!ELEMENT OPTION - O (#PCDATA)*>
<!ATTLIST OPTION
        SELECTED (SELECTED) #IMPLIED
        VALUE CDATA #IMPLIED
        %SDAFORM; "LItem"
        %SDAPREF;
        "Option: #AttVal(Value) #AttVal(Selected)"
        >

<!-- <OPTION>                  A selection option              -->
<!-- <OPTION SELECTED>         Initial state                   -->
<!-- <OPTION VALUE="...">      Form datum value for this option-->

<!ELEMENT TEXTAREA - - (#PCDATA)* -(INPUT|SELECT|TEXTAREA)>
<!ATTLIST TEXTAREA
        NAME CDATA #REQUIRED
        ROWS NUMBER #REQUIRED
        COLS NUMBER #REQUIRED
        %SDAFORM; "Para"
        %SDAPREF; "Input Text -- #AttVal(Name): "
        >

<!-- <TEXTAREA>                An area for text input          -->
<!-- <TEXTAREA NAME=...>       Name of form datum              -->
<!-- <TEXTAREA ROWS=...>       Height of area                  -->
<!-- <TEXTAREA COLS=...>       Width of area                   -->

]]>


<!--======= Document Head ======================-->

<![ %HTML.Recommended [
        <!ENTITY % head.extra "META* & LINK*">
]]>

<!ENTITY % head.extra "NEXTID? & META* & LINK*">

<!ENTITY % head.content "TITLE & ISINDEX? & BASE? &
                         (%head.extra)">

<!ELEMENT HEAD O O  (%head.content)>

<!-- <HEAD>     Document head   -->
```

```
<!ELEMENT TITLE - -  (#PCDATA)*>
<!ATTLIST TITLE
        %SDAFORM; "Ti"    >

<!-- <TITLE>    Title of document -->

<!ELEMENT LINK - O EMPTY>
<!ATTLIST LINK
        HREF %URI #REQUIRED
        %linkExtraAttributes;
        %SDAPREF; "Linked to : #AttVal (TITLE) (URN) (HREF)>"    >

<!-- <LINK>            Link from this document                -->
<!-- <LINK HREF="..."> Address of link destination            -->
<!-- <LINK URN="..."> Lasting name of destination             -->
<!-- <LINK REL=...>    Relationship to destination            -->
<!-- <LINK REV=...>    Relationship of destination to this    -->
<!-- <LINK TITLE="..."> Title of destination (advisory)       -->
<!-- <LINK METHODS="..."> Operations allowed (advisory)       -->

<!ELEMENT ISINDEX - O EMPTY>
<!ATTLIST ISINDEX
        %SDAPREF;
   "<Para>[Document is indexed/searchable.]</Para>">

<!-- <ISINDEX>         Document is a searchable index         -->

<!ELEMENT BASE - O EMPTY>
<!ATTLIST BASE
        HREF %URI; #REQUIRED     >

<!-- <BASE>            Base context document                  -->
<!-- <BASE HREF="..."> Address for this document              -->

<!ELEMENT NEXTID - O EMPTY>
<!ATTLIST NEXTID
        N %linkName #REQUIRED     >

<!-- <NEXTID>          Next ID to use for link name           -->
<!-- <NEXTID N=...>    Next ID to use for link name           -->

<!ELEMENT META - O EMPTY>
<!ATTLIST META
        HTTP-EQUIV  NAME    #IMPLIED
        NAME        NAME    #IMPLIED
        CONTENT     CDATA   #REQUIRED    >

<!-- <META>                    Generic Metainformation        -->
<!-- <META HTTP-EQUIV=...>     HTTP response header name      -->
<!-- <META NAME=...>           Metainformation name           -->
```

```
<!-- <META CONTENT="..."> Associated information -->

<!--======= Document Structure ================-->

<![ %HTML.Deprecated [
        <!ENTITY % html.content "HEAD, BODY, PLAINTEXT?">
]]>
<!ENTITY % html.content "HEAD, BODY">

<!ELEMENT HTML O O  (%html.content)>
<!ENTITY % version.attr "VERSION CDATA #FIXED '%HTML.Version;'">

<!ATTLIST HTML
        %version.attr;
        %SDAFORM; "Book"
        >

<!-- <HTML>                    HTML Document   -->
```

11.2. SGML Declaration for HTML

This is the SGML Declaration for HyperText Markup Language (HTML) as used by the World Wide Web (WWW) application:

```
<!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        150000
                GRPCAP          150000
```

```
                  ENTCAP          150000

30  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
35                    International Reference Version
                      (IRV)//ESC 2/5 4/0"
             DESCSET  0 128 0
             FUNCTION
                      RE          13
40                    RS          10
                      SPACE       32
                      TAB SEPCHAR  9


45           NAMING   LCNMSTRT ""
                      UCNMSTRT ""
                      LCNMCHAR ".-"
                      UCNMCHAR ".-"
                      NAMECASE GENERAL YES
50                             ENTITY  NO
             DELIM    GENERAL  SGMLREF
                      SHORTREF SGMLREF
             NAMES    SGMLREF
             QUANTITY SGMLREF
55                    ATTSPLEN 2100
                      LITLEN   1024
                      NAMELEN  72    -- somewhat arbitrary; taken from
                                    internet line length conventions --
                      PILEN    1024
60                    TAGLEN   2100
                      GRPGTCNT 150
                      GRPCNT   64

    FEATURES
65    MINIMIZE
        DATATAG  NO
        OMITTAG  YES
        RANK     NO
        SHORTTAG YES
70    LINK
        SIMPLE   NO
        IMPLICIT NO
        EXPLICIT NO
      OTHER
75      CONCUR   NO
        SUBDOC   NO
        FORMAL   YES
      APPINFO    "SDA"  -- conforming SGML Document Access application
```

```
                        --
>
<!--
        $Id: html.decl,v 1.15 1995/05/06 01:44:47 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
 -->
```

11.3. Sample SGML Open Entity Catalog for HTML

The SGML standard describes an ``entity manager'' as the portion or component of an SGML system that maps SGML entities into the actual storage model (e.g., the file system). The standard itself does not define a particular mapping methodology or notation.

To assist the interoperability among various SGML tools and systems, the SGML Open consortium has passed a technical resolution that defines a format for an application-independent entity catalog that maps external identifiers and/or entity names to file names.

Each entry in the catalog associates a storage object identifier (such as a file name) with information about the external entity that appears in the SGML document. In addition to entries that associate public identifiers, a catalog entry can associate an entity name with a storage object indentifier. For example, the following are possible catalog entries:

```
        -- catalog: SGML Open style entity catalog for HTML --
        -- $Id: catalog,v 1.2 1994/11/30 23:45:18 connolly Exp $ --

        -- Ways to refer to Level 2: most general to most specific --
PUBLIC  "-//IETF//DTD HTML//EN"                 html.dtd
PUBLIC  "-//IETF//DTD HTML 2.0//EN"             html.dtd
PUBLIC  "-//IETF//DTD HTML Level 2//EN"         html.dtd
PUBLIC  "-//IETF//DTD HTML 2.0 Level 2//EN"     html.dtd

        -- Ways to refer to Level 1: most general to most specific --
PUBLIC  "-//IETF//DTD HTML Level 1//EN"         html-1.dtd
PUBLIC  "-//IETF//DTD HTML 2.0 Level 1//EN"     html-1.dtd

        -- Ways to refer to Level 0: most general to most specific --
PUBLIC  "-//IETF//DTD HTML Level 0//EN"         html-0.dtd
PUBLIC  "-//IETF//DTD HTML 2.0 Level 0//EN"     html-0.dtd
```

```
             -- Ways to refer to Strict Level 2: most general to most specific
\
  &  --
 20  PUBLIC  "-//IETF//DTD HTML Strict//EN"                  html-s.dtd
     PUBLIC  "-//IETF//DTD HTML 2.0 Strict//EN"              html-s.dtd
     PUBLIC  "-//IETF//DTD HTML Strict Level 2//EN"          html-s.dtd
     PUBLIC  "-//IETF//DTD HTML 2.0 Strict Level 2//EN"      html-s.dtd

 25          -- Ways to refer to Strict Level 1: most general to most specific
\
  &  --
     PUBLIC  "-//IETF//DTD HTML Strict Level 1//EN"          html-1s.dtd
     PUBLIC  "-//IETF//DTD HTML 2.0 Strict Level 1//EN"      html-1s.dtd

             -- Ways to refer to Strict Level 0: most general to most specific
\
  &  --
 30  PUBLIC  "-//IETF//DTD HTML Strict Level 0//EN"          html-0s.dtd
     PUBLIC  "-//IETF//DTD HTML 2.0 Strict Level 0//EN"      html-0s.dtd

             -- ISO latin 1 entity set for HTML --
     PUBLIC  "ISO 8879-1986//ENTITIES Added Latin 1//EN//HTML"
ISOlat1.sg\
  &  ml
```

11.4. Character Entity Sets

The HTML DTD defines the following entities. 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.

11.4.1. Numeric and Special Graphic Entity Set

The following table lists each of the characters included from the Numeric and Special Graphic entity set, along with its name, syntax for use, and description. This list is derived from `ISO Standard 8879:1986//ENTITIES Numeric and Special Graphic//EN'. However, HTML does not include for the entire entity set -- only the entities listed below are included.

GLYPH	NAME	SYNTAX	DESCRIPTION
<	lt	<	Less than sign
>	gt	>	Greater than sign
&	amp	&	Ampersand
"	quot	"	Double quote sign

11.4.2. ISO Latin 1 Character Entity Set

```
    The following public text 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 Standard 8879:1986//ENTITIES Added Latin 1//EN.
    HTML includes the entire entity set.

    <!-- (C) International Organization for Standardization 1986
         Permission to copy in any form is granted for use with
         conforming SGML systems and applications as defined in
         ISO 8879, provided this notice is included in all copies.
 5  -->
    <!-- Character entity set. Typical invocation:
         <!ENTITY % ISOlat1 PUBLIC
           "ISO 8879-1986//ENTITIES Added Latin 1//EN//HTML">
         %ISOlat1;
10  -->
    <!--    Modified for use in HTML
            $Id: ISOlat1.sgml,v 1.2 1994/11/30 23:45:12 connolly Exp $ -->
    <!ENTITY AElig  CDATA "&#198;" -- capital AE diphthong (ligature) -->
    <!ENTITY Aacute CDATA "&#193;" -- capital A, acute accent -->
15  <!ENTITY Acirc  CDATA "&#194;" -- capital A, circumflex accent -->
    <!ENTITY Agrave CDATA "&#192;" -- capital A, grave accent -->
    <!ENTITY Aring  CDATA "&#197;" -- capital A, ring -->
    <!ENTITY Atilde CDATA "&#195;" -- capital A, tilde -->
    <!ENTITY Auml   CDATA "&#196;" -- capital A, dieresis or umlaut mark -->
20  <!ENTITY Ccedil CDATA "&#199;" -- capital C, cedilla -->
    <!ENTITY ETH    CDATA "&#208;" -- capital Eth, Icelandic -->
    <!ENTITY Eacute CDATA "&#201;" -- capital E, acute accent -->
    <!ENTITY Ecirc  CDATA "&#202;" -- capital E, circumflex accent -->
    <!ENTITY Egrave CDATA "&#200;" -- capital E, grave accent -->
25  <!ENTITY Euml   CDATA "&#203;" -- capital E, dieresis or umlaut mark -->
    <!ENTITY Iacute CDATA "&#205;" -- capital I, acute accent -->
    <!ENTITY Icirc  CDATA "&#206;" -- capital I, circumflex accent -->
    <!ENTITY Igrave CDATA "&#204;" -- capital I, grave accent -->
    <!ENTITY Iuml   CDATA "&#207;" -- capital I, dieresis or umlaut mark -->
30  <!ENTITY Ntilde CDATA "&#209;" -- capital N, tilde -->
    <!ENTITY Oacute CDATA "&#211;" -- capital O, acute accent -->
    <!ENTITY Ocirc  CDATA "&#212;" -- capital O, circumflex accent -->
    <!ENTITY Ograve CDATA "&#210;" -- capital O, grave accent -->
    <!ENTITY Oslash CDATA "&#216;" -- capital O, slash -->
35  <!ENTITY Otilde CDATA "&#213;" -- capital O, tilde -->
    <!ENTITY Ouml   CDATA "&#214;" -- capital O, dieresis or umlaut mark -->
    <!ENTITY THORN  CDATA "&#222;" -- capital THORN, Icelandic -->
    <!ENTITY Uacute CDATA "&#218;" -- capital U, acute accent -->
    <!ENTITY Ucirc  CDATA "&#219;" -- capital U, circumflex accent -->
40  <!ENTITY Ugrave CDATA "&#217;" -- capital U, grave accent -->
    <!ENTITY Uuml   CDATA "&#220;" -- capital U, dieresis or umlaut mark -->
    <!ENTITY Yacute CDATA "&#221;" -- capital Y, acute accent -->
    <!ENTITY aacute CDATA "&#225;" -- small a, acute accent -->
    <!ENTITY acirc  CDATA "&#226;" -- small a, circumflex accent -->
```

```
<!ENTITY aelig  CDATA "&#230;" -- small ae diphthong (ligature) -->
<!ENTITY agrave CDATA "&#224;" -- small a, grave accent -->
<!ENTITY aring  CDATA "&#229;" -- small a, ring -->
<!ENTITY atilde CDATA "&#227;" -- small a, tilde -->
<!ENTITY auml   CDATA "&#228;" -- small a, dieresis or umlaut mark -->
<!ENTITY ccedil CDATA "&#231;" -- small c, cedilla -->
<!ENTITY eacute CDATA "&#233;" -- small e, acute accent -->
<!ENTITY ecirc  CDATA "&#234;" -- small e, circumflex accent -->
<!ENTITY egrave CDATA "&#232;" -- small e, grave accent -->
<!ENTITY eth    CDATA "&#240;" -- small eth, Icelandic -->
<!ENTITY euml   CDATA "&#235;" -- small e, dieresis or umlaut mark -->
<!ENTITY iacute CDATA "&#237;" -- small i, acute accent -->
<!ENTITY icirc  CDATA "&#238;" -- small i, circumflex accent -->
<!ENTITY igrave CDATA "&#236;" -- small i, grave accent -->
<!ENTITY iuml   CDATA "&#239;" -- small i, dieresis or umlaut mark -->
<!ENTITY ntilde CDATA "&#241;" -- small n, tilde -->
<!ENTITY oacute CDATA "&#243;" -- small o, acute accent -->
<!ENTITY ocirc  CDATA "&#244;" -- small o, circumflex accent -->
<!ENTITY ograve CDATA "&#242;" -- small o, grave accent -->
<!ENTITY oslash CDATA "&#248;" -- small o, slash -->
<!ENTITY otilde CDATA "&#245;" -- small o, tilde -->
<!ENTITY ouml   CDATA "&#246;" -- small o, dieresis or umlaut mark -->
<!ENTITY szlig  CDATA "&#223;" -- small sharp s, German (sz ligature) -->
<!ENTITY thorn  CDATA "&#254;" -- small thorn, Icelandic -->
<!ENTITY uacute CDATA "&#250;" -- small u, acute accent -->
<!ENTITY ucirc  CDATA "&#251;" -- small u, circumflex accent -->
<!ENTITY ugrave CDATA "&#249;" -- small u, grave accent -->
<!ENTITY uuml   CDATA "&#252;" -- small u, dieresis or umlaut mark -->
<!ENTITY yacute CDATA "&#253;" -- small y, acute accent -->
<!ENTITY yuml   CDATA "&#255;" -- small y, dieresis or umlaut mark -->
```

12. Glossary

character
: An atom of information, for example a letter or a digit. Graphic characters have associated glyphs, where as control characters have associated processing semantics.

character encoding scheme
: A function whose domain is the set of sequences of octets, and whose range is the set of sequences of characters from a character repertoire; that is, a sequence of octets and a character encoding scheme determines a sequence of characters.

character repertoire
: A finite set of characters; e.g. the range of a

coded character set.

code position
: An integer. A coded character set and a code position from its domain determine a character.

coded character set
: A function whose domain is a subset of the integers and whose range is a character repertoire. That is, for some set of integers (usually of the form {0, 1, 2, ..., N}), a coded character set and an integer in that set determine a character. Conversely, a character and a coded character set determine the character's code position (or, in rare cases, a few code positions).

conforming HTML user agent
: A user agent that conforms to this specification in its processing of the Internet Media Type `text/html; version=2.0'.

data character
: Characters other than markup, which make up the content of elements.

document character set
: a coded character set whose range includes all characters used in a document. Every SGML document has exactly one document character set. Numeric character references are resolved via the document character set.

DTD
: document type definition. Rules that apply SGML to the markup of documents of a particular type, including a set of element and entity declarations. [SGML]

element
: A component of the hierarchical structure defined by a document type definition; it is identified in a document instance by descriptive markup, sually a start-tag and end-tag. [SGML]

end-tag
: Descriptive markup that identifies the end of an element. [SGML]

entity
: data with an associated notation or interpretation; for example, a sequence of octets associated with an Internet Media Type.[SGML]

HTML document
: An SGML document conforming to this document type definition.

markup
: Syntactically delimited characters added to the data of a document to represent its structure. There are four different kinds of markup: descriptive markup (tags), references, markup declarations, and processing instructions.[SGML]

may
: A document or user interface is conforming whether this statement applies or not.

message entity
: a head and body. The head is a collection of name/value fields, and the body is a sequence of octets. The head defines the content type and content transfer encoding of the body. [MIME]

must
: Documents or user agents in conflict with this statement are not conforming.

SGML document
: A sequence of characters organized physically as a set of entities and logically into a hierarchy of elements. An SGML document consists of data characters and markup; the markup describes the structure of the information and an instance of that structure.[SGML]

shall
: If a document or user agent conflicts with this statement, it does not conform to this specification.

should
: If a document or user agent conflicts with this statement, undesirable results may occur in practice even though it conforms to this specification.

start-tag
: Descriptive markup that identifies the start of an

element and specifies its generic identifier and attributes. [SGML]

syntax-reference
character set
: A coded character set whose range includes all characters used for markup; e.g. name characters and delimiter characters.

tag
: Markup that delimits an element. A tag includes a name which refers to an element declaration in the DTD, and may include attributes.[SGML]

text entity
: A finite sequence of characters. A text entity typically takes the form of a sequence of octets with some associated character encoding scheme, transmitted over the network or stored in a file.[SGML]

typical
: Typical processing is described for many elements. This is not a mandatory part of the specification but is given as guidance for designers and to help explain the uses for which the elements were intended.

URI
: A Universal Resource Identifier is a formatted string that serves as an identifier for a resource, typcally on the Internet. URIs are used in HTML to identify the destination of hypertext links, the source of in-line images, and the object of form actions. URIs in common use include Uniform Resource Locators (URLs)[URL] and Relative URLs[RELURL].

user agent
: A component of a distributed system that presents an interface and processes requests on behalf of a user; for example, a www browser or a mail user agent.

WWW
: The World-Wide Web is a hypertext-based, distributed information system created by researchers at CERN in Switzerland. Users may create, edit or browse hypertext documents. `http://www.w3.org/'

13. Bibliography

[URI]

T. Berners-Lee. ``Universal Resource Identifiers in WWW: A Unifying Syntax for the Expression of Names and Addresses of Objects on the Network as used in the World- Wide Web.'' RFC 1630, CERN, June 1994.

[URL]

T. Berners-Lee, L. Masinter, and M. McCahill. ``Uniform Resource Locators (URL).'' RFC 1738, CERN, Xerox PARC, University of Minnesota, October 1994.

[HTTP]

T. Berners-Lee, R. T. Fielding, and H. Frystyk Nielsen. ``Hypertext Transfer Protocol - HTTP/1.0.'' Work in Progress (draft-ietf-http-v10-spec-00.ps), MIT, UC Irvine, CERN, March 1995.

[MIME]

N. Borenstein and N. Freed. ``MIME (Multipurpose Internet Mail Extensions) Part One: Mechanisms for Specifying and Describing the Format of Internet Message Bodies.'' RFC 1521, Bellcore, Innosoft, September 1993.

[RELURL]

R. T. Fielding. ``Relative Uniform Resource Locators.'' Work in Progress (draft-ietf-uri-relative-url-06.txt), UC Irvine, March 1995.

[GOLD90]

C. F. Goldfarb. ``The SGML Handbook.'' Y. Rubinsky, Ed., Oxford University Press, 1990.

[IMEDIA]

J. Postel. ``Media Type Registration Procedure.'' RFC 1590, USC/ISI, March 1994.

[IANA]

J. Reynolds and J. Postel. ``Assigned Numbers.'' STD 2, RFC 1700, USC/ISI, October 1994.

[SQ91]

SoftQuad. ``The SGML Primer.'' 3rd ed., SoftQuad Inc., 1991.

[US-ASCII]
US-ASCII. Coded Character Set - 7-Bit American Standard Code for Information Interchange. Standard ANSI X3.4-1986, ANSI, 1986.

[ISO-8859-1]
ISO 8859. International Standard -- Information Processing -- 8-bit Single-Byte Coded Graphic Character Sets -- Part 1: Latin Alphabet No. 1, ISO 8859-1:1987. Part 2: Latin alphabet No. 2, ISO 8859-2, 1987. Part 3: Latin alphabet No. 3, ISO 8859-3, 1988. Part 4: Latin alphabet No. 4, ISO 8859-4, 1988. Part 5: Latin/Cyrillic alphabet, ISO 8859-5, 1988. Part 6: Latin/Arabic alphabet, ISO 8859-6, 1987. Part 7: Latin/Greek alphabet, ISO 8859-7, 1987. Part 8: Latin/Hebrew alphabet, ISO 8859-8, 1988. Part 9: Latin alphabet No. 5, ISO 8859-9, 1990.

[SGML]
ISO 8879. Information Processing - Text and Office Systems - Standard Generalized Markup Language (SGML), 1986.

14. Appendices

These appendices are provided for informational reasons only - they do not form a part of the HTML specification.

14.1. The ISO-8859-1 Coded Character Set

This list, sorted numerically, is derived from ISO-8859-1 8-bit single-byte coded graphic character set:

REFERENCE	DESCRIPTION
&#00; - &#08;	Unused
	Horizontal tab
Line feed
&#11; - &#31;	Unused
	Space
!	Exclamation mark
"	Quotation mark
#	Number sign
$	Dollar sign
%	Percent sign
&	Ampersand
'	Apostrophe
(	Left parenthesis
)	Right parenthesis

&#42;	Asterisk
&#43;	Plus sign
&#44;	Comma
&#45;	Hyphen
&#46;	Period (fullstop)
&#47;	Solidus (slash)
0 - 9	Digits 0-9
&#58;	Colon
&#59;	Semi-colon
&#60;	Less than
&#61;	Equals sign
&#62;	Greater than
&#63;	Question mark
&#64;	Commercial at
A - Z	Letters A-Z
&#91;	Left square bracket
&#92;	Reverse solidus (backslash)
&#93;	Right square bracket
&#94;	Caret
&#95;	Horizontal bar (underscore)
&#96;	Acute accent
a - z	Letters a-z
&#123;	Left curly brace
&#124;	Vertical bar
&#125;	Right curly brace
&#126;	Tilde
&#127; -	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, grave accent
&#193;	Capital A, acute accent
&#194;	Capital A, circumflex accent
&#195;	Capital A, tilde
&#196;	Capital A, dieresis or umlaut mark
&#197;	Capital A, ring
&#198;	Capital AE dipthong (ligature)
&#199;	Capital C, cedilla
&#200;	Capital E, grave accent
&#201;	Capital E, acute accent
&#202;	Capital E, circumflex accent
&#203;	Capital E, dieresis or umlaut mark
&#204;	Capital I, grave accent
&#205;	Capital I, acute accent
&#206;	Capital I, circumflex accent
&#207;	Capital I, dieresis or umlaut mark
&#208;	Capital Eth, Icelandic
&#209;	Capital N, tilde
&#210;	Capital O, grave accent
&#211;	Capital O, acute 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, grave accent
&#218;	Capital U, acute 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, grave accent
&#225;	Small a, acute accent
&#226;	Small a, circumflex accent
&#227;	Small a, tilde
&#228;	Small a, dieresis or umlaut mark
&#229;	Small a, ring
&#230;	Small ae dipthong (ligature)
&#231;	Small c, cedilla
&#232;	Small e, grave accent
&#233;	Small e, acute accent
&#234;	Small e, circumflex accent
&#235;	Small e, dieresis or umlaut mark

```
&#236;          Small i, grave accent
&#237;          Small i, acute accent
&#238;          Small i, circumflex accent
&#239;          Small i, dieresis or umlaut mark
&#240;          Small eth, Icelandic
&#241;          Small n, tilde
&#242;          Small o, grave accent
&#243;          Small o, acute 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, grave accent
&#250;          Small u, acute 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
```

14.2. Obsolete Features

This section describes elements that are no longer part of HTML. Client implementors should implement these obsolete elements for compatibility with previous versions of the HTML specification.

14.2.1. Comment Element

The Comment element is used to delimit unneeded text and comments. The Comment element has been introduced in some HTML applications but should be replaced by the SGML comment feature in new HTML interpreters (see Section 2.2.5).

14.2.2. Highlighted Phrase Element

<HP>

The Highlighted Phrase element should be ignored if not implemented. This element has been replaced by more meaningful elements (see Section 8).

Example of use:

```
<HP1>first highlighted phrase</HP1>non-
highlighted text<HP2>second highlighted phrase</HP2> etc.
```

14.2.3. Plain Text Element

<PLAINTEXT>

The Plain Text element is used to terminates the HTML entity and to indicate that what follows is not SGML which does not require parsing. Instead, an old HTTP convention specified that what followed was an ASCII (MIME ``text/plain'') body. Its presence is an optimization. There is no closing tag.

Example of use:

```
<PLAINTEXT>
0001 This is line one of a long listing
0002 file from <ANY@HOST.INC.COM> which is sent
```

14.2.4. Example and Listing Elements

<XMP> ... </XMP> and <LISTING> ... </LISTING>

The Example and Listing elements have been replaced by the Preformatted Text element (Section 10.2).

These styles allow text of fixed-width characters to be embedded absolutely as is into the document. The syntax is:

<LISTING> ... </LISTING>

or

<XMP> ... </XMP>

The text between these tags is typically rendered in a monospaced font so that any formatting done by character spacing on successive lines will be maintained.

Between the opening and closing tags:

* The text may contain any ISO Latin-1 printable characters, except for the end-tag opener. The Example and Listing elements have historically used specifications which do not conform to SGML. Specifically, the text may contain ISO Latin printable characters, including the tag opener, as long it they does not contain the closing tag in full.
* SGML does not support this form. HTML interpreters may vary on how they interpret other tags within Example and Listing elements.
* Line boundaries within the text are rendered as a move to the beginning of the next line, except for one immediately following a start-tag or immediately

preceding an end-tag.
* The horizontal tab character must be interpreted as the smallest positive nonzero number of spaces which will leave the number of characters so far on the line as a multiple of 8. Its use is not recommended.

The Listing element is rendered so that at least 132 characters fit on a line. The Example element is rendered to that at least 80 characters fit on a line but is otherwise identical to the Listing element.

14.3. Proposed Features

This section describes proposed HTML elements and entities that are not currently supported under HTML Levels 0, 1, or 2@@, but may be supported in the future.

14.3.1. Additional Character Entities

To indicate special characters, HTML uses entity or numeric representations. Additional character presentations are proposed:

CHARACTER	REPRESENTATION
Non-breaking space	
Soft-hyphen	&shy;
Registered	&reg;
Copyright	&copy;

14.3.2. Defining Instance Element

<DFN> ... </DFN>

The Defining Instance element indicates the defining instance of a term. The typical rendering is bold or bold italic. This element is not widely supported.

14.3.3. Strike Element

<STRIKE> ... </STRIKE>

The Strike element is proposed to indicate strikethrough, a font style in which a horizontal line appears through characters. This element is not widely supported.

14.3.4. Underline Element

<U> ... </U>

The Underline element is proposed to indicate that the text should be rendered as underlined. This proposed tag is not supported by all HTML interpreters.

Example of use:

The text <U>shown here</U> is rendered in the document as underlined.

15. Acknowledgments

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, which has included the addition of in-line images introduced by the NCSA Mosaic software for WWW. Dave Raggett played an important role in deriving the FORMS material from the HTML+ specification.

Dan Connolly and Karen Olson Muldrow rewrote the HTML Specification in 1994. The document was then edited by the HTML working group as a whole, with updates being made by Eric Schieler, Mike Knezovich, and Eric W. Sink at Spyglass, Inc. Finally, Roy Fielding restructured the entire draft into its current form.

Special thanks to the many people who have contributed to this specification:

Terry Allen Marc Andreessen

Tim Berners-Lee Paul Burchard

James Clark Daniel W. Connolly

Roy T. Fielding Peter Flynn

Jay Glicksman Paul Grosso

Eduardo Gutentag Bill Hefley

Chung-Jen Ho Mike Knezovich

Tom Magliery Murray Maloney

Larry Masinter Karen Olson Muldrow

Bill Perry Dave Raggett

E. Corprew Reed Yuri Rubinsky

Eric Schieler James L. Seidman

Eric W. Sink Stuart Weibel

Chris Wilson Francois Yergeau

15.1. Authors' Addresses

Tim Berners-Lee

Director, W3 Consortium
MIT Laboratory for Computer Science
545 Technology Square
Cambridge, MA 02139, U.S.A.
Tel: +1 (617) 253 9670
Fax: +1 (617) 258 8682
Email: timbl@w3.org

Daniel W.
Connolly

Research Technical Staff, W3 Consortium
MIT Laboratory for Computer Science
545 Technology Square
Cambridge, MA 02139, U.S.A.
Fax: +1 (617) 258 8682
Email: connolly@w3.org
URI: http://www.w3.org/hypertext/WWW/People/Connolly/