T. Berners-Lee
MIT/W3C
D. Connolly
September 22, 1995

Hypertext Markup Language - 2.0

Status of this Memo

Status of this Memo

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

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

To learn the current status of any Internet-Draft, please check the `1id-abstracts.txt` listing contained in the Internet-Drafts Shadow Directories on `ftp.is.co.za` (Africa), `nic.nordu.net` (Europe), `munnari.oz.au` (Pacific Rim), `ds.internic.net` (US East Coast), or `ftp.isi.edu` (US West Coast).

Distribution of this document is unlimited. Please send comments to the HTML working group (HTML-WG) of the Internet Engineering Task Force (IETF) at `<html-wg@oclc.org>`. Discussions of the group are archived at `<URL:http://www.acl.lanl.gov/HTML_WG/archives.html>`.

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 inlined 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` Internet Media Type (RFC 1590) and MIME Content Type (RFC 1521) is defined by this specification.

Contents

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.

As HTML is an application of SGML, this specification assumes a working knowledge of [SGML].

1.1 Scope

HTML has been in use by the World-Wide Web (WWW) global information initiative since 1990. Previously, informal documentation on HTML has been available from a number of sources on the Internet. This specification brings together, clarifies, and formalizes a set of features that roughly corresponds to the capabilities of HTML in common use prior to June 1994. A number of new features to HTML are being proposed and experimented in the Internet community.

This document thus defines a HTML 2.0 (to distinguish it from the previous informal specifications). Future (generally upwardly compatible) versions of HTML with new features will be released with higher version numbers.

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`. 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 aspects of the behavior of HTML user agents.

1.2.1 Documents

A document is a conforming HTML document if:

- It is a conforming SGML document, and it conforms to the HTML DTD (see Section 9.1).

  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 identified in notes like this throughout this specification.

- 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 [ISO-10646]; that is, each code position listed in Section 13 is included, and each code position in the document character set is mapped to the same character as [ISO-10646] 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 [ISO-10646] repertoire. The critical distinction is that numeric character references agree with [ISO-10646] regardless of how the document is encoded.

1.2.2 Feature Test Entities

The HTML DTD defines a standard HTML document type and several variations, by way of feature test entities. Feature test entities are declarations in the HTML DTD that control the inclusion or exclusion of portions of the DTD.

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 selects a more prescriptive document type definition that eliminates those features. It is set to `IGNORE` by default.

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 and implemented inconsistently, and their use is deprecated. This feature test entity enables a document type definition that allows these features. It is set to `INCLUDE` by default.

Documents generated by translation software or editing software should not contain deprecated idioms.

1.2.3 User Agents

An HTML user agent conforms to this specification if:

- It parses the characters of an HTML document into data characters and markup according to [SGML].

  NOTE – In the interest of robustness and extensibility, there are a number of widely deployed conventions for handling non-conforming documents. See Section 4.2.1 for details.

- It supports the `ISO-8859-1` character encoding scheme and processes each character in the ISO Latin Alphabet No. 1 as specified in Section 6.1.

  NOTE – To support non-western writing systems, HTML user agents are encouraged to support `ISO-10646-UCS-2` or similar character encoding schemes and as much of the character repertoire of [ISO-10646] as is practical.

- 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 behavior of conforming user agents.

- It allows the user to traverse (or at least attempt to traverse, resources permitting) all hyperlinks from `<A>` elements in an HTML document.

An HTML user agent is a level 2 user agent if, additionally:

- 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.

2 Terms

absolute URI
a URI in absolute form; for example, as per [URL]

anchor
one of two ends of a hyperlink; typically, a phrase marked as an `<A>` element.

base URI
an absolute URI used in combination with a relative URI to determine another absolute URI.

character
An atom of information, for example a letter or a digit. Graphic characters have associated glyphs, whereas 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`.

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, usually 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]

fragment identifier
: the portion of an `HREF` attribute value following the `#` character which modifies the presentation of the destination of a hyperlink.

form data set
: a sequence of name/value pairs; the names are given by an HTML document and the values are given by a user.

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

hyperlink
: a relationship between two anchors, called the head and the tail. The link goes from the tail to the head. The head and tail are also known as destination and source, respectively.

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.

media type
: an Internet Media Type, as per [IMEDIA].

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]

minimally conforming HTML user agent
: A user agent that conforms to this specification except for form processing. It may only process level 1 HTML documents.

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

numeric character reference
: markup that refers to a character by its code position in the document character set.

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 Uniform Resource Identifier is a formatted string that serves as an identifier for a resource, typically on the Internet. URIs are used in HTML to identify the anchors of hyperlinks. URIs in common practice 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. `<URL:http://www.w3.org/>`

3 HTML as an Application of SGML

HTML is an application of ISO 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 Section 9.

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

3.1 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.

In the SGML specification, the first production of the SGML syntax 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 that is given in Section 9.5. According to its document character set, `*` refers to an asterisk character, `*`.

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

1. start-tag: TITLE
2. data characters: “Parsing Example”

3. end-tag: TITLE
4. start-tag: P
5. data characters “Some text. ”
6. start-tag: EM
7. data characters: “*wow*”
8. end-tag: EM
9. end-tag: P

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` (Section 9.1). The terminals above parse as:

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

Some of the elements are delimited explicitly by tags, while the boundaries of others are inferred. The <HTML> element contains a <HEAD> element and a <BODY> element. The <HEAD> contains <TITLE>, which is explicitly delimited by start- and end-tags.

3.2 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 repertoire of [ISO-646]. Data characters are drawn from the document character set (see Section 6).

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.

3.2.1 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 map to single-character strings, via the document character set. Each reference to one of the general entities defined in the HTML DTD maps to a single-character string.

For example,

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

The terminating semicolon on entity or numeric character references is only necessary when the character following the reference would otherwise be recognized as part of the name (see 9.4.5 “Reference End” in [SGML]).

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

An ampersand is only recognized as markup when it is followed by a letter or a # and a 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
&	&	&	Ampersand
<	<	<	Less than
>	>	>	Greater than

NOTE – There are SGML mechanisms, CDATA and RCDATA declared content, that allow most <, >, and & characters to be entered without the use of entity references. Because these mechanisms tend to be used and implemented inconsistently, and because they conflict with techniques for reducing HTML to 7 bit ASCII for transport, they are deprecated in this version of HTML. See Section 5.5.2.1.

3.2.2 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, use the
 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, Section 9.1 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.

3.2.3 Names

A name consists of a letter followed by letters, digits, periods, or hyphens. The length of a name is limited to 72 characters by the NAMELEN parameter in the SGML declaration for HTML, Section 9.5. Element and attribute 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 `<`.

3.2.4 Attributes

In a start-tag, white space and attributes are allowed between the element name and the closing delimiter. An attribute specification typically consists of an attribute name, an equal sign, and a value, though some attribute specifications may be just a name token. 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.

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

- A name token (a sequence of letters, digits, periods, or hyphens). Name tokens are not case sensitive.

  NOTE – Some historical implementations allow any character except space or `>` in a name token.

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'>
```

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

```
          ENTITY      NUMERIC
CHARACTER REFERENCE   CHAR REF     CHARACTER DESCRIPTION
--------- ----------  -----------  ---------------------
  HT                  &#9;         Tab
  LF                  &#10;        Line Feed
```

CR	
Carriage Return
SP			Space
"	"	"	Quotation mark
&	&	&	Ampersand

For example:

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

The NAMELEN parameter in the SGML declaration (Section 9.5) limits the length of an attribute value to 1024 characters.

Attributes such as ISMAP and COMPACT may be written using a minimized syntax (see 7.9.1.2 “Omitted Attribute Name” in [SGML]). The markup:

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

can be written using a minimized syntax:

```
<UL COMPACT>
```

> **NOTE –** Some historical implementations only understand the minimized syntax.

3.2.5 Comments

To include comments in an HTML document, use a comment declaration. A comment declaration consists of `<!` followed by zero or more comments followed by `>`. Each comment starts with − and includes all text up to and including the next occurrence of −. In a comment declaration, white space is allowed after each comment, but not before the first comment. The entire comment declaration is ignored.

> **NOTE –** Some historical HTML implementations incorrectly consider any > character to be the termination of a comment.

For example:

```
<!DOCTYPE HTML PUBLIC "-//IETF//DTD HTML 2.0//EN">
<HEAD>
<TITLE>HTML Comment Example</TITLE>
<!-- Id: html-sgml.sgm,v 1.5 1995/05/26 21:29:50 connolly Exp  -->
<!-- another -- -- comment -->
<!>
</HEAD>
<BODY>
<p> <!- not a comment, just regular old data characters ->
```

3.3 HTML Public Text Identifiers

To identify information as an HTML document conforming to this specification, each document must start with one of the following document type declarations.

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

This document type declaration refers to the HTML DTD in Section 9.1.

> **NOTE –** If the body of a `text/html` message entity does not begin with a document type declaration, an HTML user agent should infer the above document type declaration.

```
<!DOCTYPE HTML PUBLIC "-//IETF//DTD HTML 2.0 Level 2//EN">
```

This document type declaration also refers to the HTML DTD which appears in Section 9.1.

```
<!DOCTYPE HTML PUBLIC "-//IETF//DTD HTML 2.0 Level 1//EN">
```

This document type declaration refers to the level 1 HTML DTD in Section 9.3. Form elements must not occur in level 1 documents.

```
<!DOCTYPE HTML PUBLIC "-//IETF//DTD HTML 2.0 Strict//EN">
<!DOCTYPE HTML PUBLIC "-//IETF//DTD HTML 2.0 Strict Level 1//EN">
```

These two document type declarations refer to the HTML DTD in Section 9.2 and Section 9.4. They refer to the more structurally rigid definition of HTML.

HTML user agents may support other document types. In particular, they may support other formal public identifiers, or other document types altogether. They may support an internal declaration subset with supplemental entity, element, and other markup declarations.

3.4 Example HTML Document

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

4 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 level 1 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). Level 2 HTML user agents should support form entry and submission.

4.1 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
level, charset

Encoding considerations
any encoding is allowed

Security considerations
see Section 10

The optional parameters are defined as follows:

Level
The level parameter specifies the feature set used in the document. The level is an integer number, implying that any features of same or lower level may be present in the document. Level 1 is all features defined in this specification except those that require the `<FORM>` element. Level 2 includes form processing. Level 2 is the 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 [HTTP].

4.2 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.

4.2.1 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; &amp; &beta; be finite sets.
  => "Let &alpha; & &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 does not conform to this specification.

4.2.2 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 13, 10.

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 a CR LF sequence. 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 treat any of the three common representations of end-of-line as starting a new line.

5 Document Structure

An HTML document is a tree of elements, including a head and body, headings, paragraphs, lists, etc. Form elements are discussed in Section 8.

5.1 Document Element: HTML

The HTML document element consists of a head and a body, much like a memo or a mail message. The head contains the title and optional elements. The body is a text flow consisting of paragraphs, lists, and other elements.

5.2 Head: HEAD

The head of an HTML document is an unordered collection of information about the document. For example:

```
<!DOCTYPE HTML PUBLIC "-//IETF//DTD HTML 2.0//EN">
<HEAD>
<TITLE>Introduction to HTML</TITLE>
</HEAD>
...
```

5.2.1 Title: TITLE

Every HTML document must contain a `<TITLE>` element.

The title should identify the contents of the document in a global context. A short title, such as “Introduction” may be meaningless out of context. A title such as “Introduction to HTML Elements” is more appropriate.

> **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.

A user agent may display the title of a document in a history list or as a label for the window displaying the document. This differs from headings (Section 5.4), which are typically displayed within the body text flow.

5.2.2 Base Address: BASE

The optional `<BASE>` element provides a base address for interpreting relative URLs when the document is read out of context (see Section 7). The value of the `HREF` attribute must be an absolute URI.

5.2.3 Keyword Index: ISINDEX

The `<ISINDEX>` element indicates that the user agent should allow the user to search an index by giving keywords. See Section 7.5 for details.

5.2.4 Link: LINK

The `<LINK>` element represents a hyperlink (see Section 7). Any number of LINK elements may occur in the `<HEAD>` element of an HTML document. It has the same attributes as the `<A>` element (see Section 5.7.3).

The `<LINK>` element is typically used to indicate authorship, related indexes and glossaries, older or more recent versions, document hierarchy, associated resources such as style sheets, etc.

5.2.5 Associated Meta-information: META

The `<META>` element is an extensible container for use in identifying specialized document meta-information. Meta-information has two main functions:

- to provide a means to discover that the data set exists and how it might be obtained or accessed; and
- to document the content, quality, and features of a data set, indicating its fitness for use.

Each `<META>` element specifies a name/value pair. 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.

> **NOTE –** The `<META>` element should not be used where a specific element, such as `<TITLE>`, would be more appropriate. Rather than a `<META>` element with a URI as the value of the `CONTENT` attribute, use a `<LINK>` element.

HTTP servers may read the content of the document `<HEAD>` to generate header fields corresponding to any elements defining a value for the attribute `HTTP-EQUIV`.

> **NOTE –** The method by which the server extracts document meta-information is unspecified and not mandatory. The `<META>` element only provides an extensible mechanism for identifying and embedding document meta-information — how it may be used is up to the individual server implementation and the HTML user agent.

Attributes of the META element:

HTTP-EQUIV
: binds the element to an HTTP header field. An HTTP server may use this information to process the document. In particular, it may include a header field in the responses to requests for this document: the header name is taken from the `HTTP-EQUIV` attribute value, and the header value is taken from the value of the `CONTENT` attribute. HTTP header names are not case sensitive.

NAME
: specifies the name of the name/value pair. If not present, `HTTP-EQUIV` gives the name.

CONTENT
: specifies the value of the name/value pair.

Examples

If the document contains:

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

then the server may include the following header fields:

```
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.

An HTTP server must not use the `<META>` element to form an HTTP response header unless the `HTTP-EQUIV` attribute is present.

An HTTP server may disregard any `<META>` elements that specify information controlled by the HTTP server, for example `Server`, `Date`, and `Last-modified`.

5.2.6 Next Id: NEXTID

The <NEXTID> element is included for historical reasons only. HTML documents should not contain <NEXTID> elements.

The <NEXTID> element gives a hint for the name to use for a new <A> element when editing an HTML document. It should be distinct from all NAME attribute values on <A> elements. For example:

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

5.3 Body: BODY

The <BODY> element contains the text flow of the document, including headings, paragraphs, lists, etc.

For example:

```
<BODY>
<h1>Important Stuff</h1>
<p>Explanation about important stuff...
</BODY>
```

5.4 Headings: H1 ... H6

The six heading elements, <H1> through <H6>, denote section headings. Although the order and occurrence of headings is not constrained by the HTML DTD, documents should not skip levels (for example, from H1 to H3), as converting such documents to other representations is often problematic.

Example of use:

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

Typical renderings are:

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

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

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

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

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

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

5.5 Block Structuring Elements

Block structuring elements include paragraphs, lists, and block quotes. They must not contain heading elements, but they may contain phrase markup, and in some cases, they may be nested.

5.5.1 Paragraph: P

The `<P>` element indicates a paragraph. The exact indentation, leading space, etc. of a paragraph is not specified 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. The first line in a paragraph is indented in some cases.

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>
```

5.5.2 Preformatted Text: PRE

The `<PRE>` element represents a character cell block of text and is suitable for text that has been formatted for a monospaced font.

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 user agent to select a suitable font and indentation.

Within preformatted text:

- Line breaks within the text are rendered as a move to the beginning of the next 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.

- Anchor elements and phrase markup may be used.

  NOTE – Constraints on the processing of `<PRE>` content may limit or prevent the ability of the HTML user agent to faithfully render phrase markup.

- Elements that define paragraph formatting (headings, address, etc.) must not be used.

  NOTE – Some historical documents contain `<P>` tags in `<PRE>` elements. User agents are encouraged to treat this as 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.

- The horizontal tab character (code position 9 in the HTML document character set) 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. Documents should not contain tab characters, as they are not supported consistently.

Example of use:

```
<PRE>
Line 1.
       Line 2 is to the right of line 1.     <a href="abc">abc</a>
       Line 3 aligns with line 2.            <a href="def">def</a>
</PRE>
```

5.5.2.1 Example and Listing: XMP, LISTING

The <XMP> and <LISTING> elements are similar to the <PRE> element, but they have a different syntax. Their content is declared as CDATA, which means that no markup except the end-tag open delimiter-in-context is recognized (see 9.6 “Delimiter Recognition” of [SGML]).

> **NOTE –** In a previous draft of the HTML specification, the syntax of <XMP> and <LISTING> elements allowed closing tags to be treated as data characters, as long as the tag name was not <XMP> or <LISTING>, respectively.

Since CDATA declared content has a number of unfortunate interactions with processing techniques and tends to be used and implemented inconsistently, HTML documents should not contain <XMP> nor <LISTING> elements — the <PRE> tag is more expressive and more consistently supported.

The <LISTING> element should be rendered so that at least 132 characters fit on a line. The <XMP> element should be rendered so that at least 80 characters fit on a line but is otherwise identical to the <LISTING> element.

> **NOTE –** In a previous draft, HTML included a <PLAINTEXT> element that is similar to the <LISTING> element, except that there is no closing tag: all characters after the <PLAINTEXT> start-tag are data.

5.5.3 Address: ADDRESS

The <ADDRESS> element contains such information as address, signature and authorship, often at the beginning or end of the body of a document.

Typically, the <ADDRESS> element is rendered in an italic typeface and may be indented.

Example of use:

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

5.5.4 Block Quote: BLOCKQUOTE

The <BLOCKQUOTE> element contains text quoted from another source.

A typical rendering might be a slight extra left and right indent, and/or italic font. The <BLOCKQUOTE> 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 play ends
<BLOCKQUOTE>
<P>Soft you now, the fair Ophelia. Nymph, in thy orisons, be all
my sins remembered.
</BLOCKQUOTE>
but I am not sure.
```

5.6 List Elements

HTML includes a number of list elements. They may be used in combination; for example, a <OL> may be nested in an <LI> element of a <UL>.

The COMPACT attribute suggests that a compact rendering be used.

5.6.1 Unordered List: UL, LI

The <UL> represents a list of items — typically rendered as a bulleted list.

The content of a <UL> element is a sequence of <LI> elements. For example:

```
<UL>
<LI>First list item
<LI>Second list item
 <p>second paragraph of second item
<LI>Third list item
</UL>
```

5.6.2 Ordered List: OL

The <OL> element represents an ordered list of items, sorted by sequence or order of importance. It is typically rendered as a numbered list.

The content of a <OL> element is a sequence of <LI> elements. For example:

```
<OL>
<LI>Click the Web button to open URI window.
<LI>Enter the URI number in the text field of the Open URI
window. The Web document you specified is displayed.
  <ol>
   <li>substep 1
   <li>substep 2
  </ol>
<LI>Click highlighted text to move from one link to another.
</OL>
```

5.6.3 Directory List: DIR

The <DIR> element is similar to the <UL> element. It represents a list of short items, typically up to 20 characters each. Items in a directory list may be arranged in columns, typically 24 characters wide.

The content of a <DIR> element is a sequence of <LI> elements. Nested block elements are not allowed in the content of <DIR> elements. For example:

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

5.6.4 Menu List: MENU

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

The content of a <MENU> element is a sequence of <LI> elements. Nested block elements are not allowed in the content of <MENU> elements. For example:

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

5.6.5 Definition List: DL, DT, DD

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.

The content of a <DL> element is a sequence of <DT> elements and/or <DD> elements, usually in pairs. Multiple <DT> may be paired with a single <DD> element. Documents should not contain multiple consecutive <DD> elements.

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 (typically one third of the display area), it may be extended across the page with the DD section moved to the next line, or it may be wrapped onto successive lines of the left hand column.

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

Unless the COMPACT attribute is present, an HTML user agent may leave white space between successive DT, DD pairs. The COMPACT attribute may also reduce the width of the left-hand (DT) column.

```
<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>
```

5.7 Phrase Markup

Phrases may be marked up according to idiomatic usage, typographic appearance, or for use as hyperlink anchors.

User agents must render highlighted phrases 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.

Phrase elements may be nested within the content of other phrase elements; however, HTML user agents may render nested phrase elements indistinctly from non-nested elements:

```
plain <B>bold <I>italic</I></B> may be rendered
```

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

5.7.1 Idiomatic Elements

Phrases may be marked up to indicate certain idioms.

> **NOTE –** User agents may support the `<DFN>` element, not included in this specification, as it has been deployed to some extent. It is used to indicate the defining instance of a term, and it is typically rendered in italic or bold italic.

5.7.1.1 Citation: CITE

The `<CITE>` element is used to indicate the title of a book or other citation. It is typically rendered as italics. For example:

```
He just couldn't get enough of <cite>The Grapes of Wrath</cite>.
```

5.7.1.2 Code: CODE

The `<CODE>` element indicates an example of code, typically rendered in a mono-spaced font. The `<CODE>` element is intended for short words or phrases of code; the `<PRE>` block structuring element (Section 5.5.2) is more appropriate for multiple-line listings. For example:

```
The expression <code>x += 1</code>
is short for <code>x = x + 1</code>.
```

5.7.1.3 Emphasis: EM

The `<EM>` element indicates an emphasized phrase, typically rendered as italics. For example:

```
A singular subject <em>always</em> takes a singular verb.
```

5.7.1.4 Keyboard: KBD

The `<KBD>` element indicates text typed by a user, typically rendered in a mono-spaced font. This is commonly used in instruction manuals. For example:

```
Enter <kbd>FIND IT</kbd> to search the database.
```

5.7.1.5 Sample: SAMP

The <SAMP> element indicates a sequence of literal characters, typically rendered in a monospaced font. For example:

```
The only word containing the letters <samp>mt</samp> is dreamt.
```

5.7.1.6 Strong Emphasis: STRONG

The <STRONG> element indicates strong emphasis, typically rendered in bold. For example:

```
<strong>STOP</strong>, or I'll say "<strong>STOP</strong>" again!
```

5.7.1.7 Variable: VAR

The <VAR> element indicates a placeholder variable, typically rendered as italic. For example:

```
Type <SAMP>html-check <VAR>file</VAR> | more</SAMP>
to check <VAR>file</VAR> for markup errors.
```

5.7.2 Typographic Elements

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

Typical renderings for idiomatic elements may vary between user agents. If a specific rendering is necessary — for example, when referring to a specific text attribute as in "The italic parts are mandatory" — a typographic element can be used to ensure that the intended typography is used where possible.

> **NOTE –** User agents may support some typographic elements not included in this specification, as they have been deployed to some extent. The <STRIKE> element indicates horizontal line through the characters, and the <U> element indicates an underline.

5.7.2.1 Bold: B

The <B> element indicates bold text. Where bold typography is unavailable, an alternative representation may be used.

5.7.2.2 Italic: I

The <I> element indicates italic text. Where italic typography is unavailable, an alternative representation may be used.

5.7.2.3 Teletype: TT

The `<TT>` element indicates teletype (monospaced)text. Where a teletype font is unavailable, an alternative representation may be used.

5.7.3 Anchor: A

The `<A>` element indicates a hyperlink anchor (see Section 7). At least one of the `NAME` and `HREF` attributes should be present. Attributes of the `<A>` element:

HREF
: gives the URI of the head anchor of a hyperlink.

NAME
: gives the name of the anchor, and makes it available as a head of a hyperlink.

TITLE
: suggests a title for the destination resource — advisory only. The `TITLE` attribute may be used:

  - for display prior to accessing the destination resource, 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;
  - for resources that do not include a title, such as graphics, plain text and Gopher menus, for use as a window title.

REL
: The `REL` attribute gives the relationship(s) described by the hyperlink. The value is a whitespace separated list of relationship names. The semantics of link relationships are not specified in this document.

REV
: same as the REL attribute, but the semantics of the relationship 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.

URN
: specifies a preferred, more persistent identifier for the head anchor of the hyperlink. The syntax and semantics of the `URN` attribute are not yet specified.

METHODS
: specifies methods to be used in accessing the destination, as a whitespace-separated list of names. The set of applicable names is a function of the scheme of the URI in the `HREF` attribute. For similar reasons as for the TITLE attribute, it may be useful to include the information in advance in the link. For example, the HTML user agent may chose a different rendering as a function of the methods allowed; for example, something that is searchable may get a different icon.

5.8 Line Break: BR

The
 element specifies a line break between words (see Section 6). For example:

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

5.9 Horizontal Rule: HR

The <HR> element is a divider between sections of text; typically a full width horizontal rule or equivalent graphic. For example:

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

5.10 Image: IMG

The <IMG> element refers to an image or icon via a hyperlink (see Section 7.3).

HTML user agents may process the value of the ALT attribute as an alternative to processing the image resource indicated by the SRC attribute.

> **NOTE –** Some HTML user agents can process graphics linked via anchors, but not <IMG> graphics. If a graphic is essential, it should be referenced from an <A> element rather than an <IMG> element. If the graphic is not essential, then the <IMG> element is appropriate.

Attributes of the <IMG> element:

ALIGN
: alignment of the image with respect to the text baseline.

- TOP specifies that the top of the image aligns with the tallest item on the line containing the image.
- MIDDLE specifies that the center of the image aligns with the baseline of the line containing the image.
- BOTTOM specifies that the bottom of the image aligns with the baseline of the line containing the image.

ALT
text to use in place of the referenced image resource, for example due to processing constraints or user preference.

ISMAP
indicates an image map (see Section 7.6).

SRC
specifies the URI of the image resource.

> **NOTE –** In practice, the media types of image resources are limited to a few raster graphic formats: typically `image/gif`, `image/jpeg`. In particular, `text/html` resources are not intended to be used as image resources.

Examples of use:

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

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

6 Characters, Words, and Paragraphs

An HTML user agent should present the body of an HTML document as a collection of typeset paragraphs and preformatted text. Except for preformatted elements (`<PRE>`, `<XMP>`, `<LISTING>`, `<TEXTAREA>`), 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.

6.1 The HTML Document Character Set

The document character set specified in Section 9.5 must be supported by HTML user agents. It includes the graphic characters of Latin Alphabet No. 1, or simply Latin-1. Latin-1 comprises 191 graphic characters, including the alphabets of most Western European languages.

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

NOTE – To support non-western writing systems, a larger character repertoire will be specified in a future version of HTML. The document character set will be [ISO-10646], or some subset that agrees with [ISO-10646]; in particular, all numeric character references must use code positions assigned by [ISO-10646].

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 the HTML document character set only three control characters are allowed: Horizontal Tab, Carriage Return, and Line Feed (code positions 9, 13, and 10).

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

```
Kurt G&ouml;del was a famous logician and mathematician.
```

See Section 9.7.2 for a table of the “Added Latin 1” entities, and Section 13 for a table of the code positions of [ISO 8859-1] and the control characters in the HTML document character set.

7 Hyperlinks

In addition to general purpose elements such as paragraphs and lists, HTML documents can express hyperlinks. An HTML user agent allows the user to navigate these hyperlinks.

A hyperlink is a relationship between two anchors, called the head and the tail of the hyperlink[DEXTER]. Anchors are identified by an anchor address: an absolute Uniform Resource Identifier (URI), optionally followed by a '#' and a sequence of characters called a fragment identifier. For example:

```
http://www.w3.org/hypertext/WWW/TheProject.html
http://www.w3.org/hypertext/WWW/TheProject.html#z31
```

In an anchor address, the URI refers to a resource; it may be used in a variety of information retrieval protocols to obtain an entity that represents the resource, such as an HTML document. The fragment identifier, if present, refers to some view on, or portion of the resource.

Each of the following markup constructs indicates the tail anchor of a hyperlink or set of hyperlinks:

- `<A>` elements with `HREF` present.
- `<LINK>` elements.
- `<IMG>` elements.

- `<INPUT>` elements with the `SRC` attribute present.
- `<ISINDEX>` elements.
- `<FORM>` elements with `METHOD=GET`.

These markup constructs refer to head anchors by a URI, either absolute or relative, or a fragment identifier, or both.

In the case of a relative URI, the absolute URI in the address of the head anchor is the result of combining the relative URI with a base absolute URI as in [RELURL]. The base document is taken from the document's `<BASE>` element, if present; else, it is determined as in [RELURL].

7.1 Accessing Resources

Once the address of the head anchor is determined, the user agent may obtain a representation of the resource.

For example, if the base URI is `http://host/x/y.html` and the document contains:

```
<img src="../icons/abc.gif">
```

then the user agent uses the URI `http://host/icons/abc.gif` to access the resource, as in [URL]..

7.2 Activation of Hyperlinks

An HTML user agent allows the user to navigate the content of the document and request activation of hyperlinks denoted by `<A>` elements. HTML user agents should also allow activation of `<LINK>` element hyperlinks.

To activate a link, the user agent obtains a representation of the resource identified in the address of the head anchor. If the representation is another HTML document, navigation may begin again with this new document.

7.3 Simultaneous Presentation of Image Resources

An HTML user agent may activate hyperlinks indicated by `<IMG>` and `<INPUT>` elements concurrently with processing the document; that is, image hyperlinks may be processed without explicit request by the user. Image resources should be embedded in the presentation at the point of the tail anchor, that is the `<IMG>` or `<INPUT>` element.

<LINK> hyperlinks may also be processed without explicit user request; for example, style sheet resources may be processed before or during the processing of the document.

7.4 Fragment Identifiers

Any characters following a # character in a hypertext address constitute a fragment identifier. In particular, an address of the form #fragment refers to an anchor in the same document.

The meaning of fragment identifiers depends on the media type of the representation of the anchor's resource. For text/html representations, it refers to the <A> element with a NAME attribute whose value is the same as the fragment identifier. The matching is case sensitive. The document should have exactly one such element. The user agent should indicate the anchor element, for example by scrolling to and/or highlighting the phrase.

For example, if the base URI is http://host/x/y.html and the user activated the link denoted by the following markup:

```
<p> See: <a href="app1.html#bananas">appendix 1</a>
for more detail on bananas.
```

Then the user agent accesses the resource identified by http://host/x/app1.html. Assuming the resource is represented using the text/html media type, the user agent must locate the <A> element whose NAME attribute is bananas and begin navigation there.

7.5 Queries and Indexes

The <ISINDEX> element represents a set of hyperlinks. The user can choose from the set by providing keywords to the user agent. The user agent computes the head URI by appending ? and the keywords to the base URI. The keywords are escaped according to [URL] and joined by +. For example, if a document contains:

```
<BASE HREF="http://host/index">
<ISINDEX>
```

and the user provides the keywords apple and berry, then the user agent must access the resource http://host/index?apple+berry.

<FORM> elements with METHOD=GET also represent sets of hyperlinks. See Section 8.2.2 for details.

7.6 Image Maps

If the ISMAP attribute is present on an <IMG> element, the <IMG> element must be contained in an <A> element with an HREF present. This construct represents a set of hyperlinks. The user can choose from the set by choosing a pixel of the image. The user agent computes the head URI by appending ? and the x and y coordinates of the pixel to the URI given in the <A> element. For example, if a document contains:

```
<!DOCTYPE HTML PUBLIC "-//IETF//DTD HTML 2.0//EN">
<head><title>ImageMap Example</title>
<BASE HREF="http://host/index"></head>
<body>
<p> Choose any of these icons:<br>
<a href="/cgi-bin/imagemap"><img ismap src="icons.gif"></a>
```

and the user chooses the upper-leftmost pixel, the chosen hyperlink is the one with the URI http://host/cgi-bin/imagemap?0,0.

8 Forms

A form is a template for a form data set and an associated method and action URI. A form data set is a sequence of name/value pair fields. The names are specified on the NAME attributes of form input elements, and the values are given initial values by various forms of markup and edited by the user. The resulting form data set is used to access an information service as a function of the action and method.

Forms elements can be mixed in with document structuring elements. For example, a <PRE> element may contain a <FORM> element, or a <FORM> element may contain lists which contain <INPUT> elements. This gives considerable flexibility in designing the layout of forms.

Form processing is a level 2 feature.

8.1 Form Elements

8.1.1 Form: FORM

The <FORM> element contains a sequence of input elements, along with document structuring elements. The attributes are:

ACTION
: specifies the action URI for the form. The action URI of a form defaults to the base URI of the document (see Section 7).

METHOD
: selects a method of accessing the action URI. The set of applicable methods is a function of the scheme of the action URI of the form. See Section 8.2.2 and Section 8.2.3.

ENCTYPE
: specifies the media type used to encode the name/value pairs for transport, in case the protocol does not itself impose a format. See Section 8.2.1.

8.1.2 Input Field: INPUT

The `<INPUT>` element represents a field for user input. The `TYPE` attribute discriminates between several variations of fields.

The `<INPUT>` element has a number of attributes. The set of applicable attributes depends on the value of the `TYPE` attribute.

8.1.2.1 Text Field: INPUT TYPE=TEXT

The default value of the `TYPE` attribute is `TEXT`, indicating a single line text entry field. (Use the `<TEXTAREA>` element for multi-line text fields.)

Required attributes are:

NAME
: name for the form field corresponding to this element.

The optional attributes are:

MAXLENGTH
: constrains the number of characters that can be entered into a text input field. If the value of `MAXLENGTH` is greater the the value of the `SIZE` attribute, the field should scroll appropriately. The default number of characters is unlimited.

SIZE
: specifies the amount of display space allocated to this input field according to its type. The default depends on the user agent.

VALUE
: The initial value of the field.

For example:

```
<p>Street Address: <input name=street><br>
Postal City code: <input name=city size=16 maxlength=16><br>
Zip Code: <input name=zip size=10 maxlength=10 value="99999-9999"><br>
```

8.1.2.2 Password Field: INPUT TYPE=PASSWORD

An `<INPUT>` element with `TYPE=PASSWORD` is a text field as above, except that the value is obscured as it is entered. (see also: Section 10).

For example:

```
<p>Name: <input name=login> Password: <input type=password name=passwd>
```

8.1.2.3 Check Box: INPUT TYPE=CHECKBOX

An `<INPUT>` element with `TYPE=CHECKBOX` represents a boolean choice. A set of such elements with the same name represents an n-of-many choice field. Required attributes are:

NAME
: symbolic name for the form field corresponding to this element or group of elements.

VALUE
: The portion of the value of the field contributed by this element.

Optional attributes are:

CHECKED
: indicates that the initial state is on.

For example:

```
<p>What flavors do you like?
<input type=checkbox name=flavor value=vanilla>Vanilla<br>
<input type=checkbox name=flavor value=strawberry>Strawberry<br>
<input type=checkbox name=flavor value=chocolate checked>Chocolate<br>
```

8.1.2.4 Radio Button: INPUT TYPE=RADIO

An `<INPUT>` element with `TYPE=RADIO` represents a boolean choice. A set of such elements with the same name represents a 1-of-many choice field. The `NAME` and `VALUE` attributes are required as for check boxes. Optional attributes are:

CHECKED
: indicates that the initial state is on.

At all times, exactly one of the radio buttons in a set is checked. If none of the <INPUT> elements of a set of radio buttons specifies CHECKED, then the user agent must check the first radio button of the set initially.

For example:

```
<p>Which is your favorite?
<input type=radio name=flavor value=vanilla>Vanilla<br>
<input type=radio name=flavor value=strawberry>Strawberry<br>
<input type=radio name=flavor value=chocolate>Chocolate<br>
```

8.1.2.5 Image Pixel: INPUT TYPE=IMAGE

An <INPUT> element with TYPE=IMAGE specifies an image resource to display, and allows input of two form fields: the x and y coordinate of a pixel chosen from the image. The names of the fields are the name of the field with .x and .y appended. TYPE=IMAGE implies TYPE=SUBMIT processing; that is, when a pixel is chosen, the form as a whole is submitted.

The NAME attribute is required as for other input fields. The SRC attribute is required and the ALIGN is optional as for the <IMG> element (see Section 5.10).

For example:

```
<p>Choose a point on the map:
<input type=image name=point src="map.gif">
```

8.1.2.6 Hidden Field: INPUT TYPE=HIDDEN

An <INPUT> element with TYPE=HIDDEN represents a hidden field.The user does not interact with this field; instead, the VALUE attribute specifies the value of the field. The NAME and VALUE attributes are required.

For example:

```
<input type=hidden name=context value="l2k3j4l2k3j4l2k3j4lk23">
```

8.1.2.7 Submit Button: INPUT TYPE=SUBMIT

An <INPUT> element with TYPE=SUBMIT represents an input option, typically a button, that instructs the user agent to submit the form. Optional attributes are:

NAME
: indicates that this element contributes a form field whose value is given by the VALUE attribute. If the NAME attribute is not present, this element does not contribute a form field.

VALUE
: indicates a label for the input (button).

```
You may submit this request internally:
<input type=submit name=recipient value=internal><br>
or to the external world:
<input type=submit name=recipient value=world>
```

8.1.2.8 Reset Button: INPUT TYPE=RESET

An <INPUT> element with TYPE=RESET represents an input option, typically a button, that instructs the user agent to reset the form's fields to their initial states. The VALUE attribute, if present, indicates a label for the input (button).

```
When you are finished, you may submit this request:
<input type=submit><br>
You may clear the form and start over at any time: <input type=reset>
```

8.1.3 Selection: SELECT

The <SELECT> element constrains the form field to an enumerated list of values. The values are given in <OPTION> elements. Attributes are:

MULTIPLE
: indicates that more than one option may be included in the value.

NAME
: specifies the name of the form field.

SIZE
: specifies the number of visible items. Select fields of size one are typically pop-down menus, whereas select fields with size greater than one are typically lists.

For example:

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

The initial state has the first option selected, unless a SELECTED attribute is present on any of the <OPTION> elements.

8.1.3.1 Option: OPTION

The Option element can only occur within a Select element. It represents one choice, and has the following attributes:

SELECTED
Indicates that this option is initially selected.

VALUE
indicates the value to be returned if this option is chosen. The field value defaults to the content of the <OPTION> element.

The content 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.

8.1.4 Text Area: TEXTAREA

The <TEXTAREA> element represents a multi-line text field. Attributes are:

COLS
the number of visible columns to display for the text area, in characters.

NAME
Specifies the name of the form field.

ROWS
The number of visible rows to display for the text area, in characters.

For example:

```
<TEXTAREA NAME="address" ROWS=6 COLS=64>
HaL Computer Systems
1315 Dell Avenue
Campbell, California 95008
</TEXTAREA>
```

The content of the <TEXTAREA> element is the field's initial value.

Typically, the ROWS and COLS attributes determine the visible dimension of the field in characters. The field is typically rendered in a fixed-width font. HTML user agents should allow text to extend beyond these limits by scrolling as needed.

8.2 Form Submission

An HTML user agent begins processing a form by presenting the document with the fields in their initial state. The user is allowed to modify the fields, constrained by the field type etc. When the user indicates that the form should be submitted (using a submit button or image input), the form data set is processed according to its method, action URI and enctype.

When there is only one single-line text input field in a form, the user agent should accept Enter in that field as a request to submit the form.

8.2.1 The form-urlencoded Media Type

The default encoding for all forms is application/x-www-form-urlencoded. A form data set is represented in this media type as follows:

1. The form field names and values are escaped: space characters are replaced by +, and then reserved characters are escaped as per [URL]; that is, non-alphanumeric characters are replaced by %HH, a percent sign and two hexadecimal digits representing the ASCII code of the character. Line breaks, as in multi-line text field values, are represented as CR LF pairs, i.e. %0D%0A.
2. The fields are listed in the order they appear in the document with the name separated from the value by = and the pairs separated from each other by &. Fields with null values may be omitted. In particular, unselected radio buttons and checkboxes should not appear in the encoded data, but hidden fields with VALUE attributes present should.

    NOTE – The URI from a query form submission can be used in a normal anchor style hyperlink. Unfortunately, the use of the & character to separate form fields interacts with its use in SGML attribute values as an entity reference delimiter. For example, the URI http://host/?x=1&y=2 must be written <a href="http://host/?x=1&y=2" or <a href="http://host/?x=1&y=2">.

    HTTP server implementors, and in particular, CGI implementors are encouraged to support the use of ; in place of & to save users the trouble of escaping & characters this way.

8.2.2 Query Forms: METHOD=GET

If the processing of a form is idempotent (i.e. it has no lasting observable effect on the state of the world), then the form method should be GET. Many database searches have no visible side-effects and make ideal applications of query forms.

To process a form whose action URL is an HTTP URL and whose method is `GET`, the user agent starts with the action URI and appends a `?` and the form data set, in `application/x-www-form-urlencoded` format as above. The user agent then traverses the link to this URI just as if it were an anchor (see Section 7.2).

> **NOTE –** The URL encoding may result in very long URIs, which cause some historical HTTP server implementations to exhibit defective behavior. As a result, some HTML forms are written using `METHOD=POST` even though the form submission has no side-effects.

8.2.3 Forms with Side-Effects: METHOD=POST

If the service associated with the processing of a form has side effects (for example, modification of a database or subscription to a service), the method should be `POST`.

To process a form whose action URL is an HTTP URL and whose method is `POST`, the user agent conducts an HTTP POST transaction using the action URI, and a message body of type `application/x-www-form-urlencoded` format as above. The user agent should display the response from the HTTP POST interaction just as it would display the response from an HTTP GET above.

8.2.4 Example Form Submission: Questionnaire Form

Consider the following document:

```
<!DOCTYPE HTML PUBLIC "-//IETF//DTD HTML 2.0//EN">
<title>Sample of HTML Form Submission</title>
<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>
```

The initial state of the form data set is:

name
: “”

gender
: “male”

family
: “”

other
: “”

nickname
: “”

Note that the radio input has an initial value, while the checkbox has none.

The user might edit the fields and request that the form be submitted. At that point, suppose the values are:

name
: “John Doe”

gender
: “male”

family
: “5”

city
: “kent”

city
: “miami”

other
: “abc\ndef”

nickname
: “J&D”

The user agent then conducts an HTTP POST transaction using the URI `http://www.w3.org/sample`. The message body would be (ignore the line break):

```
name=John+Doe&gender=male&family=5&city=kent&city=miami&
other=abc%0D%0Adef&nickname=J%26D
```

9 HTML Public Text

9.1 HTML DTD

This is the Document Type Definition for the HyperText Markup Language, level 2.

```
<!--    html.dtd

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

$Id: html.dtd,v 1.30 1995/09/21 23:30:19 connolly Exp $

Author: Daniel W. Connolly <connolly@w3.org>
See Also: html.decl, html-1.dtd
  http://www.w3.org/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 and 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
        -->

<!--========= 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 emphasis                         -->
<!-- <CODE>     Source code phrase                      -->
<!-- <SAMP>     Sample text or characters               -->
<!-- <KBD>      Keyboard phrase, e.g. user input        -->
<!-- <VAR>      Variable phrase or substitutable        -->
<!-- <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 ======================-->

<!ENTITY % linkType "NAMES">

<!ENTITY % linkExtraAttributes
        "REL %linkType #IMPLIED
        REV %linkType #IMPLIED
        URN CDATA #IMPLIED
```

```
        TITLE CDATA #IMPLIED
        METHODS NAMES #IMPLIED
        ">

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

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

<!ELEMENT A     - - %A.content -(A)>
<!ATTLIST A
        HREF CDATA #IMPLIED
        NAME CDATA #IMPLIED
        %linkExtraAttributes;
        %SDAPREF; "<Anchor: #AttList>"
        >
<!-- <A>Anchor; source/destination of link -->
<!-- <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  -->
<!-- <A TITLE="...">Title of destination (advisory)   -->
<!-- <A METHODS="...">Operations on destination (advisory) -->


<!--========== Images ==========================-->

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

<!-- <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                -->

<!--=========== Paragraphs=======================-->

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

<!-- <P>        Paragraph       -->


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

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

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

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

<!-- <H1>       Heading, level 1 -->
<!-- <H2>       Heading, level 2 -->
<!-- <H3>       Heading, level 3 -->
<!-- <H4>       Heading, level 4 -->
<!-- <H5>       Heading, level 5 -->
```

```
<!-- <H6>       Heading, level 6 -->


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

<![ %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 [

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

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

<!ENTITY % InputType "(TEXT | PASSWORD | CHECKBOX |
                        RADIO | SUBMIT | RESET |
                        IMAGE | HIDDEN )">
<!ELEMENT INPUT - O EMPTY>
<!ATTLIST INPUT
```

```
TYPE %InputType TEXT
NAME CDATA #IMPLIED
VALUE CDATA #IMPLIED
SRC CDATA #IMPLIED
CHECKED (CHECKED) #IMPLIED
SIZE CDATA #IMPLIED
MAXLENGTH NUMBER #IMPLIED
ALIGN (top|middle|bottom) #IMPLIED
        %SDAPREF; "Input: "
>

<!-- <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 -->
<!-- <INPUT CHECKED>Initial state is "on" -->
<!-- <INPUT SIZE=...>Field size hint -->
<!-- <INPUT MAXLENGTH=...>Data length maximum -->
<!-- <INPUT ALIGN=...>Image alignment -->

<!ELEMENT SELECT - - (OPTION+) -(INPUT|SELECT|TEXTAREA)>
<!ATTLIST SELECT
        NAME CDATA #REQUIRED
        SIZE NUMBER #IMPLIED
        MULTIPLE (MULTIPLE) #IMPLIED
        %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 "">
]]>
<!ENTITY % head.extra "& NEXTID?">

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

<!ELEMENT HEAD O O  (%head.content) +(META|LINK)>

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

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

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

<!ELEMENT LINK - O EMPTY>
<!ATTLIST LINK
        HREF CDATA #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 CDATA #REQUIRED     >

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

<!ELEMENT NEXTID - O EMPTY>
<!ATTLIST NEXTID
        N CDATA #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 Meta-information        -->
<!-- <META HTTP-EQUIV=...>      HTTP response header name       -->
<!-- <META NAME=...>       Meta-information 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 -->
```

9.2 Strict HTML DTD

This document type declaration refers to the HTML DTD with the `HTML.Recommended` entity defined as `INCLUDE` rather than IGNORE; that is, it refers to the more structurally rigid definition of HTML.

```
<!--html-s.dtd

        Document Type Definition for the HyperText Markup Language
with strict validation (HTML Strict DTD).

$Id: html-s.dtd,v 1.3 1995/06/02 18:55:46 connolly Exp $

Author: Daniel W. Connolly <connolly@w3.org>
See Also: http://www.w3.org/hypertext/WWW/MarkUp/MarkUp.html
-->

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

        -- Typical usage:

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

<!-- Feature Test Entities -->
<!ENTITY % HTML.Recommended "INCLUDE">
```

```
<!ENTITY % html PUBLIC "-//IETF//DTD HTML 2.0//EN">
%html;
```

9.3 Level 1 HTML DTD

This document type declaration refers to the HTML DTD with the `HTML.Forms` entity defined as `IGNORE` rather than `INCLUDE`. Documents which contain `<FORM>` elements do not conform to this DTD, and must use the level 2 DTD.

```
<!--html-1.dtd

        Document Type Definition for the HyperText Markup Language
with Level 1 Extensions (HTML Level 1 DTD).

$Id: html-1.dtd,v 1.2 1995/03/29 18:53:10 connolly Exp $

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

<!ENTITY % HTML.Version
"-//IETF//DTD HTML 2.0 Level 1//EN"

        -- Typical usage:

            <!DOCTYPE HTML PUBLIC
"-//IETF//DTD HTML Level 1//EN">
    <html>
    ...
    </html>
--
>

<!-- Feature Test Entities -->
<!ENTITY % HTML.Forms "IGNORE">

<!ENTITY % html PUBLIC "-//IETF//DTD HTML 2.0//EN">
%html;
```

9.4 Strict Level 1 HTML DTD

This document type declaration refers to the level 1 HTML DTD with the `HTML.Recommended` entity defined as `INCLUDE` rather than IGNORE; that is, it refers to the more structurally rigid definition of HTML.

```
<!--html-1s.dtd

        Document Type Definition for the HyperText Markup Language
Struct Level 1

$Id: html-1s.dtd,v 1.3 1995/06/02 18:55:43 connolly Exp $

Author: Daniel W. Connolly <connolly@w3.org>
See Also: http://www.w3.org/hypertext/WWW/MarkUp/MarkUp.html
-->

<!ENTITY % HTML.Version
"-//IETF//DTD HTML 2.0 Strict Level 1//EN"

        -- Typical usage:

            <!DOCTYPE HTML PUBLIC
"-//IETF//DTD HTML Strict Level 1//EN">
    <html>
    ...
    </html>
--
>

<!-- Feature Test Entities -->
<!ENTITY % HTML.Recommended "INCLUDE">

<!ENTITY % html-1 PUBLIC "-//IETF//DTD HTML 2.0 Level 1//EN">
%html-1;
```

9.5 SGML Declaration for HTML

This is the SGML Declaration for HyperText Markup Language.

```
<!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

SCOPE    DOCUMENT
SYNTAX
         SHUNCHAR CONTROLS 0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16
 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 127
         BASESET  "ISO 646:1983//CHARSET
                   International Reference Version
                   (IRV)//ESC 2/5 4/0"
         DESCSET  0 128 0
         FUNCTION
  RE          13
                  RS          10
                  SPACE       32
                  TAB SEPCHAR  9


         NAMING   LCNMSTRT ""
                  UCNMSTRT ""
                  LCNMCHAR ".-"
                  UCNMCHAR ".-"
```

```
                  NAMECASE GENERAL YES
                           ENTITY  NO
         DELIM    GENERAL  SGMLREF
                  SHORTREF SGMLREF
         NAMES    SGMLREF
         QUANTITY SGMLREF
                  ATTSPLEN 2100
                  LITLEN   1024
                  NAMELEN  72    -- somewhat arbitrary; taken from
                                internet line length conventions --
                  PILEN    1024
                  TAGLVL   100
                  TAGLEN   2100
                  GRPGTCNT 150
                  GRPCNT   64

    FEATURES
      MINIMIZE
        DATATAG  NO
        OMITTAG  YES
        RANK     NO
        SHORTTAG YES
      LINK
        SIMPLE   NO
        IMPLICIT NO
        EXPLICIT NO
      OTHER
        CONCUR   NO
        SUBDOC   NO
        FORMAL   YES
      APPINFO    "SDA"  -- conforming SGML Document Access application
        --
    >
    <!--
    $Id: html.decl,v 1.17 1995/06/08 14:59:32 connolly Exp $

    Author: Daniel W. Connolly <connolly@w3.org>

    See also: http://www.w3.org/hypertext/WWW/MarkUp/MarkUp.html
     -->
```

9.6 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 identifier. For example, the following are possible catalog entries:

```
-- catalog: SGML Open style entity catalog for HTML --
-- $Id: catalog,v 1.3 1995/09/21 23:30:23 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
 Strict Level 2: most general to most specific --
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

-- 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

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

9.7 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.

9.7.1 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      &lt;    Less than sign
>       gt      &gt;    Greater than sign
&       amp     &amp;   Ampersand
"       quot    &quot;  Double quote sign
```

9.7.2 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.
-->
<!-- Character entity set. Typical invocation:
     <!ENTITY % ISOlat1 PUBLIC
       "ISO 8879-1986//ENTITIES Added Latin 1//EN//HTML">
     %ISOlat1;
-->
<!--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 -->
```

```
<!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 -->
<!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 -->
<!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 -->
<!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 -->
<!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 -->
<!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 -->
```

10 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 [URL] 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 (see Section 8.1.2) — should be aware and make their users aware of the lack of confidentiality.

11 References

[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:ftp://ds.internic.net/rfc/rfc1630.txt>

[URL]
T. Berners-Lee, L. Masinter, and M. McCahill. “Uniform Resource Locators (URL).” RFC 1738, CERN, Xerox PARC, University of Minnesota, December 1994. <URL:ftp://ds.internic.net/rfc/rfc1738.txt>

[HTTP]
T. Berners-Lee, R. T. Fielding, and H. Frystyk Nielsen. “Hypertext Transfer Protocol - HTTP/1.0.” Work in Progress, MIT, UC Irvine, CERN, March 1995.

<URL:ftp://ds.internic.net/internet-drafts/draft-ietf-http-v10-spec-00.ps>

[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. <URL:ftp://ds.internic.net/rfc/rfc1521.txt>

[RELURL]
R. Fielding. "Relative Uniform Resource Locators." RFC 1808, June 1995 <URL:ftp://ds.internic.net/rfc/rfc1808.txt>

[GOLD90]
C. F. Goldfarb. "The SGML Handbook." Y. Rubinsky, Ed., Oxford University Press, 1990. <URL:>

[DEXTER]
Frank Halasz and Mayer Schwartz, "The Dexter Hypertext Reference Model", "Communications of the ACM", pp. 30-39, vol. 37 no. 2, Feb 1994, <URL:>

[IMEDIA]
J. Postel. "Media Type Registration Procedure.", USC/ISI, March 1994. <URL:ftp://ds.internic.net/rfc/rfc1590.txt>

[IANA]
J. Reynolds and J. Postel. "Assigned Numbers." STD 2, RFC 1700, USC/ISI, October 1994. <URL:ftp://ds.internic.net/rfc/rfc1700.txt>

[SQ91]
SoftQuad. "The SGML Primer." 3rd ed., SoftQuad Inc., 1991. <URL:http://www.sq.com/>

[ISO-646]
ISO/IEC 646:1991 Information technology — ISO 7-bit coded character set for information interchange <URL:http://www.iso.ch/cate/d4777.html>

[ISO-10646]
ISO/IEC 10646-1:1993 Information technology — Universal Multiple-Octet Coded Character Set (UCS) — Part 1: Architecture and Basic Multilingual Plane <URL:http://www.iso.ch/cate/d18741.html>

[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. <URL:http://www.iso.ch/cate/d16338.html>

[SGML]
ISO 8879. Information Processing — Text and Office Systems - Standard Generalized Markup Language (SGML), 1986. <URL:http://www.iso.ch/cate/d16387.html>

12 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 active participants in the HTML working group, too numerous to list individually, without whom there would be no standards process and no standard. That this document approaches its objective of carefully converging a description of current practice and formalization of HTML's relationship to SGML is a tribute to their effort.

12.1 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/
```

13 The HTML Coded Character Set

This list details the code positions and characters of the HTML document character set, specified in Section 9.5. This coded character set is based on [ISO-8859-1].

```
REFERENCE         DESCRIPTION
--------------    -----------
&#00; - &#08;     Unused
&#09;             Horizontal tab
&#10;             Line feed
&#11; - &#12;     Unused
&#13;             Carriage Return
&#14; - &#31;     Unused
&#32;             Space
&#33;             Exclamation mark
&#34;             Quotation mark
&#35;             Number sign
&#36;             Dollar sign
&#37;             Percent sign
&#38;             Ampersand
&#39;             Apostrophe
&#40;             Left parenthesis
&#41;             Right parenthesis
&#42;             Asterisk
&#43;             Plus sign
&#44;             Comma
&#45;             Hyphen
&#46;             Period (fullstop)
&#47;             Solidus (slash)
&#48; - &#57;     Digits 0-9
&#58;             Colon
&#59;             Semi-colon
&#60;             Less than
&#61;             Equals sign
&#62;             Greater than
&#63;             Question mark
&#64;             Commercial at
&#65; - &#90;     Letters A-Z
&#91;             Left square bracket
&#92;             Reverse solidus (backslash)
&#93;             Right square bracket
&#94;             Caret
&#95;             Horizontal bar (underscore)
&#96;             Acute accent
&#97; - &#122;    Letters a-z
```

{	Left curly brace
		Vertical bar
}	Right curly brace
~	Tilde
&#127; - Ÿ	Unused
	Non-breaking Space
¡	Inverted exclamation
¢	Cent sign
£	Pound sterling
¤	General currency sign
¥	Yen sign
¦	Broken vertical bar
§	Section sign
¨	Umlaut (dieresis)
©	Copyright
ª	Feminine ordinal
«	Left angle quote, guillemotleft
¬	Not sign
	Soft hyphen
®	Registered trademark
¯	Macron accent
°	Degree sign
±	Plus or minus
²	Superscript two
³	Superscript three
´	Acute accent
µ	Micro sign
¶	Paragraph sign
·	Middle dot
¸	Cedilla
¹	Superscript one
º	Masculine ordinal
»	Right angle quote, guillemotright
¼	Fraction one-fourth
½	Fraction one-half
¾	Fraction three-fourths
¿	Inverted question mark
À	Capital A, grave accent
Á	Capital A, acute accent
Â	Capital A, circumflex accent
Ã	Capital A, tilde
Ä	Capital A, dieresis or umlaut mark
Å	Capital A, ring
Æ	Capital AE dipthong (ligature)
Ç	Capital C, cedilla
È	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 Proposed Entities

The HTML DTD references the “Added Latin 1” entity set, which only supplies named entities for a subset of the non-ASCII characters in [ISO-8859-1], namely the accented characters. The following entities should be supported so that all ISO 8859-1 characters may only be referenced symbolically. The names for these entities are taken from the appendixes of [SGML].

```
<!ENTITY nbsp   CDATA " " -- no-break space -->
<!ENTITY iexcl  CDATA "&#161;" -- inverted exclamation mark -->
<!ENTITY cent   CDATA "&#162;" -- cent sign -->
<!ENTITY pound  CDATA "&#163;" -- pound sterling sign -->
<!ENTITY curren CDATA "&#164;" -- general currency sign -->
<!ENTITY yen    CDATA "&#165;" -- yen sign -->
<!ENTITY brvbar CDATA "&#166;" -- broken (vertical) bar -->
<!ENTITY sect   CDATA "&#167;" -- section sign -->
<!ENTITY uml    CDATA "&#168;" -- umlaut (dieresis) -->
<!ENTITY copy   CDATA "&#169;" -- copyright sign -->
<!ENTITY ordf   CDATA "&#170;" -- ordinal indicator, feminine -->
<!ENTITY laquo  CDATA "&#171;" -- angle quotation mark, left -->
<!ENTITY not    CDATA "&#172;" -- not sign -->
<!ENTITY shy    CDATA "&#173;" -- soft hyphen -->
<!ENTITY reg    CDATA "&#174;" -- registered sign -->
<!ENTITY macr   CDATA "&#175;" -- macron -->
<!ENTITY deg    CDATA "&#176;" -- degree sign -->
<!ENTITY plusmn CDATA "&#177;" -- plus-or-minus sign -->
<!ENTITY sup2   CDATA "&#178;" -- superscript two -->
<!ENTITY sup3   CDATA "&#179;" -- superscript three -->
<!ENTITY acute  CDATA "&#180;" -- acute accent -->
<!ENTITY micro  CDATA "&#181;" -- micro sign -->
<!ENTITY para   CDATA "&#182;" -- pilcrow (paragraph sign) -->
<!ENTITY middot CDATA "&#183;" -- middle dot -->
<!ENTITY cedil  CDATA "&#184;" -- cedilla -->
<!ENTITY sup1   CDATA "&#185;" -- superscript one -->
```

```
<!ENTITY ordm   CDATA "&#186;" -- ordinal indicator, masculine -->
<!ENTITY raquo  CDATA "&#187;" -- angle quotation mark, right -->
<!ENTITY frac14 CDATA "&#188;" -- fraction one-quarter -->
<!ENTITY frac12 CDATA "&#189;" -- fraction one-half -->
<!ENTITY frac34 CDATA "&#190;" -- fraction three-quarters -->
<!ENTITY iquest CDATA "&#191;" -- inverted question mark -->
<!ENTITY Agrave CDATA "&#192;" -- capital A, grave accent -->
<!ENTITY Aacute CDATA "&#193;" -- capital A, acute accent -->
<!ENTITY Acirc  CDATA "&#194;" -- capital A, circumflex accent -->
<!ENTITY Atilde CDATA "&#195;" -- capital A, tilde -->
<!ENTITY Auml   CDATA "&#196;" -- capital A, dieresis or umlaut mark -->
<!ENTITY Aring  CDATA "&#197;" -- capital A, ring -->
<!ENTITY AElig  CDATA "&#198;" -- capital AE diphthong (ligature) -->
<!ENTITY Ccedil CDATA "&#199;" -- capital C, cedilla -->
<!ENTITY Egrave CDATA "&#200;" -- capital E, grave accent -->
<!ENTITY Eacute CDATA "&#201;" -- capital E, acute accent -->
<!ENTITY Ecirc  CDATA "&#202;" -- capital E, circumflex accent -->
<!ENTITY Euml   CDATA "&#203;" -- capital E, dieresis or umlaut mark -->
<!ENTITY Igrave CDATA "&#204;" -- capital I, grave accent -->
<!ENTITY Iacute CDATA "&#205;" -- capital I, acute accent -->
<!ENTITY Icirc  CDATA "&#206;" -- capital I, circumflex accent -->
<!ENTITY Iuml   CDATA "&#207;" -- capital I, dieresis or umlaut mark -->
<!ENTITY ETH    CDATA "&#208;" -- capital Eth, Icelandic -->
<!ENTITY Ntilde CDATA "&#209;" -- capital N, tilde -->
<!ENTITY Ograve CDATA "&#210;" -- capital O, grave accent -->
<!ENTITY Oacute CDATA "&#211;" -- capital O, acute accent -->
<!ENTITY Ocirc  CDATA "&#212;" -- capital O, circumflex accent -->
<!ENTITY Otilde CDATA "&#213;" -- capital O, tilde -->
<!ENTITY Ouml   CDATA "&#214;" -- capital O, dieresis or umlaut mark -->
<!ENTITY times  CDATA "&#215;" -- multiply sign -->
<!ENTITY Oslash CDATA "&#216;" -- capital O, slash -->
<!ENTITY Ugrave CDATA "&#217;" -- capital U, grave accent -->
<!ENTITY Uacute CDATA "&#218;" -- capital U, acute accent -->
<!ENTITY Ucirc  CDATA "&#219;" -- capital U, circumflex accent -->
<!ENTITY Uuml   CDATA "&#220;" -- capital U, dieresis or umlaut mark -->
<!ENTITY Yacute CDATA "&#221;" -- capital Y, acute accent -->
<!ENTITY THORN  CDATA "&#222;" -- capital THORN, Icelandic -->
<!ENTITY szlig  CDATA "&#223;" -- small sharp s, German (sz ligature) -->
<!ENTITY agrave CDATA "&#224;" -- small a, grave accent -->
<!ENTITY aacute CDATA "&#225;" -- small a, acute accent -->
<!ENTITY acirc  CDATA "&#226;" -- small a, circumflex accent -->
<!ENTITY atilde CDATA "&#227;" -- small a, tilde -->
<!ENTITY auml   CDATA "&#228;" -- small a, dieresis or umlaut mark -->
<!ENTITY aring  CDATA "&#229;" -- small a, ring -->
<!ENTITY aelig  CDATA "&#230;" -- small ae diphthong (ligature) -->
<!ENTITY ccedil CDATA "&#231;" -- small c, cedilla -->
```

```
<!ENTITY egrave CDATA "&#232;" -- small e, grave accent -->
<!ENTITY eacute CDATA "&#233;" -- small e, acute accent -->
<!ENTITY ecirc  CDATA "&#234;" -- small e, circumflex accent -->
<!ENTITY euml   CDATA "&#235;" -- small e, dieresis or umlaut mark -->
<!ENTITY igrave CDATA "&#236;" -- small i, grave accent -->
<!ENTITY iacute CDATA "&#237;" -- small i, acute accent -->
<!ENTITY icirc  CDATA "&#238;" -- small i, circumflex accent -->
<!ENTITY iuml   CDATA "&#239;" -- small i, dieresis or umlaut mark -->
<!ENTITY eth    CDATA "&#240;" -- small eth, Icelandic -->
<!ENTITY ntilde CDATA "&#241;" -- small n, tilde -->
<!ENTITY ograve CDATA "&#242;" -- small o, grave accent -->
<!ENTITY oacute CDATA "&#243;" -- small o, acute accent -->
<!ENTITY ocirc  CDATA "&#244;" -- small o, circumflex accent -->
<!ENTITY otilde CDATA "&#245;" -- small o, tilde -->
<!ENTITY ouml   CDATA "&#246;" -- small o, dieresis or umlaut mark -->
<!ENTITY divide CDATA "&#247;" -- divide sign -->
<!ENTITY oslash CDATA "&#248;" -- small o, slash -->
<!ENTITY ugrave CDATA "&#249;" -- small u, grave accent -->
<!ENTITY uacute CDATA "&#250;" -- small u, acute accent -->
<!ENTITY ucirc  CDATA "&#251;" -- small u, circumflex accent -->
<!ENTITY uuml   CDATA "&#252;" -- small u, dieresis or umlaut mark -->
<!ENTITY yacute CDATA "&#253;" -- small y, acute accent -->
<!ENTITY thorn  CDATA "&#254;" -- small thorn, Icelandic -->
<!ENTITY yuml   CDATA "&#255;" -- small y, dieresis or umlaut mark -->
```