XHTML™ 1.0: The Extensible HyperText Markup Language

A Reformulation of HTML 4.0 in XML 1.0

W3C Proposed Recommendation 10 December 1999

This version:
: http://www.w3.org/TR/1999/PR-xhtml1-19991210
: (Postscript version, PDF version, ZIP archive, or Gzip'd TAR archive)

Latest version:
: http://www.w3.org/TR/xhtml1

Previous versions:
: http://www.w3.org/TR/1999/WD-xhtml1-19991124
: http://www.w3.org/TR/1999/PR-xhtml1-19990824

Authors:
: See acknowledgements [p.29] .

Abstract

This specification defines XHTML 1.0, a reformulation of HTML 4.0 as an XML 1.0 application, and three DTDs corresponding to the ones defined by HTML 4.0. The semantics of the elements and their attributes are defined in the W3C Recommendation for HTML 4.0. These semantics provide the foundation for future extensibility of XHTML. Compatibility with existing HTML user agents is possible by following a small set of guidelines.

Status of this document

This section describes the status of this document at the time of its publication. Other documents may supersede this document. The latest status of this document series is maintained at the W3C.

This specification is a Proposed Recommendation of the HTML Working Group. It is a revision of the Proposed Recommendation dated 24 August 1999 incorporating changes as a result of comments from the Proposed Recommendation review, and comments and further deliberations of the W3C HTML Working Group. A diff-marked version from the previous proposed recommendation is available for comparison purposes.

On 10 December 1999, this document enters a Proposed Recommendation review period. From that date until 8 January 2000, W3C Advisory Committee representatives are encouraged to review this specification and return comments in their completed ballots to w3c-html-review@w3.org. Please send any comments of a confidential nature in separate email to w3t-html@w3.org, which is visible to the Team only.

No sooner than 14 days after the end of the review period, the Director will announce the document's disposition: it may become a W3C Recommendation (possibly with minor changes), it may revert to Working Draft status, or it may be dropped as a W3C work item.

Publication as a Proposed Recommendation does not imply endorsement by the W3C membership. This is still a draft document and may be updated, replaced or obsoleted by other documents at any time. It is inappropriate to cite W3C Proposed Recommendation as other than "work in progress."

This document has been produced as part of the W3C HTML Activity. The goals of the HTML Working Group *(members only)* are discussed in the HTML Working Group charter *(members only)*.

A list of current W3C Recommendations and other technical documents can be found at http://www.w3.org/TR.

Public discussion on HTML features takes place on the mailing list www-html@w3.org (archive). The W3C staff contact for work on HTML is Dave Raggett.

Please report errors in this document to www-html-editor@w3.org.

The list of known errors in this specification is available at http://www.w3.org/1999/12/PR-xhtml1-19991210-errata.

Contents

1. What is XHTML?

XHTML is a family of current and future document types and modules that reproduce, subset, and extend HTML 4.0 [HTML] [p.31] . XHTML family document types are XML based, and ultimately are designed to work in conjunction with XML-based user agents. The details of this family and its evolution are discussed in more detail in the section on Future Directions [p.19] .

XHTML 1.0 (this specification) is the first document type in the XHTML family. It is a reformulation of the three HTML 4.0 document types as applications of XML 1.0 [XML] [p.31] . It is intended to be used as a language for content that is both XML-conforming and, if some simple guidelines [p.25] are followed, operates in HTML 4.0 conforming user agents. Developers who migrate their content to XHTML 1.0 will realize the following benefits:

- XHTML documents are XML conforming. As such, they are readily viewed, edited, and validated with standard XML tools.
- XHTML documents can be written to to operate as well or better than they did before in existing HTML 4.0-conforming user agents as well as in new, XHTML 1.0 conforming user agents.
- XHTML documents can utilize applications (e.g. scripts and applets) that rely upon either the HTML Document Object Model or the XML Document Object Model [DOM] [p.31] .
- As the XHTML family evolves, documents conforming to XHTML 1.0 will be more likely to interoperate within and among various XHTML environments.

The XHTML family is the next step in the evolution of the Internet. By migrating to XHTML today, content developers can enter the XML world with all of its attendant benefits, while still remaining confident in their content's backward and future compatibility.

1.1 What is HTML 4.0?

HTML 4.0 [HTML] [p.31] is an SGML (Standard Generalized Markup Language) application conforming to International Standard ISO 8879, and is widely regarded as the standard publishing language of the World Wide Web.

SGML is a language for describing markup languages, particularly those used in electronic document exchange, document management, and document publishing. HTML is an example of a language defined in SGML.

SGML has been around since the middle 1980's and has remained quite stable. Much of this stability stems from the fact that the language is both feature-rich and flexible. This flexibility, however, comes at a price, and that price is a level of complexity that has inhibited its adoption in a diversity of environments, including the World Wide Web.

HTML, as originally conceived, was to be a language for the exchange of scientific and other technical documents, suitable for use by non-document specialists. HTML addressed the problem of SGML complexity by specifying a small set of structural and semantic tags suitable for authoring relatively simple documents. In addition to simplifying the document structure, HTML added support for hypertext. Multimedia capabilities were added later.

In a remarkably short space of time, HTML became wildly popular and rapidly outgrew its original purpose. Since HTML's inception, there has been rapid invention of new elements for use within HTML (as a standard) and for adapting HTML to vertical, highly specialized, markets. This plethora of new elements has led to compatibility problems for documents across different platforms.

As the heterogeneity of both software and platforms rapidly proliferate, it is clear that the suitability of 'classic' HTML 4.0 for use on these platforms is somewhat limited.

1.2 What is XML?

XML™ is the shorthand for Extensible Markup Language, and is an acronym of Extensible Markup Language [XML] [p.31] .

XML was conceived as a means of regaining the power and flexibility of SGML without most of its complexity. Although a restricted form of SGML, XML nonetheless preserves most of SGML's power and richness, and yet still retains all of SGML's commonly used features.

While retaining these beneficial features, XML removes many of the more complex features of SGML that make the authoring and design of suitable software both difficult and costly.

1.3 Why the need for XHTML?

The benefits of migrating to XHTML 1.0 are described above. Some of the benefits of migrating to XHTML in general are:

- Document developers and user agent designers are constantly discovering new ways to express their ideas through new markup. In XML, it is relatively easy to introduce new elements or additional element attributes. The XHTML family is designed to accommodate these extensions through XHTML modules and techniques for developing new XHTML-conforming modules (described in the forthcoming XHTML Modularization specification). These modules will permit the combination of existing and new feature sets when developing content and when designing new user agents.
- Alternate ways of accessing the Internet are constantly being introduced. Some estimates indicate that by the year 2002, 75% of Internet document viewing will be carried out on these alternate platforms. The XHTML family is designed with general user agent interoperability in mind. Through a new user agent and document profiling mechanism, servers, proxies, and user agents will be able to perform best effort content transformation. Ultimately, it will be possible to develop XHTML-conforming content that is usable by any XHTML-conforming user agent.

2. Definitions

2.1 Terminology

The following terms are used in this specification. These terms extend the definitions in [RFC2119] [p.31] in ways based upon similar definitions in ISO/IEC 9945-1:1990 [POSIX.1] [p.31] :

Implementation-defined
: A value or behavior is implementation-defined when it is left to the implementation to define [and document] the corresponding requirements for correct document construction.

May
: With respect to implementations, the word "may" is to be interpreted as an optional feature that is not required in this specification but can be provided. With respect to Document Conformance [p.9] , the word "may" means that the optional feature must not be used. The term "optional" has the same definition as "may".

Must
: In this specification, the word "must" is to be interpreted as a mandatory requirement on the implementation or on Strictly Conforming XHTML Documents, depending upon the context. The term "shall" has the same definition as "must".

Reserved
: A value or behavior is unspecified, but it is not allowed to be used by Conforming Documents nor to be supported by a Conforming User Agents.

Should
: With respect to implementations, the word "should" is to be interpreted as an implementation recommendation, but not a requirement. With respect to documents, the word "should" is to be interpreted as recommended programming practice for documents and a requirement for Strictly Conforming XHTML Documents.

Supported
: Certain facilities in this specification are optional. If a facility is supported, it behaves as specified by this specification.

Unspecified
: When a value or behavior is unspecified, the specification defines no portability requirements for a facility on an implementation even when faced with a document that uses the facility. A document that requires specific behavior in such an instance, rather than tolerating any behavior when using that facility, is not a Strictly Conforming XHTML Document.

2.2 General Terms

Attribute
: An attribute is a parameter to an element declared in the DTD. An attribute’s type and value range, including a possible default value, are defined in the DTD.

DTD
: A DTD, or document type definition, is a collection of XML declarations that, as a collection, defines the legal structure, elements, and attributes that are available for use in a document that complies to the DTD.

Document
: A document is a stream of data that, after being combined with any other streams it references, is structured such that it holds information contained within elements that are organized as defined in the associated DTD. See Document Conformance [p.9] for more information.

Element
: An element is a document structuring unit declared in the DTD. The element's content model is defined in the DTD, and additional semantics may be defined in the prose description of the element.

Facilities
: Functionality includes elements, attributes, and the semantics associated with those elements and attributes. An implementation supporting that functionality is said to provide the necessary facilities.

Implementation
: An implementation is a system that provides collection of facilities and services that supports this specification. See User Agent Conformance [p.11] for more information.

Parsing
: Parsing is the act whereby a document is scanned, and the information contained within the document is filtered into the context of the elements in which the information is structured.

Rendering
: Rendering is the act whereby the information in a document is presented. This presentation is done in the form most appropriate to the environment (e.g. aurally, visually, in print).

User Agent
: A user agent is an implementation that retrieves and processes XHTML documents. See User Agent Conformance [p.11] for more information.

Validation
: Validation is a process whereby documents are verified against the associated DTD, ensuring that the structure, use of elements, and use of attributes are consistent with the definitions in the DTD.

Well-formed
: A document is well-formed when it is structured according to the rules defined in Section 2.1 of the XML 1.0 Recommendation [XML] [p.31] . Basically, this definition states that elements, delimited by their start and end tags, are nested properly within one another.

3. Normative Definition of XHTML 1.0

3.1 Document Conformance

This version of XHTML provides a definition of strictly conforming XHTML documents, which are restricted to tags and attributes from the XHTML namespace. See Section 3.1.2 [p.10] for information on using XHTML with other namespaces, for instance, to include metadata expressed in RDF within XHTML documents.

3.1.1 Strictly Conforming Documents

A Strictly Conforming XHTML Document is a document that requires only the facilities described as mandatory in this specification. Such a document must meet all of the following criteria:

1. It must validate against one of the three DTDs found in Appendix A [p.21] .
2. The root element of the document must be `<html>`.
3. The root element of the document must designate the XHTML namespace using the `xmlns` attribute [XMLNAMES] [p.31] . The namespace for XHTML is defined to be `http://www.w3.org/1999/xhtml`.
4. There must be a DOCTYPE declaration in the document prior to the root element. The public identifier included in the DOCTYPE declaration must reference one of the three DTDs found in Appendix A [p.21] using the respective Formal Public Identifier. The system identifier may be changed to reflect local system conventions.

```
<!DOCTYPE html
     PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
     "http://www.w3.org/TR/1999/PR-xhtml1-19991210/DTD/xhtml1-strict.dtd>

<!DOCTYPE html
     PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
     "http://www.w3.org/TR/1999/PR-xhtml1-19991210/DTD/xhtml1-transitional.dtd>

<!DOCTYPE html
     PUBLIC "-//W3C//DTD XHTML 1.0 Frameset//EN"
     "http://www.w3.org/TR/1999/PR-xhtml1-19991210/DTD/xhtml1-frameset.dtd>
```

Here is an example of a minimal XHTML document.

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE html
     PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
    "http://www.w3.org/TR/1999/PR-xhtml1-19991210/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="en" lang="en">
  <head>
    <title>Virtual Library</title>
  </head>
```

```
  <body>
    <p>Moved to <a href="http://vlib.org/">vlib.org</a>.</p>
  </body>
</html>
```

Note that in this example, the XML declaration is included. An XML declaration like the one above is not required in all XML documents. XHTML document authors are strongly encouraged to use XML declarations in all their documents. Such a declaration is required when the character encoding of the document is other than the default UTF-8 or UTF-16.

3.1.2 Using XHTML with other namespaces

The XHTML namespace may be used with other XML namespaces as per [XMLNAMES] [p.31] , although such documents are not strictly conforming XHTML 1.0 documents as defined above. Future work by W3C will address ways to specify conformance for documents involving multiple namespaces.

The following example shows the way in which XHTML 1.0 could be used in conjunction with the MathML Recommendation:

```
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="en" lang="en">
  <head>
    <title>A Math Example</title>
  </head>
  <body>
    <p>The following is MathML markup:</p>
    <math xmlns="http://www.w3.org/1998/Math/MathML">
      <apply> <log/>
        <logbase>
          <cn> 3 </cn>
        </logbase>
        <ci> x </ci>
      </apply>
    </math>
  </body>
</html>
```

The following example shows the way in which XHTML 1.0 markup could be incorporated into another XML namespace:

```
<?xml version="1.0" encoding="UTF-8"?>
<!-- initially, the default namespace is "books" -->
<book xmlns='urn:loc.gov:books'
    xmlns:isbn='urn:ISBN:0-395-36341-6' xml:lang="en" lang="en">
  <title>Cheaper by the Dozen</title>
  <isbn:number>1568491379</isbn:number>
  <notes>
    <!-- make HTML the default namespace for a hypertext commentary -->
    <p xmlns='http://www.w3.org/1999/xhtml'>
        This is also available <a href="http://www.w3.org/">online</a>.
    </p>
  </notes>
</book>
```

3.2 User Agent Conformance

A conforming user agent must meet all of the following criteria:

1. In order to be consistent with the XML 1.0 Recommendation [XML] [p.31] , the user agent must parse and evaluate an XHTML document for well-formedness. If the user agent claims to be a validating user agent, it must also validate documents against their referenced DTDs according to [XML] [p.31] .
2. When the user agent claims to support facilities [p.8] defined within this specification or required by this specification through normative reference, it must do so in ways consistent with the facilities' definition.
3. When a user agent processes an XHTML document as generic XML, it shall only recognize attributes of type `ID` (e.g. the `id` attribute on most XHTML elements) as fragment identifiers.
4. If a user agent encounters an element it does not recognize, it must render the element's content.
5. If a user agent encounters an attribute it does not recognize, it must ignore the entire attribute specification (i.e., the attribute and its value).
6. If a user agent encounters an attribute value it doesn't recognize, it must use the default attribute value.
7. If it encounters an entity reference (other than one of the predefined entities) for which the User Agent has processed no declaration (which could happen if the declaration is in the external subset which the User Agent hasn't read), the entity reference should be rendered as the characters (starting with the ampersand and ending with the semi-colon) that make up the entity reference.
8. When rendering content, User Agents that encounter characters or character entity references that are recognized but not renderable should display the document in such a way that it is obvious to the user that normal rendering has not taken place.
9. The following characters are defined in [XML] as whitespace characters:
   - Space ( )
   - Tab (	)
   - Carriage return (
)
   - Line feed (
)

   The XML processor normalizes different system's line end codes into one single line-feed character, that is passed up to the application. The XHTML user agent in addition, must treat the following characters as whitespace:

   - Form feed ()
   - Zero-width space ()

   In elements where the 'xml:space' attribute is set to 'preserve', the user agent must leave all whitespace characters intact (with the exception of leading and trailing whitespace characters, which should be removed). Otherwise, whitespace is handled according to the following rules:

- All whitespace surrounding block elements should be removed.
- Comments are removed entirely and do not affect whitespace handling. One whitespace character on either side of a comment is treated as two white space characters.
- Leading and trailing whitespace inside a block element must be removed.
- Line feed characters within a block element must be converted into a space (except when the 'xml:space' attribute is set to 'preserve').
- A sequence of white space characters must be reduced to a single space character (except when the 'xml:space' attribute is set to 'preserve').
- With regard to rendition, the User Agent should render the content in a manner appropriate to the language in which the content is written. In languages whose primary script is Latinate, the ASCII space character is typically used to encode both grammatical word boundaries and typographic whitespace; in languages whose script is related to Nagari (e.g., Sanskrit, Thai, etc.), grammatical boundaries may be encoded using the ZW 'space' character, but will not typically be represented by typographic whitespace in rendered output; languages using Arabiform scripts may encode typographic whitespace using a space character, but may also use the ZW space character to delimit 'internal' grammatical boundaries (what look like words in Arabic to an English eye frequently encode several words, e.g. 'kitAbuhum' = 'kitAbu-hum' = 'book them' == their book); and languages in the Chinese script tradition typically neither encode such delimiters nor use typographic whitespace in this way.

Whitespace in attribute values is processed according to [XML] [p.31] .

4. Differences with HTML 4.0

Due to the fact that XHTML is an XML application, certain practices that were perfectly legal in SGML-based HTML 4.0 [HTML] [p.31] must be changed.

4.1 Documents must be well-formed

Well-formedness [p.8] is a new concept introduced by [XML] [p.31] . Essentially this means that all elements must either have closing tags or be written in a special form (as described below), and that all the elements must nest.

Although overlapping is illegal in SGML, it was widely tolerated in existing browsers.

CORRECT: nested elements.

<p>here is an emphasized paragraph.</p>

INCORRECT: overlapping elements

<p>here is an emphasized paragraph.</p>

4.2 Element and attribute names must be in lower case

XHTML documents must use lower case for all HTML element and attribute names. This difference is necessary because XML is case-sensitive e.g. and <LI> are different tags.

4.3 For non-empty elements, end tags are required

In SGML-based HTML 4.0 certain elements were permitted to omit the end tag; with the elements that followed implying closure. This omission is not permitted in XML-based XHTML. All elements other than those declared in the DTD as `EMPTY` must have an end tag.

CORRECT: terminated elements

<p>here is a paragraph.</p><p>here is another paragraph.</p>

INCORRECT: unterminated elements

<p>here is a paragraph.<p>here is another paragraph.

4.4 Attribute values must always be quoted

All attribute values must be quoted, even those which appear to be numeric.

CORRECT: quoted attribute values

<table rows="3">

INCORRECT: unquoted attribute values

<table rows=3>

4.5 Attribute Minimization

XML does not support attribute minimization. Attribute-value pairs must be written in full. Attribute names such as `compact` and `checked` cannot occur in elements without their value being specified.

CORRECT: unminimized attributes

<dl compact="compact">

INCORRECT: minimized attributes

<dl compact>

4.6 Empty Elements

Empty elements must either have an end tag or the start tag must end with `/>`. For instance, `<br/>` or `<hr></hr>`. See HTML Compatibility Guidelines [p.25] for information on ways to ensure this is backward compatible with HTML 4.0 user agents.

CORRECT: terminated empty tags

<hr/>

INCORRECT: unterminated empty tags

<hr>

4.7 Whitespace handling in attribute values

In attribute values, user agents will strip leading and trailing whitespace from attribute values and map sequences of one or more whitespace characters (including line breaks) to a single inter-word space (an ASCII space character for western scripts). See Section 3.3.3 of [XML] [p.31] .

4.8 Script and Style elements

In XHTML, the script and style elements are declared as having #PCDATA content. As a result, < and & will be treated as the start of markup, and entities such as < and & will be recognized as entity references by the XML processor to < and & respectively. Wrapping the content of the script or style element within a CDATA marked section avoids the expansion of these entities.

```
<script>
 <![CDATA[
 ... unescaped script content ...
 ]]>
 </script>
```

CDATA sections are recognized by the XML processor and appear as nodes in the Document Object Model, see Section 1.3 of the DOM Level 1 Recommendation [DOM] [p.31] .

An alternative is to use external script and style documents.

4.9 SGML exclusions

SGML gives the writer of a DTD the ability to exclude specific elements from being contained within an element. Such prohibitions (called "exclusions") are not possible in XML.

For example, the HTML 4.0 Strict DTD forbids the nesting of an 'a' element within another 'a' element to any descendant depth. It is not possible to spell out such prohibitions in XML. Even though these prohibitions cannot be defined in the DTD, certain elements should not be nested. A summary of such elements and the elements that should not be nested in them is found in the normative Appendix B [p.23] .

4.10 The elements with 'id' and 'name' attributes

HTML 4.0 defined the name attribute for the elements a, applet, frame, iframe, img, and map. HTML 4.0 also introduced the id attribute. Both of these attributes are designed to be used as fragment identifiers.

In XML, fragment identifiers are of type ID, and there can only be a single attribute of type ID per element. Therefore, in XHTML 1.0 the id attribute is defined to be of type ID. In order to ensure that XHTML 1.0 documents are well-structured XML documents, XHTML 1.0 documents MUST use the id attribute when defining fragment identifiers, even on elements that historically have also had a name attribute. See the HTML Compatibility Guidelines [p.25] for information on ensuring such anchors are backwards compatible when serving XHTML documents as media type text/html.

Note that in XHTML 1.0, the name attribute of these elements is formally deprecated, and will be removed in a subsequent version of XHTML.

5. Compatibility Issues

Although there is no requirement for XHTML 1.0 documents to be compatible with existing user agents, in practice this is easy to accomplish. Guidelines for creating compatible documents can be found in Appendix C [p.25] .

5.1 Internet Media Type

As of the publication of this recommendation, the general recommended MIME labeling for XML-based applications has yet to be resolved.

However, XHTML Documents which follow the guidelines set forth in Appendix C [p.25] , "HTML Compatibility Guidelines" may be labeled with the Internet Media Type "text/html", as they are compatible with most HTML browsers. This document makes no recommendation about MIME labeling of other XHTML documents.

6. Future Directions

XHTML 1.0 provides the basis for a family of document types that will extend and subset XHTML, in order to support a wide range of new devices and applications, by defining modules and specifying a mechanism for combining these modules. This mechanism will enable the extension and sub-setting of XHTML 1.0 in a uniform way through the definition of new modules.

6.1 Modularizing HTML

As the use of XHTML moves from the traditional desktop user agents to other platforms, it is clear that not all of the XHTML elements will be required on all platforms. For example a hand held device or a cell-phone may only support a subset of XHTML elements.

The process of modularization breaks XHTML up into a series of smaller element sets. These elements can then be recombined to meet the needs of different communities.

These modules will be defined in a later W3C document.

6.2 Subsets and Extensibility

Modularization brings with it several advantages:

- It provides a formal mechanism for sub-setting XHTML.
- It provides a formal mechanism for extending XHTML.
- It simplifies the transformation between document types.
- It promotes the reuse of modules in new document types.

6.3 Document Profiles

A document profile specifies the syntax and semantics of a set of documents. Conformance to a document profile provides a basis for interoperability guarantees. The document profile specifies the facilities required to process documents of that type, e.g. which image formats can be used, levels of scripting, style sheet support, and so on.

For product designers this enables various groups to define their own standard profile.

For authors this will obviate the need to write several different versions of documents for different clients.

For special groups such as chemists, medical doctors, or mathematicians this allows a special profile to be built using standard HTML elements plus a group of elements geared to the specialist’s needs.

Appendix A. DTDs

This appendix is normative.

These DTDs and entity sets form a normative part of this specification. The complete set of DTD files together with an XML declaration and SGML Open Catalog is included in the zip file for this specification.

A.1 Document Type Definitions

These DTDs approximate the HTML 4.0 DTDs. It is likely that when the DTDs are modularized, a method of DTD construction will be employed that corresponds more closely to HTML 4.0.

- XHTML-1.0-Strict
- XHTML-1.0-Transitional
- XHTML-1.0-Frameset

A.2 Entity Sets

The XHTML entity sets are the same as for HTML 4.0, but have been modified to be valid XML 1.0 entity declarations. Note the entity for the Euro currency sign (`€` or `€` or `€`) is defined as part of the special characters.

- Latin-1 characters
- Special characters
- Symbols

Appendix B. Element Prohibitions

This appendix is normative.

The following elements have prohibitions on which elements they can contain (see Section 4.9 [p.15]). This prohibition applies to all depths of nesting, i.e. it contains all the descendant elements.

`a`
: cannot contain other `a` elements.

`pre`
: cannot contain the `img`, `object`, `big`, `small`, `sub`, or `sup` elements.

`button`
: cannot contain the `input`, `select`, `textarea`, `label`, `button`, `form`, `fieldset`, `iframe` or `isindex` elements.

`label`
: cannot contain other `label` elements.

`form`
: cannot contain other `form` elements.

Appendix C. HTML Compatibility Guidelines

This appendix is informative.

This appendix summarizes design guidelines for authors who wish their XHTML documents to render on existing HTML user agents.

C.1 Processing Instructions

Be aware that processing instructions are rendered on some user agents. However, also note that when the XML declaration is not included in a document, the document can only use the default character encodings UTF-8 or UTF-16.

C.2 Empty Elements

Include a space before the trailing / and > of empty elements, e.g.
, <hr /> and . Also, use the minimized tag syntax for empty elements, e.g.
, as the alternative syntax
</br> allowed by XML gives uncertain results in many existing user agents.

C.3 Element Minimization and Empty Element Content

Given an empty instance of an element whose content model is not EMPTY (for example, an empty title or paragraph) do not use the minimized form (e.g. use <p> </p> and not <p />).

C.4 Embedded Style Sheets and Scripts

Use external style sheets if your style sheet uses < or & or]]> or --. Use external scripts if your script uses < or & or]]> or --. Note that XML parsers are permitted to silently remove the contents of comments. Therefore, the historical practice of "hiding" scripts and style sheets within comments to make the documents backward compatible is likely to not work as expected in XML-based implementations.

C.5 Line Breaks within Attribute Values

Avoid line breaks and multiple whitespace characters within attribute values. These are handled inconsistently by user agents.

C.6 Isindex

Don’t include more than one isindex element in the document head. The isindex element is deprecated in favor of the input element.

C.7 The `lang` and `xml:lang` Attributes

Use both the `lang` and `xml:lang` attributes when specifying the language of an element. The value of the `xml:lang` attribute takes precedence.

C.8 Fragment Identifiers

In XML, URIs [RFC2396 [p.31]] that end with fragment identifiers of the form `"#foo"` do not refer to elements with an attribute `name="foo"`; rather, they refer to elements with an attribute defined to be of type `ID`, e.g., the `id` attribute in HTML 4.0. Many existing HTML clients don't support the use of `ID`-type attributes in this way, so identical values may be supplied for both of these attributes to ensure maximum forward and backward compatibility (e.g., `<a id="foo" name="foo">...</a>`).

Further, since the set of legal values for attributes of type `ID` is much smaller than for those of type `CDATA`, the type of the `name` attribute has been changed to `NMTOKEN`. This attribute is constrained such that it can only have the same values as type `ID`, or as the `Name` production in XML 1.0 Section 2.5, production 5. Unfortunately, this constraint cannot be expressed in the XHTML 1.0 DTDs. Because of this change, care must be taken when converting existing HTML documents. The values of these attributes must be unique within the document, valid, and any references to these fragment identifiers (both internal and external) must be updated should the values be changed during conversion.

Finally, note that XHTML 1.0 has deprecated the `name` attribute of the `a`, `applet`, `frame`, `iframe`, `img`, and `map` elements, and it will be removed from XHTML in subsequent versions.

C.9 Character Encoding

To specify a character encoding in the document, use both the encoding attribute specification on the xml declaration (e.g. `<?xml version="1.0" encoding="EUC-JP"?>`) and a meta http-equiv statement (e.g. `<meta http-equiv="Content-type" content='text/html; charset="EUC-JP"' />`). The value of the encoding attribute of the xml processing instruction takes precedence.

C.10 Boolean Attributes

Some HTML user agents are unable to interpret boolean attributes when these appear in their full (non-minimized) form, as required by XML 1.0. Note this problem doesn't effect user agents compliant with HTML 4.0. The following attributes are involved: `compact`, `nowrap`, `ismap`, `declare`, `noshade`, `checked`, `disabled`, `readonly`, `multiple`, `selected`, `noresize`, `defer`.

C.11 Document Object Model and XHTML

The Document Object Model level 1 Recommendation [DOM [p.31]] defines document object model interfaces for XML and HTML 4.0. The HTML 4.0 document object model specifies that HTML element and attribute names are returned in upper-case. The XML document object model specifies that element and attribute names are returned in the case they are specified. In XHTML 1.0, elements and attributes are specified in lower-case. This apparent difference can be addressed in two ways:

1. Applications that access XHTML documents served as Internet media type `text/html` via the DOM can use the HTML DOM, and can rely upon element and attribute names being returned in upper-case from those interfaces.
2. Applications that access XHTML documents served as Internet media types `text/xml` or `application/xml` can also use the XML DOM. Elements and attributes will be returned in lower-case. Also, some XHTML elements may or may not appear in the object tree because they are optional in the content model (e.g. the `tbody` element within `table`). This occurs because in HTML 4.0 some elements were permitted to be minimized such that their start and end tags are both omitted (an SGML feature). This is not possible in XML. Rather than require document authors to insert extraneous elements, XHTML has made the elements optional. Applications need to adapt to this accordingly.

C.12 Using Ampersands in Attribute Values

When an attribute value contains an ampersand, it must be expressed as a character entity reference (e.g. "`&`"). For example, when the `href` attribute of the `a` element refers to a CGI script that takes parameters, it must be expressed as `http://my.site.dom/cgi-bin/myscript.pl?class=guest&name=user` rather than as `http://my.site.dom/cgi-bin/myscript.pl?class=guest&name=user`.

C.13 Cascading Style Sheets (CSS) and XHTML

The Cascading Style Sheets level 2 Recommendation [CSS2 [p.31]] defines style properties which are applied to the parse tree of the HTML or XML document. Differences in parsing will produce different visual or aural results, depending on the selectors used. The following hints will reduce this effect for documents which are served without modification as both media types:

1. CSS style sheets for XHTML should use lower case element and attribute names.
2. In tables, the tbody element will be inferred by the parser of an HTML user agent, but not by the parser of an XML user agent. Therefore you should always explicitely add a tbody element if it is referred to in a CSS selector.
3. Within the XHTML name space, user agents are expected to recognize the "id" attribute as an attribute of type ID. Therefore, style sheets should be able to continue using the shorthand "#" selector syntax even if the user agent does not read the DTD.
4. Within the XHTML name space, user agents are expected to recognize the "class" attribute. Therefore, style sheets should be able to continue using the shorthand "." selector syntax.

5. CSS defines different conformance rules for HTML and XML documents; be aware that the HTML rules apply to XHTML documents delivered as HTML and the XML rules apply to XHTML documents delivered as XML.

Appendix D. Acknowledgements

This appendix is informative.

This specification was written with the participation of the members of the W3C HTML working group:

Steven Pemberton, CWI (HTML Working Group Chair)
Murray Altheim, Sun Microsystems
Daniel Austin, CNET: The Computer Network
Frank Boumphrey, HTML Writers Guild
John Burger, Mitre
Andrew W. Donoho, IBM
Sam Dooley, IBM
Klaus Hofrichter, GMD
Philipp Hoschka, W3C
Masayasu Ishikawa, W3C
Warner ten Kate, Philips Electronics
Peter King, Phone.com
Paula Klante, JetForm
Shin'ichi Matsui, W3C/Panasonic
Shane McCarron, Applied Testing and Technology (The Open Group through August 1999)
Ann Navarro, HTML Writers Guild
Zach Nies, Quark
Dave Raggett, W3C/HP (W3C lead for HTML)
Patrick Schmitz, Microsoft
Sebastian Schnitzenbaumer, Stack Overflow
Chris Wilson, Microsoft
Ted Wugofski, Gateway 2000
Dan Zigmond, WebTV Networks

Appendix E. References

This appendix is informative.

[CSS2]
"Cascading Style Sheets, level 2 (CSS2) Specification", B. Bos, H. W. Lie, C. Lilley, I. Jacobs, 12 May 1998.
Available at: http://www.w3.org/TR/REC-CSS2

[DOM]
"Document Object Model (DOM) Level 1 Specification", Lauren Wood *et al.*, 1 October 1998.
Available at: http://www.w3.org/TR/REC-DOM-Level-1

[HTML]
"HTML 4.01 Specification", D. Raggett, A. Le Hors, I. Jacobs, 24 August 1999.
Available at: http://www.w3.org/TR/1999/PR-html40-19990824

[POSIX.1]
"ISO/IEC 9945-1:1990 Information Technology - Portable Operating System Interface (POSIX) - Part 1: System Application Program Interface (API) [C Language]", Institute of Electrical and Electronics Engineers, Inc, 1990.

[RFC2046]
"RFC2046: Multipurpose Internet Mail Extensions (MIME) Part Two: Media Types", N. Freed and N. Borenstein, November 1996.
Available at http://www.ietf.org/rfc/rfc2046.txt. Note that this RFC obsoletes RFC1521, RFC1522, and RFC1590.

[RFC2119]
"RFC2119: Key words for use in RFCs to Indicate Requirement Levels", S. Bradner, March 1997.
Available at: http://www.ietf.org/rfc/rfc2119.txt

[RFC2376]
"RFC2376: XML Media Types", E. Whitehead, M. Murata, July 1998.
Available at: http://www.ietf.org/rfc/rfc2376.txt

[RFC2396]
"RFC2396: Uniform Resource Identifiers (URI): Generic Syntax", T. Berners-Lee, R. Fielding, L. Masinter, August 1998.
This document updates RFC1738 and RFC1808.
Available at: http://www.ietf.org/rfc/rfc2396.txt

[XML]
"Extensible Markup Language (XML) 1.0 Specification", T. Bray, J. Paoli, C. M. Sperberg-McQueen, 10 February 1998.
Available at: http://www.w3.org/TR/REC-xml

[XMLNAMES]
"Namespaces in XML", T. Bray, D. Hollander, A. Layman, 14 January 1999.
XML namespaces provide a simple method for qualifying names used in XML documents by associating them with namespaces identified by URI.
Available at: http://www.w3.org/TR/REC-xml-names