XHTML™ 1.0 The Extensible HyperText Markup Language (Second Edition)

A Reformulation of HTML 4 in XML 1.0

W3C Recommendation 26 January 2000, revised 1 August 2002 superseded 27 March 2018

This version:
http://www.w3.org/TR/2018/SPSD-xhtml1-20180327/
Latest version:
http://www.w3.org/TR/xhtml1
Previous version:
http://www.w3.org/TR/2002/REC-xhtml1-20020801
Authors:
See acknowledgments.

Please refer to the **errata** for this document, which may include some normative corrections. See also **translations**.

This document is also available in these non-normative formats: Multi-part XHTML file, PostScript version, PDF version, ZIP archive, and Gzip'd TAR archive.

Abstract

This specification defines the Second Edition of XHTML 1.0, a reformulation of HTML 4 as an XML 1.0 application, and three DTDs corresponding to the ones defined by HTML 4. The semantics of the elements and their attributes are defined in the W3C Recommendation for HTML 4. 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This specification is a Superseded Recommendation. A newer specification exists that is recommended for new adoption in place of this specification. New implementations should follow the latest version of the

HTML specification.

This document is the second edition of the XHTML 1.0 specification incorporating the errata changes as of 1 August 2002. Changes between this version and the previous Recommendation are illustrated in a diff-marked version.

This second edition is *not* a new version of XHTML 1.0 (first published 26 January 2000). The changes in this document reflect corrections applied as a result of comments submitted by the community and as a result of ongoing work within the HTML Working Group. There are no substantive changes in this document - only the integration of various errata.

This document has been produced as part of the W3C HTML Activity.

At the time of publication, the working group believed there were zero patent disclosures relevant to this specification. A current list of patent disclosures relevant to this specification may be found on the Working Group's patent disclosure page.

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

Quick Table of Contents

Full Table of Contents

1. What is XHTML?

This section is informative.

XHTML is a family of current and future document types and modules that reproduce, subset, and extend HTML 4 [HTML4]. 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 [XHTMLMOD].

XHTML 1.0 (this specification) is the first document type in the XHTML family. It is a reformulation of the three HTML 4 document types as applications of XML 1.0 [XML]. It is intended to be used as a language for content that is both XML-conforming and, if some simple guidelines are followed, operates in HTML 4 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 operate as well or better than they did before in existing HTML 4-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].
- 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?

HTML 4 [HTML4] 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 interoperability problems for documents across different platforms.

1.2. What is XML?

XML™ is the shorthand name for Extensible Markup Language [XML].

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 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. 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

This section is normative.

2.1. Terminology

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

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, 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".

Optional
: See "May".

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

Shall
: See "Must".

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 markup 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 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
: Facilities are elements, attributes, and the semantics associated with those elements and attributes.

Implementation
: See User Agent.

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 a system that processes XHTML documents in accordance with this specification. See User Agent Conformance 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].

3. Normative Definition of XHTML 1.0

This section is normative.

3.1. Document Conformance

This version of XHTML provides a definition of strictly conforming XHTML 1.0 documents, which are restricted to elements and attributes from the XML and XHTML 1.0 namespaces. See Section 3.1.2 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 an XML 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 conform to the constraints expressed in one of the three DTDs found in DTDs and in Appendix B.

2. The root element of the document must be `html`.

3. The root element of the document must contain an `xmlns` declaration for the XHTML namespace [XMLNS]. The namespace for XHTML is defined to be `http://www.w3.org/1999/xhtml`. An example root element might look like:

   ```
   <html xmlns="http://www.w3.org/1999/xhtml" xml:lang="en" lang="en">
   ```

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 DTDs 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/xhtml1/DTD/xhtml1-strict.dtd">

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

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

5. The DTD subset must not be used to override any parameter entities in the DTD.

An XML declaration is not required in all XML documents; however 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 and no encoding was determined by a higher-level protocol. Here is an example of an XHTML document. In this example, the XML declaration is included.

```
<?xml version="1.0" encoding="UTF-8"?>
```

```
<!DOCTYPE html
     PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
    "http://www.w3.org/TR/xhtml1/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://example.org/">example.org</a>.</p>
  </body>
</html>
```

3.1.2. Using XHTML with other namespaces

The XHTML namespace may be used with other XML namespaces as per [XMLNS], although such documents are not strictly conforming XHTML 1.0 documents as defined above. Work by W3C is addressing ways to specify conformance for documents involving multiple namespaces. For an example, see [XHTML+MathML].

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], 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].
2. When the user agent claims to support facilities 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` (i.e. the `id` attribute on most XHTML elements) as fragment identifiers.
4. If a user agent encounters an element it does not recognize, it must process 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 does not recognize, it must use the default attribute value.
7. If it encounters an entity reference (other than one of the entities defined in this recommendation or in the XML recommendation) 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 processed as the characters (starting with the ampersand and ending with the semi-colon) that make up the entity reference.
8. When processing content, user agents that encounter characters or character entity references that are recognized but not renderable may substitute another rendering that gives the same meaning, or must display the document in such a way that it is obvious to the user that normal rendering has not taken place.

9. White space is handled according to the following rules. The following characters are defined in [XML] white space characters:

    - SPACE ( )
    - HORIZONTAL TABULATION (	)
    - CARRIAGE RETURN (
)
    - LINE FEED (
)

    The XML processor normalizes different systems' line end codes into one single LINE FEED character, that is passed up to the application.

    The user agent must use the definition from CSS for processing whitespace characters [CSS2]. *Note that the CSS2 recommendation does not explicitly address the issue of whitespace handling in non-Latin character sets. This will be addressed in a future version of CSS, at which time this reference will be updated.*

Note that in order to produce a Canonical XHTML document, the rules above must be applied and the rules in [XMLC14N] must also be applied to the document.

4. Differences with HTML 4

This section is informative.

Due to the fact that XHTML is an XML application, certain practices that were perfectly legal in SGML-based HTML 4 [HTML4] must be changed.

4.1. Documents must be well-formed

Well-formedness is a new concept introduced by [XML]. 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 properly.

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

CORRECT: nested elements.

here is an emphasized paragraph.

INCORRECT: overlapping elements

here is an emphasized paragraph.

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 certain elements were permitted to omit the end tag; with the elements that followed implying closure. XML does not allow end tags to be omitted. All elements other than those declared in the DTD as `EMPTY` must have an end tag. Elements that are declared in the DTD as `EMPTY` can have an end tag *or* can use empty element shorthand (see Empty Elements).

CORRECT: terminated elements

here is a paragraph. here is another paragraph.

INCORRECT: unterminated elements

here is a paragraph. 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

<td rowspan="3">

INCORRECT: unquoted attribute values

<td rowspan=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 for information on ways to ensure this is backward compatible with HTML 4 user agents.

CORRECT: terminated empty elements

<hr/>

INCORRECT: unterminated empty elements

<hr>

4.7. White Space handling in attribute values

When user agents process attributes, they do so according to Section 3.3.3 of [XML]:

- Strip leading and trailing white space.
- Map sequences of one or more white space characters (including line breaks) to a single inter-word space.

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 type="text/javascript">
<![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].

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 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 Element Prohibitions.

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

HTML 4 defined the `name` attribute for the elements `a`, `applet`, `form`, `frame`, `iframe`, `img`, and `map`. HTML 4 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 on the elements listed above. See the HTML Compatibility Guidelines for information on ensuring such anchors are backward 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.

4.11. Attributes with pre-defined value sets

HTML 4 and XHTML both have some attributes that have pre-defined and limited sets of values (e.g. the `type` attribute of the `input` element). In SGML and XML, these are called *enumerated attributes*. Under HTML 4, the interpretation of these values was *case-insensitive*, so a value of `TEXT` was equivalent to a value of `text`. Under XML, the interpretation of these values is *case-sensitive*, and in XHTML 1 all of these values are defined in lower-case.

4.12. Entity references as hex values

SGML and XML both permit references to characters by using hexadecimal values. In SGML these references could be made using either &#Xnn; or &#xnn;. In XML documents, you must use the lower-case version (i.e. &#xnn;)

5. Compatibility Issues

This section is normative.

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.

5.1. Internet Media Type

XHTML Documents which follow the guidelines set forth in Appendix C, "HTML Compatibility Guidelines" may be labeled with the Internet Media Type "text/html" [RFC2854], as they are compatible with most

HTML browsers. Those documents, and any other document conforming to this specification, may also be labeled with the Internet Media Type "application/xhtml+xml" as defined in [RFC3236]. For further information on using media types with XHTML, see the informative note [XHTMLMIME].

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 and the gzip'd tar file for this specification. Users looking for local copies of the DTDs to work with should download and use those archives rather than using the specific DTDs referenced below.

A.1. Document Type Definitions

These DTDs approximate the HTML 4 DTDs. The W3C recommends that you use the authoritative versions of these DTDs at their defined SYSTEM identifiers when validating content. If you need to use these DTDs locally you should download one of the archives of this version. For completeness, the normative versions of the DTDs are included here:

A.1.1. XHTML-1.0-Strict

The file DTD/xhtml1-strict.dtd is a normative part of this specification. The annotated contents of this file are available in this separate section for completeness.

A.1.2. XHTML-1.0-Transitional

The file DTD/xhtml1-transitional.dtd is a normative part of this specification. The annotated contents of this file are available in this separate section for completeness.

A.1.3. XHTML-1.0-Frameset

The file DTD/xhtml1-frameset.dtd is a normative part of this specification. The annotated contents of this file are available in this separate section for completeness.

A.2. Entity Sets

The XHTML entity sets are the same as for HTML 4, but have been modified to be valid XML 1.0 entity declarations. Note the entity for the Euro currency sign (`&euro;` or `&#8364;` or `&#x20AC;`) is defined as part of the special characters.

A.2.1. Latin-1 characters

The file DTD/xhtml-lat1.ent is a normative part of this specification. The annotated contents of this file are available in this separate section for completeness.

A.2.2. Special characters

The file DTD/xhtml-special.ent is a normative part of this specification. The annotated contents of this file are available in this separate section for completeness.

A.2.3. Symbols

The file DTD/xhtml-symbol.ent is a normative part of this specification. The annotated contents of this file are available in this separate section for completeness.

B. Element Prohibitions

This appendix is normative.

The following elements have prohibitions on which elements they can contain (see SGML Exclusions). This prohibition applies to all depths of nesting, i.e. it contains all the descendant elements.

`a`
: must not contain other `a` elements.

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

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

`label`
: must not contain other `label` elements.

`form`
: must not contain other `form` elements.

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. *Note that this recommendation does not define how HTML conforming user agents should process HTML documents. Nor does it define the meaning of the Internet Media Type* `text/html`*. For these definitions, see [HTML4] and [RFC2854] respectively.*

C.1. Processing Instructions and the XML Declaration

Be aware that processing instructions are rendered on some user agents. Also, some user agents interpret the XML declaration to mean that the document is unrecognized XML rather than HTML, and therefore may not render the document as expected. For compatibility with these types of legacy browsers, you may want to avoid using processing instructions and XML declarations. Remember, however, 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   and not).

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 user agents.

C.5. Line Breaks within Attribute Values

Avoid line breaks and multiple white space 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, URI-references [RFC2396] 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. 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., ...).

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 2.3, 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.

Note that the collection of legal values in XML 1.0 Section 2.3, production 5 is much larger than that permitted to be used in the `ID` and `NAME` types defined in HTML 4. When defining fragment identifiers to be backward-compatible, only strings matching the pattern `[A-Za-z][A-Za-z0-9:_.-]*` should be used. See Section 6.2 of [HTML4] for more information.

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

C.9. Character Encoding

Historically, the character encoding of an HTML document is either specified by a web server via the charset parameter of the HTTP Content-Type header, or via a `meta` element in the document itself. In an XML document, the character encoding of the document is specified on the XML declaration (e.g., `<?xml version="1.0" encoding="EUC-JP"?>`). In order to portably present documents with specific character encodings, the best approach is to ensure that the web server provides the correct headers. If this is not possible, a document that wants to set its character encoding explicitly must include both the XML declaration an encoding declaration and a `meta` http-equiv statement (e.g., `<meta http-equiv="Content-type" content="text/html; charset=EUC-JP" />`). In XHTML-conforming user agents, the value of the encoding declaration of the XML declaration takes precedence.

Note: be aware that if a document must include the character encoding declaration in a meta http-equiv statement, that document may always be interpreted by HTTP servers and/or user agents as being of the internet media type defined in that statement. If a document is to be served as multiple media types, the HTTP server must be used to set the encoding of the document.

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 affect user agents compliant with HTML 4. 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] defines document object model interfaces for XML and HTML 4. The HTML 4 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. User agents 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. User agents that access XHTML documents served as Internet media types `text/xml`, `application/xml`, or `application/xhtml+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 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. User agents need to adapt to this accordingly. For further information on this topic, see [DOM2]

C.12. Using Ampersands in Attribute Values (and Elsewhere)

In both SGML and XML, the ampersand character ("&") declares the beginning of an entity reference (e.g., ® for the registered trademark symbol "®"). Unfortunately, many HTML user agents have silently ignored incorrect usage of the ampersand character in HTML documents - treating ampersands that do not look like entity references as literal ampersands. XML-based user agents will not tolerate this incorrect usage, and any document that uses an ampersand incorrectly will not be "valid", and consequently will not conform to this specification. In order to ensure that documents are compatible with historical HTML user agents and XML-based user agents, ampersands used in a document that are to be treated as literal characters must be expressed themselves as an 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] defines style properties which are applied to the parse tree of the HTML or XML documents. 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 explicitly add a tbody element if it is referred to in a CSS selector.
3. Within the XHTML namespace, 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 namespace, 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.

C.14. Referencing Style Elements when serving as XML

In HTML 4 and XHTML, the `style` element can be used to define document-internal style rules. In XML, an XML stylesheet declaration is used to define style rules. In order to be compatible with this convention, `style` elements should have their fragment identifier set using the `id` attribute, and an XML stylesheet declaration should reference this fragment. For example:

```
<?xml-stylesheet href="http://www.w3.org/StyleSheets/TR/W3C-REC.css" type="text/css"?>
<?xml-stylesheet href="#internalStyle" type="text/css"?>
<!DOCTYPE html
     PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
     "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="en" lang="en">
<head>
```

```
<title>An internal stylesheet example</title>
<style type="text/css" id="internalStyle">
  code {
    color: green;
    font-family: monospace;
    font-weight: bold;
  }
</style>
</head>
<body>
<p>
  This is text that uses our
  <code>internal stylesheet</code>.
</p>
</body>
</html>
```

C.15. White Space Characters in HTML vs. XML

Some characters that are legal in HTML documents, are illegal in XML document. For example, in HTML, the Formfeed character (U+000C) is treated as white space, in XHTML, due to XML's definition of characters, it is illegal.

C.16. The Named Character Reference '

The named character reference `&apos;` (the apostrophe, U+0027) was introduced in XML 1.0 but does not appear in HTML. Authors should therefore use `&#39;` instead of `&apos;` to work as expected in HTML 4 user agents.

D. Acknowledgements

This appendix is informative.

This specification was written with the participation of the members of the W3C HTML Working Group.

At publication of the second edition, the membership was:

Steven Pemberton, CWI/W3C (HTML Working Group Chair)
Daniel Austin, Grainger
Jonny Axelsson, Opera Software
Tantek Çelik, Microsoft
Doug Dominiak, Openwave Systems
Herman Elenbaas, Philips Electronics
Beth Epperson, Netscape/AOL
Masayasu Ishikawa, W3C (HTML Activity Lead)
Shin'ichi Matsui, Panasonic
Shane McCarron, Applied Testing and Technology
Ann Navarro, WebGeek, Inc.
Subramanian Peruvemba, Oracle
Rob Relyea, Microsoft
Sebastian Schnitzenbaumer, SAP
Peter Stark, Sony Ericsson

At publication of the first edition, the membership was:

Steven Pemberton, CWI (HTML Working Group Chair)
Murray Altheim, Sun Microsystems
Daniel Austin, AskJeeves (CNET: The Computer Network through July 1999)
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, Panasonic (W3C visiting engineer through September 1999)
Shane McCarron, Applied Testing and Technology (The Open Group through August 1999)
Ann Navarro, HTML Writers Guild
Zach Nies, Quark
Dave Raggett, W3C/HP (HTML Activity Lead)
Patrick Schmitz, Microsoft
Sebastian Schnitzenbaumer, Stack Overflow
Peter Stark, Phone.com
Chris Wilson, Microsoft
Ted Wugofski, Gateway 2000
Dan Zigmond, WebTV Networks

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.
Latest version available at: http://www.w3.org/TR/REC-CSS2

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

[DOM2]
"*Document Object Model (DOM) Level 2 Core Specification*", A. Le Hors, *et al.*, 13 November 2000.
Latest version available at: http://www.w3.org/TR/DOM-Level-2-Core

[HTML]
"*HTML 4.01 Specification*", D. Raggett, A. Le Hors, I. Jacobs, 24 December 1999.
Latest version available at: http://www.w3.org/TR/html401

[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.

[RFC2045]
"*Multipurpose Internet Mail Extensions (MIME) Part One: Format of Internet Message Bodies*", N.

Freed and N. Borenstein, November 1996. Note that this RFC obsoletes RFC1521, RFC1522, and RFC1590.

[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.
This document is obsoleted by [RFC3023].
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

[RFC2854]
"*RFC2854: The text/html Media Type*", D. Conolly, L. Masinter, June 2000.
Available at: http://www.ietf.org/rfc/rfc2854.txt

[RFC3023]
"*RFC3023: XML Media Types*", M. Murata, S. St.Laurent, D. Kohn, January 2001.
This document obsoletes [RFC2376].
Available at: http://www.ietf.org/rfc/rfc3023.txt

[RFC3066]
"Tags for the Identification of Languages", H. Alvestrand, January 2001.
Available at: http://www.ietf.org/rfc/rfc3066.txt

[RFC3236]
"The 'application/xhtml+xml' Media Type", M. Baker, P. Stark, January 2002.
Available at: http://www.ietf.org/rfc/rfc3236.txt

[XHTML+MathML]
"XHTML plus Math 1.1 DTD", "A.2 MathML as a DTD Module", Mathematical Markup Language (MathML) Version 2.0. Available at: http://www.w3.org/TR/MathML2/dtd/xhtml-math11-f.dtd

[XHTMLMIME]
"*XHTML Media Types*", Masayasu Ishikawa, 1 August 2002.
Latest version available at: http://www.w3.org/TR/xhtml-media-types

[XHTMLMOD]
"*Modularization of XHTML*", M. Altheim et al., 10 April 2001.
Latest version available at: http://www.w3.org/TR/xhtml-modularization

[XML]
"Extensible Markup Language (XML) 1.0 Specification (Second Edition)", T. Bray, J. Paoli, C. M. Sperberg-McQueen, E. Maler, 6 October 2000.
Latest version available at: http://www.w3.org/TR/REC-xml

[XMLNS]
"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.
Latest version available at: http://www.w3.org/TR/REC-xml-names

[XMLC14N]

"Canonical XML Version 1.0", J. Boyer, 15 March 2001.
This document describes a method for generating a physical representation, the canonical form, of an XML document.
Latest version available at: http://www.w3.org/TR/xml-c14n