**REC-xml-19980210**

# Extensible Markup Language (XML) 1.0

## W3C Recommendation 10-February-1998

**This version:**
http://www.w3.org/TR/1998/REC-xml-19980210
http://www.w3.org/TR/1998/REC-xml-19980210.xml
http://www.w3.org/TR/1998/REC-xml-19980210.html
http://www.w3.org/TR/1998/REC-xml-19980210.pdf
http://www.w3.org/TR/1998/REC-xml-19980210.ps
**Latest version:**
http://www.w3.org/TR/REC-xml
**Previous version:**
http://www.w3.org/TR/PR-xml-971208

**Editors:**
Tim Bray (Textuality and Netscape) <tbray@textuality.com>
Jean Paoli (Microsoft) <jeanpa@microsoft.com>
C. M. Sperberg-McQueen (University of Illinois at Chicago) <cmsmcq@uic.edu>

## Abstract

The Extensible Markup Language (XML) is a subset of SGML that is completely described in this document. Its goal is to enable generic SGML to be served, received, and processed on the Web in the way that is now possible with HTML. XML has been designed for ease of implementation and for interoperability with both SGML and HTML.

## Status of this document

This document has been reviewed by W3C Members and other interested parties and has been endorsed by the Director as a W3C Recommendation. It is a stable document and may be used as reference material or cited as a normative reference from another document. W3C's role in making the Recommendation is to draw attention to the specification and to promote its widespread deployment. This enhances the functionality and interoperability of the Web.

This document specifies a syntax created by subsetting an existing, widely used international text processing standard (Standard Generalized Markup Language, ISO 8879:1986(E) as amended and corrected) for use on the World Wide Web. It is a product of the W3C XML Activity, details of which can be found at http://www.w3.org/XML. A list of current W3C Recommendations and other technical documents can be found at http://www.w3.org/TR.

This specification uses the term URI, which is defined by [Berners-Lee et al.], a work in progress expected to update [IETF RFC1738] and [IETF RFC1808].

The list of known errors in this specification is available at http://www.w3.org/XML/xml-19980210-errata.

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

# Extensible Markup Language (XML) 1.0

## Table of Contents

## Appendices

documents.

XML documents are made up of storage units called entities, which contain either parsed or unparsed data. Parsed data is made up of characters, some of which form character data, and some of which form markup. Markup encodes a description of the document's storage layout and logical structure. XML provides a mechanism to impose constraints on the storage layout and logical structure.

A software module called an **XML processor** is used to read XML documents and provide access to their content and structure. It is assumed that an XML processor is doing its work on behalf of another module, called the **application**. This specification describes the required behavior of an XML processor in terms of how it must read XML data and the information it must provide to the application.

## 1.1 Origin and Goals

XML was developed by an XML Working Group (originally known as the SGML Editorial Review Board) formed under the auspices of the World Wide Web Consortium (W3C) in 1996. It was chaired by Jon Bosak of Sun Microsystems with the active participation of an XML Special Interest Group (previously known as the SGML Working Group) also organized by the W3C. The membership of the XML Working Group is given in an appendix. Dan Connolly served as the WG's contact with the W3C.

The design goals for XML are:

1. XML shall be straightforwardly usable over the Internet.
2. XML shall support a wide variety of applications.
3. XML shall be compatible with SGML.
4. It shall be easy to write programs which process XML documents.
5. The number of optional features in XML is to be kept to the absolute minimum, ideally zero.
6. XML documents should be human-legible and reasonably clear.
7. The XML design should be prepared quickly.
8. The design of XML shall be formal and concise.
9. XML documents shall be easy to create.
10. Terseness in XML markup is of minimal importance.

This specification, together with associated standards (Unicode and ISO/IEC 10646 for characters, Internet RFC 1766 for language identification tags, ISO 639 for language name codes, and ISO 3166 for country name codes), provides all the information necessary to understand XML Version 1.0 and construct computer programs to process it.

This version of the XML specification may be distributed freely, as long as all text and legal notices remain intact.

## 1.2 Terminology

The terminology used to describe XML documents is defined in the body of this specification. The terms defined in the following list are used in building those definitions and in describing the actions of an XML processor:

**may**
: Conforming documents and XML processors are permitted to but need not behave as described.

**must**
: Conforming documents and XML processors are required to behave as described; otherwise they are in error.

**error**
: A violation of the rules of this specification; results are undefined. Conforming software may detect and report an error and may recover from it.

**fatal error**
: An error which a conforming XML processor must detect and report to the application. After encountering a fatal error, the processor may continue processing the data to search for further errors and may report such errors to the application. In order to support correction of errors, the processor may make unprocessed data from the document (with intermingled character data and markup) available to the application. Once a fatal error is detected, however, the processor must not continue normal processing (i.e., it must not continue to pass character data and information about the document's logical structure to the application in the normal way).

**at user option**
: Conforming software may or must (depending on the modal verb in the sentence) behave as described; if it does, it must provide users a means to enable or disable the behavior described.

**validity constraint**

representations in ISO/IEC 10646 (e.g. characters with both precomposed and base+diacritic forms) match only if they have the same representation in both strings. At user option, processors may normalize such characters to some canonical form. No case folding is performed. (Of strings and rules in the grammar:) A string matches a grammatical production if it belongs to the language generated by that production. (Of content and content models:) An element matches its declaration when it conforms in the fashion described in the constraint "Element Valid".

**for compatibility**
A feature of XML included solely to ensure that XML remains compatible with SGML.

**for interoperability**
A non-binding recommendation included to increase the chances that XML documents can be processed by the existing installed base of SGML processors which predate the WebSGML Adaptations Annex to ISO 8879.

## 2. Documents

A data object is an **XML document** if it is well-formed, as defined in this specification. A well-formed XML document may in addition be valid if it meets certain further constraints.

Each XML document has both a logical and a physical structure. Physically, the document is composed of units called entities. An entity may refer to other entities to cause their inclusion in the document. A document begins in a "root" or document entity. Logically, the document is composed of declarations, elements, comments, character references, and processing instructions, all of which are indicated in the document by explicit markup. The logical and physical structures must nest properly, as described in "4.3.2 Well-Formed Parsed Entities".

### 2.1 Well-Formed XML Documents

A textual object is a well-formed XML document if:

1. Taken as a whole, it matches the production labeled `document`.
2. It meets all the well-formedness constraints given in this specification.
3. Each of the parsed entities which is referenced directly or indirectly within the document is well-formed.

| Document | |
|---|---|
| [1] | `document ::= prolog element Misc*` |

Matching the `document` production implies that:

1. It contains one or more elements.
2. There is exactly one element, called the **root**, or document element, no part of which appears in the content of any other element. For all other elements, if the start-tag is in the content of another element, the end-tag is in the content of the same element. More simply stated, the elements, delimited by start- and end-tags, nest properly within each other.

As a consequence of this, for each non-root element `C` in the document, there is one other element `P` in the document such that `C` is in the content of `P`, but is not in the content of any other element that is in the content of `P`. `P` is referred to as the **parent** of `C`, and `C` as a **child** of `P`.

### 2.2 Characters

A parsed entity contains **text**, a sequence of characters, which may represent markup or character data. A **character** is an atomic unit of text as specified by ISO/IEC 10646 [ISO/IEC 10646]. Legal characters are tab, carriage return, line feed, and the legal graphic characters of Unicode and ISO/IEC 10646. The use of "compatibility characters", as defined in section 6.8 of [Unicode], is discouraged.

| Character Range | | |
|---|---|---|
| [2] | `Char ::= #x9 \| #xA \| #xD \| [#x20-#xD7FF] \| [#xE000-#xFFFD] \| [#x10000-#x10FFFF]` | `/* any Unicode character, excluding the surrogate blocks, FFFE, and FFFF. */` |

This section defines some symbols used widely in the grammar.
S (white space) consists of one or more space (#x20) characters, carriage returns, line feeds, or tabs.

**White Space**

```
[3] S ::= (#x20 | #x9 | #xD | #xA)+
```

Characters are classified for convenience as letters, digits, or other characters. Letters consist of an alphabetic or syllabic base character possibly followed by one or more combining characters, or of an ideographic character. Full definitions of the specific characters in each class are given in "B. Character Classes".

A **Name** is a token beginning with a letter or one of a few punctuation characters, and continuing with letters, digits, hyphens, underscores, colons, or full stops, together known as name characters. Names beginning with the string "`xml`", or any string which would match `(('X'|'x') ('M'|'m') ('L'|'l'))`, are reserved for standardization in this or future versions of this specification.

**Note:** The colon character within XML names is reserved for experimentation with name spaces. Its meaning is expected to be standardized at some future point, at which point those documents using the colon for experimental purposes may need to be updated. (There is no guarantee that any name-space mechanism adopted for XML will in fact use the colon as a name-space delimiter.) In practice, this means that authors should not use the colon in XML names except as part of name-space experiments, but that XML processors should accept the colon as a name character.

An `Nmtoken` (name token) is any mixture of name characters.

**Names and Tokens**

```
[4] NameChar ::= Letter | Digit | '.' | '-' | '_' | ':' | CombiningChar | Extender
[5]     Name ::= (Letter | '_' | ':') (NameChar)*
[6]    Names ::= Name (S Name)*
[7]  Nmtoken ::= (NameChar)+
[8] Nmtokens ::= Nmtoken (S Nmtoken)*
```

Literal data is any quoted string not containing the quotation mark used as a delimiter for that string. Literals are used for specifying the content of internal entities (`EntityValue`), the values of attributes (`AttValue`), and external identifiers (`SystemLiteral`). Note that a `SystemLiteral` can be parsed without scanning for markup.

**Literals**

```
 [9]   EntityValue ::= '"' ([^%&"] | PEReference | Reference)* '"'
                    |  "'" ([^%&'] | PEReference | Reference)* "'"
[10]      AttValue ::= '"' ([^<&"] | Reference)* '"'
                    |  "'" ([^<&'] | Reference)* "'"
[11] SystemLiteral ::= ('"' [^"]* '"') | ("'" [^']* "'")
[12]  PubidLiteral ::= '"' PubidChar* '"' | "'" (PubidChar - "'")* "'"
[13]     PubidChar ::= #x20 | #xD | #xA | [a-zA-Z0-9] | [-'()+,./:=?;!*#@$_%]
```

## 2.4 Character Data and Markup

Text consists of intermingled character data and markup. **Markup** takes the form of start-tags, end-tags, empty-element tags, entity references, character references, comments, CDATA section delimiters, document type declarations, and processing instructions.

All text that is not markup constitutes the **character data** of the document.

The ampersand character (&) and the left angle bracket (<) may appear in their literal form *only* when used as markup delimiters, or within a comment, a processing instruction, or a CDATA section. They are also legal within the literal entity value of an internal entity declaration; see "4.3.2 Well-Formed Parsed Entities". If they are needed elsewhere, they must be escaped using either numeric character references or the strings "`&amp;`" and "`&lt;`" respectively. The right angle bracket (>) may be represented using the string "`&gt;`", and must, for compatibility, be escaped using "`&gt;`" or a character reference when it appears in the string "`]]>`" in content, when that string is not marking the end of a CDATA section.

In the content of elements, character data is any string of characters which does not contain the start-delimiter of any

# Explore Litigation Insights

Docket Alarm provides insights to develop a more informed litigation strategy and the peace of mind of knowing you're on top of things.

## Real-Time Litigation Alerts

Keep your litigation team up-to-date with **real-time alerts** and advanced team management tools built for the enterprise, all while greatly reducing PACER spend.

Our comprehensive service means we can handle Federal, State, and Administrative courts across the country.

## Advanced Docket Research

With over 230 million records, Docket Alarm's cloud-native docket research platform finds what other services can't. Coverage includes Federal, State, plus PTAB, TTAB, ITC and NLRB decisions, all in one place.

Identify arguments that have been successful in the past with full text, pinpoint searching. Link to case law cited within any court document via Fastcase.

## Analytics At Your Fingertips

Learn what happened the last time a particular judge, opposing counsel or company faced cases similar to yours.

Advanced out-of-the-box PTAB and TTAB analytics are always at your fingertips.

## API

Docket Alarm offers a powerful API (application programming interface) to developers that want to integrate case filings into their apps.

**LAW FIRMS**

Build custom dashboards for your attorneys and clients with live data direct from the court.

Automate many repetitive legal tasks like conflict checks, document management, and marketing.

**FINANCIAL INSTITUTIONS**

Litigation and bankruptcy checks for companies and debtors.

**E-DISCOVERY AND LEGAL VENDORS**

Sync your system to PACER to automate legal marketing.

**WHAT WILL YOU BUILD?** | **sales@docketalarm.com** | **1-866-77-FASTCASE**

fastcase® Smarter legal research.