WD-xml-961114

Extensible Markup Language (XML)

W3C Working Draft 14-Nov-96

This version:
http://www.w3.org/pub/WWW/TR/WD-xml-961114.html
Previous versions:
Latest version:
http://www.textuality.com/sgml-erb/WD-xml.html

Editors:
Tim Bray (Textuality) <tbray@textuality.com>
C. M. Sperberg-McQueen (University of Illinois at Chicago) <cmsmcq@uic.edu>

Status of this memo

This is a W3C Working Draft for review by W3C members and other interested parties. It is a draft document and may be updated, replaced or obsoleted by other documents at any time. It is inappropriate to use W3C Working Drafts as reference material or to cite them as other than "work in progress". A list of current W3C working drafts can be found at: http://www.w3.org/pub/WWW/TR

Note: since working drafts are subject to frequent change, you are advised to reference the above URL, rather than the URLs for working drafts themselves.

This work is part of the W3C SGML Activity.

Abstract

Extensible Markup Language (XML) is an extremely simple dialect of SGML which is completely described in this document. The goal is to enable generic SGML to be served, received, and processed on the Web in the way that is now possible with HTML. For this reason, XML has been designed for ease of implementation, and for interoperability with both SGML and HTML.
Note on status of this document: This is even more of a moving target than the typical W3C working draft. Several important decisions on the details of XML are still outstanding - members of the W3C SGML Working Group will recognize these areas of particular volatility in the spec, but those who are not intimately familiar with the deliberative process should be careful to avoid actions based on the content of this document, until the notice you are now reading has been removed.

Table of Contents

- 1. Introduction
  - 1.1 Origin and Goals
  - 1.2 Relationship to Other Standards
  - 1.3 Notation

- 1.4 Terminology
- 1.5 Common Syntactic Constructs
- 2. Documents
  - 2.1 Logical and Physical Structure
  - 2.2 Characters
  - 2.3 Syntax of Text and Markup
  - 2.4 Comments
  - 2.5 Processing Instructions
  - 2.6 Marked Sections
  - 2.7 White Space Handling
  - 2.8 Prolog and Document Type Declaration
  - 2.9 Required Markup Declaration
- 3. Logical Structures
  - 3.1 Start- and End-Tags
  - 3.2 Well-Formed XML Documents
  - 3.3 Element Declaration
    - 3.3.1 Mixed Content
    - 3.3.2 Element Content
  - 3.4 Attribute Declaration
    - 3.4.1 Attribute Types
    - 3.4.2 Attribute Defaults
  - 3.5 Partial DTD Information
- 4. Physical Structures
  - 4.1 Character and Entity References
  - 4.2 Declaring Entities
    - 4.2.1 Internal Entities
    - 4.2.2 External Entities
    - 4.2.3 Character Encoding in Entities
    - 4.2.4 The Document Entity
  - 4.3 XML Processor Treatment of Entities
  - 4.4 Notation Declaration
- 5. Conformance
- A. XML and SGML
- B. References
- C. Working Group and Editorial Review Board Membership
  - C.1 Working Group
  - C.2 Editorial Review Board

---

1. Introduction

Extensible Markup Language, abbreviated XML, describes a class of data objects stored on computers and partially describes the behavior of programs which process these objects. Such objects are called XML documents. XML is an application profile or restricted form of SGML, the Standard Generalized Markup Language [ISO 8879].

XML documents are made up of storage units called entities, which contain either text or binary data. Text is made up of characters, some of which form the character data in the document, and some of which form markup. Markup encodes a description of the document's storage layout, structure, and arbitrary attribute-value pairs associated with that 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, referred to as

the *application*. This specification describes some of the required behavior of an XML processor in terms of the manner it must read XML data, and the information it must provide to the application.

1.1 Origin and Goals

XML was developed by a Generic SGML Editorial Review Board formed under the auspices of the W3 Consortium in 1996 and chaired by Jon Bosak of Sun Microsystems, with the very active participation of a Generic SGML Working Group also organized by the W3C. The membership of these groups is given in an appendix.

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 is of minimal importance.

This specification, together with the associated standards, provides all the information necessary to understand XML version 1.0 and construct computer programs to process it.

This version of the XML specification (0.01) is for internal discussion within the SGML ERB only. It should not be distributed outside the ERB.

Known problems in version 0.01:

1. Several items in the bibliography have no references to them; several references in the text do not point to anything in the bibliograpy.
2. The EBNF grammar has not been checked for completeness, and has at least two start productions.
3. The description of conformance in the final section is incomplete.
4. Language exists in the spec which describes the effect of several decisions which have not been taken. Specifically, XML may have INCLUDE/IGNORE marked sections as does SGML, the comment syntax may change, XML may have CONREF attributes, the 8879 syntax for EMPTY elements may be outlawed, XML may choose to rule out what 8879 calls "ambiguous" content models, XML may choose to prohibit overlap between enumerated attribute values for different attributes, the handling for attribute values in the absence of a DTD may be specified, there may be a way to signal whether the DTD is complete, the DTD summary may be dropped, and XML may support parameter entities, and XML may predefine a large number of character entities, for example those from HTML 3.2.

1.2 Relationship to Other Standards

Other standards relevant to users and implementors of XML include:

1. SGML (ISO 8879-1986). Valid XML Documents are SGML documents in the sense described in ISO standard 8879.
2. Unicode, ISO 10646. This specification depends on ISO standard 10646 and the technically identical Unicode Standard, Version 2.0, which define the encodings and meanings of the characters which make up XML text data.

3. IETF RFC 1738. This specification defines the syntax and semantics of Uniform Resource Locators, or URLs.
4. World-Wide Web Consortium Working Draft WD-html32-960909: HTML 3.2 Reference Specification. This includes the repertoire of characters to be predefined by an XML processor.

1.3 Notation

The formal grammar of XML is given using a simple Extended Backus-Naur Form (EBNF) notation. Each rule in the grammar defines one non-terminal or terminal symbol of the grammar, in the form

```
symbol ::= expression
```

Symbols are written with an initial capital letter if they are defined by a regular expression, with an initial lowercase letter if they have a more complex definition (i.e. if they require a stack for proper recognition). Literal strings are quoted; unless otherwise noted they are not case-sensitive. The distinction between symbols which can and cannot be recognized using simple regular-expressions is made for clarity only. It may be reflected in the boundary between an implementation's lexical scanner and its parser, but there is are no assumptions about the placement of such a boundary, nor even that the implementation has separate modules for parser and lexical scanner.

Within the expression on the right-hand side of a rule, the meaning of symbols is as shown below:

`#NN`
: where *NN* is a decimal integer, the expression matches the character in ISO 10646 whose UCS-4 bit-string, when interpreted as an unsigned binary number, has the value indicated

`#xNN`
: where *NN* is a hexadecimal integer, the expression matches the character in ISO 10646 whose UCS-4 bit-string, when interpreted as an unsigned binary number, has the value indicated

`[#xNN-#xNN], [a-zA-Z]`
: matches any character with a value in the range(s) indicated (inclusive)

`[^#xNN-#xNN], [^a-z]`
: matches any character with a value *outside* the range indicated

`[^abc]`
: matches any character with a value not among the characters given

`"`*`string`*`"`
: matches the literal string given inside the double quotes

`'`*`string`*`'`
: matches the literal string given inside the single quotes

`a b`
: *a* followed by *b*

`a | b`
: *a* or *b* but not both

`a?`
: *a* or nothing; optional *a*

`a+`
: one or more occurrences of *a*

`a*`
: zero or more occurrences of *a*

`(`*`expression`*`)`
: *expression* is treated as a unit; allows subgroups to carry the operators `?`, `*`, or `+`

`/* ... */`
: comment

`[ WFC: ... ]`

Well-formedness check; this identifies by name a check for well-formedness that is associated with a production.

`[ VC: ... ]`
Validity check; this identifies by name a check for validity that is associated with a production.

1.4 Terminology

Some terms used with special meaning in this specification are:

`may`
Conforming data and software may but need not behave as described.

`must`
Conforming data and software must 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.

`reportable error`
An error which conforming software must report to the user, unless the user has explicitly disabled error reporting.

`validity constraint`
A rule which applies to all valid XML documents. Violations of validity constraints are errors; they must be reported by validating XML processors.

`well-formedness constraint`
A rule which applies to all well-formed XML documents. Violations of well-formedness constraints are reportable errors.

`at user option`
Conforming software may or must (depending on the verb in the sentence) provide users a means to select the behavior described; it must also allow the user *not* to select it.

`match`
Case-insensitive match: two strings or names being compared must be identical except for differences between upper- and lower-case letters in scripts which have such a distinction. Characters with multiple possible representations in ISO 10646 (e.g. both precomposed and base+diacritic forms) match only if they have the same representation, except for case differences, in both strings. Case folding must be performed as specified in The Unicode Standard, Version 2.0, section 4.1; in particular, it is recommended that case-insensitive matching be performed by folding uppercase letters to lowercase, not vice versa.

`exact(ly) match`
Case-sensitive match: two strings or names being compared must be identical. Characters with multiple possible representations in ISO 10646 (e.g. both precomposed and base+diacritic forms) match only if they have the same representation in both strings.

`for compatibility`
A feature of XML included solely to ensure that XML remains compatible with SGML; the expectation is that in many cases, those aspects of SGML that are not required to satisfy XML's requirements but mandated only to achieve conformance may be removed or replaced in the near future by the organizations that maintain that standard.

1.5 Common Syntactic Constructs

This section defines some symbols used widely in the grammar.

S (white space) consists of one or more blank characters, carriage returns, line feeds, or tabs.

< 1 White space >

DOCKET ALARM

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.