XML

Contributors to Wikimedia projects

XML (standard)

Extensible Markup Language	
Abbreviation	XML
Status	Published, W3C recommendation
Year started	1996; 26 years ago
First published	February 10, 1998; 23 years ago
Latest version	1.1 (2nd ed.) September 29, 2006; 15 years ago
Organization	World Wide Web Consortium (W3C)
Editors	Tim Bray, Jean Paoli, Michael Sperberg-McQueen, Eve Maler, François Yergeau, John W. Cowan
Base standards	SGML
Related standards	W3C XML Schema
Domain	Serialization
Website	www.w3.org/xml

XML (file format)

Filename extension	.xml
Internet media type	`application/xml`, `text/xml`[1]
Uniform Type Identifier (UTI)	public.xml
UTI conformation	public.text
Magic number	`<?xml`
Developed by	World Wide Web Consortium
Type of format	Markup language
Extended from	SGML

Extended to	[Numerous languages](#), including [XHTML](#), [RSS](#), [Atom](#), and [KML](#)
[Standard](#)	[1.0, 5th ed.](#) (November 26, 2008) [1.1, 2nd ed.](#) (August 16, 2006)
[Open format](#)?	Yes
Free format?	Yes

Extensible Markup Language (**XML**) is a [markup language](#) and [file format](#) for storing, transmitting, and reconstructing arbitrary data. It defines a set of rules for encoding [documents](#) in a format that is both [human-readable](#) and [machine-readable](#). The [World Wide Web Consortium](#)'s XML 1.0 Specification[2] of 1998[3] and several other related specifications[4]—all of them free [open standards](#)—define XML.[5]

The design goals of XML emphasize simplicity, generality, and usability across the [Internet](#).[6] It is a textual data format with strong support via [Unicode](#) for different [human languages](#). Although the design of XML focuses on documents, the language is widely used for the representation of arbitrary [data structures](#)[7] such as those used in [web services](#).

Several [schema systems](#) exist to aid in the definition of XML-based languages, while programmers have developed many [application programming interfaces](#) (APIs) to aid the processing of XML data.

Overview[[edit](#)]

The main purpose of XML is [serialization](#), i.e. storing, transmitting, and reconstructing arbitrary data. For two disparate systems to exchange information, they need to agree upon a file format. XML standardizes this process. XML is analogous to a [lingua franca](#) for representing information.[8]:1

As a [markup language](#), XML labels, categorizes, and structurally organizes information.[8]:11 XML tags represent the data structure and contain metadata. What's within the tags is data, encoded in the way the XML standard specifies.[8]:11 An additional [XML schema](#) (XSD) defines the necessary metadata for interpreting and validating XML. (This is also referred to as the canonical schema.)[8]:135 An XML document that adheres to basic XML rules is "well-formed"; one that adheres to its schema is "valid."[8]:135

[IETF](#) [RFC 7303](#) (which supersedes the older [RFC 3023](#)), provides rules for the construction of [media types](#) for use in XML message. It defines two base media types: `application/xml` and `text/xml`. They are be used for transmitting raw XML files without exposing their internal [semantics](#). RFC 7303 further recommends that XML-based languages be given media types ending in `+xml`, for example, `image/svg+xml` for [SVG](#).

Further guidelines for the use of XML in a networked context appear in [RFC 3470](#), also known as IETF BCP 70, a document covering many aspects of designing and deploying an XML-based language.

Applications[[edit](#)]

XML has come into common use for the interchange of data over the Internet. Hundreds of document formats using XML syntax have been developed,[9] including [RSS](#), [Atom](#), [Office Open XML](#), [OpenDocument](#), [SVG](#), and [XHTML](#). XML has also provides the base language for [communication protocols](#) such as [SOAP](#) and [XMPP](#). It is the message exchange format for the [Asynchronous JavaScript and XML (AJAX)](#) programming technique.

Many industry data standards, such as [Health Level 7](#), [OpenTravel Alliance](#), [FpML](#), [MISMO](#), and [National Information Exchange Model](#) are based on XML and the rich features of the XML schema specification. In publishing, [Darwin Information Typing Architecture](#) is an XML industry data standard. XML is used extensively to underpin various publishing formats.

Key terminology[[edit](#)]

The material in this section is based on the XML Specification. This is not an exhaustive list of all the constructs that appear in XML; it provides an introduction to the key constructs most often encountered in day-to-day use.

Character
> An XML document is a string of *characters*. [Almost](#) every legal [Unicode](#) character may appear in an XML document.

Processor and application
> The *processor* analyzes the markup and passes structured information to an *application*. The specification places requirements on what an XML processor must do and not do, but the application is outside its scope. The processor (as the specification calls it) is often referred to colloquially as an *XML parser*.

Markup and content
> The characters making up an XML document are divided into *markup* and *content*, which may be distinguished by the application of simple syntactic rules. Generally, strings that constitute markup either begin with the character `<` and end with a `>`, or they begin with the character `&` and end with a `;`. Strings of characters that are not markup are content. However, in a [CDATA](#) section, the delimiters `<![CDATA[` and `]]>` are classified as markup, while the text between them is classified as content. In addition, whitespace before and after the outermost element is classified as markup.

Tag
> A *tag* is a markup construct that begins with `<` and ends with `>`. There are three types of tag:
> - *start-tag*, such as `<section>`;
> - *end-tag*, such as `</section>`;
> - *empty-element tag*, such as `<line-break />`.

Element
> An *element* is a logical document component that either begins with a start-tag and ends with a matching end-tag or consists only of an empty-element tag. The characters between the start-tag and end-tag, if any, are the element's *content*, and may contain markup, including other elements, which are called *child elements*. An example is `<greeting>Hello, world!</greeting>`. Another is `<line-break />`.

Attribute
> An *attribute* is a markup construct consisting of a [name–value pair](#) that exists within a start-tag or empty-element tag. An example is `<img src="madonna.jpg"`

`alt="Madonna" />`, where the names of the attributes are "src" and "alt", and their values are "madonna.jpg" and "Madonna" respectively. Another example is `<step number="3">Connect A to B.</step>`, where the name of the attribute is "number" and its value is "3". An XML attribute can only have a single value and each attribute can appear at most once on each element. In the common situation where a list of multiple values is desired, this must be done by encoding the list into a well-formed XML attribute[i] with some format beyond what XML defines itself. Usually this is either a comma or semi-colon delimited list or, if the individual values are known not to contain spaces,[ii] a space-delimited list can be used. `<div class="inner greeting-box">Welcome!</div>`, where the attribute "class" has both the value "inner greeting-box" and also indicates the two [CSS](#) class names "inner" and "greeting-box".

XML declaration
> XML documents may begin with an *XML declaration* that describes some information about themselves. An example is `<?xml version="1.0" encoding="UTF-8"?>`.

Characters and escaping[[edit](#)]

XML documents consist entirely of characters from the [Unicode](#) repertoire. Except for a small number of specifically excluded [control characters](#), any character defined by Unicode may appear within the content of an XML document.

XML includes facilities for identifying the *encoding* of the Unicode characters that make up the document, and for expressing characters that, for one reason or another, cannot be used directly.

Valid characters[[edit](#)]

Unicode code points in the following ranges are valid in XML 1.0 documents:[10]

- U+0009 (Horizontal Tab), U+000A (Line Feed), U+000D (Carriage Return): these are the only [C0](#) controls accepted in XML 1.0;
- U+0020–U+D7FF, U+E000–U+FFFD: this excludes some non-characters in the [BMP](#) (all surrogates, U+FFFE and U+FFFF are forbidden);
- U+10000–U+10FFFF: this includes all code points in supplementary planes, including non-characters.

XML 1.1 extends the set of allowed characters to include all the above, plus the remaining characters in the range U+0001–U+001F.[11] At the same time, however, it restricts the use of C0 and [C1](#) control characters other than U+0009 (Horizontal Tab), U+000A (Line Feed), U+000D (Carriage Return), and U+0085 (Next Line) by requiring them to be written in escaped form (for example U+0001 must be written as `&#x01;` or its equivalent). In the case of C1 characters, this restriction is a backwards incompatibility; it was introduced to allow common encoding errors to be detected.

The code point [U+0000](#) (Null) is the only character that is not permitted in any XML 1.0 or 1.1 document.

Encoding detection[[edit](#)]

The Unicode character set can be encoded into bytes for storage or transmission in a variety of different ways, called "encodings". Unicode itself defines encodings that cover the entire repertoire; well-known ones include [UTF-8](#) and [UTF-16](#).[12] There are many

other text encodings that predate Unicode, such as [ASCII](#) and [ISO/IEC 8859](#); their character repertoires in almost every case are subsets of the Unicode character set.

XML allows the use of any of the Unicode-defined encodings and any other encodings whose characters also appear in Unicode. XML also provides a mechanism whereby an XML processor can reliably, without any prior knowledge, determine which encoding is being used.[13] Encodings other than UTF-8 and UTF-16 are not necessarily recognized by every XML parser.

Escaping[[edit](#)]

XML provides *[escape](#)* facilities for including characters that are problematic to include directly. For example:

- The characters "<" and "&" are key syntax markers and may never appear in content outside a [CDATA](#) section. It is allowed, but not recommended, to use "<" in XML entity values.[14]
- Some character encodings support only a subset of Unicode. For example, it is legal to encode an XML document in ASCII, but ASCII lacks code points for Unicode characters such as "é".
- It might not be possible to type the character on the author's machine.
- Some characters have [glyphs](#) that cannot be visually distinguished from other characters, such as the [non-breaking space](#) (` `) " " and the [space](#) (` `) " ", and the [Cyrillic capital letter A](#) (`А`) "А" and the [Latin capital letter A](#) (`A`) "A".

There are five [predefined entities](#):

- `<` represents "<";
- `>` represents ">";
- `&` represents "&";
- `'` represents "'";
- `"` represents '"'.

All permitted Unicode characters may be represented with a *[numeric character reference](#)*. Consider the Chinese character "中", whose numeric code in Unicode is hexadecimal 4E2D, or decimal 20,013. A user whose keyboard offers no method for entering this character could still insert it in an XML document encoded either as `中` or `中`. Similarly, the string "I <3 Jörg" could be encoded for inclusion in an XML document as `I <3 Jörg`.

`&#0;` is not permitted because the [null character](#) is one of the control characters excluded from XML, even when using a numeric character reference.[15] An alternative encoding mechanism such as [Base64](#) is needed to represent such characters.

[[edit](#)]

Comments may appear anywhere in a document outside other markup. Comments cannot appear before the XML declaration. Comments begin with `<!--` and end with `-->`. For compatibility with [SGML](#), the string "--" (double-hyphen) is not allowed inside comments;[16] this means comments cannot be nested. The ampersand has no special significance within comments, so entity and character references are not recognized as such, and there is no way to represent characters outside the character set of the document encoding.

An example of a valid comment: `<!--no need to escape <code> & such in comments-->`

International use[edit]

XML 1.0 (Fifth Edition) and XML 1.1 support the direct use of almost any [Unicode](#) character in element names, attributes, comments, character data, and processing instructions (other than the ones that have special symbolic meaning in XML itself, such as the less-than sign, "<"). The following is a well-formed XML document including [Chinese](#), [Armenian](#) and [Cyrillic](#) characters:

```
<?xml version="1.0" encoding="UTF-8"?>
<俄语 լեզու="ռուսերեն">данные</俄语>
```

Syntactical correctness and error-handling[edit]

The XML specification defines an XML document as a [well-formed](#) text, meaning that it satisfies a list of syntax rules provided in the specification. Some key points in the fairly lengthy list include:

- The document contains only properly encoded legal Unicode characters.
- None of the special syntax characters such as `<` and `&` appear except when performing their markup-delineation roles.
- The start-tag, end-tag, and empty-element tag that delimit elements are correctly nested, with [none missing](#) and none overlapping.
- Tag names are case-sensitive; the start-tag and end-tag must match exactly.
- Tag names cannot contain any of the characters !"#$%&'()*+,/;<=>?@[\]^`{|}~, nor a space character, and cannot begin with "-", ".", or a numeric digit.
- A single root element contains all the other elements.

The definition of an XML document excludes texts that contain violations of well-formedness rules; they are simply not XML. An XML processor that encounters such a violation is required to report such errors and to cease normal processing. This policy, occasionally referred to as "[draconian](#) error handling," stands in notable contrast to the behavior of programs that process [HTML](#), which are designed to produce a reasonable result even in the presence of severe markup errors.[17] XML's policy in this area has been criticized as a violation of [Postel's law](#) ("Be conservative in what you send; be liberal in what you accept").[18]

The XML specification defines a [valid XML document](#) as a [well-formed XML document](#) which also conforms to the rules of a [Document Type Definition](#) (DTD).[19][20]

Schemas and validation[edit]

In addition to being well-formed, an XML document may be *valid*. This means that it contains a reference to a [Document Type Definition](#) (DTD), and that its elements and attributes are declared in that DTD and follow the grammatical rules for them that the DTD specifies.

XML processors are classified as *validating* or *non-validating* depending on whether or not they check XML documents for validity. A processor that discovers a validity error must be able to report it, but may continue normal processing.

A DTD is an example of a *[schema](#)* or *grammar*. Since the initial publication of XML 1.0, there has been substantial work in the area of schema languages for XML. Such schema

languages typically constrain the set of elements that may be used in a document, which attributes may be applied to them, the order in which they may appear, and the allowable parent/child relationships.

Document type definition[edit]

The oldest schema language for XML is the document type definition (DTD), inherited from SGML.

DTDs have the following benefits:

- DTD support is ubiquitous due to its inclusion in the XML 1.0 standard.
- DTDs are terse compared to element-based schema languages and consequently present more information in a single screen.
- DTDs allow the declaration of standard public entity sets for publishing characters.
- DTDs define a *document type* rather than the types used by a namespace, thus grouping all constraints for a document in a single collection.

DTDs have the following limitations:

- They have no explicit support for newer features of XML, most importantly namespaces.
- They lack expressiveness. XML DTDs are simpler than SGML DTDs and there are certain structures that cannot be expressed with regular grammars. DTDs only support rudimentary datatypes.
- They lack readability. DTD designers typically make heavy use of parameter entities (which behave essentially as textual macros), which make it easier to define complex grammars, but at the expense of clarity.
- They use a syntax based on regular expression syntax, inherited from SGML, to describe the schema. Typical XML APIs such as SAX do not attempt to offer applications a structured representation of the syntax, so it is less accessible to programmers than an element-based syntax may be.

Two peculiar features that distinguish DTDs from other schema types are the syntactic support for embedding a DTD within XML documents and for defining *entities*, which are arbitrary fragments of text or markup that the XML processor inserts in the DTD itself and in the XML document wherever they are referenced, like character escapes.

DTD technology is still used in many applications because of its ubiquity.

Schema[edit]

A newer schema language, described by the W3C as the successor of DTDs, is XML Schema, often referred to by the initialism for XML Schema instances, XSD (XML Schema Definition). XSDs are far more powerful than DTDs in describing XML languages. They use a rich datatyping system and allow for more detailed constraints on an XML document's logical structure. XSDs also use an XML-based format, which makes it possible to use ordinary XML tools to help process them.

xs:schema element that defines a schema:

```
<?xml version="1.0" encoding="ISO-8859-1" ?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"></xs:schema>
```

RELAX NG[edit]

RELAX NG (Regular Language for XML Next Generation) was initially specified by OASIS and is now a standard (Part 2: *Regular-grammar-based validation* of ISO/IEC 19757 – DSDL). RELAX NG schemas may be written in either an XML based syntax or a more compact non-XML syntax; the two syntaxes are isomorphic and James Clark's conversion tool—Trang—can convert between them without loss of information. RELAX NG has a simpler definition and validation framework than XML Schema, making it easier to use and implement. It also has the ability to use datatype framework plug-ins; a RELAX NG schema author, for example, can require values in an XML document to conform to definitions in XML Schema Datatypes.

Schematron[edit]

Schematron is a language for making assertions about the presence or absence of patterns in an XML document. It typically uses XPath expressions. Schematron is now a standard (Part 3: *Rule-based validation* of ISO/IEC 19757 – DSDL).

DSDL and other schema languages[edit]

DSDL (Document Schema Definition Languages) is a multi-part ISO/IEC standard (ISO/IEC 19757) that brings together a comprehensive set of small schema languages, each targeted at specific problems. DSDL includes RELAX NG full and compact syntax, Schematron assertion language, and languages for defining datatypes, character repertoire constraints, renaming and entity expansion, and namespace-based routing of document fragments to different validators. DSDL schema languages do not have the vendor support of XML Schemas yet, and are to some extent a grassroots reaction of industrial publishers to the lack of utility of XML Schemas for publishing.

Some schema languages not only describe the structure of a particular XML format but also offer limited facilities to influence processing of individual XML files that conform to this format. DTDs and XSDs both have this ability; they can for instance provide the infoset augmentation facility and attribute defaults. RELAX NG and Schematron intentionally do not provide these.

[edit]

A cluster of specifications closely related to XML have been developed, starting soon after the initial publication of XML 1.0. It is frequently the case that the term "XML" is used to refer to XML together with one or more of these other technologies that have come to be seen as part of the XML core.

- XML namespaces enable the same document to contain XML elements and attributes taken from different vocabularies, without any naming collisions occurring. Although XML Namespaces are not part of the XML specification itself, virtually all XML software also supports XML Namespaces.
- XML Base defines the `xml:base` attribute, which may be used to set the base for resolution of relative URI references within the scope of a single XML element.
- XML Information Set or XML Infoset is an abstract data model for XML documents in terms of *information items*. The infoset is commonly used in the specifications of XML languages, for convenience in describing constraints on the XML constructs those languages allow.
- XSL (Extensible Stylesheet Language) is a family of languages used to transform and render XML documents, split into three parts:
- XSLT (XSL Transformations), an XML language for transforming XML documents into other XML documents or other formats such as HTML, plain text, or XSL-FO.

XSLT is very tightly coupled with XPath, which it uses to address components of the input XML document, mainly elements and attributes.

- [XSL-FO](#) (XSL Formatting Objects), an XML language for rendering XML documents, often used to generate PDFs.
- [XPath](#) (XML Path Language), a non-XML language for addressing the components (elements, attributes, and so on) of an XML document. XPath is widely used in other core-XML specifications and in programming libraries for accessing XML-encoded data.
- [XQuery](#) (XML Query) is an XML query language strongly rooted in XPath and XML Schema. It provides methods to access, manipulate and return XML, and is mainly conceived as a query language for [XML databases](#).
- [XML Signature](#) defines syntax and processing rules for creating [digital signatures](#) on XML content.
- [XML Encryption](#) defines syntax and processing rules for [encrypting](#) XML content.
- XML model (Part 11: *Schema Association* of [ISO/IEC 19757 – DSDL](#)) defines a means of associating any xml document with any of the schema types mentioned [above](#).

Some other specifications conceived as part of the "XML Core" have failed to find wide adoption, including [XInclude](#), [XLink](#), and [XPointer](#).

Programming interfaces[[edit](#)]

The design goals of XML include, "It shall be easy to write programs which process XML documents."[6] Despite this, the XML specification contains almost no information about how programmers might go about doing such processing. The [XML Infoset](#) specification provides a vocabulary to refer to the constructs within an XML document, but does not provide any guidance on how to access this information. A variety of [APIs](#) for accessing XML have been developed and used, and some have been standardized.

Existing APIs for XML processing tend to fall into these categories:

- Stream-oriented APIs accessible from a programming language, for example [SAX](#) and [StAX](#).
- Tree-traversal APIs accessible from a programming language, for example [DOM](#).
- [XML data binding](#), which provides an automated translation between an XML document and programming-language objects.
- Declarative transformation languages such as [XSLT](#) and [XQuery](#).
- Syntax extensions to general-purpose programming languages, for example [LINQ](#) and [Scala](#).

Stream-oriented facilities require less memory and, for certain tasks based on a linear traversal of an XML document, are faster and simpler than other alternatives. Tree-traversal and data-binding APIs typically require the use of much more memory, but are often found more convenient for use by programmers; some include declarative retrieval of document components via the use of XPath expressions.

XSLT is designed for declarative description of XML document transformations, and has been widely implemented both in server-side packages and Web browsers. XQuery overlaps XSLT in its functionality, but is designed more for searching of large [XML databases](#).

Simple API for XML[[edit](#)]

Simple API for XML (SAX) is a lexical, event-driven API in which a document is read serially and its contents are reported as callbacks to various methods on a handler object of the user's design. SAX is fast and efficient to implement, but difficult to use for extracting information at random from the XML, since it tends to burden the application author with keeping track of what part of the document is being processed. It is better suited to situations in which certain types of information are always handled the same way, no matter where they occur in the document.

Pull parsing[edit]

Pull parsing treats the document as a series of items read in sequence using the iterator design pattern. This allows for writing of recursive descent parsers in which the structure of the code performing the parsing mirrors the structure of the XML being parsed, and intermediate parsed results can be used and accessed as local variables within the functions performing the parsing, or passed down (as function parameters) into lower-level functions, or returned (as function return values) to higher-level functions.[21] Examples of pull parsers include Data::Edit::Xml in Perl, StAX in the Java programming language, XMLPullParser in Smalltalk, XMLReader in PHP, ElementTree.iterparse in Python, System.Xml.XmlReader in the .NET Framework, and the DOM traversal API (NodeIterator and TreeWalker).

A pull parser creates an iterator that sequentially visits the various elements, attributes, and data in an XML document. Code that uses this iterator can test the current item (to tell, for example, whether it is a start-tag or end-tag, or text), and inspect its attributes (local name, namespace, values of XML attributes, value of text, etc.), and can also move the iterator to the next item. The code can thus extract information from the document as it traverses it. The recursive-descent approach tends to lend itself to keeping data as typed local variables in the code doing the parsing, while SAX, for instance, typically requires a parser to manually maintain intermediate data within a stack of elements that are parent elements of the element being parsed. Pull-parsing code can be more straightforward to understand and maintain than SAX parsing code.

Document Object Model[edit]

Document Object Model (DOM) is an API that allows for navigation of the entire document as if it were a tree of node objects representing the document's contents. A DOM document can be created by a parser, or can be generated manually by users (with limitations). Data types in DOM nodes are abstract; implementations provide their own programming language-specific bindings. DOM implementations tend to be memory intensive, as they generally require the entire document to be loaded into memory and constructed as a tree of objects before access is allowed.

Data binding[edit]

XML data binding is the binding of XML documents to a hierarchy of custom and strongly typed objects, in contrast to the generic objects created by a DOM parser. This approach simplifies code development, and in many cases allows problems to be identified at compile time rather than run-time. It is suitable for applications where the document structure is known and fixed at the time the application is written. Example data binding systems include the Java Architecture for XML Binding (JAXB), XML Serialization in .NET Framework.[22] and XML serialization in gSOAP.

XML as data type[edit]

XML has appeared as a [first-class data type](#) in other languages. The [ECMAScript for XML](#) (E4X) extension to the [ECMAScript](#)/JavaScript language explicitly defines two specific objects (XML and XMLList) for JavaScript, which support XML document nodes and XML node lists as distinct objects and use a dot-notation specifying parent-child relationships.[23] E4X is supported by the [Mozilla](#) 2.5+ browsers (though now deprecated) and Adobe [Actionscript,](#) but has not been adopted more universally. Similar notations are used in Microsoft's [LINQ](#) implementation for Microsoft .NET 3.5 and above, and in [Scala](#) (which uses the Java VM). The open-source xmlsh application, which provides a Linux-like shell with special features for XML manipulation, similarly treats XML as a data type, using the <[ ]> notation.[24] The [Resource Description Framework](#) defines a data type `rdf:XMLLiteral` to hold wrapped, [canonical XML](#).[25] Facebook has produced extensions to the [PHP](#) and [JavaScript](#) languages that add XML to the core syntax in a similar fashion to E4X, namely [XHP](#) and [JSX](#) respectively.

History[[edit](#)]

XML is an application [profile](#) of SGML (ISO 8879).[26]

The versatility of SGML for dynamic information display was understood by early digital media publishers in the late 1980s prior to the rise of the Internet.[27][28] By the mid-1990s some practitioners of SGML had gained experience with the then-new [World Wide Web](#), and believed that SGML offered solutions to some of the problems the Web was likely to face as it grew. [Dan Connolly](#) added SGML to the list of W3C's activities when he joined the staff in 1995; work began in mid-1996 when [Sun Microsystems](#) engineer [Jon Bosak](#) developed a charter and recruited collaborators. Bosak was well connected in the small community of people who had experience both in SGML and the Web.[29]

XML was compiled by a [working group](#) of eleven members,[30] supported by a (roughly) 150-member Interest Group. Technical debate took place on the Interest Group mailing list and issues were resolved by consensus or, when that failed, majority vote of the Working Group. A record of design decisions and their rationales was compiled by [Michael Sperberg-McQueen](#) on December 4, 1997.[31] [James Clark](#) served as Technical Lead of the Working Group, notably contributing the empty-element `<empty />` syntax and the name "XML". Other names that had been put forward for consideration included "MAGMA" (Minimal Architecture for Generalized Markup Applications), "SLIM" (Structured Language for Internet Markup) and "MGML" (Minimal Generalized Markup Language). The co-editors of the specification were originally [Tim Bray](#) and [Michael Sperberg-McQueen](#). Halfway through the project Bray accepted a consulting engagement with [Netscape](#), provoking vociferous protests from Microsoft. Bray was temporarily asked to resign the editorship. This led to intense dispute in the Working Group, eventually solved by the appointment of Microsoft's [Jean Paoli](#) as a third co-editor.

The XML Working Group never met face-to-face; the design was accomplished using a combination of email and weekly teleconferences. The major design decisions were reached in a short burst of intense work between August and November 1996,[32] when the first Working Draft of an XML specification was published.[33] Further design work continued through 1997, and XML 1.0 became a [W3C](#) Recommendation on February 10, 1998.

Sources[[edit](#)]

XML is a profile of an ISO standard SGML, and most of XML comes from SGML unchanged. From SGML comes the separation of logical and physical structures (elements

and entities), the availability of grammar-based validation (DTDs), the separation of data and metadata (elements and attributes), mixed content, the separation of processing from representation ([processing instructions](#)), and the default angle-bracket syntax. The SGML declaration was removed; thus XML has a fixed delimiter set and adopts [Unicode](#) as the document [character set](#).

Other sources of technology for XML were the [TEI](#) (Text Encoding Initiative), which defined a profile of SGML for use as a "transfer syntax"; and [HTML](#), in which elements were synchronous with their resource, document character sets were separate from resource encoding, the `xml:lang` attribute was invented, and (like [HTTP](#)) metadata accompanied the resource rather than being needed at the declaration of a link. The ERCS(Extended Reference Concrete Syntax) project of the SPREAD (Standardization Project Regarding East Asian Documents) project of the ISO-related China/Japan/Korea Document Processing expert group was the basis of XML 1.0's naming rules; SPREAD also introduced hexadecimal numeric character references and the concept of references to make available all Unicode characters. To support ERCS, XML and HTML better, the SGML standard IS 8879 was revised in 1996 and 1998 with WebSGML Adaptations. The XML header followed that of ISO [HyTime](#).

Ideas that developed during discussion that are novel in XML included the algorithm for encoding detection and the encoding header, the processing instruction target, the xml:space attribute, and the new close delimiter for empty-element tags. The notion of well-formedness as opposed to validity (which enables parsing without a schema) was first formalized in XML, although it had been implemented successfully in the Electronic Book Technology "Dynatext" software;[34] the software from the University of Waterloo New Oxford English Dictionary Project; the RISP LISP SGML text processor at Uniscope, Tokyo; the US Army Missile Command IADS hypertext system; Mentor Graphics Context; Interleaf and Xerox Publishing System.

Versions[[edit](#)]

1.0 and 1.1[[edit](#)]

The first (XML 1.0) was initially defined in 1998. It has undergone minor revisions since then, without being given a new version number, and is currently in its fifth edition, as published on November 26, 2008. It is widely implemented and still recommended for general use.

The second (XML 1.1) was initially published on February 4, 2004, the same day as XML 1.0 Third Edition,[35] and is currently in its second edition, as published on August 16, 2006. It contains features (some contentious) that are intended to make XML easier to use in certain cases.[36] The main changes are to enable the use of line-ending characters used on [EBCDIC](#) platforms, and the use of scripts and characters absent from Unicode 3.2. XML 1.1 is not very widely implemented and is recommended for use only by those who need its particular features.[37]

Prior to its fifth edition release, XML 1.0 differed from XML 1.1 in having stricter requirements for characters available for use in element and attribute names and unique identifiers: in the first four editions of XML 1.0 the characters were exclusively enumerated using a specific version of the [Unicode](#) standard (Unicode 2.0 to Unicode 3.2.) The fifth edition substitutes the mechanism of XML 1.1, which is more future-proof but reduces [redundancy](#). The approach taken in the fifth edition of XML 1.0 and in all editions of XML 1.1 is that only certain characters are forbidden in names, and everything else is allowed to accommodate suitable name characters in future Unicode versions. In the fifth

edition, XML names may contain characters in the [Balinese](), [Cham](), or [Phoenician]() scripts among many others added to Unicode since Unicode 3.2.[36]

Almost any Unicode code point can be used in the character data and attribute values of an XML 1.0 or 1.1 document, even if the character corresponding to the code point is not defined in the current version of Unicode. In character data and attribute values, XML 1.1 allows the use of more [control characters]() than XML 1.0, but, for "robustness", most of the control characters introduced in XML 1.1 must be expressed as numeric character references (and #x7F through #x9F, which had been allowed in XML 1.0, are in XML 1.1 even required to be expressed as numeric character references[38]). Among the supported control characters in XML 1.1 are two line break codes that must be treated as whitespace characters, which are the only control codes that can be written directly.

2.0[[edit]()]

There has been discussion of an XML 2.0, although no organization has announced plans for work on such a project. XML-SW (SW for [skunkworks]()), which one of the original developers of XML has written,[39] contains some proposals for what an XML 2.0 might look like, including elimination of DTDs from syntax, as well as integration of [XML namespaces](), [XML Base]() and [XML Information Set]() into the base standard.

Binary XML[[edit]()]

The World Wide Web Consortium also has an XML Binary Characterization Working Group doing preliminary research into use cases and properties for a binary encoding of XML Information Set. The working group is not chartered to produce any official standards. Since XML is by definition text-based, ITU-T and ISO are using the name [Fast Infoset]() for their own binary format (ITU-T Rec. X.891 and ISO/IEC 24824-1) to avoid confusion.

Criticism[[edit]()]

XML and its extensions have regularly been criticized for verbosity, complexity and redundancy.[40]

Mapping the basic tree model of XML to [type systems]() of programming languages or databases can be difficult, especially when XML is used for exchanging highly structured data between applications, which was not its primary design goal. However, [XML data binding]() systems allow applications to access XML data directly from objects representing a [data structure]() of the data in the programming language used, which ensures [type safety](), rather than using the [DOM]() or [SAX]() to retrieve data from a direct representation of the XML itself. This is accomplished by automatically creating a mapping between elements of the XML schema [XSD]() of the document and members of a class to be represented in memory.

Other criticisms attempt to refute the claim that XML is a [self-describing]() language[41] (though the XML specification itself makes no such claim).

[JSON](), [YAML](), and [S-Expressions]() are frequently proposed as simpler alternatives (see [Comparison of data serialization formats]())[42] that focus on representing highly structured data rather than documents, which may contain both highly structured and relatively unstructured content. However, W3C standardized XML schema specifications offer a broader range of structured [XSD]() data types compared to simpler serialization formats and offer modularity and reuse through [XML namespaces]().

See also[edit]

- List of XML markup languages
- List of XML schemas
- Extensible programming
- Comparison of layout engines (XML)
- Comparison of data serialization formats
- Binary XML
- EBML
- WBXML
- XML Protocol

Notes[edit]

1. ^ i.e., embedded quote characters would be a problem
2. ^ A common example of this is CSS class or identifier names.

References[edit]

1. ^ *"XML Media Types, RFC 7303"*. *Internet Engineering Task Force. July 2014.*
2. ^ *"XML 1.0 Specification"*. *World Wide Web Consortium. Retrieved 22 August 2010.*
3. ^ *"Extensible Markup Language (XML) 1.0". www.w3.org.*
4. ^ *"XML and Semantic Web W3C Standards Timeline" (PDF). Dblab.ntua.gr. Retrieved 14 August 2016.*
5. ^ *"W3C DOCUMENT LICENSE". W3.org. Retrieved 24 July 2020.*
6. ^ Jump up to: *a b "XML 1.0 Origin and Goals". W3.org. Retrieved 14 August 2016.*
7. ^ *Fennell, Philip (June 2013). "Extremes of XML". XML London 2013: 80–86. doi:10.14337/XMLLondon13.Fennell01. ISBN 978-0-9926471-0-0.*
8. ^ Jump up to: *a b c d e Dykes, Lucinda (2005). XML for Dummies (4th ed.). Hoboken, N.J.: Wiley. ISBN 978-0-7645-8845-7.*
9. ^ *"XML Applications and Initiatives". Xml.coverages.org. Retrieved 16 November 2017.*
10. ^ *"Extensible Markup Language (XML) 1.0 (Fifth Edition)". World Wide Web Consortium. 2008-11-26. Retrieved 23 November 2012.*
11. ^ *"Extensible Markup Language (XML) 1.1 (Second Edition)". World Wide Web Consortium. Retrieved 22 August 2010.*
12. ^ *"Characters vs. Bytes". Tbray.org. Retrieved 16 November 2017.*
13. ^ *"Autodetection of Character Encodings". W3.org. Retrieved 16 November 2017.*
14. ^ *"Extensible Markup Language (XML) 1.0 (Fifth Edition)". W3.org. Retrieved 16 November 2017.*
15. ^ *"W3C I18N FAQ: HTML, XHTML, XML and Control Codes". W3.org. Retrieved 16 November 2017.*
16. ^ *"Extensible Markup Language (XML)". W3.org. Retrieved 16 November 2017.* Section "Comments"
17. ^ *Pilgrim, Mark (2004). "The history of draconian error handling in XML". Archived from the original on 2011-07-26. Retrieved 18 July 2013.*
18. ^ *"There are No Exceptions to Postel's Law [dive into mark]". DiveIntoMark.org. Archived from the original on 2011-05-14. Retrieved 22 April 2013.*
19. ^ *"XML Notepad". Xmlnotepad/codeplex.com. Retrieved 16 November 2017.*
20. ^ *"XML Notepad 2007". Microsoft.com. Retrieved 16 November 2017.*
21. ^ *DuCharme, Bob. "Push, Pull, Next!". Xml.com. Retrieved 16 November 2017.*
22. ^ *"XML Serialization in the .NET Framework". Msdn.microsoft.com. Retrieved 31*

July 2009.

23. ^ _"Processing XML with E4X"_. _Mozilla Developer Center. Mozilla Foundation._

24. ^ _"XML Shell: Core Syntax"_. _Xmlsh.org. 2010-05-13. Retrieved 22 August 2010._

25. ^ _"Resource Description Framework (RDF): Concepts and Abstract Syntax"_. _W3.org. Retrieved 22 August 2010._

26. ^ _"ISO/IEC 19757-3". ISO/IEC. 1 June 2006: vi._

27. ^ _Bray, Tim (February 2005). "A conversation with Tim Bray: Searching for ways to tame the world's vast stores of information". Association for Computing Machinery's "Queue site". Retrieved 16 April 2006._

28. ^ _Ambron, Sueann & Hooper, Kristina, eds. (1988). "Publishers, multimedia, and interactivity". Interactive multimedia. Cobb Group. ISBN 1-55615-124-1._

29. ^ _Eliot Kimber (2006). "XML is 10". Drmacros-xml-rants.blogspot.com. Retrieved 16 November 2017._

30. ^ The working group was originally called the "Editorial Review Board." The original members and seven who were added before the first edition was complete, are listed at the end of the first edition of the XML Recommendation, at http://www.w3.org/TR/1998/REC-xml-19980210.

31. ^ _"Reports From the W3C SGML ERB to the SGML WG And from the W3C XML ERB to the XML SIG"_. _W3.org. Retrieved 31 July 2009._

32. ^ _"Oracle Technology Network for Java Developers - Oracle Technology Network - Oracle"_. _Java.sun.com. Retrieved 16 November 2017._

33. ^ _"Extensible Markup Language (XML)"_. _W3.org. 1996-11-14. Retrieved 31 July 2009._

34. ^ _Jon Bosak; Sun Microsystems (2006-12-07). "Closing Keynote, XML 2006". 2006.xmlconference.org. Archived from the original on 2007-07-11. Retrieved 31 July 2009._

35. ^ _"Extensible Markup Language (XML) 1.0 (Third Edition)"_. _W3.org. Retrieved 22 August 2010._

36. ^ Jump up to: _**a b**_ _"Extensible Markup Language (XML) 1.1 (Second Edition) , Rationale and list of changes for XML 1.1"_. _W3.org. Retrieved 20 January 2012._

37. ^ _Harold, Elliotte Rusty (2004). Effective XML. Addison-Wesley. pp. 10–19. ISBN 0-321-15040-6._

38. ^ _"Extensible Markup Language (XML) 1.1 (Second Edition)"_. _W3.org. Retrieved 22 August 2010._

39. ^ _Bray, Tim (10 February 2002). "Extensible Markup Language, SW (XML-SW)"_.

40. ^ _"XML: The Angle Bracket Tax"_. _Codinghorror.com. 11 May 2008. Retrieved 16 November 2017._

41. ^ _"The Myth of Self-Describing XML" (PDF). Workflow.HealthBase.info. September 2003. Retrieved 16 November 2017._

42. ^ _"What usable alternatives to XML syntax do you know?"_. _StackOverflow.com. Retrieved 16 November 2017._

Further reading[edit]

- Annex A of ISO 8879:1986 (SGML)
- _Lawrence A. Cunningham (2005). "Language, Deals and Standards: The Future of XML Contracts". Washington University Law Review. SSRN 900616._
- _Bosak, Jon; Bray, Tim (May 1999). "XML and the Second-Generation Web". Scientific American. **280** (5): 89. Bibcode:1999SciAm.280e..89B. doi:10.1038/scientificamerican0599-89. Archived from the original on 1 October 2009._
- _Kelly, Sean (February 6, 2006). "Making Mistakes with XML". Developer.com. Retrieved 26 October 2010._
- _St. Laurent, Simon (February 12, 2003). "Five Years Later, XML." O'Reilly XML_

Blog. *O'Reilly Media*. Retrieved 26 October 2010.

- *"W3C XML is Ten!"*. *World Wide Web Consortium*. 12 February 2008. Retrieved 26 October 2010.
- *"Introduction to XML"* (PDF). Course Slides. *Pierre Geneves*. October 2012. Archived from the original on 2015-10-16.`{{cite web}}`: CS1 maint: bot: original URL status unknown (link)

External links[edit]

	Wikimedia Commons has media related to XML.

- W3C XML homepage
- XML 1.0 Specification
- Retrospective on Extended Reference Concrete Syntax by Rick Jelliffe
- *XML, Java and the Future of the Web* (1997) by Jon Bosak
- http://validator.w3.org/ The Official [W3C] Markup Validation Service
- The XML FAQ originally for the W3C's XML SIG by Peter Flynn