[Template:Stack](/wiki/Template:Stack" \o "Template:Stack)

**Extensible Markup Language** (**XML**) is a [markup language](/wiki/Markup_language) that defines a set of rules for encoding documents in a [format](/wiki/File_format) that is both [human-readable](/wiki/Human-readable_medium) and [machine-readable](/wiki/Machine-readable_data). It is defined by the [W3C's](/wiki/World_Wide_Web_Consortium) XML 1.0 Specification[[1]](#cite_note-1) and by several other related specifications,[[2]](#cite_note-2) all of which are free [open standards](/wiki/Open_standard).[[3]](#cite_note-3) The design goals of XML emphasize simplicity, generality and usability across the [Internet](/wiki/Internet).[[4]](#cite_note-4) It is a textual data format with strong support via [Unicode](/wiki/Unicode) for different [human languages](/wiki/Language). Although the design of XML focuses on documents, it is widely used for the representation of arbitrary [data structures](/wiki/Data_structure)[[5]](#cite_note-5) such as those used in [web services](/wiki/Web_service).

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

## Contents

* 1 Applications of XML[[edit](/index.php?title=(none)&action=edit&section=1)]
* 2 Key terminology[[edit](/index.php?title=(none)&action=edit&section=2)]
* 3 Characters and escaping[[edit](/index.php?title=(none)&action=edit&section=3)]
  + 3.1 Valid characters[[edit](/index.php?title=(none)&action=edit&section=4)]
  + 3.2 Encoding detection[[edit](/index.php?title=(none)&action=edit&section=5)]
  + 3.3 Escaping[[edit](/index.php?title=(none)&action=edit&section=6)]
  + 3.4 Comments[[edit](/index.php?title=(none)&action=edit&section=7)]
  + 3.5 International use[[edit](/index.php?title=(none)&action=edit&section=8)]
* 4 Well-formedness and error-handling[[edit](/index.php?title=(none)&action=edit&section=9)]
* 5 Schemas and validation[[edit](/index.php?title=(none)&action=edit&section=10)]
  + 5.1 Document Type Definition[[edit](/index.php?title=(none)&action=edit&section=11)]
  + 5.2 XML Schema[[edit](/index.php?title=(none)&action=edit&section=12)]
  + 5.3 RELAX NG[[edit](/index.php?title=(none)&action=edit&section=13)]
  + 5.4 Schematron[[edit](/index.php?title=(none)&action=edit&section=14)]
  + 5.5 ISO DSDL and other schema languages[[edit](/index.php?title=(none)&action=edit&section=15)]
* 6 Related specifications[[edit](/index.php?title=(none)&action=edit&section=16)]
* 7 Programming interfaces[[edit](/index.php?title=(none)&action=edit&section=17)]
  + 7.1 Simple API for XML[[edit](/index.php?title=(none)&action=edit&section=18)]
  + 7.2 Pull parsing[[edit](/index.php?title=(none)&action=edit&section=19)]
  + 7.3 Document Object Model[[edit](/index.php?title=(none)&action=edit&section=20)]
  + 7.4 Data binding[[edit](/index.php?title=(none)&action=edit&section=21)]
  + 7.5 XML as data type[[edit](/index.php?title=(none)&action=edit&section=22)]
* 8 History[[edit](/index.php?title=(none)&action=edit&section=23)]

## Applications of XML[[edit](/index.php?title=(none)&action=edit&section=1)]

[Template:As of](/wiki/Template:As_of), hundreds of document formats using XML syntax have been developed,[[6]](#cite_note-6) including [RSS](/wiki/RSS), [Atom](/wiki/Atom_(standard)), [SOAP](/wiki/SOAP), and [XHTML](/wiki/XHTML). XML-based formats have become the default for many office-productivity tools, including [Microsoft Office](/wiki/Microsoft_Office) ([Office Open XML](/wiki/Office_Open_XML)), [OpenOffice.org](/wiki/OpenOffice.org) and [LibreOffice](/wiki/LibreOffice) ([OpenDocument](/wiki/OpenDocument)), and [Apple's](/wiki/Apple_Computer) [iWork](/wiki/IWork). XML has also been employed as the base language for [communication protocols](/wiki/Communication_protocol), such as [XMPP](/wiki/Extensible_Messaging_and_Presence_Protocol). Applications for the [Microsoft](/wiki/Microsoft) [.NET Framework](/wiki/.NET_Framework) use XML files for configuration. Apple has an implementation of a registry based on XML.[[7]](#cite_note-7) XML has come into common use for the interchange of data over the Internet. [IETF](/wiki/History_of_the_Internet#Internet_Engineering_Task_Force) RFC 7303 gives rules for the construction of [Internet Media Types](/wiki/Internet_media_type) for use when sending XML. It also defines the media types *application/xml* and *text/xml*, which say only that the data is in XML, and nothing about its [semantics](/wiki/Semantics). The use of *text/xml* has been criticized as a potential source of encoding problems and it has been suggested that it should be deprecated.[[8]](#cite_note-8) RFC 7303 also recommends that XML-based languages be given media types ending in *+xml*; for example *image/svg+xml* for [SVG](/wiki/SVG).

Further guidelines for the use of XML in a networked context may be found in RFC 3470, also known as IETF BCP 70, a document covering many aspects of designing and deploying an XML-based language.

## Key terminology[[edit](/index.php?title=(none)&action=edit&section=2)]

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.

(Unicode) character

By definition, an XML document is a string of characters. Almost every legal [Unicode](/wiki/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](/wiki/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 markup construct that begins with < and ends with >. Tags come in three flavors:

* *start-tags*; for example: <section>
* *end-tags*; for example: </section>
* *empty-element tags*; for example: [Template:Code](/wiki/Template:Code)

Element

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- and end-tags, if any, are the element's *content*, and may contain markup, including other elements, which are called *child elements*. An example of an element is <Greeting>Hello, world.</Greeting>. Another is <line-break />.

Attribute

A markup construct consisting of a name/value pair that exists within a start-tag or empty-element tag. In the example (below) the element *img* has two attributes, *src* and *alt*:

<source lang="xml"><img src="madonna.jpg" alt='Foligno Madonna, by Raphael' /></source> Another example would be

<source lang="xml"><step number="3">Connect A to B.</step></source> where the name of the attribute is "number" and the 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[[note 1]](#cite_note-9) 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,[[note 2]](#cite_note-10) a space-delimited list can be used. :<source lang="xml">

Hello!

</source> where the attribute "class" has both the value "inner greeting-box" and also indicates the two [CSS](/wiki/CSS) class names "inner" and "greeting-box".

XML declaration

XML documents may begin by declaring some information about themselves, as in the following example:

<source lang="xml"><?xml version="1.0" encoding="UTF-8"?></source>

## Characters and escaping[[edit](/index.php?title=(none)&action=edit&section=3)]

XML documents consist entirely of characters from the [Unicode](/wiki/Unicode) repertoire. Except for a small number of specifically excluded [control characters](/wiki/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](/index.php?title=(none)&action=edit&section=4)]

[Template:Main](/wiki/Template:Main) Unicode code points in the following ranges are valid in XML 1.0 documents:[[9]](#cite_note-11)\* U+0009 (Horizontal Tab), U+000A (Line Feed), U+000D (Carriage Return): these are the only [C0](/wiki/C0_and_C1_control_codes) controls accepted in XML 1.0;

* U+0020–U+D7FF, U+E000–U+FFFD: this excludes *some* (not all) non-characters in the [BMP](/wiki/Basic_Multilingual_Plane) (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[[10]](#cite_note-12) extends the set of allowed characters to include all the above, plus the remaining characters in the range U+0001–U+001F. At the same time, however, it restricts the use of C0 and [C1](/wiki/C0_and_C1_control_codes) 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](/wiki/Null_character) (Null) is the only character that is not permitted in any XML 1.0 or 1.1 document.

### Encoding detection[[edit](/index.php?title=(none)&action=edit&section=5)]

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](/wiki/UTF-8) and [UTF-16](/wiki/UTF-16).[[11]](#cite_note-13) There are many other text encodings that predate Unicode, such as [ASCII](/wiki/ASCII) and [ISO/IEC 8859](/wiki/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.[[12]](#cite_note-14) Encodings other than UTF-8 and UTF-16 are not necessarily recognized by every XML parser.

### Escaping[[edit](/index.php?title=(none)&action=edit&section=6)]

XML provides [*escape*](/wiki/Escape_sequence) 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](/wiki/CDATA) section.[[13]](#cite_note-15)\* 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](/wiki/Homoglyph) that cannot be visually distinguished from other characters: examples are
  + [non-breaking space](/wiki/Non-breaking_space) (&#xa0;) " "

compare space (&#x20;) " "

* + [Cyrillic Capital Letter A](/wiki/А) (&#x410;) "А"

compare [Latin Capital Letter A](/wiki/A) (&#x41;) "A"

There are five [predefined entities](/wiki/List_of_XML_and_HTML_character_entity_references#Predefined_entities_in_XML):

* &lt; represents "<"
* &gt; represents ">"
* &amp; represents "&"
* &apos; represents "'"
* &quot; represents '"'

All permitted Unicode characters may be represented with a [*numeric character reference*](/wiki/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 &#20013; or &#x4e2d;. Similarly, the string "I <3 Jörg" could be encoded for inclusion in an XML document as "I &lt;3 J&#xF6;rg".

"&#0;" is not permitted, however, because the [null character](/wiki/Null_character) is one of the control characters excluded from XML, even when using a numeric character reference.[[14]](#cite_note-16) An alternative encoding mechanism such as [Base64](/wiki/Base64) is needed to represent such characters.

### Comments[[edit](/index.php?title=(none)&action=edit&section=7)]

Comments may appear anywhere in a document outside other markup. Comments cannot appear before the XML declaration. Comments start with "<!--" and end with "-->". For compatibility with SGML, the string "--" (double-hyphen) is not allowed inside comments;[[15]](#cite_note-17) 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](/index.php?title=(none)&action=edit&section=8)]

[Template:ChineseText](/wiki/Template:ChineseText)[Template:CyrillicText](/wiki/Template:CyrillicText) XML 1.0 (Fifth Edition) and XML 1.1 support the direct use of almost any [Unicode](/wiki/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](/wiki/Chinese_character), [Armenian](/wiki/Armenian_alphabet) and [Cyrillic](/wiki/Cyrillic_script) characters: <source lang="xml"> <?xml version="1.0" encoding="UTF-8"?> <俄语 լեզու="ռուսերեն">данные</俄语> </source>

## Well-formedness and error-handling[[edit](/index.php?title=(none)&action=edit&section=9)]

[Template:Main](/wiki/Template:Main) The XML specification defines an XML document as a [well-formed](/wiki/Well-formed_element) 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 begin, end, and empty-element tags that delimit the elements are correctly nested, with none missing and none overlapping
* The element tags are case-sensitive; the beginning and end tags must match exactly.
* Tag names cannot contain any of the characters !"#$%&'()\*+,/;<=>?@[\]^`{|}~, nor a space character, and cannot start 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](/wiki/Draco_(lawgiver)) error handling," stands in notable contrast to the behavior of programs that process [HTML](/wiki/HTML), which are designed to produce a reasonable result even in the presence of severe markup errors.[[16]](#cite_note-18) XML's policy in this area has been criticized as a violation of [Postel's law](/wiki/Postel's_law) ("Be conservative in what you send; be liberal in what you accept").[[17]](#cite_note-19) The XML specification defines a [valid XML document](/wiki/XML_validation) as a [well-formed XML document](/wiki/Well-formed_XML_document) which also conforms to the rules of a [Document Type Definition](/wiki/Document_Type_Definition) (DTD).

## Schemas and validation[[edit](/index.php?title=(none)&action=edit&section=10)]

[Template:Unreferenced section](/wiki/Template:Unreferenced_section) In addition to being well-formed, an XML document may be *valid*. This means that it contains a reference to a [Document Type Definition](/wiki/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*](/wiki/XML_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](/index.php?title=(none)&action=edit&section=11)]

[Template:Main](/wiki/Template:Main) The oldest schema language for XML is the [Document Type Definition](/wiki/Document_Type_Definition) (DTD), inherited from [SGML](/wiki/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](/wiki/SGML_entity) 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](/wiki/Feature_(software_design)) of XML, most importantly [namespaces](/wiki/XML_Namespace).
* 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](/wiki/Macro_(computer_science))), which make it easier to define complex grammars, but at the expense of clarity.
* They use a syntax based on [regular expression](/wiki/Regular_expression) syntax, inherited from [SGML](/wiki/SGML), to describe the schema. Typical XML APIs such as [SAX](/wiki/Simple_API_for_XML) 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 and/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.

### XML Schema[[edit](/index.php?title=(none)&action=edit&section=12)]

[Template:Main](/wiki/Template:Main) A newer schema language, described by the W3C as the successor of DTDs, is [XML Schema](/wiki/XML_Schema_(W3C)), often referred to by the [initialism](/wiki/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](/wiki/Data_type) 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:

<source lang="xml"><?xml version="1.0" encoding="ISO-8859-1" ?> <xs:schema xmlns:xs="[http://www.w3.org/2001/XMLSchema">](http://www.w3.org/2001/XMLSchema%22%3e) </xs:schema></source>

### RELAX NG[[edit](/index.php?title=(none)&action=edit&section=13)]

[RELAX NG](/wiki/RELAX_NG) was initially specified by [OASIS](/wiki/OASIS_(organization)) and is now also an ISO/IEC International Standard (as part of [DSDL](/wiki/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](/wiki/Isomorphic) and [James Clark's](/wiki/James_Clark_(programmer)) conversion tool - '[Trang](http://www.thaiopensource.com/relaxng/trang.html)', 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](/wiki/Datatype) framework [plug-ins](/wiki/Plug-in_(computing)); a RELAX NG schema author, for example, can require values in an XML document to conform to definitions in XML Schema Datatypes.

### Schematron[[edit](/index.php?title=(none)&action=edit&section=14)]

[*Schematron*](/wiki/Schematron) is a language for making [assertions](/wiki/Assertion_(computing)) about the presence or absence of patterns in an XML document. It typically uses [XPath](/wiki/XPath) expressions.

### ISO DSDL and other schema languages[[edit](/index.php?title=(none)&action=edit&section=15)]

The ISO [*DSDL*](/wiki/DSDL) (Document Schema Description Languages) standard brings together a comprehensive set of small schema languages, each targeted at specific problems. DSDL includes [RELAX NG](/wiki/RELAX_NG) full and compact syntax, [Schematron](/wiki/Schematron) assertion language, and languages for defining datatypes, character repertoire constraints, renaming and entity expansion, and namespace-based [routing](/wiki/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](/wiki/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](/wiki/Infoset) augmentation facility and attribute defaults. RELAX NG and Schematron intentionally do not provide these.

## Related specifications[[edit](/index.php?title=(none)&action=edit&section=16)]

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](/wiki/XML_Namespace) enable the same document to contain XML elements and attributes taken from different vocabularies, without any [naming collisions](/wiki/Naming_collision) occurring. Although XML Namespaces are not part of the XML specification itself, virtually all XML software also supports XML Namespaces.
* [XML Base](/wiki/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.
* The [XML Information Set](/wiki/XML_Information_Set) or *XML infoset* describes 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.
* xml:id Version 1.0 asserts that an attribute named xml:id functions as an "ID attribute" in the sense used in a DTD.
* [XPath](/wiki/XPath) defines a syntax named *XPath expressions*, which identifies one or more of the internal components (elements, attributes, and so on) included in an XML document. XPath is widely used in other core-XML specifications and in programming libraries for accessing XML-encoded data.
* [XSLT](/wiki/XSLT) is a language with an XML-based syntax that is used to transform XML documents into other XML documents, HTML, or other, unstructured formats such as plain text or RTF. XSLT is very tightly coupled with XPath, which it uses to address components of the input XML document, mainly elements and attributes.
* [XSL Formatting Objects](/wiki/XSL_Formatting_Objects), or XSL-FO, is a markup language for XML document formatting most often used to generate [PDFs](/wiki/PDF).
* [XQuery](/wiki/XQuery) is an XML-oriented 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](/wiki/XML_database).
* [XML Signature](/wiki/XML_Signature) defines syntax and processing rules for creating [digital signatures](/wiki/Digital_signature) on XML content.
* [XML Encryption](/wiki/XML_Encryption) defines syntax and processing rules for [encrypting](/wiki/Encryption) XML content.

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

## Programming interfaces[[edit](/index.php?title=(none)&action=edit&section=17)]

The design goals of XML include, "It shall be easy to write programs which process XML documents."[[4]](#cite_note-4) Despite this, the XML specification contains almost no information about how programmers might go about doing such processing. The [XML Infoset](/wiki/XML_Infoset) specification provides a vocabulary to refer to the constructs within an XML document, but also does not provide any guidance on how to access this information. A variety of [APIs](/wiki/API) 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](/wiki/Simple_API_for_XML) and [StAX](/wiki/StAX).
* Tree-traversal APIs accessible from a programming language, for example [DOM](/wiki/DOM_(XML_API)).
* [XML data binding](/wiki/XML_data_binding), which provides an automated translation between an XML document and programming-language objects.
* Declarative transformation languages such as [XSLT](/wiki/XSLT) and [XQuery](/wiki/XQuery).

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](/wiki/XML_database).

### Simple API for XML[[edit](/index.php?title=(none)&action=edit&section=18)]

[*Simple API for XML*](/wiki/Simple_API_for_XML) (SAX) is a [lexical](/wiki/Lexical_analysis), [event-driven](/wiki/Event-driven_programming) interface in which a document is read serially and its contents are reported as [callbacks](/wiki/Callback_(computer_science)) to various [methods](/wiki/Method_(computer_science)) on a [handler object](/wiki/Event_handler) 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](/index.php?title=(none)&action=edit&section=19)]

*Pull parsing*[[18]](#cite_note-20) treats the document as a series of items read in sequence using the [iterator design pattern](/wiki/Iterator_pattern). This allows for writing of [recursive descent parsers](/wiki/Recursive_descent_parser) 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 methods performing the parsing, or passed down (as method parameters) into lower-level methods, or returned (as method return values) to higher-level methods. Examples of pull parsers include [StAX](/wiki/StAX) in the [Java](/wiki/Java_(programming_language)) programming language, XMLPullParser in [Smalltalk](/wiki/Smalltalk), XMLReader in [PHP](/wiki/PHP), ElementTree.iterparse in [Python](/wiki/Python_(programming_language)), System.Xml.XmlReader in the [.NET Framework](/wiki/.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 or end element, or text), and inspect its attributes (local name, [namespace](/wiki/XML_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](/index.php?title=(none)&action=edit&section=20)]

The [*Document Object Model*](/wiki/Document_Object_Model) (DOM) is an [interface](/wiki/User_interface)-oriented [application programming interface](/wiki/Application_programming_interface) that allows for navigation of the entire document as if it were a tree of [node](/wiki/Node_(computer_science)) [objects](/wiki/Object_(computer_science)) 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](/wiki/Language_binding). DOM implementations tend to be [memory](/wiki/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](/index.php?title=(none)&action=edit&section=21)]

Another form of XML processing API is [*XML data binding*](/wiki/XML_data_binding), where XML data are made available as a hierarchy of custom, strongly typed classes, in contrast to the generic objects created by a [Document Object Model](/wiki/Document_Object_Model) parser. This approach simplifies code development, and in many cases allows problems to be identified at compile time rather than run-time. Example data binding systems include the [Java Architecture for XML Binding](/wiki/Java_Architecture_for_XML_Binding) (JAXB) and XML Serialization in .NET.[[19]](#cite_note-21)

### XML as data type[[edit](/index.php?title=(none)&action=edit&section=22)]

XML has appeared as a [first-class data type](/wiki/First-class_data_type) in other languages. The [ECMAScript for XML](/wiki/E4X) (E4X) extension to the [ECMAScript](/wiki/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.[[20]](#cite_note-22) E4X is supported by the [Mozilla](/wiki/Mozilla) 2.5+ browsers (though now deprecated) and Adobe [Actionscript](/wiki/Actionscript), but has not been adopted more universally. Similar notations are used in Microsoft's [LINQ](/wiki/LINQ) implementation for Microsoft .NET 3.5 and above, and in [Scala](/wiki/Scala_(programming_language)) (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.[[21]](#cite_note-23) The [Resource Description Framework](/wiki/Resource_Description_Framework) defines a data type rdf:XMLLiteral to hold wrapped, [canonical XML](/wiki/Canonical_XML).[[22]](#cite_note-24) Facebook has produced extensions to the [PHP](/wiki/PHP) and [JavaScript](/wiki/JavaScript) languages that add XML to the core syntax in a similar fashion to E4X, namely [XHP](/wiki/XHP) and [JSX](/wiki/React_(JavaScript_library)#JSX) respectively.

## History[[edit](/index.php?title=(none)&action=edit&section=23)]

XML is an application profile of [SGML](/wiki/SGML) (ISO 8879).[[23]](#cite_note-25) The versatility of [SGML](/wiki/SGML) for dynamic information display was understood by early digital media publishers in the late 1980s prior to the rise of the Internet.[[24]](#cite_note-26)[[25]](#cite_note-27) By the mid-1990s some practitioners of SGML had gained experience with the then-new [World Wide Web](/wiki/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](/wiki/Dan_Connolly_(computer_scientist)) 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](/wiki/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.[[26]](#cite_note-28) XML was compiled by a [working group](/wiki/Working_group) of eleven members,[[27]](#cite_note-29)