



The XML FAQ

Frequently-Asked Questions about the
Extensible Markup Language

v4.0 (2005-01-01)

Peter Flynn (ed.)

Contents

Summary	iii
Current revision	v
Legal stuff	v
Acknowledgements	vi
A General questions	1
A.1 What is XML?	1
A.2 What is XML for?	1
A.3 What is SGML?	1
A.4 What is HTML?	2
A.5 Aren't XML, SGML, and HTML all the same thing?	2
A.6 Who is responsible for XML?	3
A.7 Why is XML such an important development?	3
A.8 Why not just carry on extending HTML?	3
A.9 Why do we need all this SGML stuff? Why not just use Word or Notes?	4
A.10 Where do I find more information about XML?	4
A.11 Where can I discuss implementation and development of XML?	5
A.12 What is the difference between XML and C or C++?	6
B Existing users of SGML (including HTML: everyone who browses the Web)	7
B.1 What do I have to do to use XML?	7
B.2 Should I use XML instead of HTML?	7
B.3 Where can I get an XML browser?	8
B.4 Do I have to switch from SGML or HTML to XML?	10
B.5 Can I use XML for ordinary office applications?	10
C Authors of SGML (including writers of HTML: Web page owners)	12
C.1 Does XML replace HTML?	12
C.2 Do I have to know HTML or SGML before I learn XML?	12
C.3 What does an XML document look like inside?	12
C.4 How does XML handle white-space in my documents?	13
C.5 Which parts of an XML document are case-sensitive?	14
C.6 How can I make my existing HTML files work in XML?	15
C.7 Is there an XML version of HTML?	17
C.8 If XML is just a subset of SGML, can I use XML files directly with existing SGML tools?	18
C.9 I'm used to authoring and serving HTML. Can I learn XML easily?	18
C.10 Can XML use non-Latin characters?	18
C.11 What's a Document Type Definition (DTD) and where do I get one?	19
C.12 Does XML let me make up my own tags?	20
C.13 How do I create my own DTD?	21
C.14 Can a root element type be explicitly declared in the DTD?	22

C.15	I keep hearing about alternatives to DTDs. What's a Schema?	22
C.16	How do I get XML into or out of a database?	23
C.17	How will XML affect my document links?	24
C.18	Can I do mathematics using XML?	25
C.19	How does XML handle metadata?	25
C.20	Can I use JavaScript, ActiveX, etc in XML files?	25
C.21	Can I use Java to create or manage XML files?	26
C.22	How do I execute or run an XML file?	26
C.23	How do I control formatting and appearance?	27
C.24	How do I use graphics in XML?	28
C.25	What is parsing and how do I do it in XML?	30
C.26	How do I include one XML file in another?	32
C.27	When should I use a CDATA Marked Section (aka 'Can I embed HTML in XML')?	32
D	Developers and Implementors (including WebMasters and server operators)	33
D.1	Where's the spec?	33
D.2	What are these terms DTDless, valid, and well-formed?	33
D.3	Which should I use in my DTD, attributes or elements?	36
D.4	What else has changed between SGML and XML?	37
D.5	What's a namespace?	37
D.6	What XML software is available?	38
D.7	What's my information? DATA or TEXT?	38
D.8	Do I have to change any of my server software to work with XML? .	39
D.9	Can I still use server-side inclusions?	40
D.10	Can I (and my authors) still use client-side inclusions?	40
D.11	I'm trying to understand the XML Spec: why does it have such difficult terminology?	41
D.12	I have to do an overview of XML for my manager/client/investor/advisor. What should I mention?	41
D.13	Is there a conformance test suite for XML processors?	43
D.14	I've already got SGML DTDs: how do I convert them for use with XML?	43
D.15	How do I include one DTD (or fragment) in another?	44
D.16	What's the story on XML and EDI?	45
E	References	46
E.1	Bibliography	46
E.2	How far are we going?	47
E.3	Revision history	47

Summary

This is the list of Frequently-Asked Questions about the Extensible Markup Language (XML). It is restricted to questions about XML: if you are seeking answers to questions about SGML, HTML, CGI scripts, Java, databases, or penguins, you may find some pointers, but you should probably look elsewhere as well. It is intended as a first resource for users, developers, and the interested reader. It does not form part of the XML Specification.

Organisation

§ This FAQ was originally maintained on behalf of the World Wide Web Consortium's XML Special Interest Group. It is divided into four sections: General [p.1], Users [p.7], Authors [p.12], and Developers [p.33]. The questions are numbered independently within each section. As the numbering may change with each version, comments and suggestions should refer to the version number (see the revision history [p.v]) as well as the section and question number.

Please submit bug reports, suggestions for improvement, and other comments about *this FAQ only* to the editor¹. Questions and comments about XML should go to the relevant mailing list or newsgroup [p.5]. Comments about the XML Specification [p.33] itself and related specifications should be directed to the W3C².

Updates

- ¶ Paragraphs which have been *added* since the last version are indicated with a pilcrow.
- § Paragraphs which have been *changed* since the last version are indicated with a section sign.
- ± Paragraphs *marked for deletion* but retained at the moment for information are indicated with a plus/minus sign.

Availability

This XML document is at <http://www.ucc.ie/xml/>. It is XML served auto-converted to HTML by Cocoon, so what you read online is HTML in your browser.

§ </> You can download the unconverted file³ (avoiding the .xml filetype which over-enthusiastic browsers want to usurp);

§ </> the DTD⁴ is a lightly modified version of DocBook⁵;

¹xmlfaq@silmaril.ie

²<http://www.w3.org/>

³<http://www.ucc.ie/xml/faq.sgml>

⁴<http://www.ucc.ie/xml/faq.dtd>

⁵<http://www.docbook.org/>

- § </> there are XSL stylesheets for serving as HTML⁶ and for converting to L^AT_EX⁷ to make the PDF and PostScript versions;
- § </> the preconverted HTML version is at <http://www.ucc.ie/xml/faq.html> and this has been used to save a copy in OpenOffice⁸, Microsoft Word⁹, and plaintext¹⁰ formats.
- </> A notification of the new versions is posted periodically to the `comp.text.xml` Usenet newsgroup and to the XML-L¹¹ mailing list for the archives.
 - </> for printed copies there are versions for A4 PostScript¹², A4 PDF¹³, Letter PostScript¹⁴ and Letter PDF¹⁵ available. Viewers can be downloaded for PostScript¹⁶ and PDF file formats (GhostView¹⁷, Xpdf¹⁸, Adobe Acrobat Reader¹⁹);
 - </> WAP (if anyone's still using it), OEB (eBook), and cHTML versions have been proposed for your handheld devices, and I'm open to offers if anyone wants to write the code.

The FAQ is also available in carbon-based toner on flattened dead trees by sending €10 (\$15 or equivalent in any convertible currency) to the editor²⁰ (email first to check rates and postal address).

You can download the XML logo as a GIF²¹, JPG²², or EPS²³ file; and an icon for your file system in ICO²⁴ (Microsoft Windows), Mac²⁵, or XPM²⁶ (X Window system) format.

Translations

Those I know about are in:

- </> German²⁷ (partial translation of some questions) [Karin Driesen];
- </> Amharic²⁸ [Abass Alamnehe];
- </> Japanese²⁹ [Murata Makoto];

⁶<http://www.ucc.ie/xml/webfaq.xsl>

⁷<http://www.ucc.ie/xml/printfaq.xsl>

⁸<http://www.ucc.ie/xml/faq.sxw>

⁹<http://www.ucc.ie/xml/faq.doc>

¹⁰<http://www.ucc.ie/xml/faq.txt>

¹¹<http://listserv.heanet.ie/xml-l.html>

¹²<http://www.ucc.ie/xml/faq.a4.ps>

¹³<http://www.ucc.ie/xml/faq.a4.pdf>

¹⁴http://www.ucc.ie/xml/faq_letter.ps

¹⁵http://www.ucc.ie/xml/faq_letter.pdf

¹⁶<http://www.cs.wisc.edu/~ghost/gsview/>

¹⁷<http://www.cs.wisc.edu/~ghost/gsview/>

¹⁸<http://www.foolabs.com/xpdf/>

¹⁹<http://www.adobe.com/products/acrobat/readstep.html>

²⁰peter@silmaril.ie

²¹<http://www.ucc.ie/xml/xmllogo.gif>

²²<http://www.ucc.ie/xml/xmllogo.jpg>

²³<http://www.ucc.ie/xml/xmllogo.eps>

²⁴<http://www.ucc.ie/xml/xml.ico>

²⁵http://www.ucc.ie/xml/xml_folder_icon.sit.hqx

²⁶<http://www.ucc.ie/xml/xml.xpm>

²⁷http://www.oreilly.de/xml/xml_faq_fragen.html

²⁸http://www.senamirmir.com/xml/faq/xml_faq_amh.html

- </> Spanish³⁰ (currently inaccessible) [Jaime Sagarduy];
- </> Korean³¹ (currently inaccessible). [Kangchan Lee];
- </> Chinese³² (currently inaccessible) [Neko]. Also in Chinese³³ (also inaccessible) [Jiang Luqin];
- </> French³⁴ [Jacques André];
- </> Czech³⁵ [Miloslav Nic].

I would be grateful if the translators of those copies which have become inaccessible would contact me with the new URL.

Current revision

Earlier: 0.0 0.1 0.2 0.3 0.4 0.5 1.0 1.1 1.2 1.3 1.4 1.5 1.6 2.0 2.1 3.0 3.01 3.02 (details on p.47).

4.0 (2005-01-01)

Went back to DocBook³⁶ markup using qandaset³⁷ instead of the QAML that has been used for the last two major releases. Revised text in most sections for clarity in wording, and recast some now-established explanatory material into the past tense. Added new dates for 2005. Added explicit references to the GNU FDL in the legal section. Took the tip on types of XML out into a new question [p.38], and added new questions on file inclusions [p.32] and the use of CDATA Marked Sections [p.32].

Legal stuff

This document is copyright © 1996–2005 by the editor and released under the terms of the GNU Free Documentation License (see below). Quotations of the contributions of others remain copyright of the individual contributors. You may copy and distribute this document in any form provided you acknowledge this source (and the individual in the case of a fragment) and don't try to pretend you wrote it yourself. If you want to republish or reprint it in bulk, or copy it on a web site, ask the editor first to make sure you get the right edition, to make provision for updating, and to ensure you use the correct legal wording.

'Permission is granted to copy, distribute and/or modify this document under the terms of the GNU Free Documentation License, Version 1.2 or any later version published by the Free Software Foundation; with no Invariant Sections, no Front-Cover Texts, and no Back-Cover Texts. A copy of the license is available here³⁸. You are allowed to distribute,

²⁹<http://www.fxis.co.jp/DMS/sgml/cafe/library/etc/xmlfaq.html>

³⁰<http://slug.ctv.es/~olea/sgml-esp/xfaq15.html>

³¹<http://xml.t2000.co.kr/faq/index.html>

³²<http://zxd.webjump.com/xml.html>

³³http://weblab.crema.unimi.it/xmlzh/XML_FAQ.htm

³⁴<http://www.gutenberg.eu.org/pub/GUTenberg/publications/HTML/FAQXML/faqxml-fr.html>

³⁵http://zvon.vscht.cz/ZvonHTML/Translations/xmlFAQ/front_all.html

³⁶<http://www.docbook.org/>

³⁷<http://www.docbook.org/tdg/en/html/qandaset.html>

reproduce, and modify it without fee or further requirement for consent subject to the conditions in the section on Modifications³⁹.'

The editor and authors have asserted their right to be identified as the editor and authors of this document.

Acknowledgements

The following people helped with the original contributions, plus many other members of the W3C XML SIG as well as FAQ readers around the world.

Terry Allen, Tom Borgman, Tim Bray, Robin Cover, Bob DuCharme, Christopher Maden, Eve Maler, Makoto Murata, Peter Murray-Rust, Liam Quin, Michael Sperberg-McQueen, Joel Weber .

³⁸<http://www.ctan.org/tex-archive/info/beginlatex/html/appendixA.html#gfdl>

³⁹<http://www.ctan.org/tex-archive/info/beginlatex/html/appendixA.html#gfdl-4>

A General questions

A.1 What is XML?

XML is the Extensible Markup Language. It is designed to improve the functionality of the Web by providing more flexible and adaptable information identification.

The Extensible Markup Language, more flexible than HTML

It is called extensible because it is not a fixed format like HTML (a single, predefined markup language). Instead, XML is actually a metalanguage—a language for describing other languages—which lets you design your own customized markup languages for limitless different types of documents. XML can do this because it's written in SGML [p.1], the international standard metalanguage for text markup systems (ISO 8879).

A.2 What is XML for?

XML is intended 'to make it easy and straightforward to use SGML on the Web: easy to define document types, easy to author and manage SGML-defined documents, and easy to transmit and share them across the Web.'

Making SGML technology usable on the Web

It defines 'an extremely simple dialect of SGML which is completely described in the XML Specification. 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.'

(Quotes are from the XML specification [p.33]). XML is not just for Web pages: it can be used to store any kind of structured document, and to enclose or encapsulate information in order to pass it between different computing systems which would otherwise be unable to communicate.

A.3 What is SGML?

SGML is the Standard Generalized Markup Language (ISO 8879:1985⁴⁰), the international standard for defining descriptions of the structure of different types of electronic document. There is an SGML FAQ at <http://lamp.man.deakin.edu.au/sgml/sgmlfaq.txt> which is posted every month to the `comp.text.sgml` newsgroup, and the SGML Web pages are at <http://xml.coverpages.org/sgml>.

Standard Generalized Markup Language, ISO 8879:1995

SGML is very large, powerful, and complex. It has been in heavy industrial and commercial use for nearly two decades, and there is a significant body of expertise and software to go with it. XML is a lightweight cut-down version of SGML which keeps enough of its functionality to make it useful but removes all the optional features which made SGML too complex to program for in a Web environment.

⁴⁰<http://www.iso.ch/>

Note

ISO standards like SGML are governed by the International Organization for Standardization in Geneva, Switzerland, and voted into or out of existence by representatives from every country's national standards body.

If you have a query about an international standard, you should contact your national standards body for the name of your country's representative on the relevant ISO committee or working group.

If you have a query about your country's representation in Geneva or about the conduct of your national standards body, you should contact the relevant government department in your country, or speak to your public representative.

The representation of countries at the ISO is not a matter for this FAQ. Please do not submit queries to the editor about how or why your country's ISO representatives have or have not voted on a specific standard.

A.4 What is HTML?

§ HTML is the HyperText Markup Language⁴¹ (RFC 1866⁴²), which started as a small application of SGML [p.1] used on the Web.

§ It defines a very simple class of report-style documents, with section headings, paragraphs, lists, tables, and illustrations, with a few informational but very few presentational elements⁴³, plus some hypertext and multimedia. See the question on extending HTML [p.3]. The current recommendation is to use the XML version, XHTML.

HyperText Markup Language, RFC 1866, the language of Web pages.

A.5 Aren't XML, SGML, and HTML all the same thing?

Not quite; SGML [p.1] is the mother tongue, and has been used for describing thousands of different document types in many fields of human activity, from transcriptions of ancient Irish manuscripts⁴⁴ to the technical documentation for stealth bombers⁴⁵, and from patients' clinical records⁴⁶ to musical notation⁴⁷. SGML is very large and complex, however, and probably overkill for most common desktop applications.

XML is an abbreviated version of SGML, to make it easier for you to define your own document types, and to make it easier for programmers to write programs to handle them. It omits all the options, and most of the more complex and less-used parts of SGML in return for the benefits of being easier to write applications for, easier to understand, and more suited to delivery and interoperability over the Web. But it is still SGML, and XML files may still be processed in the same way as any other SGML file (see the question on XML software [p.38]).

HTML [p.1] is just one of many SGML or XML applications, the one most frequently used on the Web.

No, SGML and XML are met-alanguages. HTML is an application of them.

⁴¹<http://www.w3.org/MarkUp>

⁴²<http://ds.internic.net/rfc/rfc1866.txt>

⁴³Flynn Making more use of markup.

⁴⁴<http://www.ucc.ie/celt/>

⁴⁵<http://web.deskbook.osd.mil/>

⁴⁶<http://www.hl7.org>

⁴⁷<http://www.tecno.com/smdl.htm>

Technical readers may find it more useful to think of XML as being SGML— rather than HTML++.

A.6 Who is responsible for XML?

The W3C

XML is a project of the World Wide Web Consortium (W3C)⁴⁸, and the development of the specification is supervised by an XML Working Group. A Special Interest Group of co-opted contributors and experts from various fields contributed comments and reviews by email.

§ XML is a public format: it is not a proprietary development of any company, although the membership of the WG and the SIG represented companies as well as industrial research and academic institutions. The v1.0 specification [p.33] was accepted by the W3C as a Recommendation on Feb 10, 1998.

A.7 Why is XML such an important development?

It removes two constraints which were holding back Web developments:

1. dependence on a single, inflexible document type (HTML [p.2]) which was being much abused for tasks it was never designed for;
2. the complexity of full SGML [p.1], whose syntax allows many powerful but hard-to-program options.

It overcomes the inflexibility of HTML and the complexity of SGML

XML allows the flexible development of user-defined document types. It provides a robust, non-proprietary, persistent, and verifiable file format for the storage and transmission of text and data both on and off the Web; and it removes the more complex options of SGML, making it easier to program for.

A.8 Why not just carry on extending HTML?

§ HTML [p.2] was already overburdened with dozens of interesting but incompatible inventions from different manufacturers, because it provides only one way of describing your information.

HTML is already too overburdened with proprietary add-ons.

XML allows groups of people or organizations to create their own customized markup applications for exchanging information in their domain (music, chemistry, electronics, hill-walking, finance, surfing, petroleum geology, linguistics, cooking, knitting, stellar cartography, history, engineering, rabbit-keeping, mathematics [p.25], genealogy⁴⁹, etc).

§ HTML is now well beyond the limit of its usefulness as a way of describing information, and while it will continue to play an important role for the content it currently represents, many new applications require a more robust and flexible infrastructure.

A.9 Why do we need all this SGML stuff? Why not just use Word or Notes?

§ Information on a network which connects many different types of computer has to be usable on all of them. Public information in particular cannot afford to be restricted to one make or model or manufacturer, or to cede control of its data format to private hands. It is also helpful for such information to be in a form that can be reused in many different ways, as this will minimize wasted time and effort. Proprietary data formats⁵⁰, no matter how well documented or publicized, are simply not an option: their control still resides in private hands and they can be changed or withdrawn arbitrarily without notice.

Restrictive
proprietary
data formats
are unsuitable
for durable
public
information

§ SGML [p.1] is the international standard for defining this kind of application, and was therefore the natural choice for XML, but those who need an alternative based on different software for other purposes are entirely free to implement similar services using such a system, especially if they are for private use.

A.10 Where do I find more information about XML?

Online, there's the XML Specification [p.33] and ancillary documentation available from the W3C; Robin Cover's SGML/XML Web pages⁵¹ with an extensive list of online reference material and links to software; and a summary⁵² and condensed FAQ⁵³ from Tim Bray.

At
<http://xml.coverpages.org>

§ For offline resources, see the lists of books, articles, and software for XML in Robin Cover's SGML and XML Web pages⁵⁴. That site should always be your first port of call.

§ The events listed below are the ones I have been told about. Please mail me⁵⁵ if you come across others.

§ </> The annual XML Conferences are run in North America and Europe by IDEAlliance⁵⁶ (formerly the GCA). In 2005, XML Europe⁵⁷ (now known as XTech) is in Amsterdam on 25–27 May, and the US XML Conference⁵⁸ is in Atlanta on 14–18 November (note that like 2004 this is a month earlier than in previous years).

§ </> The Extreme Markup Languages⁵⁹ conference (also an IDEAlliance event) takes place on 2–6 August 2005 in Montréal.

§ </> The annual XML Summer School⁶⁰, organised by CSW⁶¹, takes place in Wadham College, Oxford on 26–30 July 2005 (win a place!⁶²).

⁴⁸<http://www.w3.org/>

⁴⁹<http://users.iclway.co.uk/mhkay/gedml/index.html>

⁵⁰<http://www.ucc.ie:8080/cocoon/cc/docs/markup>

⁵¹<http://xml.coverpages.org/>

⁵²<http://www.textuality.com/xml/>

⁵³<http://www.textuality.com/xml/faq.html>

⁵⁴<http://xml.coverpages.org/sgml-xml.html>

⁵⁵xmlfaq@silmaril.ie

⁵⁶<http://www.idealliance.org>

⁵⁷<http://www.xtech-conference.org/>

⁵⁸<http://www.xmlconference.org/xmlusa/>

⁵⁹<http://www.extrememarkup.com/extreme/>

⁶⁰<http://www.xmlsummerschool.com/>

⁶¹<http://www.csw.co.uk>

⁶²<http://www.xmlsummerschool.com/competition.htm>

There are many other XML events around the world: most of them are announced on the mailing lists and newsgroups [p.5].

A.11 Where can I discuss implementation and development of XML?

On mailing lists and in Usenet newsgroups

The two principal online media are Usenet newsgroups and mailing lists. See the question *Where do I find more information about XML?* [p.4] for details of conferences.

- </> The newsgroups are `comp.text.xml`⁶³ and to a certain extent `comp.text.sgml`⁶⁴. Ask your Internet Provider for access to these, or use a Web interface like Google Groups⁶⁵.
- </> The general-purpose mailing list for public discussion is XML-L: to subscribe, visit the Web site⁶⁶ and click on the link to join. You can also access the XML-L archives from the same URL.
- </> For those developing software components for XML there is an xml-dev mailing list. You can subscribe by sending a 1-line mail message to `xml-dev-request@lists.xml.org` saying just SUBSCRIBE. The xml-dev archives are at OASIS⁶⁷. Note that this list is for those people actively involved in developing resources for XML. It is not for general information about XML (use the XML-L list above for that).
- </> There is the XSL-List for discussing XSL (both XSLT and XSL:FO). For details of how to subscribe, see <http://www.mulberrytech.com/xsl/xsl-list>.

Andrew Watt writes:

There is a mailing list specifically for XSL-FO only, on eGroups.com⁶⁸. You can subscribe by sending a message to `XSL-FO-subscribe@egroups.com`.

Warning

This is the Yahoo XSL-FO list. Be aware that it sends out regular automated spam to non-members claiming that they have asked to join.

Gianni Rubagotti writes:

A new Italian mailing list about XML is born: to subscribe, send a mail message without a subject line but with text saying `subscribeXML-IT` to `majordomo@ananas.usr.dsi.unimi.it`. Everyone, Italian or not, who wants to debate about XML in our tongue is welcome.

Gianni also runs the Humanities XML List⁶⁹.

⁶³news:comp.text.xml

⁶⁴news:comp.text.sgml

⁶⁵<http://groups.google.com/>

⁶⁶<http://listserv.heanet.ie/xml-l.html>

⁶⁷<http://lists.xml.org/archives/xml-dev/>

⁶⁸<http://www.egroups.com/group/XSL-FO>

⁶⁹<http://groups.yahoo.com/group/x-humanities/>

J-P Theberge writes:

A French mailing list about XML has been created. To subscribe, send `subscribe` to `xml-request@trisome.com`.

Mailing lists

When you join a mailing list you will be sent details of how to use it. Please Read The Fine Documentation because it contains important information, particularly about what to do if your company or ISP changes your email address.

Please note that there is a lot of inaccurate and misleading information published in print and on the Web about subscribing to and unsubscribing from mailing lists. Don't guess: Read The Fine Documentation.

A.12 What is the difference between XML and C or C++?

C and C++ (and other languages like FORTRAN, or Pascal, or BASIC, or Java or hundreds more) are *programming languages* with which you specify calculations, actions, and decisions to be carried out in order:

```
mod curconfig[if left(date,6) = "01-Apr", t.put "April Fool!",
  f.put days('31102005', 'DDMMYYYY')-days(sdate, 'DDMMYYYY')
  " more shopping days to Samhain"];
```

C is for writing programs;
XML is for storing text.

XML is a markup specification language with which you can design ways of describing information (text or data), usually for storage, transmission, or processing by a program. It says nothing about what you should do with the data (although your choice of element names may hint at what they are for):

```
<part num="DA42" models="LS AR DF HG KJ" update="2001-11-22">
  <name>Camshaft end bearing retention circlip</name>
  <image drawing="RR98-dh37" type="SVG" x="476" y="226"/>
  <maker id="RQ778">Ringtown Fasteners Ltd</maker>
  <notes>Angle-nosed insertion tool <tool id="GH25"/> is
    required for the removal and replacement of this
    part.</notes>
</part>
```

§ On its own, an SGML or XML file (including HTML) doesn't do anything. It's a data format which just sits there until you run a program which does something with it. See also the question about how to run or execute XML files [p.26].

B Existing users of SGML (including HTML: everyone who browses the Web)

B.1 What do I have to do to use XML?

§ For the average user of the Web, nothing except use a browser which works with XML (see the question about browsers [p.8]). Remember some XML components are still being invented or implemented, so some features are still either undefined or have yet to be written. Don't expect everything to work faultlessly yet!

To read it: an XML browser (eg Firefox or IE). To create: an XML editor (Emacs, Spy, etc).

You can use XML browsers to look at some of the stable XML material, such as Jon Bosak's Shakespeare plays⁷⁰ and the molecular experiments of the Chemical Markup Language (CML)⁷¹. There are some more example sources listed at <http://xml.coverpages.org/xml.html#examples>, and you will find XML (particularly in the disguise of XHTML [p.17]) being introduced in places where it won't break older browsers.

If you want to start preparations for creating your own XML files, see the questions in the Authors' Section [p.12] and the Developers' Section [p.33].

B.2 Should I use XML instead of HTML?

§ Yes, if you need robustness, accuracy, and persistence. XML allows authors and providers to design their own document markup [p.19] instead of being limited to HTML. Document types can be explicitly tailored to an application, so the cumbersome fudging and poodlefaking that has to take place with HTML [p.2] becomes a thing of the past: your markup can always say what it means (trivial example:

Yes if you need robustness, accuracy, and persistence.

```
<date yyyyymmdd="2005-12-26">next Monday</date>).
```

- </> Information content can be richer and easier to use, because the descriptive and hypertext linking abilities of XML [p.24] are much greater than those available in HTML.
- </> XML can provide more and better facilities for browser presentation and performance, using CSS and XSLT stylesheets;
- </> It removes many of the underlying complexities of SGML in favor of a more flexible model, so writing programs to handle XML is much easier than doing the same for full SGML.
- </> Information becomes more accessible and reusable, because the more flexible markup of XML can be used by any XML software instead of being restricted to specific manufacturers as has become the case with HTML.
- </> Valid XML files [D.2, p.35] are still SGML, so they can be used outside the Web as well, in existing document-handling environments.

¶ If your information is transient, or completely static *and* unreferenced, or very short and simple, and unlikely to need updating, HTML may be all you need.

⁷⁰<ftp://sunsite.unc.edu/pub/sun-info/standards/xml/eg/>

⁷¹<http://www.xml-cml.org>

B.3 Where can I get an XML browser?

MSIE 5.5 or 6.*;
Mozilla Firefox
0.9.6 up

Mike Brown writes:

The concept of 'browsing' is primarily the result of HTML having the semantics that it does. In an HTML document there are sections of text called anchors that are 'hyperlinked' to other documents that might be at remote locations on a network or filesystem. HTML documents provide cues to a web browser regarding how the document should be displayed and what kind of behaviors are expected of the browser when the user interacts with it. The HTML specification provides many suggestions and requirements for the browser, and provides specific meanings for many different examples of markup, such as the fact that a `` element refers to an image that should be retrieved by the browser and rendered inline with the adjacent text.

Unlike HTML, XML does not have such inherent semantics at all. There is no prescribed method for rendering XML documents. Therefore, what it means to 'browse' XML is open to interpretation. For example, an XML document describing the characteristics of a machine part does not carry any information about how that information should be presented to a user. An application is free to use the data to produce an image of the part, generate a formatted text listing of the information, display the XML document's markup with a pretty color scheme, or restructure the data into a format for storage in a database, transmission over a network, or input to another program.

However, despite the fact that XML documents are purely descriptive data files, it is possible to 'browse' them in a sense, by rendering them with stylesheets. A stylesheet is a separate document that provides hints and algorithms for rendering or transforming the data in the XML document. HTML users may be familiar with Cascading Style Sheets (CSS). The CSS stylesheet language is general and powerful enough to be applied to XML documents, although it is oriented toward visual rendering of the document and does not allow for complex processing of the document's data. By associating an XML document with a CSS stylesheet, it may be possible to load an XML document in a CSS-aware web browser, and the browser may be able to provide some kind of rendering of it, even if the browser does not otherwise know how to read and process XML documents. However, not all web browsers will load an XML document correctly, and they are not required to recognize the XML markup that associates the document with a stylesheet, so one cannot assume that XML documents can be opened with just any web browser.

A more complex and powerful stylesheet language (p.27) is XSLT, the Transformations part of the Extensible Stylesheet Language, which can be used to transform XML to other formats, including HTML, other forms of XML, and plain text. If the output of this transformation is HTML, it can be viewed in a web browser as any other HTML document would.

The degree of support for XML and stylesheets in web browsers varies greatly. Although loading and rendering XML in the browser is possible in some cases, it is not universally supported. Therefore, much XML content on the web is translated to HTML on the servers. It is this generated HTML that is delivered to the browsers. Most of Microsoft⁷²'s web site, for example, exists as XML that is converted to HTML on the fly. The web browser never knows the difference.

⁷²<http://www.microsoft.com>

Current state of existing browser support for XML (1 Jan 2005):

- § </> Microsoft Internet Explorer 5.0 and 5.5 handle XML, processing it by default using a built-in stylesheet written in a Microsoft-specific, obsolete predecessor of XSLT called XSL (not to be confused with the real XSLT). The output of the stylesheet is DHTML, which, when rendered in the browser, shows a coloured, syntax-highlighted version of the XML document, with collapsible views. If the XML document references a stylesheet, that stylesheet will be used instead, within the limitations of MSIE's incomplete implementation of CSS. MSIE 5.0 and 5.5 can also use stylesheets in another obsolete syntax called WD-xsl, which should be avoided. These versions can be upgraded to support real XSLT: see the MSXML FAQ⁷³. MSIE 6.0 and up use real XSLT 1.0, but can use both the obsolete syntaxes as well.
- § </> Mozilla Firefox⁷⁴ 0.9 up and Netscape 6 and 7 (there is no Netscape 5) both have full XML support with XSLT and CSS. In general, Firefox, is more robust than MSIE, and provides better standards adherence.
- § </> The authors of the former MultiDoc Pro SGML browser, CITEC⁷⁵ (whose engine was also used in Panorama and other SGML browsers), joined forces with Mozilla to produce a multi-everything browser called DocZilla, which reads HTML, XML, and SGML, with XSLT and CSS stylesheets. This runs under Windows and Linux and is currently at release 1.0. See <http://www.doczilla.com> for details. This is by far the most ambitious browser project, and is backed by solid SGML and XML expertise.
- </> Opera⁷⁶ supports XML and CSS on MS-Windows and Linux. The browser size is tiny by comparison with the others, but features are good and the speed is excellent, although the earlier slavish insistence on mimicking everything old (pre-Mozilla) Netscape did, especially the bugs, still shows through in places.
- </> Don't use Netscape 4.* or Internet Explorer 4.* or earlier, or early versions of Mozilla if you want XML support: they don't have it. Upgrade to Firefox⁷⁷ as soon as possible.

¶ I have no information on the XML capabilities of the new (OS/X) Mac browser (Safari), which is based on the Gecko engine, like Firefox. I also have no information on what the new Konqueror does (in KDE under Fedora Core 3, for example). Anyone with experience of these please mail the editor⁷⁸.

See also the notes on software for authors [p.38] and XML for developers [p.38], and the more detailed list on the XML pages in the SGML Web site at <http://xml.coverpages.org/>.

B.4 Do I have to switch from SGML or HTML to XML?

Not if you don't want to

No, existing SGML and HTML applications software will continue to work with existing files. But as with any enhanced facility, if you want to view or download and use XML files, you will need to use XML-aware software. There is much more being developed for XML than there ever was for SGML, so a lot of users are moving.

B.5 Can I use XML for ordinary office applications?

Yes, use Star Office, Open Office, WordPerfect, or even MS-Office (11/XP only).

Yes, most office productivity suites already do this, and there are more on the way:

- </> Sun's Star Office⁷⁹ and its Open Source fork, OpenOffice⁸⁰, have been saving their files as XML by default for a couple of years. Both comprise a wordprocessor, spreadsheet, presentation software, and a vector drawing package, and share the same DTD/Schema.
- </> Corel's WordPerfect⁸¹ suite has shipped with a fully-fledged XML editor for several years (which also does full SGML as well). It can save the formatted result as a Microsoft Word .doc file, but it uses its own stylesheet technology to format documents, not XSLT or CSS.
- </> Microsoft Office⁸² (Office-11 for XP) extends the existing 'Save As... XML' to all parts of the suite except Powerpoint, although XML is not the default format (yet). Office-11 supports other W3C Schemas (not DTDs) as well as its own, and provides a method for binding element types to Word's named styles (like Microsoft's earlier product SGML Author for Word⁸³ did).
- </> Avoid Microsoft's 'Works' package, as it is incompatible both with Office and with XML.
- </> I have no information on Lotus products. They don't seem to be interested, so neither am I.

Sun and the OpenOffice team have made the DTD/Schema used in Star Office⁸⁴ and Open Office⁸⁵ publicly available for use by any office application and development by the OASIS Open Office XML Formats TC⁸⁶. It has recently been suggested⁸⁷ that it should be proposed as the basis for a new International Standard for office documents.

⁷³<http://www.netcrucible.com/xslt/msxml-faq.htm>

⁷⁴<http://www.mozilla.org/>

⁷⁵<http://www.citec.fi/>

⁷⁶<http://www.opera.com/opera5/specs.html>

⁷⁷<http://www.mozilla.org/>

⁷⁸xmlfaq@silmaril.ie

⁷⁹<http://www.sun.com/>

⁸⁰<http://www.openoffice.org/>

⁸¹<http://www.corel.com/servlet/Satellite?pagename=Corel2/Products/Home&pid=1047022958453>

⁸²<http://www.microsoft.com/presspass/press/2002/Oct02/10-25XMLArchitectMA.asp>

⁸³<http://www.microsoft.com/catalog/display.asp?subid=38&site=723>

⁸⁴<http://www.sun.com/>

⁸⁵<http://www.openoffice.org/>

⁸⁶<http://www.oasis-open.org/committees/office/>

⁸⁷<http://tbray.org/ongoing/When/200x/2004/09/24/SmartEC>

There is more detail under XML File Formats for Office Documents⁸⁸ in the XML Cover Pages which briefly describes and points to further information on: GNOME Office, KOffice, Microsoft XDocs, OASIS TC for Open Office XML File Format, 1DOK.org Project, and OpenOffice.org XML File Format.

⁸⁸<http://xml.coverpages.org/xmlFileFormats.html>

C Authors of SGML (including writers of HTML: Web page owners)

C.1 Does XML replace HTML?

No.

No. XML itself does not replace HTML. Instead, it provides an alternative which allows you to define your own set of markup elements. HTML is expected to remain in common use for some time to come, and the current version of HTML is in XML syntax [p.17]. XML is designed to make the writing of DTDs much simpler than with full SGML. (See the question on DTDs [p.19] for what one is and why you might want one.)

C.2 Do I have to know HTML or SGML before I learn XML?

No, but it's useful.

No, although it's useful because a lot of XML terminology and practice derives from nearly two decades' experience of SGML.

Be aware that 'knowing HTML' is not the same as 'understanding SGML'. Although HTML was written as an SGML application, browsers ignore most of it (which is why so many useful things don't work), so just because something is done a certain way in HTML browsers does not mean it's correct, least of all in XML.

C.3 What does an XML document look like inside?

Pointy brackets like HTML

The basic structure is very similar to most other applications of SGML, including HTML. XML documents can be very simple, with straightforward nested markup of your own design:

```
<?xml version="1.0" standalone="yes"?>
<conversation>
  <greeting>Hello, world!</greeting>
  <response>Stop the planet, I want to get off!</response>
</conversation>
```

Or they can be more complicated, with a Document Type Description (DTD) or Schema (see the question on document types [p.19]), and maybe an internal subset (local DTD changes in [square brackets]), and an arbitrarily complex nested structure:

```

<?xml version="1.0" encoding="iso-8859-1"?>
<!DOCTYPE titlepage SYSTEM "http://www.foo.bar/dtds/typo.dtd"
[<!ENTITY % active.links "INCLUDE">]>
<titlepage id="BG12273624">
  <white-space type="vertical" amount="36"/>
  <title font="Baskerville" alignment="centered"
    size="24/30">Hello, world!</title>
  <white-space type="vertical" amount="12"/>
    <!-- In some copies the following decoration is
      hand-colored, presumably by the author -->
  <image location="http://www.foo.bar/fleuron.eps" type="URL"
    alignment="centered"/>
  <white-space type="vertical" amount="24"/>
  <author font="Baskerville" size="18/22" style="italic">Vitam
    cepi</author>
  <white-space type="vertical" revisionflag="filler"/>
</titlepage>

```

Or they can be anywhere between: a lot will depend on how you want to define your document type (or whose you use) and what it will be used for.

C.4 How does XML handle white-space in my documents?

§ The SGML rules regarding white-space have been changed for XML. All white-space, including linebreaks, TAB characters, and normal spaces, *even between those elements where no text can ever appear*, is passed by the parser *unchanged* to the application (browser, formatter, viewer, converter, etc), identifying the context in which the white-space was found (element content, data content, or mixed content, if this information is available to the parser, eg from a DTD or Schema). This means *it is the application's responsibility to decide what to do with such space, not the parser's*:

Parsers keep it all. It's up to the application to decide what to do with it.

- </> *insignificant white-space* between structural elements (space which occurs where only element content is allowed, ie between other elements, where text data never occurs) will get passed to the application (in SGML this white-space gets suppressed, which is why you can put all that extra space in HTML documents and not worry about it);
- </> *significant white-space* (space which occurs within elements which *can* contain text and markup mixed together, usually mixed content or PCDATA) will still get passed to the application exactly as under SGML. It is the application's responsibility to handle it correctly.

The parser must inform the application that white-space has occurred in element content, if it can detect it. (Users of SGML will recognize that this information is not in the ESIS⁸⁹, but it *is* in the Grove⁹⁰.)

⁸⁹<http://xml.coverpages.org/WG8-n931a.html>

⁹⁰<http://xml.coverpages.org/topics.html#groves>

```

<chapter>
  <title>
    My title for
    Chapter 1.
  </title>
  <para>
text
  </para>
</chapter>

```

In the example above, the application will receive all the pretty-printing linebreaks, TABs, and spaces between the elements *as well as those* embedded in the chapter title. It is the function of the application, not the parser, to decide which type of white-space to discard and which to retain. Many XML applications have configurable options to allow programmers or users to control how such white-space is handled.

Note

Why? Because XML can be used without a DTD or Schema, and in those cases it's impossible for the parser to know in advance whether white-space has occurred in element content (and would therefore be discardable) or in mixed content or PCDATA (where it must be preserved). In order to handle this case, the general rule was imposed that *all* white-space must be reported to the application.

C.5 Which parts of an XML document are case-sensitive?

All of it, both markup and text.

All of it, both markup and text. This is significantly different from HTML and most other SGML applications. It was done to allow markup in non-Latin-alphabet languages, and to obviate problems with case-folding in scripts which are caseless.

- </> Element type names are case-sensitive: you must stick with whatever combination of upper- or lower-case you use to define them (either by first usage or in a DTD or Schema [p.19]). So you can't say <BODY>...</body>: upper- and lower-case must match; thus , , and are three different element types;
- </> For well-formed XML documents with no DTD, the first occurrence of an element type name defines the casing;
- § </> Attribute names are also case-sensitive, for example the two width attributes in <PIC width="7in"/> and <PIC WIDTH="6in"/> (if they occurred in the same file) are separate attributes, because of the different case of width and WIDTH;
- § </> Attribute values are also case-sensitive. CDATA values (eg Url="MyFile.SGML") always have been, but NAME types (ID and IDREF attributes, and token list attributes) are now case-sensitive as well;
- </> All general and parameter entity names (eg Á), and your data content (text), are case-sensitive as always.

C.6 How can I make my existing HTML files work in XML?

Either convert them to conform to some new document type (with or without a DTD or Schema) and write a stylesheet to go with them; or edit them to conform to XHTML [p.17].

Either make them XHTML or use a different document type.

It is necessary to convert existing HTML files because XML does not permit end-tag minimisation (missing `</p>`, etc), unquoted attribute values, and a number of other SGML shortcuts which are normal in most HTML DTDs. However, many HTML authoring tools already produce almost (but not quite) well-formed XML.

In preparation for XML, the W3C's HTML Tidy⁹¹ program can clean up some of the formatting mess left behind by inadequate HTML editors, and even separate out some of the formatting to a stylesheet, but there is usually still some hand-editing to do.

Converting valid HTML to XHTML

If your HTML files are valid (full formal validation with an SGML parser, not just a simple syntax check), then try validating them as XHTML with an XML parser. If you have been creating clean HTML without embedded formatting then this process should throw up only mismatches in upper/lowercase element and attribute names, and empty elements (plus perhaps the odd non-standard element type name if you use them). Simple hand-editing or a short script should be enough to fix these changes.

If your HTML validly uses end-tag omission, this can be fixed automatically by a normalizer program like `sgmlnorm` (part of SP⁹²) or by the `sgml-normalize` function in an editor like Emacs/`psgml` (don't be put off by the names, they both do XML).

If you have a lot of valid HTML files, you could write a script to do this in a programming language which understands SGML markup (such as Omnimark⁹³, Balise⁹⁴, SGMLC⁹⁵, or a scripting language (eg Perl, Python, Tcl, etc), using their SGML XML libraries; or you could even use editor macros if you know what you're doing.

Converting to a new document type

If you want to move your files out of HTML into some other DTD entirely, there are many native XML application DTDs, and standard XML versions of popular DTDs like TEI and DocBook to choose from. There is a pilot site run by CommerceNet (<http://www.xmlx.com/>) for the exchange of XML DTDs.

Alternatively you could just make up your own markup: so long as it makes sense and you create a well-formed file, you should be able to write a CSS or XSLT stylesheet and have your document displayed in a browser.

⁹¹<http://www.w3.org/People/Raggett/tidy/>

⁹⁵<http://www.jclark.com/sp/>

⁹⁵<http://www.omnimark.com>

⁹⁵<http://www.balise.com>

⁹⁵<http://sgml.dircon.co.uk/>

Converting invalid HTML to well-formed XHTML

If your files are invalid HTML (95% of the Web) they can be converted to well-formed DTDless files as follows:

1. replace the DOCTYPE Declaration with the XML Declaration
`<?xml version="1.0" standalone="yes" encoding="iso-8859-1"?>`
2. If there was no DOCTYPE Declaration, just prepend the XML Declaration.
3. Change any EMPTY elements (eg every BASE, ISINDEX, LINK, META, NEXTID and RANGE in the header, and every AREA, ATOPARA, AUDIOSCOPE, BASEFONT, BR, CHOOSE, COL, FRAME, HR, IMG, KEYGEN, LEFT, LIMITTEXT, OF, OVER, PARAM, RIGHT, SPACER, SPOT, TAB, and WBR in the body of the document) so that they end with `>` instead, for example ``;
4. Make all element names and attribute names lowercase;
5. Ensure there are correctly-matched explicit end-tags for all non-EMPTY elements; eg every `<para>` must have a `</para>`, etc;
6. Escape all `<` and
& non-markup (ie literal text) characters as `<` and `&` respectively (there shouldn't be any isolated `<` characters to start with);
7. Ensure all attribute values are in matched quotes (values with embedded single quotes must be in double quotes, and vice versa—if you need both, use the `"` character entity reference);
8. Ensure all script URLs which have
& as field separator are changed to use `&` instead.

Be aware that some obsolete HTML browsers may not accept XML-style EMPTY elements with the trailing slash, so the above changes may not be backwards-compatible. An alternative is to add a dummy end-tag to all EMPTY elements, so `` becomes ``. This is valid XML provided you guarantee never to put any text content in such elements. Adding a space before the closing slash in EMPTY elements (eg ``) may also fool older browsers into accepting XHTML as HTML.

If you answer Yes to any of the questions in the *Checklist for invalid HTML* [p.17], you can save yourself a lot of grief by fixing those problems first before doing anything else. You will likely then be getting close to having well-formed files.

Markup which is syntactically correct but semantically meaningless or void should be edited out before conversion. Examples are spacing devices such as repeated empty paragraphs or linebreaks, empty tables, invisible spacing GIFs etc. XML uses stylesheets, so you won't need any of these.

Unfortunately there is rather a lot of work to do if your files are invalid: this is why many Webmasters now insist that only valid or well-formed files are used (and why you should instruct your designers to do the same), in order to avoid unnecessary manual maintenance and conversion costs later.

Checklist for invalid HTML

If your HTML files fall into this category (HTML created by most WYSIWYG editors is usually invalid) then they will almost certainly have to be converted manually, although if the deformities are regular and carefully constructed, the files may actually be almost well-formed, and you could write a program or script to do as described above. The oddities you may need to check for include:

- ↔ Do the files contain markup syntax errors? For example, are there any missing angle-brackets, backslashes instead of forward slashes on end-tags, or elements which nest incorrectly (eg `those starting <I>inside another but ending outside</I>)?`
- ↔ Are there any URLs (eg in `hrefs` or `srcs`) which use Microsoft Windows-style backslashes instead of normal forward slashes?
- ↔ Do the files contain markup which conflicts with HTML DTDs, such as headings or lists inside paragraphs, list items outside list environments, header elements like `base` preceding the first `html`, etc? (another sloppy editor trick)
- ↔ Do the files use imaginary elements which are not in any known HTML DTD? (large amounts of these are used in proprietary markup systems masquerading as HTML). Although this is easy to transform to a DTDless well-formed file (because you don't have to define elements in advance) most proprietary or browser-specific extensions have never been formally defined, so it is often impossible to work out meaningfully where the element types can be used.
- ↔ Are there any invalid (non-XML) characters in your files? Look especially for native Apple Mac Roman-8 characters left by careless designers; any of the illegal characters (the 32 characters at decimal codes 128–159 inclusive) inserted by MS-Windows editors; and any of the ASCII control characters 0–31 (except those permitted like TAB, CR, and LF). These need to be converted to the correct characters in ISO 8859-1 (a common default in browsers), or the relevant plane of Unicode (in which case you will probably need to use UTF-8 as your document encoding).
- ↔ Do your files contain invalid (old Mosaic/Netscape-style) comments? Comments must look `<!-- like this -->` with double-dashes each end and no double (especially not multiple) dashes in between.

C.7 Is there an XML version of HTML?

Yes, XHTML
from W3C

Yes, the W3C recommends using XHTML⁹⁶ which is 'a reformulation of HTML 4 in XML 1.0'. This specification defines HTML as an XML application, and provides three DTDs corresponding to the ones defined by HTML 4.* (Strict, Transitional, and Frameset).

The semantics of the elements and their attributes are as defined in the W3C Recommendation for HTML 4. These semantics provide the foundation for future extensibility of XHTML. Compatibility with existing HTML user agents is possible by following a small set of guidelines (see the W3C site).

HTML 5 is on the way: see the W3C web site for details.

⁹⁶<http://www.w3.org/TR/xhtml1/>

C.8 If XML is just a subset of SGML, can I use XML files directly with existing SGML tools?

Yes, if they are up to date

Yes, provided you use up-to-date SGML software which knows about the WebSGML Adaptations TC to ISO 8879⁹⁷ (the features needed to support XML, such as the variant form for EMPTY elements; some aspects of the SGML Declaration such as NAMECASE GENERAL NO; multiple attribute token list declarations, etc).

An alternative is to use an SGML DTD to let you create a fully-normalised SGML file, but one which does not use empty elements; and then remove the DocType Declaration so it becomes a well-formed DTDless XML file. Most SGML tools now handle XML files well, and provide an option switch between the two standards. (see the pointers in the question *What XML software is available?* [p.38]).

C.9 I'm used to authoring and serving HTML. Can I learn XML easily?

Yes

Yes, very easily, but at the moment there is still a need for more tutorials, simpler tools, and more examples of XML documents. 'Well-formed' XML documents [D.2, p.35] may look similar to HTML except for some small but very important points of syntax.

The big practical difference is that XML has to stick to the rules. HTML browsers let you serve them even fatally broken or ridiculously corrupt HTML because they don't do a formal parse but elide all the broken bits instead. With XML your files have to be completely correct or they simply won't work at all. One outstanding problem is that some browsers claiming XML conformance are also broken, and some browsers' support for CSS styling is dubious at the best. Try yours on the test file at <http://imbolc.ucc.ie/test.xml>.

C.10 Can XML use non-Latin characters?

Yes, this is the default

Yes, the XML Specification [p.33] explicitly says XML uses ISO 10646⁹⁸, the international standard character repertoire which covers most known languages. Unicode⁹⁹ is an identical repertoire, and the two standards track each other. The spec says (2.2): 'All XML processors must accept the UTF-8 and UTF-16 encodings of ISO 10646...'.¹⁰⁰

UTF-8 is an encoding of Unicode into 8-bit characters: the first 128 are the same as ASCII, and higher-order characters are used to encode anything else from Unicode into sequences of between 2 and 6 bytes¹⁰⁰. UTF-8 in its single-octet form is therefore the same as ISO 646 IRV (ASCII), so you can continue to use ASCII for English or other languages using the Latin alphabet without diacritics. Note that UTF-8 is incompatible with ISO 8859-1 (ISO Latin-1) after code point 127 decimal (the end of ASCII).

UTF-16 is an encoding of Unicode into 16-bit characters, which lets it represent 16 planes. UTF-16 is incompatible with ASCII because it uses two 8-bit bytes per character (four bytes above U+FFFF).

⁹⁷<http://www.ornl.gov/sgml/sc34/document/0029.htm>

⁹⁸<http://www.iso.ch/>

⁹⁹<http://www.unicode.org/>

¹⁰⁰<http://www.cl.cam.ac.uk/~mgk25/ucs/examples/UTF-8-test.txt>

Bertilo Wennergren writes:

UTF-16 is an encoding that represents each Unicode character of the first plane (the first 64K characters) of Unicode with a 16-bit unit—in practice with two bytes for each character. Thus it is backwards compatible with neither ASCII nor Latin-1. UTF-16 can also access an additional 1 million characters by a mechanism known as surrogate pairs (two 16-bit units for each character).

‘... the mechanisms for signalling which of the two are in use, and for bringing other encodings into play, are (...) in the discussion of character encodings.’ The XML Specification (p.33) explains how to specify in your XML file which coded character set you are using.

‘Regardless of the specific encoding used, any character in the ISO 10646 character set may be referred to by the decimal or hexadecimal equivalent of its bit string’: so no matter which character set you personally use, you can still refer to specific individual characters from elsewhere in the encoded repertoire by using `&#dddd;` (decimal character code) or `&#xHHHH;` (hexadecimal character code, in uppercase). The terminology can get confusing, as can the numbers: see the ISO 10646 Concept Dictionary¹⁰¹. Rick Jelliffe has XML-ized the ISO character entity sets¹⁰². Mike Brown’s encoding information at <http://skew.org/xml/tutorial/>¹⁰³ is a very useful explanation of the need for correct encoding. There is an excellent online database of glyphs and characters in many encodings from the Estonian Language Institute server at <http://www.eki.ee/letter/>¹⁰⁴.

C.11 What’s a Document Type Definition (DTD) and where do I get one?

A DTD is a formal description in XML Declaration Syntax of a particular type of document. It sets out what names are to be used for the different types of element, where they may occur, and how they all fit together. For example, if you want a document type to be able to describe Lists which contain Items, the relevant part of your DTD might contain something like this:

```
<!ELEMENT List (Item)+>
<!ELEMENT Item (#PCDATA)>
```

This defines a list as an element type containing one or more items (that’s the plus sign); and it defines items as element types containing just plain text (Parsed Character Data or PCDATA). Validating parsers read the DTD before they read your document so that they can identify where every element type ought to come and how each relates to the other, so that applications which need to know this in advance (most editors, search engines, navigators, databases) can set themselves up correctly. The example above lets you create lists like:

```
<List><Item>Chocolate</Item><Item>Music</Item><Item>Surfing</Item></List>
```

A specification of document structure. You can write one or download them.

¹⁰⁴http://cns-web.bu.edu/pub/djohnson/web_files/i18n/ISO-10646.html

¹⁰⁴<http://xml.coverpages.org/xml-ISOents.txt>

¹⁰⁴<http://skew.org/xml/tutorial/>

¹⁰⁴<http://www.eki.ee/letter/>

How the list appears in print or on the screen depends on your stylesheet: you do not normally put anything in the XML to control formatting like you had to do with HTML before stylesheets. This way you can change style easily without ever having to edit the document itself.

§ A DTD provides applications with advance notice of what names and structures can be used in a particular document type. Using a DTD when editing files means you can be certain that all documents which belong to a particular type will be constructed and named in a consistent and conformant manner. DTDs are less important for processing documents already known to be well-formed, but they are still needed if you want to take advantage of XML's special attribute types like the built-in ID/IDREF cross-reference mechanism, or the use of default attribute values.

§ There are thousands of DTDs already in existence in all kinds of areas (see the SGML/XML Web pages¹⁰⁵ for pointers). Many of them can be downloaded and used freely; or you can write your own (see the question on creating your own DTD [p.21]. Old SGML DTDs need to be converted to XML for use with XML systems: read the question on converting SGML DTDs to XML [p.43]. Most popular SGML DTDs are already available in XML format.

The alternatives to a DTD are various formats of Schema [p.22]. These provide much finer data validation capabilities than DTDs. The W3C Schemas use XML Instance Syntax, which makes them readable as XML documents in their own right, but they can get very unwieldy. The RelaxNG Schemas can be expressed in several different formats, including one similar in simplicity to DTD syntax.

C.12 Does XML let me make up my own tags?

No, it lets you make up names for your own element types. If you think tags and elements are the same thing you are already in trouble: read the rest of this question carefully.

Yes but they're not called tags. They're element types.

Bob DuCharme writes:

Don't confuse the term 'tag' with the term 'element'. They are not interchangeable. An element usually contains two different kinds of tag: a start-tag and an end-tag, with text or more markup between them.

XML lets you decide which elements you want in your document and then indicate your element boundaries using the appropriate start- and end-tags for those elements. Each `<!ELEMENT . . .` declaration defines a type of element that may be used in a document conforming to that DTD. We call this type of element an 'element type'. Just as the HTML DTD includes the `H1` and `P` element types, your document can have `color` and `price` element types, or anything else you want.

Normal non-empty elements are made up of a start-tag, the element's content, and an end-tag. `<color>red</color>` is a complete instance of the `color` element. `<color>` is only the start-tag of the element, showing where it begins; it is not the element itself.

Empty elements are a special case that may be represented either as a pair of start- and end-tags with nothing between them (eg `<price retail="123"></price>`) or as a single empty element start-tag that has a closing slash to tell the parser 'don't go looking for an end-tag to match this' (eg `<price retail="123"/>`).

¹⁰⁵<http://xml.coverpages.org/>

C.13 How do I create my own DTD?

You need to use the XML Declaration Syntax (very simple: declaration keywords begin with `<!` rather than just the open angle bracket, and the way the declarations are formed also differs slightly). Here's an example of a DTD for a shopping list, based on the fragment used in earlier [p.19]:

```
<!ELEMENT Shopping-List (Item)+>
<!ELEMENT Item (#PCDATA)>
```

It says that there shall be an element called `Shopping-List` and that it shall contain elements called `Item`: there must be at least one `Item` (that's the plus sign) but there may be more than one. It also says that the `Item` element may contain only parsed character data (PCDATA, ie text: no further markup).

Because there is no other element which contains `Shopping-List`, that element is assumed to be the 'root' element, which encloses everything else in the document. You can now use it to create an XML file: give your editor the declarations:

```
<?xml version="1.0"?>
<!DOCTYPE Shopping-List SYSTEM "shoplist.dtd">
```

(assuming you put the DTD in that file). Now your editor will let you create files according to the pattern:

```
<Shopping-List>
  <Item>Chocolate</Item>
  <Item>Sugar</Item>
  <Item>Butter</Item>
</Shopping-List>
```

It is possible to develop complex and powerful DTDs of great subtlety, but for any significant use you should learn more about document systems analysis and document type design. See for example *Developing SGML DTDs: From Text to Model to Markup*¹⁰⁶: this was written for SGML but perhaps 95% of it applies to XML as well, as XML is much simpler than full SGML—see the list of restrictions [D.4, p.37] which shows what has been cut out.

Warning

Incidentally, a DTD file *never* has a DOCTYPE Declaration in it: that only occurs in an XML document instance (it's what references the DTD). And a DTD file also never has an XML Declaration at the top either.

C.14 Can a root element type be explicitly declared in the DTD?

No, use the Document Type Declaration.

No. This is done in the document's Document Type Declaration, not in the DTD.

Bob DuCharme writes:

In a Document Type Declaration like this:

```
<!DOCTYPE chapter SYSTEM "docbookx.dtd">
```

the whole point of the chapter part is to identify which of the element types declared in the specified DTD should be used as the root element. I believe the highest level element in DocBook is `set`, but I find it hard to imagine someone creating a document to represent a set of books. We are free to use `set`, `book`, `chapter`, `article`, or even `para` as the document element for a valid DocBook document.

(One job some parsers do is determine which element type(s) in a DTD are not contained in the content model of any other element type: these are by deduction the prime candidates for being default root elements. (PF))

This is A Good Thing, because it adds flexibility to how the DTD is used. It's the reason that XML (and SGML) have lent themselves so well to electronic publishing systems in which different elements were mixed and matched to create different documents all conforming to the same DTD.

I've seen schema proposals that let you specify which of a schema's element types could be a document's root element, but after a quick look at section 3.3 of Part 1 of the W3C Schema Recommendation¹⁰⁷ and the RELAX NG schema for RELAX, I don't believe that either of these let you do this. I could be wrong.

C.15 I keep hearing about alternatives to DTDs. What's a Schema?

Like a DTD for validating content as well as structure.

A DTD [p.19] is for specifying the structure (only) of an XML file: it gives the names of the elements, attributes, and entities that can be used, and how they fit together. DTDs are designed for use with traditional text documents, not rectangular or tabular data, so the concept of data types hardly exists: text is just text. If you need to specify numeric ranges or to define limitations or checks on the text content, a DTD is the wrong tool.

The W3C XML Schema recommendation¹⁰⁸ provides a means of specifying formal data typing and validation of element content in terms of data types, so that document type designers can provide criteria for checking the data content of elements as well as the markup itself. Schemas are written in XML Instance Syntax, as XML files, avoiding the need for processing software to be able to read XML Declaration Syntax (used for DTDs).

There is a separate Schema FAQ at <http://www.schemavalid.com>. The term 'vocabulary' is sometimes used to refer to DTDs and Schemas together. Designers should note that Schemas are aimed at database-style applications where element data content requires validation; where stricter data control is needed than is possible with DTDs; or where strong data typing is required: they are usually inappropriate for traditional text document publishing applications.

¹⁰⁶Maler/el Andaloussi.

¹⁰⁷http://www.w3.org/TR/xmlschema-1/#cElement_Declarations

¹⁰⁸<http://www.w3.org/TR/xmlschema-0/>

¹⁰⁹<http://xml.coverpages.org/properSpellingForPluralOfDTD.html>

Warning

Authors and publishers should note that the English plural of Schema is Schemas: the use of the singular to do duty for the plural is a foible dear to the semi-literate; the use of the old (Greek) plural schemata is unnecessary didacticism. Writers should also note that the plural of DTD is DTDs¹⁰⁹: there is no apostrophe—see *Eats, Shoots & Leaves: The Zero-Tolerance Approach to Punctuation*^l.

^lTruss.

Bob DuCharme writes:

Many XML developers were dissatisfied with the syntax of the markup declarations described in the XML spec for two reasons. First, they felt that if XML documents were so good at describing structured information, then the description of a document type's own structure (its schema) should be in an XML document instead of written with its own special syntax. In addition to being more consistent, this would make it easier to edit and manipulate the schema with regular document manipulation tools. Secondly, they felt that traditional DTD notation didn't allow document type designers the power to impose enough constraints on the data—for example, the ability to say that a certain element type must always have a positive integer value, that it may not be empty, or that it must be one of a list of possible choices. This eases the development of software using that data because the developer has less error-checking code to write.

C.16 How do I get XML into or out of a database?

Ask your database manufacturer: they all provide XML import and export modules. In some trivial cases there will be a 1:1 match between field names and element types; but in most cases some programming is required to establish the matches, but this can usually be stored as a procedure so that subsequent uses are simply commands or calls with the relevant parameters.

Ask your
database
manufacturer

Warning

Users from a database or computer science background should be aware that XML is not a database management system: it is a text markup system. While there are many similarities, some of the concepts of one are simply non-existent in the other: XML does not possess some database-like features in the same way that databases do not possess markup-like ones. It is a common error to believe that XML is a DBMS like Oracle or Access and therefore possesses the same facilities. It doesn't.

Database users should read the article *Requirements for XML Document Database Systems*¹¹⁰. (Thanks to Bart Lateur for identifying this.) Ronald Bourret also maintains a good resource on XML and Databases discussing native XML databases at <http://www.rpbouret.com/xml/XMLAndDatabases.htm>.

¹¹⁰Salminen/Tompa.

C.17 How will XML affect my document links?

XML Links are much more powerful, but not yet implemented in browsers

The linking abilities of XML systems are potentially much more powerful than those of HTML, so you'll be able to do much more with them. Existing href-style links will remain usable, but the new linking technology is based on the lessons learned in the development of other standards involving hypertext, such as TEI¹¹¹ and HyTime¹¹², which let you manage bidirectional and multi-way links, as well as links to a whole element or span of text (within your own or other documents) rather than to a single point. These features have been available to SGML users for many years, so there is considerable experience and expertise available in using them. Currently only Mozilla Firefox implements XLink.

The XML Linking Specification (XLink)¹¹³ and the XML Extended Pointer Specification (XPointer)¹¹⁴ documents contain the details. An XML link can be either a URL or a TEI-style Extended Pointer (XPointer [p.24]), or both. A URL on its own is assumed to be a resource; if an XPointer or XLink follows it, it is assumed to be a sub-resource of that URL; an XPointer on its own is assumed to apply to the current document (all exactly as with HTML).

An XLink is always preceded by one of #, ?, or |. The # and ? mean the same as in HTML applications; the | means the sub-resource can be found by applying the link to the resource, but the method of doing this is left to the application. An XPointer can only follow a #.

The TEI Extended Pointer Notation¹¹⁵ (EPN) is much more powerful than the fragment address on the end of some URLs, as it allows you to specify the location of a link end using the structure of the document as well as (or in addition to) known, fixed points like IDs. For example, the linked second occurrence [p.24] of the word 'XPointer' two paragraphs back could be referred to with the URL (shown here with linebreaks and spaces for clarity: in practice it would of course be all one long string):

```
http://www.ucc.ie/xml/faq.sgml#ID(hypertext).child(1,#element,'answer').  
child(2,#element,'para').child(1,#element,'link')
```

This means the first link element within the second paragraph within the answer in the element whose ID is "hypertext" (this question). Count the objects from the start of this question (which has the ID "hypertext") in the XML source¹¹⁶:

1. the first child object is the element containing the question (<qandaentry>);
2. the second child object is the answer (the <answer> element);
3. within this element go to the second paragraph;
4. find the first link element.

¹¹¹<http://www.tei-c.org/>

¹¹²<http://xml.coverpages.org/hytime.html>

¹¹³<http://www.w3.org/TR/xlink/>

¹¹⁴<http://www.w3.org/TR/WD-xptr>

¹¹⁵<http://etext.virginia.edu/bin/tei-tocs?div=DIV2;id=SAXR>

¹¹⁶<http://www.ucc.ie/xml/faq.sgml>

David Megginson has produced an `xpointer`¹¹⁷ function for Emacs/`psgml` which will deduce an XPointer for any location in an XML document. XML Spy has a similar function.

C.18 Can I do mathematics using XML?

Yes, using MathML.

Yes, if the document type [p.19] you use provides for math, and your users' browsers are capable of rendering it. The mathematics-using community has developed the MathML Recommendation¹¹⁸ at the W3C, which is a native XML application suitable for embedding in other DTDs and Schemas.

It is also possible to make XML fragments from other DTDs, such as the long-expired HTML3, the near-obsolete HTML Pro, ISO 12083 Math¹¹⁹, or OpenMath¹²⁰, or one of your own making. Browsers which display math embedded in SGML have existed for many years (eg DynaText, Panorama, Multidoc Pro), and mainstream browsers are now rendering MathML (eg Mozilla, Netscape). David Carlisle has produced a set of stylesheets¹²¹ for rendering MathML in browsers. It is also possible to use XSLT to convert XML math markup to \LaTeX for print (PDF) rendering, or to use XSL:FO.

C.19 How does XML handle metadata?

Any way you want.

Because XML lets you define your own markup languages, you can make full use of the extended hypertext features of XML (see the question on Links [p.24]) to store or link to metadata in any format (eg using ISO 11179¹²², as a Topic Maps Published Subject¹²³, with Dublin Core, Warwick Framework¹²⁴, or with Resource Description Framework (RDF)¹²⁵, or even Platform for Internet Content Selection (PICS)¹²⁶).

There are no predefined elements in XML, because it is an architecture, not an application, so it is not part of XML's job to specify how or if authors should or should not implement metadata. You are therefore free to use any suitable method. Browser makers may also have their own architectural recommendations or methods to propose.

C.20 Can I use JavaScript, ActiveX, etc in XML files?

Not in the XML file itself, but via a stylesheet.

This will depend on what facilities the browser makers implement. XML is about describing information; scripting languages and languages for embedded functionality are software which enables the information to be manipulated at the user's end, so these languages do not normally have any place in an XML file itself, but in stylesheets like XSL and CSS.

¹¹⁷<http://www.megginson.com/Software/psgml-xpointer.el>

¹¹⁸<http://www.w3.org/Math/>

¹¹⁹<http://xml.coverpages.org/gen-apps.html#iso12083DTDs>

¹²⁰<http://www.openmath.org/>

¹²¹<http://www.mathmlconference.org/2002/presentations/carlisle/>

¹²²<http://www.sdct.itl.nist.gov/~ftp/x318/other/Standards/iso11179/>

¹²³<http://www.oasis-open.org/committees/tm-pubsubbj/>

¹²⁴<http://purl.oclc.org/metadata/dublin-core/>

¹²⁵<http://www.dstc.edu.au/RDU/RDF/>

¹²⁶<http://www.w3.org/PICS/>

XML itself provides a way to define the markup needed to implement scripting languages: as a neutral standard it neither encourages nor discourages their use, and does not favour one language over another, so it is possible to use XML markup to store the program code, from where it can be retrieved by (for example) XSLT and re-expressed in a HTML `script` element.

Server-side script embedding, like PHP or ASP, can be used with the relevant server to modify the XML code on the fly, as the document is served, just as they can with HTML. Authors should be aware, however, that embedding server-side scripting may mean the file as stored is not valid XML: it only becomes valid when processed and served, so care must be taken when using validating editors or other software to handle or manage such files. A better solution may be to use an XML serving solution like Cocoon, AxKit, or PropelX.

C.21 Can I use Java to create or manage XML files?

Sure.

Yes, any programming language can be used to output data from any source in XML format. There is a growing number of front-ends and back-ends for programming environments and data management environments to automate this. Java appears to be the most popular one at the moment.

There is a large body of middleware (APIs) written in Java and other languages for managing data either in XML or with XML input or output. There is a suite of Java tutorials (with source code and explanation) available at <http://developerlife.com>¹²⁷.

Note

Please do not mail the FAQ editor with questions about your Java programming bugs. Ask one of the Java newsgroups instead, or sign up for the XML SummerSchool (A.10, p.4), where there is usually a session on using Java and XML.

C.22 How do I execute or run an XML file?

Not a meaningful question. XML is a data format.

You can't and you don't. XML itself is not a programming language, so XML files don't 'run' or 'execute'. XML is a markup specification language and XML files are just data: they sit there until you run a program which displays them (like a browser) or does some work with them (like a converter which writes the data in another format, or a database which reads the data), or modifies them (like an editor).

If you want to view or display an XML file, open it with an XML editor or an XML browser.

The water is muddied by XSL (both XSLT and XSL:FO) which use XML syntax to specify declarative programming steps. In these cases it is arguable that you can 'execute' XML code, by running a processing application like Saxon, which implements the directives specified in XSLT files to process XML.

C.23 How do I control formatting and appearance?

Use a CSS or XSLT stylesheet.

In HTML, default styling was built into the browsers because the tagset of HTML was predefined and hardwired into browsers. In XML, where you can define your own tagset, browsers cannot possibly be expected to guess or know in advance what names you are going to use and what they will mean, so you need a stylesheet if you want to display formatted text.

Browsers which read XML [p.8] will accept and use a CSS stylesheet at a minimum, but you can also use the more powerful XSLT stylesheet language to transform your XML into HTML—which browsers, of course, already know how to display (and that HTML can still use a CSS stylesheet). This way you get all the document management benefits of using XML, but you don't have to worry about your readers needing XML smarts in their browsers.

Mike Brown writes:

XSLT is an XML document processing language that uses source code that happens to be written in XML. An XSLT document declares a set of rules for an XSLT processor to use when interpreting the contents of an XML document. These rules tell the XSLT processor how to generate a new XML-like data structure and how that data should be emitted—as an XML document, as an HTML document, as plain text, or perhaps in some other format.

This transformation can be done either inside the browser, or by the server before the file is sent. Transformation in the browser offloads the processing from the server, but may introduce browser dependencies, leading to some of your readers being excluded. Transformation in the server makes the process browser-independent, but places a heavier processing load on the server.

As with any system where files can be viewed at random by arbitrary users, the author cannot know what resources (such as fonts) are on the user's system, so the same care is needed as with HTML using fonts. To invoke a stylesheet from an XML file for standalone processing in the browser, include one of the stylesheet declarations:

```
<?xml-stylesheet url="foo.xml" type="text/xml"?>
<?xml-stylesheet url="foo.css" type="text/css"?>
```

(substituting the URL of your stylesheet, of course). The Cascading Stylesheet Specification (CSS)¹²⁸ provides a simple syntax for assigning styles to elements, and has been implemented in most browsers.

Dave Pawson maintains a comprehensive FAQ at <http://www.dpawson.co.uk/xsl/xslfaq.html>¹²⁹. XSL uses XML syntax (an XSL stylesheet is just an XML file) and has widespread support from several major browser vendors (see the questions on browsers [p.8] and other software [p.38]). XSL comes in two flavours:

↔ XSL itself, which is a pure formatting language, outputting a Formatted Objects (FO)

¹²⁷<http://developerlife.com>

¹²⁸<http://www.w3.org/Style/css>

¹²⁹<http://www.dpawson.co.uk/xsl/xslfaq.html>

file, which needs a text formatter like FOP¹³⁰, PassiveTeX¹³¹, XEP¹³², or others to create printable output (in PDF). Currently I am not aware of any Web browsers which support direct XSL rendering to PDF;

- ↔ XSLT (T for Transformation), which is a language to specify transformations of XML into HTML either inside the browser or at the server before transmission. It can also specify transformations from one vocabulary of XML to another, and from XML to plaintext (which can be any format, including RTF and L^AT_EX).

Currently only Microsoft Internet Explorer 5.5 and above, and Firefox¹³³ 0.9.6 and above handle XSLT inside the browser (MSIE5.5 needs some post-installation surgery¹³⁴ to remove the obsolete WD-xsl and replace it with the current XSL-Transform processor; MSIE6 and Firefox work as installed).

There is a growing use of server-side processors like Cocoon¹³⁵, AxKit¹³⁶ and PropelX¹³⁷, which let you store your information in XML but serve it auto-converted to HTML or some other format, thus allowing the output to be used by any browser. XSLT is also widely used to transform XML into non-SGML formats for input to other systems (for example to transform XML into L^AT_EX for typesetting).

C.24 How do I use graphics in XML?

Graphics have traditionally just been links which happen to have a picture file at the end rather than another piece of text. They can therefore be implemented in any way supported by the XLink and XPointer specifications (see earlier question [p.24]), including using similar syntax to existing HTML images. They can also be referenced using XML's built-in NOTATION and ENTITY mechanism in a similar way to standard SGML, as external unparsed entities.

Reference them as for HTML or use XLink. Or embed SVG.

The linking specifications, however, give you much better control over the traversal and activation of links, so an author can specify, for example, whether or not to have an image appear when the page is loaded, or on a click from the user, or in a separate window, without having to resort to scripting.

XML itself doesn't predicate or restrict graphic file formats: GIF, JPG, TIFF, PNG, CGM, and SVG at a minimum would seem to make sense; however, vector formats are normally preferred for non-photographic images.

You cannot embed a raw binary graphics file (or any other binary [non-text] data) directly into an XML file because any bytes happening to resemble markup would get misinterpreted: you must refer to it by linking (see below). It is, however, possible to include a text-encoded transformation of a binary file as a CDATA Marked Section, using something like UUencode with the markup characters], & and } removed from the map so

¹³⁰<http://xml.apache.org/>

¹³¹<http://users.ox.ac.uk/~rahtz/passivetex/>

¹³²<http://www.renderx.com/>

¹³³<http://www.mozilla.org/>

¹³⁴<http://www.netcrucible.com/xslt/msxml-faq.htm>

¹³⁵<http://xml.apache.org/>

¹³⁶<http://www.axkit.org/>

¹³⁷<http://www.propylon.com/>

that they could not occur as an erroneous CDATA termination sequence and be misinterpreted. You could even use simple hexadecimal encoding as used in PostScript. For vector graphics, however, the solution is to use SVG (see *SVG* [p.30]).

The point about using entities to manage your graphics is that you can keep the list of entity declarations separate from the rest of the document, so you can re-use the names if an image is needed more than once, but only store the physical file specification in a single place.

Bob DuCharme writes:

All the data in an XML document entity must be parseable XML. You can define an external entity as either a parsed entity (parseable XML) or an unparsed entity (anything else). Unparsed entities can be used for picture files, sound files, movie files, or whatever you like. They can only be referenced from within a document as the value of an attribute (much like a bitmap picture on an HTML Web page is the value of the `img` element's `src` attribute) and not part of the actual document. In an XML document, this attribute must be declared to be of type `ENTITY`, and the entity's declaration must specify a declared `NOTATION`, because if the entity isn't XML, the XML processor needs to know what it is. For example, in the following document, the `colliepic` entity is declared to have a JPEG notation, and it's used as the value of the empty `dog` element's `picfile` attribute.

```
<?xml version="1.0"?>
<!DOCTYPE dog [
  <!NOTATION JPEG SYSTEM "Joint Photographic Experts Group">
  <!ENTITY colliepic SYSTEM "lassie.jpg" NDATA JPEG>
  <!ELEMENT dog EMPTY>
  <!ATTLIST dog picfile ENTITY #REQUIRED>
]>
<dog picfile="colliepic"/>
```

The XLink and XPointer linking specifications describe other ways to point to a non-XML file such as a graphic. These offer more sophisticated control over the external entity's position, handling, and appearance within the XML document.

¹⁴³<http://www.w3.org/Graphics/SVG>

¹⁴³<http://www.siliconpublishing.org/svgfaq/>

¹⁴³<http://www.svgfaq.com/>

¹⁴³<http://www.siliconpublishing.org/svgfaq/XSLT.asp>

¹⁴³<http://www.netcrucible.com/xslt/msxml-faq.htm>

¹⁴³<http://www.xml.com/pub/a/2000/03/22/style/index.html>

Peter Murray-Rust writes:

SVG

GIFs and JPEGs cater for bitmaps (pixel representations of images: all made up of coloured dots). Vector graphics (scalable, made up of drawing specifications) are being addressed in the W3C's graphics activity as Scalable Vector Graphics (see <http://www.w3.org/Graphics/SVG>¹³⁸). (With the specification now virtually complete,) it will be possible to transmit the graphical representation as vectors directly within the XML file. For many graphics objects this will mean greatly decreased download time and scaling without loss of detail.

Max Dunn writes: SVG has really taken off recently, and is quite an XML success story (...) there are already nearly conformant implementations. We recently started an SVG FAQ at <http://www.siliconpublishing.org/svgfaq/>¹³⁹ which we are planning to move to <http://www.svgfaq.com/>¹⁴⁰.

XSLT can be used to generate SVG from XML; details are at <http://www.siliconpublishing.org/svgfaq/XSLT.asp>¹⁴¹ (be careful to use XSLT, not Microsoft's obsolete WD-xsl¹⁴²). Documents can also interact with SVG images (see <http://www.xml.com/pub/a/2000/03/22/style/index.html>¹⁴³).

C.25 What is parsing and how do I do it in XML?

Parsing is the act of splitting up information into its component parts (schools used to teach this in language classes until the teaching profession collectively caught the anti-grammar disease).

Parsing is
splitting up
information
into its
component
parts

'Mary feeds Spot' parses as

1. Subject = Mary, proper noun, nominative case
2. Verb = feeds, transitive, third person singular, present tense
3. Object = Spot, proper noun, accusative case

In computing, a parser is a program (or a piece of code or API that you can reference inside your own programs) which checks files to see if they follow a certain pattern. Most applications that open files have a parser, otherwise they'd never be able to figure out what the information means. Microsoft Word contains a parser which runs when you open a .doc file and checks that it can be understood. Give it a corrupted file and you'll get an error message.

XML applications are just the same: they contain a parser which reads XML, works out what function all the pieces of the document play, and makes that information available in memory to the rest of the program.

```
<person corpid="abc123" birth="1960-02-31" gender="female">
  <name>
    <forename>Judy</forename>
    <surname>O' Grady</surname>
  </name>
</person>
```

The example above parses as:

1. Element `person` identified with Attribute `corpid` containing "abc123" and Attribute `birth` containing "1960-02-31" and Attribute `gender` containing "female" containing ...
2. Element `name` containing ...
3. Element `forename` containing text 'Judy' followed by ...
4. Element `surname` containing text 'O'Grady'

(and lots of other stuff too).

But you can also get stand-alone parser-validators, which read an XML file and tell you if they find an error (like missing angle-brackets or quotes, or misplaced markup). This is very useful for testing files before doing something else with them, especially if they have been created by hand without an XML editor, or by a program elsewhere which may or may not have done the job properly.

Bill Rayer writes:

Using `nsgmls` or `rxp`

For standalone parsing/validation use software like James Clark's `nsgmls`¹⁴⁴ or Richard Tobin's `rxp`¹⁴⁵. Both work under Linux and Windows/DOS. The difference is in the format of the error listing (if any), and that some versions of `nsgmls` do not retrieve DTDs or other files over the network, whereas `rxp` does.

Make sure your XML file correctly references its DTD in a Document Type Declaration, and that the DTD file(s) are locally accessible (`rxp` will retrieve them if you have an Internet connection; `nsgmls` may not, so it may need a local copy).

Download and install the software. Make sure it is installed to a location where your operating system can find it. If you don't know what any of this means, you will need some help from someone who knows how to download and install software on your type of operating system.

For `nsgmls`, copy `pubtext/xml.soc` and `pubtext/xml.dcl` to your working directory.

To validate `myfile.xml`, open a shell window (Linux) or an MS-DOS ('command') window (Microsoft Windows). In these examples we'll assume your XML file is called `myfile.xml` and it's in a folder called `myfolder`. Use the real names of your folder and file when you type the commands.

For `nsgmls`: `$ nsgmls -wxml -wundefined -cxml.soc -s myfile.xml` There are many other options for `nsgmls` which are described on the Web page¹⁴⁶. The ones given here are required because it's an SGML parser and these options switch it to XML mode and suppress the normal output, leaving just the errors (if any). (In Microsoft Windows you may have to prefix the `nsgmls` with the full path to wherever it was installed, eg `C:\Program Files\nsgmls\nsgmls`).

For `rxp`: `$ rxp myfile.xml` `Rxp` also has some options which are described on its Web page¹⁴⁷. (In Microsoft Windows you may have to prefix the `rxp` with the full path to wherever it was installed, eg `C:\Program Files\rxp\rxp`).

¹⁴⁷<http://www.jclark.com/sp>

¹⁴⁷<http://www.cogsci.ed.ac.uk/~richard/rxp.html>

¹⁴⁷<http://www.jclark.com/sp>

¹⁴⁷<http://www.cogsci.ed.ac.uk/~richard/rxp.html>

C.26 How do I include one XML file in another?

Use a general entity, same as for SGML

This works exactly the same as for SGML. First you declare the entity you want to include, and then you reference it by name:

```
<!ENTITY chap3 SYSTEM "mydocs/chapter3.xml">
...
&chap3;
```

The difference between this method and the one used for including a DTD fragment (see *How do I include one DTD (or fragment) in another?* [p.44]) is that this uses a (external) general entity which is referenced in the same way as a character entity (with an ampersand).

The one thing to make sure of is that the included file *must not* have a DOCTYPE Declaration on it. If you've been using one for editing the fragment, remove it before using it in this way. Yes, this is a pain in the butt, but if you have lots of inclusions, write a script to strip off the declaration (or paste it back on again for editing).

C.27 When should I use a CDATA Marked Section (aka 'Can I embed HTML in XML')?

Only to hold text containing markup characters.

Almost never. The CDATA mechanism is designed to let an author quote examples of text containing markup characters (the open-angle-bracket and the ampersand). It turns off all parsing for the duration of the section (it gets turned on again by the closing sequence of double end-square-brackets and a close-angle-bracket).

Consequently, *nothing* in a CDATA section is recognised as anything to do with markup: it's just a string of opaque characters, and if you use an XML processor like XSLT set to XML output, any markup characters in it will get turned into their character-entity equivalent.

As a result you *cannot* expect HTML (valid, well-formed, or any other kind) to pass through XSL processing simply because you thought it was hidden inside a CDATA section. It's fine if you just want to *quote* HTML (for example to output it into a `pre` element for documentation) but it can't be used to embed HTML in an XML document in the hope that XSL processing will leave the markup characters untouched for future recognition as markup. If you're outputting in text-mode, or using different software, other rules may apply.

If you really want to embed HTML markup in another flavour of XML, and have it processed as such, you'll need to provide for the handling of that markup with templates in your XSLT stylesheet or whatever processor you use (and in your DTD or Schema, if you use one).

There are several ways around this: for more details, see the relevant question in the XSLT FAQ¹⁴⁸.

¹⁴⁸<http://www.dpawson.co.uk/xsl/sect2/cdata.html>

D Developers and Implementors (including WebMasters and server operators)

D.1 Where's the spec?

Right here¹⁴⁹

Right here¹⁵⁰ (<http://www.w3.org/TR/REC-xml>). Includes the EBNF. There are also versions in Japanese¹⁵¹; Spanish¹⁵²; Korean¹⁵³; and a Java-ised annotated version¹⁵⁴.

Eve Maler maintains the DTD used for the spec itself¹⁵⁵; the DTD is also to encode several other W3C specifications, such as XLink, XPointer, DOM, XML Schema, etc. There is documentation¹⁵⁶ available for the DTD. Note that the XML spec needs to use a special one-off version of the DTD¹⁵⁷, since the real original DTD used for it has long since been lost.

D.2 What are these terms DTDless, valid, and well-formed?

Well-formed means syntactically correct (DTD or not); valid means a DTD has been used.

XML lets you use a Document Type Definition (DTD) to describe the markup (elements and other constructs) available in any specific type of document. However, the design and construction of a DTD can be complex and non-trivial, so XML also lets you work without a DTD. DTDless operation means you can invent markup without having to define it formally, provided you stick to the rules of XML syntax.

To make this work, a DTDless file is assumed to define its own markup by the existence and location of elements where you create them. When an XML application encounters a DTDless file, it builds its internal model of the document structure while it reads it, because it has no DTD to tell it what to expect. There must therefore be no surprises or ambiguous syntax: the document must be 'well-formed' (must follow the rules).

To understand why this concept is needed, look at standard HTML as an example:

§ </> The `img` element is declared (in the DTDs for HTML) as EMPTY, so it doesn't have an end-tag (there is no such thing as ``);

</> Many other HTML elements (such as `para`) allow you to omit the end-tag for brevity when using the SGML version of HTML.

§ </> If an XML processor reads an HTML file without knowing this (because it isn't using a DTD), and it encounters an `` or a `<para>` (or any other start-tag), it would have no way to know whether or not to expect an end-tag. This makes it impossible to know if the rest of the file is correct or not, because it has now no evidence of whether it is inside an element or if it has finished with it.

Well-formed documents therefore *require* start-tags and end-tags on every normal element, and any EMPTY elements must be made unambiguous, either by using normal start-tags

¹⁵⁰<http://www.w3.org/TR/REC-xml>

¹⁵¹<http://www.fxis.co.jp/DMS/sgml/xml/>

¹⁵²<http://www.ucc.ie/xml/faq-es.html>

¹⁵³<http://xml.t2000.co.kr/faq/index.html>

¹⁵⁴<http://www.xml.com/axml/testaxml.htm>

¹⁵⁵<http://www.w3.org/XML/1998/06/xmlspec-v21.dtd>

¹⁵⁶<http://www.w3.org/XML/1998/06/xmlspec-report-v21.htm>

¹⁵⁷<http://www.w3.org/XML/1998/06/xmlspec-v21a.dtd>

and end-tags, or by appending a slash to the name of the start-tag before the closing `>` as a sign that there will be no separate end-tag.

All XML documents, both DTDless and valid, must be well-formed. They must start with an XML Declaration if necessary (for example, identifying the character encoding or using the Standalone Document Declaration):

```
<?xml version="1.0" encoding="iso-8859-1" standalone="yes"?>
<foo>
  <bar>...<blort/>...</bar>
</foo>
```

David Brownell writes:

XML that's just well-formed doesn't need to use a Standalone Document Declaration at all. Such declarations are there to permit certain speedups when processing documents while ignoring external parameter entities—basically, you can't rely on external declarations in standalone documents. The types that are relevant are entities and attributes. Standalone documents must not require any kind of attribute value normalisation or defaulting, otherwise they are invalid.

It's also possible to use a Document Type Declaration with DTDless files, even though there is no Document Type to refer to:

Richard Lander writes:

If you need character entities (other than the five built-in ones) in a DTDless file, you can declare them in an internal subset without referencing anything other than the root element type:

```
<?xml version="1.0" standalone="yes"?>
<!DOCTYPE example [
  <!ENTITY mdash "---">
]>
<example>Hindsight&mdash;a wonderful thing.</example>
```

Rules for well-formedness:

- ↔ All tags must be balanced: that is, every element which may contain character data or sub-elements must have both the start-tag and the end-tag present (omission is not allowed except for EMPTY elements, see below);
- ↔ All attribute values must be in quotes. The single-quote character (the apostrophe) may be used if the value contains a double-quote character, and vice versa. If you need isolated quotes as data as well, you can use `'` or `"`; . Do not under any circumstances use the automated typographic ('curly') inverted commas substituted by some wordprocessors for quoting attribute values.
- ↔ Any EMPTY elements (eg those with no end-tag like HTML's `img`, `hr`, and `br` and others) must *either* end with `/` or they must look like non-EMPTY elements by having a real end-tag (but no content). Example: `
` would become either `
` or `
</br>` (with nothing in between).
- ↔ There must not be any isolated markup-start characters (`<` or `&`) in your text data. They must be given as `<` and `&` respectively, and the sequence `]]&` may only occur as the end of a CDATA marked section: if you are using it for any other purpose it must be given as `]]&` .
- ↔ Elements must nest inside each other properly (no overlapping markup, same as for HTML);
- ↔ DTDless well-formed documents may use attributes on any element, but the attributes are all assumed to be of type CDATA. You cannot use ID/IDREF attribute types for parser-checked cross-referencing in DTDless documents.
- ↔ XML files with no DTD are considered to have `<`, `>`, `'`, `"`, and `&` predefined and thus available for use. With a DTD, all character entities used must be declared, including these five.

Rules for validity

Valid XML files are well-formed files which have a Document Type Definition (DTD) (p.19) and which conform to it. They must already be well-formed (p.35), so all the rules above apply.

A valid file begins with a Document Type Declaration, but may have an optional XML Declaration prepended:

```
<?xml version="1.0"?>
<!DOCTYPE advert SYSTEM "http://www.foo.org/ad.dtd">
<advert>
  <headline>...<pic/>...</headline>
  <text>...</text>
</advert>
```

The XML Specification predefines an SGML Declaration for XML which is fixed for all instances and is therefore hard-coded into all XML software and never specified separately (the declaration has been removed from the text of the Specification but is available at a separate document¹⁵⁸). The specified DTD must be accessible to the XML processor using

¹⁵⁸<http://www.w3.org/TR/NOTE-sgml-xml-971215#null>

the URL supplied in the SYSTEM Identifier, either by being available locally (ie the user already has a copy on disk), or by being retrievable via the network.

It is possible (many people would say preferable) to supply a Formal Public Identifier with the PUBLIC keyword, and use an XML Catalog to dereference it, but the Specification mandates a SYSTEM Identifier so this must still be supplied (after the PUBLIC identifier: no further keyword is needed):

```
<!DOCTYPE advert PUBLIC "-//Foo, Inc//DTD Advertisements//EN"
"http://www.foo.org/ad.dtd">
<advert>...</advert>
```

The test for validity is that a validating parser finds no errors in the file: it must conform absolutely to the definitions and declarations in the DTD.

¶ XML (W3C) Schemas are not usually linked directly from within an XML document instance in the way that DTDs are: the relevant Schema (XSD file) for a document instance is normally specified to the parser separately, either by file system reference, or using a Target Namespace¹⁵⁹.

D.3 Which should I use in my DTD, attributes or elements?

There is no single answer to this: a lot depends on what you are designing the document type for.

Traditional editorial practice for normal text documents is to put the real text (what would be printed) as character data content, and keep the metadata (information about the text) in attributes, from where they can more easily be isolated for analysis or special treatment like display in the margin or in a mouseover:

```
<l n="184">
  <spara>Portia</spara>
  <text>The quality of mercy is not strain'd,</text>
</l>
```

But from the systems point of view, there is nothing wrong with storing the data the other way round, especially where the volume of text data on each occasion is relatively small:

```
<line speaker="Portia" text="The quality of mercy is not strain'd,">184</line>
```

A lot will depend on what you want to do with the information and which bits of it are easiest accessed by each method. A rule of thumb for conventional text documents is that if the markup were all stripped away, the bare text should still be readable and usable, even if unformatted and inconvenient. For database output, however, or other machine-generated documents like e-commerce transactions, human reading may not be meaningful, so it is perfectly possible to have documents where all the data is in

See
<http://xml.coverpages.org/elementsAndAttrs.html>

¹⁵⁹<http://www.w3.org/TR/xmlschema-0/#NS>

attributes, and the document contains no character data in content models at all. See <http://xml.coverpages.org/elementsAndAttrs.html> for more information.

Mike Kay writes:

From a user: *'(...) do most of you out there use element-based or attribute-based xml? why?'*

Beginners always ask this question. Those with a little experience express their opinions passionately. Experts tell you there is no right answer.

(<http://lists.xml.org/archives/xml-dev/200006/msg00293.html>)

D.4 What else has changed between SGML and XML?

The principal changes are in what you can do in writing a Document Type Definition (DTD). To simplify the syntax and make it easier to write processing software, a large number of SGML markup declaration options have been suppressed (see the list of omitted features [p.43]).

Stricter syntax and no options.

An extra Name Start Character is permitted in XML Names (the colon) for use with namespaces [p.37] (enabling DTDs to distinguish element source, ownership, or application). Despite its classification, a colon may only appear in mid-name, *not* at the start or the end.

D.5 What's a namespace?

Randall Fowle writes:

A namespace is a collection of element and attribute names identified by a Uniform Resource Identifier reference. The reference may appear in the root element as a value of the `xmlns` attribute. For example, the namespace reference for an XML document with a root element `x` might appear like this:

```
<x xmlns="http://www.company.com/company-schema">
```

More than one namespace may appear in a single XML document, to allow a name to be used more than once. Each reference can declare a prefix to be used by each name, so the previous example might appear as

```
<x xmlns:spc="http://www.company.com/company-schema">
```

which would nominate the namespace for the `'spc'` prefix:

```
<spc:name>Mr. Big</spc:name>
```

A named DTD/Schema fragment identified by a URI (URL).

¹⁶⁰<http://www.rpbouret.com/xml/NamespacesFAQ.htm>

James Anderson writes:

In general, note that the binding may also be effected by a default value for an attribute in the DTD.

The reference does not need to be a physical file; it is simply a way to distinguish between namespaces. The reference should tell a person looking at the XML document where to find definitions of the element and attribute names using that particular namespace. Ronald Bourret maintains the Namespace FAQ at <http://www.rpbouret.com/xml/NamespacesFAQ.htm>¹⁶⁰.

D.6 What XML software is available?

Hundreds, possibly thousands, of programs. Details are no longer listed in this FAQ as they are now too many and are changing too rapidly to be kept up to date: see the XML Web pages at <http://xml.coverpages.org/>¹⁶¹ and watch for announcements on the mailing lists and newsgroups [p.5].

For a detailed guide to some examples of XML programs and the concepts behind them, see the editor's book *Understanding SGML and XML Tools*¹⁶².

Details of some XML software products are held on the XML Web pages¹⁶³. For browsers see the question on XML Browsers [p.8] and the details of the xml-dev mailing list [p.5] for software developers. Bert Bos keeps a list of some XML developments¹⁶⁴ in Bison, Flex, Perl, and Python. The long-established conversion and application development engines like Balise, Omnimark, and SGMLC all have XML capability and they all provide APIs.

Information for developers of Chinese XML systems can be found at the Chinese XML Now! website of Academia Sinica: <http://www.ascc.net/xml/>¹⁶⁵ This site includes an FAQ and test files.

D.7 What's my information? DATA or TEXT?

Some important distinctions exist between the major classes of XML applications and the way in which they are used:

Two classes of applications are usually referred to as 'document' and 'data' applications, and this is reflected in the software, which is usually (but not always) aimed at one class or the other. Document-style applications are in the nature of traditional publishers' work: text and images in a structured environment, with fonts and formatting; this includes Web pages as well as material destined for print like books and magazines. Data-style applications are found mostly in e-commerce and process or application control, with XML being used as a container for information being stored or passed between systems, usually unformatted and unseen by humans. There is a third major area, Web Development, whose requirements are often hybrid, and span the features of both document and data applications.

¹⁶¹<http://xml.coverpages.org/>

¹⁶²Flynn *Understanding SGML and XML Tools*.

¹⁶³<http://xml.coverpages.org/sgml-xml.html>

¹⁶⁴<http://www.w3.org/XML/notes.html>

¹⁶⁵<http://www.ascc.net/xml/>

Thousands of programs: too many to list here.

It depends on what you're using it for.

While in theory it would be possible to use a data-class software to write a novel, or a document-class software to create invoices, it would probably be severely suboptimal. Because of the nature of the information used by the two classes, data-class applications tend to use Schemas [p.22], and document-class applications tend to use DTDs [p.19], but there is a considerable degree of overlap.

The way in which XML gets used in these two classes is also divided in two: XML can be used manually or under program control. Manual usage means editing and maintaining the files with an editor, from the keyboard, seeing the information on the screen as you do so. This is suitable for individual documents, especially in the publishing field, and for developers working on single instances such as sample files. Manual processing also implies running production programs like formatters, converters, and database queries on a one-by-one basis, using the keyboard and mouse in the normal way. Much of the software for manual usage can be run from the command line, which makes it easy to use for one-off applications and for hidden applications like Web scripts. Programmable usage means writing programs which call on software services from APIs, libraries, or the network to handle XML files from inside the program. This is the normal method of operating for e-commerce applications, Web automation, and other process or application controls. There are libraries and APIs for many languages, including Java, C, and C++ as well as the usual scripting languages like Python, Perl, and Tcl.

In addition to these axes, there are two different ways of processing XML, memory-mapped or event-triggered, usually called the Document Object Model (DOM)¹⁶⁶ and the Simple API for XML (SAX)¹⁶⁷ respectively, after their best-known instantiations. Both use a model of document engineering based on the tree-like structure of hierarchical document markup known as a grove¹⁶⁸ (a collection of trees, effectively an in-memory map of the result of parsing the document markup), where every item of information from the outermost element down through every element and attribute to each snippet of unmarked text can be identified as a 'node'. For Schemas, a Post-Validation Infoset is defined, which specifies what information a parser should make available to the application.

Grossly oversimplified, a DOM-based application reads an entire XML document into memory and then provides programmable access to every node in the grove; whereas a SAX-based application reads the XML document, and events are triggered by the occurrence of nodes for which rules or actions have been programmed (actually it's more complex than that, and both methods share a lot of concepts in common). Both models provide an abstract API for constructing, accessing, and manipulating XML documents. A binding of the abstract API to a particular programming language provides a concrete API. Vendors provide concrete APIs which let you use either method to query and manipulate XML documents.

D.8 Do I have to change any of my server software to work with XML?

The only changes needed are to make sure your server serves up .xml, .css, .dtd, .xsl, and whatever other file types you will use as the correct MIME content (media) types.

Make sure
your server
sends XML files
as text/xml

¹⁶⁶<http://www.w3.org/TR/REC-DOM-Level-1>

¹⁶⁷<http://www.megginson.com/SAX/index.html>

¹⁶⁸<http://xml.coverpages.org/topics.html#groves>

The details of the settings are specified in RFC 3023¹⁶⁹. Most new versions of Web server software come preset.

All that is needed is to edit the `mime-types` file (or its equivalent: as a server operator you already know where to do this) and add or edit the relevant lines for the right media types. In some servers (eg Apache), individual content providers or directory owners may also be able to change the MIME types for specific file types from within their own directories by using directives in a `.htaccess` file. The media types required are:

- </> `text/xml` for XML documents which are 'readable by casual users';
- </> `application/xml` for XML documents which are 'unreadable by casual users';
- </> `text/xml-external-parsed-entity` for external parsed entities such as document fragments (eg separate chapters which make up a book) subject to the readability distinction of `text/xml`;
- </> `application/xml-external-parsed-entity` for external parsed entities subject to the readability distinction of `application/xml`;
- </> `application/xml-dtd` for DTD files and modules, including character entity sets.

The RFC has further suggestions for the use of the `+xml` media type suffix for identifying ancillary files such as XSLT (`application/xslt+xml`).

If you run scripts generating XHTML which you wish to be treated as XML rather than HTML, they may need to be modified to produce the relevant Document Type Declaration as well as the right media type if your application requires them to be validated.

D.9 Can I still use server-side inclusions?

Yes, so long as what they generate ends up as part of an XML-conformant file (ie either valid [D.2, p.35] or just well-formed [D.2, p.35]).

Yes, just make sure the output conforms to XML

Server-side tag-replacers like `shtml`, PHP, JSP, ASP, Zope, etc store almost-valid files using comments, Processing Instructions, or non-XML markup, which gets replaced at the point of service by text or XML markup. It is unclear why some of these systems continue to use non-XML markup. There are also some XML-based preprocessors for formats like XVRL¹⁷⁰ (eXtensible Value Resolution Language) which resolve specialised references to external data and output a normalised XML file.

D.10 Can I (and my authors) still use client-side inclusions?

The same rule applies as for server-side [p.40] inclusions, so you need to ensure that any embedded code which gets passed to a third-party engine (eg calls to SQL, VB, Java, etc) does not contain any characters which might be misinterpreted as XML markup (ie no angle brackets or ampersands). Either use a CDATA marked section to avoid your XML application parsing the embedded code, or use the standard `<`, and `&` character entity references instead.

Yes, just make sure the output conforms to XML

¹⁶⁹<ftp://ftp.isi.edu/in-notes/rfc3023.txt>

¹⁷⁰<http://www.xvrl.org>

D.11 I'm trying to understand the XML Spec: why does it have such difficult terminology?

It has to be formal to be accurate.

For implementation to succeed, the terminology needs to be precise. Design goal eight of the specification tells us that 'the design of XML shall be formal and concise'. To describe XML, the specification therefore uses formal language drawn from several fields, specifically those of text engineering, international standards and computer science. This is often confusing to people who are unused to these disciplines because they use well-known English words in a specialised sense which can be very different from their common meanings—for example: grammar, production, token, or terminal.

The specification does not explain these terms because of the other part of the design goal: the specification should be concise. It doesn't repeat explanations that are available elsewhere: it is assumed you know this and either know the definitions or are capable of finding them. In essence this means that to grok the fullness of the spec, you do need a knowledge of some SGML and computer science, and have some exposure to the language of formal standards.

Sloppy terminology in specifications causes misunderstandings and makes it hard to implement consistently, so formal standards have to be phrased in formal terminology. This FAQ is not a formal document, and the astute reader will already have noticed it refers to 'element names' where 'element type names' is more correct; but the former is more widely understood.

Those new to the terminology may find it useful to read something like the *Gentle Introduction to XML*¹⁷¹ or *XML: The Annotated Specification*¹⁷².

D.12 I have to do an overview of XML for my manager/client/investor/advisor. What should I mention?

Non-proprietary multi-purpose flexible markup

Tad McClellan writes:

- <> XML is *not* a markup language. XML is a 'metalanguage', that is, it's a language that lets you define *your own* markup languages (see definition (p.1)).
- <> XML *is* a markup language (two (seemingly) contradictory statements one after another is an attention-getting device that I'm fond of), *not* a programming language. XML is data: it does not 'do' anything, it has things done to it.
- <> XML is non-proprietary: your data cannot be held hostage by someone else.
- <> XML allows multi-purposing of your data.
- <> Well-designed XML applications most often separate 'content' from 'presentation'. You should describe what something *is* rather than what something *looks like* (the exception being data content which never gets presented to humans).

¹⁷¹Sperberg-McQueen/Burnard.

¹⁷²DuCharme.

Saying ‘the data is in XML’ is a relatively useless statement, similar to saying ‘the book is in a natural language’. To be useful, the former needs to specify ‘we have used XML to define our own markup language’ (and say what it is), similar to specifying ‘the book is in French’.

A classic example of multipurposing [p.41] and separation [p.41] that I often use is a pharmaceutical company. They have a large base of data on a particular drug that they need to publish as:

- </> reports to the FDA;
- </> drug information for publishers of drug directories/catalogs;
- </> ‘prescribe me!’ brochures to send to doctors;
- </> little pieces of paper to tuck into the boxes;
- </> labels on the bottles;
- </> two pages of fine print to follow their ad in Reader’s Digest;
- </> instructions to the patient that the local pharmacist prints out;
- </> etc.

Without separation of content and presentation, they need to maintain essentially identical information in 20 places. If they miss a place, people die, lawyers get rich, and the drug company gets poor. With XML (or SGML), they maintain one set of carefully validated information, and write 20 programs to extract and format it for each application. The same 20 programs can now be applied to all the hundreds of drugs that they sell.

In the Web development area, the biggest thing that XML offers is fixing what is wrong with HTML:

- </> browsers allow non-compliant HTML to be presented;
- </> HTML is restricted to a single set of markup (‘tagset’).

If you let broken HTML work (be presented), then there is no motivation to fix it. Web pages are therefore tag soup that are useless for further processing. XML specifies that processing must not continue if the XML is non-compliant, so you keep working at it until it complies. This is more work up front, but the result is not a dead-end.

If you wanted to mark up the names of things: people, places, companies, etc in HTML, you don’t have many choices that allow you to distinguish among them. XML allows you to name things as what they are:

```
<person>Charles Goldfarb</person> worked at <company>IBM</company>
```

gives you a flexibility that you don’t have with HTML:

```
<B>Charles Goldfarb</B> worked at <B>IBM</B>
```

With XML you don’t have to shoe-horn your data into markup that restricts your options.

D.13 Is there a conformance test suite for XML processors?

James Clark has a collection of test cases for testing XML parsers at <http://www.jclark.com/xml/>¹⁷³ which includes a conformance test.

Yes, see
<http://www.oasis-open.org/committees/xmltest/testsuite.htm>

Mary Brady, OASIS XML Conformance TC Chair writes:

A much larger and more comprehensive suite is the NIST/OASIS Conformance Test Suite, available from <http://www.oasis-open.org/committees/xmltest/testsuite.htm>¹⁷⁴, which contains contributions from James Clark, OASIS and NIST, Sun, and Fuji Xerox.

Carmelo Montanez writes:

NIST has developed a number of XSLT/XPath tests, which will be part of the official OASIS XSLT/XPath suite (not yet released). These tests are available from our web site at <http://xw2k.sdct.itl.nist.gov/xml/index.html>¹⁷⁵ (click on 'XSL Testing'). The expected output may be slightly different from one implementation to another. The OASIS XSLT technical committee has a solution for that problem, however our tests do not yet implement such solution. Please forward any comments to carmelo@nist.gov.

Jon Noring writes:

For those who are interested, I took the current and complete Unicode 3.0 'cast' of characters and their hex codes, and created a simple XML document of it to test XML browsers for Unicode conformity. It is not finished yet—I need to add comments and to fix the display of rtl characters (ie Hebrew, Arabic). It is found at: <http://www.windspun.com/unicode-test/unicode.xml>¹⁷⁶. It is quite large, almost 900K in size, so be prepared. IE5 renders many of the characters in this XML document—and for the ones it does render it appears to do so correctly. I look forward to when Opera will do likewise. I haven't tested the current version of Mozilla/Netscape for Unicode conformity.

D.14 I've already got SGML DTDs: how do I convert them for use with XML?

There are numerous projects to convert common or popular SGML DTDs to XML format (for example the TEI DTD¹⁷⁷, both Lite and full versions, is available in both SGML and XML, in Schema and DTD formats).

Edit by hand or use software like Near+Far Designer.

¹⁷³<http://www.jclark.com/xml/>

¹⁷⁴<http://www.oasis-open.org/committees/xmltest/testsuite.htm>

¹⁷⁵<http://xw2k.sdct.itl.nist.gov/xml/index.html>

¹⁷⁶<http://www.windspun.com/unicode-test/unicode.xml>

¹⁷⁷<http://www.tei-c.org/>

Seán McGrath writes:

1. No equivalent of the SGML Declaration. So keywords, character set etc are essentially fixed;
2. Tag minimisation is not allowed, so `<!ELEMENT x - O (A,B)>` becomes `<!ELEMENT X (A,B)>` and `<!ELEMENT x - O EMPTY>` becomes `<!ELEMENT X EMPTY>`;
3. #PCDATA must only occur at the extreme left (ie first) in an OR model, eg `<!ELEMENT x - - (A|B|#PCDATA|C)>` (in SGML) becomes `<!ELEMENT x (#PCDATA|A|B|C)*>`, and `<!ELEMENT x (A, #PCDATA)>` is illegal;
4. No CDATA, RCDATA elements (declared content);
5. Some SGML attribute types are not allowed in XML eg NUTOKEN;
6. Some SGML attribute defaults are not allowed in XML eg CONREF;
7. Comments cannot be inline to declarations like `<!ELEMENT x - - (A,B) -- an SGML comment in a declaration -->`;
8. A whole bunch of SGML optional features are not present in XML: all forms of tag minimisation (OMITTAG, DATATAG, SHORTREF, etc); Link Process Definitions; Multiple DTDs per document; and many more: see <http://www.w3.org/TR/NOTE-sgml-xml-971215> for the list of bits of SGML that were removed for XML;
9. And (nearly) last but not least, no CONCUR!
10. There are some important differences between the internal and external subset portion of a DTD in XML: Marked Sections can only occur in the external subset; and Parameter Entities must be used to replace entire declarations in the internal subset portion of a DTD, eg the following is invalid XML:

```
<!DOCTYPE x [  
<!ENTITY % modelx "(A|B)*">  
<!ELEMENT x %modelx;>  
>  
<x></x>
```

For more information, see *XML by Example: Building E-Commerce Applications*⁴.

⁴Seán McGrath.

D.15 How do I include one DTD (or fragment) in another?

This works exactly the same as for SGML. First you declare the entity you want to include, and then you reference it by name as a parameter:

Use a parameter entity, same as for SGML

```
<!ENTITY % mylists SYSTEM "dtds/listfrag.ent">
...
%mylists;
```

Such declarations traditionally go all together towards the top of the main DTD file, where they can be managed and maintained, but this is not essential so long as they are declared before they are used. You use Parameter Entity Syntax for this (the percent sign) because the file is to be included at DTD compile time, not when the document instance itself is parsed.

Note that a URL is compulsory in XML as the System Identifier for all external file references: standard rules for dereferencing URLs apply (assume the same method, server, and directory as the containing document). A Formal Public Identifier can also be used, following the same rules as elsewhere [D.2, p.35].

D.16 What's the story on XML and EDI?

Electronic Data Interchange has been used in e-commerce for many years to exchange documents between commercial partners to a transaction. It requires special proprietary software and is prohibitively expensive to implement for small and medium-sized enterprises. There are moves to enable EDI documents to travel inside XML, as well as proposals to replace the existing EDI formats with XML ones. There are guideline documents at <http://www.eccnet.com/xmledi/guidelines-styled.xml> and <http://www.geocities.com/WallStreet/Floor/5815/guide.htm>.

Getting there:
still needs
more work
and
agreement.

Probably the biggest effect on EDI will be the rise of standardisation attempts for XML business documents and transactions. A standard jointly sponsored by OASIS and United Nations/CEFACT is ebXML¹⁷⁸ (Electronic Business XML) which provides Schemas for the common commercial transaction document types. Normal office documents (letters, reports, spreadsheets, etc) are already being done using the materials under the charge of the OASIS Open Office XML Formats TC, detailed above [p.10].

¹⁷⁸<http://www.ebxml.org/>

E References

E.1 Bibliography

This list covers only documents directly referenced in this FAQ.

DuCharme, Bob: XML: The Annotated Specification. Upper Saddle River, NJ: Prentice Hall PTR, 1999 [⟨URL: http://www.snee.com/bob/xmlann⟩](http://www.snee.com/bob/xmlann), ISBN 0-13-082676-6

Flynn, Peter: Understanding SGML and XML Tools. Boston: Kluwer, 1998 [⟨URL: http://imbolc.ucc.ie/~pflynn/books/sgml/⟩](http://imbolc.ucc.ie/~pflynn/books/sgml/), ISBN 0-7923-8169-6

Flynn, Peter: Making more use of markup. in: SGML'95. Boston, MA, December 1995 [⟨URL: http://imbolc.ucc.ie/~pflynn/articles/moreuse.html#meaning⟩](http://imbolc.ucc.ie/~pflynn/articles/moreuse.html#meaning) 158-167

Maler, Eve/el Andaloussi, Jeanne: Developing SGML DTDs: From Text to Model to Markup. Upper Saddle River, NJ: Prentice Hall PTR, 1995 [⟨URL: http://www.amazon.com/exec/obidos/tg/detail/-/0133098818/qid=1104447963/sr=8-1/ref=sr_8_xs_ap_i1_xgl14/002-9386245-9385639?v=glance&s=books&n=507846⟩](http://www.amazon.com/exec/obidos/tg/detail/-/0133098818/qid=1104447963/sr=8-1/ref=sr_8_xs_ap_i1_xgl14/002-9386245-9385639?v=glance&s=books&n=507846), ISBN 0133098818

McGrath, Seán: XML by Example: Building E-Commerce Applications. Upper Saddle River, NJ: Prentice Hall PTR, 1998 [⟨URL: http://www.amazon.com/exec/obidos/tg/detail/-/0139601627/qid=1104449400/sr=8-1/ref=sr_8_xs_ap_i1_xgl14/002-9386245-9385639?v=glance&s=books&n=507846⟩](http://www.amazon.com/exec/obidos/tg/detail/-/0139601627/qid=1104449400/sr=8-1/ref=sr_8_xs_ap_i1_xgl14/002-9386245-9385639?v=glance&s=books&n=507846), ISBN 0139601627

Salminen, Airi/Tompa, Frank: Requirements for XML Document Database Systems. in: ACM Symposium on Document Engineering. Atlanta, GA, November 2001 [⟨URL: http://db.uwaterloo.ca/~fwtompa/.papers/xmldb-desiderata.pdf⟩](http://db.uwaterloo.ca/~fwtompa/.papers/xmldb-desiderata.pdf)

Sperberg-McQueen, Michael/Burnard, Lou: Gentle Introduction to XML. Oxford, Providence, Charlottesville, Bergen: Text Encoding Initiative Consortium, 2002 [⟨URL: http://www.tei-c.org/Guidelines2/gentleintro.pdf⟩](http://www.tei-c.org/Guidelines2/gentleintro.pdf)

Truss, Lynne: Eats, Shoots & Leaves: The Zero-Tolerance Approach to Punctuation. London: Profile Books, 2003 [⟨URL: http://www.amazon.com/exec/obidos/tg/detail/-/1592400876/qid=1104449308/sr=8-1/ref=pd_csp_1/002-9386245-9385639?v=glance&s=books&n=507846⟩](http://www.amazon.com/exec/obidos/tg/detail/-/1592400876/qid=1104449308/sr=8-1/ref=pd_csp_1/002-9386245-9385639?v=glance&s=books&n=507846), ISBN 1-86197-612-7

There is a much larger SGML and XML bibliography at <http://xml.coverpages.org/biblio.html>.

E.2 How far are we going?

To infinity and beyond!

```
Date: Fri, 09 Jul 1999 14:26:17 -0500 (EST)
From: The Internet Oracle <oracle@cs.indiana.edu>
Subject: The Oracle replies!
To: pflynn@imbolc.ucc.ie
X-Planation: X-Face can be viewed with ftp.cs.indiana.edu:/pub/faces.
```

The Internet Oracle has pondered your question deeply.
Your question was:

```
> Oh Oracle most wise, all-seeing and all-knowing,
> in thy wisdom grant me a response to my request:
>
> Is XML really going to cut the mustard?
```

And in response, thus spake the Oracle:

```
} Well, since XML is a subset of SGML, and SGML
} has a <cut mustard> tag, I'd have to say yes.
}
} You owe the Oracle a BlFF parser.
```

For the SGML-curious among our readers, that's:

```
<!element cut - - (#pcdata)>
<!attlist cut mustard (mustard) #REQUIRED>
<!-- :-) -->
```

E.3 Revision history

0.0 (1996-12-27) First test. Unpublished.

0.1 (1997-01-31) First draft. Sample questions devised by participants.

0.2 (1997-02-03) Revised draft. Additional questions and answers.

0.3 (1997-02-17) Extensive revision following comments from the group. Changes to markup and organization.

0.4 (1997-02-23) Minor editorial changes

0.5 (1997-04-01) Added Multidoc Pro as SGML browser; question on XML math; fixed ambiguity in explanation of NETs; added JUMBO; ERB changes of March 26; more details of linking and tools; adding element declaration minimisation to the forbidden list.

1.0 (1997-05-01) Added reference to ToC and printed URLs; added disclaimer at A6; combined old A11 with A5 to explain SGML/XML/HTML; clarified explanation of XML not replacing HTML at C1; added new course and conference at (new) A11; clarified B1, C4, C8; added FPI server at C12; removed examples in C13.

- 1.1 (1997-10-01)** No more minimisation parameters in element declarations; parsers must now pass all white-space to the application; everything is now case-sensitive, including all markup; a new proposal for stylesheets: XSL, which combines DSSSL and CSS in an XML format; Java[Script] and and metadata and their use in XML; updated list of software; first XML book is published; new public mailing list XML-L
- 1.2 (1998-02-01)** Added a Mac icon (thanks to Martin Winter and others); removed Draft from references to the spec; changed revision colours; the RMD is gone: replaced references to it with standalone; updated some broken URLs; [1.21] minor edits to URLs and updates on translation; added XUA to details of MIME types.
- 1.3 (1998-06-01)** Removed the math plugin (Linux Netscape is broken and refused to elide it); updated list of events (need more); fixed some broken URLs; added Spanish and Korean translations and the Annotated Spec; updated details of MS/NS browser development; clarified the use of FPI vs SysID; updated link to Feb 10 Rec Spec; added pointers to the SGML Decl for XML; updated references to XLink and XPointer; corrected a reference to ancient Sumerian writing; clarified the need for conversion of HTML DTDs to XML.
- 1.4 (1998-10-01)** Added maintainer's email address under Availability; Added note about ISO representation and voting on standards; added Greek translation; updated details of conferences; changed the URL for the new SGML/XML Web Pages; updated details of browsers; corrected reference to the SGML omitted features from XML; updated details of converting HTML to XML; added mention of comp.text.xml; extended the questions on graphics and how to use XML with current browsers; added questions on DOM, conformance testing, DTD includes, SGML DTDs into XML, EDI; (1.41) corrected errors in MIME types, URLs, SDD, and images.
- 1.5 (1999-06-01)** Added new XML mailing lists in Italian and in French; added details of developer resources in Chinese; two more translations under way (Chinese and Czech); updated links to the question on DTDs; added question on the use of Java to generate and manage XML; added question on when to use attributes and when to use element markup; added question on the use of XML syntax to describe DTD data (schemas); expanded on the explanation of the use of formal language in the spec; added question on the difference between XML and C++; separated information on XML versions of HTML into a separate question.
- 1.6 (2000-07-01)** Added French and Czech translations and a Finnish mailing list, and reorganised the list of translations; updated URLs for newsgroups; clarified reference to Unicode; reworded question on terminology; added more links to the question on conformance testing; corrected error in content model example for mixed content; updates to the question on stylesheets; Minor edits to the question on software; major changes to the question on servers and media types; updated question on XML Schemas; added new question on 'executing' XML 'programs'; replaced the math example with one less likely to distress the gentle susceptibilities of some readers; added a new question on knowing SGML/HTML before XML.
- 2.0 (2001-06-01)** DTD changed from DocBook SGML to QAML XML; removed query form due to abuse; most questions revised and in some cases rewritten; updated references to new versions of associated standards, recommendations, and working drafts; added pointer to Jon Noring's Unicode test page and NIST's XSLT/XPath test suite; updated Eve Maler's links to the DTD for the spec; added warnings on spelling

and punk chew asian; added question on namespaces; fixed bug in question on stylesheets; inserted explanation of 'document' vs 'data' software; added new mailing list on XSL:FO; updated Robin Cover's URL throughout; updated the question on media types for RFC 3023; Extended question of graphics to cover SVG. For 2.01 there were minor typos, some updated links (to recent versions of the standards, and in the section on More Information), and a few wording changes. Thanks to James Cummings for a very thorough proofread. Editing was done using GNU Emacs and psgml-mode.

2.1 (2002-01-01) Added Humanities mailing list [p.5]; added more references for XML and databases [p.23]; added the Namespaces FAQ [p.38]; corrected some misunderstandings in character encodings [p.19]; changed the editor's email address; added a new question on root elements [p.22]; updated the XLink [p.24] to W3C Recommendation; updated the SGML FAQ address [p.1]; fixed some broken links; added translations into German [p.iv] and Amharic [p.iv]; minor revisions to some wording. Editing this time was done in epcEdit 1.02¹⁷⁹. V2.11 includes new material on expectations and XML browsers [p.8], the removal of a mailing list, and a few corrections to typos and links. Thanks to Seán Cannon and Dave^Nikki for debugging the CSS style-sheet.

3.0 (2003-01-01) Added information on Office Applications [p.10] including Corel, Microsoft, and Sun (to keep alphabetical order :-); updated details of conferences and training [p.4]; updated browser [p.8] details; reworded a few ungainly sentences; removed some obsolete URLs (mostly for *nice idea* sites which died); changed the phrasing of the question on databases [p.23]; added details on how to do standalone validation to the question on parsing [p.30] (thanks to Bill Rayer); added question on how to present XML to management [p.41] (thanks to Tad McClellan); the questions on APIs and the DOM have been subsumed into the question on software [p.38], which has been extensively rewritten; added yet more explanation to the section on Unicode [p.18]; 3.01 fixes minor typos; 3.02 adds updated dates for 2004 events.

3.01 (2004-01-01) Minor typographic changes

3.02 (2004-01-12) Added updates for 2004 events

¹⁷⁹<http://www.epcedit.com>