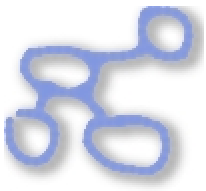
XML in 10 points

(7, really...)

***XML, XLink, Namespace, DTD, Schema, CSS, XHTML,*... If you are new to XML, it may be hard to know where to begin. This summary in 10 points attempts to capture enough of the basic concepts to enable a beginner to see the forest through the trees. And if you are giving a presentation on XML, why not start with these 10 points? They are hereby offered for your use.**

1. XML is a method for putting structured data in a text file



For "structured data" think of such things as spreadsheets, address books, configuration parameters, financial transactions, technical drawings, etc. Programs that produce such data often also store it on disk, for which they can use either a binary format or a text format. The latter allows you, if necessary, to look at the data without the program that produced it. XML is a set of rules, guidelines, conventions, whatever you want to call them, for designing text formats for such data, in a way that produces files that are easy to generate and read (by a computer), that are unambiguous, and that avoid common pitfalls, such as lack of extensibility, lack of support for internationalization/localization, and platform-dependency.

2. XML looks a bit like HTML but isn't HTML

Like HTML, XML makes use of *tags* (words bracketed by '<' and '>') and *attributes* (of the form `name="value"`), but while HTML specifies what each tag & attribute means (and often how the text between them will look in a browser), XML uses the tags only to delimit pieces of data, and leaves the interpretation of the data completely to the application that reads it. In other words, if you see "<p>" in an XML file, don't assume it is a paragraph. Depending on the context, it may be a price, a parameter, a person, a p... (b.t.w., who says it has to be a word with a "p"?)

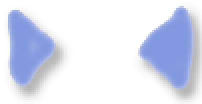
3. XML is text, but isn't meant to be read

XML files are text files, as I said above, but even less than HTML are they meant to be read by humans. They are text files, because that allows experts (such as programmers) to more easily *debug* applications, and in emergencies, they can use a simple text editor to fix a broken XML file. But the rules for XML files are much stricter than for HTML. A forgotten tag, or a an attribute without quotes makes the file unusable, while in HTML such practice is often explicitly allowed, or at least tolerated. It is written in the official XML specification: applications are not *allowed* to try to second-guess the creator of a broken XML file; if the file is broken, an application has to stop right there and issue an error.

4. XML is a family of technologies

There is ***XML 1.0***, the specification that defines what "tags" and "attributes" are, but around XML 1.0, there is a growing set of optional modules that provide sets of tags & attributes, or guidelines for specific tasks. There is, e.g., ***Xlink*** (still in development as of November 1999) which describes a standard way to add hyperlinks to an XML file. ***XPointer*** & ***XFragments*** (also still being developed) are syntaxes for pointing to parts of an XML document. (An Xpointer is a bit like a URL, but instead of pointing to documents on the Web, it points to pieces of data inside an XML file.) ***CSS***, the style sheet language, is applicable to XML as it is to HTML. ***XSL*** (autumn 1999) is the advanced language for expressing style sheets. It is based on ***XSLT***, a transformation language that is often useful outside XSL as well, for rearranging, adding or deleting tags & attributes. The ***DOM*** is a standard set of function calls for manipulating XML (and HTML) files from a programming language. ***XML Namespaces*** is a specification that describes how you can associate a URL with every single tag and attribute in an XML document. What that URL is used for is up to the application that reads the URL, though. (RDF, W3C's standard for metadata, uses it to link every piece of metadata to a file defining the type of that data.) ***XML Schemas 1*** and ***2*** help developers to precisely define their own XML-based formats. There are several more modules and tools available or under development. Keep an eye on W3C's technical reports page.

5. XML is verbose, but that is not a problem



Since XML is a text format, and it uses tags to delimit the data, XML files are nearly always larger than comparable binary formats. That was a conscious decision by the XML developers. The advantages of a text format are evident (see 3 above), and the disadvantages can usually be compensated at a different level. Disk space isn't as expensive anymore as it used to be, and programs like zip and gzip can compress files very well and very fast. Those programs are available for nearly all platforms (and are usually free). In addition, communication protocols such as modem protocols and HTTP/1.1 (the core protocol of the Web) can compress data on the fly, thus saving bandwith as effectively as a binary format.

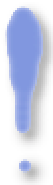
6. XML is new, but not that new

Development of XML started in 1996 and it is a W3C standard since February 1998, which may make you suspect that this is rather immature technology. But in fact the technology isn't very new. Before XML there was SGML, developed in the early '80s, an ISO standard since 1986, and widely used for large documentation projects. And of course HTML, whose development started in 1990. The designers of XML simply took the best parts of SGML, guided by the experience with HTML, and produced something that is no less powerful than SGML, but vastly more regular and simpler to use. Some evolutions, however, are hard to distinguish from revolutions... And it must be said that while SGML is mostly used for technical documentation and much less for other kinds of data, with XML it is exactly the opposite.

7, 8, 9...

These I don't know yet.

10. XML is license-free, platform-independent and well-supported



By choosing XML as the basis for some project, you buy into a large and growing community of tools (one of which may already do what you need!) and engineers experienced in the technology. Opting for XML is a bit like choosing SQL for databases: you still have to build your own database and your own programs/procedures that manipulate it, but there are many tools available and many people that can help you. And since XML, as a W3C technology, is license-free, you can build your own software around it without paying anybody anything. The large and growing support means that you are also not tied to a single vendor. *XML isn't always the best solution, but it is always worth considering.*

Bert Bos
Created 27 Mar 1999 (last update: $Date: 2000/05/26 15:48:52 $)