

# **XML Primer**

Eleni Stroulia

## **Markup languages**

- RTF
- Tex/Latex
- Postscript

<http://matwww.ee.tut.fi/hmintro/chap3.html>

# Procedural vs. Declarative markup

- Browsers render HTML documents by interpreting the markup commands interspersed in the document text
  - They produce high quality results because they are robust to erroneous markups
- However HTML markup is procedural
  - The logical structure of a document is not expressed in HTML
    - Extracting data from HTML documents is extremely brittle process
  - Changes in the style require revising all markup commands
    - If the style is completely and explicitly captured in CSS, this is alleviated
- XML is a declarative markup language
  - It declares the logical structure of a document
  - CSS or XSL can be used to specify the style to be used for rendering

11-07

3

## HTML vs. XML

- HTML enables aesthetically pleasing rendering of information for human consumption
- XML simplifies data interchange.
  - It is easier to agree on a small set of core types of data to exchange and to share than it is to agree on platforms, languages and APIs.
    - Easier to design, agree and share a DB schema than a presentation
- XML enables smart code.
  - XML documents are structured to identify important pieces of information and the relationships between them
    - XML tags correspond to important domain-specific abstractions
  - There is substantial tool support for processing XML documents; thus using XML data is fairly simple
- XML enables smart searches.
  - Search engines still often return erroneous results; XML documents can provide additional semantics to natural language text
    - Instead of looking at the content only, look at the tags; more thought has gone to choosing the “right” word for this smaller closed set of terms

11-07

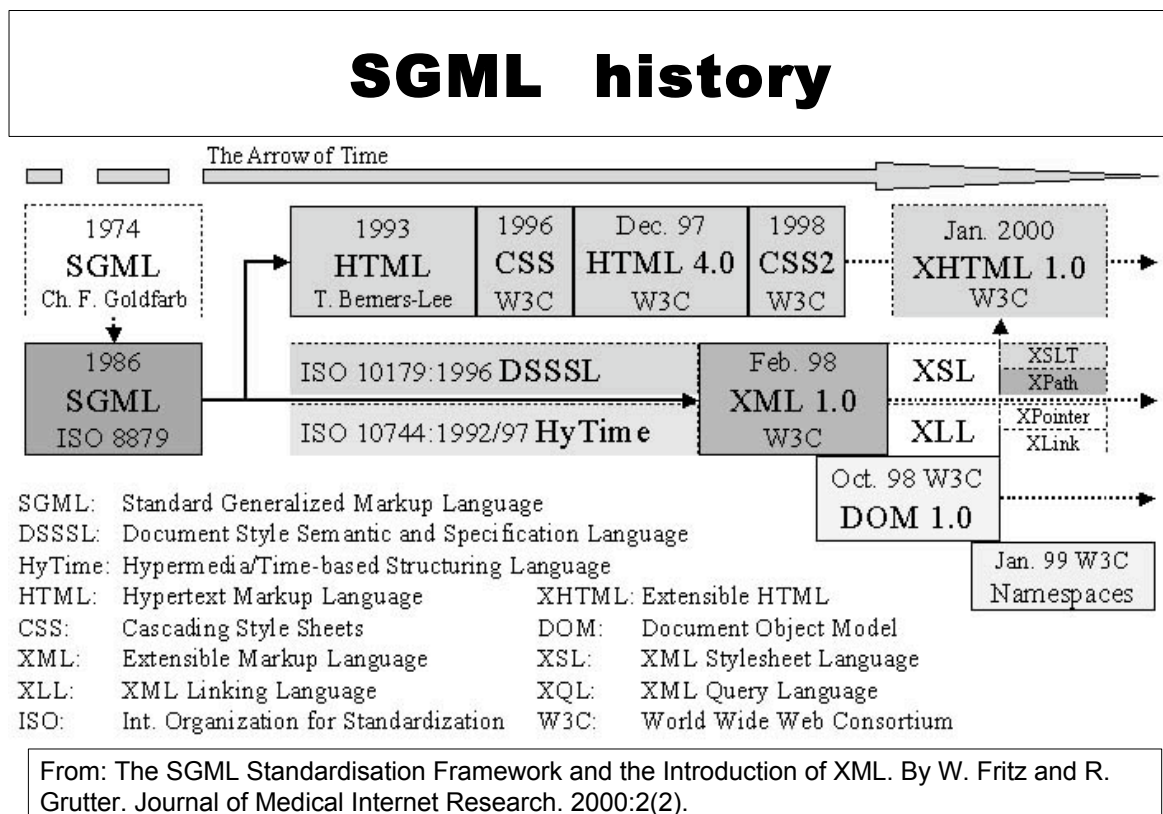
4

# SGML

- Standard Generalized Markup Language
  - "Sounds great, maybe later"
- Adopted by ISO in 1986
- Large, complicated, expensive
- Meta-language
  - Defines how to define markup languages

11-07

5



11-07

6

# **XML in 10 points (a)**

<http://www.w3.org/XML/1999/XML-in-10-points.html>

1. XML is for structuring data
  - is not a programming language
  - is extensible, platform-independent, and supports internationalization and localization
2. XML looks a bit like HTML
  - uses tags only to delimit pieces of data; it leaves the data interpretation completely to the application that reads it;
  - the XML specification requires a parser to reject any XML document that doesn't follow the basic rules.
3. XML is text, but isn't really meant to be read
  - people, if necessary, can look at the data
  - the rules for XML files allow fewer variations
4. XML is verbose by design
  - nearly always XML is larger than binary formats, but disk space is less expensive; compression is effective and efficient and can be done on the fly

11-07

7

# **XML in 10 points (b)**

<http://www.w3.org/XML/1999/XML-in-10-points.html>

5. XML is a family of technologies
  - XML 1.0 is the specification that defines what "tags" and "attributes" are.
  - XLink describes a standard way to add hyperlinks to an XML file.
  - XPointer is a syntax in development for pointing to parts of an XML document.
  - CSS, the style sheet language, is applicable to XML as it is to HTML.
  - XSL is the advanced language for expressing style sheets. It is based on XSLT, a transformation language used for rearranging, adding and deleting tags and attributes.
  - DOM is a standard set of function calls for manipulating XML (and HTML) files from a programming language.
  - XML Schemas 1 and 2 help developers to precisely define the structures of their own XML-based formats.

11-07

8

# XML in 10 points (c)

<http://www.w3.org/XML/1999/XML-in-10-points.html>

6. XML is new, but not that new
  - Development of XML started in 1996 and it has been a W3C Recommendation since February 1998
7. XML leads HTML to XHTML
  - XHTML has many of the same elements as HTML. The syntax has been changed slightly to conform to the rules of XML
    - XHTML allows "<p>", but not "<br>";
    - it also adds meaning to that syntax ("<p>" stands for "paragraph")
8. XML is modular
  - It allows to define a new document format by combining and reusing other formats through the namespace mechanism.
  - XML Schema has a similar feature.
9. XML is the basis for RDF and the Semantic Web
10. XML is license-free, platform-independent and well-supported

11-07

9

## Document- vs. Data-centric XML

- Document-centric  $\Rightarrow$  structuring documents for human consumption
- Data-centric  $\Rightarrow$  intermediate representation format for data exchange among applications (<http://odfi.org/>)
- Data-centric XML
  - Has more markup (more detailed structure description - less room for human interpretation)
  - Has machine-generated information (usually produced as “serialization” of application back-end data)
  - Has highly-structured tag organization
  - Usually lives less long (document-centric information is organization/process description, does not change so often; data are specific to a process instance)

11-07

10

# XML Instances

11-07

11

## Parts of an XML document

- Prolog: a set of XML declarations
  - If there is one, it must be the first thing in the document.
  - The declaration can contain up to three name-value pairs
    - The version is the version of XML used; currently 1.0 or 1.1
    - The encoding is the character set used in this document.
    - Standalone can be either yes or no; defines whether this document can be processed without reading any other files.
- An XML document must be contained in a single element, the root element
- A tag is the text between the left angle bracket (<) and the right angle bracket (>).
- An element is the starting tag, the ending tag, and everything in between.
- An attribute is a name-value pair inside the starting tag of an element.

11-07

12

# XML Rules

- Elements can't overlap
- End tags are required
- Elements are case sensitive
  - Names can contain letters, numbers, and other characters
  - Names must not start with a number or punctuation character
  - Names must not start with the letters xml (or XML or Xml ..)
  - Names cannot contain spaces
- Elements can have element content, mixed content, simple content, or empty content and attributes

11-07

13

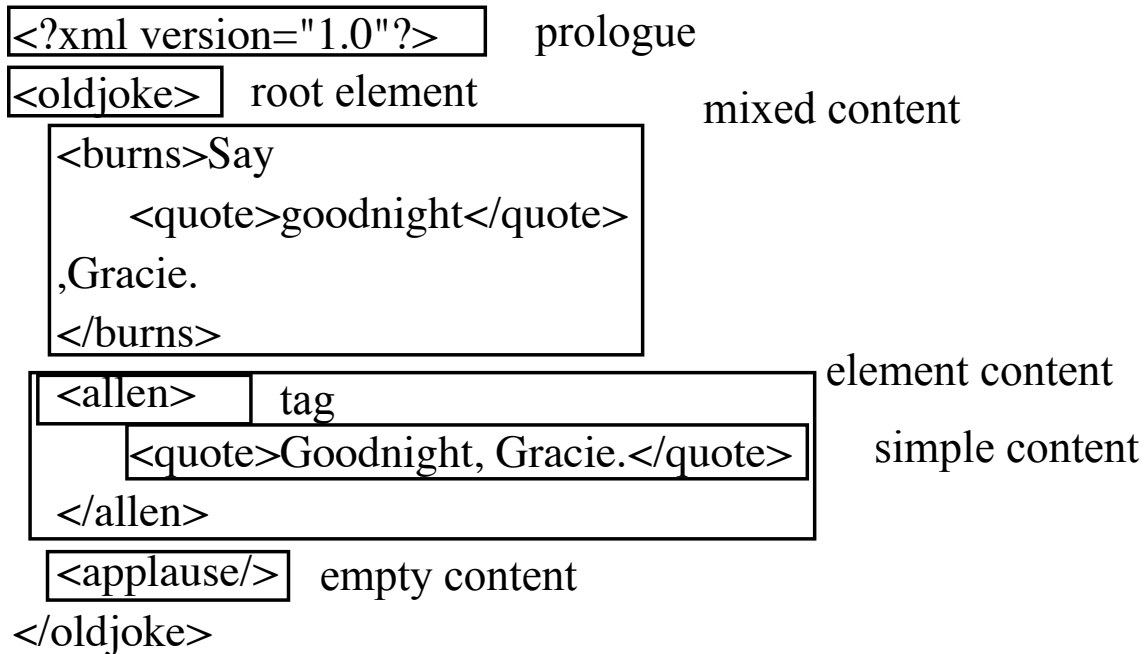
# XML Rules

- Attributes must have quoted values
  - either single or double quotes can be used.
    - `<person sex="female">`
    - `<person sex='female'>`
    - `<gangster name='George "Shotgun" Ziegler'>`
    - `<gangster name="George 'Shotgun' Ziegler">`

11-07

14

# An XML document



11-07

15

## Modeling in XML

- Should it be an element or an attribute?
  - If it is “data” then it should probably be an element
  - If it is meta-data then it should probably be an attribute
- Technically speaking...
  - attributes cannot describe structures (child elements can)
  - attributes are more difficult to manipulate by program code
  - attribute values are not easy to test against a Document Type Definition (DTD)

11-07

16



# **XML content specification**

11-07

17

## **DTD**

- Elements
- Tags
- Attributes
- Entities (~= macros)
  - &lt;
- PCDATA: Parsed character data
- CDATA: Character data (not parsed)
- Comments can appear anywhere in the document
  - begin with <!-- and end with -->

11-07

18

```

<!-- oldjokes.dtd -->
<!ELEMENT oldjoke (burns+, allen, applause?)>
<!ELEMENT burns (#PCDATA | quote)*>
<!ELEMENT allen (#PCDATA | quote)*>
<!ELEMENT quote (#PCDATA)*>
<!ELEMENT applause EMPTY>

<!ENTITY % personcontent "#PCDATA | quote">
<!ELEMENT burns (%personcontent;)*>
<!ELEMENT allen (%personcontent;)*>

<!-- ATTLLIST oldjoke
name ID #REQUIRED
label CDATA #IMPLIED
status ( funny | notfunny ) 'funny'>
name - type - default

```

11-07

19

```

<?xml version="1.0" encoding="iso-8859-1"? standalone="yes">
<!DOCTYPE library [
<!-- ELEMENT book (isbn, title, author+, character+) -->
<!-- ATTLLIST author aid ID #IMPLIED -->
<!-- ATTLLIST book bid ID #IMPLIED -->
<!-- ATTLLIST character cid ID #IMPLIED -->
]>
<library>
  <book bid="book_0836217462">
    <isbn>0836217462</isbn>
    <title>Being a Dog Is a Full-Time Job</title>
    <author href="author_Charles-M.-Schulz"/>
    <character href="character_Snoopy"/>
  </book>
  <book bid="book_0836217461">
    <isbn>0836217468</isbn>
    <title>Being a Dog Is a Full-Time Job also</title>
    <author href="author_Charles-M.-Schulz"/>
    <character href="character_Snoopy"/>
  </book>
  <author aid="author_Charles-M.-Schulz">
    ....
  </author>
  <character cid="character_Snoopy">
    ...
  </character>
  ....
</library>

```

11-07

## The id-href attributes

To eliminate duplicate information

- Put the information to be reused once in an element
- Add an “id” attribute to this element
- Refer to this element with the “href” attribute

20

# Namespaces

- XML's power comes from its flexibility; anybody can define tags to describe data.
- This also results in a problem: the same name may be used for different elements (with different semantics)
- Namespaces provide contexts to disambiguate the semantics of elements
  - To use a namespace, you define a namespace prefix and map it to a particular string: `xmlns:addr="http://www.xyz.com/addresses/"`
  - Then `<addr:name><title>Mrs.</title> ... </addr:name>` is unambiguous
  - The string in a namespace definition is just a string, although they look like URLs
  - The content of a namespace-prefixed element is assumed to belong to the parent namespace, unless stated otherwise
  - Attributes can also be prefixed with a namespace

11-07

21

## XML schema vs. DTD

- XML schemas use XML syntax.
  - an XML schema is an XML document
- XML schemas support data types:
  - All of the original data types from DTDs (IDs and ID references)
  - Also integers, floating point numbers, dates, times, strings, URLs, and others
- XML schemas are extensible:
  - You can create complex data types
  - You can derive new data types based on other data types (by extension or restriction)
- XML schemas have more expressive power: e.g.
  - the value of any `<state>` attribute can't be longer than 2 characters
  - the value of any `<postal-code>` element must match the regular expression `[0-9]{5}(-[0-9]{4})?`.

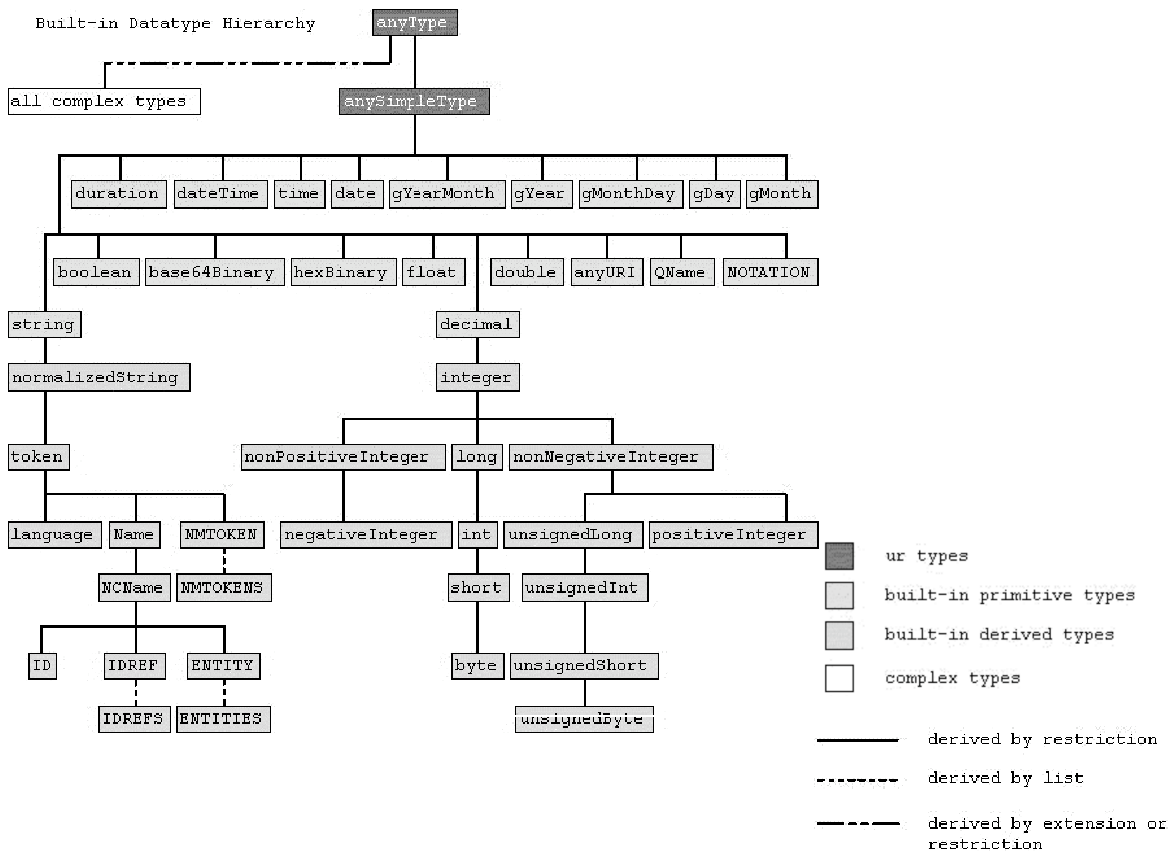
11-07

22

```

<?xml version="1.0" encoding="UTF-8"?>
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema">
  <xsd:element name="address"> <xsd:complexType> <xsd:sequence>
    <xsd:element ref="name"/>
    <xsd:element ref="street"/>
    <xsd:element ref="city"/>
    <xsd:element ref="state"/>
    <xsd:element ref="postal-code"/>
  </xsd:sequence> </xsd:complexType> </xsd:element>
  <xsd:element name="name"> <xsd:complexType> <xsd:sequence>
    <xsd:element ref="title" minOccurs="0"/>
    <xsd:element ref="first-Name"/>
    <xsd:element ref="last-Name"/>
  </xsd:sequence> </xsd:complexType> </xsd:element>
  <xsd:element name="title" type="xsd:string"/>
  <xsd:element name="first-Name" type="xsd:string"/>
  <xsd:element name="last-Name" type="xsd:string"/>
  <xsd:element name="street" type="xsd:string"/>
  <xsd:element name="city" type="xsd:string"/>
  <xsd:element name="state"> <xsd:simpleType>
    <xsd:restriction base="xsd:string"> <xsd:length value="2"/>
  </xsd:restriction></xsd:simpleType> </xsd:element>
  <xsd:element name="postal-code"> <xsd:simpleType>
    <xsd:restriction base="xsd:string"> <xsd:pattern value="[0-9]{5}(-[0-9]{4})"/>
  </xsd:restriction> </xsd:simpleType> </xsd:element>
</xsd:schema>

```



# Reusability with Schemas

- Element references
  - E.g. the “name” element in the schema before
- Groups (example in <http://www.w3.org/TR/xmlschema-0/>)
  - Substitutable element groups (substitutionGroup ~superclasses)
  - Attribute groups (attributeGroup ~“attribute interfaces”)
- Schema includes and imports
  - Documents must reference the namespaces of included schemas (~package includes)
  - When a schema is imported in another schema, its types become part of the importing namespace (~cut and paste in new package)
- Schema extensions and restrictions

11-07

25

## Validity and well-formedness

- Invalid documents don't follow the syntax rules defined by the XML specification (or the DTD they refer to)
- Well-formed documents follow the XML syntax rules but don't have a DTD or schema.
- Valid documents follow both the XML syntax rules and the rules defined in their DTD or schema.

11-07

26

# **XML programming interfaces**

11-07

27

## **Programming standards**

- “Platform- and language-neutral interfaces that allow programs and scripts to dynamically access and update the content, structure, and style of XML documents.”
- Document Object Model (DOM) - one-step parsing
  - Object-based
  - Better for complex documents
  - High memory usage, slower
  - Documents can be updated
- Simple API for XML (SAX) - push parsing
  - Event-based
  - Better for simple documents
  - Low memory usage, faster
  - Documents cannot be updated
- Pull parsing

11-07

28

# The Document Object Model

- It defines a set of interfaces to the parsed version of an XML document.
- You can
  - move through the tree to see what the original document contained,
  - delete sections of the tree,
  - rearrange the tree,
  - add new branches...
- The parser reads in the entire document and builds an in-memory tree.
  - If the document is very large, this requires a significant amount of memory.
  - The DOM creates objects that represent everything in the original document; if you only care about a small portion of the original document, it's extremely wasteful.
  - A DOM parser has to read the entire document before your code gets control. For very large documents, this could cause a significant delay.

11-07

29

## The Simple API for XML

- A SAX parser sends events to your code when it finds
  - the start of an element, the end of an element, text, the start or end of the document, ...
- A SAX parser doesn't create any objects at all, it simply delivers events to your application.
- A SAX parser starts delivering events to you as soon as the parsing begins.
  - Your application starts generating results right away; if you're only looking for certain things, your code can throw an exception once it has found them to stop the SAX parser
- But
  - SAX events are stateless; they simply give you the text that was found, not the element that contains the text.
  - SAX events are not permanent; if the application needs a data structure that models the XML document, the code has to be written.
  - SAX is not controlled by a centrally managed organization.

11-07

30

# Java packages

- JDOM
  - JDOM works with SAX and DOM parsers, so it's implemented as a relatively small set of Java classes.
  - It greatly reduces the amount of code you have to write.
    - JDOM applications are typically one-third as long as DOM applications, and about half as long as SAX applications.
- JAXP, the Java API for XML Parsing
  - It provides common interfaces for processing XML documents using DOM, SAX, and XSLT.

11-07

31

# Perl packages

- XML::Parser is the ultimate ancestor of XML processing in Perl.
- SAX defines an event-oriented interface that allows various XML processors to communicate. XML::DOM, XML::Grove, XML::Path and XML::XQL, amongst others, offer a SAX interface.
- XML::Parser::PerlSAX is a Perl SAX parser.
- XML::DOM is a Perl implementation of the W3C's DOM Level 1, plus extensions.
- XML::Simple was first written to allow easy loading and updating of configuration files written in XML. It can be used to process other kinds of simple XML documents. It does not grok mixed content (<p>this is <b>mixed</b>content</p>).
- XML::Twig offers another tree-oriented interface to XML documents. It allows loading of only parts of the document in order to keep memory requirements to a minimum.
- XML::TokeParser offers a simple pull-based API (not the XMLPULL API) on top of XML::Parser.
- XML::Grove loads an XML document in memory and creates a tree of Perl objects that can be accessed and manipulated.

11-07

32



# XML standards

- The XML specification
  - <http://w3.org/TR/REC-xml> <http://w3.org/TR/REC-xml-names/>
- XML Schema
  - <http://w3.org/TR/xmlschema-0> - <http://w3.org/TR/xmlschema-1> - <http://w3.org/TR/xmlschema-2>
- XSL, XSLT, and XPath
  - <http://w3.org/TR/xsl> - <http://w3.org/TR/xslt> - <http://w3.org/TR/xpath>
- DOM
  - <http://w3.org/TR/DOM-Level-2-Core>
  - <http://w3.org/TR/DOM-Level-2-Events>
  - <http://w3.org/TR/DOM-Level-2-Style>
  - <http://w3.org/TR/DOM-Level-2-Traversal-Range>
  - <http://w3.org/TR/DOM-Level-2-Views/>
- SAX, JDOM, and JAXP
  - <http://www.saxproject.org> - <http://jdom.org> - <http://java.sun.com/xml/jaxp>
- Linking and referencing
  - <http://w3.org/TR/xlink> - <http://www.w3.org/TR/xptr>

11-07

33

## Resources

- Building Web Services with Java: Making Sense of XML, SOAP, WSDL and UDDI by Graham et al. (chapter 2)
- A Technical Introduction to XML by Norman Walsh
  - <http://www.xml.com/lpt/a/98/10/guide0.html>
- Introduction to XML
  - <http://www-106.ibm.com/developerworks/edu/x-dw-xmlintro-i.html>
- W3schools tutorials
  - <http://www.w3schools.com/xml>
  - <http://www.w3schools.com/DTD>
  - <http://www.w3schools.com/schema/>
- Perl resources
  - <http://perl-xml.sourceforge.net/faq/>
  - <http://www.xml.com/lpt/a/2000/04/05/feature/index.html>

11-07

34