XML Technologies

XML: History, Basics, Syntax

Christian Grün
Database & Information Systems Group
Universität Konstanz

Preliminaries

Welcome...

...to this course on XML technologies!

This course will be about:

- languages: XPath, XQuery, XForms, XSLT
- information retrieval: full text, tokenization, stemming
- the art of compiler construction: how are languages evaluated?
- the architecture of databases: storage concepts, index structures
- building web applications with **X** technologies

Preliminaries

Organization

- please register in **StudIS** to take part in the exam
- please register in **LSF** to get news via mail
- if you have questions, don't be shy, don't feel dumb... please ask!

Schedule

- lecture: Tuesday, 13^{30} –15^{00}
  Christian Grün (christian.gruen@uni-konstanz.de)
- tutorial: Thursday, 15^{15} – 16^{45}
  Alexander Holupirek (alexander.holupirek@uni-konstanz.de)

Introduction: Markup Languages

- the term **markup** has been coined by the *typesetting community*, not by computer scientists
- with the advent of the *printing press*, writers and editors used (often marginal) notes to instruct printers to
  - select certain fonts
  - let passages of text stand out
  - indent a line of text, etc.
- proofreaders use a special set of symbols, a **markup language**, to identify typos, formatting glitches, and similar erroneous text fragments

☞ markup is designed to be easily recognizable in the text flow

Example: Typesetting Rules

11. **Überflüssige Buchstaben** oder **Wörter** werden durchgestrichen und auf ~~auf~~ dem Rand durch ℘ (für: deleatur, d. h. „es werde getilgt") angezeichnet.

12. **Fehlende** oder **überflüssige Satzzeichen** werden wie fehlende oder überflüssige Buchstaben angezeichnet.

13. **Verstellte Buchstaben** werden durchgestrichen und auf dem Rand in der richtigen Reihenfolge angegeben.
Verstellte Wörter durch werden das Umstellungszeichen gekennzeichnet.
Die Wörter werden bei größeren Umstellungen beziffert.
Verstellte Zahlen sind immer ganz durchzustreichen und in der richtigen Ziffernfolge auf den Rand zu schreiben, z. B. ~~1684.~~

14. **Für unleserliche** oder **zweifelhafte Manuskriptstellen,** die noch nicht blockiert sind, sowie für noch **zu ergänzenden Text** wird vom Korrektor eine Blockade verlangt, z. B.
~~Hyladen~~ sind Insekten mit unbeweglichem Prothorax (s. S. ~~...~~).

Source:
Duden, 1996,
21st Edition

- computer scientists adopted the markup idea – originally to annotate **source code**
- markup languages were designed such that their constructs were *easily recognizable* by a machine:
  - markup was written using a *special set* of characters, disjoint from the characters that form the tokens of the program
  - markup occurred in places in the source file where program code may not appear (*program layout*)
- example: syntax rules for Fortran 77 language
  - statements start in col 7 and do not exceed col 72
  - long statements are moved to next line; character in col 6 may not be `0`, `1`, or `_`
  - comment lines start with `C` or `*` in col 1
  - numeric labels (`DO`, `FORMAT` statements) have to be placed in col 1-5

Example: Fortran 77 Source Code

```
C This program calculates the area of a circle with radius R.
C
C     R: radius of circle
C     PI: value of PI = 3.14159
C     AREA: area of the circle = PI*R*R
*********************************************************************
      REAL R,AREA
C
C Set value of PI and R:
      PI = 3.14159
      R = 4.0
C
C Calculations:
      AREA = PI*R*R
C
C Write results:
      WRITE(6,*)  '  for a circle of radius', R,
     +            '  and the area is ', AREA
C
      END
```

- hm... does this look like the ultimate markup solution?

- increased computing power and more sophisticated parsing technology made fixed column positions obsolete
- low-level markup, however, is still being omnipresent in today's programming languages and systems:
  - ASCII defines a set of **non-printable** control characters (code range 0x00–0x1f):
    - `0x04`: EOT (end of transmission)
    - `0x09`: HT (horizontal tab)
    - `0x0A`: LF (line feed)
    - `0x0C`: FF (form feed)
    - `0x0D`: CR (carriage return)
  - blocks (containers) are defined using various form of matching **delimiters**:
    - `begin...end`, `\begin{foo}...\end{foo}`
    - `/* ... */`, `{ ... }`, `// ... LF`
    - `do ... done`, `if ... fi`, `case ... esac`, `$[...]`

Markup Languages: Categories

Presentational Markup

- (traditionally) binary encoding, not visible to end users
- examples: PDF, Word

Procedural Markup

- provides instructions on how to process the text
- examples: LaTeX, PostScript, troff

Descriptive Markup

- *semantic* markup: labels and structures text
- examples: SGML, XML

☞ every categorization is flawed…

✎ apart from XML, do you know other markups or text notations?

Other Text Notations

JSON: JavaScript Object Notation

Standard designed for human-readable data interchange

```
[
  "Address",
  {
    "Name"      : "John Hanson",
    "Address"   : "350 Fifth Avenue",
    "Town"      : "New York, NY 10118",
    "Phone"     : "+1-212-234-7263“
  }
]
```

Try it live:
`json.parser.online.fr`

Lightweight Markup Languages

WikiText, MarkDown: simple text format, later converted to HTML

```
Introduction
------------
Dear reader! This is an example for
a Markdown encoded text. Markdown...

* is easy to read and write.
* it contains a simple syntax for
  _formatting_ __text__ or
  [representing links](http://go.com)
```

Try it live:
`www.ctrlshift.net/project/markdowneditor`

XML

What is XML?

- TLA for eXtensible Markup Language
- defines a *set of rules* for encoding documents in a *human-readable* and *machine-readable* format
- goals: simplicity, generality and usability over the internet
- specified by the World Wide Web Consortium (W3C), which is also responsible for XPath, XQuery, XSLT, ...
- standard was finalized in 1998
- derived from **SGML**: **S**tandard **G**eneralized **M**arkup **L**anguage
- hundreds of XML dialects: RSS, KML, GraphML, ... ✎ what else?

Example: HyperText Markup Language

```
<html>
  <!-- Header -->
  <head id="0">
    <title>XML</title>
  </head>
  <!-- Body -->
  <body id="1" bgcolor="#FFFFFF" text="#000000">
    <h2>XML Technologies</h1>
    <div align="left">
      <b>Assignments</b>
      <ul>
        <li><a href="e1.html">Exercise 1</a></li>
        <li><a href="e2.html">Exercise 2</a></li>
      </ul>
    </div>
  </body>
  <?pi bogus?>
</html>
```

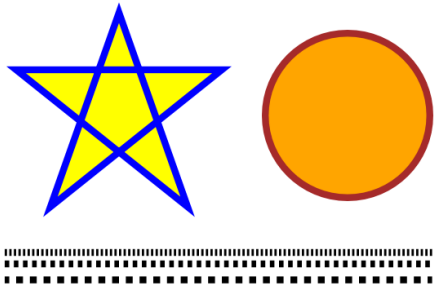
XML Technologies

Assignments

- Exercise 1
- Exercise 2

Example: Scalable Vector Graphics

```
<svg xmlns="http://www.w3.org/2000/svg"
     version="1.1">
  <polygon points="50,5 20,90 95,30 5,30 80,90"
    style="fill:yellow;stroke:blue;stroke-width:3"/>
  <circle cx="150" cy="50" r="36"
    stroke="brown" stroke-width="3" fill="orange"/>
  <g fill="none" stroke="black" stroke-width="3">
    <path stroke-dasharray="1" d="M 0 110 l188 0"/>
    <path stroke-dasharray="2" d="M 0 115 l188 0"/>
    <path stroke-dasharray="3" d="M-1 120 l188 0"/>
  </g>
</svg>
```



Try it live:
www.w3schools.com/svg/svg_examples.asp

Example: Mathematical Markup Language

```
<math xmlns="http://www.w3.org/1998/Math/MathML">
  <mfrac>
    <mrow>
      <mo>−</mo> <mi>b</mi> <mo>±</mo>
      <msqrt>
        <msup> <mi>b</mi> <mn>2</mn> </msup>
        <mo>−</mo>
        <mn>4</mn> <mo>&times;</mo> <mi>a</mi>
      </msqrt>
    </mrow>
    <mrow>
      <mn>2</mn> <mo>&times;</mo> <mi>a</mi>
    </mrow>
  </mfrac>
</math>
```

$$\frac{-b \pm \sqrt{b^2 - 4 \times a}}{2 \times a}$$

Currently supported by Firefox, Opera and Safari

Example: Microsoft Office XML

```
...
<a:p>
  <a:pPr lvl="1" eaLnBrk="1" hangingPunct="1">
    <a:buFont typeface="OfficinaSansITCStd Book"
              charset="0"/>
    <a:buChar char="•"/>
  </a:pPr>
  <a:r>
    <a:rPr lang="en-US">
      <a:sym typeface="OfficinaSansITCStd Book"
             pitchFamily="18" charset="2"/>
    </a:rPr>
    <a:t>goals: simplicity, generality
      and usability over the internet</a:t>
  </a:r>
</a:p>
...
    <a:t>specified by the World Wide Web
      Consortium (W3C),</a:t>
...
```

XML

What is XML?

- TLA for e**X**tensible **M**arkup **L**anguage
- defines a *set of rules* for encoding documents in a *human-readable* and *machine-readable* format
- goals: simplicity, generality and usability over the internet
- specified by the World Wide Web Consortium (W3C), which is also responsible for XPath, XQuery, XSLT, ...
- standard was finalized in 1998
- derived from **SGML**: **S**tandard **G**eneralized **M**arkup **L**anguage
- hundreds of XML dialects: RSS, KML, GraphML, ... ✎ what else?

How Large Can XML Documents Get?


```
<Entry id="100K_RAT" class="STANDARD" mtype="PRT" seqlen="889"> <AC>Q62671</AC> <Mod date="01-NOV-1997" Rel="35" type="Created"></Mod> <Mod date="01-NOV-1997"
Rel="35" type="Last sequence update"></Mod> <Mod date="15-JUL-1999" Rel="38" type="Last annotation update"></Mod> <Descr>100 KDA PROTEIN (EC 6.3.2.-)</Descr>
<Species>Rattus norvegicus (Rat)</Species> <Org>Eukaryota</Org> <Org>Metazoa</Org> <Org>Chordata</Org> <Org>Craniata</Org> <Org>Vertebrata</Org>
<Org>Euteleostomi</Org> <Org>Mammalia</Org> <Org>Eutheria</Org> <Org>Rodentia</Org> <Org>Sciurognathi</Org> <Org>Muridae</Org> <Org>Murinae</Org>
<Org>Rattus</Org> <Ref num="1" pos="SEQUENCE FROM N.A">  <Comment>STRAIN=WISTAR</Comment>  <Comment>TISSUE=TESTIS</Comment>  <DB>MEDLINE</DB>
<MedlineID>92253337</MedlineID>  <Author>Mueller D</Author>  <Author>Rehbein M</Author>  <Author>Baumeister H</Author>  <Author>Richter D</Author>
<Cite>Nucleic Acids Res. 20:1471-1475(1992)</Cite> </Ref> <Ref num="2" pos="ERRATUM">  <Author>Mueller D</Author> <Author>Rehbein M</Author>  <Author>Baumeister
H</Author>  <Author>Richter D</Author>  <Cite>Nucleic Acids Res. 20:2624-2624(1992)</Cite> </Ref> <EMBL prim_id="X64411" sec_id="CAA45756"></EMBL> <INTERPRO
prim_id="IPR000569" sec_id="-"></INTERPRO> <INTERPRO prim_id="IPR002004" sec_id="-"></INTERPRO> <PFAM prim_id="PF00632" sec_id="HECT" status="1"></PFAM> <PFAM
prim_id="PF00658" sec_id="PABP" status="1"></PFAM> <Keyword>Ubiquitin conjugation</Keyword> <Keyword>Ligase</Keyword> <Features>  <DOMAIN from="77" to="88">
<Descr>ASP/GLU-RICH (ACIDIC)</Descr>  </DOMAIN>  <DOMAIN from="127" to="150">  <Descr>PRO-RICH</Descr>  </DOMAIN>  <DOMAIN from="420" to="439">
<Descr>ARG/GLU-RICH (MIXED CHARGE)</Descr>  </DOMAIN>  <DOMAIN from="448" to="457">  <Descr>ARG/ASP-RICH (MIXED CHARGE)</Descr>  </DOMAIN>  <DOMAIN from="485"
to="514">  <Descr>PABP-LIKE</Descr>  </DOMAIN>  <DOMAIN from="579" to="590">  <Descr>ASP/GLU-RICH (ACIDIC)</Descr>  </DOMAIN>  <DOMAIN from="786" to="889">
<Descr>HECT DOMAIN</Descr>  </DOMAIN>  <DOMAIN from="827" to="847">  <Descr>PRO-RICH</Descr>  </DOMAIN>  <BINDING from="858" to="858">  <Descr>UBIQUITIN (BY
SIMILARITY)</Descr>  </BINDING> </Features></Entry><Entry id="104K_THEPA" class="STANDARD" mtype="PRT" seqlen="924"> <AC>P15711</AC> <Mod date="01-APR-1990"
Rel="14" type="Created"></Mod> <Mod date="01-APR-1990" Rel="14" type="Last sequence update"></Mod> <Mod date="01-AUG-1992" Rel="23" type="Last annotation
update"></Mod> <Descr>104 KDA MICRONEME-RHOPTRY ANTIGEN</Descr> <Species>Theileria parva</Species> <Org>Eukaryota</Org> <Org>Alveolata</Org>
<Org>Apicomplexa</Org> <Org>Piroplasmida</Org> <Org>Theileriidae</Org> <Org>Theileria</Org> <Ref num="1" pos="SEQUENCE FROM N.A"><Comment>STRAIN=MUGUGA
</Comment> <DB>MEDLINE</DB>  <MedlineID>90158697</MedlineID>  <Author>Iams K.P</Author>  <Author>Young J.R</Author>  <Author>Nene V</Author>  <Author>Desai
J</Author> <Author>Webster P</Author>  <Author>Ole-Moiyoi O.K</Author>  <Author>Musoke A.J</Author>  <Cite>Mol. Biochem. Parasitol. 39:47-60(1990)</Cite> </Ref>
<EMBL prim_id="M29954" sec_id="AAA18217"></EMBL> <PIR prim_id="A44945" sec_id="A44945"></PIR> <Keyword>Antigen</Keyword> <Keyword>Sporozoite</Keyword>
<Keyword>Repeat</Keyword> <Features> <DOMAIN from="1" to="19">  <Descr>HYDROPHOBIC</Descr>  </DOMAIN>  <DOMAIN from="905" to="924">  <Descr>HYDROPHOBIC</Descr>
</DOMAIN> </Features></Entry><Entry id="108_LYCES" class="STANDARD" mtype="PRT" seqlen="102"> <AC>Q43495</AC> <Mod date="15-JUL-1999" Rel="38"
type="Created"></Mod> <Mod date="15-JUL-1999" Rel="38" type="Last sequence update"></Mod> <Mod date="15-JUL-1999" Rel="38" type="Last annotation update"></Mod>
<Descr>PROTEIN 108 PRECURSOR</Descr> <Species>Lycopersicon esculentum (Tomato)</Species> <Org>Eukaryota</Org> <Org>Viridiplantae</Org> <Org>Embryophyta</Org>
<Org>Tracheophyta</Org> <Org>Spermatophyta</Org> <Org>Magnoliophyta</Org> <Org>eudicotyledons</Org> <Org>Asteridae</Org> <Org>euasterids I</Org>
<Org>Solanales</Org> <Org>Solanaceae</Org> <Org>Solanum</Org> <Ref num="1" pos="SEQUENCE FROM N.A">  <Comment>STRAIN=CV. VF36</Comment>  <Comment>
TISSUE=ANTHER</Comment> <DB>MEDLINE</DB>  <MedlineID>94143497</MedlineID>  <Author>Chen R</Author>  <Author>Smith A.G</Author>  <Cite>Plant Physiol. 101:1413-
1413(1993)</Cite>
```


...this is a 4KB snippet from 4GB XML protein data

XML: Cool Features

Text Format

- text can be *read* by everyone
- simple text editors can be used to *edit* XML documents

Standardization

- in the past and today, proprietary formats represent a huge cost factor
- *XML parsers* are readily available

Separation of Concerns (SoC)

- *data* and *presentation* is separated
- one XML document may be presented as HTML, PDF or in any other format
- border case: *presentation markup*

Tree Representation

- data can be hierarchically organized
- not possible via tables or RDBMS (✎ well, is this true?)

Formalization of XML

- we will now approach XML in a more formal way
- the good thing: the nuts and bolts of XML are pleasingly easy to grasp!
- this discussion will be based on the technical specification of XML

Extensible Markup Language (XML) 1.0 (Fifth Edition)

www.w3.org/TR/REC-xml

Technical Specifications

...are *boring* to read, which is why we will focus on the basic principles

- however, they are required to avoid ambiguities in interfaces and implementations

XML: Terminology

Characters

An XML document is a *string of Unicode characters*.

Processor vs. Application

A processor (XML parser) *analyzes* markup and *passes* structured information to an application.

Markup vs. Content

Distinguished by the application of *syntactic rules*.

Well-Formedness

A document is well-formed if it *satisfies* all *syntax rules* and *constraints* of the specification. Examples:

- legal Unicode characters
- correctly nested **tags**
- no duplicate **attributes**
- single root **element**
- characters < and & only used as markup (not in contents)

XML: Elements

- six *node types* exist; the **element** is one of them
- marked up document regions (*element content*) are enclosed in matching *start* and *closing (end) tags*:
  `<hello>World!</hello>`, `<こんにちは>世界</こんにちは>`
- element contents may contain characters and sub elements:
  `<TEXT>Hello <b>W</b>orld!</TEXT>`
- elements may also be empty: `<no-contents/>`
- tags must be properly nested. The following example is not *well-formed*:
  `<root><hello>World!</oops>`
- which data structure is useful for checking a proper nesting of elements?
- document can only have one *root element* (wrong: `<A/><B/>`)

XML: Attributes

- elements may be enriched with attributes:
  `<item currency="$">23.45</price>`
- attributes consist of *names* and *values*
- their values are confined to *characters* (i.e., they cannot be nested)
- attributes are not considered as *children* of the containing element (instead, they are *owned* by the element)
- all attribute names must be *unique*. this example is not well-formed:
  `<element id="item0" id="item1"/>`
- in contrast to elements, the *order* of attributes need not be preserved by an XML parser or serializer (✎ what is a serializer?)

XML: Remaining Node Types

Texts

- children of elements, pure character data:
  `<element>TEXT</element>`
- cannot have nodes as children

Comments

- used for documenting XML
  `<!-- comment -->`
- no nesting allowed

Processing Instructions

- carry "application instructions", embracing a *target* and *content*
- used to specify stylesheets or embed other languages; e.g.:
  `<?php echo "hehe" ?>`

Document Nodes

- every document has an invisible document node, which is the parent of the **root element**

XML: Entites

- five characters introduce XML markup syntax:
  `< > & " '`
- in order to include these chars as content, they need to be encoded as entities:

```
<    &lt;
>    &gt;
&    &amp;
"    &quot;
'    &apos;
```

- additional entities may be defined in a DTD (see later) or by an XML dialect
- all Unicode characters can also be specified as numeric entities (decimal or hexidecimal)
- HTML examples:

```
ä   &auml;   &#xe4;   &#228;
×   &times;  &#xd7;   &#215;
日  &#26085;  &#x65E5;
```

- entities may also be recursively defined (we'll skip the details)

XML: Context-free properties

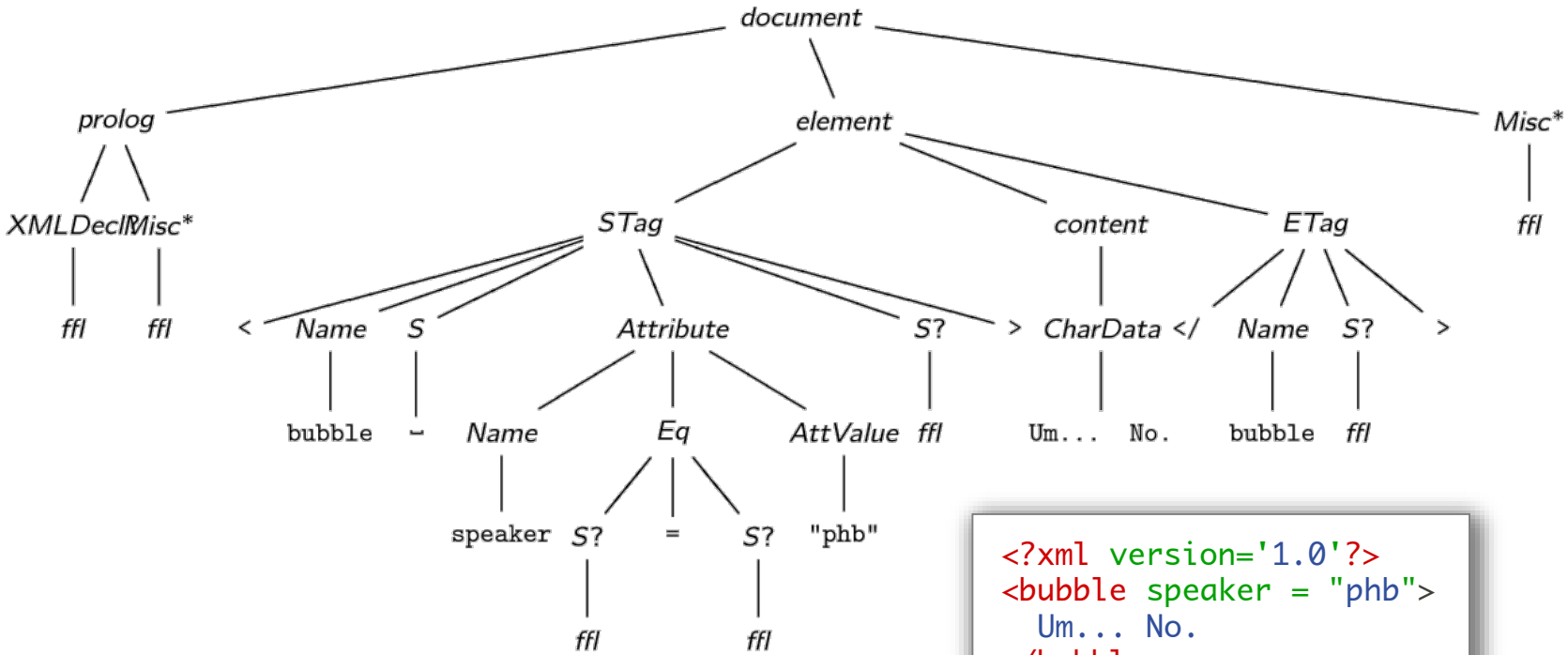
- all *context-free* properties of XML documents are concisely captured by a *grammar*
- EBNF grammars contain production rules of the form `lhs ::= rhs`
- Syntax (incomplete):

`r*`	zero or more *r*
`r+`	one or more *r*
`a|b`	alternations
`[abc]`	character class
`[^abc]`	inverted class

[1]	*document*	::=	*prolog element Misc**
[2]	*Char*	::=	⟨a Unicode character⟩
[3]	*S*	::=	('␣' | '\t' | '\n' | '\r')$^+$
[4]	*NameChar*	::=	*Letter* | *Digit* | '.' | '-' | '_' | ':'
[5]	*Name*	::=	(*Letter* | '_' | ':') (*NameChar*)*
[10]	*AttValue*	::=	'"' ([^<&"] | *Reference*)* '"'
		|	'’' ([^<&’] | *Reference*)* '’'
[14]	*CharData*	::=	[^<&]*
[22]	*prolog*	::=	*XMLDecl*? *Misc**
[23]	*XMLDecl*	::=	'<?xml' *VersionInfo EncodingDecl*? *S*? '?>'
[24]	*VersionInfo*	::=	*S* 'version' *Eq* ('’' *VersionNum* '’' | '"' *VersionNum* '"')
[25]	*Eq*	::=	*S*? '=' *S*?
[26]	*VersionNum*	::=	([a-zA-Z0-9_.:] | '-')$^+$
[27]	*Misc*	::=	*S*
[39]	*element*	::=	*EmptyElemTag*
		|	*STag content ETag*
[40]	*STag*	::=	'<' *Name* (*S Attribute*)* *S*? '>'
[41]	*Attribute*	::=	*Name Eq AttValue*
[42]	*ETag*	::=	'</' *Name S*? '>'
[43]	*content*	::=	(*element* | *CharData* | *Reference*)*
[44]	*EmptyElemTag*	::=	'<' *Name* (*S Attribute*)* *S*? '/>'
[67]	*Reference*	::=	*EntityRef*
[68]	*EntityRef*	::=	'&' *Name* ';'

- many syntax rules are implicitly provided by the grammar; e.g.:
  [1] an XML documents contains exactly one root element
  [10] attribute values are enclosed in " or '
  [22] XML documents may include an optional declaration prolog
  [14] characters < and & may not appear literally in element content
  [43] element content may contain character data and entity references as well as nested elements
  [68] entity references may contain arbitrary entity names (other than lt, amp, ...)
- the XML grammar is usually implemented as an *XML parser*, which reads the input as character-wise, creates *tokens*, matches these against the grammar and creates a **parse tree**
- a parse tree is a *hierarchical main-memory* representation of the input, which can be serialized back to its textual form
- a command-line tool for parsing XML documents (incl. schemas) is `xmllint` (comes e.g. with `libxml`)

XML: Parse Tree




```
<?xml version='1.0'?>
<bubble speaker = "phb">
  Um... No.
</bubble>
```

XML: Well-Formedness Constraints

- an XML grammar cannot express *Well-Formedness Constraints* (WFCs)
- WFCs are *context-dependent properties* and must be additionally checked
- such constraints...
  - depend on what the XML parser has *seen before* in its input,
  - access a *global state* (e.g. the definitions of user-declared entities), or
  - would require *very complex* or verbose grammar rules
- examples:
  - uniqueness of attributes → *no duplicate attribute names*
  - matching element names of opening and closing tags → *proper nesting*
  - `'&#x' [0-9a-fA-F]+ ';'` → restriction to *legal characters*

✎ how can these WFCs be algorithmically checked?

Conclusion

Potential drawbacks of XML

- takes *more space* than plain text (`"Hi"` vs. `<text>Hi</text>`)
- *slower* processing, compared to text or binary data
- incompatible with *legacy conventions* (e.g. backslash escaping)
- parts of the specification seem overly *complex* (e.g. DTD processing)

No drawbacks in practice

☞ XML can be *compressed* very well

☞ parsing overhead is very *low* on today's hardware

☞ legacy conventions are not *relevant* in many areas (such as publishing)

☞ specification details are *irrelevant* for most users and developers