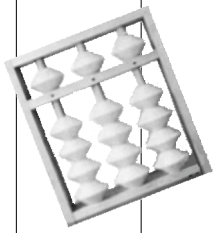
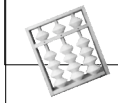


CECAM 2003

Introduction to XML

X. Gonze



Why is XML important ?

(Douglas Lovell, Advanced Internet Publishing
IBM T.J. Watson Research Center)

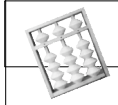
- **"For the first time in the history of computing, we have a universally acceptable syntax rich enough to handle all kinds of structured information"**
- **"XML represents a fundamental change in computing ... away from proprietary file and data formats to a world of open interchange"**
- **"The driver for this change is the desire by companies and individuals to access and exploit the mass of information made available via the internet"**

Goals of this tutorial

- To answer the following questions :
  - What is the XML syntax ?
  - What are its advantages over other data representations ?
  - What is a well-formed XML document ?
  - What is a valid XML document ?
  - What are the existing tools and standards to manipulate XML files, especially in view of interchange of data, over the Web ?
- To make you able to :
  - write a XML file, and validate it
- To familiarize you with the recent developments triggered by the “World Wide Web Consortium” W3C

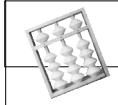
Overview

- HTML : the Web markup language
- What is XML ? Rules for a well-formed document
- Defining a markup language :
  - DTDs (Document Type Declarations)
  - XML Schemas
- Climbing the tree structure of XML : XPath
- Programming interfaces : DOM and SAX
- Transformation of a XML document : XSLT
- XLink, XQuery, RDF, SOAP
- More on the definition and validation of a XML document using a DTD ...



HTML

- **Hyper Text Markup Language**
- **Pros**
  - **Easy to use (proliferation of web pages)**
  - **Hyperlink support, multimedia support**
  - **Very good industry support for the user**
  - **Authors write pages displaying information**
  - **Portability and easy delivery over the network**
- **Cons**
  - A **fixed set of tags**
  - **Content and presentation mixed together**



HTML : an example

```
<HTML>   <HEAD>
           <TITLE>Welcome-Readme</TITLE>
         </HEAD>
<BODY>
  <H1>
    <CENTER> <IMG SRC="Images/pcpm.gif" ALIGN=bottom> </CENTER>
  </H1>
<P> <HR>
<p></P> Dear user of ABINIT (in short : ABINITioner),
<p> If this is the first time that you have access to ABINIT,
 or that you receive an ABINIT announcement, welcome !
<p> On the Web site, you will find a lot of things, including installation notes for
 different <a href="http://www.abinit.org/index.html#availables"> versions</a>
 of ABINIT,<a href="http://www.abinit.org/index.html#PSP">pseudopotentials</a>,
 some <a href="http://www.abinit.org/index.html#utile">utilities</a>,
  ....
```

What is XML ?

- XML stands for E**X**tensible **M**arkup **L**anguage
- XML is a “meta-language” to devise markup languages
- XML tags are not predefined in XML. You must define your own tags
- XML syntax is strict
- XML uses a Document Type Definition (DTD) or an XML Schema to formulate a language
- XML with a DTD or XML Schema is designed to be self-descriptive
- Proposed by the W3C (World Wide Web consortium) in 1999
- Ancestor : SGML (1980, already DTDs, but was too complex)

W3C WORLD WIDE WEB consortium

Leading the Web to its Full Potential...

XML Languages

- **XML = Meta-language used to define languages**
- **Examples of languages defined using XML:**

- **MathML - Math**ematical **M**arkup **L**anguage
- **XML Schema - Schema for XML documents**
- **SVG - S**calable **V**ector **G**raphics (a bit like postscript)
- **XSL -** e**X**tensible **S**tyle **L**anguage
- **XHTML - X H**yper **T**ext **M**arkup **L**anguage
- **CML - C**hemical **M**arkup **L**anguage
- (as of today, hundreds of DTDs available)

A first XML example

```
<?xml version="1.0"?>                  Header
 <List_of_participants>                Root element
  <Instructor id="id1">                Element with attribute (id)
   <FirstName>Konrad</FirstName>
   <LastName>Hinsen</LastName>         Simple elements
   <Language>French</Language>
   <Language>English</Language>        Second occurence of Language
   <Picture url="portrait.gif"/>       Empty element with attribute
                                        (link)
  </Instructor>
 </List_of_participants>
```


Well-formed documents

- **Each start-tag must have an associated end-tag**
- **Special markup for empty elements**

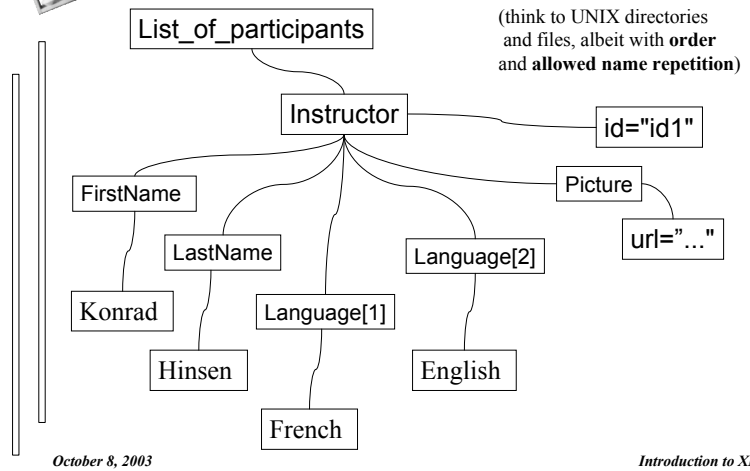
  `<IMG SRC="picture.gif"/>`

  `equivalent to : <IMG SRC="picture.gif"> </IMG>`
- **Elements must nest properly**

  HTML : Haha Hoho Hihi   Wrong in XML
- **Documents must have a single root element**
- **Upper/Lower case matters**
- **An element cannot have empty attributes**

`<DL COMPACT>`	`<DL COMPACT="">`
Wrong	**Right**

Tree structure of XML



Valid documents

- A **well-formed** document does not have any constraint about type of elements, attributes ..., and their content, but it fulfills the basic rules of XML
- A **valid** document must be a well-formed document **and** must comply with a grammar
  (allowed elements, attributes ...)
- One mechanism for specifying a grammar is called a **DTD**, another relies on a **XML Schema**

DTD

- Document Type Definition
- Set of syntactic rules for a type of document (Grammar definition language)
- A document can be validated against a DTD (xmllint is a simple validator on UNIX/Linux platforms)
- Grammar of a DTD file is NOT XML structured

- Definition of possible elements and their content
- Definition of possible attributes
- + ... (see later)

DTDs : definition of elements

- ***Syntax : <!ELEMENT name content>***

- ***Examples :***

```
Simple content
<!ELEMENT FirstName (#PCDATA) >
<!ELEMENT LastName (#PCDATA) >
<!ELEMENT Language (#PCDATA) >   PCDATA="parsed character data"
<!ELEMENT Picture EMPTY >
Complex content
<!ELEMENT Instructor
  (FirstName,LastName,(Language)*,Picture?) >
<!ELEMENT List_of_participants (Instructor|Student)* >

* = 0,1 or more , ? = 0 or 1 , + = 1 or more , | = "or" , ","="and"
```

DTDs : definition of attributes

- ***Syntax : <!ATTLIST element-name***
  (multiple) ***attribute-name type default>***

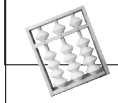
- ***Examples :***

```
<!ATTLIST Instructor
                id     ID  #implied >
                       ID="identifier"
<!ATTLIST Picture
                url    CDATA  #required >
                       CDATA="character data"
```


A full DTD

```
<!ELEMENT FirstName (#PCDATA) >
<!ELEMENT LastName (#PCDATA) >
<!ELEMENT Language (#PCDATA) >
<!ELEMENT Picture EMPTY >
<!ELEMENT Instructor
  (FirstName,LastName,(Language)*,Picture?) >
<!ELEMENT Student
  (FirstName,LastName,(Language)*,Picture?) >
<!ELEMENT List_of_participants (Instructor|Student)* >
<!ATTLIST Instructor
              id    ID  #implied >
<!ATTLIST Student
              id    ID  #implied >
<!ATTLIST Picture
              url   CDATA  #required >
```

Specifying a DTD in a XML file (I)

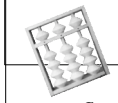


First possibility : no DTD !

```
<?xml version="1.0"?>
<List_of_participants>
     <Instructor id="id1">
     ....
     </Instructor>
</List_of_participants>
```

A XML parser will be able to check whether the document is well-formed, but it will not check whether it is valid

Specifying a DTD in a XML file (II)

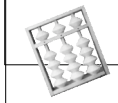


Second possibility : mention the DTD in the document !

```
<?xml version="1.0"?>
<!DOCTYPE List_of_participants [
  <!ELEMENT List_of_participant ... ! Here, one mentions
     ...                            ! the DTD
  <!ATTLIST ...                     !
 ]>
<List_of_participants>
     <Instructor id="id1">
     ....
```

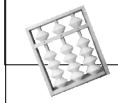
A XML parser will be able to check whether the document is well-formed and whether it is valid. But the DTD should better be independent of the document.

Specifying a DTD in a XML file (III)



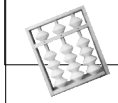
```
Third possibility : reference to the DTD file !

<?xml version="1.0"?>
<!DOCTYPE List_of_participants            !DTD reference
      SYSTEM "List_of_participants.dtd" > !
<List_of_participants>
     <Instructor id="id1">
     ....

The List_of_participants.dtd file contains :

<!ELEMENT List_of_participant ...
<!ATTLIST ...
```


Problems with the DTD mechanism



- **The syntax is specific to the DTD mechanism !**
  Not even an XML file ...
  It is contradictory to claim to have a universally acceptable syntax, and not use it to specify the XML languages !
- **The DTD typing possibilities are very weak :**
  Cannot define an integer, a float, a boolean variable, a date, a URL, while grammar rules might be made stronger by relying on such types.
- **So, development of new specifications :**
  - **XML Schema (W3C recommendation, May 2001) Next slides**
  - **RELAX NG (ISO/IEC technical recommendation)**
  - **Schematron**

A XML Schema is an XML file



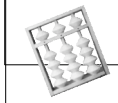
A XML file, with a particular grammar !

Also specified by a XML Schema ... of course.

Mechanism : the XML “name space”

```
<?xml version="1.0"?>                    ! The header
<xs:schema xmlns:xs=“http://www.w3.org/2001/XMLSchema”>
  (Here, one will use elements of the XML Schema
   language, all prefixed by  xs:...  )
</xs:schema>
```


XML Schemas : simple elements



- *Syntax of simple elements (do not have children, do not have attributes):*

<xs:element name=“element_name” type=“element_type”/>

(Note that this syntax is the one of an empty XML element)

- *Examples :*

```
<xs:element name=“FirstName” type=“xs:string” />
<xs:element name=“LastName” type=“xs:string” />
<xs:element name=“Language” type=“xs:string” />
```

Different simple types are possible :

xs:string, xs:ID, xs:anyURI, xs:float, xs:double, xs:integer, xs:boolean, xs:dateTime, ...

(more than 40 simple types)

XML Schemas : complex elements

- *Syntax (complex elements with children, but no attribute):*

```
<xs:element name="..." >
 <xs:complexType>
  <xs:sequence>
    Here, the list of permitted elements, referenced
  </xs:sequence>
 </xs:complexType>
</xs:element>
```

- *List of permitted elements : references, and occurence specification, example :*

```
<xs:element ref="Unique_mandatory_element" />
<xs:element ref="Repeated_element" maxOccurs="unbounded" />
<xs:element ref="Optional_element" minOccurs="1" />
```


XML Schemas : attributes

- *Syntax of attribute definitions (similar to syntax of element definitions):*

```
<xs:attribute name="..." type="..."/>
```

- *Mention an attribute to an element :*

```
<xs:element name="...">
 <xs:complexType>
  <xs:sequence>
   Here, the list of permitted elements, referenced
  </xs:sequence>
  <xs:attribute ref="name_of_attribute"/>   ! HERE
 </xs:complexType>
</xs:element>
```

XML Schemas : a full example (I)

- *The XML schema corresponding to the previous DTD*

```
<?xml version="1.0"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">
 <xs:element name="FirstName" type="xs:string" />
 <xs:element name="LastName" type="xs:string" />
 <xs:element name="Language" type="xs:string" />
 <xs:attribute name="id" type="xs:ID" />
 <xs:attribute name="url" type="xs:anyURI" />
 <xs:element name="Picture">
  <xs:complexType>
   <xs:attribute ref="url"/>
  </xs:complexType>
 </xs:element>
```

(continued ...)

XML Schemas : a full example (II)

```
<xs:element name="Instructor">
 <xs:complexType>
  <xs:sequence>
   <xs:element ref="FirstName"/>
   <xs:element ref="LastName"/>
   <xs:element ref="Language" min0ccurs="0"
                              max0ccurs="unbounded" >
   <xs:element ref="Picture" min0ccurs="0" >
  </xs:sequence>
  <xs:attribute ref="id"/>
 </xs:complexType>
</xs:element>
```

(continued ...)

XML Schemas : a full example (III)

```
<xs:element name="Student">
 <xs:complexType>
  <xs:sequence>
   <xs:element ref="FirstName"/>
   <xs:element ref="LastName"/>
   <xs:element ref="Language" min0ccurs="0"
                              max0ccurs="unbounded" >
   <xs:element ref="Picture" min0ccurs="0" >
  </xs:sequence>
  <xs:attribute ref="id"/>
 </xs:complexType>
</xs:element>
```

(continued ...)

XML Schemas : a full example (IV)

```
<xs:element name="List_of_participants">
 <xs:complexType>
  <xs:sequence>
   <xs:element ref="Instructor" min0ccurs="0"
                              max0ccurs="unbounded" >
   <xs:element ref="Student" min0ccurs="0"
                              max0ccurs="unbounded" >
  </xs:sequence>
 </xs:complexType>
</xs:element>
</xs:schema>                (the end)
```

XML Schema is much more verbose than the corresponding DTD !

Beyond the language definition

- **Suppose that we have a DTD or a XML Schema, and a set of XML documents that are well-formed and valid (can be validated by the DTD or XML schema)**
- **This rich and well-defined structure allows other layers of standards !**
- **XPath, API (DOM and SAX), XSLT, XLink, XQuery, RDF, SOAP ... (so many new acronyms)**

XPath/XPointer (I)

- **A standard to address any part or set of parts of an XML document**
- **Very similar to UNIX/Linux paths**
- **Examples of absolute paths :**

/ select the root node of the document

/List_of_participants/Instructor

select the “Instructor” node(s), children of the List_of_participants node

/List_of_participants/Instructor/Language[2]

select the second “Language” node, in the specified path

/List_of_participants/Instructor/Language[2]/text()

select the text in the second “Language” node, in the specified path

/List_of_participants/Instructor/@id

select the “id” attribute in the “Instructor” node(s), in the specified path

//Language select all the “Language” node(s), descendants of the root

XPath (II)

- **Examples of relative paths (need to know the "context node"):**

..	select the parent node
Instructor	select the "Instructor" child(ren), if any

- **Also :**
  - **wild cards ;**
  - **predicates ;**
  - **axis addressing (child,parent,self,attribute,ancestor,descendant, ...)**
  - **functions (count the number of nodes ...) ;**
  - **boolean logic**

API

- **Application Programming Interfaces**
- **Structure of XML document known**

⇩

possibility to define
standardized parsing methodologies

(please, do not reinvent the wheel)

Parsers written in : Python, Perl, C, C++, Java, F90 ...
Two standardized API methodologies :
SAX (Simple API for XML)
DOM (Document Object Model)

SAX

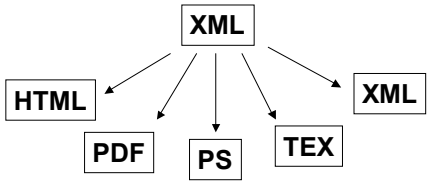
- Simple API for XML
- Idea 1 : Read the XML document sequentially
- Idea 2 : Consider each element, attribute, etc ... , as an "event", that will trigger an "action"
- Idea 3 : SAX routines to be integrated in a language-specific parser, that includes also routines defining the "action" triggered by each event type
- Advantage : the document need not be stored in memory
- However, the on-the-flight treatment of the events is not always easy to code !

DOM

- Document Object Model
- Idea : read the whole XML document, and represent it by a tree in main memory
- Need : the possibility to handle the tree data structure - allocation of pointers (F77 NO, F90 OK)
- The DOM specification is a recommandation of W3C
- Type of objects (all DOM applications use the same names !): Document, Element, Attr, Text ...
- Methods to act on the objects : set(), get() ...
- DOM usually based on SAX !

Transformation of XML documents

- Idea : one wants to automate (and standardize) the generation of .html, .pdf, .ps, .tex, ... or even other XML files from the XML documents

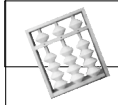


- Oldest technique : Cascading Style Sheet (formatting)
- New approach (XML based !) :

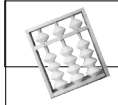
  EXtensible Stylesheet Language for Transformations

Other acronyms

- XLink : XML Linking Language
  - Allows to create (hyper)links between resources (XML documents)
  - Recommendation of the W3C (http://www.w3.org/TR/xlink)
- XQuery : XML Query Language
  - a query language for databases, based on XPath
  - Similarities with SQL
  - http://www.w3.org/XML/Query
- RDF : Resource Description Framework
  - a standard vocabulary to represent Metadata
  - goal : interoperability between applications that exchange informations to be treated automatically (Web oriented)
  - http://www.w3.org/TR/REC-rdf-syntax
- SOAP : Simple Object Access Protocol
  - protocol for exchaning information in a distributed environment
  - http://www.w3.org/TR/SOAP



More on XML and DTD



Back to the XML syntax : reminder

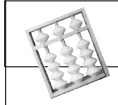
```
<?xml version="1.0"?>                Header
<!DOCTYPE List_of_participants
      SYSTEM "List_of_participants.dtd" >
<List_of_participants>               Root element
 <Instructor id="id1">               Element with attribute (id)
  <FirstName>Konrad</FirstName>
  <LastName>Hinsen</LastName>        Simple elements
  <Language>French</Language>
  <Language>English</Language>       Second occurence of Language
  <Picture url="portrait.gif"/>      Empty element with attribute
                                       (link)
 </Instructor>
</List_of_participants>
```

The header of a XML document (I)

```
<?xml version="1.0"?>
<!DOCTYPE List_of_participants
      SYSTEM "List_of_participants.dtd" >

Possible attributes of the xml declaration :
version (mandatory),
encoding, standalone (default is parser-dependent)

<?xml version='1.0' encoding="US-ASCII" standalone='yes'?>
<?xml version='1.0' encoding='iso-8859-1' standalone='no'?>

Note : can use single or double quote, but must be
   coherent.
```


The header of a XML document (II)

```
DOCTYPE syntax :
<!DOCTYPE root_element
        URI_of_the_dtd(=external subset)
 [     (internal subset)
 ]>

For the internal and external subset, see the DTD syntax.
Higher precedence of the internal subset : it can be used
   to change the definitions of the external subset.

<!DOCTYPE root_element SYSTEM "local.dtd">
<!DOCTYPE root_element SYSTEM "/usr/common.dtd">
<!DOCTYPE root_element SYSTEM "http://www.XX.org/YY.dtd">
```



Syntax of XML elements

Element syntax :

```
<name attribute1="val1" attribute2="val2">
   (optional content)
</name>
(or) <name attribute1="val1" attribute2="val2"/>
```

No space between "<" and "name"

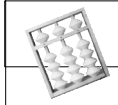
The number of attributes can be 0, 1 ...

The name of an element must begin with a letter or underscore (_), followed by an undefined number of letters, digits, underscore, dots, ... (can include accents, greek letters, hirigana, chinese ideograms ...)

However, the colon(:) is to be avoided, as it is booked for the namespace specification.

Upper/lower case is important !

<Bob>, <bob>, <chapter.title>, <_>, <THX-1328>



Syntax of XML attributes

Attribute syntax :

attribute="val" (or) attribute='val'

Each attribute of a single element must have a different name.

If needed, play with single or double quote :

<choice test='msg="hi"'>

or with so-called entities (see later)

' for single quote

" for double quote

If an attribute is of ID type, its value must be unique in the document

XML entities (I)

- Can be viewed as an abbreviation, or a container.
- Two main types : parameter entities and general entities. Parameter entities are used ONLY in DTDs, general entities can be used in XML documents, and defined in XML Schemas.
- DTD Definition of a general entity :

```
<!ENTITY name_of_entity “content”>
```

- XML call to a general entity :

```
&name_of_entity;
```


XML entities (II)

- Example :

```
<?xml version="1.0"?>
<!DOCTYPE Text [
  <!ENTITY PRL “Phys.Rev.Lett.”>
  <!ENTITY PRB “Phys.Rev.B”>
 ]>
<References>
 [1] X.Author &PRL;61,1(2001)
 [2] Y.Another &PRL;62,2(2002)
 [3] Z.Friend &PRB;73,33(2003)
</References>
```

Predefined XML entities

- Predefined entities exist (no need to define them in the DTD):

Unicode set of characters, numbered from 0 to 65535

Syntax : &#n;

where n is the index of the character in the Unicode set

Example : Use ç for ç

Also :

```
Use  &amp;  for  &
Use  &apos; for  ‘
Use  &gt;   for  >
Use  &lt;   for  <
Use  &quot; for  “
```


External XML entities

- External files can be imported in a XML document by defining them as an entity
- Example :

```
<?xml version="1.0"?>
<!DOCTYPE Longdoc [
  <!ENTITY part1 SYSTEM “p1.xml”>  !External entities
  <!ENTITY part2 SYSTEM “p2.xml”>  !defined in the
                                   !internal subset
 ]>
<Longdoc>
 &part1;
 &part2;
</Longdoc>
```

Comments and instructions

- **Comments :**

```
<!-- This is a comment -->
```

Warning : do not use -- in comments (other than in start-tag and end-tag)

- **Instructions :**

```
<?name_of_instruction data?>
```

Should be passed to the calling application (higher than the parser)

Entities in DTD : parameter entities

- DTD definition of a parameter entity :

```
<!ENTITY % name_of_entity "content" >
```

Note : content can be external, thus imported

Example :

```
<!ENTITY % list "(FirstName,(Language)*,Picture?)" >
<!ENTITY % part1 SYSTEM part1.dtd >
```

- DTD call to a parameter entity

  %name_of_entity;

Note that a parameter entity content is DTD-like, while a general entity content is XML-like.

Conditional sections in DTD

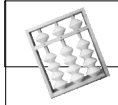
- Possibility to include/ignore part of DTD :

```
<![INCLUDE[ part of DTD to be included ]]>
<![IGNORE[ part of DTD to be ignored ]]>

This seems dummy, but its use is generally associated
with a parameter entity, allowing to maintain different versions
   of the DTD in one single document.
Example :
<!ENTITY % case_advanced "INCLUDE">
...
<![%case_advanced;[
  <!ENTITY Warning "This is an advanced version"> ]]>
<!ENTITY Warning "This is robust version">
In DTDs, the first occurence of an entity is used.
```


Links not yet mentioned

All at http://www.w3.org/TR/... (except SAX) :

- **XML : REC-XML**
- **Namespaces : REC-xml-names**
- **XML Schemas : xmlschema-0 (primer), xmlschema-1 (structures), xmlschema-2 (datatypes)**
- **XPath : xpath**
- **CSS : REC-CSS1 and REC-CSS2**
- **XSL(T) : xsl and xslt**
- **DOM : DOM-Level-2-Core, DOM-Level-2-Views, DOM-Level-2-Events, DOM-Level-2-Style, DOM-Level-2-Traversal-Range**
- **XHTML : xhtml1**
- **SOAP : SOAP**
- **SAX : http://www.megginson.com/SAX**



Books / Applications

- See the list at http://www.ibiblio.org/xml/books
- http://xml.oreilly.com (also http://www.xml.com)
  "Learning XML", "Learning XSLT", "Python & XML", "SAX2", "XML in a nutshell", "XML Schema", "XPath and XPointer", "XSLT" ...)
- The XML Bible
  (http://www.ibiblio.org/xml/books/bible)
- XML parsers : xmllint, Apache project (Xerces, Xalan), Doczilla, ... many commercial applications