XML

UPPSALA
UNIVERSITET
Marie Dubremetz
marie.dubremetz@lingfil.uu.se

Uppsala, April 2014

Presentation Plan

Table of Contents

Definition

A (document) markup language is a modern system for annotating a document in a way that is syntactically distinguishable from the text.
Examples ?

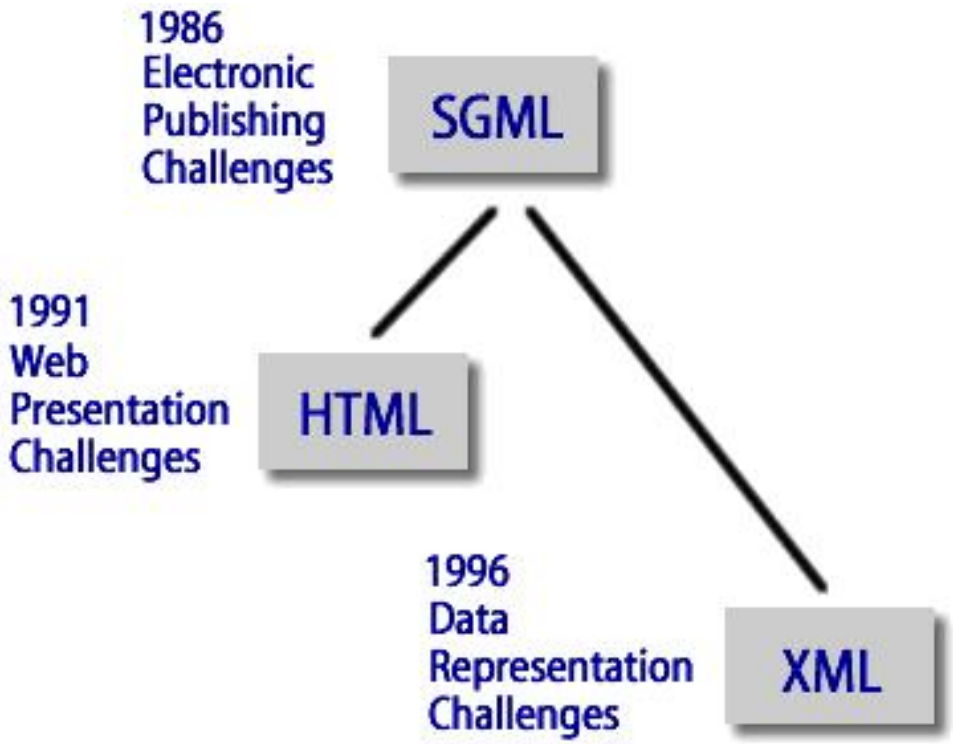
Definition

A (document) markup language is a modern system for annotating a document in a way that is syntactically distinguishable from the text.

Examples ?

- LaTeX \textbf{•} \section{•}
- HTML
- XML
- ...

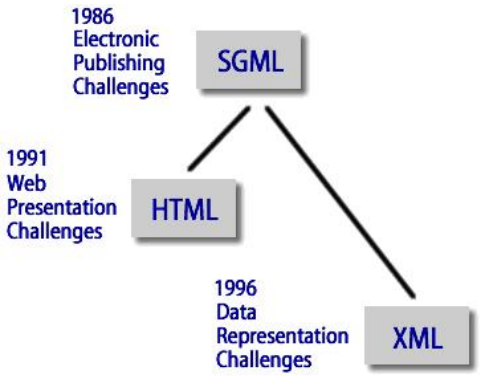
Why? Historical reasons



Why?

Internet is huge, diverse, heterogeneous, thus how can we perform:

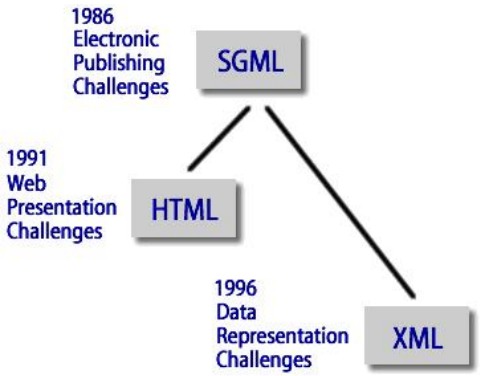
- data transmission ?
- standardization ?
- easy manipulation ?



Why?

Internet is huge, diverse, heterogeneous, thus how can we perform:

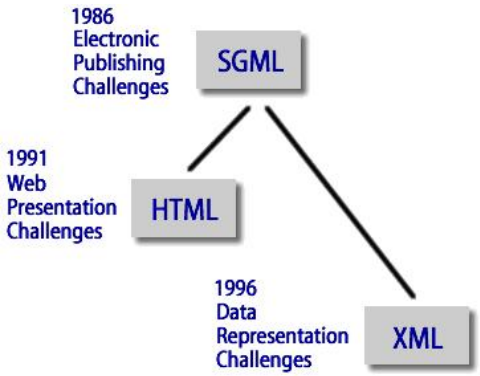
- data transmission ?
- standardization ?
- easy manipulation ?



Why?

Internet is huge, diverse, heterogeneous, thus how can we perform:

- data transmission ?
- standardization ?
- easy manipulation ?



Why?

Internet is huge, diverse, heterogeneous, thus how can we perform:

- data transmission ?
- standardization ?
- easy manipulation ?

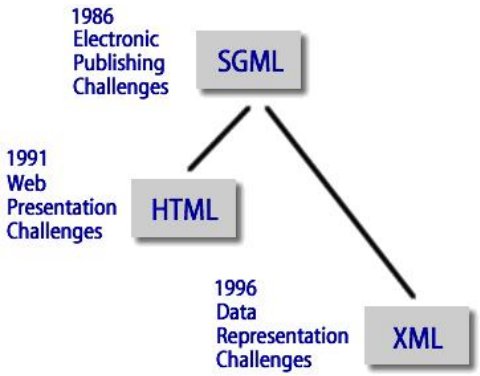


Table of Contents

XML Specificities

XML:

- XML is made for storing data
- Is not made for displaying information
- Lets you invent your own tags

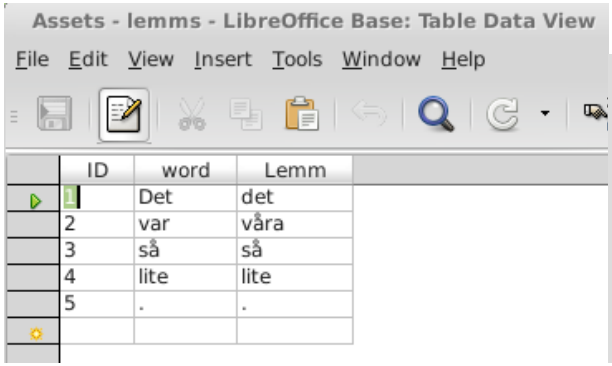
XML Specificities

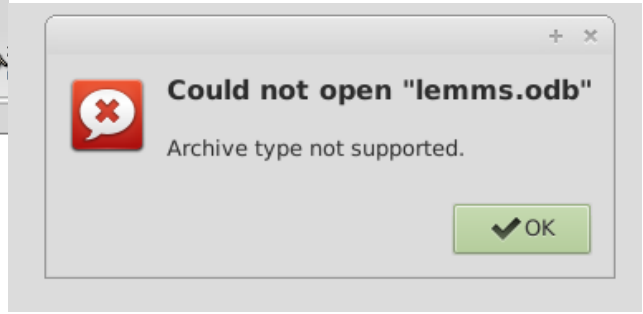
XML:

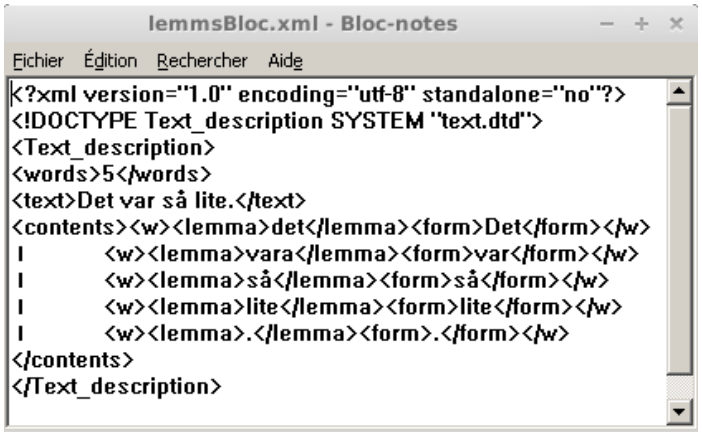
- XML is made for storing data
- Is not made for displaying information
- Lets you invent your own tags

```
<?xml version="1.0" encoding="UTF-8"?>
<俄语>данные</俄语>
```

Why is XML useful?



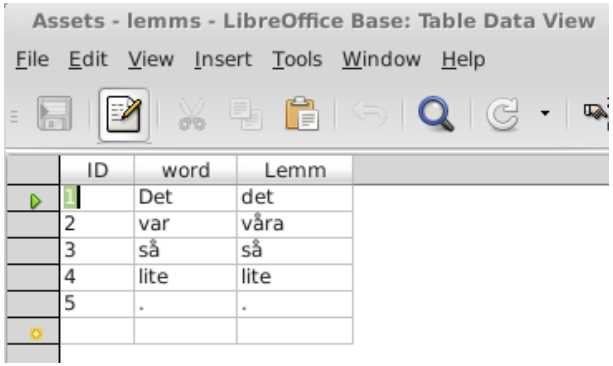


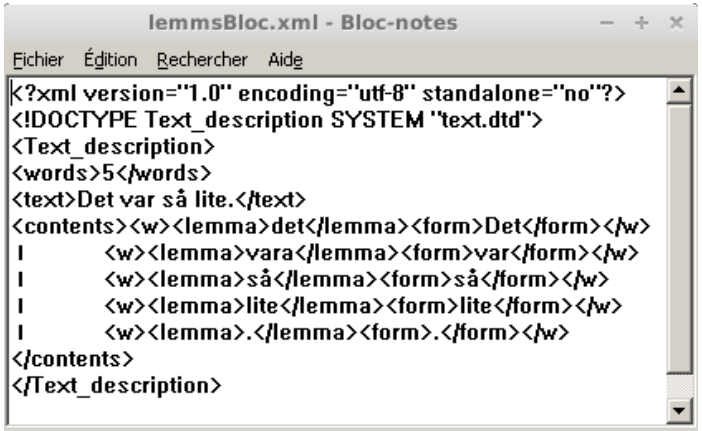


Why is XML useful?

XML is useful because:

- it allows you to share highly compatible data, between systems, over time...





- It is often used in NLP

Why is it useful?

Because XML is flexible, it allows to make some by-products.

XML by-products

Because XML is flexible it allows to make some by-products.

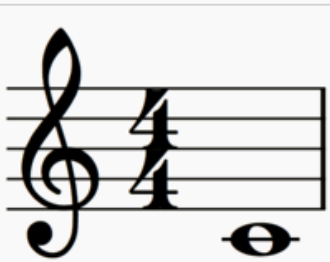
- OWL
- MusicXML

XML by-products

Because XML is flexible it allows to make some by-products.

- OWL
- MusicXML

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<!DOCTYPE score-partwise PUBLIC
    "-//Recordare//DTD MusicXML 3.0 Partwise//EN"
    "http://www.musicxml.org/dtds/partwise.dtd">
<score-partwise version="3.0">
  <part-list>
    <score-part id="P1">
      <part-name>Music</part-name>
    </score-part>
  </part-list>
  <part id="P1">
    <measure number="1">
      <attributes>
        <divisions>1</divisions>
        <key>
          <fifths>0</fifths>
        </key>
        <time>
          <beats>4</beats>
          <beat-type>4</beat-type>
        </time>
        <clef>
          <sign>G</sign>
          <line>2</line>
        </clef>
      </attributes>
      <note>
        <pitch>
          <step>C</step>
```



Representation of middle C on the treble clef created through MusicXML code.

XML by-products

Because XML is flexible it allows to make some by-products.

- OWL
- MusicXML
- RSS

Table of Contents

Flexible does not mean rule-free

XML document must respect some syntax rules:

- a good nested order
  <book><title></title><author></author></book>
  *<book><title><author></title></author></book>
  note: you can just as well create empty tags: <author/>
- XML is case sensitive
- root element is mandatory
- write comments like that: <! -- My comment -->
- attributes are between " ", <myTag myAttribute="0">

Vocabulary definition

When an XML document respects this syntax we say that it is "well formed"

Flexible does not mean rule-free

XML document must respect some syntax rules:

- a good nested order
- XML is case sensitive
  <Title></Title><author></author>
  *<Title></title><author></author>
- root element is mandatory
- write comments like that: <! -- My comment -->
- attributes are between " ", <myTag myAttribute="0">

Vocabulary definition

When an XML document respects this syntax we say that it is "well formed"

Flexible does not mean rule-free

XML document must respect some syntax rules:

- a good nested order
- XML is case sensitive
- root element is mandatory
- write comments like that: <! -- My comment -->
- attributes are between " ", <myTag myAttribute="0">

Vocabulary definition

When an XML document respects this syntax we say that it is "well formed"

Flexible does not mean rule-free

XML document must respect some syntax rules:

- a good nested order
- XML is case sensitive
- root element is mandatory
- write comments like that: <! -- My comment -->
- attributes are between " ", <myTag myAttribute="0">

Vocabulary definition

When an XML document respects this syntax we say that it is "well formed"

Flexible does not mean rule-free

XML document must respect some syntax rules:

- a good nested order
- XML is case sensitive
- root element is mandatory
- write comments like that: <! -- My comment -->
- attributes are between " ", <myTag myAttribute="0">

Vocabulary definition

When an XML document respects this syntax we say that it is "well formed"

QUIZ

```
<?xml version="1.0"?>
<note>
<to>Tove</to>
<from>Jani</from>
<heading>Reminder</heading>
<body>Don't forget me this weekend!</body>
</note>
```

Is this (above) a "well formed" XML document?

1. Yes
2. No

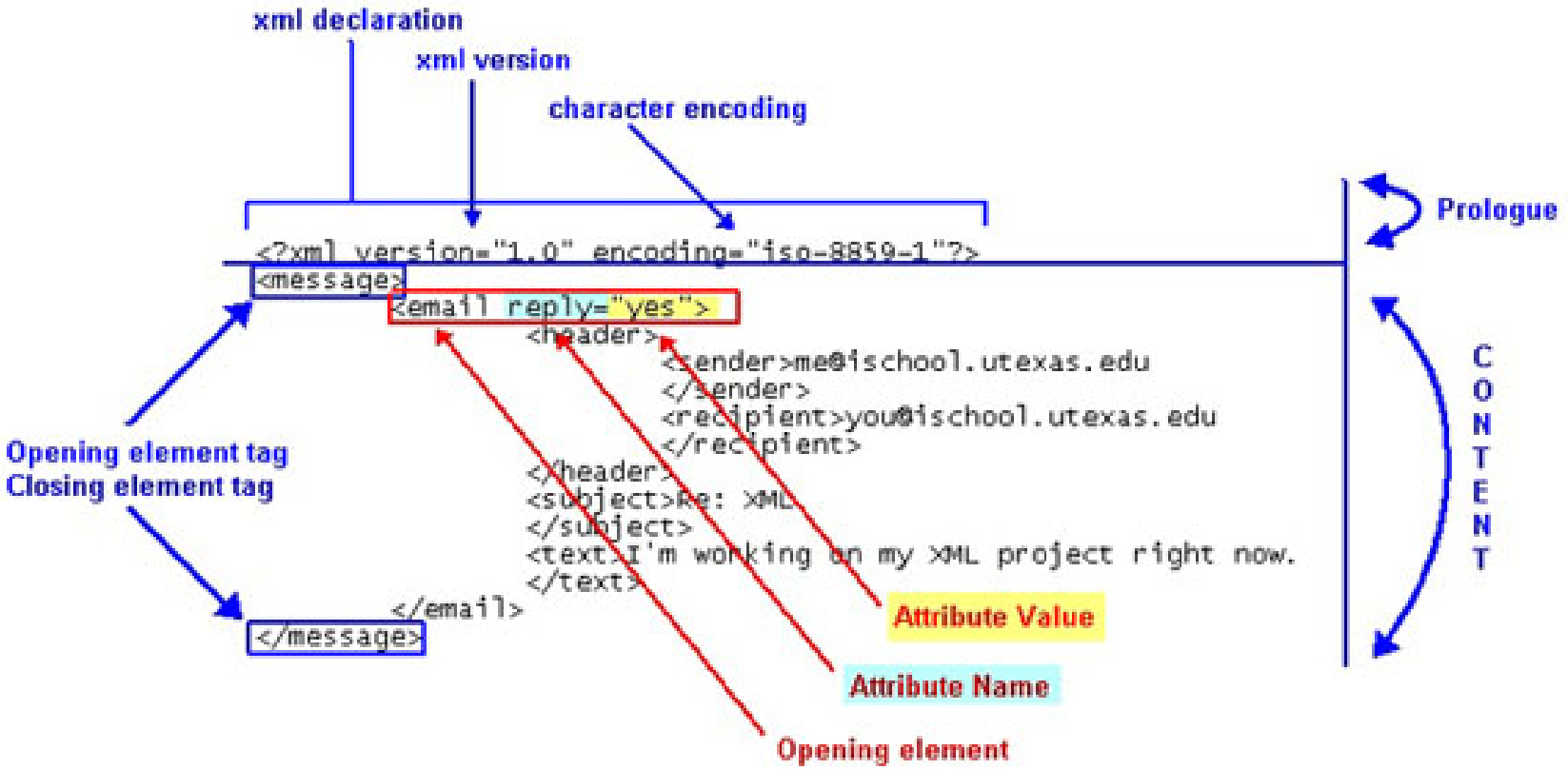
QUIZ

```
<?xml version="1.0"?>
<to>Tove</to>
<from>Jani</from>
<heading>Reminder</heading>
<body>Don't forget me this weekend!</body>
```

Is this a "well formed" XML document?

1. Yes
2. No

XML Structure



XML Structure

XML has what we call a tree structure.

XML Structure

XML has what we call a tree structure.

Example of tree structure

```
<dataroot>
    <Employees>
        <EmployeeID>1</EmployeeID>
        <LastName>Davolio</LastName>
        <FirstName>Nancy</FirstName>
        <Title>Sales Representative</Title>
    ...
```

This document fragment would be parsed into a tree structure as illustrated in Figure 1:

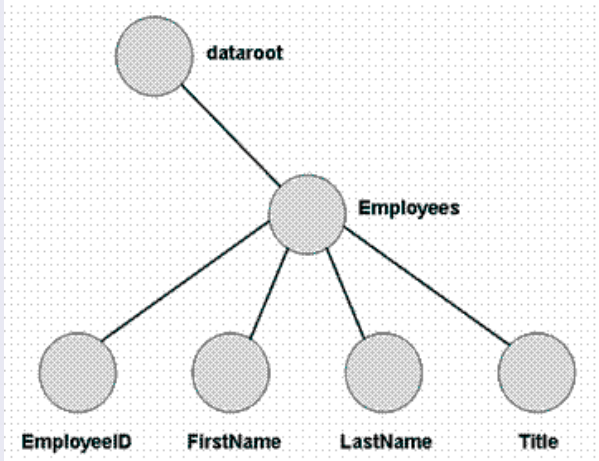


Figure 1. Example of Employees data as a tree structure

Let's practice

Are you able to give me the tree structure of this XML?

```
<Text_description>
<words>5</words>
<text>Det var så lite.</text>
<contents><w><lemma>det</lemma><form>Det</form></w>
        <w><lemma>vara</lemma><form>var</form></w>
        <w><lemma>så</lemma><form>så</form></w>
        <w><lemma>lite</lemma><form>lite</form></w>
        <w><lemma>.</lemma><form>.</form></w>
</contents>
</Text_description>
```

DTD

The structure definition of an XML document is described by a "Document Type Definition", or "dtd"

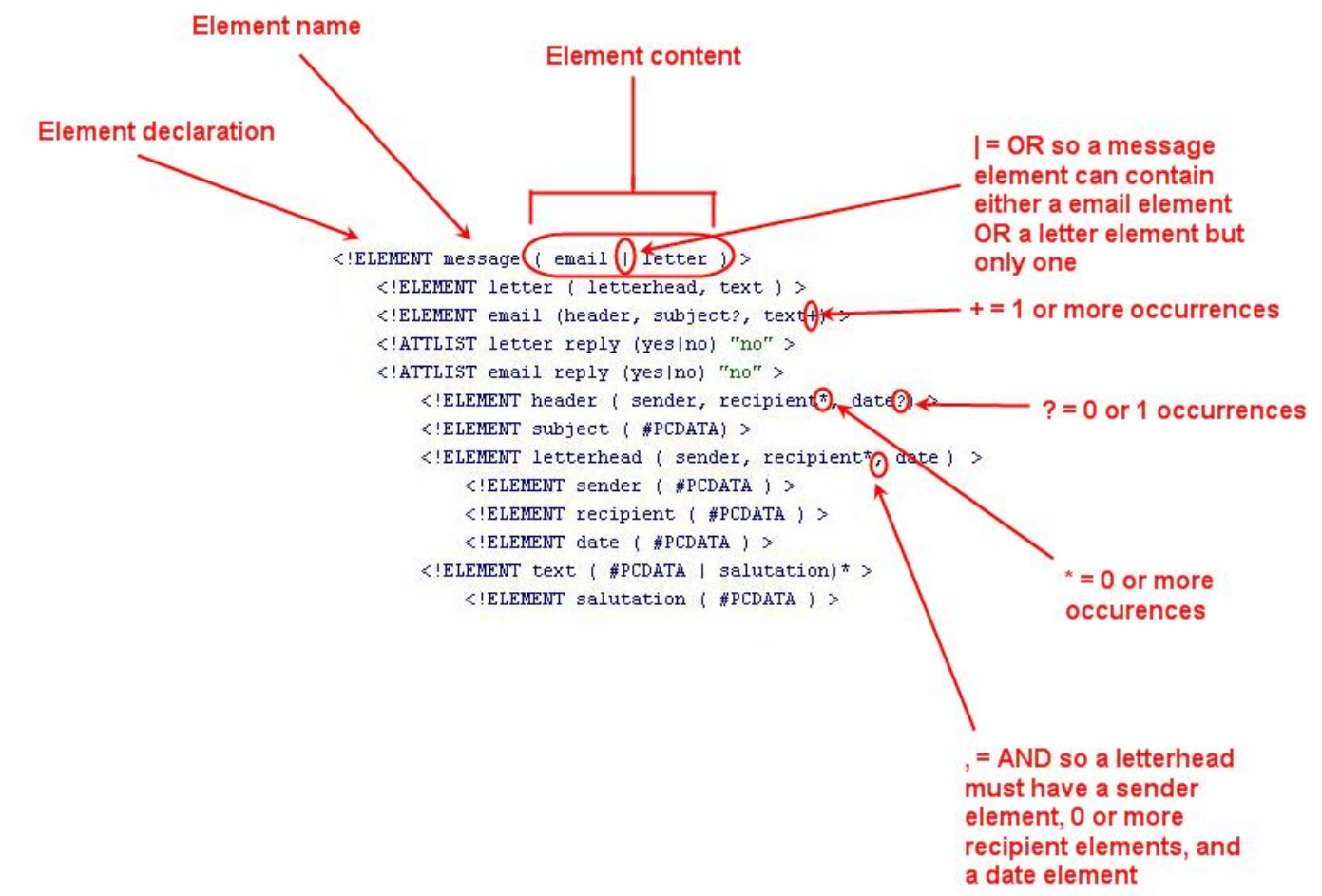
DTD

An XML is sometimes associated to a DTD thanks to an extra line added at the beginning of the XML.

```
<?xml version="1.0" encoding="utf-8" standalone="no"?>
<!DOCTYPE Text_description SYSTEM "text.dtd">
<Text_description>
<words>5</words>
<text>Det var så lite.</text>
```

(DTD declaration is on the second line)

Anatomy of a DTD



DTD

Example:

XML

```
<Text_description>
<words>5</words>
<text>Det var så lite.</text>
<contents><w><lemma>det</lemma><form>Det</form></w>
        <w><lemma>vara</lemma><form>var</form></w>
        <w><lemma>så</lemma><form>så</form></w>
        <w><lemma>lite</lemma><form>lite</form></w>
        <w><lemma>.</lemma><form>.</form></w>
</contents>
</Text_description>
```

DTD

```
<!ELEMENT Text_description (words,text,contents)>
<!ELEMENT words (#PCDATA)>
<!ELEMENT text (#PCDATA)>
<!ELEMENT contents (w+)>
<!ELEMENT w (lemma,form)>
<!ELEMENT form (#PCDATA)>
<!ELEMENT lemma (#PCDATA)>
```

DTD

Vocabulary

When an XML document respects the DTD we say that it is "valid"

Tools

Commands exist to check that your XML is well formed/valid

- Well formed:
  ```
  xmllint -noout document.xml
  ```
- Valid:
  ```
  xmllint -noout -valid document.xml
  ```

XPath

Definition

XPath, the XML Path Language, is a query language for selecting nodes from an XML document. XPath uses path expressions to select nodes or node-sets in an XML document. These path expressions look very much like the expressions you see when you work with a traditional computer file system.

XPath tool: xmlstarlet

Among other functions, you can select elements in your XML with xmlstarlet like this:

```
xmlstarlet sel -t -v "XpathCommand" document.xml
```

Example:

```
xmlstarlet sel -t -v "/Text_description/contents/w[2]/form" lemms.xml
```

XPath

```
<Text_description>
<words>5</words>
<text>Det var så lite.</text>
<contents><w><lemma>det</lemma><form>Det</form></w>
        <w><lemma>vara</lemma><form>var</form></w>
        <w><lemma>så</lemma><form>så</form></w>
        <w><lemma>lite</lemma><form>lite</form></w>
        <w><lemma>.</lemma><form>.</form></w>
</contents>
</Text_description>
```

XPath commands look like computer file system paths

Can you guess which output should give this XPath command?

`/Text_description/contents/w[2]/form`

1. vara
2. .
3. var
4. Det var så lite.
5. 5

What XPath is used for?

XPath is useful for:

- Navigating in the XML
- Creating XSLT

Definition

XSL stands for EXtensible Stylesheet Language, and is a style sheet language for XML documents. XSLT stands for XSL Transformations.

XSL

XSL is to XML what CSS is to HTML

XSL

cdcatalog.xml

```xml
<?xml version="1.0" encoding="UTF-8"?>
<?xml-stylesheet type="text/xsl" href="cdcatalog.xsl"?>
 <catalog>
   <cd>
     <title>Empire Burlesque</title>
     <artist>Bob Dylan</artist>
     <country>USA</country>
     <company>Columbia</company>
     <price>10.90</price>
     <year>1985</year>
   </cd>
   <cd>
     <title>Please Please Me</title>
     <artist>Beatles</artist>
     <country>UK</country>
     <company>Polydor</company>
     <price>15.5</price>
     <year>1963</year>
   </cd>

 </catalog>
```

cdcatalog.xsl

```xml
<?xml version="1.0" encoding="UTF-8"?>

 <xsl:stylesheet version="1.0"
 xmlns:xsl="http://www.w3.org/1999/XSL/Transform">

 <xsl:template match="/">
   <html>
   <body>
   <h2>My CD Collection</h2>
   <table border="1">
     <tr bgcolor="#9acd32">
       <th>Title</th>
       <th>Artist</th>
     </tr>
     <xsl:for-each select="catalog/cd">
     <tr>
       <td><xsl:value-of select="title"/></td>
       <td><xsl:value-of select="artist"/></td>
     </tr>
     </xsl:for-each>
   </table>
   </body>
   </html>
 </xsl:template>
```

XSL advantages and drawbacks

The XSL language manages:

- loops <xsl:for-each>
- conditions <xsl:if>
- sorting <xsl:sort>

- this language is very wordy
- no regex (must use XSL 2.0)

What should you do when you get an XML and an XSLT?

If you want to display it in a browser: add this red line to the XML and open the XML:

```
<?xml version="1.0" encoding="UTF-8"?>
<?xml-stylesheet type="text/xsl" href="cdcatalog.xsl"?>
<catalog>
  <cd>
    <title>Empire Burlesque</title>
    <artist>Bob Dylan</artist>
    <country>USA</country>
```

If you want to get the output directly, use a processor.

```
xsltproc document.xml document.xsl > whatever
```

Conclusion

What experience will tell you

Even if XML can (almost)always be handled by writing your own program and regex... Handling XML tools will save you time. Look in your favourite language documentation: Python, Java etc. there is always a library for it. As any language XML, XPath, XSLT are learned through practice!

Conclusion

What experience will tell you

Even if XML can (almost)always be handled by writing your own program and regex... Handling XML tools will save you time. Look in your favourite language documentation: Python, Java etc. there is always a library for it. As any language XML, XPath, XSLT are learned through practice!

Conclusion

What experience will tell you

Even if XML can (almost)always be handled by writing your own program and regex... Handling XML tools will save you time. Look in your favourite language documentation: Python, Java etc. there is always a library for it. As any language XML, XPath, XSLT are learned through practice!

QUIZ

When an XML document is conform to the dtd we say that it is:

1. "Well formed"
2. "Conformist"
3. "Valid"

QUIZ

The reference, short and clear, with quizzes and practical exercises:
http://www.w3schools.com/xml/default.asp
If you are interested in XML and web programming in general this is a good interactive course:
https://www.udacity.com/course/viewer#!/c-cs253/l-48296556/m-48369761