Goals of the Course

- Learn about central models and languages for
  - manipulating
  - representing
  - transforming and
  - querying
  structured documents (or XML)
- "Generic XML processing technology"

Methodological Goals

- Some central professional skills
  - consulting of technical specifications
  - experimenting with SW implementations
- Ability to think…?
  - to find out relationships
  - to apply knowledge in new situations
  - ("Pidgin English" for scientific communication)

XML?

- Extensible Markup Language is not a markup language!
  - does not fix a tag set nor its semantics
    (like markup languages like HTML do)
- XML is
  - A way to use markup to represent information
  - A metasync
    - supports definition of specific markup languages through XML
      DTDs or Schemas
    - e.g. XHTML a reformulation of HTML using XML

XML Encoding of Structure: Example

```
<S>
  <W> Hello </W> <E A='1'> world! </E>
</S>
```

Basics of XML DTDs

- A Document Type Declaration provides a grammar (document type definition, DTD) for a class of documents
- Syntax (in the prolog of a document instance):
  ```
  <!DOCTYPE rootElemType SYSTEM "ex.dtd"
  <!-- "external subset" in file ex.dtd -->
  [<!-- "internal subset" may come here -->
  ]>
  DTD is the union of the external and internal subset

How do Declarations Look Like?

```xml
<!ELEMENT invoice (client, item+)>  
<!ATTLIST invoice num NMTOKEN #REQUIRED>  
<!ELEMENT client (name, email?)>  
<!ATTLIST client num NMTOKEN #REQUIRED>  
<!ELEMENT name (#PCDATA)>  
<!ELEMENT email (#PCDATA)>  
<!ELEMENT item (#PCDATA)>  
<!ATTLIST item  
  price NMTOKEN #REQUIRED  
  unit (FIM | EUR) "EUR" >
```
XML Schema Definition Language

- XML syntax
  - schema documents easier to manipulate by programs (than the special DTD syntax)
- Compatibility with namespaces
  - can validate documents using declarations from multiple sources
- Content datatypes
  - 44 built-in datatypes (including primitive Java datatypes, datatypes of SQL, and XML attribute types)
  - mechanisms to derive user-defined datatypes

XML Namespaces

```xml
<xsl:stylesheet version="1.0"
  xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
  xmlns="http://www.w3.org/TR/xhtml1/strict">
  <!-- XHTML is the 'default namespace' -->
  <xsl:template match="doc/title">
    <h1>
      <xsl:apply-templates/>
    </h1>
  </xsl:template>
</xsl:stylesheet>
```

3. XML Processor APIs

- How can applications manipulate structured documents?
  - An overview of document parser interfaces

3.1 SAX: an event-based interface
3.2 DOM: an object-based interface
3.3 JAXP: Java API for XML Processing

DOM: What is it?

- An object-based, language-neutral API for XML and HTML documents
  - Allows programs and scripts to build, navigate, and modify documents
- In contrast to "Serial Access XML" could think as "Directly Obtainable in Memory"

Overview of XSLT Transformation

JAXP (Java API for XML Processing)

- An interface for "plugging-in" and using XML processors in Java applications
  - includes packages
    - `org.xml.sax`: SAX 2.0 Interface
    - `org.w3c.dom`: DOM Level 2 Interface
    - `javax.xml.parsers`: initialization and use of parsers
    - `javax.xml.transform`: initialization and use of transformers (XSLT processors)
- Included in JDK starting from vers. 1.4
JAXP: Using a SAX parser (1)

```java
f.xml
SAXParserFactory.newInstance().newSAXParser()
```

JAXP: Using a DOM parser (1)

```java
DocumentBuilderFactory.newInstance().newDocumentBuilder()
Document document = documentBuilder.parse("f.xml")
```

JAXP: Using Transformers (1)

```java
TransformerFactory.newInstance().newTransformer()
```

CSS - Cascading Style Sheets

- A stylesheet language
  - mainly to specify the representation of web pages by attaching style (fonts, colours, margins, ...) to HTML/XML documents

```
H1 {color: blue; font-weight: bold;}
```

CSS Processing Model (simplified)

0. Parse the document into a tree
1. Match style rules to elements of the tree
   - annotate each element with a value assigned for each relevant property
     » inheritance and, in case of competing rules, elaborate "cascade" rules applied to select which value is assigned
2. Generate a formatting structure of the annotated document tree
   - consists of nested rectangular boxes
3. Render the formatting structure
   - display, print, audio-synthesize, ...

XSL: Transformation & Formatting

```
XSLT script
XSL Transform
XSL Formatter
```

Page regions

- A simple page can contain 1-5 regions, specified by child elements of the `simple-page-master`

```
Region-start
Region-body
Region-end
```

Top-level formatting objects

```
fo:root
fo:layout-master-set
fo:page-sequence+

```

- Slightly simplified:
  - specify masters for page sequences, by referring to `simple-page-masters`
XQuery in a Nutshell

- Functional expression language
  - A query is a side-effect-free expression
- Strongly-typed: (XML Schema) types may be assigned to expressions statically, and results can be validated
- Extends XPath 2.0 (but not all axes required)
  - common for XQuery 1.0 and XPath 2.0:
    - Functions and Operators, W3C Cand. Rec. 11/2005
- Roughly: XQuery = XPath' + XSLT' + SQL'

FLWOR ("flower") Expressions

- Constructed from for, let, where, order by and return clauses (~SQL select-from-where)
- Form: (ForClause | LetClause)+
  - WhereClause?
  - OrderByClause?
- "return" Expr
- FLWOR binds variables to values, and uses these bindings to construct a result (an ordered sequence of nodes)

XQuery Example

```xml
for $pn in distinct-values(
  doc("sp.xml")//pno)
let $sp:=doc("sp.xml")//sp_tuple[pno=$pn]
where count($sp) >= 3
order by $pn
return
<well_supplied_item>
  <pno>{$pn}</pno>
</well_supplied_item>
<avgprice> {avg($sp/price)} </avgprice>
<well_supplied_item>
```

Course Main Message

- XML is a universal way to represent information as tree-like data structures
- There are specialized and powerful technologies for processing it
  - Worst hype has settled
  - Lots of R&D activities going on