Goals of the Course

- To familiarize with the most important models and languages for
  - manipulating
  - representing
  - transforming and
  - querying
  structured documents (or XML)
- Generic XML processing technology
  - very little about specific XML applications or commercial systems

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 metalanguage
    > 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?

```
<!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" >
```

Element type declarations

- The general form is
  ```
  <!ELEMENT elementType (E)>  
  where E is a content model
  ```
- regular expression of element names
- Content model operators:
  ```
  E | F : alternation
  E* : zero or more
  E+ : one or more
  (E) : grouping
  ```
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

A SAX-based application

```xml
<?xml version='1.0'?><A i="1">Hi!</A>
```

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
  - a foundation for developing querying, filtering, transformation, rendering etc. applications on top of DOM implementations
- In contrast to "Serial Access XML" could think as "Directly Obtainable in Memory"

Overview of XSLT Transformation

```
<invoice form="00"
type="estimated">
  <addressdata>
    <name>John Doe</name>
    <address>
      <streetaddress>Pyynpolku 1</streetaddress>
      <postoffice>70460 KUOPIO</postoffice>
    </address>
  </addressdata>
</invoice>
```

JAXP 1.1

- 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)

```
newSAXParser().
```

JAXP: Using a DOM parser (1)

```
documentBuilderFactory.
```

JAXP: Using Transformers (1)

```
TRANSFORMER
```

Transformation & Formatting

```
XSLT
```

Page regions

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

```
Region-before
Region-body
Region-end
Region-after
```

Top-level formatting objects

- Slightly simplified:
  ```
fos:root
```

XML-wrapping

- Need "XML-wrappers" (aka extractors)
  - interface/conversion program to produce an XML representation for source data

```
source1
source2
source3
```

XW-architecture (3)

```
application
```

```
<part-a>
<e1>x1</e1>
<e2>x2</e2>
</part-a>
```

```
<part-b>
<line-1>
<line-2>
<line-3>
</line-3>
"x1"="x2"
</part-b>
```

```
result document
```
```
Rearranging result structures

XQuery in a Nutshell

- **Functional expression language**
- **Strongly-typed**: (XML Schema) types may be assigned to expressions statically
- **Includes XPath 2.0** (says Draft, but not all XPath axes included!)
  - XQuery 1.0 and XPath 2.0 share extensive functionality:
    - XQuery 1.0 and XPath 2.0 Functions and Operators, WD 15/11/2002
- **Roughly**: XQuery ≈ XPath' + XSLT' + SQL'

Course Main Message

- XML is a universal way to represent info as tree-like data structures
- There are specialized and powerful technologies for processing it
- The development is going on