Chapter 2

XML - eXtensible Markup Language

XML Documents
XML DTD (Document Type Definition)
XML Namespaces
XML Schema
What is a Markup?

- **Markup = additional information in document text, e.g., format hints or links**
  - Each text processing system generates markup, e.g., \texttt{\textbackslash par} (in RTF) identifies paragraph begin or \texttt{\textbackslash sb240} specifies 12-pt space
  - Users usually do not enter a markup; opposite example is LaTeX

- **HTML = Hyper-Text Markup Language**
  - Language for information representation in the Web
  - Provides layout directives but does not contain generalized markup
    - \texttt{<h1>, <h2>, <h3>, <em>, <b>, <ul>, <ol> or <li>}

- **Generalized markup separates between logical and layout structure**
  - Allows structuring and view definitions
XML History

Taken from Neil Bradley: The XML companion

1960
Generalized Markup

1986
SGML

1992
HTML

1997
XML

WWW
Internet
What is XML?

- eXtensible Markup Language

- Specifies data format XML
  - Syntax
  - Elements, Attributes, Entities
  - Character Sets

- and logical and physical structure
  - Document Type Definition (DTD)

- but does not determine how to process XML

- Goal: Format for exchange, processing, and representation of information in the Web

- Meta language for defining markup languages
  - Part of SGML (Standard Generalized Markup Language)
  - Examples for Markup Languages: MathML, ChessML, XUL (User Interfaces), CellML, Gene Expression Markup Language, Chemical Markup Language, XML/EDI, UN/EDIFACT, ...
XML Document: Overview

**XML Document**
Comment, Character Set Reference

**Prologue** (Document Type Declaration)

**Document Type Definition (DTD)**
- Element,
- Attribute,
- Entity

**External DTD**
Conditional Fragments

**Document Element**

**Document**
- Start tag, end tag, empty element tag, PCDATA, CDATA, entity references
<?xml version="1.0" encoding="UTF-8"?>
<!– This XML document contains information about books -->
<!– A comment may contain arbitrary character sequences such as <tagname> or &entity; -->

<bookstore>
    <book genre="autobiography">
        <title>The Autobiography of Benjamin Franklin</title>
        <author>
            <first-name>Benjamin</first-name>
            <last-name>Franklin</last-name>
        </author>
        <price>8.99</price>
    </book>
    <book genre="novel">
        ...
    </book>
</bookstore>
Structure of XML: Elements and Attributes

- **XML defines tree structure!**
  - Root = document element
  - Leaves contain the values
  - Single data type: character sequence

- **Elements described by tags:**
  - Case-sensitive tag names
  - Each opened tag must be closed
  - Tags may not overlap, e.g., the following is not allowed: `<A> <B> </A> </B>`
  - Empty tags: `<A/>` (Shortcut for `<A> </A>`)  
  - Tags may contain attributes:
    `<A AttributeName="Value" ...> </A>`

```
<book genre="autobiography">
  <title>
    Autobiography of Benjamin Franklin
  </title>
  <price><CHF/> 8,99</price>
</book>
```
XML Element vs. XML Attribute

- Main difference: While XML elements may have subelements, XML attributes cannot be described any further.

- Example: Two different syntactical representations of the same content

  `<author>
   <id>170470</id>
   <first-name>Herman</first-name>
   <last-name>Melville</last-name>
   </author>`

  `<author id="170470">
   <first-name>Herman</first-name>
   <last-name>Melville</last-name>
   </author>`

- Distinction between XML element and XML attribute not uniquely possible at design time.
### XML Entities

- **Reuse XML document fragments**

<table>
<thead>
<tr>
<th>Internal Entity</th>
<th>External Entity</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>&lt;!ENTITY Entityname &quot;Replacementtext&quot;&gt;</code></td>
<td></td>
</tr>
<tr>
<td><code>&lt;!ENTITY Entityname SYSTEM &quot;URL&quot;&gt;</code></td>
<td></td>
</tr>
</tbody>
</table>

- **Entity reference:** & Entityname ;

- **Example:**

```xml
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE lecture [ 
  <!ENTITY semester "FS16"> 
]> 
<lecture>The lecture &quot;XML and Databases&quot; will be taught in &semester; by Dr. Can T&uuml;rker. For further lectures in &semester; see the university calendar. </lecture>
```

<table>
<thead>
<tr>
<th>Specific characters</th>
<th>Internal Entity</th>
<th>External Entity</th>
</tr>
</thead>
<tbody>
<tr>
<td>&lt;</td>
<td></td>
<td>&lt;</td>
</tr>
<tr>
<td>&gt;</td>
<td></td>
<td>&gt;</td>
</tr>
<tr>
<td>&amp;</td>
<td></td>
<td>&amp;</td>
</tr>
<tr>
<td>'</td>
<td></td>
<td>’</td>
</tr>
<tr>
<td>&quot;</td>
<td></td>
<td>&quot;</td>
</tr>
</tbody>
</table>

- Special characters for internal entities:
  
<table>
<thead>
<tr>
<th>Internal Entity</th>
<th>External Entity</th>
</tr>
</thead>
<tbody>
<tr>
<td>ä</td>
<td>ä</td>
</tr>
<tr>
<td>Ä</td>
<td>Ä</td>
</tr>
<tr>
<td>ö</td>
<td>ë</td>
</tr>
<tr>
<td>Ö</td>
<td>Ö</td>
</tr>
<tr>
<td>ü</td>
<td>ü</td>
</tr>
<tr>
<td>Ü</td>
<td>Ü</td>
</tr>
<tr>
<td>ß</td>
<td>ß</td>
</tr>
</tbody>
</table>
## XML Document: Grammar

<table>
<thead>
<tr>
<th>XMLDocument</th>
<th>::= Prolog XMLElement Comment*</th>
</tr>
</thead>
<tbody>
<tr>
<td>Prologue</td>
<td>::= XMLDeclaration? Comment* (DTDDescription Comment*)?</td>
</tr>
<tr>
<td>XMLDeclaration</td>
<td>::= &lt;?xml version=&quot;nr&quot; (encoding=&quot;code&quot;)? (standalone=&quot;yesno&quot;)?&gt;</td>
</tr>
<tr>
<td>Comment</td>
<td>::= &lt;!-- CharData --&gt;</td>
</tr>
<tr>
<td>XMLElement</td>
<td>::= &lt;Tagname Attribute*&gt; Elementcontent &lt;/Tagname&gt;</td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td>Elementcontent</td>
<td>::= CharData? ((XMLElement</td>
</tr>
<tr>
<td>Attribute</td>
<td>::= Attributename=&quot;Attributevalue&quot;</td>
</tr>
<tr>
<td>Reference</td>
<td>::= &amp; Entityname ;</td>
</tr>
</tbody>
</table>

Note: The XML grammar here ignores only a few parts that are not relevant for the lecture. CharData stands for character sequence. Comments shall not contain the symbol '-'. standalone="yes" requires DTD definition within the document.
XML as Data Model

- **Logical document structure:** Tree

- **Node properties:**
  - **Type:** document, element, attribute, comment, etc.
  - **Name:** element name, attribute name, etc.
  - **Content/Value:** element content, attribute value

- **Element node relationships:**
  - **Hierarchy:** parent/child, ancestor/descendant
  - **Sequence:** preceding/succeeding sibling
  - **Position:** absolute, relative, range
Flexibility of XML

- XML provides many possibilities to structure XML documents
  - Document type (schema) not explicitly defined/known

- Problems:
  - Applications are written for certain document types
  - Missing standard for data exchange
  - Significant failure potential (data formats, missing data fields, etc.)

- Solution: Restricted Flexibility
  - Definition of common data format
  - Standard classes of XML documents

- Approaches for defining document types/schemas
  - XML DTD
  - XML Schema
Document Type Definition (DTD)

- Uptonow focus was only on syntactical properties of XML
  - Open question: which tags can occur in an XML document?

- **DTD:** Grammatical description of elements and attributes for a class of XML documents
  - DTD defines which tags/markups can occur where in a document of that type
  - DTD allows specification and reuse of different document types which might come with different markups

- **Standard DTDs provide the basis for compatibility in case of data exchange**
### Example: Bookstore XML Document

<table>
<thead>
<tr>
<th>XML Document Header</th>
<th>&lt;?xml version=&quot;1.0&quot; encoding=&quot;UTF-8&quot;?&gt;</th>
</tr>
</thead>
<tbody>
<tr>
<td>XML Prologue with internal XML DTD</td>
<td>&lt;!DOCTYPE bookstore [</td>
</tr>
<tr>
<td></td>
<td>&lt;!ELEMENT bookstore (book)*&gt;</td>
</tr>
<tr>
<td></td>
<td>&lt;!ELEMENT book (title, author+, price?)&gt;</td>
</tr>
<tr>
<td></td>
<td>&lt;!ATTLIST book genre CDATA #REQUIRED&gt;</td>
</tr>
<tr>
<td></td>
<td>&lt;!ELEMENT title (#PCDATA)&gt;</td>
</tr>
<tr>
<td></td>
<td>&lt;!ELEMENT author ( name</td>
</tr>
<tr>
<td></td>
<td>&lt;!ELEMENT price (#PCDATA)&gt;</td>
</tr>
<tr>
<td></td>
<td>&lt;!ELEMENT name (#PCDATA)&gt;</td>
</tr>
<tr>
<td></td>
<td>&lt;!ELEMENT first-name (#PCDATA)&gt;</td>
</tr>
<tr>
<td></td>
<td>&lt;!ELEMENT last-name (#PCDATA)&gt;</td>
</tr>
<tr>
<td></td>
<td>]&gt;</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>XML Document Element</th>
<th>&lt;bookstore&gt;</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>&lt;book genre=&quot;autobiography&quot;&gt;</td>
</tr>
<tr>
<td></td>
<td>&lt;title&gt;The Autobiography of Benjamin Franklin&lt;/title&gt;</td>
</tr>
<tr>
<td></td>
<td>&lt;author&gt;</td>
</tr>
<tr>
<td></td>
<td>&lt;first-name&gt;Benjamin&lt;/first-name&gt;</td>
</tr>
<tr>
<td></td>
<td>&lt;last-name&gt;Franklin&lt;/last-name&gt;</td>
</tr>
<tr>
<td></td>
<td>&lt;/author&gt;</td>
</tr>
<tr>
<td></td>
<td>&lt;price&gt;8.99&lt;/price&gt;</td>
</tr>
<tr>
<td></td>
<td>&lt;/book&gt;</td>
</tr>
<tr>
<td></td>
<td>&lt;book genre=&quot;novel&quot;&gt;</td>
</tr>
<tr>
<td></td>
<td>...</td>
</tr>
<tr>
<td></td>
<td>&lt;/book&gt;</td>
</tr>
<tr>
<td></td>
<td>&lt;/bookstore&gt;</td>
</tr>
</tbody>
</table>
DTD Declaration: Grammar

| DTDDeclaration | ::= <!DOCTYPE Documentelementname ExterneDTDDeclaration? ([MarkupDeclaration*])? > |
| ExterneDTDDeclaration | ::= SYSTEM "URI" | PUBLIC "PublicName" "URI" |
| MarkupDeclaration | ::= ElementDeclaration | AttributeDeclaration | EntityDeclaration | Comment |

Note: The XML grammar here ignores only a few parts that are not relevant for the lecture

SYSTEM defines URI of the DTD for the private usage in the document

PUBLIC assigns a unique name to the DTD, which can then be used by the XML processor
# DTD Element Declaration

- **ElementDeclaration**: define the structure of the content of the given element

<table>
<thead>
<tr>
<th>Syntax</th>
<th>&lt;!ELEMENT Elementname Contentmodel&gt;</th>
</tr>
</thead>
</table>

- **Contentmodel**: a regular expression which specifies the structure of the content

<table>
<thead>
<tr>
<th>Subelement</th>
<th>&lt;!ELEMENT name (a)&gt;</th>
</tr>
</thead>
<tbody>
<tr>
<td>Element value</td>
<td>&lt;!ELEMENT name (#PCDATA)&gt;</td>
</tr>
<tr>
<td>Empty element</td>
<td>&lt;!ELEMENT name EMPTY&gt;</td>
</tr>
<tr>
<td>Arbitrary content</td>
<td>&lt;!ELEMENT name ANY&gt;</td>
</tr>
</tbody>
</table>

  - PCDATA → Parsed Character DATA
  - EMPTY → no child elements or character data
  - ANY → zero or more child elements of any declared type, as well as character data (mixed content)

- **Constructor**

<table>
<thead>
<tr>
<th>Sequence</th>
<th>&lt;!ELEMENT name (a, b)&gt;</th>
</tr>
</thead>
<tbody>
<tr>
<td>Choice</td>
<td>&lt;!ELEMENT name (a</td>
</tr>
</tbody>
</table>

- **Cardinality**

<table>
<thead>
<tr>
<th>Optional (zero or one)</th>
<th>&lt;!ELEMENT name (a)&gt;?</th>
</tr>
</thead>
<tbody>
<tr>
<td>Optional &amp; repetetive</td>
<td>&lt;!ELEMENT name (a)*&gt;</td>
</tr>
<tr>
<td>Required &amp; repetetive</td>
<td>&lt;!ELEMENT name (a)+&gt;</td>
</tr>
</tbody>
</table>

- **Using parentheses, content models can be combined arbitrarily**
DTD Element Declaration: Grammar

ElementDeclaration ::=  <!ELEMENT Elementname Contentmodel>

Contentmodel ::=  (Elementtype)
                   | ANY
                   | EMPTY

Elementtype ::=  #PCDATA
                 | Elementname
                 | (Elementtype)
                 | Elementtype*
                 | Elementtype+
                 | Elementtype?
                 | Elementtype, Elementtype
                 | Elementtype | Elementtype

Example:
<!ELEMENT emptyelement EMPTY>
<!ELEMENT container ANY>
<!ELEMENT email (from, to, cc*, bcc*, subject?, body)>
<!ELEMENT book (((toc, abstract?) | (abstract?, toc)), chapter+, literature)>
<!ELEMENT text (#PCDATA | separator)*>
DTD Attribute Declaration

- XML elements can be annotated by arbitrary many attributes

<table>
<thead>
<tr>
<th>Syntax</th>
<th>&lt;!ATTLIST Elementname Attributename Type Constraint ...&gt;</th>
</tr>
</thead>
</table>

- Attribute Type

<table>
<thead>
<tr>
<th>Character type</th>
<th>CDATA</th>
</tr>
</thead>
<tbody>
<tr>
<td>Enumeration</td>
<td>(value_1</td>
</tr>
<tr>
<td>Unique element id</td>
<td>ID</td>
</tr>
<tr>
<td>Element id reference</td>
<td>IDREF(S)</td>
</tr>
<tr>
<td>Entity reference</td>
<td>ENTITY(S)</td>
</tr>
</tbody>
</table>

- Attribute Constraint

<table>
<thead>
<tr>
<th>Required attribute</th>
<th>#REQUIRED</th>
</tr>
</thead>
<tbody>
<tr>
<td>Optional attribute</td>
<td>#IMPLIED</td>
</tr>
<tr>
<td>Fixed attribute value</td>
<td>#FIXED &quot;Value&quot;</td>
</tr>
</tbody>
</table>

- Different orders of attribute name, type, and constraint does not have an effect
DTD Attribute Declaration: Grammar

XMLAttributelist ::= <!ATTLIST Elementname Attribute*>  
Attribute ::= Attributename Attributetype Constraint  
Attributetype ::= CDATA  
| (Value | Value)*  
| Attributename ID  
| Attributename IDREFS  
Constraint ::= #REQUIRED CharData?  
| #IMPLIED CharData?  
| #FIXED CharData

Example:

<!ELEMENT book (title, price)>  
<!ATTLIST book genre CDATA #REQUIRED authors IDREFS # IMPLIED>  

<!ELEMENT author (name)>  
<!ATTLIST author id #REQUIRED>  

<!ELEMENT price (#PCDATA)>  
<!ATTLIST price currency (CHF | EUR | USD) #IMPLIED>  

<!ELEMENT timestamp (id)>  
<!ATTLIST timestamp hour CDATA #IMPLIED minutes CDATA #IMPLIED seconds CDATA #FIXED "0">
Example: ID/IDREF Attributes

```
<?xml version="1.0"?>
<!-- DTD declaration ... -->
<bookstore>
  ...
  <author id="hm">
    <first-name>Herman</first-name>
    <last-name>Melville</last-name>
  </author>
  <book genre="novel" authors="hm">
    <title>The Confidence Man</title>
    <price currency="USD">11.99</price>
  </book>
  ...
</bookstore>
```
DTD Declaration

External DTD Declaration

```xml
<?xml version="1.0" encoding="ISO-8859-1"?>
<!DOCTYPE test PUBLIC "-//DBRG//DTD MyDoc V1.0//DE"
"http://www.fgcz.ch/mydoc.dtd">
<test> "test" is the root </test>
```

Internal DTD Declaration

```xml
<?xml version="1.0" encoding="ISO-8859-1"?>
<!DOCTYPE test [ <!ELEMENT test (#PCDATA)> ]>
<test> "test" is the root </test>
```

Mixed Usage

```xml
<?xml version="1.0" encoding="ISO-8859-1"?>
<!DOCTYPE test SYSTEM "http://www.fgcz.ch/mydoc.dtd"
[ <!ENTITY hallo "hallo world"> ]>
<test> &hallo; </test>
```
Example: External DTD Declaration

```xml
<?xml version="1.0" encoding="ISO-8859-1"?>
<!-- This file represents a fragment of a book store inventory database -->
<!DOCTYPE bookstore SYSTEM "http://www.books.com/bookstore.dtd">
<bookstore>
  <book genre="autobiography">
    <title>The Autobiography of Benjamin Franklin</title>
    <author>
      <first-name>Benjamin</first-name>
      <last-name>Franklin</last-name>
    </author>
    <price currency="USD">8.99</price>
  </book>
  <book genre="novel">
    <title>The Confidence Man</title>
    <author>
      <first-name>Herman</first-name>
      <last-name>Melville</last-name>
    </author>
    <price currency="USD">11.99</price>
  </book>
  <book genre="philosophy">
    <title>The Gorgias</title>
    <author>
      <name>Plato</name>
    </author>
    <price currency="EUR">9.99</price>
  </book>
</bookstore>
```

bookstore.xml

```xml
<!ELEMENT bookstore (book)*>
<!ELEMENT book (title,author+,price?)>
<!ATTLIST book genre CDATA #REQUIRED>
<!ELEMENT title (#PCDATA)>
<!ELEMENT author (name |(first-name,last-name))>
<!ELEMENT price (#PCDATA)>
<!ATTLIST price currency (CHF | EUR | USD)>
<!ELEMENT name (#PCDATA)>
<!ELEMENT first-name (#PCDATA)>
<!ELEMENT last-name (#PCDATA)>
```

bookstore.dtd
Well-Formed vs. Valid XML Documents

- **well-formed**
  - Document follows ordered, syntactical XML properties (see grammar)

- **valid**
  - Document is well-formed and satisfies the schema given by a DTD

**Example:** incompletely marked up document fragment

```xml
<A><B>pipapo<C>hello world</C></A>
```

possible structures:

![Diagram showing possible structures of the XML document fragment.](image-url)
XML Namespaces

- Mechanism to include predefined element and attribute names given by a markup vocabulary
- Required to solve naming conflicts when several DTDs are included
- Declaration as attribute of the document element:

  ```xml
  <Elementname xmlns:Namensraum-ID = "Namensraum-URI">
  ...
  </Elementname>
  ```

- Example:

  ```xml
  <tourismusinformation
    xmlns:hotelinfo="http://www.hotels.ch"
    xmlns:eventinfo="http://www.events.ch">
    <hotelinfo:city>Bern</hotelinfo:city>
    <eventinfo:city>Winterthur</eventinfo:city>
  </tourismusinformation>
  ```
Disadvantages of DTDs

- **No XML syntax**
  - Positive: DTD syntax is concise, easy readable and understandable

- **No real data typing: all values are character sequences**
  - Document-oriented origin distinctive
  - Strictly limited representation and processing possibilities in applications

- **No integrity constraints, e.g., no exact lower/upper borders for cardinality of sequences**
  - Semantical correctness cannot be checked

- **No Namespaces**

- **Conclusions**: Additional typing (schema description) desired to better support processing of XML documents in applications ⇒ XML Schema
XML Schema

- Mechanism to separate tags and types (structure vs. typing)
- Modular schema definition based on rich type system
- Introduces basic data types and allows type construction including type inheritance
- W3C Recommendation since May 2001
- Schema definition carried out in XML syntax ("looks bloated")
- Frame of a schema definition:

```xml
<xsd:schema xmlns:xsd= "http://www.w3.org/2001/XMLSchema">
 ...
</xsd:schema>
```
XML Schema type system obviously influenced by SQL and Java!
Data Types

- **Simple vs. Complex Data Types**
  - Simple data types do not have subelements and attributes in contrast to complex ones
  - Both applicable for XML elements as well as for XML attributes
  - All DTD attribute types are available in XML Schema

- **Predefined vs. Constructed Data Types**
  - Type constructors:
    - **Restriction**: Restrict simple data types
    - **List**: List of simple data types
    - **Union**: Union of simple data types
  - Subtyping:
    - **Extension**: Extend data types by XML elements/attributes
      (Result is a complex data type)
Simple User-Defined Data Types

- **Restriction**
  ```xml
  <xsd:simpleType name="Schoolgrade">
    <xsd:restriction base="xsd:short">
      <xsd:minInclusive value="1"/>
      <xsd:maxInclusive value="6"/>
    </xsd:restriction>
  </xsd:simpleType>
  ```

- **List**
  ```xml
  <xsd:simpleType name="Hobbies">
    <xsd:list itemType="xsd:string"/>
  </xsd:simpleType>
  ```

- **Union**
  ```xml
  <xsd:simpleType name="Vehicle">
    <xsd:union memberType="Car Bicycle ..."/>
  </xsd:simpleType>
  ```

```xml
<xsd:simpleType name="Color">
  <xsd:restriction base="xsd:string">
    <xsd:enumeration value="red"/>
    <xsd:enumeration value="blue"/>
    <xsd:enumeration value="white"/>
  </xsd:restriction>
</xsd:simpleType>
```
Restriction

- **Lower/Upper borders for ordered domains**
  - minInclusive, maxInclusive (border values included)
  - minExclusive, maxExclusive (border values excluded)

- **Character sequence length**
  - length, minLength, maxLength (exact, minimum, maximum length)

- **List cardinality**
  - length, minLength, maxLength (exact, minimum, maximum number of list elements)

- **Numeric value length**
  - totalDigits (maximum number of digits)
  - fractionDigits (number of fraction digits)

- **Enumeration of allowed values**
  - enumeration
Complex, User-Defined Data Types

- Sequence stands for list of elements with fixed order
- Choice allows to choose an element from specified alternatives
- Type inheritance by extension

```xml
<xsd:complexType name="Car" abstract="true">
  <xsd:complexContent>
    <xsd:sequence>
      <xsd:element name="model" type="xsd:string"/>
      <xsd:element name="year" type="xsd:integer"/>
    </xsd:sequence>
    <xsd:choice>
      <xsd:element ref="ottomotor"/>
      <xsd:element ref="dieselmotor"/>
    </xsd:choice>
  </xsd:complexContent>
  <xsd:attribute name="doors" type="xsd:integer" default="4"/>
</xsd:complexType>

<xsd:complexType name="BusinessCar">
  <xsd:complexContent>
    <xsd:extension base="Car">
      <xsd:sequence>
        <xsd:element name="Firm" type="xsd:string"/>
        <xsd:element name="LastCheck" type="xsd:date"/>
      </xsd:sequence>
    </xsd:extension>
  </xsd:complexContent>
</xsd:complexType>
```
Example: DTD vs. XML Schema

XML DTD

```xml
<!ELEMENT book (title, author+, price?)>
<!ATTLIST book genre (autobiography | novel | philosophy) #REQUIRED>
<!ELEMENT author (#PCDATA)>
<!ELEMENT title (#PCDATA)>
<!ELEMENT price (#PCDATA)>
```

XML Schema

```xml
<?xml version="1.0"?>
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema">
  <xsd:element name="book" type="Book"/>
  <xsd:complexType name="Book">
    <xsd:sequence>
      <xsd:element name="title" type="xsd:string"/>
      <xsd:element name="author" type="xsd:string" minOccurs="1" maxOccurs="unbounded"/>
      <xsd:element name="price" type="xsd:decimal" minOccurs="0" maxOccurs="1"/>
    </xsd:sequence>
    <xsd:attribute name="genre" type="Genre" #REQUIRED>
    </xsd:complexType>
  <xsd:simpleType name="Genre">
    <xsd:restriction base="xsd:string">
      <xsd:enumeration value="autobiography"/>
      <xsd:enumeration value="novel"/>
      <xsd:enumeration value="philosophy"/>
    </xsd:restriction>
  </xsd:simpleType>
</xsd:schema>
```
DTD

- Grammar
- Elements + Attributes
- Only Part-Of-Relationships
- PCDATA, CDATA

- Included in the XML Document Prologue
- Example:

```xml
<!DOCTYPE bookstore SYSTEM "bookstore.dtd">
```

XML Schema

- Structure and Typing
- Elements, Attribute, Simple and Complex Data Types
- Type Inheritance
- Rich Data Type System

- Included via Document Element Attribute
- Example:

```xml
<bookstore
    xmlns:xsd="http://www.w3.org/..."
    xsd:schemaLocation="bookstore.xsd">
  ...
</bookstore>
```
XML Compliance

- **well-formed**
  - Document follows ordered, syntactical XML properties (see grammar)

- **valid**
  - Document is well-formed and satisfies the schema given by a DTD

- **namespace-compliant**
  - Document is well-formed and all markups are valid according the declaration of the referenced namespace

- **schema-compliant**
  - Document is well-formed and satisfies the schema given by an XML Schema
Example: Valid but not Schema-Compliant

```xml
<?xml version="1.0"?>
<bookstore
    xmlns:xsd="http://www.w3.org/2001/XMLSchema"
    xsi:schemaLocation="bookstore.xsd">
    <book genre="autobiography">
        <title>The Autobiography of Benjamin Franklin</title>
        <author>Benjamin Franklin</author>
        <price>8.99</price>
    </book>

    <book genre="novel">
        <title>The Confidence Man</title>
        <author>Herman Melville</author>
        <price>11.99</price>
    </book>

    <book genre="action">
        <title>The Gorgias</title>
        <author>Plato</author>
        <price>9.99</price>
    </book>
</bookstore>
```

violates schema-compliance
W3C XML Architecture

CSS  Cascading Style Sheets
     W3C Recommendation 17.12.1996, revised 11.01.1999 (level 1)
     W3C Recommendation 12.05.1998 (level 2)

DOM  Document Object Model (DOM) Level 1

RDF/RDFS  Resource Description Framework
           W3C Recommendation 22.02.1999

XLink  Extensible Linking Language
       W3C Recommendation 27.05.2001

XML  Extensible Markup Language
     W3C Recommendation, 10.02.1998

XMLns  Namespaces in XML
       W3C Recommendation, 14.01.1999

XMLschema  Schemas in XML
           W3C Recommendation, 02.05.2001

XPointer  XML Pointer Language
          W3C Recommendation, 27.06.2001

XSL  Extensible Stylesheet Language
     W3C Recommendation 15.10.2001
Standard DTDs: Example

- **Official W3C DTDs:**
  - MathML: Mathematical Markup Language
  - SMIL: Synchronized Multimedia Integration Language
  - Appel: A Privacy Preference Exchange Language
  - RDF: Resource Description Framework
  - RDFS: Resource Description Framework Schemas
  - SVG: Scaleable Vector Graphics
  - HTML: Hypertext Markup Language

- **E-Commerce:**
  - OFX: Open Financial Exchange

- **Research:**
  - CML: Chemical Markup Language

- **Computer, Communication:**
  - CDF: Channel Definition Format
  - OSD: Open Software Distribution
  - WIDL: Web Interface Definition Language
  - JSML: Java Speech Markup Language
  - DocBook
  - XMLEDI: Electronic Data Interchange
Conclusions

- We know how XML documents are structured!
  - XML defines tree structure
    - Semi-structured data model provides flexibility
    - Well-formed XML documents
  - XML DTD defines grammar
    - Makes document type explicit
    - Single data type: character sequences; no constraints beyond cardinality specifications
    - Valid XML documents
  - XML Namespaces define vocabulary
    - Conflict resolution
  - XML Schema provides rich data type system à la SQL/Java
    - Schema-compliant XML documents

- We now want to know how XML documents can be processed in applications
  - XML Processors