# Conception of Information Systems
# Lecture 2: XML

15 March 2005

http://lsirwww.epfl.ch/courses/cis/2005ss/

# Outline

1. Motivation for XML

2. XML Basics
   - Well-formed XML
   - Document Type Definition (DTD)
   - Entities
   - Namespaces

3. Advanced XML
   - Document Object Model (DOM)
   - XML Schema

4. References

# 1. Motivation for XML

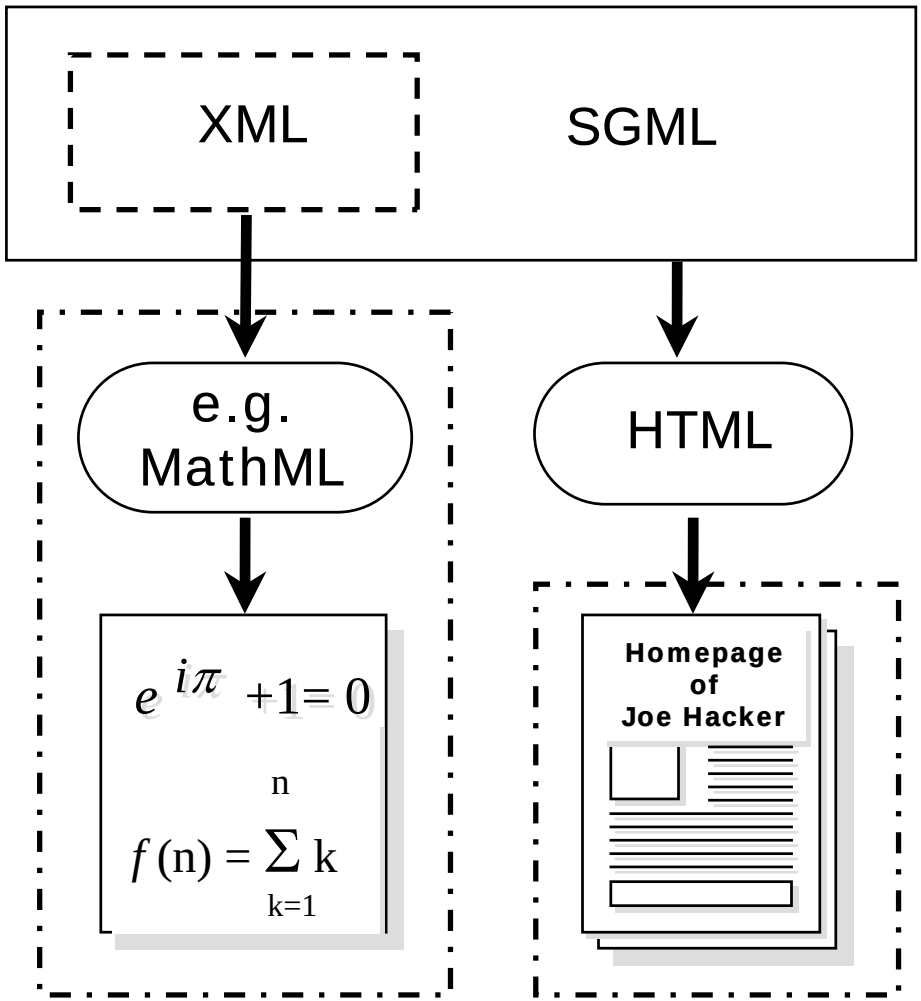
- The Web is
  - A huge collection of hypermedia documents
  - A gateway to databases and applications
  - A platform for interacting with services (Web Services)

- It started with HTML
  - Markup language to define document layout
  - Hyperlinks for navigation
  - Interactive forms

- Limitations of HTML
  - Structure of data expressed as layout (example)
  - Semantics of data hard to analyse and difficult to share
  - No schemas, no constraints

- Thus XML (eXtensible Markup Language) has been developed
  - Markup language to define structured documents
  - Document schemas to fix the structure of documents
  - User-defined markup to express semantics
  - XML architecture for processing and extended functionality

# Relationship between HTML and XML



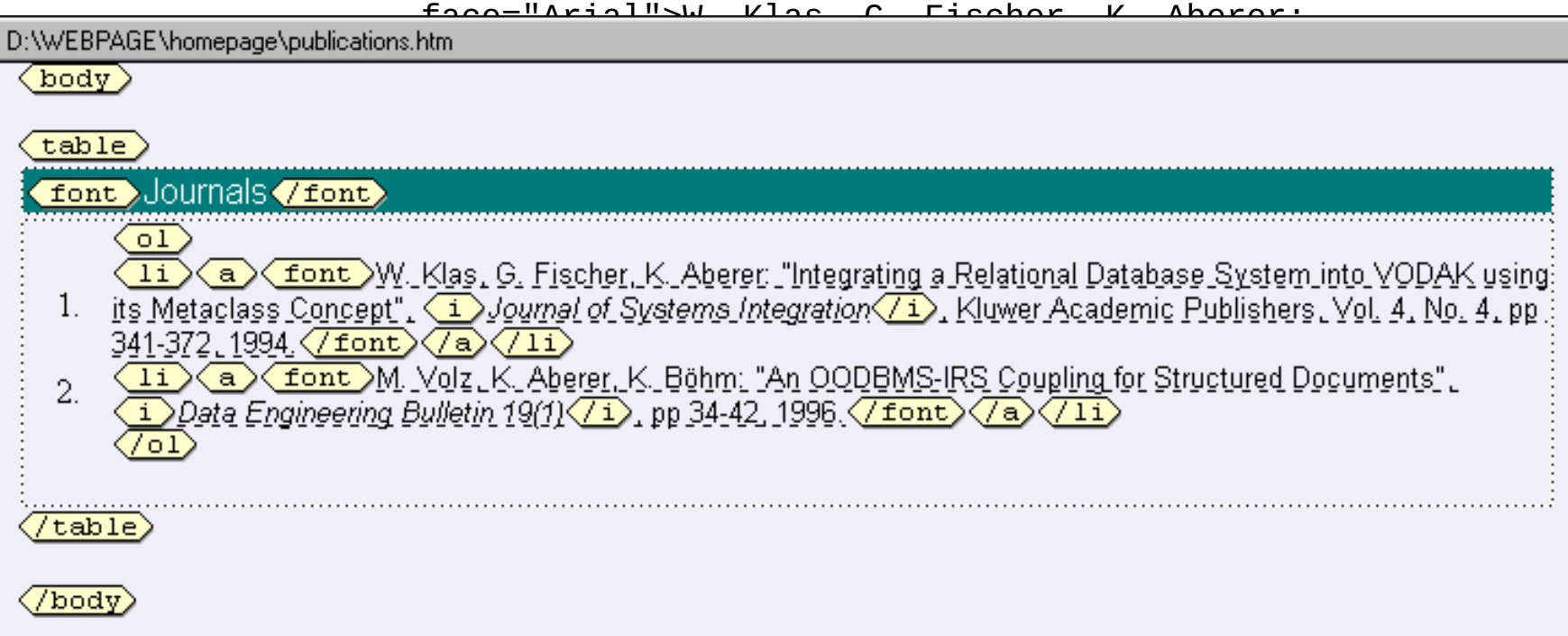

**Data model**

**Document type "Schema"**

**Document Database**

# Example: Data in HTML vs. XML (1)

```
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1">
<title>PUBLICATIONS</title>
<body bgcolor="#F2EFFC" link="#0000FF" vlink="#800080">
<table border="0" width="100%">
    <tr><td width="100%" bgcolor="#008080"><font color="#FFFFFF"
        face="Arial">Journals</font></td>
    </tr><tr><td width="100%"><ol>
            <li><a name="_Ref443966216"><font size="2"
                face="Arial">W. Klas, G. Fischer, K. Aberer:
```

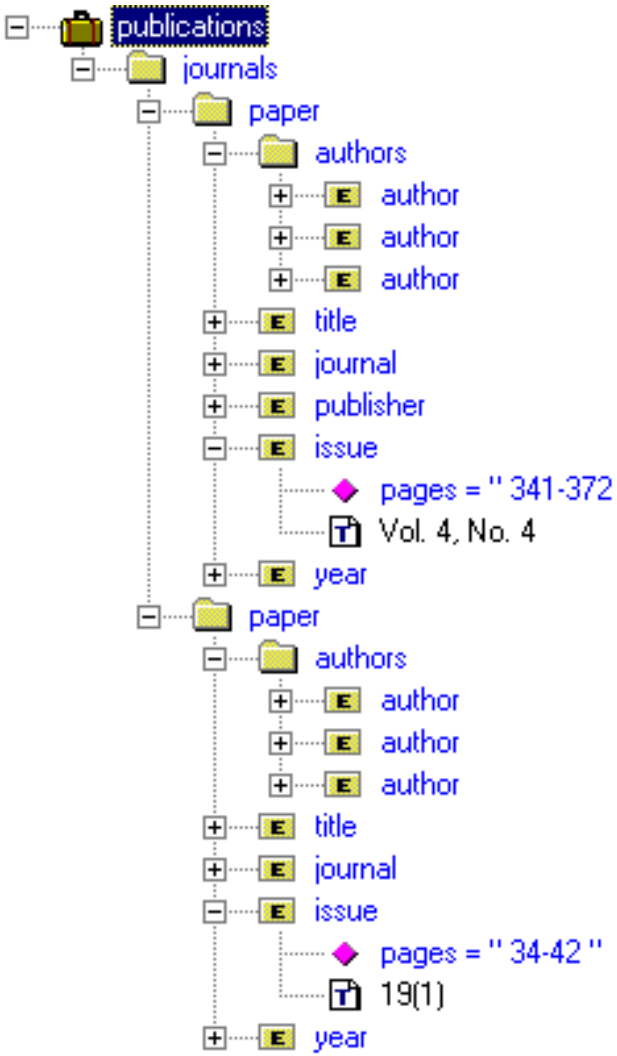



```
li>
```

# Example: Data in HTML vs. XML (2)

```
<publications>
        <journals>
                <paper>
                        <authors>
                                <author>W. Klas</author>
                                <author>G. Fischer</author>
                                <author>K. Aberer</author>
                        </authors>
                        <title>Integrating a Relational Database System
                        into VODAK using its Metaclass Concept</title>
                        <journal>Journal of Systems Integration</journal>
                        <publisher>Kluwer Academic Publishers</publisher>
                        <issue pages='341-372'>Vol. 4, No. 4</issue>
                        <year>1994</year>
                </paper>
                <paper>
                        <authors>
                                <author>M. Volz</author>
                                <author>K. Aberer</author>
                                <author>K. Böhm</author>
                        </authors>
                        <title>An OODBMS-IRS Coupling for Structured Documents</title>
                        <journal>Data Engineering Bulletin</journal>
                        <issue pages='34-42'>19(1)</issue>
                        <year>1996</year>
                </paper>
        </journals>
</publications>
```

# Example: Data in HTML vs. XML (3)



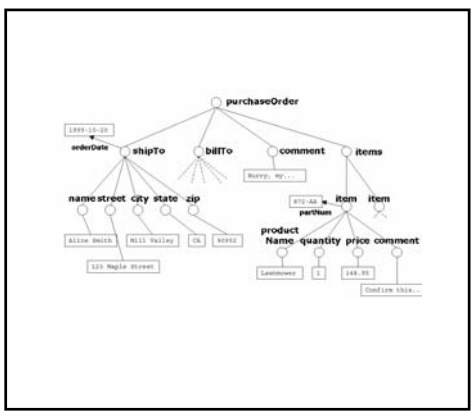

```
<!DOCTYPE publication [
<!ELEMENT publications (journals,conferences,
          books)>
<!ELEMENT journals (paper)>
<!ELEMENT paper (authors, title,
          journal, publisher?, issue, year)>
<!ELEMENT authors (author*)>
<!ELEMENT author, title, journal, publisher,
          year (#PCDATA)>
<!ELEMENT issue (#PCDATA)>
<!ATTLIST issue pages CDATA #REQUIRED>
]>
```

**Document type "Schema"**

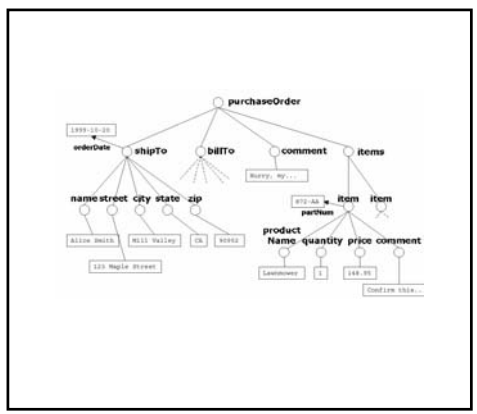
# Data and Documents

- "Serialization"

**Document = medium for exchange of information**








**Communication**

**Information system 1**

**Information system 2**

# What is XML ?

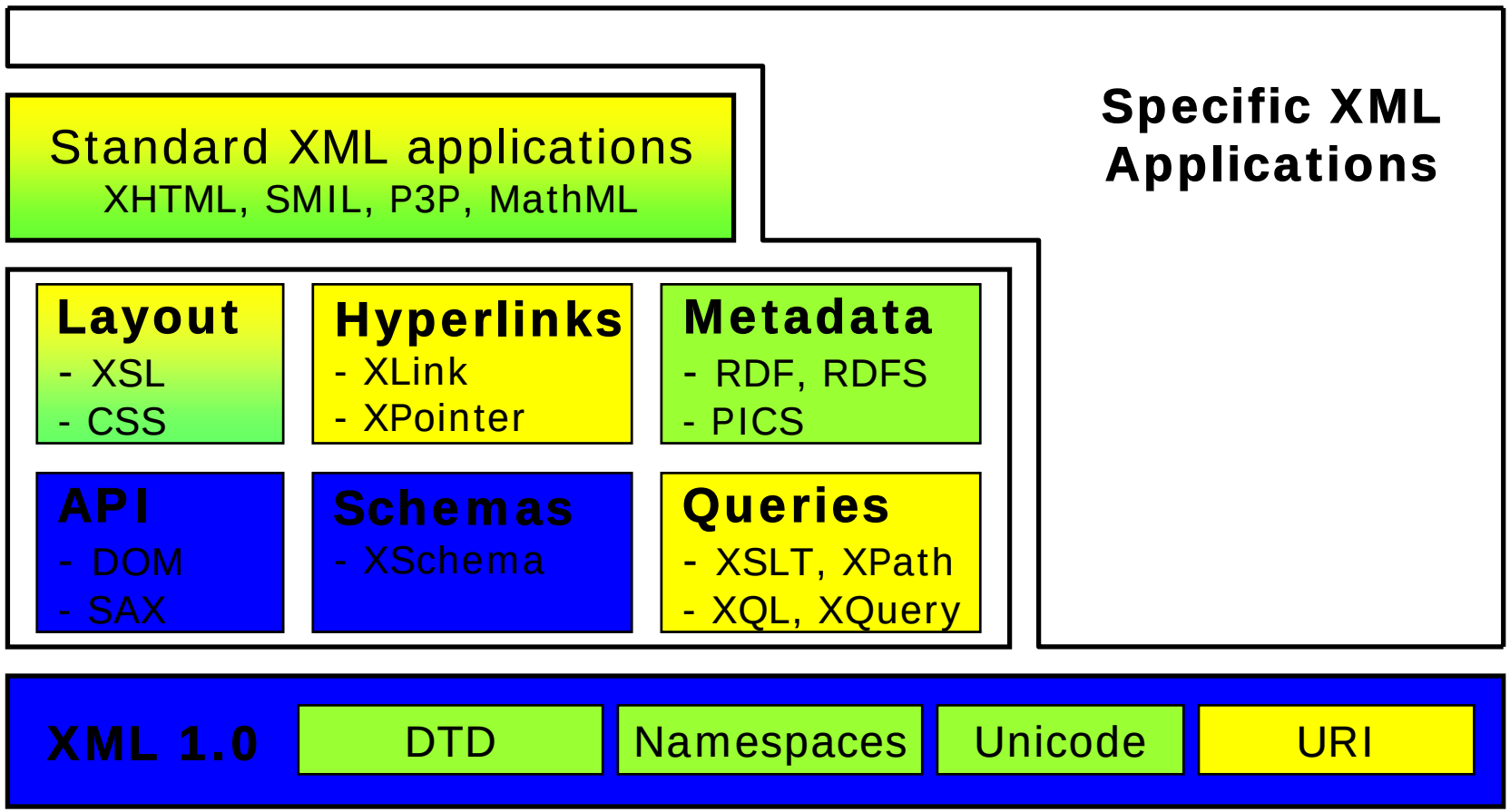
- Interpretation depends on viewpoint and intended use
  - a language to describe the structure of documents.
  - the foundation of the W3C architecture for Hypermedia documents on the Web.
  - the successor of HTML.
  - a method to put structured data into text documents.
  - a standard data exchange format.
  - a data model for semi-structured (partially structured) data.

- What are the main characteristics of the XML language ?
  - No schema is required, but schemas can be used
  - Flexible data model: complex data structures, optional elements
  - Canonical serialization for data exchange
  - Use of names allows to impose (agreed) semantics structural elements.
  - Human-readability
  - Widely accepted and standardized
  - Wide availability of tools for processing XML data

# Why Do We Need XML ?

- Tim Bray: "XML will be the ASCII of the Web – basic, essential, unexciting"
- Distributed information systems
  - XML as data model for managing semi-structured data
  - distinction between documents and data disappears
  - XML as canonical model to integrate heterogeneous data
  - XML as canonical data format to exchange data among information systems
- Web information systems
  - separation of layout and structure
  - better support to keep data consistent
  - reuse of data
  - client-side processing
  - more semantics available for more intelligent processing (personalization, agents, search engines)
- Electronic business
  - Integration of businesses processes
  - messages and contracts represented in XML
  - replaces former EDI formats
  - Standardization
- XML is an **integration technology**

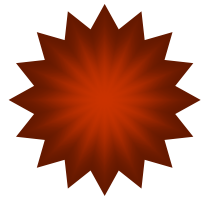
# XML Architecture



# 2. XML Syntax

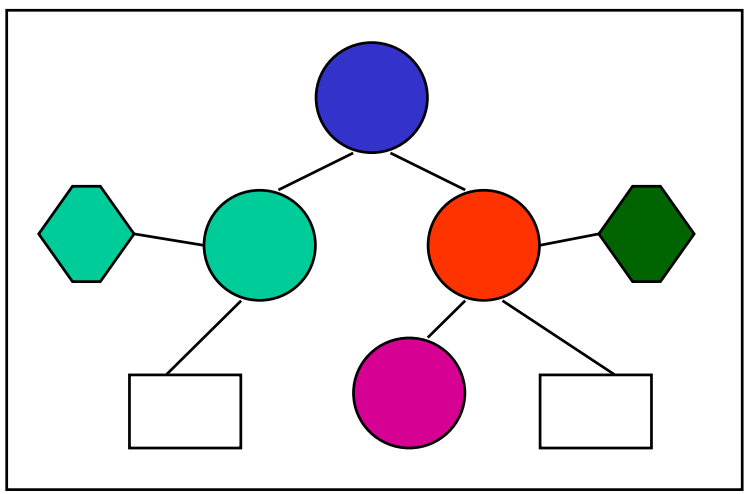
- Well-formed XML
- Document Type Definitions
- Entities
- Namespaces

# 2.1 Well-formed XML



- Well-formed XML conforms to a basic XML syntax and some semantic constraints for well-formedness

- Main concepts
  - Elements: used to structure the document, identify a portion of the document
  - Attributes: associate data values with elements, used to reduce the number of elements and for typed data
  - Character data (PCDATA): textual content of the document



```
<journal>
   <issue page=1>
      PCDATA,
      the content
   </issue>
   <issue page=2>
      <last/>
      more PCDATA
   </issue>
</journal>
```

# Well-Formed XML Syntax

- Syntax (excerpt)

```
document        ::=   prolog element Misc*
element         ::=   EmptyElemTag | STag content Etag
STag            ::=   '<' Name (S Attribute)* S? '>'
ETag            ::=   '</' Name S? '>'
Attribute       ::=   Name Eq AttValue
```

- Syntactic Properties
  - single root element
  - tag names start with letter
  - tags must be properly nested
  - hierarchic structure induced by parents-child relationship
  - special syntax for empty tags

- Semantic Constraints
  - Start and end tag name must match
  - Attribute names within an element are unique

# Structure of a Well-formed XML Document

**XML document**

**Prologue**

```
<?xml version="1.0" ?>
```

**Document Type Definition**

```
<!DOCTYPE publication [
<!ELEMENT publications (journals, conferences, books)>
...
<!ELEMENT author (#PCDATA)>
<!ELEMENT issue (#PCDATA)>
<!ATTLIST issue pages CDATA #REQUIRED>
<!ENTITY JSI " <journal>Journal of Systems Integration</journal>
               <publisher>Kluwer Academic Publishers</publisher>">
]>
```

**Root Document Element**

```
<publications>
```

**Document**

```
<journals>
...
&JSI;
...
</publications>
```

# 2.2 XML Document Type Definitions

- Declarations
  - Definition of element and attribute names
- Content Model (regular expressions)
  - Association of attributes with elements
  - Association of elements with other elements (containment)
  - Order and cardinality constraints

# Element Declarations

- Basic form
  - `<!ELEMENT elementname (contentmodel)>`
  - Contentmodel determines which other elements can be contained
  - Given by a regular expression

- Atomic contents
  - Element content
    `<!ELEMENT example ( a )>`
  - Text content
    `<!ELEMENT example (#PCDATA)>`
  - Empty Element
    `<!ELEMENT example EMPTY>`
  - Arbitrary content
    `<!ELEMENT example ANY>`

# Element Declarations

- Sequence

```
<!ELEMENT example ( a, b )>
```

- Alternative

```
<!ELEMENT example ( a | b )>
```

- Optional (zero or one)

```
<!ELEMENT example ( a )?>
```

- Optional and repeatable (zero or more)

```
<!ELEMENT example ( a )*>
```

- Required and repeatable (one or more)

```
<!ELEMENT example ( a )+>
```

Mixed content

```
<!ELEMENT example (#PCDATA | a)*>
```

- Content model can be grouped by parentheses
- Cyclic element containment is allowed

# Attribute Declarations

- Each element can be associated with an arbitrary number of attributes
- Basic form
  - `<!ATTLIST Elementname Attributename Type Default`
    `Attributename Type Default`
    `... >`

- Example:
  Document Type Definition

```
<!ELEMENT shipTo (     #PCDATA)>
<!ATTLIST shipTo       country CDATA #REQUIRED "US"
                       state CDATA #IMPLIED
                       version CDATA #FIXED "1.0"
                       payment (cash|creditCard) "cash">
```

  Document

```
     <shipTo  country="Switzerland"
                       version="1.0"
                       payment="creditCard"> … </shipTo>
```

# Attribute Declarations - Types

- CDATA
  - String
  - `<!ATTLIST example HREF CDATA #REQUIRED>`
- Enumeration
  - Token from given set of values, Default possible
  - `<!ATTLIST example selection ( yes | no | maybe ) "yes">`
- Possible Defaults
  - Required attribute: `#REQUIRED`
  - Optional attribute: `#IMPLIED`
  - Fixed attribute: `#FIXED "value"`
  - Default for enumeration: `"value"`
- Other attribute types: IF, IDREF, ENTITY, ENTITIES, NOTATION, NAME, NAMES, NMTOKEN, NMTOKENS

# ID/IDREF

- ID, IDREF
  - ID is a unique identifier **within** the document
  - IDREF is a reference to an ID
  - Referential integrity checked by the parser
  - ID's determined by the application
  - **<!ATTLIST example identity ID #IMPLIED**
    **reference IDREF #IMPLIED>**

# Example: ID/IDREF

**DTD fragment:**

```
<!ATTLIST fig          id      ID    #IMPLIED>
<!ATTLIST figref       refid   IDREF #IMPLIED>
```

**Document fragment:**

```
<chapter>
  <title>Apples<\title>
  <para>
    <fig id="1">
     <caption> this is a figure<\caption>
    <\fig>
  <para>
<\chapter>
<chapter>
  <title>Frogs<\title>
    <references>
      <figref refid="1"\>
    <\references>
<\chapter>
```

# Inclusion of XML Document Type Definitions

**External DTD Declaration**

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<!DOCTYPE test PUBLIC "-//Test AG//DTD test V1.0//EN"
                      SYSTEM "http://www.test.org/test.dtd">
<test> "test" is a document element </test>
```

**Internal DTD Declaration**

```
<!DOCTYPE test [ <!ELEMENT test EMPTY> ]>
<test/>
```

**Mixed usage**

```
<!DOCTYPE test SYSTEM "http://www.test.org/test.dtd" [
   <!ENTITY hello "hello world">
]>
<test>&hello;</test>
```

# More Constructs of Well-formed XML

- CDATA
  - Is not processed by the parser
  - Used for code examples etc.

```
...
<script>
   <![CDATA[
      if ( a < b ) {
         &subroutine(a,b)
      } else {
         &subroutine(b,a)
      }
   ]]>
</script>
...
```

- Comments

```
...
<!--
Comment may contain <tagnames> or
&entities; but not "--" -->

...
```

# Processing Instructions

- PI are not part of the document but calls to external applications
- Are forwarded to the application without change
- Poor programming practice ("hacker style")
- *<?xml* is reserved for the prolog of an XML document and is used to determine the character encoding

```
<?TARGETAPPLICATION
   Parameter, Program,
   etc.
?>
...
<?xml
   version="1.0"
   encoding="ISO-8859-1"
   standalone="no"
?>

...
<?php echo $title;?>
...
```

# 2.3 Entities

- Entities allow to organize XML documents physically
- Entities work like macros
- External entities
  - Distribution of one logical document over several physical files
  - Reduce the size of files
  - Organization of files
  - Integration of non-XML resources
  - Reuse of DTD fragments
- Internal entities
  - Factorization of repeating contents within a document
  - Better readability of document
  - Less code, reuse
  - Consistency

# Example

```
<!ENTITY journals SYSTEM "http://publications.org/journals.xml">
<!ENTITY conferences SYSTEM "http://publications.org/conferences.xml">
...
<publications>
&journals;
&conferences;
</publications>
```

http://publications.org/journals.xml:

```
<!ENTITY JSI " <journal>Journal of Systems Integration</journal>
               <publisher>Kluwer Academic Publishers</publisher>"
...
<journals>
<paper>
<authors>
<author>W. Klas</author><author>G. Fischer</author><author>K. Aberer</author>
</authors>
<title>Integrating a Relational Database System into VODAK using its
Metaclass Concept</title>
&JSI;
<issue pages='341-372'>Vol. 4, No. 4</issue>
<year>1994</year></paper>
...
```

# Parsed vs. Unparsed entities

- Internal entities become always part of the document and are parsed

```
<!ENTITY JSI " <journal>Journal of Systems Integration</journal>
               <publisher>Kluwer Academic Publishers</publisher>"
```

- External entities can either be parsed or not
  - Parsed entity

```
<!ENTITY journals SYSTEM "http://publications.org/journals.xml">
```

  - Unparsed entity: used to include non-XML data

```
<!ENTITY pic SYSTEM "logo.gif" NDATA GIF>
```

# General vs. Parameter Entities

- Difference in usage of the entities
  - General entities are used in the document content
  - Parameter entities are used in the DTD declarations

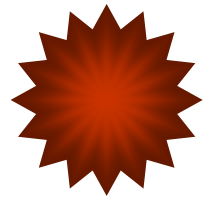
- Parameter entities
  - Allow to modularize DTDs
  - Fewer declarations in a DTD
  - Both external and internal
  - Always parsed
  - Different syntax

```
<!ENTITY % address "(name, street, zip)">
<!ELEMENT customer %address;>
<!ELEMENT supplier %address;>
```

# 2.4 XML Namespaces

- An **XML namespace** is a collection of names (markup vocabulary)
  - identified by a URI reference
  - used in XML documents as element and attribute names
- URIs are used just as unique identifiers, nothing else
  - in particular they do not refer to a DTD or schema
- Uses
  - universally agreed names
  - combination of names from different DTDs without name conflicts
  - But not combination of different DTDs

# Declaration



- Declaration of
  - Default namespace: xmlns (without :ns-name)
  - Identification by prefix: xmlns:ns (ns is the prefix)

- Declared in different places
  - In internal DTD by using default attributes
  - In document by using attributes

```
<?xml version= “1.0“?>
<!-- element names without prefix belong to “books” -->
 <book xmlns=’urn:loc.gov:book’
       xmlns:isbn= ’urn:ISBN.0-395-36341-6’>
   <title>XML Handbook</title>
   <isbn:number>1591240349</isbn:number>
 </book>
```

```
<!DOCTYPE doc [
<!ELEMENT doc (x)>
<!ELEMENT x EMPTY>
<!ATTLIST x xmlns CDATA #FIXED "http://www.jclark.com/"> ]>
<doc><x/></doc>
```

# 3. Standard XML applications

- Document Object Model
- XML Schema

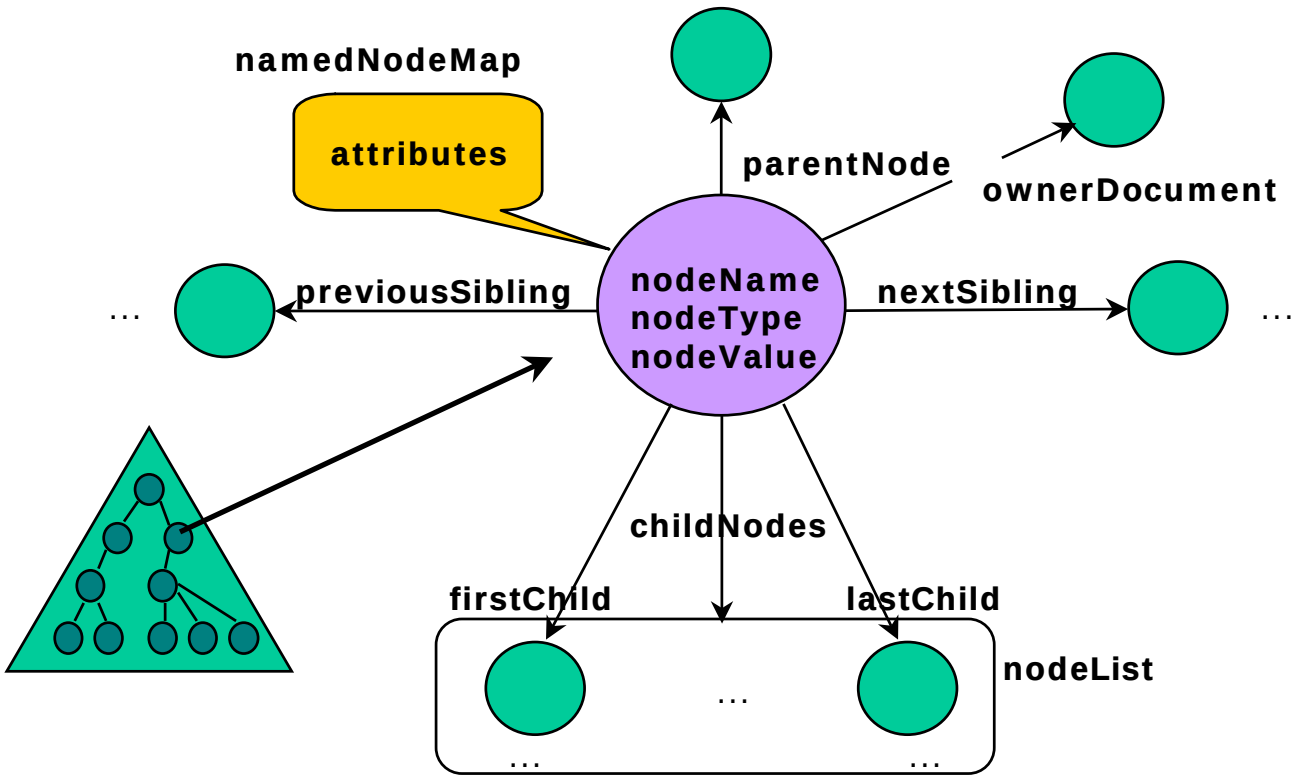
# 3.1 Document Object Model

- Standardized object-oriented API for accessing XML document
  - Specifies only interfaces (in IDL), not implementation
  - Views XML document as tree structure
  - Methods for navigation and manipulation
  - Not all concepts supported, e.g. DTDs, Entities

- Abstract Classes
  - Node: superclass for all constituents of a document
  - NodeList: representations of node lists
  - NamedNodeMap: attributes of an element

- Concrete Classes
  - Document, Element, Attribute, etc.

# Attributes and Methods of Abstract Class Node



# Example: Class Node Interface

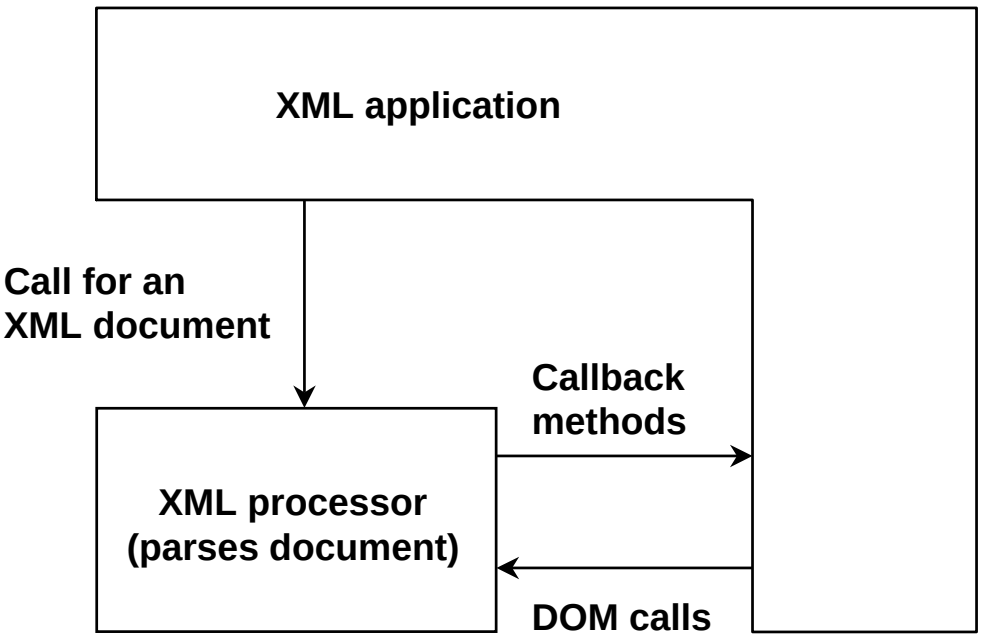
```
interface Node {
NodeType          const unsigned short ELEMENT_NODE = 1;
                  ...
                  const unsigned short NOTATION_NODE = 12;
readonly attribute DOMString      nodeName;
         attribute DOMString      nodeValue;
readonly attribute unsigned short nodeType;
// Navigation
readonly attribute Node     parentNode;
readonly attribute NodeList childNodes;
readonly attribute Node     firstChild;
readonly attribute Node     lastChild;
readonly attribute Node     previousSibling;
readonly attribute Node     nextSibling;
readonly attribute NamedNodeMap   attributes;
readonly attribute Document       ownerDocument;
// Methods
Node insertBefore(in Node newChild, in Node refChild) raises(DOMException);
Node replaceChild(in Node newChild, in Node oldChild) raises(DOMException);
Node removeChild(in Node oldChild) raises(DOMException);
Node appendChild(in Node newChild) raises(DOMException);
boolean hasChildNodes();
Node cloneNode(in boolean deep);};
```

# Concrete Classes and Their Relationships

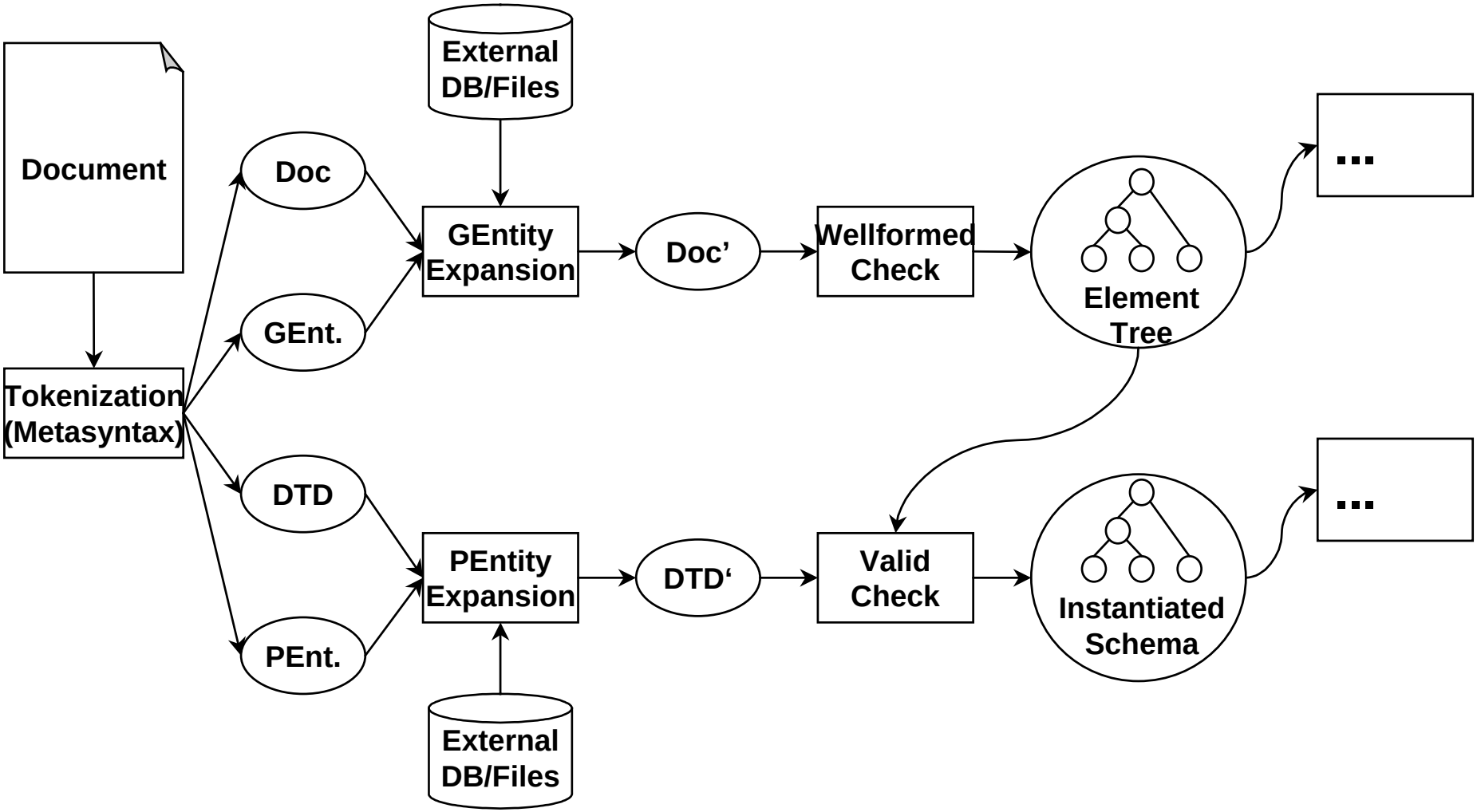
| **Node type** | **Possible sons** |
|---|---|
| Document | Element (at most one), ProcessingInstruction, Comment, DocumentType |
| DocumentFragment, Element, Entity, EntityReference | Element, ProcessingInstruction, Comment, Text, CDATASection, EntityReference |
| Attr | Text, EntityReference |
| DocumentType, ProcessingInstruction, Comment, Text, CDATASection, Notation | none |

# Processing of XML Documents

- XML Processor (XML engine): supports the access to the structure and content of XML documents
- XML Application: Software that uses the services provided by the XML processor



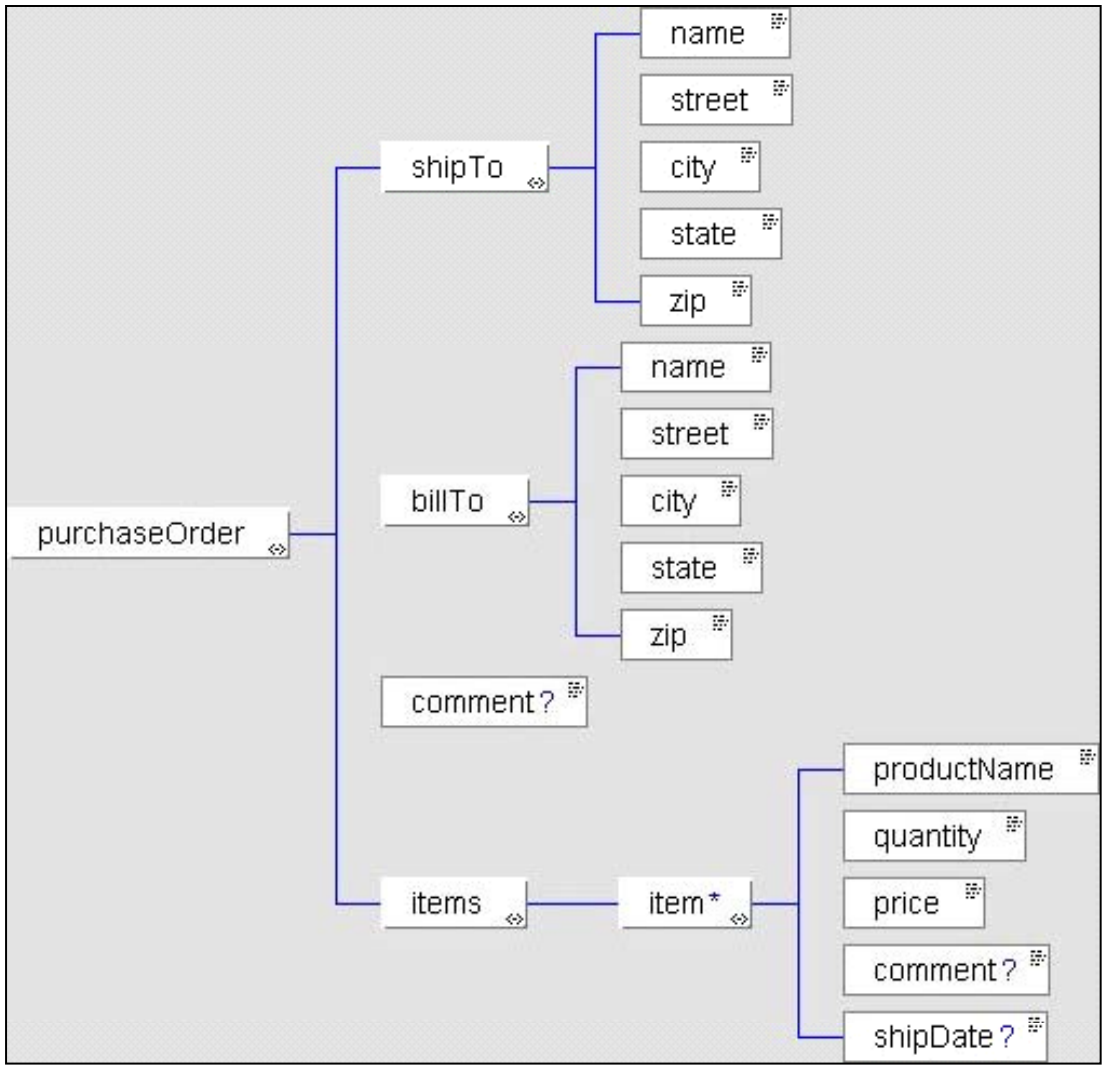
# Parsing of XML Documents (XML processor)



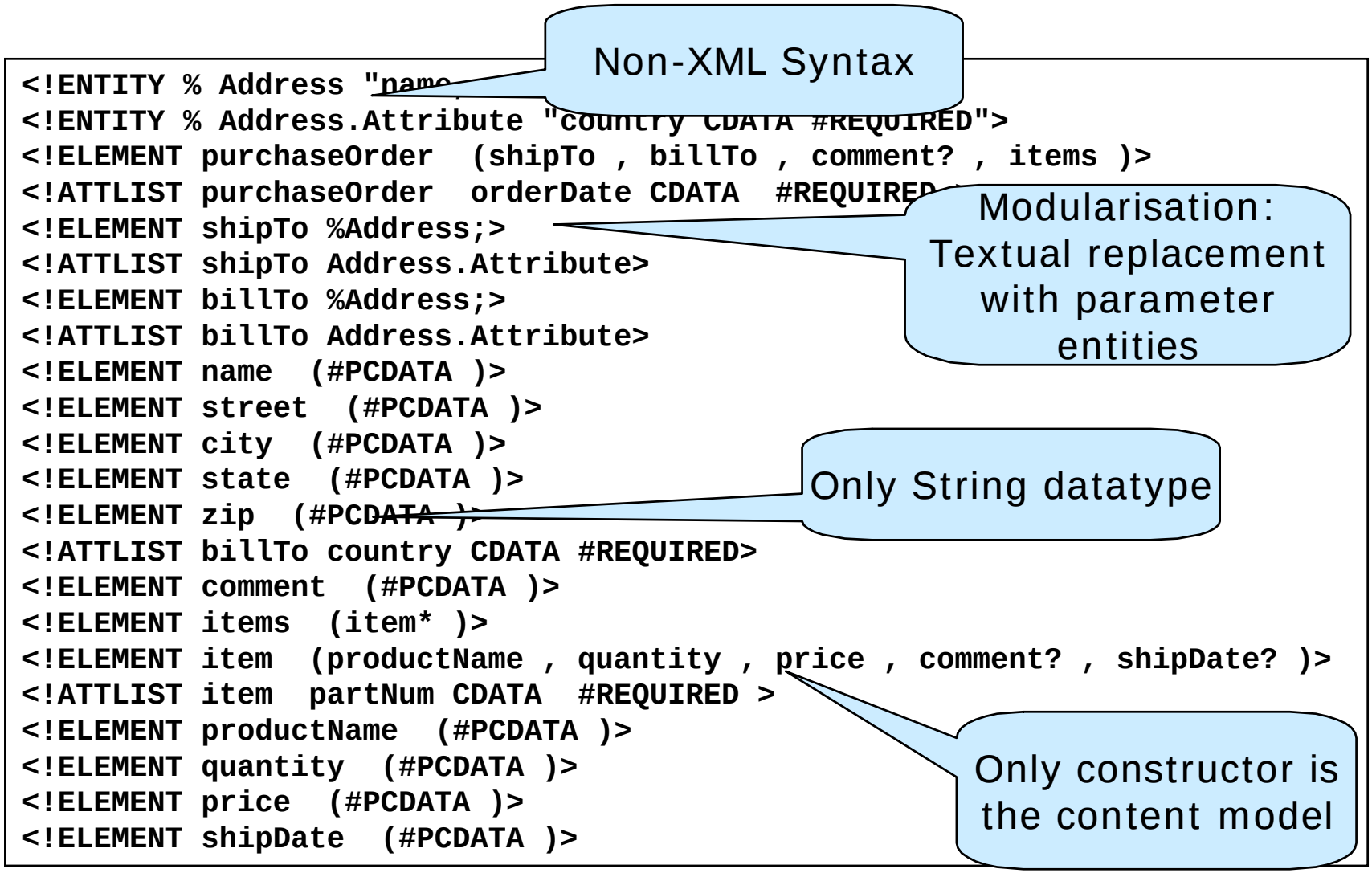
# Example: Use of DOM in JavaScript

```
<HTML><HEAD>
<TITLE>Generic Page for DOM tests</TITLE>
<SCRIPT>
window.onload = myLoad;
function myLoad()
{
if(go)
{show("If you are reading this, your DOM is working!");}}

function show(str)
{
resultText = document.createTextNode(str); // create a text object
resultBR = document.createElement("BR"); // create an element with tag BR
resultElement = document.body; // access the BODY element of the document
resultElement.appendChild(resultText); // insert the element as child
resultElement.appendChild(resultBR); // insert the BR element
}
</SCRIPT>
</HEAD>
<BODY>

</BODY></HTML>
```

# 3.2 XML Schema - Motivation



# Limitations of XML DTDs



# XML Schema Overview

- Unifies a typical object-oriented modeling paradigm with the DTD constructs in an XML syntax

- Main concepts
  - Simple types: rich set of basic types, user-definable simple types
  - Complex types: extend the content model of DTDs
  - Anonymous complex types
  - Choice and sequence construct (instead of | and , in DTDs)
  - Explicit cardinality constraints (instead of ?, + and * in DTDs)
  - Inheritance mechanisms by extensions and restriction
  - Integrity constraints (uniqueness constraints)

# Example Document – Sequence Constructor

- XML Document

```
 <USAddress country="US">
        <name>Alice Smith</name>
        <street>123 Maple Street</street>
        <city>Mill Valley</city>
        <state>CA</state>
        <zip>90952</zip>
</USAddress >
```

- DTD

```
<!ELEMENT USAdress  (name , street , city , state, zip )>
<!ATTLIST USAdress country CDATA  #FIXED >
<!ELEMENT name  #PCDATA> etc.
```

- XML Schema

```
<xsd:complexType name="USAddress">
        <xsd:sequence>
                <xsd:element name="name" type="xsd:string"/>
                <xsd:element name="street" type="xsd:string"/>
                <xsd:element name="city" type="xsd:string"/>
                <xsd:element name="state" type="xsd:string"/>
                <xsd:element name="zip" type="xsd:decimal"/>
        </xsd:sequence>
        <xsd:attribute name="country" type="xsd:NMTOKEN"
                        use="fixed" value="US"/>
</xsd:complexType>
```

# XML Schema – DTD Differences

```
<xsd:complexType name="PurchaseOrderType">
 <xsd:sequence>
  <xsd:element name="shipTo" type="Address"/>
  <xsd:element name="billTo" type="Address"/>
  <xsd:element ref="comment" minOccurs="0" />
  <xsd:element name="items"  type="Items"/>
  <xsd:attribute name="orderDate" type="xsd:date"/>
 </xsd:sequence>
</xsd:complexType>
<xsd:element name="comment" type="xsd:string"/>
```

**versus**

```
<!ELEMENT purchaseOrder  (shipTo , billTo , comment? , items )>
<!ATTLIST purchaseOrder  orderDate CDATA  #REQUIRED >
<!ELEMENT shipTo %Address;>
<!ATTLIST shipTo ...>
<!ELEMENT comment  (#PCDATA )>
```

# Anonymous Types and User-Defined Simple Types

```
<xsd:complexType name="Items">
<xsd:sequence>
    <xsd:element name="item" minOccurs="0" maxOccurs="unbounded">
    <xsd:complexType>
        <xsd:sequence>
        <xsd:element name="productName" type="xsd:string"/>
        <xsd:element name="quantity">
            <xsd:simpleType>
            <xsd:restriction base="xsd:positiveInteger">
            <xsd:maxExclusive value="100"/>
            </xsd:restriction>
            </xsd:simpleType>
        </xsd:element>
        <xsd:element name="USPrice" type="xsd:decimal"/>
        <xsd:element ref="comment" minOccurs="0"/>
        <xsd:element name="shipDate" type="xsd:date"
                                     minOccurs="0"/>
        </xsd:sequence>
    <xsd:attribute name="partNum" type="SKU"/>
    </xsd:complexType>
    </xsd:element>
</xsd:sequence>
</xsd:complexType>
```

Anonymous complex type

User-defined simple type

# Model Groups and Choice

```
<xsd:complexType name="PurchaseOrderType">
  <xsd:sequence>
        <xsd:choice>
              <xsd:group ref="shipAndBill"/>
              <xsd:element name="singleAddress" type="Address"/>
        </xsd:choice>
        <xsd:element ref="comment" minOccurs="0" />
        <xsd:element name="items"  type="Items"/>
  </xsd:sequence>
  <xsd:attribute name="orderDate" type="xsd:date"/>
</xsd:complexType>
<xsd:element name="comment" type="xsd:string"/>

<xsd:group name="shipAndBill">
  <xsd:sequence>
        <xsd:element name="shipTo" type="Address"/>
        <xsd:element name="billTo" type="Address"/>
  </xsd:sequence>
</xsd:group>
```

# All

```
<xsd:complexType name="PurchaseOrderType">
  <xsd:all>
        <xsd:element name="shipTo" type="Address"/>
        <xsd:element name="billTo" type="Address"/>
        <xsd:element ref="comment" minOccurs="0" />
        <xsd:element name="items"  type="Items"/>
  </xsd:all>
  <xsd:attribute name="orderDate" type="xsd:date"/>
</xsd:complexType>
<xsd:element name="comment" type="xsd:string"/>
```

# XML Schema: Inheritance

```
<complexType name="Address">
 <element name="name"   type="string"/>
 <element name="street" type="string"/>
 <element name="city"   type="string"/>
</complexType>

<complexType name="US-Address"
              base="Address"  derivedBy="extension">
 <element name="state" type="US-State"/>
 <element name="zip"   type="positiveInteger"/>
</complexType>

<complexType name="UK-Address"
              base="Address" derivedBy="extension">
 <element   name="postcode" type="UK-Postcode"/>
 <attribute name="export-code" type="positiveInteger"
            use="fixed" value="1"/>
</complexType>
<!-- etc -->
```

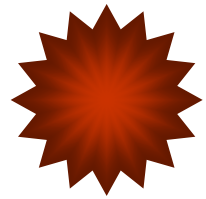
# Using Derived Types

```
<shipTo exportCode="1" xsi:type="ipo:UKAddress">
     <name>Helen Zoe</name>
     <street>47 Eden Street</street>
     <city>Cambridge</city>
     <postcode>CB1 1JR</postcode>
</shipTo>

<billTo xsi:type="ipo:USAddress">
     <name>Robert Smith</name>
     <street>8 Oak Avenue</street>
     <city>Old Town</city>
     <state>PA</state>
     <zip>95819</zip>
</billTo>
```

# Derived Types by Restriction

```
<complexType name="Items">
     <sequence>
     <element name="item" minOccurs="0" maxOccurs="unbounded">
     <complexType>
             <sequence>
             <element name="productName" type="string"/>
             <element name="quantity">
             </sequence>
     </complexType>
     </element>
     </sequence>
</complexType>


<complexType name="ConfirmedItems">
<restriction base="ipo:Items">
     <sequence>
     <element name="item" minOccurs="1" maxOccurs="unbounded">
     <complexType> … same as before … </complexType>
     </sequence>
</restriction>
</complexType>
```

# XML Schema: Integrity Constraints



```
<element name="purchaseReport">
 <complexType>
  <element name="regions"
           type="RegionsType"/>
  <element name="parts"
           type="PartsType"/>
  <attribute name="period"
             type="timeDuration"/>
  <attribute name="periodEnding"
             type="date"/>
 </complexType>
 <unique>
  <selector>regions/zip</selector>
  <field>@code</field>
 </unique>
 <key name="pNumKey">
  <selector>parts/part</selector>
  <field>@number</field>
 </key>
 <keyref refer="pNumKey">
  <selector>regions/zip/part</selector>
  <field>@number</field>
 </keyref>
</element>
```

**XPath**

```
<complexType name="RegionsType">
 <element name= "zip" minOccurs="1"
          maxOccurs="unbounded">
  <complexType>
   <element name="part">
    <complexType content="empty">
     <attribute name="number" type="Sku"/>
      <attribute name="quantity"
                 type="positiveInteger"/>
    </complexType>
   </element>
   <attribute name="code"
              type="positiveInteger"/>
  </complexType>
 </element>
</complexType>

<complexType name="PartsType>
 <element name="part" minOccurs="1"
          maxOccurs="unbounded">
  <complexType content="textOnly">
    <attribute name="number" type="Sku"/>
  </complexType>
 </element>
</complexType>
```

# Integrity Constraints at Instance Level

```
<purchaseReport xmlns="http://www.example.com/Report"
      period="P3M" periodEnding="1999-12-31">

<regions>
      <zip code="95819">
      <part number="872-AA" quantity="1"/>
      <part number="926-AA" quantity="1"/>
      <part number="833-AA" quantity="1"/>
      <part number="455-BX" quantity="1"/>
      </zip>

      <zip code="63143">
      <part number="455-BX" quantity="4"/>
      </zip>
</regions>

<parts>
      <part number="872-AA">Lawnmower</part>
      <part number="926-AA">Baby Monitor</part>
      <part number="833-AA">Lapis Necklace</part>
      <part number="455-BX">Sturdy Shelves</part>
</parts>

</purchaseReport>
```

# References

- Books
  - Box, Skonnard, Lam: Essential XML, Addison Wesley, 2000.
  - Goldfarb, Prescod: The XML Handbook, Prentice Hall, 2000.
  - St. Laurent, Cerami, Building XML Applications, McGraw Hill, 1999.
  - Abiteboul, Bunemann, Suciu: Data on the Web: From Relations to Semistructured Data and XML, Morgan Kaufman, 2000.
  - Professional XML, WROX Press, 2000.
- Web Sites
  - www.w3c.org
    - www.w3.org/XML/
    - www.w3.org/DOM
    - www.w3.org/TR/xpath
    - www.w3.org/Style/XSL/
    - www.w3.org/XML/Linking.0
    - www.w3.org/XML/Schema
    - www.w3.org/TR/xquery/
    - www.w3.org/TR/xmlquery-use-cases