Net-Based Applications

Chapter 6: XML Basics

Holger Schwarz
Universität Stuttgart

Winter Term 2016/2017

Overview

- Motivation
- Fundamentals
- Document Type Definition (DTD)
- XML Namespaces
- Examples
  - XHTML
  - ebXML

What is a Markup Language?

A Markup Language allows to represent text as well as details about the structure and appearance of the text.

- Some history: The term goes back to the “marking up” of paper manuscripts with revision instructions and typesetting instructions

Purpose of Markup

- In general markup allows to separate text/data, metadata and presentation
- In documents
  - describe structure of the document
  - describe appearance of text
  - e.g. HTML

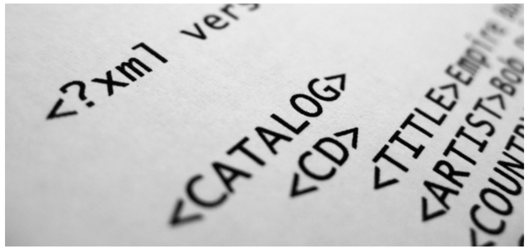
tags

```
<html>
<head>
   <title> A Simple HTML Document </title>
</head>
<body>
   <h1>Introduction</h1>
   <p>This is a very simple HTML document</p>
   <p>It only has <b>two</b> paragraphs</p>
</body>
</html>
```

- For data representation
  - describe structure of data
  - provide metadata for data elements

```
<catalog>
   <book id="bk101">
      <author>Gambardella, Matthew</author>
      <title>XML Developer's Guide</title>
      <genre>Computer</genre>
      <price>44.95</price>
      <publish_date>2000-10-01</publish_date>
   </book>
</catalog>
```

XML



- Extensible Markup Language (XML) defined by WWW Consortium (W3C)
- Focus on data and metadata: “Tell me what it is, not what it looks like.”
- Users may specify presentational aspects by using separate languages
  - eXtensible Stylesheet Language (XSL) – a family of recommendations for defining XML document transformation and presentation
  - Cascading Style Sheets (CSS) - a simple mechanism for adding style (e.g. fonts, colors, spacing) to Web documents
- Important properties:
  - Self-descriptive: Tags can be used as metadata
  - Extensible: XML does not provide a fixed set of pre-defined tags, i.e., users may add new tags (compare to HTML)

XML Origin

- Derived from SGML (Standard Generalized Markup Language)
  - By construction, XML documents are conforming SGML documents.
  - XML is simpler to use than SGML.
  - ISO 8879:1986(E). Information processing — Text and Office Systems — Standard Generalized Markup Language (SGML). First edition — 1986-10-15.
  - DIN EN 28879:1991

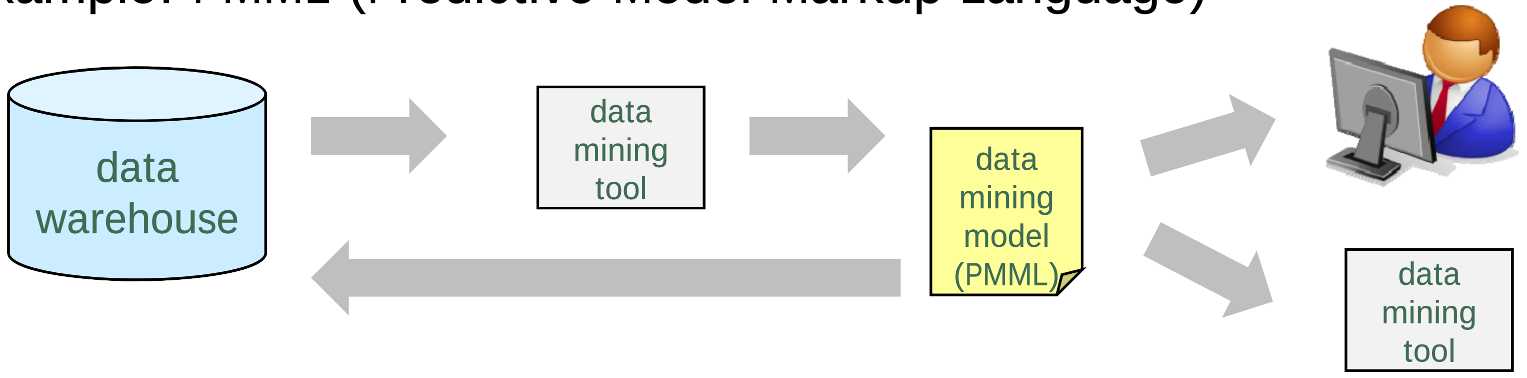
$$\begin{tikzcd}
 & \text{SGML} \arrow[dl, "\text{application}"'] \arrow[dr, "\text{subset}"] & \\
\text{HTML} & & \text{XML} \arrow[d, "\text{application}"] \\
 & & \text{XHTML}
\end{tikzcd}$$

XML Standardization

- Defined by the WWW Consortium (W3C)
  - XML 1.0:
    Extensible Markup Language (XML), Version 1.0 (Fifth Edition)
    W3C Recommendation 26 November 2008
  - XML 1.1:
    Extensible Markup Language (XML) 1.1 (Second Edition)
    W3C Recommendation 16 August 2006
    - updates XML so that it no longer depends on the specific Unicode version
    - developers are encouraged to create or generate XML 1.0 documents if you do not need the new features.

XML Usage: XML Markup Languages

- XML-based standards define what valid elements are:
  - XML type specification languages to specify the syntax
  - Textual descriptions of the semantics
- application-specific data formats for potentially complex structures
  - ebXML: electronic business Extensible Markup Language
  - GML: Geographic Markup Language
  - CML: Chemical Markup Language
  - MathML: mathematical expressions
  - many more!
- Example: PMML (Predictive Model Markup Language)



XML Usage: Data Exchange

- XML is heavily used for data exchange.
- Data interchange is critical in today’s networked world.
- XML is a key technology for interoperation. E.g.
  - SOAP (Simple Object Access Protocol)
  - WSDL (Web Service Definition Language)
  - WS-Business Process Execution Language (BPEL)
  - ...

XML Usage: XML Storage and Processing

- Native XML database management systems:
  - Database built specifically for storing XML data, supporting DOM model and declarative querying, e.g., TAMINO (Software AG)
  - XML as document format!
- Extended relational database management systems:
  - add storage and processing capabilities to RDBMS, e.g., IBM DB2
- A wide variety of tools is available for parsing, browsing and querying XML documents/data.

Overview

- Motivation
- Fundamentals
- Document Type Definition (DTD)
- XML Namespaces
- Examples
  - XHTML
  - ebXML

XML Syntax

- XML Declaration (Prologue):
  - Defines the XML version and the character encoding used in the document.
  - This declaration is optional in XML 1.0.

```
<?xml version="1.0" encoding="ISO-8859-1"?>

<bank>
      <customer>
         <customer-name>Hayes</customer-name>
         <customer-street>Main</customer-street>
         <customer-city>Harrison</customer-city>
         <account>
            <!– first account of customer -->
            <account-number>A-102</account-number>
            <branch-name>Perryridge</branch-name>
            <balance>400</balance>
         </account>
         …
</customer>
     …
</bank>
```

prologue

root element

comment

Elements

- Tag:
  - Label for a section of data.
  - Tag name is an XML name
    - Consists of alphanumeric characters and _ - .
    - Starts with a letter or _

```
<tag-name>
   content
</tag-name>
```

- Element:
  - Section beginning with <tag-name> and ending with matching </tag-name>.
  - Content of an element: text and/or child element(s).
  - Mixture of text and child elements is useful for document markup, but discouraged for data representation.

```
<account>
This account is seldom used.
   <account-number>A-102</account-number>
   <branch-name>Perryridge</branch-name>
   <balance>400</balance>
</account>
```

Empty Elements and CDATA Sections

- Elements without subelements or text content can be abbreviated by ending the start tag with a `/>` and omitting the end tag.

```
<account></account>
```
=
```
<account/>
```

- To store string data that may contain tags, without the tags being interpreted as subelements, use CDATA as below:

```
<![CDATA[<account> … </account>]]>
```

  - Here, <account> and </account> are treated just as strings
  - CDATA sections can contain any string except]]>

Attributes

- Attributes:
  - Attributes can be used to describe elements.
  - Attributes are specified by name="value" pairs inside the starting tag of an element.
  - Each element may have several attributes.
  - Attribute names must be unique within the element.
- Example:

```
…
<account acct-type = “checking” monthly-fee=“5” >
<account-number>A-102</account-number>
<branch-name>Perryridge</branch-name>
<balance>400</balance>
</account>
…
```

Attributes vs. Subelements

- In the context of documents, attributes are part of markup, while subelement contents are part of the basic document contents.
- In the context of data representation, the difference is unclear and may be confusing.
- Same information may be represented in two ways:

```
<account>
    <account-number>A-101</account-number>
    …
</account>
```

```
<account  account-number = “A-101”>
…
</account>
```

Attributes vs. Subelements

- attributes cannot contain multiple values (child elements can)
- attributes are not easily expandable (for future changes)
- attributes cannot describe structures (child elements can)
- attributes are more difficult to manipulate by program code
- attribute values are not easy to test against a Document Type Definition (DTD) - which is used to define the legal elements of an XML document

- Suggestion: use attributes for identifiers of elements, and use subelements for content

Well-formed Documents

An XML document is called well-formed if:

- It has a single top-level element.
- It is properly nested,
  i.e., every start tag must have a unique matching end tag, that is in the context of the same parent element.

- Example: Is this document well-formed?

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<account>
<account-number>A-102</account-number>
<branch-name>Perryridge</branch-name>
<balance>400
</account>
<customer>
<name>Joe
</customer>
</name>
```

XML Validation

The process to confirm that an XML document is

- well-formed and
- follows a defined structure

is called XML validation.

- How do we describe the expected structure of an XML document?
  - DTD
  - XML schema

Overview

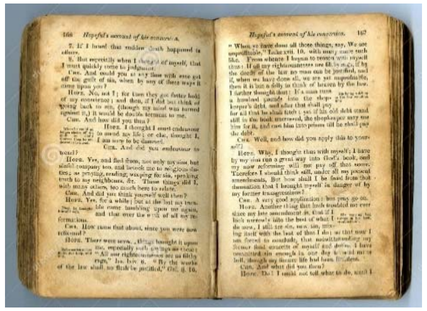
- Motivation
- Fundamentals
- Document Type Definition (DTD)
- XML Namespaces
- Examples
  - XHTML
  - ebXML

Describing XML Data

- XML is extremely flexible.
- Most applications can only handle specific XML documents.
- Metadata and schemas constrain:
  - what information can be stored
  - information structure
  - data types of stored values
- Metadata is very important for data exchange:
  - Guarantees automatic and correct data interpretation.
- XML documents are not required to have an associated schema.

How Structured is Your Data?

- We distinguish between unstructured, semi-structured and structured data.
- No clear separation of these categories.

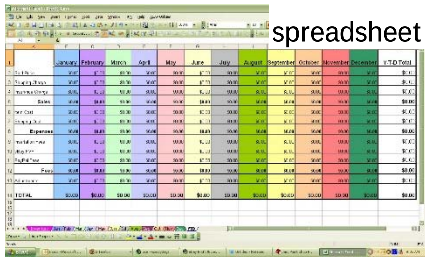


JSON

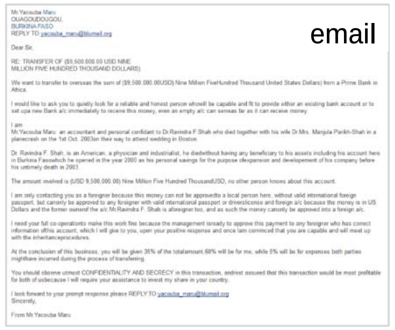
```
{
  "firstName": “Paul",
  "lastName": “Adam",
  "age": 45,
  "address":
  {
    "streetAddress": "22 2nd Street",
    "city": "New York",
    "state": "NY",
    "postalCode": "10021"
  }
}
```

Sid	Name	Subject	Age	Fees
105	Amit	History	21	30k
106	Babita	Science	23	33k
107	Neetu	Math	22	35k
108	Mamta	Commerce	20	32k
111	Pawan	Management	24	36k

spreadsheet



email



Mr.Yacouba Maru
OUAGOUDOUGOU,
BURKINA FASO
REPLY TO: yacouba_maru@blumail.org

Dear Sir,

RE: TRANSFER OF ($9,500,000.00 USD NINE MILLION FIVE HUNDRED THOUSAND DOLLARS)

We want to transfer to overseas the sum of ($9,500,000.00USD) Nine Million FiveHundred Thousand United States Dollars) from a Prime Bank in Africa.

I would like to ask you to quietly look for a reliable and honest person whowill be capable and fit to provide either an existing bank account or to set upa new Bank a/c immediately to receive this money, even an empty a/c can serveas far as it can receive money.

I am
Mr.Yacouba Maru an accountant and personal confidant to Dr.Ravindra F.Shah who died together with his wife Dr.Mrs. Manjula Parikh-Shah in a planecrash on the 1st Oct. 2003on their way to attend wedding in Boston.

Dr. Ravindra F. Shah, is an American, a physician and industrialist, he diedwithout having any beneficiary to his assets including his account here in Burkina Fasowhich he opened in the year 2000 as his personal savings for the purpose ofexpansion and developement of his company before his untimely death in 2003.

The amount involved is (USD 9,500,000.00) Nine Million Five Hundred ThousandUSD, no other person knows about this account.

I am only contacting you as a foreigner because this money can not be approvedto a local person here, without valid international foreign passport, but canonly be approved to any foreigner with valid international passport or driverslicense and foreign a/c because the money is in US Dollars and the former ownerof the a/c Mr.Ravindra F. Shah is aforeigner too, and as such the money canonly be approved into a foreign a/c.

I need your full co-operationto make this work fine because the management isready to approve this payment to any foreigner who has correct information ofthis account, which I will give to you, upon your positive response and once Iam convinced that you are capable and will meet up with the inheritanceprocedures.

At the conclusion of this business, you will be given 35% of the totalamount,60% will be for me, while 5% will be for expenses both parties mighthave incurred during the process of transferring.

You should observe utmost CONFIDENTIALITY AND SECRECY in this transaction, andrest assured that this transaction would be most profitable for both of usbecause I will require your assistance to invest my share in your country.

I look forward to your prompt response please REPLY TO:yacouba_maru@blumail.org
Sincerely,

From Mr.Yacouba Maru

XML

```
<bank>
  <customer>
    <customer-name>Hayes</customer-name>
    <customer-street>Main</customer-street>
    <customer-city>Harrison</customer-city>
  </customer>
  <customer>
    <customer-name>Miller</customer-name>
  </customer>
</bank>
```

unstructured semi-structured structured

Element Declaration

- No content
  - <!ELEMENT element-name EMPTY>
- Unrestricted content
  - <!ELEMENT element-name ANY>
  - Child elements must be declared.
- Simple content
  - Unstructured character strings
  - <!ELEMENT element-name (#PCDATA)>

```
<!ELEMENT customer-id EMPTY>
<!ELEMENT customer-name (#PCDATA)>
<!ELEMENT customer-street (#PCDATA)>
<!ELEMENT customer-city (#PCDATA)>
```

Element Content (Complex Content)

- Sequence of child elements
- Typically used for data
- <!ELEMENT element-name (list)quantifier>
  - Sequence list: particle1, particle2, particle3, ...
  - Choice list: particle1 | particle2 | particle3 | ...
  - Particle
    - element-name quantifier
    - (list)quantifier
  - Quantifier (optional): * + ?

```
<!ELEMENT bank (account, customer, depositor)*>
<!ELEMENT account (account-number, branch-name, balance)>
<!ELEMENT customer (customer-name, customer-street, customer-city)>
<!ELEMENT depositor (customer-name, account-number)>
<!ELEMENT account-number (#PCDATA)>
<!ELEMENT branch-name (#PCDATA)>
<!ELEMENT balance (#PCDATA)>
```

Mixed Content

- Sequence of text and child elements.
- Typically used for text markup.
- <!ELEMENT element-name (#PCDATA | element-name1 | element-name2 | ...)*>
- No other list type or quantifier allowed.
- #PCDATA must be the first list element.

```
<!ELEMENT text (#PCDATA | b | i)*>
<!ELEMENT b (#PCDATA)>
<!ELEMENT i (#PCDATA)>
```

```
<text>A text using<i>italics</i> and <b>bold face</b>.</text>
```

External DOCTYPE Declaration

- DOCTYPE declaration may refer to a separate file including the DTD.

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE bank SYSTEM "bank1.dtd">
<bank>
  <customer>
    <customer-name>Joe</customer-name>
    <customer-street>Monroe</customer-street>
    <customer-city>Madison</customer-city>
  </customer>
  <account>
    <account-number>A-401<
    <branch-name>Downtown<
    <balance>500</balance>
  </account>
  <customer>
    <customer-name>Mary</c
    <customer-street>Erin<
    <customer-city>Newark<
  </customer>
</bank>
```

bank1.dtd

```
<!ELEMENT bank (account | customer | depositor)+>
<!ELEMENT account (account-number,branch-name,balance)>
<!ELEMENT customer (customer-name,customer-street,customer-city)>
<!ELEMENT depositor (customer-name,account-number)>
<!ELEMENT account-number (#PCDATA)>
<!ELEMENT branch-name (#PCDATA)>
<!ELEMENT balance (#PCDATA)>
<!ELEMENT customer-name (#PCDATA)>
<!ELEMENT customer-street (#PCDATA)>
<!ELEMENT customer-city (#PCDATA)>
```

Internal DOCTYPE Declaration

- DOCTYPE can be declared inline in an XML document.

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE bank [
    <!ELEMENT bank (account | customer | depositor)+>
    <!ELEMENT account (account-number,branch-name,balance)>
    <!ELEMENT customer (customer-name,customer-street,customer-city)>
    <!ELEMENT depositor (customer-name,account-number)>
    <!ELEMENT account-number (#PCDATA)>
    <!ELEMENT branch-name (#PCDATA)>
    <!ELEMENT balance (#PCDATA)>
    <!ELEMENT customer-name (#PCDATA)>
    <!ELEMENT customer-street (#PCDATA)>
    <!ELEMENT customer-city (#PCDATA)>
] >
<bank>
    <customer>
        <customer-name>Joe</customer-name>
        <customer-street>Monroe</customer-street>
        <customer-city>Madison</customer-city>
    </customer>
...
</bank>
```

Attribute Declaration

- <!ATTLIST element-name attribute-name attribute-type default>
- One attribute declaration may declare multiple attributes for an element.
- Attribute type:
  - CDATA: character string
  - ID: identifier
  - IDREF (IDREFS): (list of) reference(s) to an ID
  - NMTOKEN (NMTOKENS): (list of) token(s)
    - alphanumeric characters and _ - . :
  - (val1 | val2 | val3 | ...): list of allowed attribute values
- Default declaration:
  - #REQUIRED: attribute must be present
  - #IMPLIED: attribute is optional
  - "value": attribute is optional with default value
  - #FIXED "value": if present, attribute must have the given value

```
<!ATTLIST account
 account-number   ID        #REQUIRED
 owners           IDREFS    #REQUIRED>
```

IDs and IDREFs

- An element can have at most one attribute of type ID.
- The ID attribute value of each element in an XML document must be distinct.
  - ID attribute (value) is an object identifier.
- An attribute of type IDREF must contain the ID value of an element in the same document.
- An attribute of type IDREFS contains a set of (0 or more) ID values.
  - Each ID value must contain the ID value of an element in the same document.
- IDs and IDREFs are untyped, unfortunately.

Extended Bank DTD

bank2.dtd

```
<!ELEMENT bank (account | customer | depositor)+>
<!ELEMENT account (branch-name,balance)>
<!ATTLIST account
           account-number ID       #REQUIRED
           owners     IDREFS       #REQUIRED>
<!ELEMENT customer (customer-name,customer-street,customer-city)>
<!ATTLIST customer
          customer-id     ID       #REQUIRED
          accounts        IDREFS   #REQUIRED>
<!ELEMENT depositor (customer-name,account-number)>
<!ELEMENT account-number (#PCDATA)>
<!ELEMENT branch-name (#PCDATA)>
<!ELEMENT balance (#PCDATA)>
<!ELEMENT customer-name (#PCDATA)>
<!ELEMENT customer-street (#PCDATA)>
<!ELEMENT customer-city (#PCDATA)>
```

- The owners attribute of an account may contain a reference to another account, which is meaningless.
- Owners attribute should ideally be constrained to refer to customer elements.

Extended Bank Document

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE bank SYSTEM "bank2.dtd">
<bank>
    <account account-number="A-401" owners="C100 C102">
        <branch-name>Downtown</branch-name>
        <balance>500</balance>
    </account>
    <account account-number="A-402" owners="C100">
        <branch-name>Main Station</branch-name>
        <balance>1000</balance>
    </account>
    <customer customer-id="C100" accounts="A-401">
        <customer-name>Joe</customer-name>
        <customer-street>Monroe</customer-street>
        <customer-city>Madison</customer-city>
    </customer>
    <customer customer-id="C102" accounts="A-401 A-402">
        <customer-name>Mary</customer-name>
        <customer-street>Erin</customer-street>
        <customer-city>Newark</customer-city>
    </customer>
</bank>
```

Could also refer
to an account number

Entity Declarations

- General parsed entities
  - Other entities: parameter, external, ...
- Counterpart in programming languages
  - Constants
  - Processor macros
- Five predefined entity references
- More can be defined in a DTD
- <!ENTITY entity-name "replacement text">
- Entities may contain (well formed) markup

entity ref.	character
&lt;	<
&gt;	>
&amp;	&
&quot;	"
&apos;	'

```
<!ENTITY joe '<customer customer-id="C100" accounts="A-401">
 <customer-name>Joe</customer-name>
 <customer-street>Monroe</customer-street>
 <customer-city>Madison</customer-city>
 </customer>'>
…
<bank> &joe; </bank>
```

Overview

- Motivation
- Fundamentals
- Document Type Definition (DTD)
- XML Namespaces
- Examples
  - XHTML
  - ebXML

XML Namespaces

- XML data is often exchanged between organizations.
- Same tag name may have different meanings in different organizations.
  - Student is different in German and American education system (research assistant vs. PHD student)
  - A book has a title, a person may have a title as well.
- Collisions of names:
  - when we exchange documents
  - when we want to combine data from different documents
- How to differentiate between different origins/meanings?
- Namespaces allow to construct universally unique names.

XML Namespaces

- An XML namespace is a collection of names, identified by a URI reference.

```
scheme:[//[user:password@]host[:port]][/]path[?query][#fragment]
```

- URI references which identify namespaces are considered identical when they are exactly the same character-for-character.
- Namespace name should be unique and persistent.

Namespace Binding

- A namespace is declared using a family of reserved names for special attributes (xmlns).
- Namespace declarations may be part of any element.
- Default namespace exists for all elements and attributes without prefix.
- Nested default namespaces overwrite previous namespace declarations.

declaring a default namespace

```
<bank xmlns=‘http://www.FirstBank.com’>
…
<branch>
<branchname>Downtown</branchname>
<branchcity>Brooklyn</branchcity>
</branch>
…
</bank>
```

named namespace

```
<bank xmlns:MS=‘http://www.MegaShop.com’>
…
<MS:branch>
<MS:branchname>Downtown</MS:branchname>
<MS:location>Queens</MS:location>
</MS:branch>
…
</bank>
```

Namespace Binding

- Expanded name = pair of namespace name and local name
  - often inconveniently long
- QName (Qualified name) is used instead
  - prefixed or unprefixed
  - Prefixed Name: Prefix ":" LocalName
    - Prefix = Name of a namespace provided as NCName
    - LocalName = Name within a namespace provided as NCName
  - Unprefixed Name: LocalName
    - Namespace Name is the default namespace
- NCName (Non-Colonized Name): A name without a ":"
  - Prefix and LocalNames must be NCNames
  - This is important to identify them in a QName

Additional Attributes

- Attribute xml:lang
  - useful to identify the natural or formal language in which the content is written

```
<p xml:lang="en">The quick brown fox jumps over the lazy dog.</p>
<p xml:lang="en-GB">What colour is it?</p>
<p xml:lang="en-US">What color is it?</p>
<sp who="Faust" desc='leise' xml:lang="de">
    <l>Habe nun, ach! Philosophie,</l>
    <l>Juristerei, und Medizin</l>
    <l>und leider auch Theologie</l>
    <l>durchaus studiert mit heißem Bemüh'n.</l>
</sp>
```

- Attribute xml:space
  - use spaces, tabs, and blank lines to set apart the markup for greater readability.
  - typically not intended for inclusion in the delivered version of the document
  - "significant" white spaces should be preserved (poetry, source code)
    ... xml:space="preserve" ...

Overview

- Motivation
- Fundamentals
- Document Type Definition (DTD)
- XML Namespaces
- Examples
  - XHTML
  - ebXML

XHTML

- XML-compliant version of HTML 4
  - XHTML documents are well-formed XML
  - XHTML is basically a subset of HTML 4
- Design goals:
  - Can be processed by HTML 4 user agents.
  - Support separation of content and presentation (in DTD Strict).
    - Attributes and elements like color or font removed.
    - Use of Cascading Style Sheets (CSS) for controlling formatting.
- Advantage: Standard XML tools available
  - Create Web pages using an XML editor.
  - Build browsers and search engines based on XML parsers.

XHTML

- Documents must be well-formed XML
  - End-tags required for non-empty elements
    - Illegal: <p>A paragraph<p>Next paragraph
    - Correct: <p>A paragraph</p><p>Next paragraph</p>
  - Empty elements must be terminated
    - Illegal: <hr>
    - Correct: <hr/>
  - Attribute values must be quoted
    - Illegal: <td rowspan=3>
    - Correct: <td rowspan="3">
  - Attribute minimization not allowed
    - Illegal: <dl compact>
    - Correct: <dl compact="compact">
- XML is case-sensitive
  - Element names and attribute names must be in lower case.

XHTML Document

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
          "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="en" lang="en">
<head>
<meta http-equiv="Content-type" content="text/html; charset=UTF-8" />
<title>XHTML Example</title>
</head>
<body>
<h1>A Section</h1>
<p>The first paragraph.</p>
<ul>
<li>list item</li>
</ul>
<hr />
</body>
</html>
```

DTDs for XHTML 1.0

- XHTML 1.0 Strict
  - Supports clean markup, free of presentational clutter.
  - For user agents (browsers) supporting Cascading Style Sheets.
- XHTML 1.0 Transitional
  - Adds some elements and attributes to control formatting.
  - For user agents without style sheet support.
- XHTML 1.0 Frameset
  - Adds support for HTML Frames to partition the browser window into two or more frames.

Modularization in XHTML 1.1

- Increased interoperability by modularization.
- Decomposition of XHTML 1.0 into a collection of abstract modules that provide specific types of functionality (elements and attributes).
- User agents do not have to support all modules.
  - Light-weight XHTML browsers possible (PDAs, ...).
- Data providers may define additional modules.
- Module-based XHTML document types:
  - XHTML Basic 1.1 (small subset of XHTML 1.1)
  - XHTML 1.1 (basically equivalent to XHTML 1.0 Strict)

ebXML

- XML specifications for:
  - Business Process Specifications
  - Partner Profile and Agreements
  - Registry and Repository
  - Core Components
  - Messaging Service

DTD for one part of the process specification

```
<!ELEMENT BinaryCollaboration (Documentation*, InitiatingRole,
RespondingRole, (Documentation | Start | Transition | Success |
Failure | BusinessTransactionActivity | CollaborationActivity
| Fork | Join)*)>
<!ATTLIST BinaryCollaboration
name                CDATA    #REQUIRED
nameID              ID       #IMPLIED
pattern             CDATA    #IMPLIED
beginsWhen          CDATA    #IMPLIED
endsWhen                     CDATA    #IMPLIED
precondition        CDATA    #IMPLIED
postCondition       CDATA    #IMPLIED
timeToPerform       CDATA    #IMPLIED
>
```

ebXML

```
<BusinessPartnerRole name="Retailer">
    <Performs respondingRole="provider"/>
    <Performs respondingRole="seller"/>
    <Performs initiatingRole="Creditor"/>
    <Performs initiatingRole="buyer"/>
    <Performs initiatingRole="Payee"/>
    <Performs respondingRole="Payor"/>
    <Performs initiatingRole="requestor"/>
    <Transition fromBusinessState="Create Order" toBusinessState="Check Credit"/>
    <Transition fromBusinessState="Check Credit" toBusinessState="Create Order"/>
</BusinessPartnerRole>
...
<BinaryCollaboration name="Credit Payment">
    <InitiatingRole name="payee"/>
    <RespondingRole name="payor"/>
    <BusinessTransactionActivity name="Process Credit Payment"
            businessTransaction="Process Credit Payment"
            fromAuthorizedRole="payee"
            toAuthorizedRole="payor"/>
</BinaryCollaboration>
...
```

Summary

- XML: standardized (meta) markup language to represent text and data structures
  - document-centered: mixture of structured data (author, meta data) and unstructured data (headings, text body, ...)
  - data-centered: strong data structure (e.g. from databases)
  - well-formed XML: compliant to the standard XML structure
- XML document consists of:
  - document declarations (<? ... ?>)
  - elements (opening and closing tags)
  - attributes (inside opening tags)
- Namespace to avoid ambiguity
- One standard to define an XML language: DTD (document type definition)
  - valid XML: compliant to the DTD (or XML Schema, see next chapter)
- Examples: XHTML, ebXML, MathML, ...

Literature & Information

- World Wide Web Consortium: Extensible Markup Language (XML), Version 1.0 (Fifth Edition). W3C Recommendation, 26 November 2008. http://www.w3.org/TR/2008/REC-xml-20081126/
- XML 1.1 (Second Edition), W3C Recommendation, 16 August 2006. http://www.w3.org/TR/2006/REC-xml11-20060816/
- World Wide Web Consortium: Namespaces in XML 1.1 (Second Edition). W3C Recommendation, 16 August 2006. http://www.w3.org/TR/2006/REC-xml-names11-20060816/
- World Wide Web Consortium: XHTML™ 1.0 The Extensible HyperText Markup Language (Second Edition) 2002. http://www.w3.org/TR/xhtml1/