

```
<?xml version="1.0"?>
<quiz>
  <qanda seq="1">
    <question>
      Who was the forty-second
      president of the U.S.A.?
    </question>
    <answer>
      William Jefferson Clinton
    </answer>
  </qanda>
  <!-- Note: We need to add
  more questions later.-->
</quiz>
```

**XML**

# Structured Web Documents in XML (a)

Adapted from slides from Grigoris  
Antoniou and Frank van Harmelen

# Outline

## **(1) Introduction**

(2) XML details

(3) Structuring

- DTDs
- XML Schema

(4) Namespaces

(5) Accessing, querying XML documents: XPath

(6) Transformations: XSLT

# Role of XML in the Semantic Web

- The Semantic Web involves ideas and languages at a fairly abstract level, e.g.: for defining ontologies, publishing data using them
- XML is a
  - Source of many key SW concepts & technology bits;
  - Potential alternative for sharing data that newer schemes must improve on; and
  - Common serialization for SW data

# To paraphrase Jamie Zawinski

Some people, when confronted with a problem, think, "I know, I'll use XML."

Now they have two problems.

"Some people, when confronted with a problem, think "I know, I'll use regular expressions." Now they have two problems."

-- Wikiquote

# History

- XML's roots are in SGML
  - [Standard Generalized Markup Language](#)
  - A [\*metalanguage\*](#) for defining document markup languages
  - Extensible, but complicated, verbose, hard to parse, ...
- HTML was defines using SGML, ~1990 by TBL
  - A markup language, not a markup *metalanguage*
- XML proposal to W3C in July 1996
  - Simplified SGML to greatly expand power and flexibility of Web
- Evolving series of W3C recommendations
  - Current recommendation: [XML 5](#) (2008)

# An HTML Example

<h2>Nonmonotonic Reasoning: Context-  
Dependent Reasoning</h2>

<i>by <b>V. Marek</b> and

<b>M. Truszczyński</b></i><br>

Springer 1993<br>

ISBN 0387976892

# The Same Example in XML

<book>

<title>Nonmonotonic Reasoning: Context-Dependent Reasoning</title>

<author>V. Marek</author>

<author>M. Truszczyński</author>

<publisher>Springer</publisher>

<year>1993</year>

<ISBN>0387976892</ISBN>

</book>

# HTML versus XML: Similarities

- Both use **tags** (e.g. <h2> and </year>)
- Tags may be nested (tags within tags)
- Human users can read and interpret both HTML and XML representations “easily”

... **But how about machines?**



# Problems Interpreting HTML Documents

Problems for a machine trying to get the author names of the book

- Authors' names could appear immediately after the title
- or immediately after the word “*by*” (or “*van*” if it's in Dutch)
- Are there two authors or just one, called “*V. Marek and M. Truszczyński*”?

```
<h2>Nonmonotonic Reasoning: Context-  
Dependent Reasoning</h2>  
<i>by <b>V. Marek</b> and <b>M.  
Truszczyński</b></i><br>  
Springer 1993<br>  
ISBN 0387976892
```

# HTML vs XML: Structural Information

- HTML documents don't carry **structured information**: pieces document and their relations
- XML more easily accessible to machines since
  - Every piece of information is described
  - Relations defined through nesting structure
  - E.g., **<author>** tags appear within **<book>** tags, so they describe properties of a particular book

# HTML vs XML: Structural Information

- A machine processing the XML document can assume (deduce/infer) that
  - **author** element refers to enclosing **book** element
  - Without using background knowledge, *proximity* or other heuristics
- XML allows definition of constraints on values
  - E.g., a year must be a integer of four digits

# HTML vs. XML: Formatting

- HTML representation provides more than XML representation:
  - Formatting of the document is described
- Main use of an HTML document is to display information: it must define formatting
- **XML: separation of content from display**
  - same information can be displayed in different ways
  - Presentation specified by documents using other XML standards (CSS, XSL)

# HTML vs. XML: Another Example

## In HTML

`<h2>Relationship matter-energy</h2>`

`<i> E = M × c2 </i>`

## In XML

`<equation>`

`<gloss>Relationship matter energy </gloss>`

`<leftside> E </leftside>`

`<rightside> M × c2 </rightside>`

`</equation>`

# HTML vs. XML: Different Use of Tags

- All HTML documents use the **same tags**
  - HTML tags come from a finite, pre-defined collection
  - Define properties for display: font, color, lists ...
- XML documents can use completely different tags
  - XML tags not fixed: **user definable tags**
  - XML is a *meta markup language*, i.e., a language for defining markup languages

# XML Vocabularies

- Applications must agree on common vocabularies to communicate and collaborate
- Communities and business sectors define their specialized vocabularies
  - mathematics ([MathML](#))
  - bioinformatics (BSML)
  - human resources (HRML)
  - Syndication (RSS)
  - Vector graphics (SVG)
  - ...

# Outline

(1) Introduction

**(2) Description of XML**

(3) Structuring

- DTDs
- XML Schema

(4) Namespaces

(5) Accessing, querying XML documents: XPath

(6) Transformations: XSLT



# The XML Language

An XML document consists of

- A **prolog**
- A number of **elements**
- An optional **epilog** (not discussed, not used much)

XML documents are tree data structures

# Prolog of an XML Document

The prolog consists of

- An XML declaration and
- An optional reference to external structuring documents

**<?xml version="1.0" encoding="UTF-16"?>**

**<!DOCTYPE book SYSTEM "book.dtd">**

# XML Elements

- Elements are the *things* the XML document talks about
  - E.g., books, authors, publishers, ...
- An element consists of:
  - An opening tag
  - The content
  - A closing tag

**<lecturer> David Billington </lecturer>**

# XML Elements

- Tag names can be chosen almost freely
- First character must be a letter, underscore, or colon
- No name may begin with the string “xml” in any combination of cases
  - E.g. “Xml”, “xML”

# Content of XML Elements

- Content is what's between the tags
- It can be text, or other elements, or nothing

```
<lecturer>
```

```
  <name>David Billington</name>
```

```
  <phone> +61 – 7 – 3875 507 </phone>
```

```
</lecturer>
```

- If there is no content, then element is called empty; it can be abbreviated as follows:

```
<lecturer/> = <lecturer></lecturer>
```

# XML Attributes

- An empty element isn't necessarily meaningless
  - It may have properties expressed as *attributes*
- An **attribute** is a name-value pair inside the opening tag of an element

```
<lecturer  
  name="David Billington"  
  phone="+61 – 7 – 3875 507" />
```

# XML Attributes: An Example

```
<order orderNo="23456"  
      customer="John Smith"  
      date="October 15, 2017" >  
  <item itemNo="a528" quantity="1" />  
  <item itemNo="c817" quantity="3" />  
</order>
```

# The Same Example without Attributes

```
<order>  
  <orderNo>23456</orderNo>  
  <customer>John Smith</customer>  
  <date>October 15, 2017</date>  
  <item>  
    <itemNo>a528</itemNo>  
    <quantity>1</quantity>  
  </item>  
  <item>  
    <itemNo>c817</itemNo>  
    <quantity>3</quantity>  
  </item>  
</order>
```



# XML Elements vs. Attributes

- Attributes can be replaced by elements
- When to use elements and when attributes is a mostly matter of taste
- But attributes cannot be nested

# Further Components of XML Docs

- **Comments**

- A piece of text that is to be ignored by parser

**<!-- This is a comment -->**

- **Processing Instructions (PIs)**

- Define procedural attachments

**<?stylesheet type="text/css"  
href="mystyle.css"?>**

# Well-Formed XML Documents

Constraints on syntactically correct documents:

- Only one outermost element (**root element**)
- Each element contains opening and corresponding closing tag (except self-closing tags like <foo/>)
- Tags may not overlap

`<author><name>Lee Hong</author></name>`

- Attributes within an element have unique names
- Element and tag names must be permissible  
e.g.: can't use strings beginning with digit "2ndbest"

# The Tree Model of XML Docs

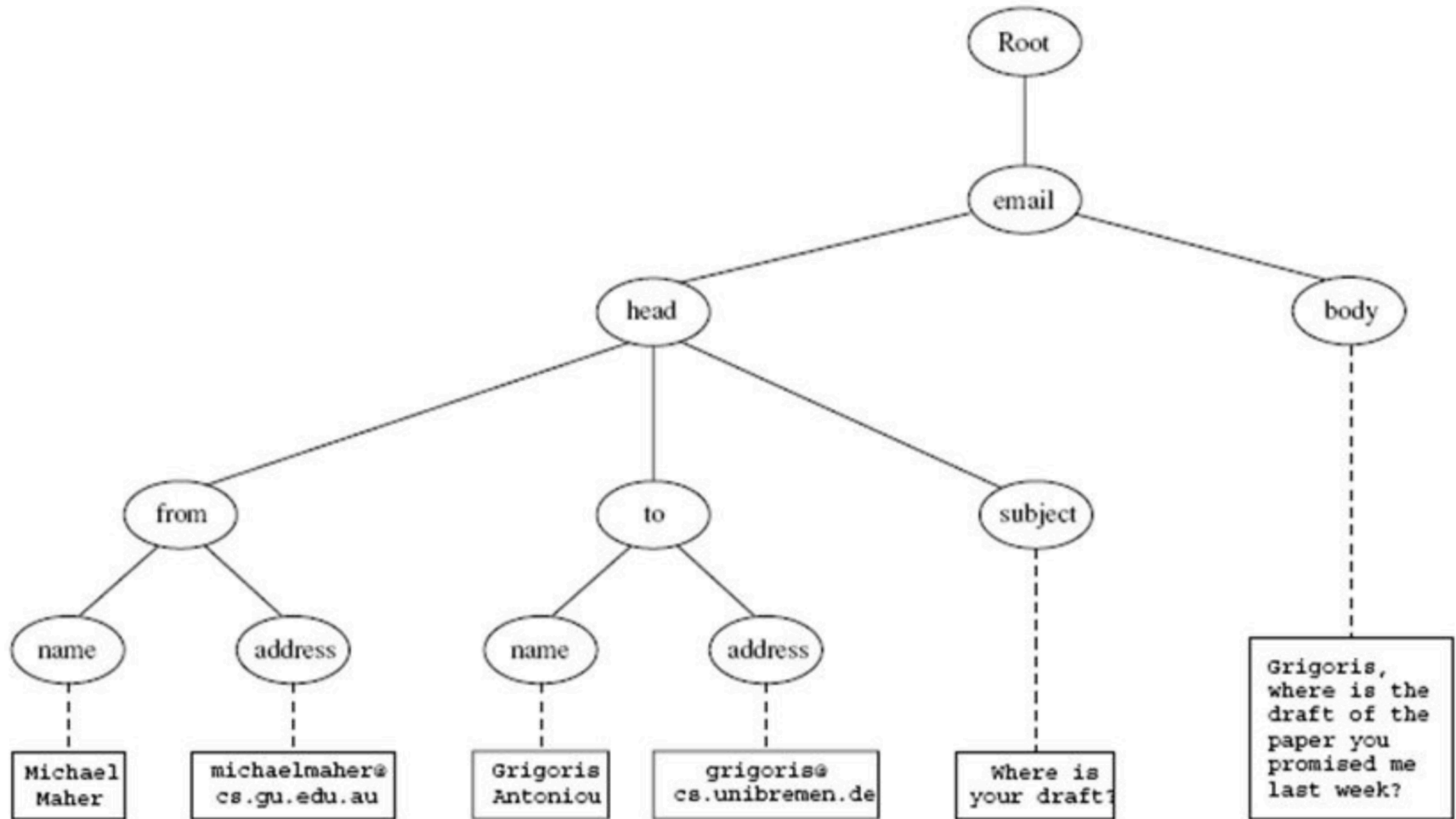
The tree representation of an XML document is an **ordered** labeled tree:

- Exactly one root
- No cycles
- Each non-root node has exactly one parent
- Each node has a label
- Order of elements is important
- ... but order of attributes is not

# Tree Model of XML Documents

```
<email>
  <head>
    <from name="Michael Maher"
      address="michaelmaher@cs.gu.edu.au" />
    <to name="Grigoris Antoniou"
      address="grigoris@cs.unibremen.de" />
    <subject>Where is your draft?</subject>
  </head>
  <body>
    Grigoris, where is the draft of the paper you
    promised me last week?
  </body>
</email>
```

# Tree Model of XML Documents



# Outline

(1) Introduction

(2) Description of XML

(3) **Structuring**

- **DTDs**
- XML Schema

(4) Namespaces

(5) Accessing, querying XML documents: XPath

(6) Transformations: XSLT

# Structuring XML Documents

- Some XML documents must follow constraints defined in a “template” that can...
  - define the *element* and *attribute names* that may be used
  - define the *structure*
    - what values an attribute may take
    - which elements may or must occur within other elements, etc.
- If such structuring information exists, the document can be **validated**



# Structuring XML Documents

- An XML document is **valid** if
  - it is well-formed XML
  - respects the structuring information it uses
- Ways to define structure of XML documents:
  - **DTDs** (*Document Type Definition*) came first, was based on SGML's approach
  - **XML Schema** (aka *XML Schema Definition*, XSD) is more recent and expressive
  - RELAX NG and DSDs are two alternatives

# DTD: Element Type Definition

**<lecturer>**

**<name>David Billington</name>**

**<phone> +61 – 7 – 3875 507 </phone>**

**</lecturer>**

DTD for above element (and all **lecturer** elements):

**<!ELEMENT lecturer (name, phone) >**

**<!ELEMENT name (#PCDATA) >**

**<!ELEMENT phone (#PCDATA) >**

# The Meaning of the DTD

```
<!ELEMENT lecturer (name, phone) >  
<!ELEMENT name (#PCDATA) >  
<!ELEMENT phone (#PCDATA) >
```

- The element types **lecturer**, **name**, and **phone** may be used in the document
- **lecturer** elements contain a **name** element and a **phone** element, in that order (*sequence*)
- **name** and **phone** elements may have any content

In DTDs, **#PCDATA** is the only atomic element type; stands for “*parsed character data*”

# Disjunction in Element Type Definitions

- We say that **lecturer** elements contains *either* a **name** *or* a **phone** element like:

**<!ELEMENT lecturer ( name | phone )>**

- A **lecturer** element contains a **name** element and a **phone** element in *any order*

**<!ELEMENT**

**lecturer((name,phone)|(phone,name))>**

- Do you see a problem with this approach?

# Example of an XML Element

```
<order orderNo="23456"  
      customer="John Smith"  
      date="October 15, 2017">  
  <item itemNo="a528" quantity="1" />  
  <item itemNo="c817" quantity="3" />  
</order>
```

# Corresponding DTD

<!ELEMENT order (item+)>

<!ATTLIST order

orderNo ID #REQUIRED

customer CDATA #REQUIRED

date CDATA #REQUIRED >

<!ELEMENT item EMPTY>

<!ATTLIST item

itemNo ID #REQUIRED

quantity CDATA #REQUIRED

comments CDATA #IMPLIED >

# Comments on the DTD

- The **item** element type is defined to be empty
  - i.e., it can contain no elements
- **+** (after **item**) is a **cardinality operator**:
  - It specifies how many item elements can be in an order
  - **?**: zero times or once
  - **\***: zero or more times
  - **+**: one or more times
  - No cardinality operator:  
once

```
<!ELEMENT order (item+)>
<!ATTLIST
    order orderNo ID #REQUIRED
    customer CDATA #REQUIRED
    date CDATA #REQUIRED >
<!ELEMENT item EMPTY>
<!ATTLIST
    item itemNo ID #REQUIRED
    quantity CDATA #REQUIRED
    comments CDATA #IMPLIED >
```

# Comments on the DTD

- In addition to defining elements, we define attributes
- Done in an **attribute list** containing:
  - Name of element type to which list applies
  - List of triples of attribute name, attribute type, and value type
- *Attribute name*: name that may be used in an XML document using a DTD



# DTD: Attribute Types

- Similar to predefined data types, but limited ...
- The most important types are
  - **CDATA**, a string (sequence of characters)
  - **ID**, a name that is *unique* across the entire XML document (~DB key)
  - **IDREF**, reference to another element with ID attribute carrying same value as IDREF attribute (~ DB foreign key)
  - **IDREFS**, a series of IDREFs
  - **(v1 | ... | vn)**, an enumeration of all possible values
- Limitations: no dates, number ranges, etc.

# DTD: Attribute Value Types

- **#REQUIRED**

- Attribute must appear in every occurrence of the element type in the XML document

- **#IMPLIED**

- The appearance of the attribute is optional

- **#FIXED "value"**

- Every element must have this attribute

- **"value"**

- This specifies the default value for the attribute

# Referencing with IDREF and IDREFS

<!ELEMENT family (person\*)>

<!ELEMENT person (name)>

<!ELEMENT name (#PCDATA)>

<!ATTLIST person

|    |    |           |
|----|----|-----------|
| id | ID | #REQUIRED |
|----|----|-----------|

|        |       |          |
|--------|-------|----------|
| mother | IDREF | #IMPLIED |
|--------|-------|----------|

|        |       |          |
|--------|-------|----------|
| father | IDREF | #IMPLIED |
|--------|-------|----------|

|          |        |            |
|----------|--------|------------|
| children | IDREFS | #IMPLIED > |
|----------|--------|------------|

# An XML Document Respecting the DTD

```
<family>
  <person id="bob" mother="mary" father="peter">
    <name>Bob Marley</name>
  </person>
  <person id="bridget" mother="mary">
    <name>Bridget Jones</name>
  </person>
  <person id="mary" children="bob bridget">
    <name>Mary Poppins</name>
  </person>
  <person id="peter" children="bob">
    <name>Peter Marley</name>
  </person>
</family>
```

# Email Element DTD 1/2

<!ELEMENT **email** (head,body)>

<!ELEMENT **head** (from,to+,cc\*,subject)>

<!ELEMENT **from** EMPTY>

<!ATTLIST **from**

    name        CDATA  #IMPLIED

    address     CDATA  #REQUIRED>

<!ELEMENT **to** EMPTY>

<!ATTLIST **to**

    name        CDATA  #IMPLIED

    address     CDATA  #REQUIRED>

# Email Element DTD 2/2

<!ELEMENT **cc** EMPTY>

<!ATTLIST **cc**

    name    CDATA    #IMPLIED

    address  CDATA    #REQUIRED>

<!ELEMENT **subject** (#PCDATA) >

<!ELEMENT **body** (text,attachment\*) >

<!ELEMENT **text** (#PCDATA) >

<!ELEMENT **attachment** EMPTY >

<!ATTLIST **attachment**

    encoding  (mime|binhex)    "mime"

    file        CDATA        #REQUIRED>

# Outline

(1) Introduction

(2) Description of XML

**(3) Structuring**

- DTDs
- **XML Schema**

(4) Namespaces

(5) Accessing, querying XML documents: XPath

(6) Transformations: XSLT

# XML Schema (XSD)

- XML Schema is a significantly richer language for defining the structure of XML documents
- Syntax based on XML itself, so separate tools to handle them not needed
- Reuse and refinement of schemas => can expand or delete existing schemas
- Sophisticated set of **data types**, compared to DTDs, which only supports strings
- XML Schema recommendation published by W3C in 2001, version 1.1 in 2012



# XML Schema

- An XML schema is an element with an opening tag like

**<schema**

**"http://www.w3.org/2000/10/XMLSchema"**

**version="1.0">**

- Structure of schema elements
  - Element and attribute types using data types

# Element Types

`<element name="email"/>`

`<element name="head"  
 minOccurs="1"  
 maxOccurs="1"/>`

`<element name="to" minOccurs="1"/>`

Cardinality constraints:

- **minOccurs="x"** (default value 1)
- **maxOccurs="x"** (default value 1)
- Generalizations of \*, ?, + offered by DTDs

# Attribute Types

**<attribute name="id" type="ID" use="required"/>**

**<attribute name="speaks" type="Language"  
use="default" value="en"/>**

- Existence: **use="x"**, where **x** may be **optional** or **required**
- Default value: **use="x" value="..."**, where **x** may be **default** or **fixed**

# Data Types

- Many **built-in data types**
  - Numerical data types: **integer**, **short**, etc.
  - String types: **string**, **ID**, **IDREF**, **CDATA**, etc.
  - Date and time data types: **time**, **month**, etc.
- Also **user-defined data types**
  - **simple data types**, which can't use elements or attributes
  - **complex data types**, which can use them

# Complex Data Types

**Complex data types** are defined from existing data types by defining some attributes (if any) and using:

- **sequence**, a sequence of existing data type elements (order is important)
- **all**, a collection of elements that must appear (order is not important)
- **choice**, a collection of elements, of which one will be chosen

# XML Schema: The Email Example

```
<element name="email" type="emailType" />
```

```
<complexType name="emailType">
```

```
  <sequence>
```

```
    <element name="head" type="headType" />
```

```
    <element name="body" type="bodyType" />
```

```
  </sequence>
```

```
</complexType>
```

# XML Schema: The Email Example

```
<complexType name="headType">
```

```
  <sequence>
```

```
    <element name="from" type="nameAddress"/>
```

```
    <element name="to" type="nameAddress"  
      minOccurs="1" maxOccurs="unbounded"/>
```

```
    <element name="cc" type="nameAddress"  
      minOccurs="0" maxOccurs="unbounded"/>
```

```
    <element name="subject" type="string"/>
```

```
  </sequence>
```

```
</complexType>
```

# XML Schema: The Email Example

```
<complexType name="nameAddress">  
  <attribute name="name" type="string"  
    use="optional"/>  
  <attribute name="address"  
    type="string" use="required"/>  
</complexType>
```

- Similar for bodyType



# Next

(1) Introduction

(2) Description of XML

(3) Structuring

- DTDs
- XML Schema

**(4) Namespaces**

**(5) Accessing, querying XML documents: XPath**

**(6) Transformations: XSLT**