XML

Extensible Markup Language

References

- Chapter 2 – Semantic Web Primer
- www.w3c.org
- Numerous good tutorials on the web

  (Nobody offers me a program that combines all of the available presentations on the web to make one for myself ➔ I have to create one !)

Why XML?

- HTML documents = data + formatting instructions
- Data is presented for ‘human’-processing – HTML is for ‘displaying data’

➔ HTML documents = machine-(un)readable

Why XML?

- Current data on the web is often ‘self-describing’ (including name of attributes within the data fields), especially dynamic data – exporting to the Web by applications (e.g., database applications)

Example of self-describing data

```
<html>
<head> <title>Student List</title>
<body>
   <h1>List Name: Students</h1>
   <dl> <dt> Name: John Doe </dt>
          <dt> Id: 111111111 </dt>
          <dt> Address:
               <ul> Number: 123 </ul>
               <ul> Street: Main St. </ul>
          </dt>
           …….
    <dl>
</body>
</html>
```

Students

Name	Id	Address	
		Number	Street
John Doe	111111111	123	Main St
Joe Public	666666666	666	Hollow Rd

Vision for Web data

- *Object-like* – it can be represented as a collection of objects of the form described by the conceptual data model
- *Schemaless* – not conformed to any type structure
- *Self-describing* – necessary for machine readable data

➔*HTML* is not sufficiently good for describing data

➔*XML*

XML – Overview

- Simplifying the data exchange between software agents
- Popular thanks to the involvement of W3C (World Wide Web Consortium – independent organization

  www.w3c.org)

XML – Characteristics

- Simple, open, widely accepted
- HTML-like (tags) but extensible by users (no fixed set of tags)
- No predefined semantics for the tags (because XML is developed not for the displaying purpose)
- Semantics is defined by *stylesheet* (later)

XML vs. HTML

- XML was designed for describing data
- XML is not a replacement for HTML
- XML and HTML were designed with different goals:
  - XML was designed to describe data and to focus on what data is.
  - HTML was designed to display data and to focus on how data looks.
  - HTML is about displaying information, while XML is about describing information.

XML Syntax

XML element

- Begin with a *opening tag* of the form

  <XML_element_name>
- End with a *closing tag*

  </XML_element_name>
- The text between the beginning tag and the closing tag is called the *content* of the element

XML Elements

<tag> ← opening tag

content content

</tag> ← closing tag

- represented "*things*"
- tag can be (almost) arbitrary string
- can be nested

XML Element – Example

<PersonList>

here is the list of persons with names,

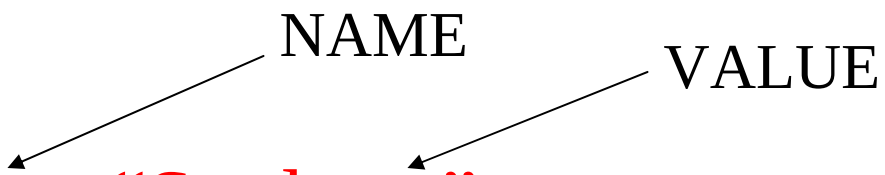
addresses

…

</PersonList>

Attributes

- A name-value pair within the opening tag of a XML element – describes a property of the element



<PersonList Type="Students">

here is the list of students with names, addresses

</PersonList>

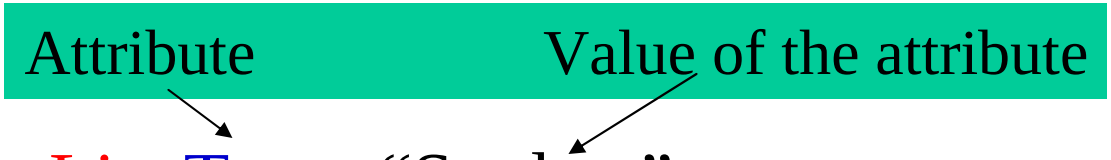
XML Attributes

- Can occur within an element (arbitrary many attributes, order unimportant, same attribute only one)
- Allow a more concise representation
- Could be replaced by elements
- Less powerful than elements (only string value, no children)
- Can be declared to have unique value, good for integrity constraint enforcement (later)

XML Attributes

- Can be declared to be the type of ID, IDREF, or IDREFS
- ID: unique value throughout the document
- IDREF: refer to a valid ID declared in the same document
- IDREFS: space-separated list of strings of references to valid IDs

XML element and attributes




```
<PersonList Type="Student">
   <Student StudentID="123">
   <Name> <First>"XYZ"</First>
        <Last>"PQR"</Last> </Name>
   <CrsTaken CrsName="CS582" Grade="A"/>
   </Student>
   …
</PersonList>
```

Comments and Instructions

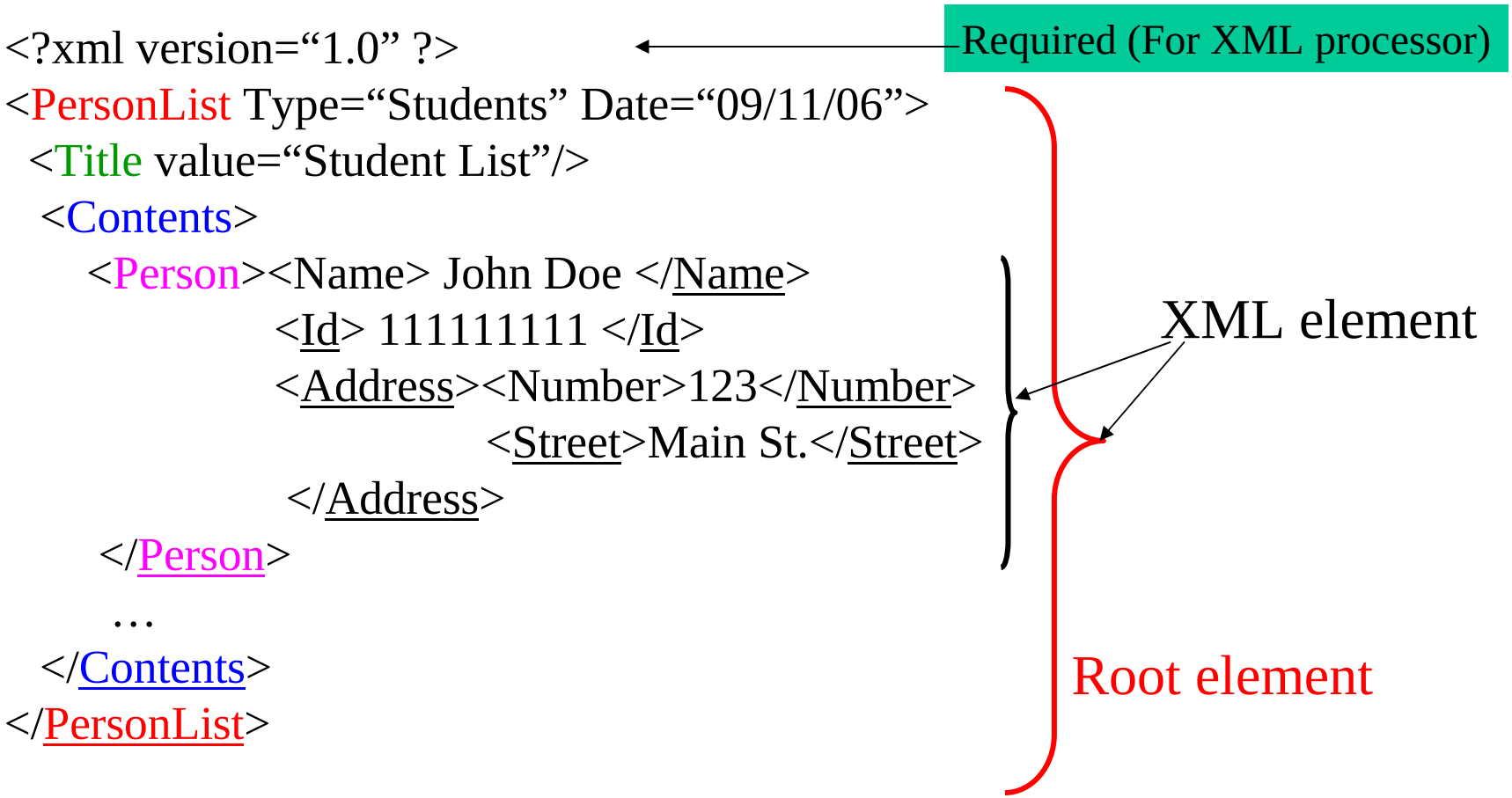
Comment:

<! -- This is a comment -->

Processing instructions

<? target instruction ?>

A Student List in XML representation



XML Documents

- User-defined tags:

  <tag> info </tag>
- Properly nested:<tag1>.. <tag2>…</tag1></tag2>

  is not valid
- Root element: an element contains all other elements
- Processing instructions <?command ….?>
- Comments <!--- comment --- >
- CDATA type
- DTD

Relationship between XML elements

- Child-parent relationship
  - Elements nested directly in an element are the children of this element (*Student* is a child of *PersonList*, *Name* is a child of *Student*, etc.)
- Ancestor/descendant relationship: important for querying XML documents (extending the child/parent relationship)

Well-formed XML Document

- It has a root element
- Every opening tag is followed by a matching closing tag, elements are properly nested
- Any attribute can occur at most once in a given opening tag, its value must be provided, quoted

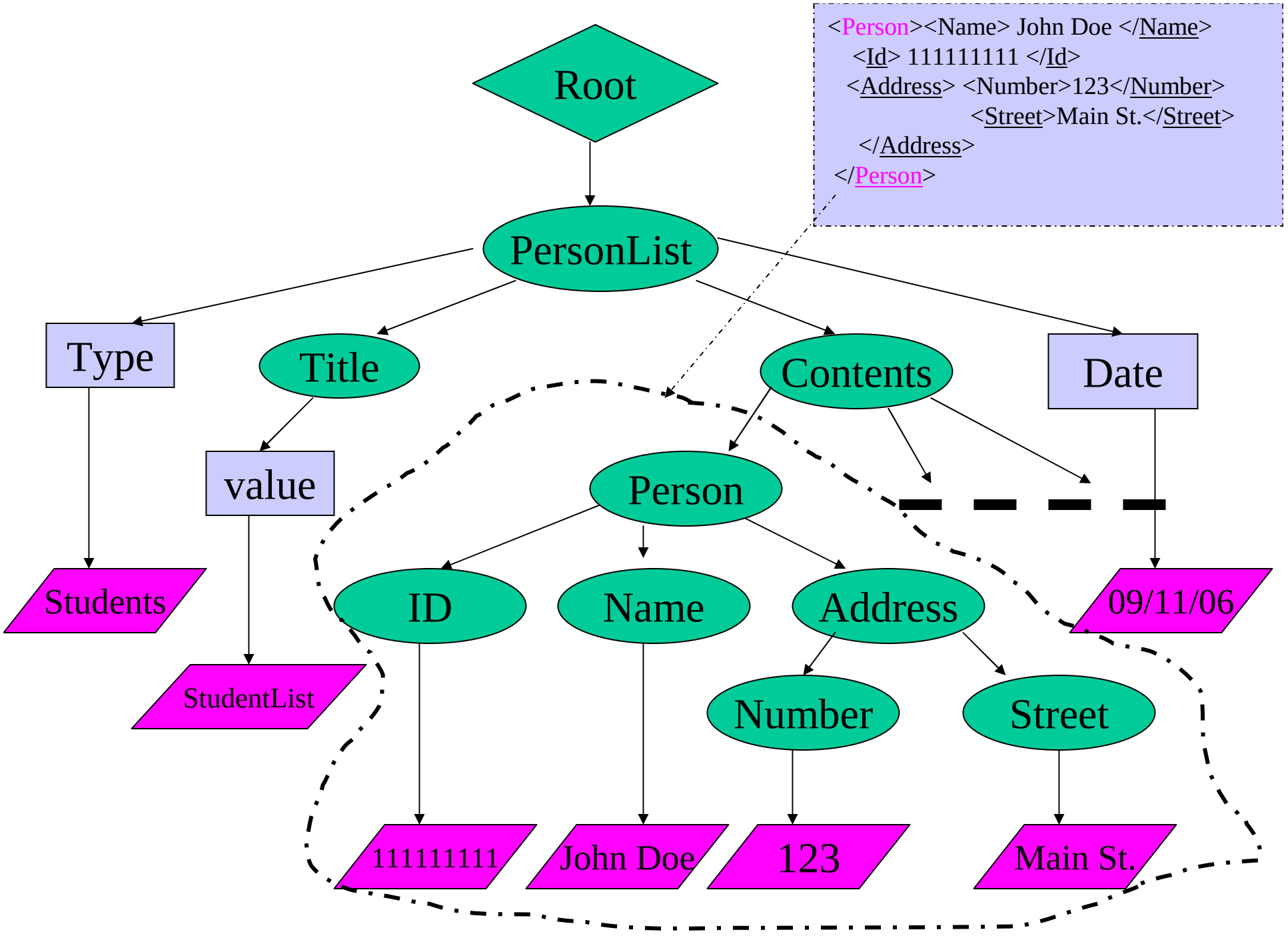
Model of XML Documents

- A well-formed XML document can be represented by a tree
  - A node corresponds to an element
  - A child/parent relationship in tree corresponds to a child/parent relationship in document
  - The root of the tree corresponds to the document
- What's good about this representation?

XML document as Tree

```
<?xml version="1.0" ?>
<PersonList Type="Students" Date="09/11/06">
  <Title value="Student List"/>
   <Contents>
      <Person><Name> John Doe </Name>
           <Id> 111111111 </Id>
           <Address><Number>123</Number>
                  <Street>Main St.</Street>
             </Address>
       </Person>
        …
   </Contents>
</PersonList>
```

→ NEXT PAGE


<Person><Name> John Doe </Name>
<Id> 111111111 </Id>
<Address> <Number>123</Number>
<Street>Main St.</Street>
</Address>
</Person>
Root
PersonList
Type
Title
Contents
Date
value
Person
Students
ID
Name
Address
09/11/06
StudentList
Number
Street
111111111
John Doe
123
Main St.

What's good about … ?

- Algorithms operated on trees can be applied in
  - search (for something in the document)
  - merge (Union, Set difference, etc.)
- Easy to validate data consistency: data in a XML document can be validated according to a predefined document, which is agreed upon by parties using the data

Document Type Definition

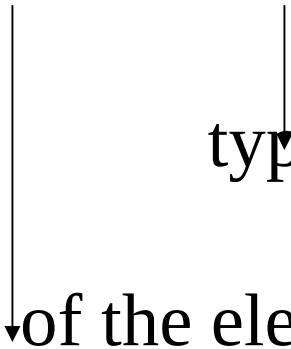
- Set of rules (by the user) for structuring an XML document
- Can be part of the document itself, or can be specified via a URL where the DTD can be found
- A document that conforms to a DTD is said to be *valid*
- Viewed as a *grammar* that specifies a legal XML document, based on the tags used in the document

DTD Components

- A **name** – must coincide with the tag of the root element of the document conforming to the DTD
- A set of **ELEMENT**s – one ELEMENT for each allowed tag, including the root tag
- **ATTLIST** statements – specifies the allow attributes and their type for each tag
- *, +, ? – like in grammar definition
  - * : zero or finitely many number
  - + : at least one
  - ? : zero or one

DTD Components – Element

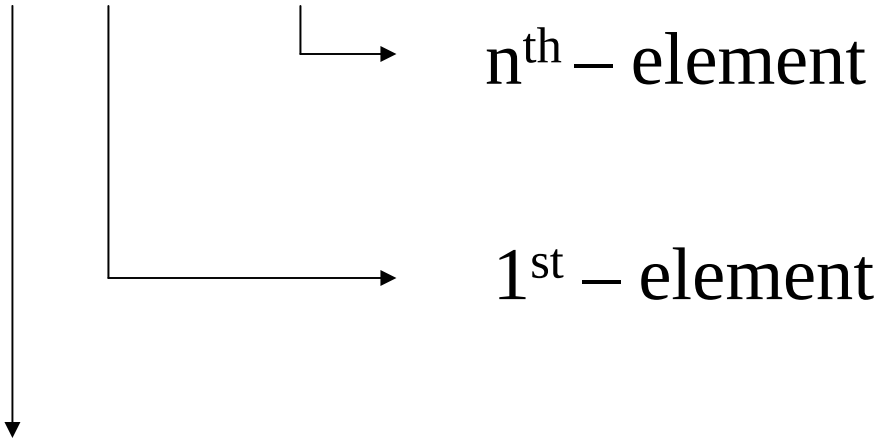
<!ELEMENT Name definition>



definition can be: EMPTY, (#PCDATA, CDATA), or element list (e1,e2,…,en) where the list (e1,e2,…,en) can be shortened using grammar like notation

DTD Components – Element

<!ELEMENT Name(e1,…,en)>



<!ELEMENT PersonList (Title,Contents)>

<!ELEMENT Contents(Person *)>

DTD Components – Element

<!ELEMENT Name EMPTY>

no child for the element Name

<!ELEMENT Name (#PCDATA)>

value of Name is a character string

<!ELEMENT Title EMPTY>

<!ELEMENT Id (#PCDATA)>

DTD Components – Attribute List

<!ATTLIST EName Att {Type} Property>

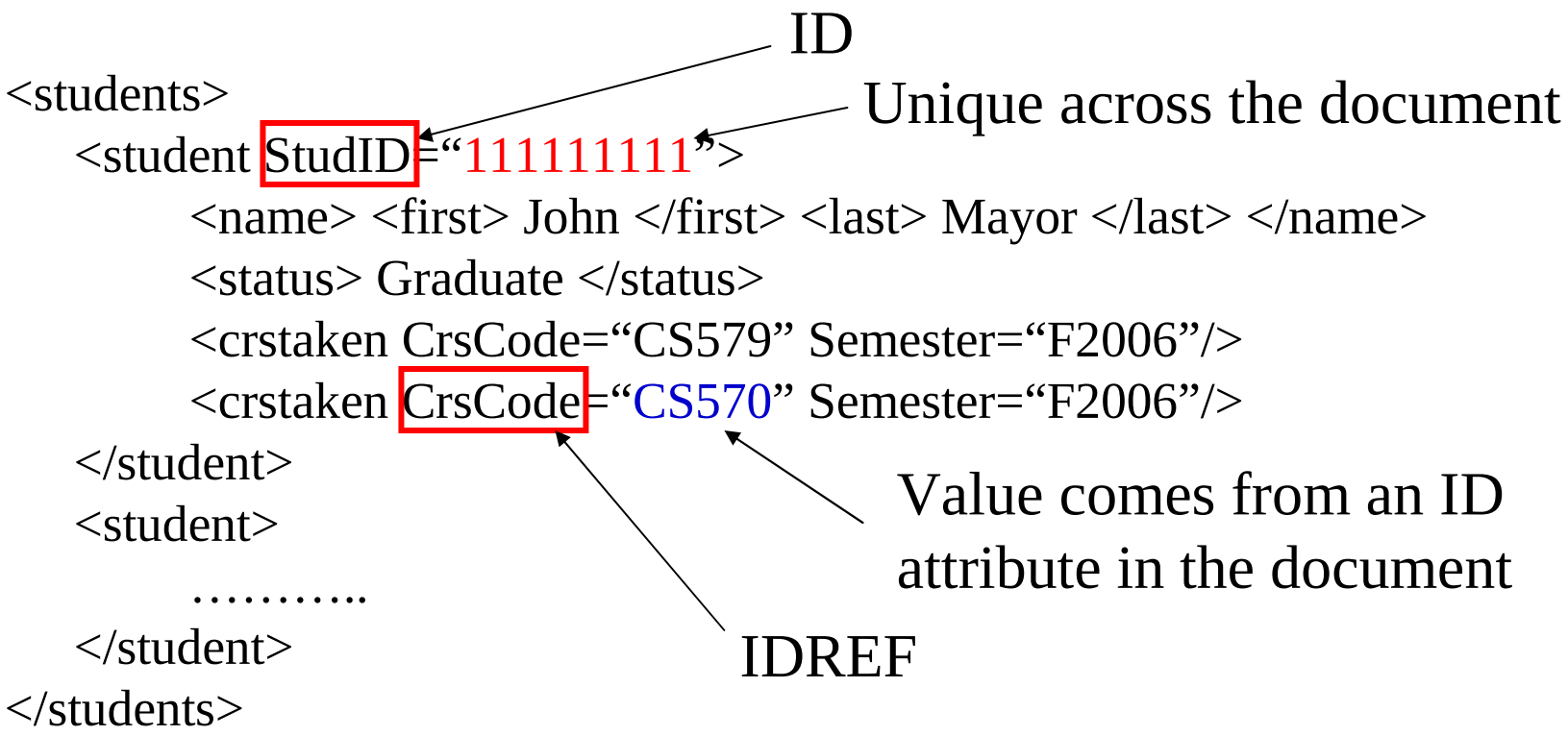
where

- Ename – name of an element defined in the DTD
- Att – attribute name allowed to occur in the opening tag of Ename
- {type} – might/might not be there; specify the type of the attribute (CDATA, ID, IDREF, IDREFS)
- Property – either #REQUIRED or #IMPLIED

DTD Example

```
<!DOCTYPE Report[
    <!ELEMENT Report (Students, Classes, Courses)>
    <!ELEMENT Students (Student*)>
    <!ELEMENT Classes (Class *)>
    <!ELEMENT Courses (Course *)>
    <!ELEMENT Student (Name, Status, CrsTaken *)>
    <!ELEMENT Name (First, Last)>
    <!ELEMENT First (#PCDATA)>
    …
    <!ELEMENT CrsTaken EMPTY>
    <!ELEMENT Class (CrsCode, Semester, ClassRoster)>
    <!ELEMENT Course (CrsName)>
    …
    <!ELEMENT ClassRoster EMPTY>
    <!ATTLIST   Report Date #IMPLIED>
     <!ATTLIST  Student StudID ID #REQUIRED>
     <!ATTLIST  Course CrsCode ID #REQUIRED>
     <!ATTLIST  CrsTaken CrsCode IDREF #REQUIRED>
     <!ATTLIST  CrsTaken Semester IDREF #REQUIRED>
     <!ATTLIST  ClassRoster Members IDREFS #REQUIRED>
]>
```

The element “Students”



DTD as Data Definition Language?

- Can specify what is allowed on the document
- XML elements can be converted into objects
- Can specify integrity constraints on the elements
- Is is good enough?

Inadequacy of DTD as a Data Definition Language

- Goal of XML: for specifying documents that can be exchanged and automatically processed by software agents
- DTD provides the possibility of querying Web documents but has many limitations (next slide)

Inadequacy of DTD as a Data Definition Language

- Designed for a single document (combination of dtd?)
- Syntax is very different than that of XML
- Limited basic types
- Limited means for expressing data consistency constrains
- Enforcing referential integrity for attributes but not elements
- XML data is ordered; not database data
- Element definitions are global to the entire document

That’s why we will meet in the next class ☺

and a new homework

Homework 2

- Create a XML document along with a DTD for representing a list of graduate students in the semantic web class that satisfies the following conditions:
  - The information for each student should include:
    - Student ID,
    - Name (First, Last, Middle),
    - Major,
    - Number of Semester at NMSU,
    - Degree pursuit,
    - Interests, and
    - his/her Transcript.
  - Do you best to make sure that
    - the Student ID is unique,
    - the Major is one of the department names at NMSU,
    - the degree is one of the degrees offered by the CS department (doctoral, master, bachelor)
    - the class on the transcripts is one of the class offered by NMSU
  - List the constraints that you can (and cannot) enforce in this representation.

So far

- Why XML?
- XML elements
- XML attributes
- Well-formed XML document