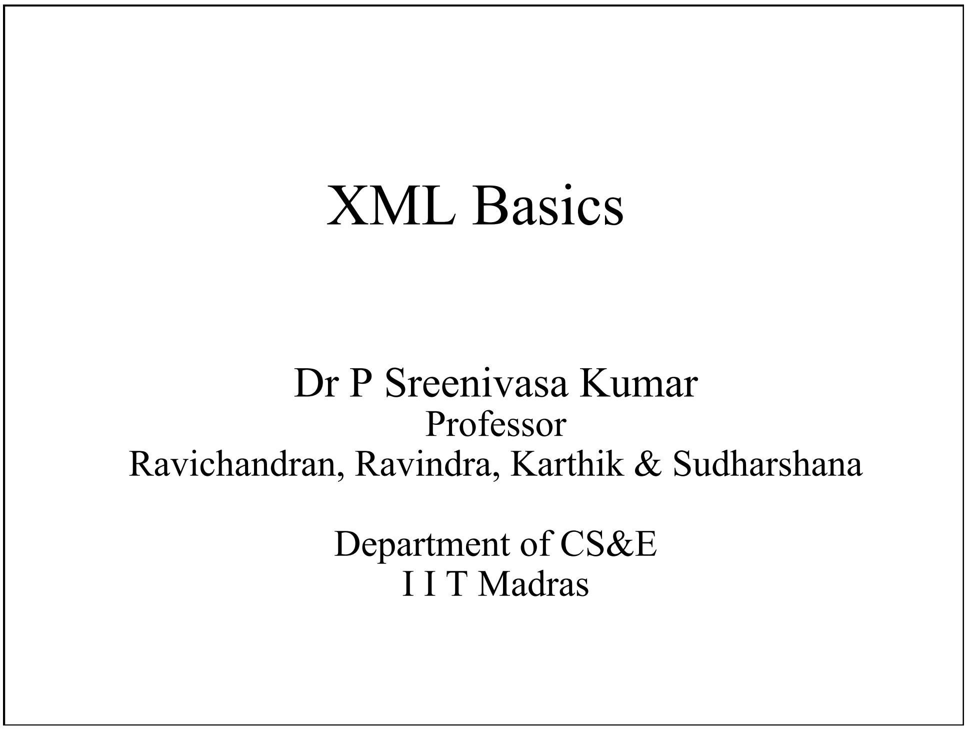
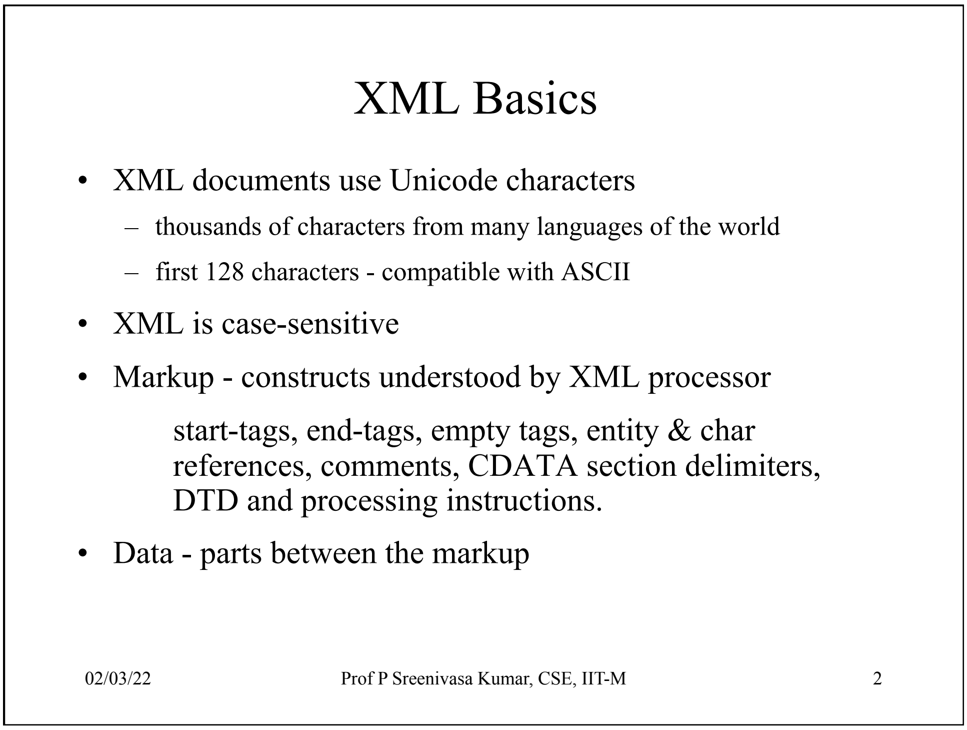


XML Basics

Dr P Sreenivasa Kumar
Professor
Ravichandran, Ravindra, Karthik & Sudharshana

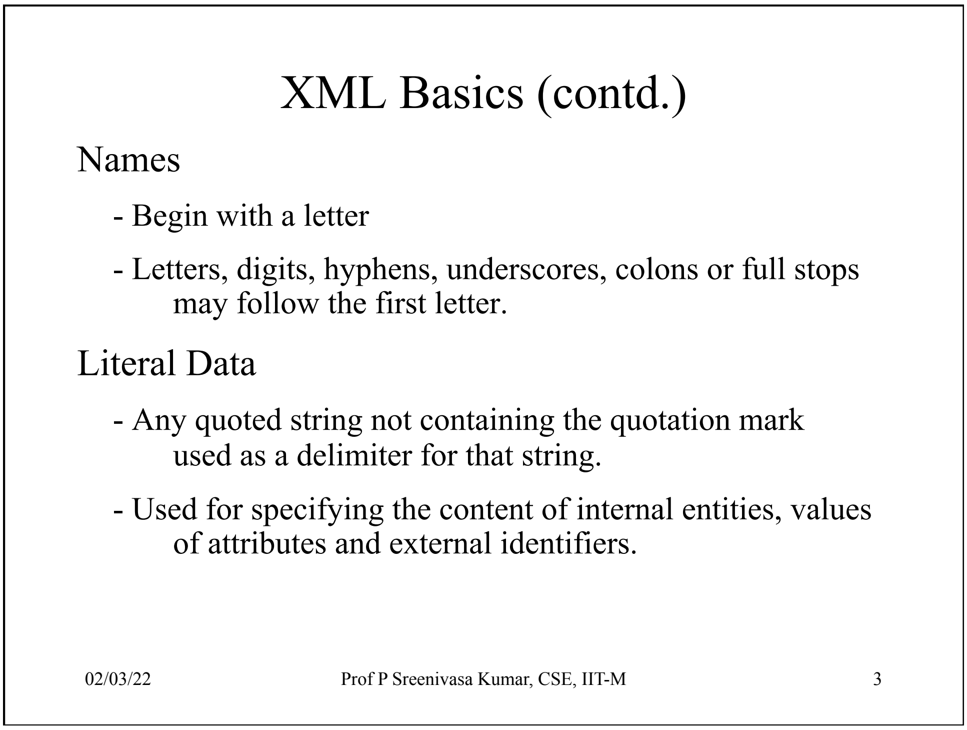
Department of CS&E
I I T Madras



XML Basics

- XML documents use Unicode characters
  - thousands of characters from many languages of the world
  - first 128 characters - compatible with ASCII
- XML is case-sensitive
- Markup - constructs understood by XML processor

  start-tags, end-tags, empty tags, entity & char references, comments, CDATA section delimiters, DTD and processing instructions.
- Data - parts between the markup



XML Basics (contd.)

Names

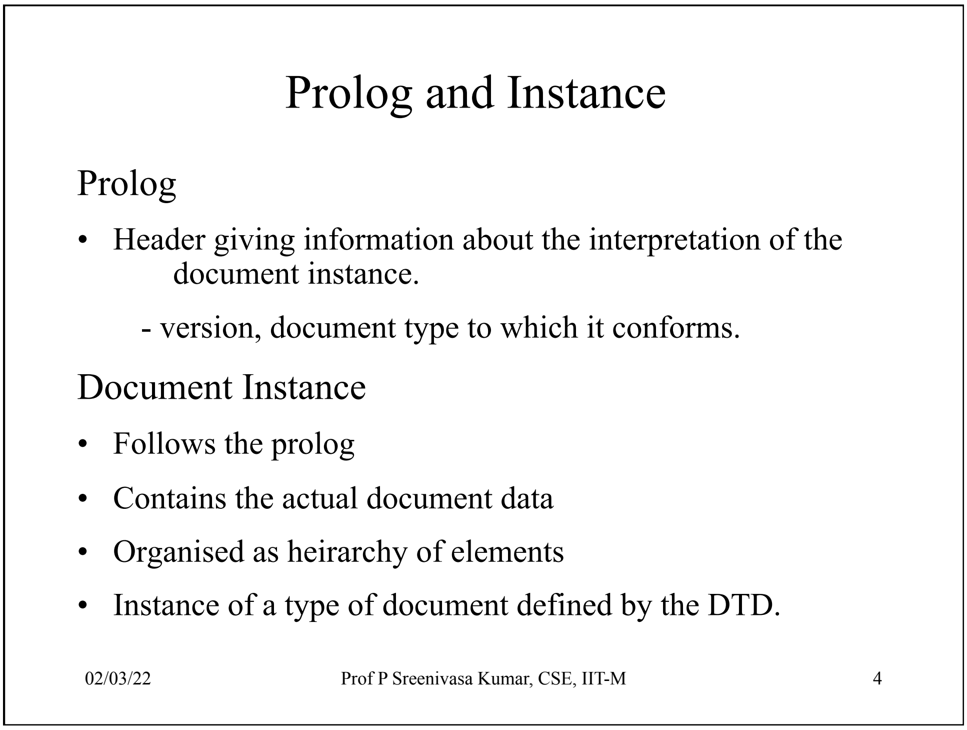
- Begin with a letter

- Letters, digits, hyphens, underscores, colons or full stops may follow the first letter.

Literal Data

- Any quoted string not containing the quotation mark used as a delimiter for that string.

- Used for specifying the content of internal entities, values of attributes and external identifiers.



Prolog and Instance

Prolog

- Header giving information about the interpretation of the document instance.
  - version, document type to which it conforms.

Document Instance

- Follows the prolog
- Contains the actual document data
- Organised as heirarchy of elements
- Instance of a type of document defined by the DTD.

Document Prolog

Every XML document should start with a Prolog.

Syntax :

<?xml *version*=“1.0” *encoding*=“UTF-8” *standalone*=“yes”?>

Version- Declares the version of XML that is in use.
It is required in all declarations.

Encoding- Describes the character encoding used.

Standalone-Declares if any external components of the DTD are necessary for complete processing of the document.

Other Markup Constructs

Predefined Entities

- Used to escape from markup interpretation

&amp; for “ & ” &lt; for “ < ” &gt; for “ > ”,
&apos; for “ ’ ” &quot; for “ ” ”

CDATA Section

- Stands for Character data
- Not interpreted by the procesor

<! [CDATA[*content*]]>

- “]]> ” must not occur anywhere else in *content*

Other Markup Constructs (contd.)

Comments

Ignored by the computer processes and renditions of the document

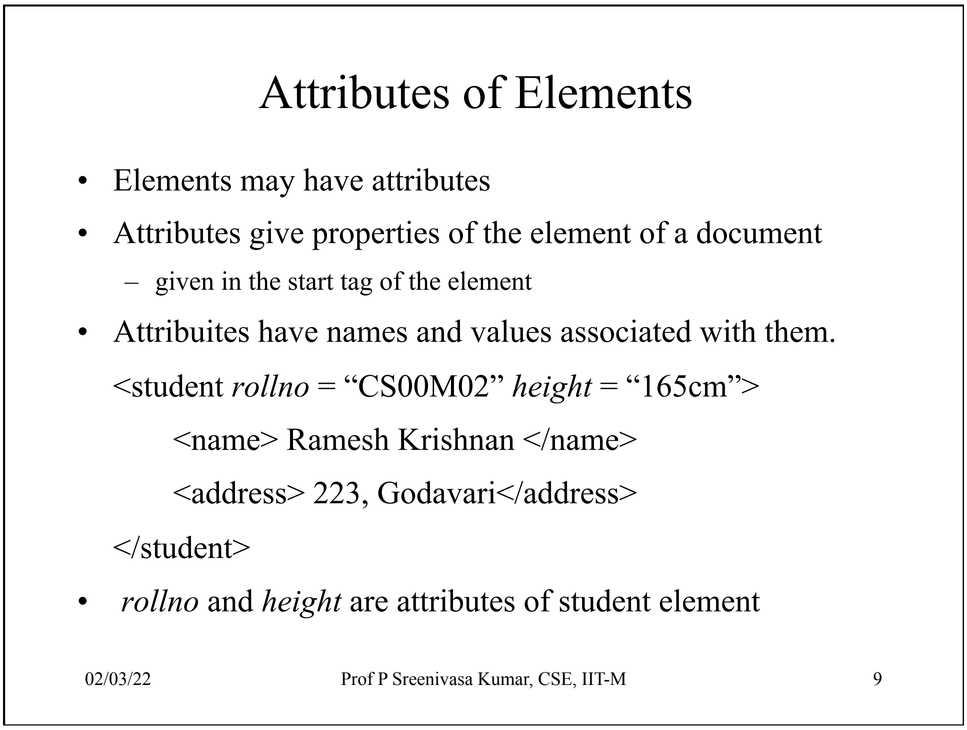
Syntax:

<!- - This is a comment - - >

comments cannot contain the characters “- -” in the middle

Logical Structure - Elements

- Most important component
  - A start tag and an end tag together with the data enclosed by them represent an element.
- Each XML document has *exactly* one root element
- Elements have character data, other elements or both as their content
- Empty Element – does not have any content
  <element-name/>
  – used mainly for its attributes!



Attributes of Elements

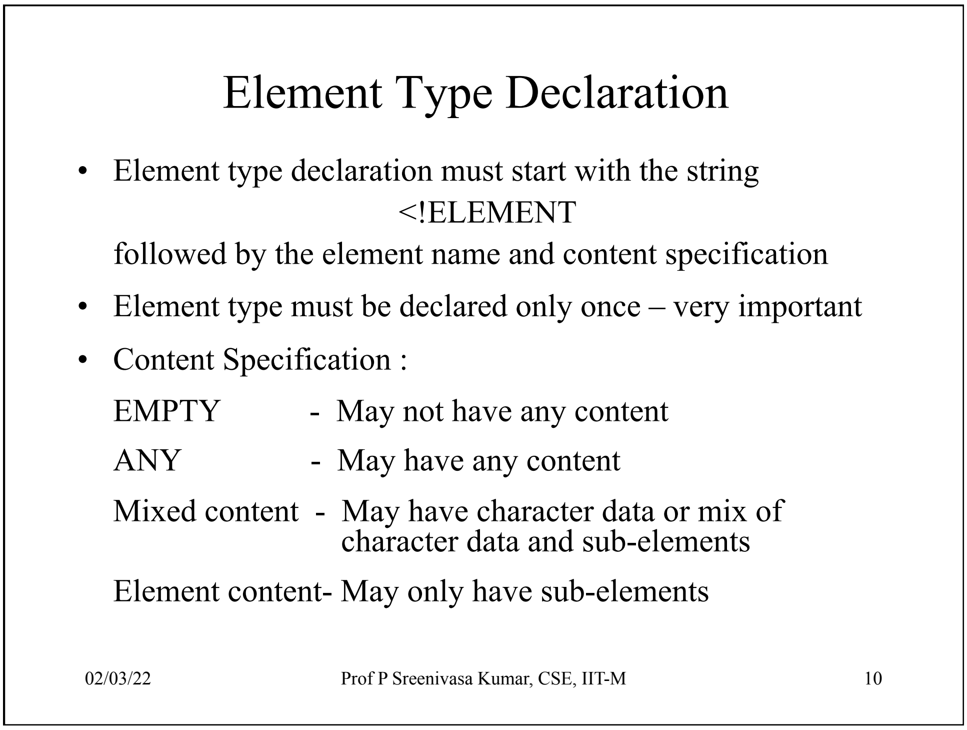
- Elements may have attributes
- Attributes give properties of the element of a document
  - given in the start tag of the element
- Attribuites have names and values associated with them.

  <student *rollno* = “CS00M02” *height* = “165cm”>

  <name> Ramesh Krishnan </name>

  <address> 223, Godavari</address>

  </student>
- *rollno* and *height* are attributes of student element



Element Type Declaration

- Element type declaration must start with the string

  <!ELEMENT

  followed by the element name and content specification
- Element type must be declared only once – very important
- Content Specification :

  EMPTY - May not have any content

  ANY - May have any content

  Mixed content - May have character data or mix of character data and sub-elements

  Element content- May only have sub-elements

Content Model

- A pattern that declares the sub-elements allowed and their order of occurence

```
<!ELEMENT memo( from, to, subject, body )>
```

- XML allows us to specify that a content particle is optional or repeatable using Occurence Indicators
- Element Occurence Indicators:

  * - optional and repeatable (0 or more times)

  ? - optional (0 or 1 time)

  + - required and repeatable (1 or more times)

City DTD

```
<!ELEMENT country (name, city+) >
<!ATTLIST country countryId ID #FIXED “IN”>
<!ELEMENT city (name, population?, state)>
<!ATTLIST city cityId ID #REQUIRED >
<!ELEMENT name (#PCDATA)>
<!ELEMENT population (NUMBER)>
<!ELEMENT state (#PCDATA)>
<!ATTLIST state stateId ID #REQUIRED>
```

Attribute List Declarations

- Declaration starts with the string

  <!ATTLIST

  followed by element name, attribute name, its type and its default.

  <!ATTLIST shirt size NUMBER #IMPLIED >
  #IMPLIED – attribute is optional

  <!ATTLIST person email CDATA #REQUIRED>
  #REQUIRED – attribute must be always present

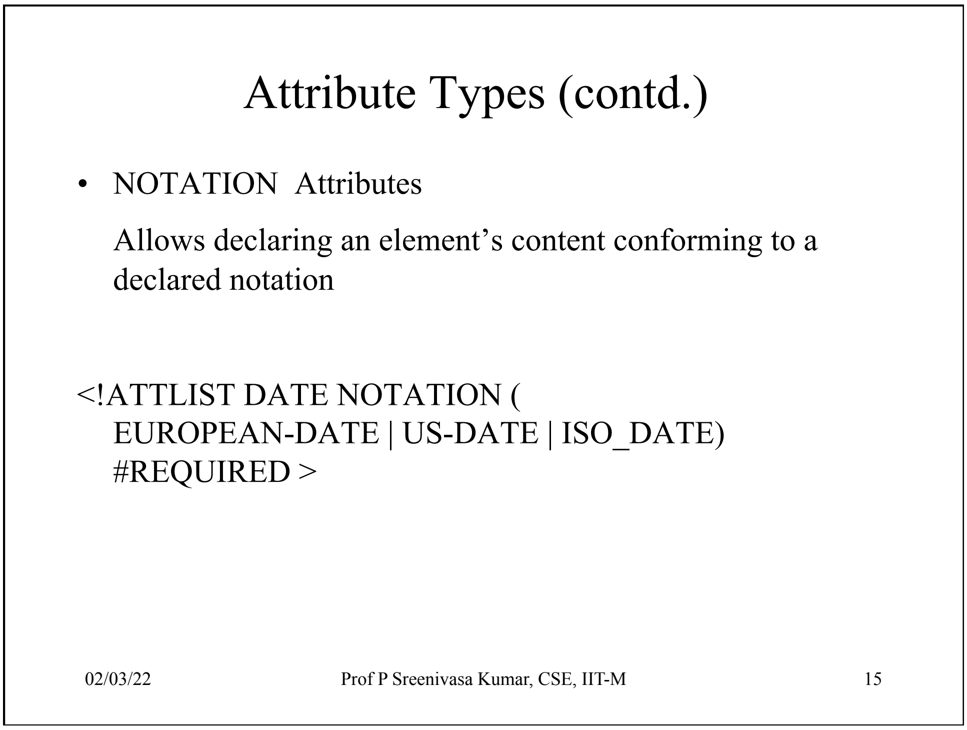
  #FIXED – attribute has a constant value
- Attributes can have default values.

  <!ATTLIST shoes size NUMBER “8” >

Attribute Types

- CDATA
  - Stands for character data.
  - Attribute values of this type can be any string of characters.
- Enumerated attributes
  - Provides a choice of options for attribute values.
  - Syntax:

    <!ATTLIST person sex (male | female) #REQUIRED>

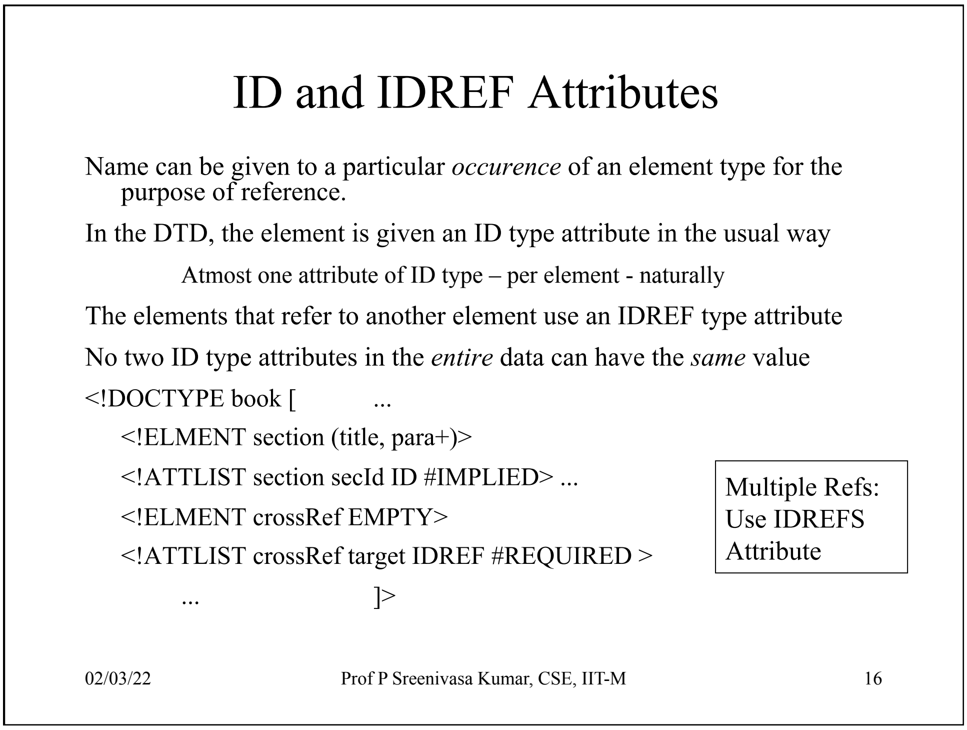


Attribute Types (contd.)

- NOTATION Attributes

  Allows declaring an element’s content conforming to a declared notation

```
<!ATTLIST DATE NOTATION (
   EUROPEAN-DATE | US-DATE | ISO_DATE)
   #REQUIRED >
```



ID and IDREF Attributes

Name can be given to a particular *occurence* of an element type for the purpose of reference.

In the DTD, the element is given an ID type attribute in the usual way

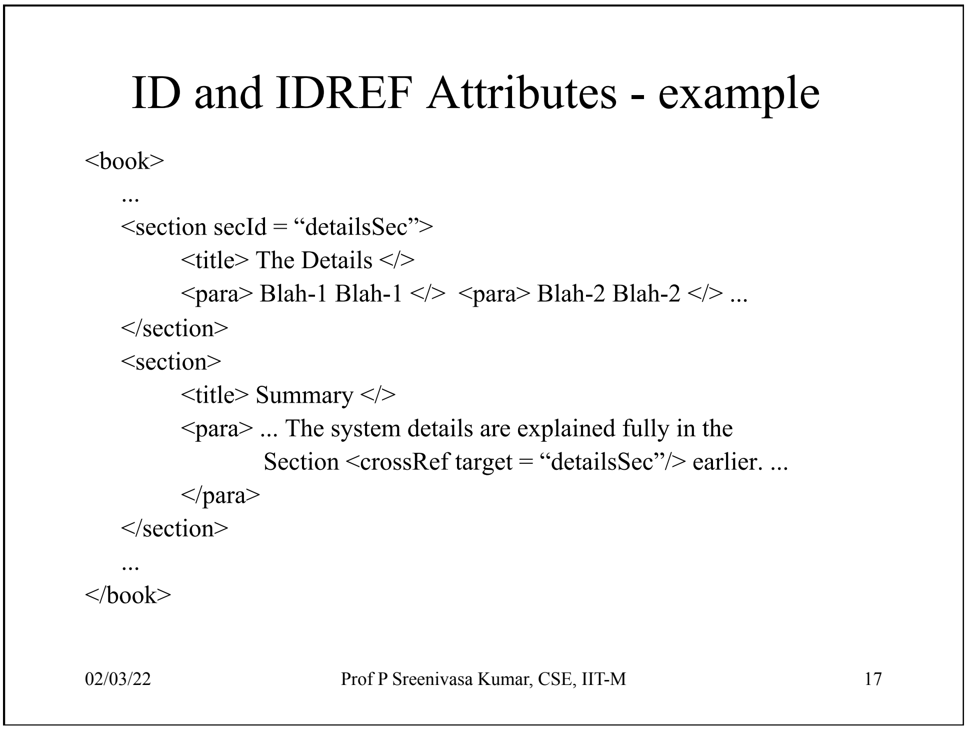
Atmost one attribute of ID type – per element - naturally

The elements that refer to another element use an IDREF type attribute

No two ID type attributes in the *entire* data can have the *same* value

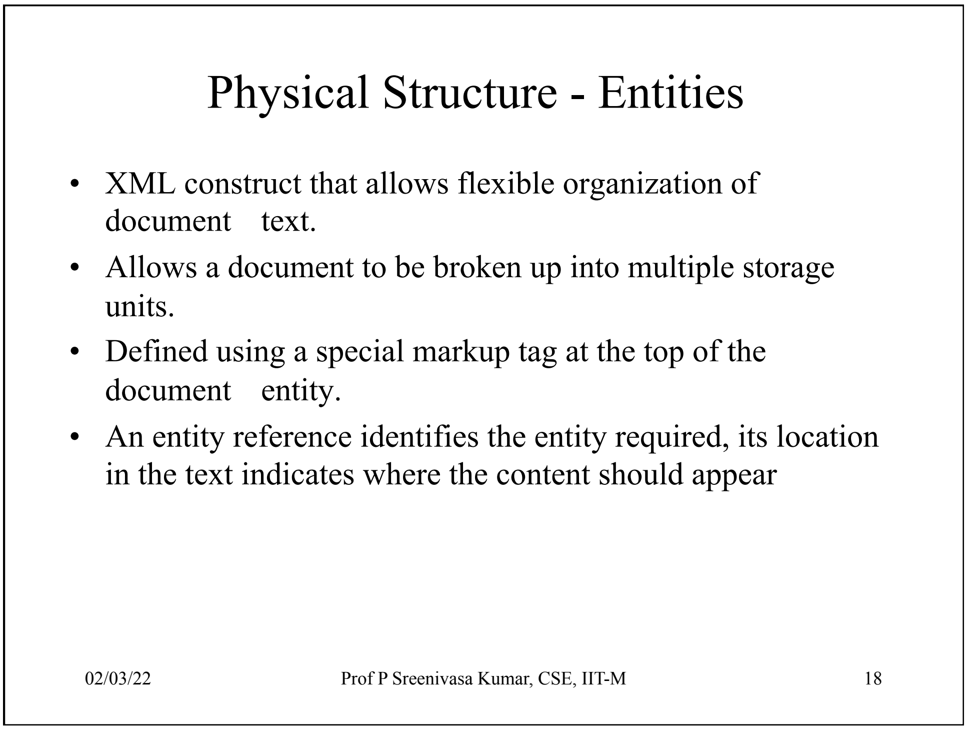
```
<!DOCTYPE book [           ...
   <!ELMENT section (title, para+)>
   <!ATTLIST section secId ID #IMPLIED> ...
   <!ELMENT crossRef EMPTY>
   <!ATTLIST crossRef target IDREF #REQUIRED >
          ...                    ]>
```

Multiple Refs: Use IDREFS Attribute



ID and IDREF Attributes - example

```
<book>
    ...
    <section secId = “detailsSec”>
            <title> The Details </>
            <para> Blah-1 Blah-1 </>  <para> Blah-2 Blah-2 </> ...
    </section>
    <section>
            <title> Summary </>
            <para> ... The system details are explained fully in the
                        Section <crossRef target = “detailsSec”/> earlier. ...
            </para>
    </section>
    ...
</book>
```



Physical Structure - Entities

- XML construct that allows flexible organization of document text.
- Allows a document to be broken up into multiple storage units.
- Defined using a special markup tag at the top of the document entity.
- An entity reference identifies the entity required, its location in the text indicates where the content should appear

Types of Entities

Internal Entity :

- Content is stored within the main document

External Entity :

- Content is stored in a separate file

General Entity :

- Referenced within the document instance

Parameter Entity :

- Referenced only within markup declarations

Entity Declaration

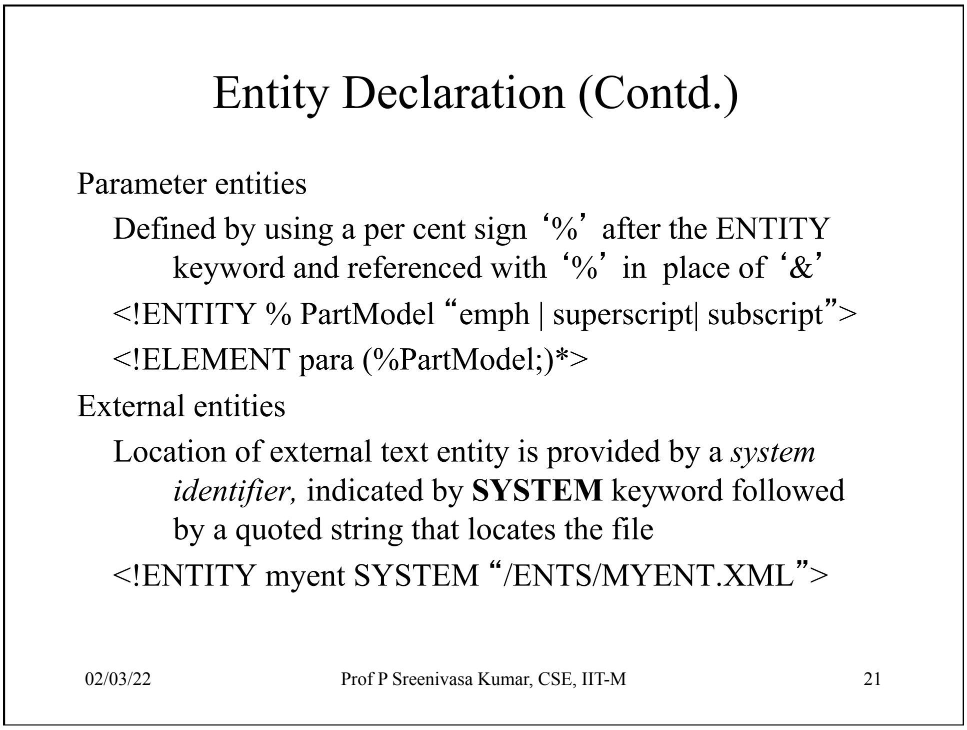
Internal text entities

- Replacement text is contained within quote delimiters following the entity name.

  <!ENTITY XML “extensible markup language ”>

- The above entity is referenced as &XML;
- In character entity reference a # (hash symbol) is inserted after the ampersand.

  ‘&#60’ refers to less-than symbol (‘<’)



Entity Declaration (Contd.)

Parameter entities

- Defined by using a per cent sign ‘%’ after the ENTITY keyword and referenced with ‘%’ in place of ‘&’
- <!ENTITY % PartModel “emph | superscript| subscript”>
- <!ELEMENT para (%PartModel;)*>

External entities

- Location of external text entity is provided by a *system identifier,* indicated by **SYSTEM** keyword followed by a quoted string that locates the file
- <!ENTITY myent SYSTEM “/ENTS/MYENT.XML”>