# XML

Prof. Vipul Dabhi
Department of Information Technology,
D. D. University.

# What Is Markup?

- Information added to a text to make its structure comprehensible
- Pre-computer markup (punctuational and presentational)
  - Word divisions
  - Punctuation
  - Copy-editor and typesetters marks
  - Formatting conventions

# Computer markup

- Any kind of codes added to a document
  - Typesetting (presentational markup)
    - MS Word and its ilk, TeX, Scribe, Lout, Script, nroff, XYVision
  - Declarative markup
    - HTML (sometimes)
    - XML

# What is XML?

- a meta language that allows you to create and format your own document markups
- a method for putting structured data into a text file;  these files are
    - easy to read
    - unambiguous
    - extensible
    - platform-independent

# Comparisons: HTML Vs XML

## XML

- Extensible set of tags: allows user to specify what each tag and attribute means
- Content orientated
- Standard Data infrastructure
- Allows multiple output forms
- content and format are separate; formatting is contained in a stylesheet
- XML is case sensitive
- XML is dynamic and used to transfer data

## HTML

- Fixed set of tags: tags and attributes are pre-determined and rigid
- Presentation oriented
- No data validation capabilities
- Single presentation
- Content and formatting can be placed together

  <p><font=”Arial”>text</font>
- HTML is not case sensitive
- HTML is static

# Weaknesses of HTML

- Fixed set of tags
- Predefined semantics for each tag
- Predefined data structure
- No formal validation
- Does not support semantic search
- Based solely on appearance and not on content
- Won’t do complex document

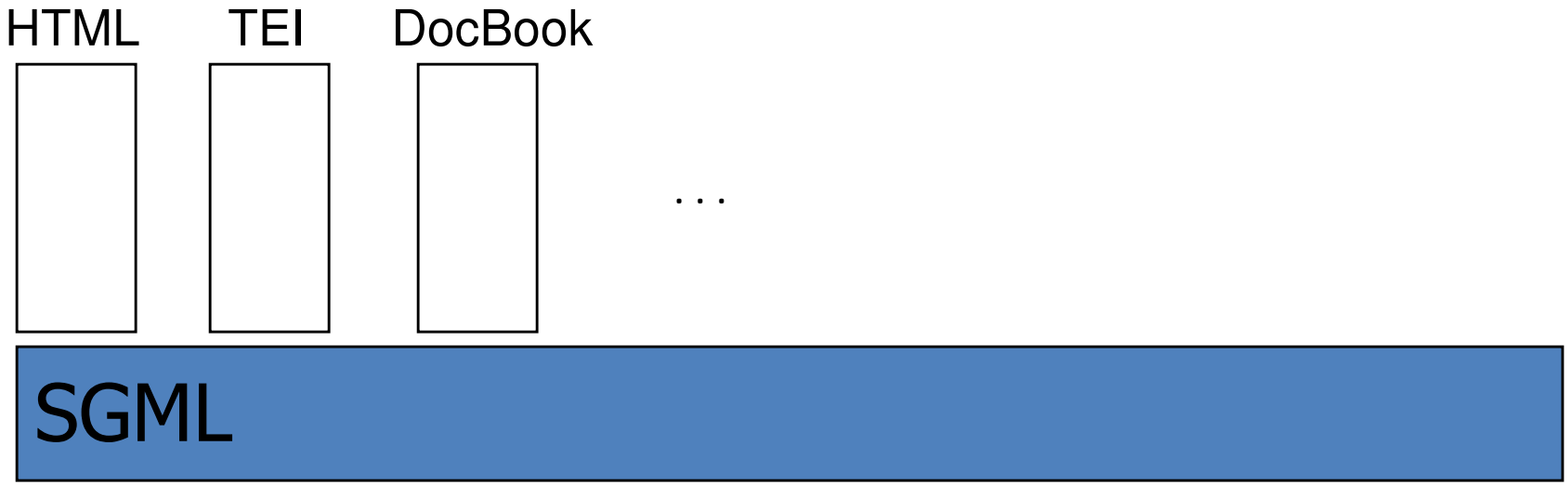
# HTML Problems

- Desire for ***personalized*** tags
- Want to put data into HTML form
  - mathematics, database entries, literary text, poems, purchase orders ….
- HTML just isn't designed for that!

# Idea: Back to the Basics

- HTML was defined using ***SGML***
  - ***S***tandard ***G***eneralized ***M***arkup ***L***anguage
  - A meta-language for defining languages.
- Complex, sophisticated, powerful
- Idea: Use ***SGML***

# Languages based on SGML



# Idea : "Webified" SGML

- New e**X**tensible **M**arkup **L**anguage: ***XML***
- Can use ***XML*** to define new languages
- ***Distributes*** easily on the Web
- Can ***mix*** different types of data together

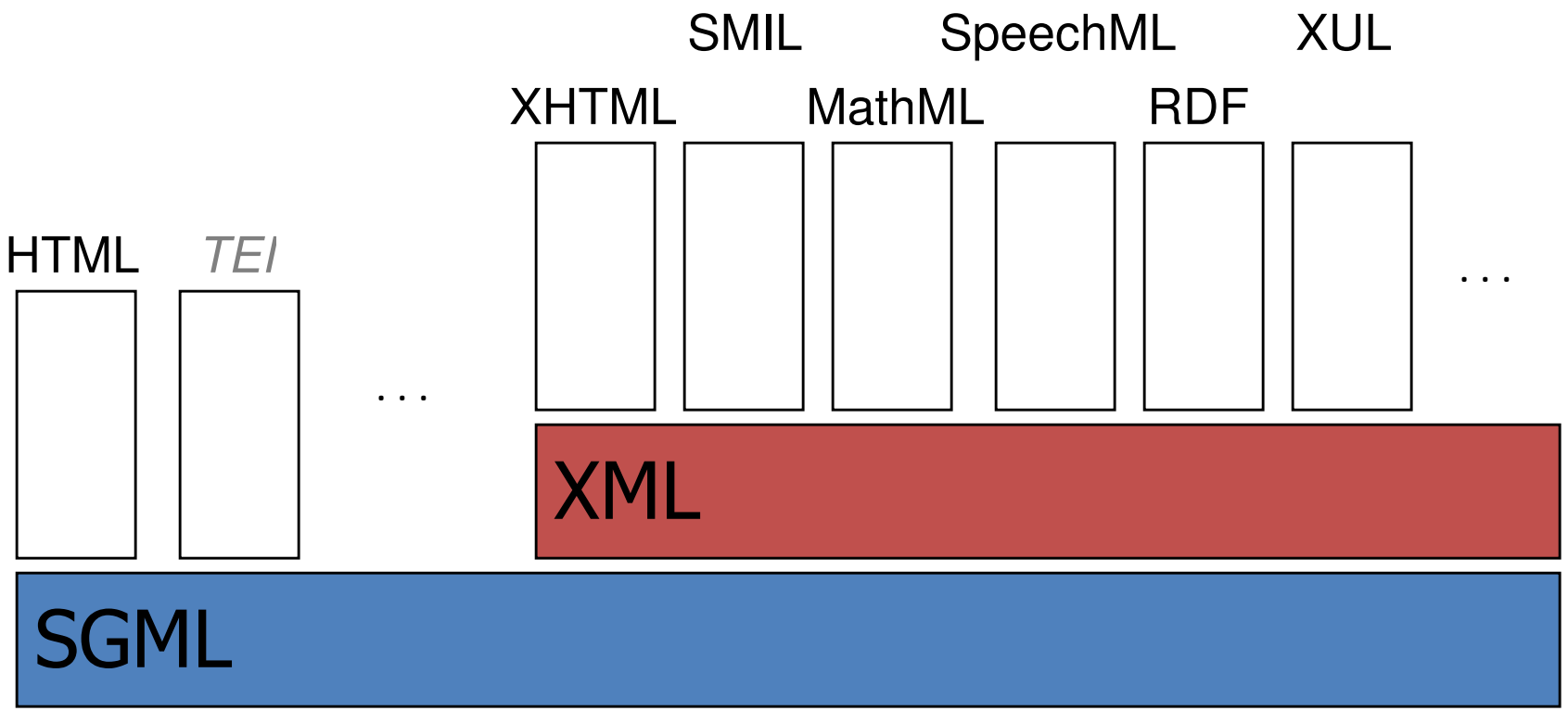
# XML Design Goals

- XML can be used straightforwardly over internet
- It shall be easy to write programs which process XML document
- XML design should be prepared quickly
- XML shall be compatible with SGML
- XML shall support a wide variety of applications
- XML documents should be human-legible and reasonably clear.

# Key features of XML

- Extensibility
- Media and Presentation independence
  - Separation of contents from presentation
- Structured
- Validation

# The XML Family Tree




***MathML*** -- for mathematics, ***SMIL*** -- for synchronised multimedia
***RDF*** -- for describing "things", ***XUL*** -- for describing the Navigator 5 user interface

# XML Pieces

- There are 3 components for XML content:
  - XML document
  - XML Schema or DTD (Document Type Declaration)
  - XSL (Extensible Stylesheet Language)
- The XML Schema/DTD and XSL do not need to be present in all cases

# A well-formed XML document

- elements have an open and close tag, unless it is an empty element
- attribute values are quoted
- if a tag is an empty element, it has a closing / before the end of the tag
- open and close tags are nested correctly
- there are no isolated mark-up characters in the text (i.e. < > & ]]>)
- if there is no DTD, all attributes are of type CDATA by default

# XML Extensibility

- XML is Meta-markup language
- Define your own markup language (tags) for your own problem domain
- Tags can be more than formatting
- Tags can be anything
  - Semantic data representation
  - Business Rules
  - Data Relationship
- Many Domain specific markup language
  - Portable across various domain
    - Healthcare and insurance
    - Medicine and Chemical

# XML Example

- <Employee>
  - <Name>
    - <FirstName> John </FirstName>
    - <LastName> Doe </LastName>
  - </Name>
  - <EmployeeID> 1234 </EmployeeID>
- </Employee>
- There are two kinds of information in the example
  - Markup – like <Employee>, <FirstName> tags
  - Text or Character Data -- like John, Doe, 1234

# XML Processing: XML Validation

- XML data is "constrained" by a Rule
  - Employee data has to have Name and Employee ID elements
  - Name has to have both FirstName and LastName
- <Employee>
  - <Name>
    - <FirstName> John </FirstName>
    - <LastName> Doe </LastName>
  - </Name>
  - <EmployeeID> 1234 </EmployeeID>
- </Employee>

# XML Processing: XML Schema

- Defines Syntax
  - Structure
  - Vocabulary
- XML Document + XML schema work together
  - XML document alone has to be well formed
  - XML schema checks validity of XML document
- Earlier DTD was used to represent XML document structure

# Parts of an XML Document

- The DTD / Schema
- Elements
- Attributes
- General entities
- Character references

- Comments
- Marked sections
- Processing instructions
- Notations
- Identifiers and catalogs

# XML basics : XML Declaration

- <? xml
  version= “version_number”
  encoding = “encoding_declaration”
  standalone = “standalone_status”
  ?>

Attributes

- The XML declaration (also called XML Prolog)
  - not required, but typically used. If it is present in XML document, it must be placed as the first line in XML document. <?xml> is written in lower case
  - attributes include:

(a) version – specifies the version of the XML standard used

(b) encoding – specifies the character encoding used in the document. UTF-8 is the default encoding used.

(c) standalone – It informs the parser whether the document relies on the information from an external source, such as external DTD for its content. The default value is set to “No”. Setting it to “Yes” tells the processor that there are no external declarations required for parsing the document.

Example : <?xml version=”1.0” encoding=”UTF-8”>

<?xml version=”1.0” standalone=”yes”>

# XML basics: Comments

- <!-- Your Comment -->
  - A comment starts with <!- - and ends with -->
  - contents are ignored by the processor
  - cannot come before the XML declaration
  - cannot appear inside an element tag
  - may not include double hyphens
- Example

  <?xml version="1.0" encoding = "UTF-8" ?>

  <!- - Students Marks are Uploaded -->

# XML basics : Tags and Elements

- An XML file (document) is structured by several XML-elements (also called XML tags).
- The names of XML elements are enclosed in triangular brackets < >
  - Example : <Employee>
- The names of XML elements are case sensitive.
  - Example: <contact-info> is different from <Contact-Info>

# XML basics: Rules for XML Elements

- Each XML element needs to be closed either with start or with end elements
  - Example:
    - <student> …. </student>
    - <student/> -- Also called empty element or empty tag
- An XML element can contain multiple XML elements as its children.
  - <Employee>
    - <Name>
      - <FirstName> John </FirstName>
      - <LastName> Doe </LastName>
    - </Name>
    - <EmployeeID> 1234 </EmployeeID>
  - </Employee>
- Elements in XML document should not overlap
  - Example
    - <class> <teacher> Ricky </class> </teacher> ---- Incorrect
    - <class> <teacher> Ricky </teacher> </class> ---- Correct

# XML basics: Elements

- An XML element is made up of a start tag, an end tag, and data in between.
- Example:

  <director> Matthew Dunn </director>
- Example of another element with the same value:

  <actor> Matthew Dunn </actor>
- XML tags are case-sensitive:

  <CITY> <City> <city>
- XML can abbreviate empty elements, for example:

<married> </married> can be abbreviated to

<married/>

# XML basics : Root Element

- An XML document must have only one root element.
- The root element encloses all other elements
- Example:
  - <message> is the root element and <to>, <from> and <subject> are child elements.
    - <?xml version="1.0" encoding="UTF-8"?>
      - <message>
        - » <to> Jim </to>
        - » <from> Jayesh </from>
        - » <subject> Message from Teacher> </subject>
      - </message>

# XML basics : Namespace

- Namespace
  - not mandatory, but useful in giving uniqueness to elements and attributes in an XML document.
  - Helps to avoid element collision (in case XML document contain identical names for elements and attributes, but different definitions)
  - The namespace is identified by URI (Uniform Resource Identifier).
  - The namespace is declared using reserved attributes (such as “xmlns”)
  - Syntax is : <element xmlns:name = “URI”>
    - where word name is Namespace prefix and URI is Namespace identifier.
  - Example
    - <cont:contact  xmlns:cont = www.abc.com/profile>
      - Here, the prefix is cont and URI is www.abc.com/profile.  This means, the element names and attribute names with the cont prefix, all belong to the namespace www.abc.com/profile.
      - each element tag can be divided into 2 parts – *namespace*:*tag name*

# XML basics : Namespace

- You can define one Namespace as the default namespace. The default Namespace is declared in the root element.
- The default Namespace is used in XML document to save you from using prefixes in all child elements. There is no need to use a prefix in default Namespace.
- A default namespace is considered to apply to all unqualified elements ( elements with no prefix) in the document.
- Default Namespace apply to elements only and not to attributes.
- If the URI reference in default Namespace declaration is empty then it suggests no default namespace is in use.

# XML basics : Namespace

- Example of Default Namespace
  - <tutorials xmlns="www.java.com/tutorial">
    - <tutorial>
      - <title> Java Tutorial </title>
      - <author> ABCD </author>
    - </tutorial>
  - </tutorials>

# XML basics : Attributes

- Attributes are part of XML elements.
- An element can have multiple unique attributes.
- Attribute gives more information about XML elements.
- Attribute define properties of elements
- An attribute is a name-value pair separated by an equal sign (=).
- Example:

  <City ZIP="94608"> Emeryville </City>
- Reserved attributes are

  - xml:lang
  - xml:space

# XML basics: Elements Vs Attributes

- Data can be stored in child elements or in attributes. For ex.
  - <person gender = “female”>
    - <firstname> Anna </firstname>
    - <lastname> Smith </lastname>
  - </person>
  - <person>
    - <gender = “female>
    - <firstname> Anna </firstname>
    - <lastname> Smith </lastname>
  - </person>

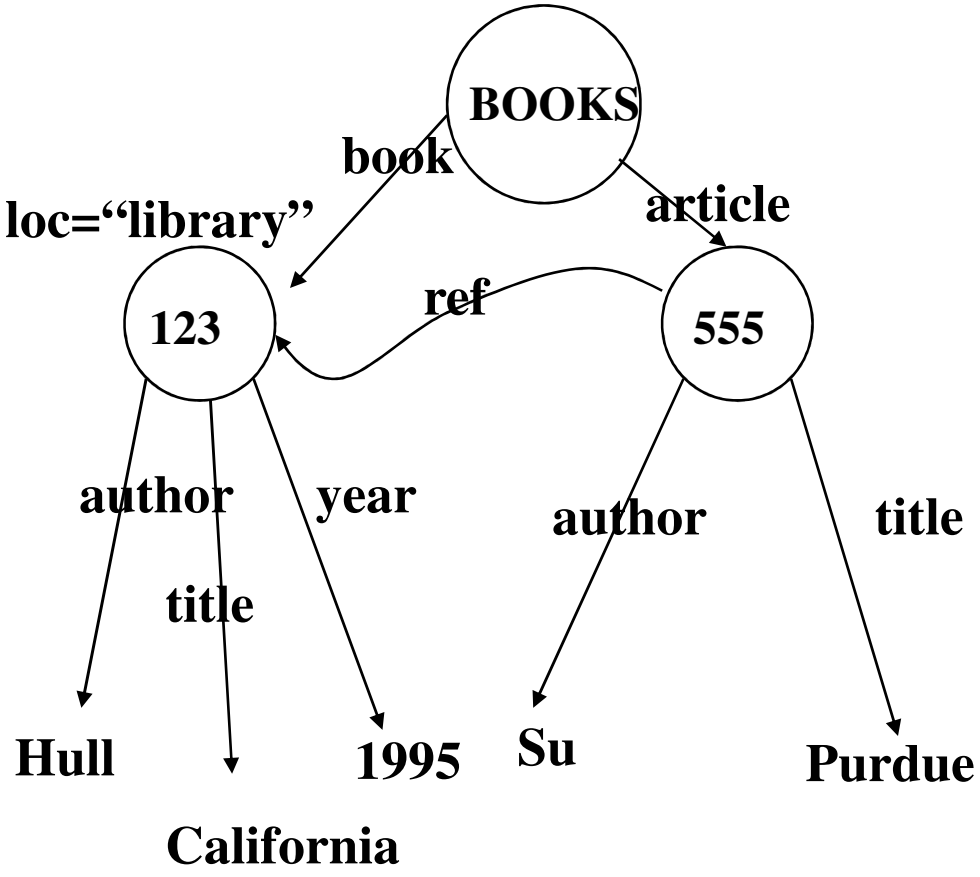
Here, gender is an attribute of person element

Here, gender is a child element of person element

There are no rules about when to use attributes and when to use child elements. Generally, one should use child elements if the information is data

# XML Data Model: Example

```
<BOOKS>
<book id="123"
    loc="library">
  <author>Hull</author>
  <title>California</title>
  <year> 1995 </year>
</book>
<article id="555" ref="123">
  <author>Su</author>
  <title> Purdue</title>
</article>
</BOOKS>
```



# Authoring XML Documents

- Authoring guidelines:
  - All elements must have an end tag.
  - All elements must be cleanly nested (overlapping elements are not allowed).
  - All attribute values must be enclosed in quotation marks.
  - Each document must have a unique first element, the root node.

# Motivations of XML Schema

- Provide more powerful and flexible schema language than DTD
- Represent XML document syntax in XML language
- Support non-textual data types
  - B2B, e-Commerce
- Handle complex syntax

# Document Processing

- Valid V/s Schema-Valid
  - XML schema is not part of XML 1.0
  - XML document that is validated with DTD is “Valid”
  - XML document that conforms to XML schema is “Schema-valid”
  - XML document that conforms to a particular XML schema is called “instance document” of that schema
- Definition and Declaration
  - Definition: Create new types (elements)
  - Declaration: Enable elements and attributes with specific names and types to appear in XML document instances.