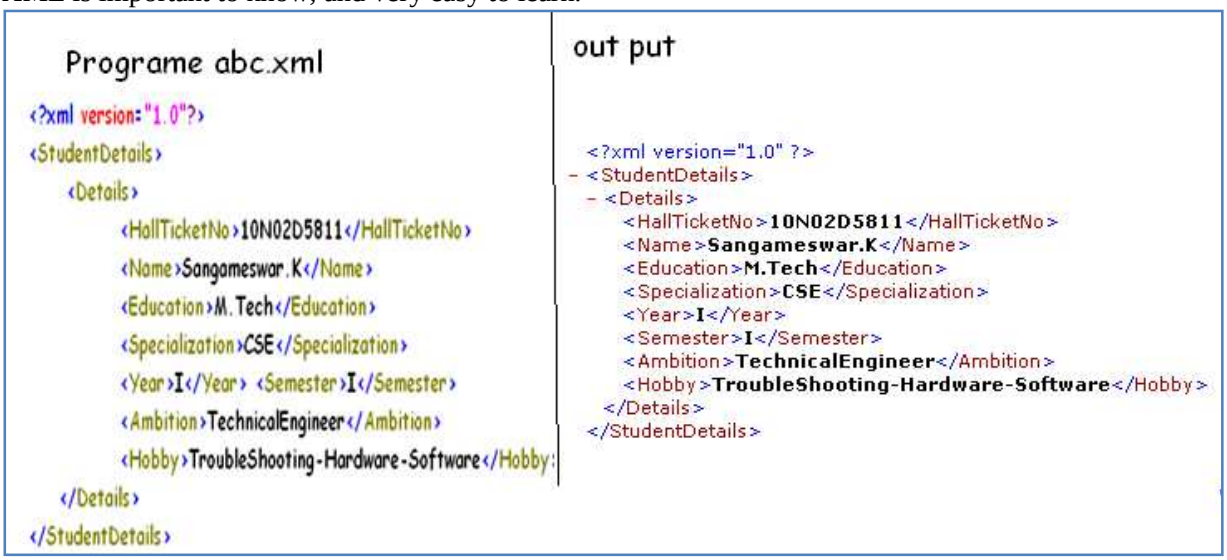
XML Tutorial

XML stands for Extensible Markup Language.
XML was designed to transport and store data.
XML is important to know, and very easy to learn.



XML Document Example

Introduction to XML

XML was designed to transport and store data.

What is XML?

1. XML stands for EXtensible Markup Language
2. XML is a markup language much like HTML
3. XML was designed to carry data, not to display data
4. XML tags are not predefined. You must define your own tags
5. XML is designed to be self-descriptive
6. XML is a W3C Recommendation

The Difference between XML and HTML

XML is not a replacement for HTML.
XML and HTML were designed with different goals:

1. XML was designed to transport and store data, with focus on what data is.
2. HTML was designed to display data, with focus on how data looks.
3. HTML is about displaying information, while XML is about carrying information.
4. XML is used in many aspects of web development, often to simplify data storage and sharing.

XML Tree

XML documents form a tree structure that starts at "the root" and branches to "the leaves".

An Example XML Document

XML documents use a self-describing and simple syntax:
The first line is the XML declaration. It defines the XML version (1.0) and the encoding used (ISO-8859-1 = Latin-1/West European character set).
The next line describes the **root element** of the document (like saying: "this document is a note"):

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<note>
 <to>Tove</to>
 <from>Jani</from>
 <heading>Reminder</heading>
 <body>Don't forget me this weekend!</body>
</note>
```

<note>
The next 4 lines describe 4 **child elements** of the root (to, from, heading, and body):
<to>Tove</to>
<from>Jani</from>
<heading>Reminder</heading>

<body>Don't forget me this weekend!</body>
And finally the last line defines the end of the root element:
</note>

XML Syntax Rules
The syntax rules of XML are very simple and logical. The rules are easy to learn, and easy to use.

1. All XML Elements Must Have a Closing Tag
2. XML Tags are Case Sensitive
3. XML Elements Must be Properly Nested
4. XML Documents Must Have a Root Element
5. XML Attribute Values Must be quoted

```
<note date=12/11/2007>
  <to>Tove</to>
  <from>Jani</from>
</note>
<note date="12/11/2007">
  <to>Tove</to>
  <from>Jani</from>
</note>
```

Entity References
Some characters have a special meaning in XML.
There are 5 predefined entity references in XML:

&lt;	<	less than
&gt;	>	greater than
&amp;	&	ampersand
&apos;	'	apostrophe
&quot;	"	quotation mark

Comments in XML
The syntax for writing comments in XML is similar to that of HTML.
<!-- This is a comment -->

White-space is preserved in XML
HTML truncates multiple white-space characters to one single white-space:

HTML:	Hello my name is Tove
Output:	Hello my name is Tove.

With XML, the white-space in a document is not truncated.

XML Naming Rules
XML elements must follow these naming rules:

1. Names can contain letters, numbers, and other characters
2. Names cannot start with a number or punctuation character
3. Names cannot start with the letters xml (or XML, or Xml, etc)
4. Names cannot contain spaces

XML Attributes
XML elements can have attributes in the start tag, just like HTML.
Attributes provide additional information about elements.

XML Attributes

1. XML Attributes Must be quoted
2. **Example Programe**

```
<person gender="female">
  <firstname>Anna</firstname>
  <lastname>Smith</lastname>
</person>
<person>
  <gender>female</gender>
  <firstname>Anna</firstname>
  <lastname>Smith</lastname>
</person>
```

.

Goals of XML:

1. XML should support many number of applications
2. Xml should be directly usable over internet
3. Xml should easier to write program
4. Xml documents should be easily and clearly and understandable by humans
5. The design of Xml document should be faster to generate
6. Minimum importance should be given to xml documents

XML Validation

1. XML with correct syntax is "Well Formed" XML.
2. XML validated against a DTD is "Valid" XML.

Valid XML Documents

A "Valid" XML document is a "Well Formed" XML document, which also conforms to the rules of a Document Type Definition (DTD):

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<!DOCTYPE note SYSTEM "Note.dtd">
<note>
<to>Tove</to>
<from>Jani</from>
<heading>Reminder</heading>
<body>Don't forget me this weekend!</body>
</note>
```

The DOCTYPE declaration in the example above, is a reference to an external DTD file. The content of the file is shown in the paragraph below.

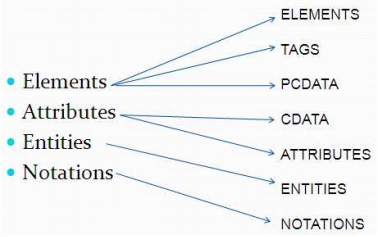
XML DTD

The purpose of a DTD is to define the structure of an XML document. It defines the structure with a list of legal elements:

```
<!DOCTYPE note
[
<!ELEMENT note
(to,from,heading,body)>
<!ELEMENT to (#PCDATA)>
<!ELEMENT from (#PCDATA)>
<!ELEMENT heading (#PCDATA)>
<!ELEMENT body (#PCDATA)>
]>
```

DTD= **Document type Definition**

Building Blocks from DTD View



ELEMENTS:

- Elements are the main building blocks of XML.
- Elements can contain text, other elements or be empty.
- Examples:
    - To
    - From
    - Header
    - Message

Tags: tags are used to markup elements

A starting tag like <element_name> marks up the beginning of an element and an ending tag <element_name> marks up the end of an element.

Example:

<message>message information</message>

PCDATA

1. PCDATA means parsed character data.
2. PCDATA is text that will be parsed by parser.
3. PCDATA is used with ELEMENTS

ATTRIBUTES

- Attributes provide extra information about elements

"Attributes are always placed inside the starting tag of an element. Attributes always comes in name/value pair"

Example:

<img src="logo.gif">

Attribute = value

src = logo.gif

CDATA

1. CDATA means Character Data
2. CDATA is text that will NOT be parsed by a parser.

ENTITIES

- Entities are variables used to define common text.
- XML entities must be used for special characters

&lt;	<	less than
&gt;	>	greater than
&amp;	&	ampersand
&apos;	'	apostrophe
&quot;	"	quotation mark

TYPES OF DTD

DTD are of two types based on the location of the dtd file They are

- Internal DTD(inside the document)
- External DTD(outside the document)

Internal DTD

Specifying the DTD within the document is referred as internal DTD

- General Structure:

```
<!DOCTYPE root-element [
<! ELEMENT root-element(sub-elements)>
 <! ELEMENT sub-element(data-type)>
                                    ]>
```

External DTD

Specifying the DTD outside the document is referred as External DTD

General Structure

```
<!ELEMENT root-element(sub-element)>
<!ELEMENT sub-element(data-type)>
```

saved in "ext.dtd"

Accessed in document as

```
<!DOCTYPE root-element SYSTEM "ext.dtd">      OR
<!DOCTYPE root-element PUBLIC "URL/ext.dtd">
```

XML Schema

- XML Schema is an XML based alternative to DTD.
- XML Schema specifies the structure of an XML document and constraints on its content.
- XML Schema language is also referred to as XML Schema Definition(XSD)
- XML Schemas are Microsoft's alternative to DTDs.

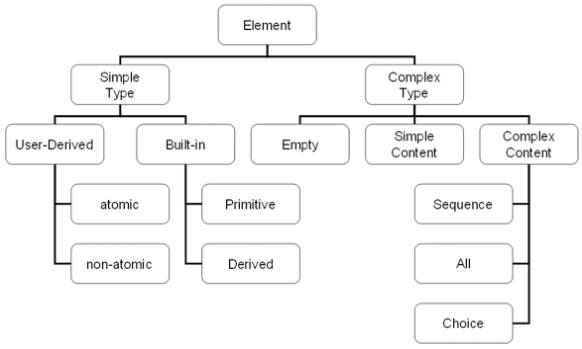
```
<xs:element name="note">
<xs:complexType>
 <xs:sequence>
  <xs:element name="to" type="xs:string"/>
  <xs:element name="from" type="xs:string"/>
  <xs:element name="heading"
type="xs:string"/>
  <xs:element name="body" type="xs:string"/>
 </xs:sequence>
</xs:complexType>
</xs:element>
```

filename=abc.xsd

What is an XML Schema?

- The purpose of an XML Schema is to define the legal building blocks of an XML document, just like a DTD.
  - An XML Schema:
    - Defines elements that can appear in a document
    - Defines attributes that can appear in a document
    - Defines which elements are child elements
    - Defines the order of child elements
    - Defines the number of child elements
    - Defines whether an element is empty or can include text
    - Defines data types for elements and attributes
    - Defines default and fixed values for elements and attributes

Schema Elements



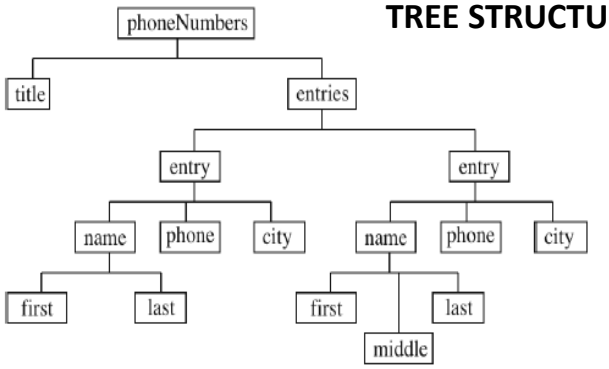
XML-Data Types;

1. String
2. Date
3. Numeric
4. Boolean

XML-Parsers

1. DOM=Document object modeling
2. SAX=Simple API (Application Programming Interface)

TREE STRUCTUR



WHAT IS DOM?

- The DOM is an Application Program Interface (API) for XML documents.
- API is a set of data items and operations which can be used by developers of application programs.
- The DOM API specifies the logical structure of XML documents and the ways in which they can be accessed and manipulated.
- The DOM API is just a specification

DOM (Document Object Modeling)

1. It stores the entire xml document into memory before parsing
2. It occupies more memory
3. We can insert delete a node
4. Traversing in any direction
5. Dom is tree based parser
6. Dom preserve comments
7. Dom parser always serves the client application with the entire document.
8. Package are
    a. import javax.xml.parsers.*;
    b. import org.w3c.dom.*;

SAX: Simple API

1. It is used as an alternative to dom parses
2. Parser node by node
3. Does not store xml file in memory
4. We cannot insert delete a node
5. Top to bottom traversing
6. Sax is event based parser
7. Sax does not preserve comments
8. Sax parser serves the client application always only with places of the document.
9. Package are
    a. import javax.xml.parsers.*;
    b. import org.xml.sax.*;
    c. import org.xml.sax.helpers.*;

Internal DTD Example Program

```
<?xml version="1.0"?>
<!DOCTYPE book[
<!ELEMENT catalogue (book+)>
<!ELEMENT book (title,price)>
<!ELEMENT title (#PCDATA)>
<!ELEMENT price (#PCDATA)>
]>
<catalogue>
<book><title>xml</title><price>500</price></book></catalogue>
```

XML With CSS:

```
<?xml version="1.0"?>
<!DOCTYPE book[
<!ELEMENT catalogue (book+)>
<!ELEMENT book (title,price)>
<!ELEMENT title (#PCDATA)>
<!ELEMENT price (#PCDATA)>
]><?xml-stylesheet type="text/css" href="a.css"?>
<catalogue>
<book>
<title>xml</title>
<price>100</price>
</book></catalogue>
```

a.css file is =

```
book { background-color:red; color:cyan}
```