Database Management System
Prof. Dr. S. Srinath
Department of Computer Science Engineering
Indian Institute of Technology, Madras

Lecture No. # 38
XML – Introductory Concepts

(Refer Slide Time: 01:20)

Hello and welcome, in the previous two lectures, we have seen at different kinds of databases, or try to see what happens when we try to manage data, that not easily amenable to storage in a relational database. And in this context we looked at object oriented databases where data that has stored or complex data object in the sense that, not only abstract the structure of some data item but also its behavior, what kind of behavior. In this lecture and next three lectures, we are going to look at another important kind of data management issues that that occur in practice day in and day out. I have we are which can be in a natural kind of captured as combined data management issues related to unstructured, semi-structured data and heterogeneous data sets and what are called as self describing data sets. See, what happens in the real world is the set there are rather than having one specific database, even in specific, even in a given U R D or even in a given system context like let us say we have a huge company. Let us say huge multinational company which spans several countries or even not even not so huge company which let us say

spans several cities. They do not use one single database for the entire company; it is quite impractical for several reasons, for historical and for several other kinds of for practical reasons, they use many different kinds of databases and many different kinds of data stores.

As it is in what happens is over a period of time, trying to reconcile these different data stores becomes more and more difficult. And whereas it could well be the case set one part of the company is using let say windows based system, another part is after companies using is Linux based system, another part is using Mac system so on and you would not have, it would not be possible to create a single database system, which spans across all these different operating platforms in the works coherently. And in a addition, in addition to the above huge amount of unstructured or in some sense semi-structured is cre[ate] is been created day in and day out in the form of letters, faxes, memos, webpage's, emails, documents and so on and so forth. So much of data is being generated, all of which cannot be easily accessed or we it is not possible to simply define a database so that you put everything into this into this database. What is required, in some kind of simple mechanism that can that can help describe a data element in a fashion that is independent of any operating platform or any encoding, I mean there could be different encoding as well. I mean one could, one could be using ASCI encoding, one could be using unique code encoding, one could be using some other kind of encoding and so what are encoding, what you are be ah way data is being stored. It should be able to, one should be able to reconcile between all of them in uniform fashion.

Now what is how do we do that? And how do we reconcile all of these different data elements under one, under one simple in a simple fashion. For this the emerging answer over the last few years is XML or the extensible (( )) language. So we should be looking into XML and XML as applied to semi-structure data in the next in the following two lectures. And XML is although it is so simple in practice where it would seem that there is nothing to it has become a very important source of data interchange and data representation and data description, in the post internet world (( )).

(Refer Slide Time: 06:03)

Why XML and what are the, what are the reasons when or what are situation when when we can say that XML make sense, when we can use XML. Managing heterogeneous data source is very valid example, when there are several different data sources, let us say you are you are designing a data warehouse and you are taking transactional data from different (( )) data bases or let us say you are trying to reconcile data bit from different different files. Where let say you have different spread sheet or word document or some other kinds of files and you are able, you are planning to reconcile all these data that is stored each of these different data bases and integrate them into one common data source, that is when XML comes really comes into (( )).

And similarly data self-description, what if there is no, what if you do not know or what if whoever is using your data does not know how your data is organized. Let us say your your employee record, what if whoever is requesting for your employee record does not know what are all the fields that is your employee record contains and what are the constraints on those fields and so on. What if there is no single way by which employee record can be organized at all. May be everybody has a different way of organizing employee information, one might say we require social security number as the primary key and in cases where in places where there is no such concept as social security number you could say something like PAN number and or ration card number whatever our just name and so on. So what all should go into some places have first name, last name, middle name, initial and so on and in some other places we just say

name. So, there may not be a single way by which we describe data about any specific entity. So the based way to manage such situations is data self-description. What your data that you are sending across between application in different context, let the data itself describe, in what way it is organize. Let the data in the meta data, that is data about data be sent in one packet that is be integrated in a way that just by looking at this packet of information we can not only get the data but also the way in which data is organized.

And the third major use of XML is semi-structured data management. I mean several different kinds of data sources are not really structured documents for example documents, letters or faxes and so on. Where there is some resemblance there some resemblance of structures for example, letter should have from and to and some kind of addressing like dear sir and whatever and subject and complementary closing and so on, but beyond that text itself or what goes into each of them, there is no specific rules that says this is way the data has to be organized. So such kinds of data bases or data sources are semi-structured data sources. And huge supply of semi-structured data is of course the World Wide Web, I mean any HTML data that is that is written that is present over the or that is served over the World Wide Web is semi-structured data, that is there is no specific structure to HTML document, however part of the HTML document is some sense describes let say suppose you say h one and slash h one, it means there this is the first header, at the first level header semi-structured in a HTML document.

Similarly, you say p and slash denote that the paragraph is started here and it is ends here and so on. So there some sentence of structuring, but there is no rigid structure in that exist for the for any HTML document. And to manage such data sources or manage such data XML becomes very important.

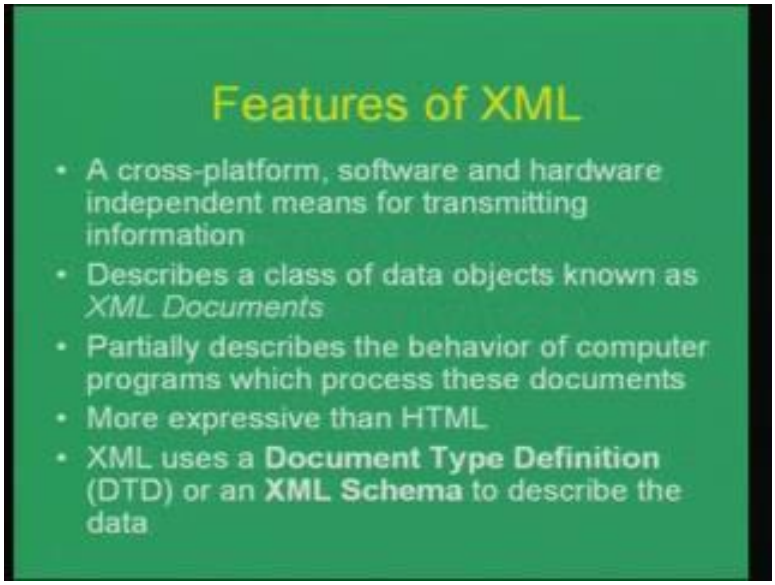
(Refer Slide Time: 10:42)

So, what is XML? What its properties? XML are the extensible markup language. It is actually a subject of an earlier markup language called SGML – standard generalized markup language. What do you understand by the term markup language? What is it mean by the term markup language? I am sure; you would have probably used the well-known HTML or the hypertext markup language in creating web pages. Even though there are number of web creation tools that are available today, most of us would have tried our handset working straight with plain vanilla html; that is go straight to HTML and start changing the HTML document directly open it up in notepad or (( )) or some text editor like that and open change it directly.

HTML is the markup language. What is the markup language? Markup language is essentially, the name used for languages where the Meta data that goes into describing the data is embedded within the data itself. That is a HTML document, the Meta data that is says which is the first level header, which is the second level header, which is the paragraph, which is the hyper link and where does the hyper link connect it and so on and so forth. All these Meta data is embedded as part of the data itself and it is embedded in this same form, that is using the same encoding and the same even the same font I mean if it is rendered by any text editors, editors are its rendered using the same font as them data itself is rendered. So therefore if you are using unique code as your base then the entire set of data and meta data would follow unique code.

And let us said earlier XML is actually a derivative of the lesser known SGML. The idea behind here XML although, XML is has become so famous and in the post internet era. The idea XML for creates the internet itself and it tends from SGML which was the standard that was used for document management, how to how to manage documents within how to manage data and Meta data within documents. So there is several SGML based tools and ensure and SGML and related to for example, if you use Linux I am sure, you might have used this software called info which is, which is based markup kind of language where which in some sense predates the World Wide Web in terms of hyper text usage within a single machine. So, it was originally designed to be a flexible text formatting, text format for electronic publishing. However, today XML plays a major role in data exchange seamless data exchange over through web and several different other application areas. In fact, the de facto standard of exchanging application data over the web is fast becoming XML. Of course there are other standards, but XML because of its simplicity is faster (( )) or taking over all other different standards of data exchange over the web.

(Refer Slide Time: 14:30)

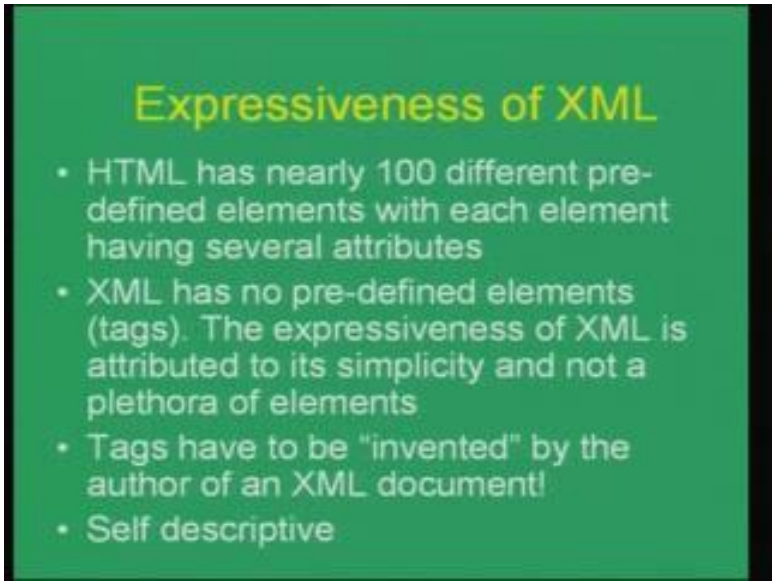


So what are, what are some other features of XML? XML is essentially a cross-platform, entity which is independent of any software and hardware means for transmitting information. That is it does not the way in which an XML document is stored is not dependent upon what software you are using, for example, whether you are working in windows or Linux, and even hardware, what hardware you are using – that you are using an Intel based pc, or Macintosh having Motorola

based, Macintosh powered by Motorola processor and so on. So, it does not really change, I mean across all of these platforms XML the structure and description of an XML document would remain the same. An XML basically describe data in the form of one or more XML documents, where each XML documents is in the form of some kind of tree containing both data and Meta data. And XML is a markup language, but it is not a computer language; that is not a procedure language.

However, there are it also, there are some kinds of computer program namely especially say XSLT which read XML documents and behave in different ways accordingly. For example, if we have C program or Perl program and you run it through the Perl interpreter, the Perl interpreter behaves in different ways based on ah what is written in the Perl program. So, in a sense, so in that sense one can call XML as a programming language; however it is not a complete programming language. In that sense, it does not have all construct the makeup the programming language and hence it is just a markup language. And XML can, can also use what is called as DTD or document type definition which is again being first replaced by what is called as an XML schema to actually describe the structure in which data has to be organized or to describe the data itself.

(Refer Slide Time: 16:57)



So, what is XML contain? If you worked in HTML, where HTML is a markup language for creating hypertext documents. You would probably come across several different deserved tags that HTML itself provides, for example we talked about h one. So suppose you (( )) h one tags within (( )). It means that whatever lies within this tags is the, is the first level header in the HTML document. H one is for example, it is an example of reserve tag in HTML. Similarly, the tag called p defines the paragraph, p and slash p defines the paragraph, so again p is the reserved tag in HTML. Like that HTML has nearly hundred different pre-defined elements only using which you can define a HTML document.

On the other hand, XML has no pre-defined elements, that is there is no h there is no h one, h two, h three there is no P, there is no PR, there is no UL, there is no LI any such element that is defined by, that is defined by XML. In fact, it is the ownership or it is the responsibility of the creation a document to creative to actually invent tags or invent Meta data that describes the data. And it is completely up to the user or up to the creator of the XML document to describe or to find have to find what may be termed as the appropriate Meta data terms in order to define the data that is available. And that that is what makes an XML document self-descriptive.

We will see how that happens in a short while. But before that let us look at the, look at what are the, what are the motivation behind XML. What is the meant to do? Let us take a contrasting feature to begin with, HTML for example. HTML was basically meant to meant to describe how data have to be displayed by a browser. It is a logical data description language but nevertheless it is meant for display; HTML is meant for display; that is it tells the browser how to render the data. It tells the browser, for example, this is the first level header so do whatever you are doing in order to describe first level header. This is the paragraph break, so do whatever you are doing to display a paragraph break ok or this is bold so this is do whatever you are doing to display bold characters and so on. So primarily HTML is design for display or logically describes describing how data are to be rendered on a browser.

(Refer Slide Time: 20:12)

In contrast XML is designed to describe data and focus on what the data really is, so essentially XML is make to was oriented towards data exchange rather the data rendering. And therefore XML isn't really a replacement for a HTML. In fact XML is complement of HTML. And there are several tools that given an XML document will parse an XML document and convert it to appropriate HTML document, so that it can be rendered by a browser. So XML is basically meant for describing the data, while HTML is meant for describing how the data are to be rendered.

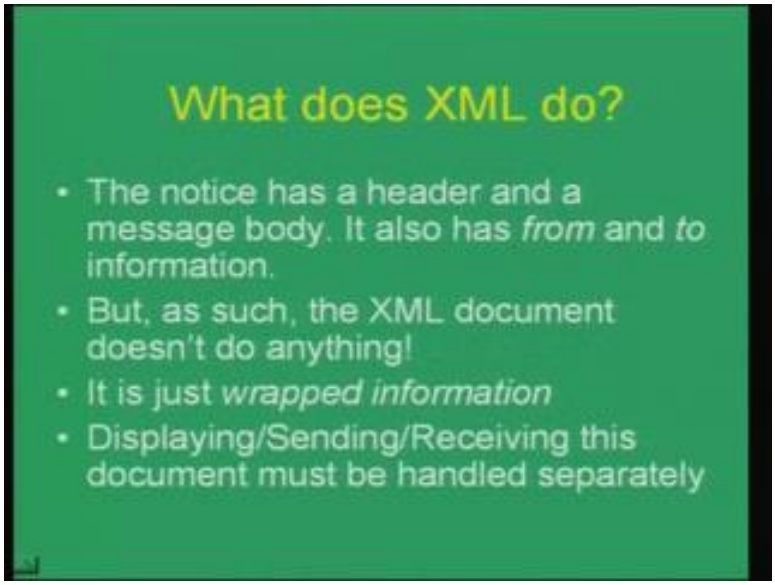
(Refer Slide Time: 21:07)

So what is an XML do? Have look at the slide that that is shown here. This slide shows us small XML fragment. So, look at how the fragment looks like. There is a startup of tag here, which looks like there is startup of tag here, which looks like any HTML tag in the sense that it is it is embedded within angular braces. So and just like HTML tag tags end with the slash tag. So notice tag ends with the slash notice. The to tag ends with slash to and from tag ends with slash from. However as you see here, each of these or arbitrary, I mean just it is notice and it is to and from and so on. So essentially what is fragments is describing is that this is the notice where the notice begins here and ends here. The notice is send to the students of first year from hostel warden with the heading air conditioner, with the body of the notice as whatever that is written here right. So body is denoted between body and slash body and heading is denoted between heading and slash heading. Now you might ask a how to any browser note how to interpret to and from and heading and body and so on. The simple answer is it does not. And it is the responsibility of the parser or whoever writes the parser to also write what have to be done when the parser come, comes across and notice or comes across the two field in a notice and so on ok.

Otherwise if an XML document is rendered in a generic browser like say internet explorer as so. If this particular fragment to be rendered in a internet explorer, you would just get a tree like structure as you can see this is tree like structure, this is a hierarchy, that is at the first level there is there is notice comma slash notice ok which is the first level element and there are one, two,

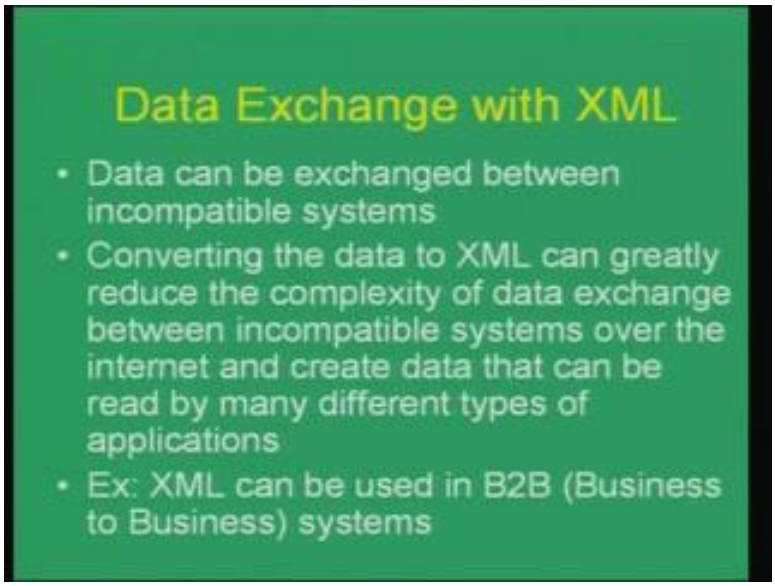
three and four second level element. So it is a tree comprising the two level of hierarchy one node at the top denoting notice, and four nodes below it or four children denoting to, from, heading and body.

(Refer Slide Time: 23:35)



So as we saw that the XML fragment denoted something called notice and which has from, and to and message body and so on. But, as such, the XML document itself does not do anything; it is just describes that. So it just say notice, slash notice and from and to, it beyond that nothing else. It does not do anything else. It is just some kind of what may be term this wrapped information, that is information in this sense I'm using the term information to mean Meta data. So Meta data is wrapped around data elements in the XML document. And how to display this XML documents, how to send, how to receive, how to what actions to perform in response to each of these elements all of them have to be handled separately. XML by itself does not describe any (( )).

(Refer Slide Time: 24:37)



Now what is the use of creating such a one might ask two generic a module for creating data, creating describing data. What is the use of such an element or such a mechanism. Firstly, as you can see regardless of what kind of data that you are describing XML is pure text; it is pure everything is in textual form, only that the encoding may differ obviously, but it is possible to find out what is the encoding and detect what is encoding and change it accordingly. But whatever it is, it is given an encoding it is just plain text; there has no preparatory forms by which data within the data attributes or values within the XML within the XML document is described. So, it is just plain (( )) data. So, because it is just textual data, it can be exchange seamlessly between different systems. So, whether it is different operating platforms like Linux, or windows, or Unix or Solaris whatever or different processors itself different hardware platforms, it does not matter.

So, over the web or over any web an XML document can be exchange seamlessly without any need for any kind of interfacing for XML document. But one which still ask that huge amount of an XML document is a huge waste of space, in the sense that just to describe a small notice and and to say this the notice was send from someone and send to someone and with the heading so and so, unit two at lot of tagging information like to and slash, to and from, slash from and so on. However, that is not too much of a problem, because today storage is for cheaper than what it was earlier. And storage is much more cheaper than what it was earlier and disk can disk can

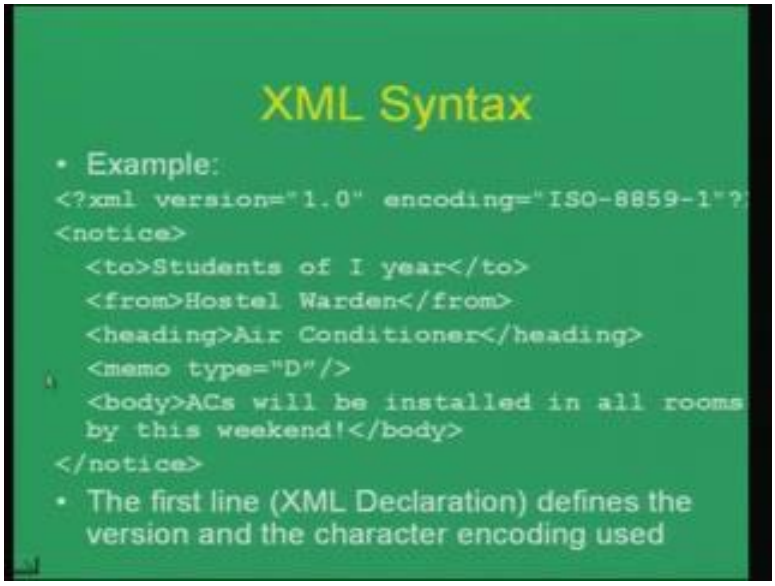
store large and larger amounts of data. And processing is also faster, so it is quite and then many numbers of many very efficient XML parses that is freely available; therefore parsing an XML document and creating a structure is quite simple. Also, there, there are different kinds of , there are different terms kinds of software that are available for compressing data you have, may have WinZip or G-zip or (( )) and compress and so on and so forth. So which, which can be used to compress textual data very efficiently; and especially with textual data contains repetition which an XML document describing some class of elements is apt to contain different kinds of repetition, one can one can expect huge percentage of compression to be performed. So even if a XML document is takes a lot of spaces for describing a set of data, it can still be stored in a efficient fashion by compressing the document and decompressing it on the fly and so on.

And converting any data set like preparatory block level data set to XML can greatly reduce the complexity of data exchange between different incompatible systems and again over the internet. And huge application area of XML naturally is in B2B or business to business interchange. So each business house or even within one single business house, there could be different standards that are used. And trying to interface between these standards become huge challenge and that is where the utility of XML comes becomes significant.

(Refer Slide Time: 28:52)

So, XML see unlike for example Ms-Word data which can be opened only by Ms-Word or postscript data which can be only opened only by postscript viewers or PDF data which can be opened only by PDF viewers. Unlike those XML is just text data, so it is not bound to a specific application or specific browsers. You can open XML using any application that can deal with textual data, you can open XML in notepad; you can open it in VI, you can open it in normal internet explorer, which will show, which can support XML directly and which just show as a tree structured. If a browser does not support XML directly, just shows you textual form of the XML data. And there are several different that are written, wherein you can access XML data sources as though your accessing your databases.

(Refer Slide Time: 30:01)

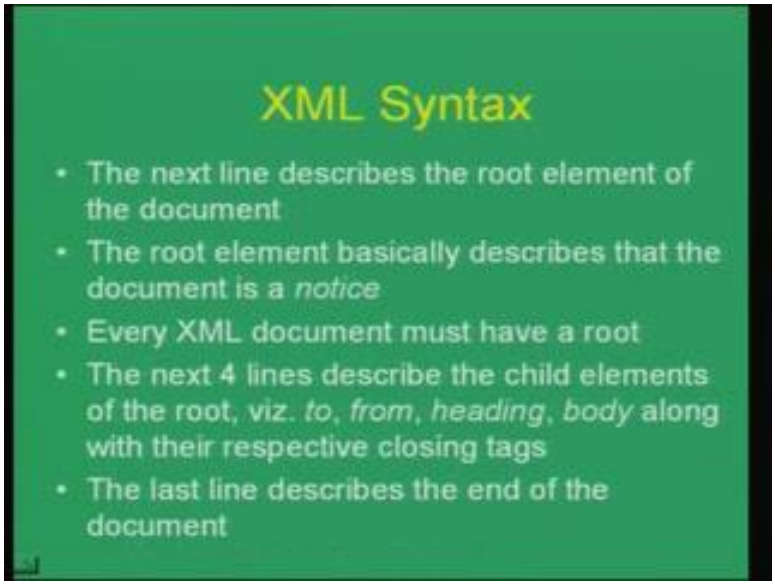


So, let us come back to XML syntax and look at some more aspects of what makes up an XML document. Now here again this is the (( )) notice notice comma slash notice however, with a few more things that are added. Now, you might have noticed already that any symbol here, any tag that is opened here has to be ended. So, notice is ended by slash notice, body is ended by slash body, to is ended by slash to. So, in XML, it is mandatory for, for properly closing every tag, which is not show in html; in html certain kinds of tags like say p and font and so on and so forth, need not be closed and whenever a second p tag comes in the the browser automatically closes the first p tag, but here an XML parser would flag syntax error if a

given tag is not closed that is it is open and it is not closed. However, there are exceptions have a look at the memo tag , memo type equal to d slash like this; so that is an other way of expressing a tag or a or a meta data that does not contain a large amount of data.

In fact, the meta data contains what is called as an attribute of name type with value D, and that is it and there is no other no other data that is associated with this tag or tag is a meta data meta data item actually. So, by default an XML document should be properly nested and properly closed that is every open tag should be properly closed, and it goes without saying that the nesting of XML element has also to be (( )) exception cases where that is closing of for the case of closing of tags. There are cases, where you can close a tag within a single line I have shown in this memo example here in addition look at the first line here, where the first line starts with one angular braces; question mark and xml, the key word called xml, if the document begins with this, with this pattern here then the XML pattern, then and if it (( )) XML parser, the XML parser knows that this is the valid XML document. So, this one defines that this is the XML document, and this one defines the version of this document, and this one defines the encoding that is used for describing this document.

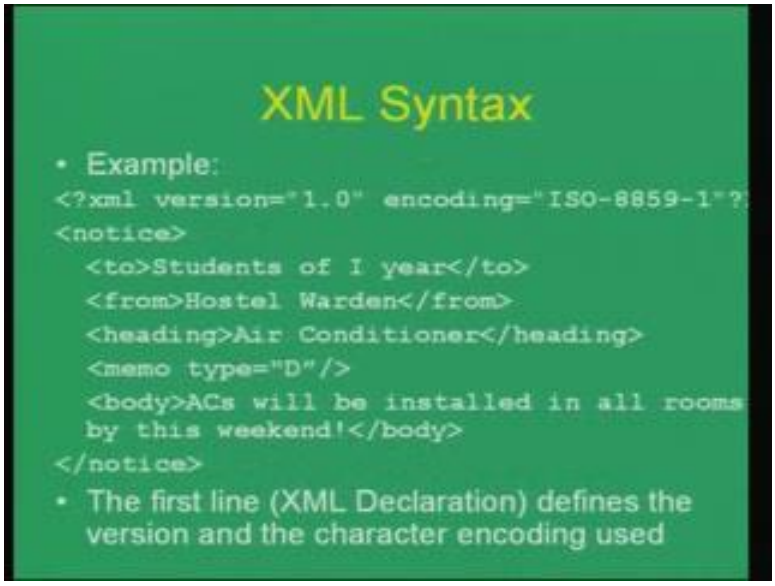
(Refer Slide Time: 32:46)



So, just to summarize what we have seen, so the first line basically is a declaration that says that this document is an XML document with the given version, and with the given version and

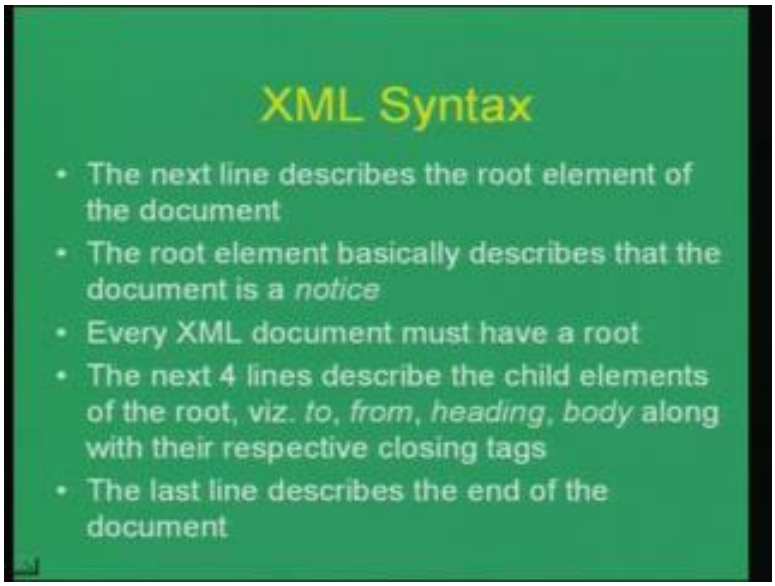
encoding and following the XML declaration is the root element of the XML document. So, so every x m l document should be a rooted tree, that is a there should be one root element that begins here and ends at the end of the document right. So, in this case the root of the XML document is the notice tag or the notice element.

(Refer Slide Time: 33:36)



A tag is again some more x m l based definitions, this is what is called as a tag notice or from or body and so on, they are called tags. A tag is a simple meta data, that is embedded within angular braces on the other hand the entire set of XML data from notice to slash notice inclusive is called an x m l element, that is it denotes an XML elements called notice, which in turn contain several different x m l elements like to element, and from element and so on right.

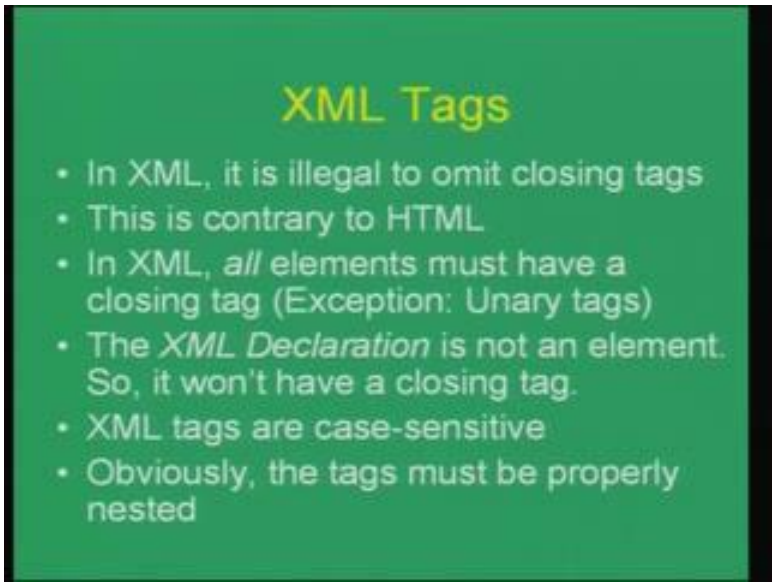
(Refer Slide Time: 34:15)



So, the root element is the one which is the biggest element, which contains the entire XML document and every XML document must have a root element, and of course the next four lines describe the child element from the root, and to from heading body and so on and the last line describes the end of the document. So, as you said before in XML, it is illegal to omit closing tags. So, every tag that is open has to be closed properly. So, notice has to be closed by slash notice and from has to be closed by slash from and so on, this is again contrary to XML.

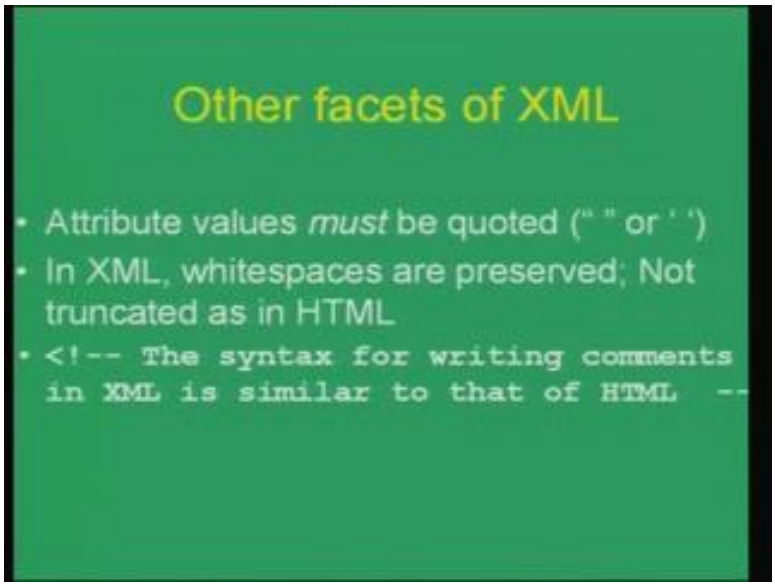
So, all elements must have a closing tag with exception of unary tags, like the memo tag that is shown in this slide here. So, where there can be a single end tag which can be closed directly.

(Refer Slide Time: 35:09)

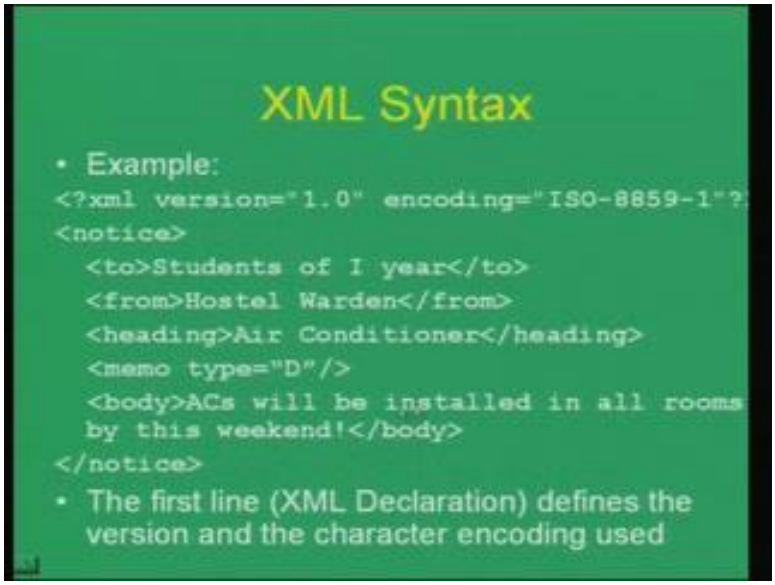


And the XML declaration, that is the first line here in this document here, this is not an element that is why to distinguish between a declaration, and meta data a declarations starts with less than and question mark. Therefore, that that gives an implicit constraint that that any name of an XML tag cannot begin with a punctuation mark, like question mark. So, because it will be treated differently, it will be parsed differently and that is not an element; therefore, it would not have a closing tag and XML tags are case sensitive; therefore, notice with all with all small letters and notice with capital n or different tags, and it goes without saying that the tags must be properly nested.

(Refer Slide Time: 36:03)



(Refer Slide Time: 36:13)

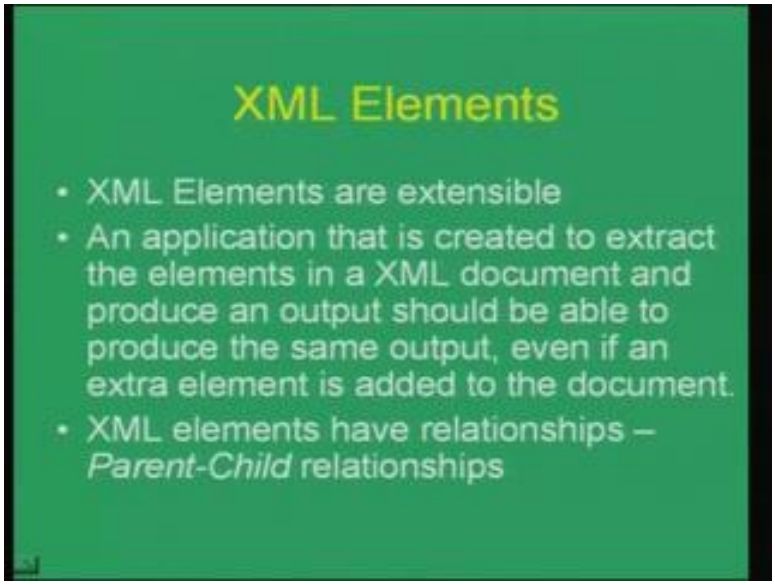


And as we saw in the previous slide tag or meta data can have different attributes. So, let us look at it again that for example here memo, memo is a Meta data which says that this is the memo, and here is an attribute which (( )) is to memo, and the attribute name is type and the value is D so… So, therefore it says that this is the memo whose which contains an attribute called the type,

and who is the value is D. So, it is type d king of memo. So, and attributes values must be coated. So, even when I say the type equal to D and D is a single letter it has to be coated it it should it should lying within double quotes. So, so that is double quotes or single quotes.

So, that is in contrast HTML, where it is optional to coat an argument attribute value unless the values contains a space with in it, and in XML white spaces are preserved and not truncated as in HTML. So, what this mean is that for example, if I put a white space here air conditioner, and three spaces and then slash heading then the data for this heading is actually air space conditioner followed by three spaces in html (( )) all the training. And leading white spaces are ignored when HTML is being ranted on a browser this is not. So, an XML and all spaces are considered, and of course the syntax for writing comments in HTML in XML with slash exclamation mark, and two dash is and end with two dashes and open close ah angular brace right. So, this is same syntax as that is used in html.

(Refer Slide Time: 38:05)



So, XML elements are given are XML document forms as specific tree structure, and the relationship between elements primarily there are exception to it which will which will look in to more detail in the next session an xml, but primarily the relationship between in between two or more tags is either parent child or siblings; that is that is one is (( )) of the other or there it, the

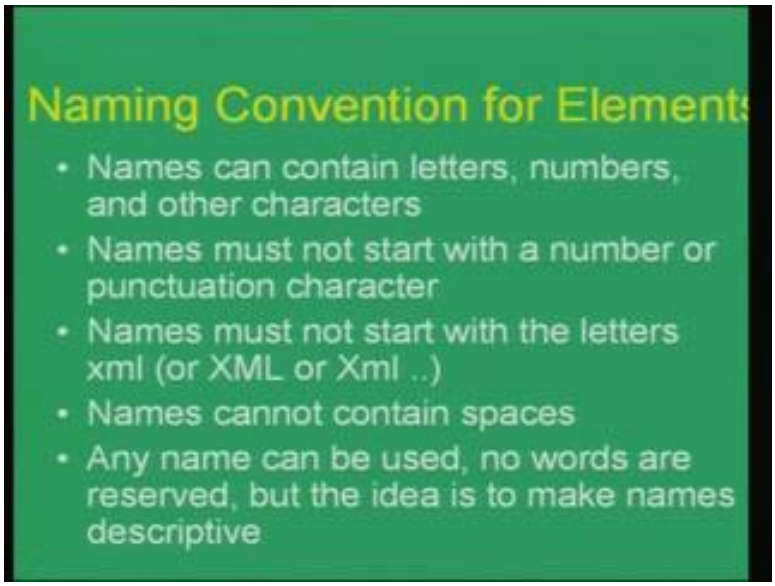
there it the same level in some sense that is or one is at a at a higher level; and one is at a lower level.

So, so it can associated level value for each tag in an XML document, and XML element as said that what is mean by an XML element a XML element is everything from starting tag to the ending tag include the tag (( )). So, so notice to slash notice including the tags forms the root element in the XML document, and an element what (( )) element contain; obviously, we have seen that an element can contain other elements for example, here the element call notice contains other element call to from heading and so on.

(Refer Slide Time: 39:45)

So, so an element can contain other elements or can have a element element or mixed content; that is it can contain both other elements. And some textual data as part of this or could contains simple content, where it is just simple textual data or an element could be empty as well which is an empty content an example of empty element, we saw was the memo element. So, which where you need not separately specify a closing tag for empty empty element, you can just close up the elements within a single line and an element also have attributes.

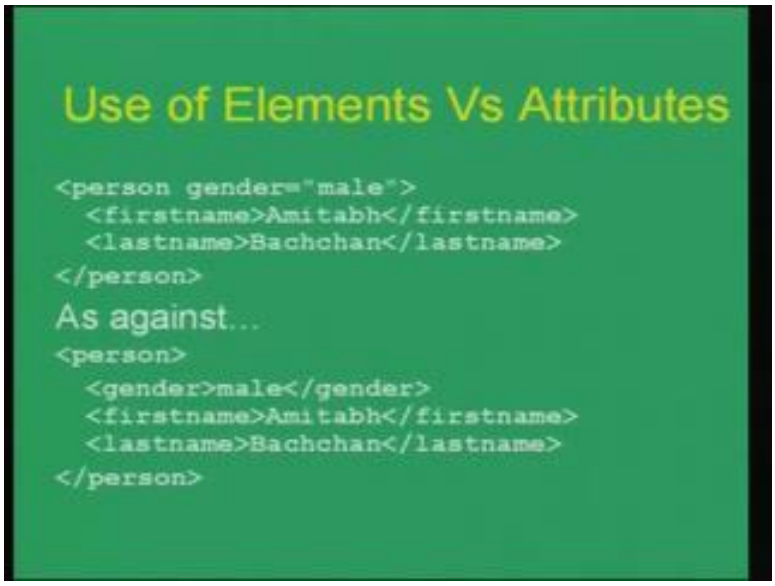
(Refer Slide Time: 40:23)



So, how can you define a XML tag or an XML, XML element what tell what are the rules for describing XML element as we saw earlier; obviously, you cannot begin an XML element with the punctuation mark like exclamation mark or question mark and so on and so forth.

The element names, names of an element can contain letters numbers, and other kinds of characters like dot colon and so on; however, they cannot start with number are a punctuation character, and also they cannot start with letters XML in any form that is case (( )) form a xml, because they a are reserved for a possible future that use and names cannot contain spaces. So, you cannot quote name within double quotes in say this, this is the XML element. So, an XML element tag should be describe by a single word and any name can be used, because there is no such thing as a reserved word except for XML of course, right. So, and you should try to make the names as descriptive as possible in order to describe the meta data.

(Refer Slide Time: 41:40)

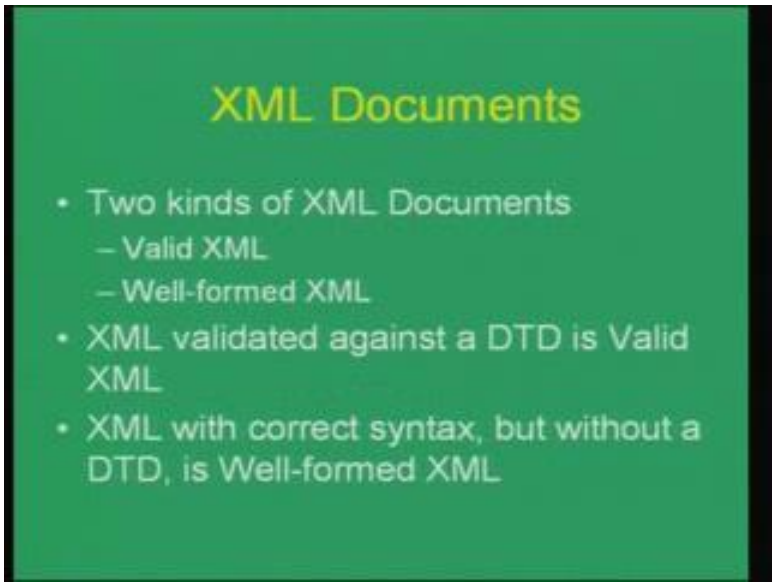


And this slide show an example of how attribute can be used as children element and vice versa. So, the in the first example here, there is an element call person and person which slash person.

And the person element has an attribute (( )) gender, whose value is male and there is the first name, and last name elements which which are all as shown. And in the second example this general equal to male attribute becomes another element here, that is it becomes an element for general and the value is male. Now, which is preferable and when this is similar to the problem that we faced in e r modeling are entity relationship modeling in one of our best classes. For example, if we were asking question whether an employee an employee working in a department should it be shown as department id attribute for the employee entity or an employee working with department is a relationship, now which you are going to show depends upon of course, depends upon this specific situation, but rulers (( )) one needs to describe an XML element just like one, it is to describe an entity only for those concepts that have independent existence; that is ah a person has an independent existence in a UOD regardless of whatever else there is…

However, general is closely associated with a person. So, it is does not have a separate independent entities. So, so that can help you decided, when the user particular data element as as in attribute versus as another child element of an existing XML element,

(Refer Slide Time: 43:47)



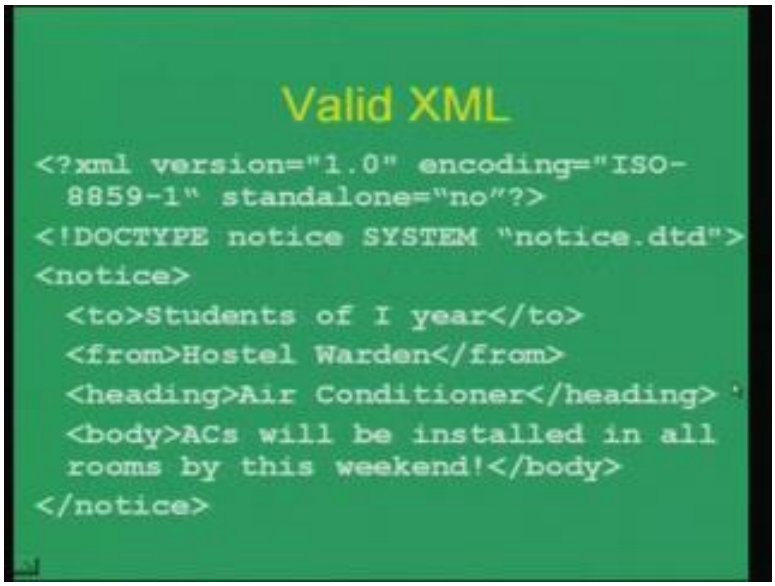
When we talked about XML document, there is often a distinction between what is called as a valid XML document, and the well form XML document the distinction between the two is is very important. Mainly when when we note that well formed XML document (( )) invalid where, whereas all valid documents are well formed. So, well formed XML document simply is ah a XML document that that conforms to all the XML syntax specification, that is what are the XML syntax specifications namely that a XML document should be a rooted tree that was a it has it has a root element, and elements do not have spaces in them. And every starting element has as a ending element and there elements are properly nested so on.

So, all these if an XML document conforms to all of these syntax rule, then it is called a well form XML document, but a well formed XML document need not always be valid various notion of validity come from validity. We say that an XML document is valid if it conforms to a schema speciation. So, so that… So, so it means that according to an XML schema for example, person person element has a attribute called gender, and no child element called gender in that case even though both of these are valid XML fragments; the first one both both of these are well formed XML fragment, the first one is a valid XML fragment where as the second one is not a valid XML fragment. So, varies the schema described in in an XML this schema described in what is called as DTD or a document type definition.

(Refer Slide Time: 46:00)

So, I am a look at this fragment again here, where this is well formed XML document, and then here there is an extra declaration; that is say stand alone equal to yes in the XML declaration, if this stand alone is equal to yes declaration is there. Then it is when it explicit states that the XML document here is only well formed, that is it does not have a schema associated with this is stand alone XML document. So, in a sense the stand alone equal to yes indicates that the document is well formed provided, of course that that it is well formed that is parser gives no error when parsing,

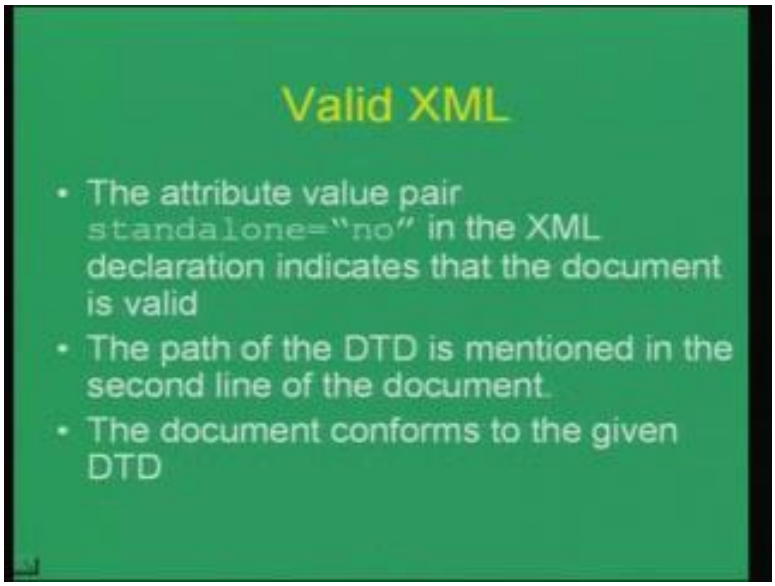
(Refer Slide Time: 46:49)



And on other hand you can say stand alone equal to no which is optional, and give up link here with with less than exclamation mark declaration doc type declaration.

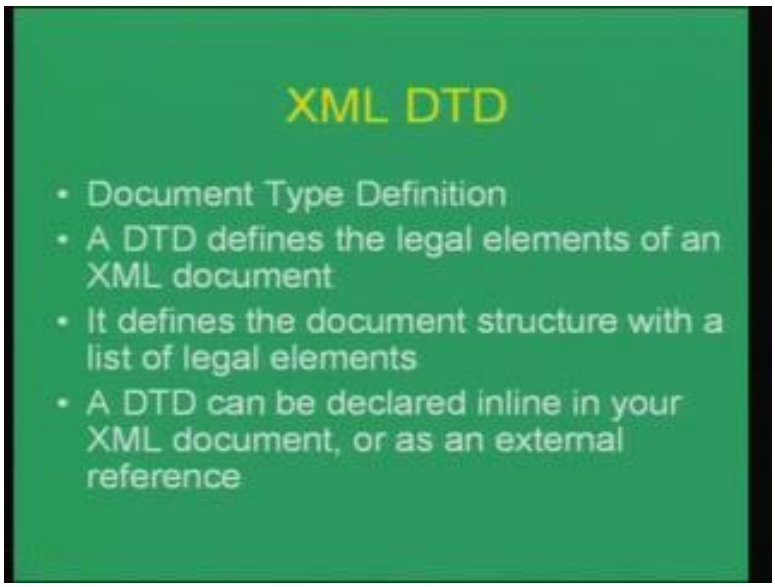
So, with the doc type declaration you can specify the DTD, which describes this XML document. So, note that it says doc type notice system notice dot DTD. So, it means that in the file call notice dot DTD the DTD for this XML fragment for this XML fragment document is available,

(Refer Slide Time: 47:26)



And the XML document are the XML fragment is valid if and only if of course first if it is well formed and it conforms to the DTD specification.

(Refer Slide Time: 47:45)



So, what is a DTD, and and how do you describe a DTD, and how do you describe an XML document using a DTD a DTD simply says that ah DTD; simply defines, what all should an

XML document contain that is ah which element should be a child of which other element and which element contains what kind of data and so on and so forth.

So, DTD stands for document type definition and DTD, DTD defines all the legal elements of a XML document, and it also defines the documents structure within a list of legal elements and DTD can be either declared inline, that is as part of an XML document or it can also be a recall from a external reference, and this external reference can be anywhere on the internet; that is it does not have to be on a on a different file or a it can be anywhere on the internet as long as you can you can dereference the DTD by a URI or a or a uniform resource indicator, where where which is which is the http equivalent slash whatever. So, so which how you specify you are (( )) document as long as you are able to dereference DTD document with url it is it is possible to place a reference in your XML document.

(Refer Slide Time: 49:17)

So, if I XML if I DTD included within the XML source then it should be wrapped with in a doc type declaration like this. So, here it is says doc type root and this is the this is the name of the root tag, and these are the element declaration what are the element that should going to this,

(Refer Slide Time: 49:35)

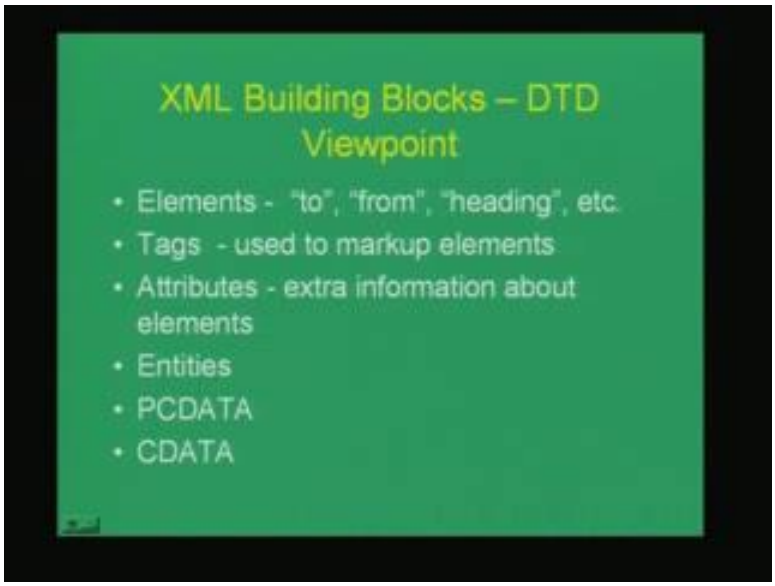
And if if the DTC DTD is external to the XML source file, then it is should be a wrapped with a doc type definition with the following syntax like this; that is doc type and root here specifies that specifies that root is the specifies the name of the root element in the in the XML document

(Refer Slide Time: 50:08)

And this is the file name of the XML document, and this says that this is the external reference, here is a example of a XML DTD as shown in this slide as you can see this is this is an inline d t

d; that means, it starts the doctype here and notice says that notice is the root element of this x m l document, and this one says that notice contains from here to here there (( )) to be a box brace here. So, or rather from here to here that is this is the definition of notice that this is the definition of the document, and the first one says element notice contains to from heading and body; that is four different elements in this particular order, that is to from heading, and body and everything else that is element to element from element heading, and element body contains what is called as pc data or hash p c data.

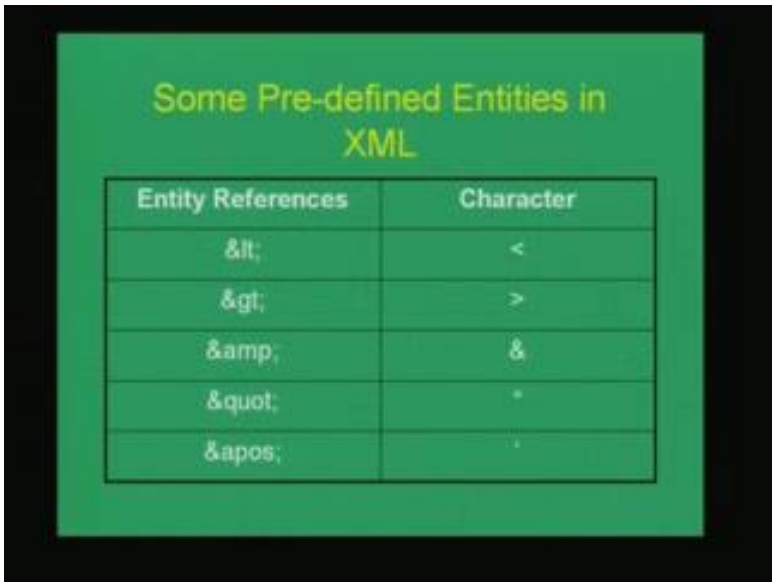
(Refer Slide Time: 51:07)



So, what is this hash p c data. So, let us go to this slide which talks about what each of this elements mean. So, elements to from heading etcetera are all this element names, and tags are used to markup elements and then an element can contain other elements like to from heading and. So, on or it can contain c data or p c data. So, what is c data or p c data. So, what is c data what is p c data. So, a p c data essentially means that parsed character data and c data just means character data.

So, so ah a parsed character data can contain other x m l tags as part of the data which which will actually be parsed and opened as and when required, but character data is not parsed it is just dumped in in water form it is present.

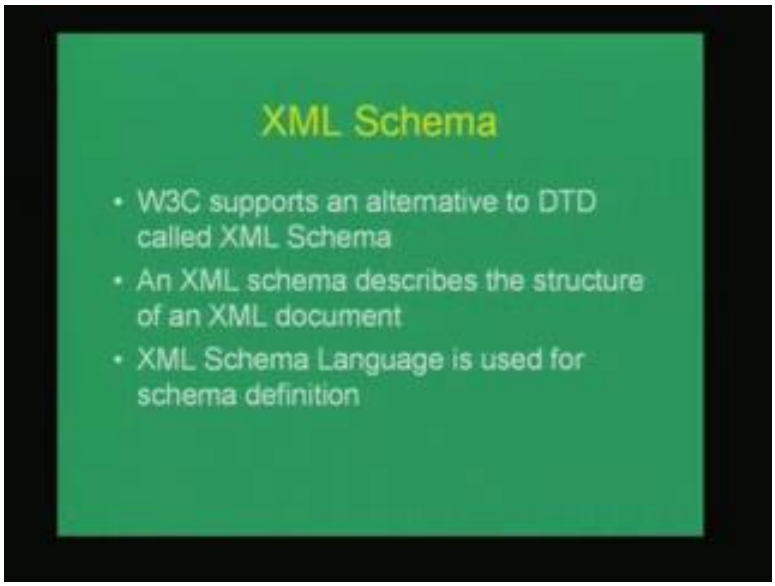
(Refer Slide Time: 52:03)



And of course, you can use this predefined entity type like ampersand l t ampersand g t like using normal h t m l to to denote each of these characters, and we talked about p c data and c data where which stands for parsed character data, and character data respectively and in in addition to each of this; there are also other kinds of option that that are provided when defining an x m l element. For example, if let us say the the option called from is if the element called from is optional in a notice a notice should just have a two address, and the from address can be optional then you can a suffix from with a question mark that is from question mark says that the the element called from may may exist either zero or one times.

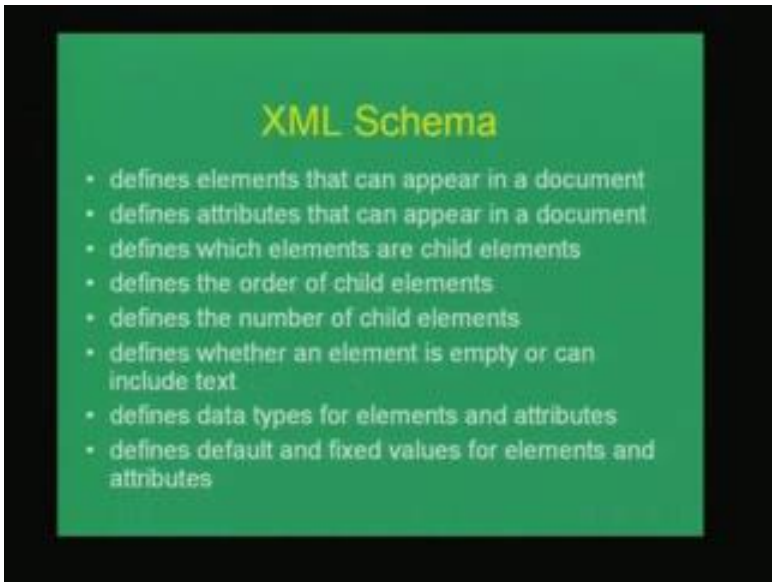
Similarly, if I put a star here, let us say heading star it means that I can have zero or more heading elements in a notice element, similarly something like body plus means that a notice can have one or more body elements and so on. So, so these are what are called as wilde card declaration (( )) elements that says how elements can be structured, and in fact one can even says something like to from, and then heading pipe body pipe is the or symbol, so which says that notice can contain to element from element and either heading or body, so one of the two because there is only one of them and so on.

(Refer Slide Time: 53:57)



So, that is the that is the simple way in which a DTD is written, and whenever an XML document fails to conforms with a DTD a validating XML parser, that is a parser which validates XML documents against a DTD will flag an error, and stop the parsing at that point now in recent times the world wide web consortium or what is called as the w three c has come up with alternative to XML DTD's, which are called as x m l schemas or sometimes also termed as x schemas XML schema is a much more detailed description of of an XML document. And it is it it is used for schema definition

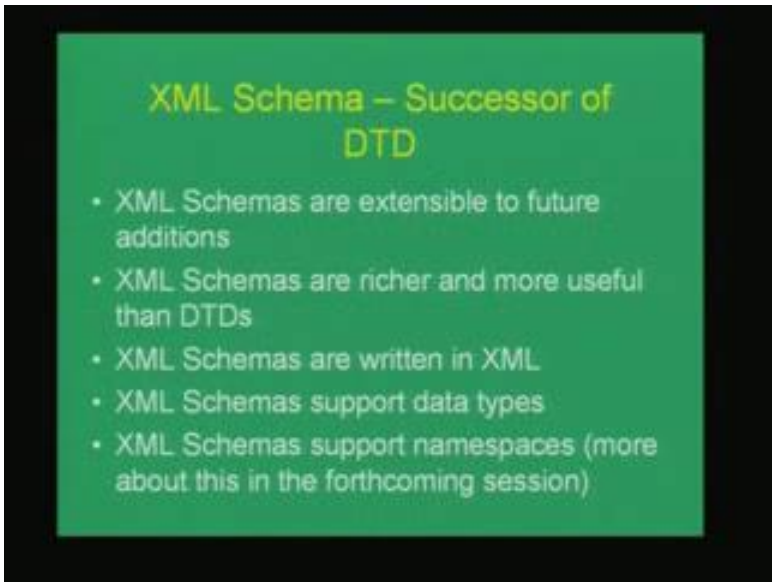
(Refer Slide Time: 54:49)



We shall not be looking into XML schema in more detail in this session due to time constraints, but but we can look at them in one of the subsequent sessions on XML documents in one of the advanced classes on managing x m l data.

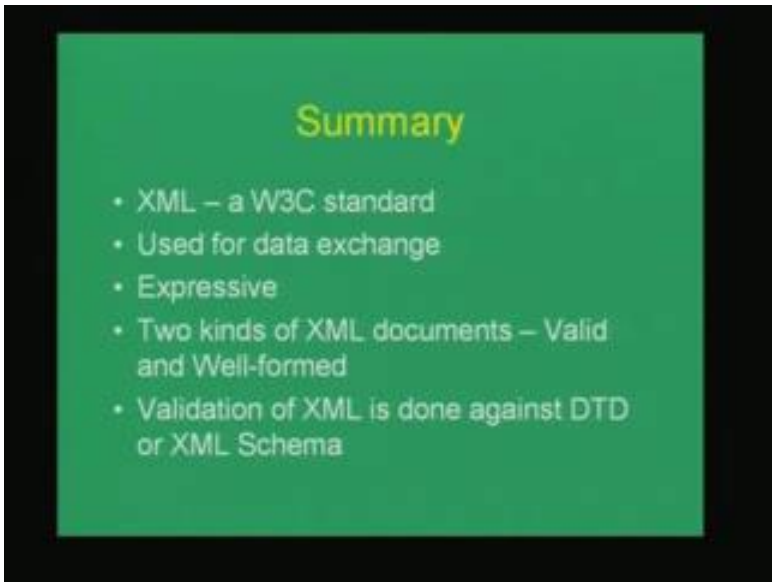
So, XML schema basically defines elements that can appear in ah in a in a document that is defines elements in attributes, which are ah which appear in a document, and like a DTD it can also it also define defines parent child relationship, and defines order in which the child elements are two appear. And it defines the number of child elements one can have and several others things that that are typically described by a DTD,

(Refer Slide Time: 55:31)



And there there for more richer and suppose it will more useful then then DTD is in described in schematic structure for XML documents. And XML schemas good thing about XML schema is that they do not have a different syntax from the XML syntax itself, that is unlike DTD's which have a different syntax in the XML syntaxes, XML schemas are written in XML itself. And they also support data types and name spaces and so on, which which we look at in more detail in one of the subsequencetion.

(Refer Slide Time: 56:12)



So, let us summaries what we have a learnt, today we have started this series of ah a few session, and XML data bases which is a world wide web constructions standard; that is used for data exchange and and managing semi structure data, and and self describing data and… So, it is a it is a self expressive, it is it is a self describing way of specifying set of data elements, and then when we talk about an XML document, it is important to distinguish between what is a valid XML document versus what it is a well formed XML document, and well formednesss all valid documents have (( )) well formed, but validity itself has to be checked again DTD's specification, and now which is being replace by the XML schema specification. So, that brings as to the end of this session.