Internet Programming CSC2233

XML

T.D.Gilmini
Department of Computer Science
University of Ruhuna

Snake Oil?

- Snake Oil is the all-curing drug these strange guys in wild-west movies sell, travelling from town to town, but visiting each town only once.
- Google: „snake oil“ xml ⇒
- **some 2000 hits**
  - XML revolutionizes software development“
  - „XML is the all-healing, world-peace inducing tool for computer processing“
  - „XML enables application portability“
  - „Forget the Web, XML is the new way to business“
  - „XML is the cure for your data exchange, information integration, data exchange, [x-2-y], [you name it] problems“
  - „XML, the Mother of all Web Application Enablers“
  - „XML has been the best invention since sliced bread”

Introduction

- XML stands for **eXtensible Markup Language.**
- XML is a markup language much like HTML.
  - Markup language for creating documents containing structured information
- XML was designed to describe data
  - Meta language that gives meaning to data that other application can use

Introduction

XML?

- XML tags are not predefined in XML.
  - Allow user defined tags
- XML is self describing
- XML uses a DTD (Document Type Definition) to formally describe the data.

- **XML is extensible:** XML allows you to create your own self-descriptive tags or language, that suits your application.
- **XML carries the data, does not present it:** XML allows you to store the data irrespective of how it will be presented.
- **XML is a public standard:** XML was developed by an organization called the World Wide Web Consortium (W3C) and is available as an open standard.
- XML Does Not Use Predefined Tags

XML Vs HTML

- 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,** XML is about **describing information**

XML is not a

- A replacement for HTML (but HTML can be generated from XML)
- A presentation format (but XML can be converted into one)
- A programming language (but it can be used with almost any language)
- A network transfer protocol (but XML may be transferred over a network)
- A database (but XML may be stored into a database)

What is markup?

- (XML is a markup language that defines a set of rules for encoding documents in a format that is both human-readable and machine-readable)
- a set of symbols that can be placed in the text of a document to define and label the parts of that document

```
<message>
  <text>Hello, world!</text>
</message>
```

Is XML a Programming Language?

- A programming language consists of grammar rules and its own vocabulary which is used to create computer programs. These programs instruct the computer to perform specific tasks.
- XML does not qualify to be a programming language as it does not perform any computation or algorithms. It is usually stored in a simple text file and is processed by special software that is capable of interpreting XML.

```
<?xml version="1.0"?>
  <contact-info>
   <name>Tanmay Patil</name>
   <company>TutorialsPoint</company>
   <phone>(011) 123-4567</phone>
</contact-info>
```

What XML does

- XML does not DO anything.
- XML is just information wrapped in tags
- Someone must write a piece of software to send, receive, store, or display it

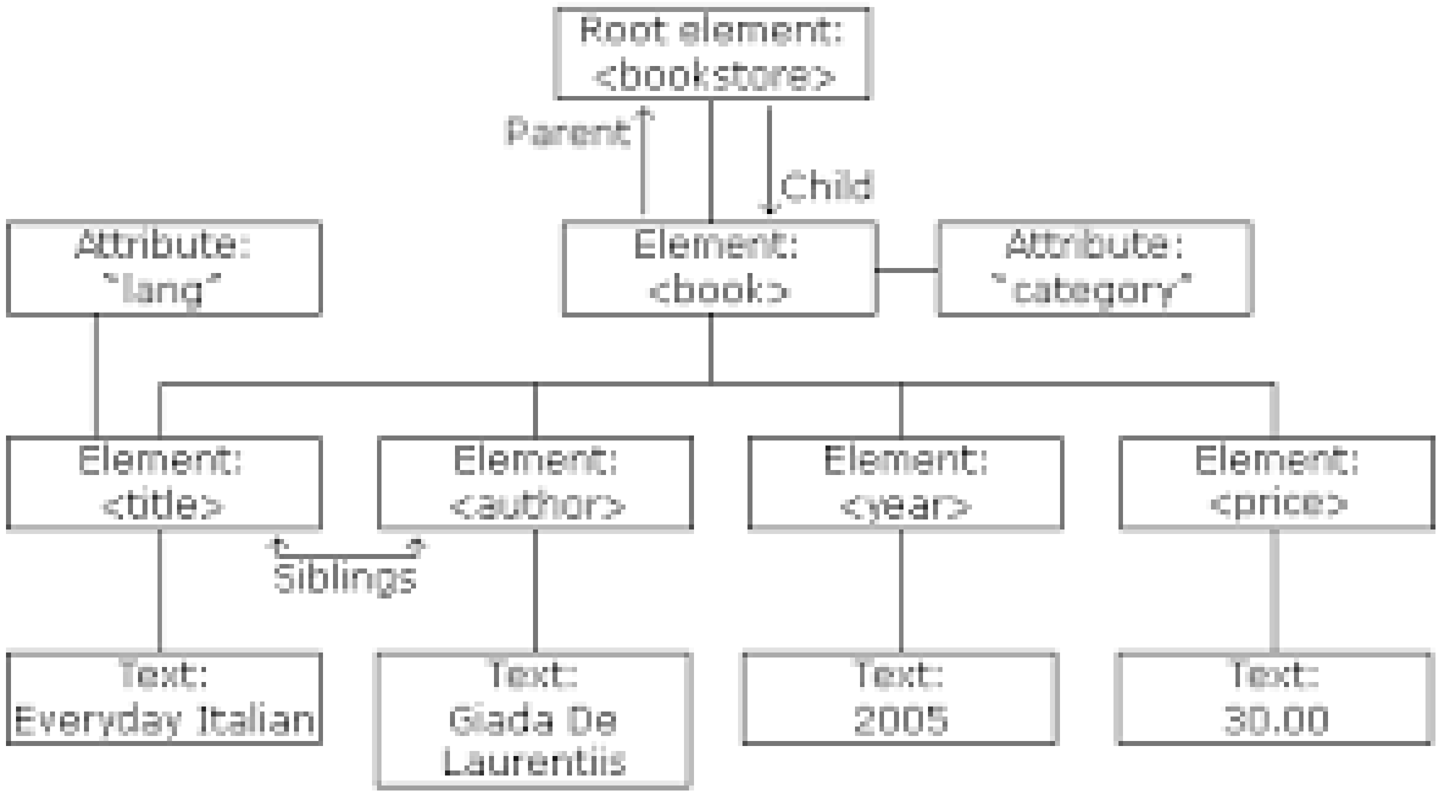
XML Tree Structure

- XML documents are formed as element trees.
- An XML tree starts at a root element and branches from the root to child elements.
- All elements can have sub elements (child elements):

-
  ```
  <root>
       <child>
         <subchild>.....</subchild>
       </child>
  </root>
  ```

XML Tree Structure

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



```
<?xml version="1.0" encoding="UTF-8"?>
<bookstore>
  <book category="cooking">
      <title lang="en">Everyday Italian</title>
      <author>Giada De Laurentiis</author>
      <year>2005</year>
      <price>30.00</price>
  </book>
</bookstore>
```

Structure of XML Document

- An XML Document Consist of
  - Prolog
  - Element
  - Attribute
  - Entity Reference
  - Comment

XML: Prolog

- Prolog is the first structural element that is presented in the xml document
- A prolog defines the XML version and the character encoding:

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

or

<?xml version="1.0" ?>

XML:Elements

- Elements are most common form of mark up
- An element can contain text, attributes, other element or a mix of the above
  - Element names are case-sensitive
  - Element names must start with a letter or underscore
  - Element names cannot start with the letters xml (or XML, or Xml, etc)
  - Element names can contain letters, digits, hyphens, underscores, and periods
  - Element names cannot contain spaces

- Empty elements can be defined in XML

<element></element> or <element />

- The next line is the root element of the document:

<bookstore>

- The next starts other element/s:

<book category="cooking">

- The next starts child element/s:

<title lang="en">Everyday Italian</title>

<author>Giada De Laurentiis</author>

<year>2005</year>

<price>30.00</price>

- The next ends elements:

</book>

XML: Attributes

- XML attributes are attached to elements
- They are name value pairs that occur inside start tags after the element names
- Must begin with a letter or underscore
- Must not contain any white spaces

```
<book category="cooking">
```

Entity References

- If you place a character like "<" inside an XML element, it will generate an error <salary>amount < 1000</salary>
- To avoid this error, replace the "<" character with an entity reference <salary>amount **&lt;** 1000</salary>

XML: Comments

- The syntax for writing comments in XML is similar to that of HTML.

<!-- This is a comment -->

- All data between those tags are ignored by the XML Processor

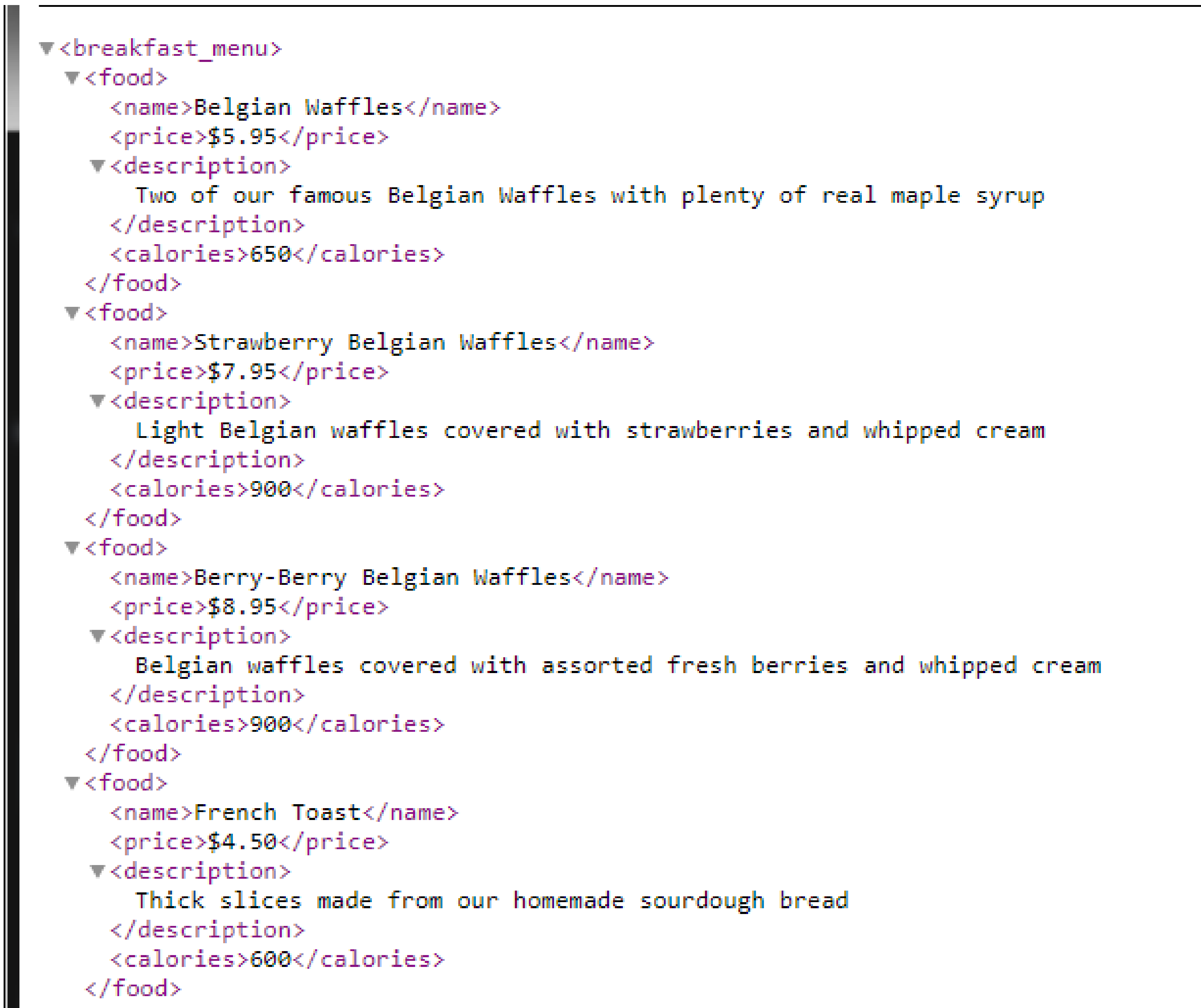
XML Syntax Rules

- The XML prolog is optional.
- If it exists, it must come first in the document.
- XML documents must contain one root element that is the parent of all other elements:
- All XML Elements Must Have a Closing Tag
- XML Tags are Case Sensitive
- XML Elements Must be Properly Nested
- XML Attribute Values Must be Quoted
- Use entity references for some characters which have a special meaning in XML.
- Two dashes in the middle of a comment are not allowed.
- White-space is Preserved in XML (HTML truncates multiple white-spaces to one single white-space)

- Write a XML document to handle the following data which shows the summery details of three degree programs

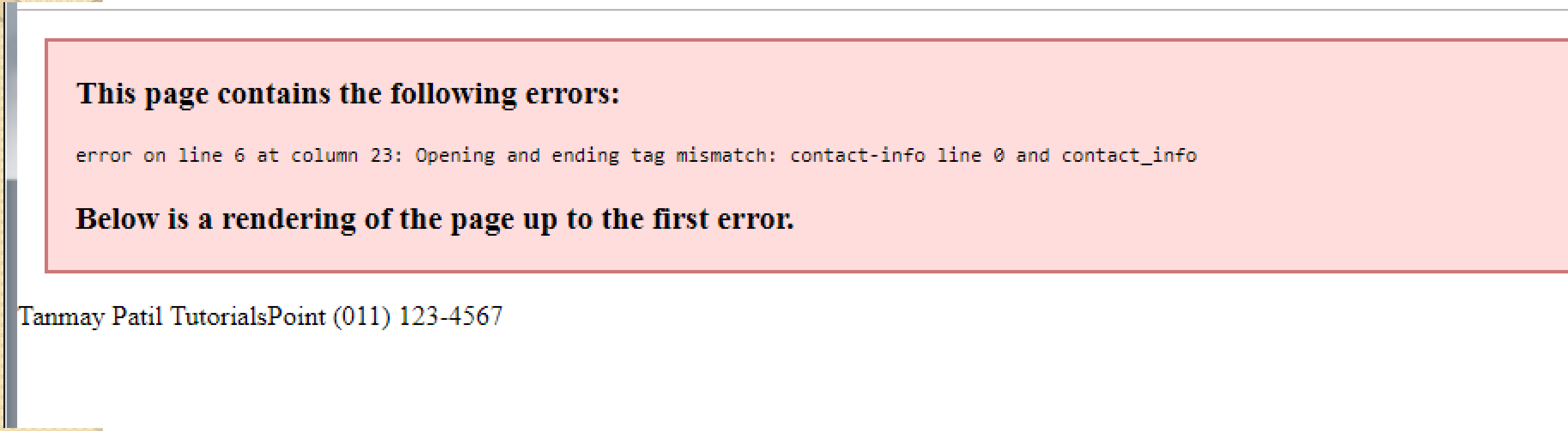
BCS		BSC		BCOM	
Academic year	2016	Academic year	2011	Academic year	2016
No: Students	45	No: Students	120	No: Students	200
Start Date	2016-Feb-02	Start Date	2011-Feb-15	Start Date	2016-Mar-30
Finished date		Finished date	2014-01-25	Finished date	Define later

```
<degree>
   <program type="BCS">
        <AcademicYear> 2016</AcademicYear>
        <NoStudent> 45</NoStudent>
         <sdate>2016-Feb-02 </sdate>
         <fdate/>
   </program>
   <program type="BSC">
        <AcademicYear> 2011</AcademicYear>
        <NoStudent> 120</NoStudent>
        <sdate> 2011-Feb-15</sdate>
        <fdate>2014-01-25</fdate>
   </program>
 </degree>
```



```
▼<breakfast_menu>
  ▼<food>
     <name>Belgian Waffles</name>
     <price>$5.95</price>
   ▼<description>
      Two of our famous Belgian Waffles with plenty of real maple syrup
     </description>
     <calories>650</calories>
   </food>
  ▼<food>
     <name>Strawberry Belgian Waffles</name>
     <price>$7.95</price>
   ▼<description>
      Light Belgian waffles covered with strawberries and whipped cream
     </description>
     <calories>900</calories>
   </food>
  ▼<food>
     <name>Berry-Berry Belgian Waffles</name>
     <price>$8.95</price>
   ▼<description>
      Belgian waffles covered with assorted fresh berries and whipped cream
     </description>
     <calories>900</calories>
   </food>
  ▼<food>
     <name>French Toast</name>
     <price>$4.50</price>
   ▼<description>
      Thick slices made from our homemade sourdough bread
     </description>
     <calories>600</calories>
   </food>
```

```
<?xml version = "1.0"?>
<contact-info>
   <name>Tanmay Patil</name>
   <company>TutorialsPoint</company>
   <phone>(011) 123-4567</phone>
</contact_info>
```



This page contains the following errors:

```
error on line 6 at column 23: Opening and ending tag mismatch: contact-info line 0 and contact_info
```

Below is a rendering of the page up to the first error.

Tanmay Patil TutorialsPoint (011) 123-4567

Viewing a XML Document

- XML files do not be displayed as HTML pages
- XML documents do not carry information about how to display the data.
- Therefore the bowser just display the XML document as it is

Character Data

- • The term CDATA is used about text data that should not be parsed by the XML parser
- An example:

```
<![CDATA
        [ If(temp<0) temp =-temp ]
 ]>
```

CDATA Section

- The predefined entities such as &lt;, &gt;, and &amp; require typing and are generally difficult to read in the markup.
- In such cases, CDATA section can be used.
- By using CDATA section, you are commanding the parser that the particular section of the document contains no markup and should be treated as regular text.

XML Parser

- Most of browsers have a built-in XML parser to access and manipulate XML
- before an XML document can be accessed, it must be loaded into an XML DOM object.
- Parser can convert text into an XML DOM object
- The XML DOM (Document Object Model) defines the properties and methods for accessing and editing XML

XML Data Models

- Data Model defines the logical structure of set of data.
- It specifies what type of data the document can contain in terms of element, attribute, comments …etc.
- Eg:-
  - DTD(Document Type Declaration)
  - XML Schema(XSD- XML Schema definition)

Document Type Declaration(DTD)

- DTD is a set of markup declarations that define a document type for an markup language (SGML, XML, HTML)
- DTD defines the legal building blocks of an XML document.
- It defines the document structure with a list of legal elements and attributes.
- DTD allows a document to send meta information to the parser about its content
- Sequence and order of tags

Document Type Declaration(DTD)

- With a DTD, independent groups of people can agree on a standard DTD for interchanging data.
- An application can use a DTD to verify that XML data is valid
- Four kinds of definitions in xml
  - Element type declaration
  - Attribute list declaration
  - Entity declaration
  - Notation declaration

- The DTD is declared in a DOCTYPE declaration beneath the XML declaration contained within an XML document:
- Inline definition

  <?xml version="1.0"?> <!DOCTYPE documentelement [definition]>
- External définition

   <?xml version="1.0"?> <!DOCTYPE documentelement SYSTEM "documentelement.dtd">

Element type declaration

- In a DTD, an element declaration defines one of the kinds of elements you can use, that is, one of the tag types.
- All element declarations have this general form: <!ELEMENT gi (content)>
- gi is the element name (also called the "generic identifier") and the content describes what content (if any) can go inside the element

Element Content Types

- Content of elements declaration in a DTD can be categorized as below −
    - Empty content
    - Element content
    - Mixed content
    - Any content

element content

- For example, the following describes the Element Type Declaration that designates the content of "product_name" as a text string:
- <!ELEMENT product_name   (#PCDATA)>
- The correct element description
  - <product_name>television</product_name>.
- <product_name><abc/></product_name> will cause an error.

Element type declaration

- Declaring empty elements
  - If you don't want a certain element to have any content, that is, you want that element always to be represented by an empty tag.

Syntax: <!ELEMENT gi (EMPTY)>

Example: <!ELEMENT pagebreak (EMPTY)>

```
<pagebreak/>
```

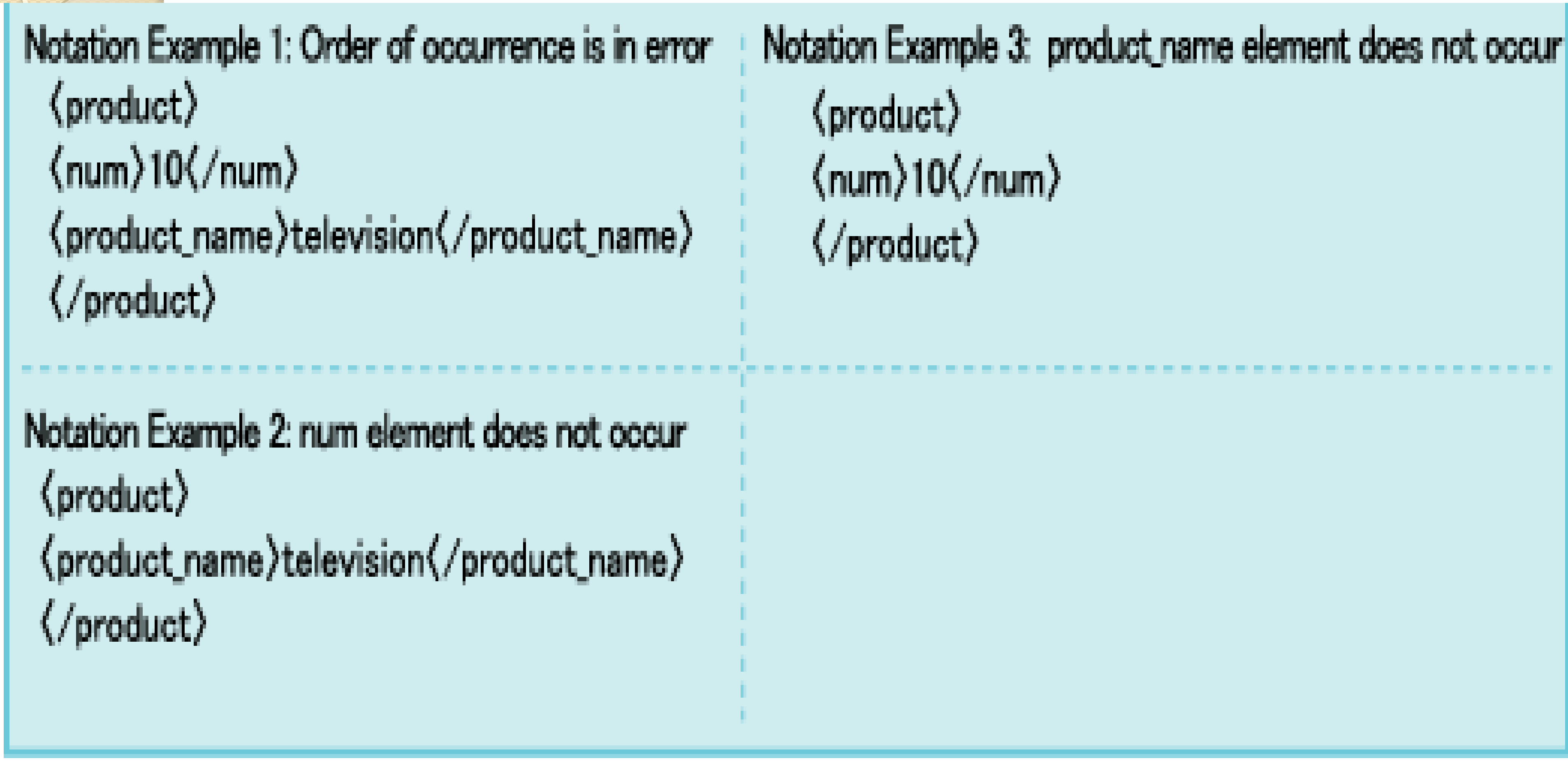
Element type declaration

- **Declaring multiple children of an element**
  - Multiple children are declared using commas (,). Commas fix the sequence in which the children are allowed to appear in the XML document
  - Example:

```
<!ELEMENT parent_name child1,child2,child3)>
<!ELEMENT child1 contents>
<!ELEMENT child2 contents>
<!ELEMENT child3 contents>
```

- <!ELEMENT product (product_name,num)>
- The following is a valid element description for this type of Element Type Declaration:

  -
    ```
    <product>
        <product_name>television</product_name>
        <num>10</num>
    </product>
    ```
- examples of invalid notation:



Notation Example 1: Order of occurrence is in error

```
〈product〉
〈num〉10〈/num〉
〈product_name〉television〈/product_name〉
〈/product〉
```

Notation Example 2: num element does not occur

```
〈product〉
〈product_name〉television〈/product_name〉
〈/product〉
```

Notation Example 3: product_name element does not occur

```
〈product〉
〈num〉10〈/num〉
〈/product〉
```

- To describe an Element Type Declaration where either the "portable" or "home" element (child elements of "tel") occurs:
- <!ELEMENT tel (portable|home)>
- In this case, the following would be an error when describing both the portable and home elements:

〈tel〉

〈portable〉 ＊＊＊＊＊＊＊＊＊＊ 〈/portable〉

〈home〉 ＊＊＊＊＊＊＊＊＊＊＊＊＊＊/home〉

/tel〉

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

- !DOCTYPE note defines that the root element of the document is note
- !ELEMENT note defines that the note element must contain the elements: "to, from, heading, body"
- !ELEMENT to defines the to element to be of type "#PCDATA"
- !ELEMENT from defines the from element to be of type "#PCDATA"
- !ELEMENT heading defines the heading element to be of type "#PCDATA"
- !ELEMENT body defines the body element to be of type "#PCDATA“
- #PCDATA means parse-able text data.

Below example demonstrates a simple example for element declaration with element content −

```
<?xml version = "1.0" encoding = "UTF-8" standalone = "yes"
   ?>
 <!DOCTYPE address
[ <!ELEMENT address (name,company,phone)>
  <!ELEMENT name (#PCDATA)>
  <!ELEMENT company (#PCDATA)>
  <!ELEMENT phone (#PCDATA)>
 ]>
 <address>
    <name>Tanmay Patil</name>
    <company>TutorialsPoint</company>
     <phone>(011) 123-4567</phone>
</address>
```

Cardinality of child elements

- The number of times a child XML Element appears within the XML Document can be controlled using the cardinality operators.

Cardinality of child elements

Operator	Cardinality	Description	Example
	1-1	By default a child element is required (it must appear once within the XML document).	<!ELEMENT address (name)>
?	0-1	Optional operator, the element does not have to appear within the XML Document	<!ELEMENT address (name?)>
*	0-n	Zero to Many operator, the element can appear 0 or more times	<!ELEMENT address (name*)>
+	1-n	One to Many operator, the element can appear 1 or more times.	<!ELEMENT address (name+)>

Mixed Element Content

- This is the combination of (#PCDATA) and children elements. PCDATA stands for parsed character data, that is, text that is not markup. Within mixed content models, text can appear by itself or it can be interspersed between elements. The rules for mixed content models are similar to the element content as discussed in the previous section.
- **Syntax**
- <!ELEMENT elementname (#PCDATA|child1|child2)*>
- **ELEMENT** is the element declaration tag.
- **elementname** is the name of the element.
- **PCDATA** is the text that is not markup. #PCDATA must come first in the mixed content declaration.

Choices of child elements

Example:

```
<?xml version="1.0" encoding="UTF-8" ?>
<!DOCTYPE address
        <!ELEMENT address (#PCDATA|name)*>
        <!ELEMENT name (#PCDATA)>
<address>
         Here's a bit of text mixed up with the
child element.
        <name>Peter</name>
</address>
```

Purchase Order XML Schema

[1] The root element is "orderform“

[2] The content of "orderform" is a "customer" element and a "product" element in that order. "customer" occurs once, and "product" may occur zero or more times.

[3] The content of "customer" is the "name", "address", and "tel" elements, each occurring once in order

[4] The content of "name" and "address" is a text string

[5] The content of "tel" is the "portable" and "home" elements, with either one or the other occurring

[6] The content of "portable" and "home" is a text string

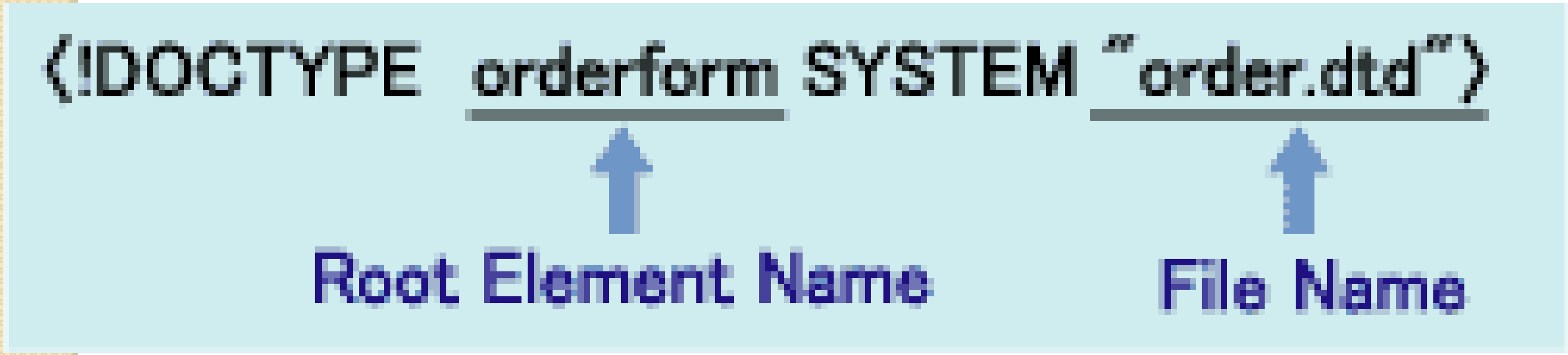
[7] The content of "product" is the "product_name" and "num" elements, each occurring once in order

[8] The content of "product_name" is a text string

[9] The content of "num" is a numeric value

- orderform.xml

```
<!DOCTYPE orderform SYSTEM "order.dtd">
<orderform>
        <customer>
                <name>Jenny</name>
        <address>Tokyo</address>
                <tel> <portable>555-5555-5555</portable> </tel>
        </customer>
        <product>
                <product_name>washing machine</product_name>
                <num>1</num>
        </product>
         <product>
                 <product_name>television</product_name>
                <num>2</num>
        </product>
</orderform>
```



```
<!DOCTYPE orderform SYSTEM "order.dtd">
```

Root Element Name
File Name

Attribute Type Definition

- If an element is to have attributes, the names and possible values of those attributes must be declared in the DTD. Here is the general form:
- <!ATTLIST ename {aname atype default} ...>
  - ename is the name of the element for which you're defining attributes, aname is the name of one of that element's possible attributes, atype describes what values it can have, and default describes whether it has a default value.

Attribute Type Definition

- The last part of the declaration, default, specifies whether the attribute can be omitted, and what value it will have if omitted.
- Example:

<!ATTLIST element-name attribute-name attribute-type attributevalue>

<!ATTLIST play title CDATA #REQUIRED>

Attribute Type Definition

- #REQUIRED
  - The attribute must always be supplied.
- #IMPLIED
  - The attribute can be omitted, and the DTD does not provide a default value.
- "value"
  - The attribute can be omitted, and the default value is the quoted string that you provide.
- #FIXED "value"
  - The attribute must be given and must have the given "value".

Type	Description
CDATA	The value is character data
(*en1*|*en2*|..)	The value must be one from an enumerated list
ID	The value is a unique id
IDREF	The value is the id of another element
IDREFS	The value is a list of other ids
NMTOKEN	The value is a valid XML name
NMTOKENS	The value is a list of valid XML names
ENTITY	The value is an entity
ENTITIES	The value is a list of entities
NOTATION	The value is a name of a notation

Default attribute Value

DTD:

<!ELEMENT square EMPTY> <!ATTLIST square width CDATA "0">

Valid XML:

<square width="100" />

Enumerated attribute value

DTD:

<!ATTLIST payment type (check|cash) "cash">

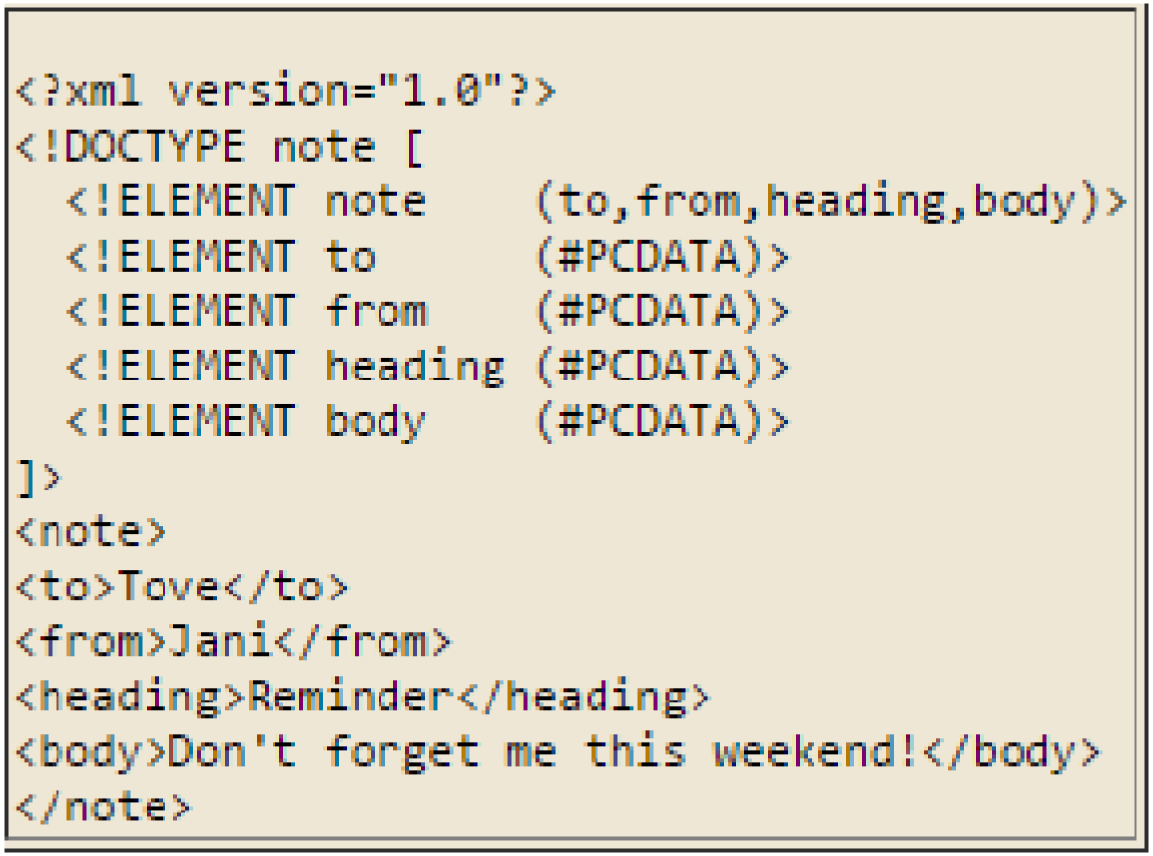
XML example:

<payment type="check" />

or

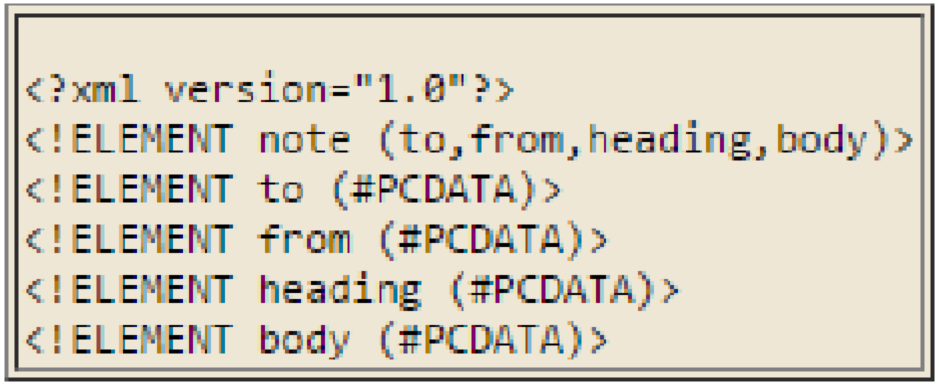
<payment type="cash" />

- A DTD can be declared inline in your XML document, or as an external reference
- Inline DTD



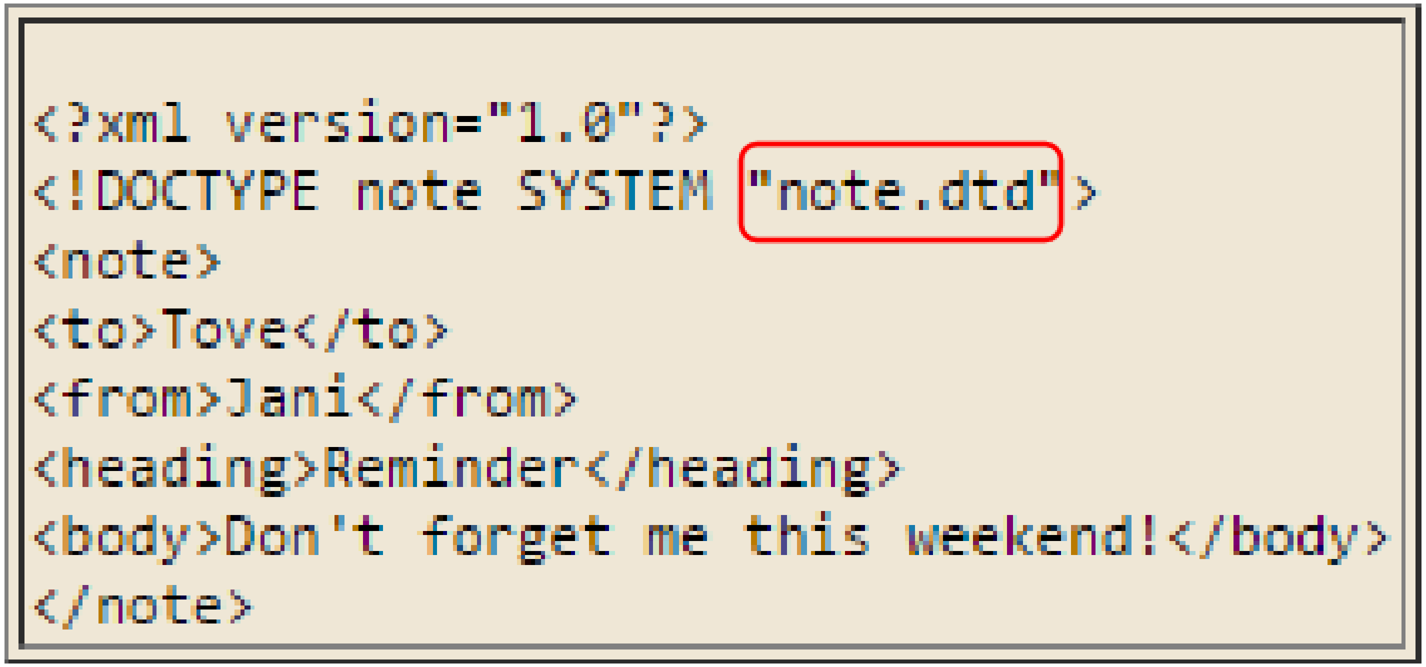
```
<?xml version="1.0"?>
<!DOCTYPE note [
  <!ELEMENT note    (to,from,heading,body)>
  <!ELEMENT to      (#PCDATA)>
  <!ELEMENT from    (#PCDATA)>
  <!ELEMENT heading (#PCDATA)>
  <!ELEMENT body    (#PCDATA)>
]>
<note>
<to>Tove</to>
<from>Jani</from>
<heading>Reminder</heading>
<body>Don't forget me this weekend!</body>
</note>
```

- External DTD, save as note.dtd



```
<?xml version="1.0"?>
<!ELEMENT note (to,from,heading,body)>
<!ELEMENT to (#PCDATA)>
<!ELEMENT from (#PCDATA)>
<!ELEMENT heading (#PCDATA)>
<!ELEMENT body (#PCDATA)>
```

- Call the DTD file to xml document



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

Well Formed and Valid

- An XML document with correct syntax is called "Well Formed".
- An XML document validated against a DTD is both "Well Formed" and "Valid"

What is XML?

- **XML** stands for *eXtensible Markup Language*. It is a simple and flexible markup language. It is known as universal language for data on the web because XML documents can be created and used in any language. It is universal standard for information interchange.
- XML technology facilitates you to create your own markup language.

How XML is different from HTML?

-
- HTML stands for Hyper Text Markup Language while XML stands for eXtensible Markup Language.
- The key differences between HTML and XML are given below:
- 1)HTML is used **to display data** and focuses on how data looks.XML is a software and hardware independent tool used **to transport and store data**. It focuses on what data is.
- 2)HTML is a **markup language** itself.XML provides a **framework to define markup languages**.
- 3)HTML is **not case sensitive**.XML is **case sensitive**.
- 4)HTML is a presentation language.XML is neither a presentation language nor a programming language.

- What is the meaning of version in XML?
  - Version is a tag used to show which version of XML is used.

What are the benefits of XML?

- These are the main benefits of using XML.
- **Simplicity:** Very easy to read and understand the information coded in XML.
- **Openness:** It is a W3C standard, endorsed by software industry market leaders.
- **Extensibility:** It is extensible because it has no fixed set of tags. You can define them as you need.
- **Self-descriptive:** XML documents do not need special schema set-up like traditional databases to store data. XML documents can be stored without such definitions, because they contain metadata in the form of tags and attributes

What is XML DOM?

- **DOM** stands for *Document Object Model* which is used to describe the logical structure of XML document. It is a hierarchical model that provides a way to access and manipulate an XML document.
- DOM methods and objects can be used with any languages like C#, VB, JavaScript and VB Script.

What is a well formed XML document?

- A syntactically correct document is called well formed XML document.

- A well formed XML document must follow the XML?s basic rules of syntax:
  - It must have a closing tag.
  - The closing tag must exactly match the open tag: XML is case sensitive.
  - All elements should be included within a single root tag.
  - Child elements must be closed within parent tag.

What is a valid XML document?

- A structurally correct element is called a valid XML document. It should follow some predefined rules of a specific type of document. These rules determine the type of data that each part of the document can contain.
- These rules can be written by the author of an XML document or someone other.

What is DTD?

- **DTD** stands for *Document Type Definition*. It defines a leading building block of an XML document. It defines:
  - Names of elements
  - How and where they can be used
  - Element attributes
  - Proper nesting