After MID

L#1 XML (Extensible Markup language)

```
<?xml version="1.0" encoding="iso-8859-1"?>
<?xsl-stylesheet type="text/xsl" href="book.xsl" ?>
<bookstore>
    <book category="Interstone">
        <title lang="en">Selan</title>
        <author>Author</author>
        <year>2000</year>
        <price>20000</price>
    </book>
    <book catagory="WEB">
        <title lang="en">Extract XML</title>
        <author></author>
        <year></year>
        <price></price>
    </book>
</bookstore>
```

→

Title	Author	year	Price

book.xsl

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<xsl-stylesheet version="1.0"
        xmlns:xsl="https://www.org/xml/XLS/transform">
    <xsl:output method="html" version="1.0"
                encoding="utf-8"/>

    <xsl:template match="/">
        <html> <body>
            <h1> the Book collection </h1>
            <table border="1">
                <tr><th> title </th>
                    <th> Author </th>
                    <th> year </th>
                    <th> price </th>
                </tr>
            <xsl:for-each select="bookstore/book">
                <tr>
                    <xsl:for-each select="./t">
                        <td> <xsl:value-of
                                select="."/>
                        </td>
                    </xsl:foreach>
                </tr>
            </xsl:foreach>
            </table>
        </body> </html>
    </xsl:template>
</xsl:stylesheet>
```

→ XML Documents has 2 parts
① Prolog
② Body

~~Prolog~~

① Prolog
- XML ~~Document~~ Declaration
- Optional Processing ~~Instruction~~ (PI)
- Comments.
- Document type Declaration (DTD)

① Declaration :
<?xml version="UTF-8"

② Processing Instruction :
<?xml href="Simple.xsl"
type="text/xsl"?>

③ Comments :
<!-- comments -->

④ Document type Declaration :

DOCTYPE

L#2

<?xml . . . ?>
<greeting> <\greeting> → two types

→ Elements
- Naming Rule
- Empty Element
- Attributes
- Well formed XML

<para type="node">XML data can contain Character</para>

type → attribute name
"node" → attribute value
attribute.

Content
Character entity reference

Element type name

Element

Anatomy of an XML Element.

SAMATHI

→ Well formed XML

- Single root element.
- all tags must be closed.
- tags must be properly nested.
- attributes must be quoted.
- Predefined Entity

<lt; ⟶ <
<gt; ⟶ >
<amp; ⟶ &
<quot; ⟶ "
<apos; ⟶ '

- Validation :

XmlLint --Valid Sample.xml.

→ XML namespaces :-

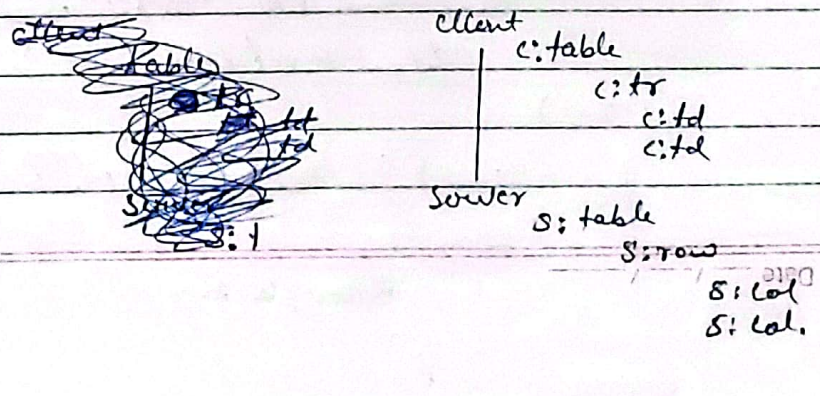
client.xml

```
<table>
    <tr><td> John </td><td> C++ </td></tr>
</table>
```

Server.xml

```
<table>
    <row> <Col> JSP </Col> <Col> ASP </Col> </row>
</table>
```

⟹ technology.



Namespace binding

xmlns : prefix = "URI".

```
<technology>
    client.xml
    <C:table.xmlns:C="https://mits.ac.in/client">
        <C:tr><C:td> Java <C:td>
              <C:td> Javascript <C:td>
        <C:tr>
    <C:table>
    server.xml
    <S:table xmlns:S="https://mits.ac.in/server">
        <S:row><S:col> JSP <S:col>
               <S:col> ASP <S:col>
        <S:row>
    <S:table>

<technology>
```

इन दोनों को हटा के <technology> लगा देना

L3 New Topic

XML DTD

Document type defination.

=> XML Schema language.

Example.xml

```
<?xml version="1.0" standalone="no"?>
<!Doctype HTML PUBLIC '-//W3C//DTD HTML 4.0
    Transitional EN' 'http://www.w3c.org/TR/REC-
    html 4.0/loose.dtd'>
<HTML>
    <Head><Title> An HTML <Title><head>
    <body>
        Example text.
    <body>
<html>
```

→ Internal DTD

```
<!DOCTYPE xml-element [
        <!-- DTD here -->
]>
```

Eg.

```
<?xml version="1.0"?>
<!DOCTYPE greeting [
        <!ELEMENT greeting (#PCDATA)>
]>
<greeting> Hello world </greeting>
```

→ External DTD

in xml Document.

```
<?xml version="1.0" standalone="no"?>
<!DOCTYPE root-element [ SYSTEM OR PUBLIC
'URI"> → optional.
```

→ Element Type declaration

7.6

```
<!ELEMENT element-name Type>   → [Type / empty / ANY]
        or
<!ELEMENT element-name (content)>
```

Eg.

```
<!ELEMENT br EMPTY>  ⇝  </br>
```

7.6.3 Simple element

```
<!ELEMENT element-name (#PCDATA)>
```

Parsed Character Data.

7.6.4.

```
<code>
<!PCDATA [
        <script language="Javascript">
        function less (x,y) {
                return x<y;
        }
]>      </script>  a= less(2,3);
</code>
```

7.6.5 Compound element

```
<!ELEMENT element (child-element1, child-element2
                   , ------ )>
<!ELEMENT child element, ------ >
```

Eg. ~~<?xml version~~

```
                    root element
<!DOCTYPE employee [
                                          child element
    <!ELEMENT employee (name)>
    <!ELEMENT name (#PCDATA)>
]>
```

- ---- xml look like ----

```
<employee>
    <name> UC Roy </name>
    <name> Uttam Roy </name>
</employee>
```

दो नाम नहीं लिख सकते पर ऐसा करने के लिए employee (name*) करना होगा।

*	→ 0 or more
+	→ 1 or more
?	→ 0 or 1

Eg. $a^* \rightarrow \phi, a, aa, aaa \ldots$

$a^+ \rightarrow a, aa, aaa \ldots$

$a? \rightarrow \phi / a.$

→ Multiple children :-

Eg.

```
<!DOCTYPE employee [
    <!ELEMENT employee (fname, mname, lname)>
    <!ELEMENT fname (#PCDATA)>
    <!ELEMENT mname (#PCDATA)>
    <!ELEMENT lname (#PCDATA)>
]>
```

<!-- With this we can write -->

```
<employee>
    <fname> Utkarsh </fname>
    <mname> Kumar </mname>
    <lname> Roy </lname>
</employee>
```

SAAFTAI

more complicated.

```
<! ELEMENT employee (name, contact*) >
        <! Element name (fname, mname, lname) >
        <! " " fname (#PCDATA) >
        <! " " mname ( " " ) >
        <! " " lname ( " " ) >
        <! " " contact (#PCDATA) >
<— with this we can write >
<employee>
        <name>
                <fname> — </fname>
                <mname> —
                <lname> —
        </name>
        <contact> 7568— </contact>
</employee>
```

#14 → Choice operator (|)

Eg.

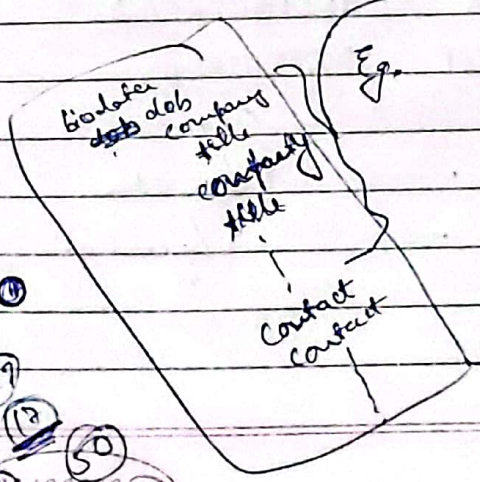
```
<! ELEMENT contact (mobile | land) >
<! ELEMENT mobile (#PCDATA) >
<! ELEMENT land (#PCDATA) >
```

→ Composit operator (,)

→ multiple

Eg.

```
<! ELEMENT biodata (dob, (company, title)*, contact*) >
<! ELEMENT item (#PCDATA | description)* >
<! ELEMENT description (#PCDATA) >
```



7.7 Attribute declaration

<!ATTLIST elem-name attrib-name attrib-type default-value>

<!ELEMENT line EMPTY> - - - > <line/>

<!ATTLIST line width CDATA '100'> - - < line

no '#' sign

(7.7.1)

Some-value/default

Eg 1. <!ATTLIST price CDATA #REQUIRED> — #REQUIRED
#IMPLIED
#FIXED value.

<Price> 100 </Price>

Eg 2.

7.7.2 Attribute type

7.7.3) String type → CDATA

7.7.4) tokenized type

7.7.4.1) ID

<!ATTLIST question no. ID #REQUIRED>

<Question no. = "1">

</Question>

7.7.4.2) IDREF

<!ATTLIST question no. ID #REQUIRED>

<!ATTLIST answer IDREF #REQUIRED>

<Question no. = "Q1" > - - -

7.7.4.3 IDREFS

<!ATTLIST question no. ID #REQUIRE>

7.7.4.4 NMTOKEN

<!ATTLIST item ~~Serial no~~ serial no, NMTOKEN #REQUIRE>

<Item serial no. = "12345-ABCD">
"12345 ABCD".

7.7.5 Enumerated type,

Eg, <!ATTLIST employee (male|Female) # REQUIRED>

7.7.5.3 multiple attribute

<!ATTLIST employee
homepage CDATA #IMPLIED.
gender (male|Female) #REQUIRED.
>

7.8 Entity declaration.

7.9 DTD Validation.

SAATHI

Web Tech

eXtensible Markup Language (XML)

```
prolog { <?xml version="1.0" encoding="ISO-8859-1" ?>
         <?xml-stylesheet type="text/xsl" href="book.xsl" ?>

<bookstore>
    <book category="literature">
    <title lang="bang">Sanchaita</title>
    <author> Rabindranath Tagore </author>
    <year> 2009 </year>
    <price> 200 </price>
    </book>
    <book category="web">
        <title lang="en"> Essential XML </title>
        <author> Don Box </author>
        <year> 2000 </year>
        <price> 150 </price>
    </book>
</bookstore>
```

book.xsl

```
<?xml version="1.0" encoding="ISO-8851-1" ?>
<xsl:stylesheet version="1.0"
        xmlns:xsl="http://www.w3.org/1999/XLS/transform" >

<xsl-output method="html" version="1.0" encoding="UTF-8"/>

<xsl:template match="/" >
    <html> <body>
        <h1> Book collection </h1>
        <table border="1" >
            <tr><th> Title </th><th> Author </th><th> year
                </th><th> Price </th></tr>
            <xsl:for-each select="book store/book">
                <tr>
                    <xsl:for-each select="./t
                        <td> <xsl:value-of select="."/> </td>
                    </xsl:for-each>
                </tr>
            </xsl:foreach>
        </table>
    </body> </html>
</xsl:template>

</xsl:stylesheet>
```

Title	Author	Year	Price

XML document has 2 parts.

- Prolog
- Body.

Prolog

- XML declaration
- → optional processing instructions. (PI)
- comments.
- Document type declaration (DTD)

declaration

```
<? xml version = "1.0" encoding = "UTF-8"
        standalone = "no" ?>
```

processing instructions.

```
<? xml-stylesheet href = "simple.xsl" type = "text/xsl"?
```

comments.

```
<! -- comment -->
```

Document type declaration

DOCTYPE

Elements

- Naming rules
-

Elements.

<para type="note"> ML data can contain << character with no problem </para>

attribute value — content

attribute name

element type/name

Element type/name

Element

Anatomy of an XML element

Elements.

- Naming rules
- Empty elements

Attributes

Wellformed XML

- single root element
- All tags must be closed
- Tags must be properly nested
- attribute values must be quoted

Predefined Entity

< → <

> → >

& → &

" → "

' → '

Validation

xmllint --valid sample.xml

XML Namespaces

client.xml

```
<table>
    <tr><td> Java </td><td> c++ </td></tr>
</table>
```

Server.xml

```
<table>
    <row><col> JSP </col><col> Asp </col></row>
</table>
```

Namespace binding

xmlns:prefix = "URI"

```
<technology>
  client.xml
    <c:table xmlns:c="http://nits.ac.in/client">
        <c:tr><c:td> Java </c:td><c:td> Javascript </c:td>
                                  </c:tr>
    </c:table>
  server.xml
    <s:table xmlns:s="http://nits.ac.in/server">
        <s:tr><s:td> JSP </s:td><s:td> Asp </s:td></s:tr>
    </s:table>
</technology>
```

Web Tech

XSLT : XML stylesheet Translation language.

Source

```
<? xml version = '1.0' ?>
<employee>
    <name> P S Hongiman </name>
    <sex> Male </sex>
    <status> Unmarried </status>
    <salary> basic = '15000' DA = '10000' /></
</employee>
```

target

```
<? xml version = '1.0' ?>
<employee profile = 'male unmarried' >
    <name> P S Hongiman </name>
    <salary>
        <basic> 15000 </basic>
        <DA> 1000 </DA>
    </salary>
</employee>
```

```
<? xml version='1.0' ? >
                    → xsl:stylesheet
<xsl: transform version = 2.0. xmlns:xsl= "http://
        www.w3c.org/1999/XLS/Transform' >
    <xsl: template match = '/employee' >
        <xsl: element name = 'employee' >
            - <xsl:attribute name = 'profile' ~~select~~
                    select= 'concat (status," ", sex)'/ >
            <name> <xsl: value of select='name'/> </name>
            <salary> .
                <basic> <xsl: value of select = 'salary/@basic
                    '/> </basic>
                <DA> <xsl: value of select= 'salary/@DA
                    </DA>
            </salary>
        </xsl : element>
    </xsl: template>
</xsl : transform>
```

11.12 Conditionals

11.12.1 If

```
<xsl:if test = 'Condition'>
  — action —
  - - - - -
</xsl:if>
```

Eg.

```
<book title = '----'>
  - - - - -
  - - - - -
</book>
```

Eg2

```
<xsl: template match='//book'>
<xsl:if test = '@title="Computer Network"'>
    Found Computer Networks
</xsl:if>
</xsl:template>
```

11.12.2

Choose (like switch in C) when = case

Syntax

```
<xsl: choose>
  <xsl: when test = 'expression1'> action 1 </xsl:when>
  <xsl: when test = 'expression2'> action 2 </xsl:when>
  — — —
  — — —
  <xsl: otherwise> default action </xsl: otherwise>
</xsl: when>
```

Exam

Eg.

```
<result>
  <student rollno = '01'> <marks> 50 </marks> </student>
  <student rollno. = '20'> <marks> 80 </marks> </student>
  — — —
  — — —
</result>
```

```
<xsl: template match = '/'>
  Roll no, ---- Grade.
<xsl: for-each select = '// student'>
  <xsl: value-of select = '@ rollno'> ----
```

```
<xsl:choose>
    <xsl:when test='marks-gt; 70 and marks-lt; 100>
        A</xsl:when>
    <xsl:when test='marks-gt; 40 and marks-lt; 70>B
        </xsl:when>
    <xsl:default> F </xsl:default>
<xsl:choose>
</xsl:for-each>
</xsl:template>
```

output

Roll no.	—	Grade
01	—	B
02	—	A
⋮		⋮

11.14 Creating Nodes & Sequences.

11.14.1 Creating element nodes.

Example.

```
<xsl:element name='greeting'>
    Hellowoord!
</xsl:element>
```

→ output :

<greeting> Hello world ! </greeting>

11.14.2

Creating attributes

```
<xsl: element name = 'greeting'>
        <xsl: attribute name = 'lang'> en </xsl: attribute>
</xsl: element>
```

output :

```
<element lang = 'en'> Hello world ! </greeting>
```

11.1 (fig 11.3)

Employee
name
sex
Salary (basic, DA)

=>

Employee (Profile < sex, status >)
name
Salary
basic
DA

```
<xsl: element name = 'employee'>
        <xsl: attributes name = 'Profile' select = 'concat(status, " ", sex)' />
</xsl: element>
```

```
<xsl: attribute name = 'Profile'>
        <xsl: value-of select = 'concat (@status, " ", @sex)' />
</xsl: attribute>
```

both are same

Can do any one of these