Module 6

The eXtensible Markup Language

Dr. L.M. Jenila Livingston

VIT Chennai

OEM

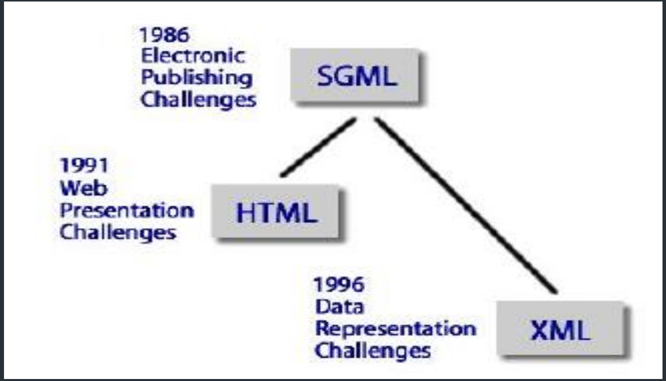
- Object Exchange Model (OEM)
- Model for the exchange semi-structured data (XML,JSON,.csv) between object oriented databases.
- It acts as the basic data model for several projects

OEM

- Each OEM object is encoded with a < symbol, followed by a label, a value, and a terminating >

```
<Birthday {
<Month “April">
<Day 14>
<Year 2007>
}>
```

Background for XML



- An Extensible Markup Language (XML) document describes the *structure of data*
- XML and HTML have a similar syntax ... both derived from SGML (Standard Generalized Markup Language)
- An XML document resides in its own file with an '.xml' extension

Quick Comparison

- HTML

- uses tags and attributes
- content and formatting can be placed together
  <p><font=”Arial”>text</font>
- tags and attributes are pre-determined and rigid

- XML

- uses tags and attributes
- content and format are separate; formatting is contained in a stylesheet
- allows user to specify what each tag and attribute means

HTML and XML

- HTML
  - describes web pages
  - You don’t get error messages about the web pages you visit
  - Browsers ignore and/or correct as many HTML errors as they can, so HTML is often sloppy
- XML
  - XML describes data
  - The rules are strict and errors are not allowed
    - In this way, XML is like a programming language
  - Current versions of most browsers can display XML
    - However, browser support of XML is spotty at best

Example of an HTML Document

```
<html>
  <head><title>Example</title></head.
<body>
  <h1>This is an example of a page.</h1>
  <h2>Some information goes here.</h2>
</body>
</html>
```

XML Rules

- XML is case sensitive
- All start tags must have end tags
- Tags must be properly nested
  - **<name><email>…</name></email> is not allowed.**
  - **<name><email>…</email><name> is.**
- XML declaration is the first statement
- Every document must contain a root element

XML Rules

- Attribute values must have quotation marks
- Certain characters are reserved for parsing
- Tags are enclosed in angle brackets.
- Tags that do not have end-tags must be terminated by a '/'.
  - <br /> is an html example.

The pieces

- There are 3 components for XML content:

  - the XML document
  - DTD (Document Type Declaration)
  - XSD (XML Schema Definition)

XML Comments

- <!-- --> comments

- contents are ignored by the processor

- cannot come before the XML declaration

- cannot appear inside an element tag

- may not include double hyphens

XML Declaration

```
<?xml
version ="version_number"
encoding ="encoding_declaration"
standalone="standalone_status"
?>
```

encoding – the character encoding used in the document

standalone –if an external DTD is required

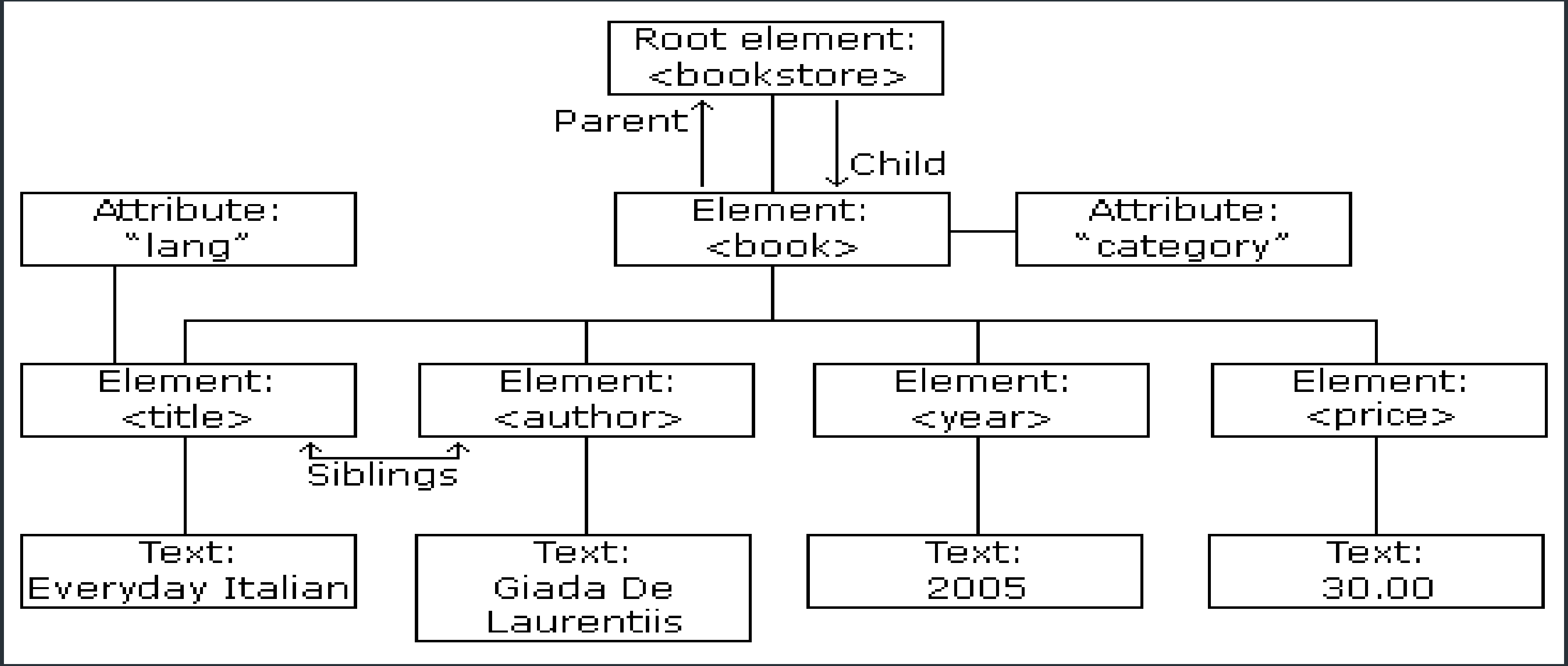
<?xml version=”1.0” encoding=”UTF-8”>

<?xml version=”1.0” standalone=”yes”>

XML Trees

- An XML document has a single root node.
- The tree is a general ordered tree.
  - A parent node may have any number of children.
  - Child nodes are ordered, and may have siblings.
- Preorder traversals are usually used for getting information out of the tree.

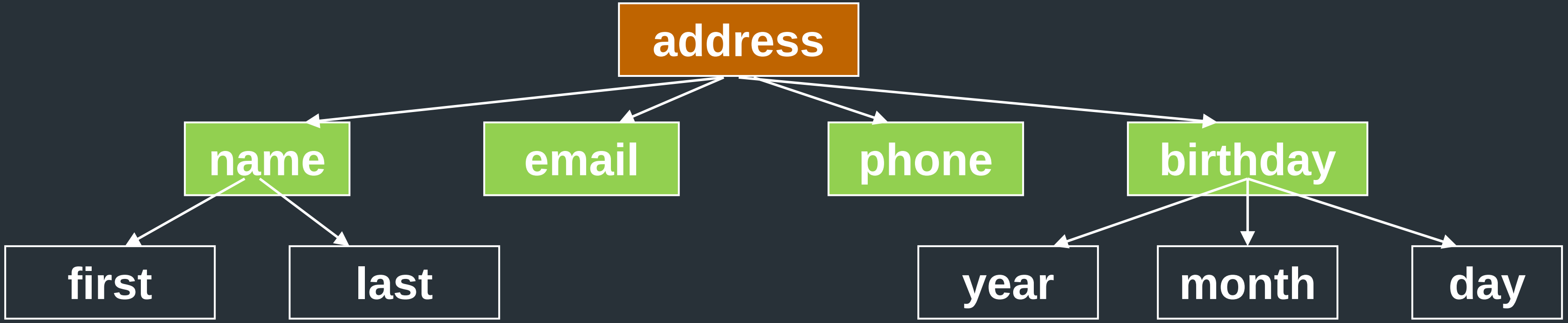
XML Tree Structure



XML Tree Structure

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

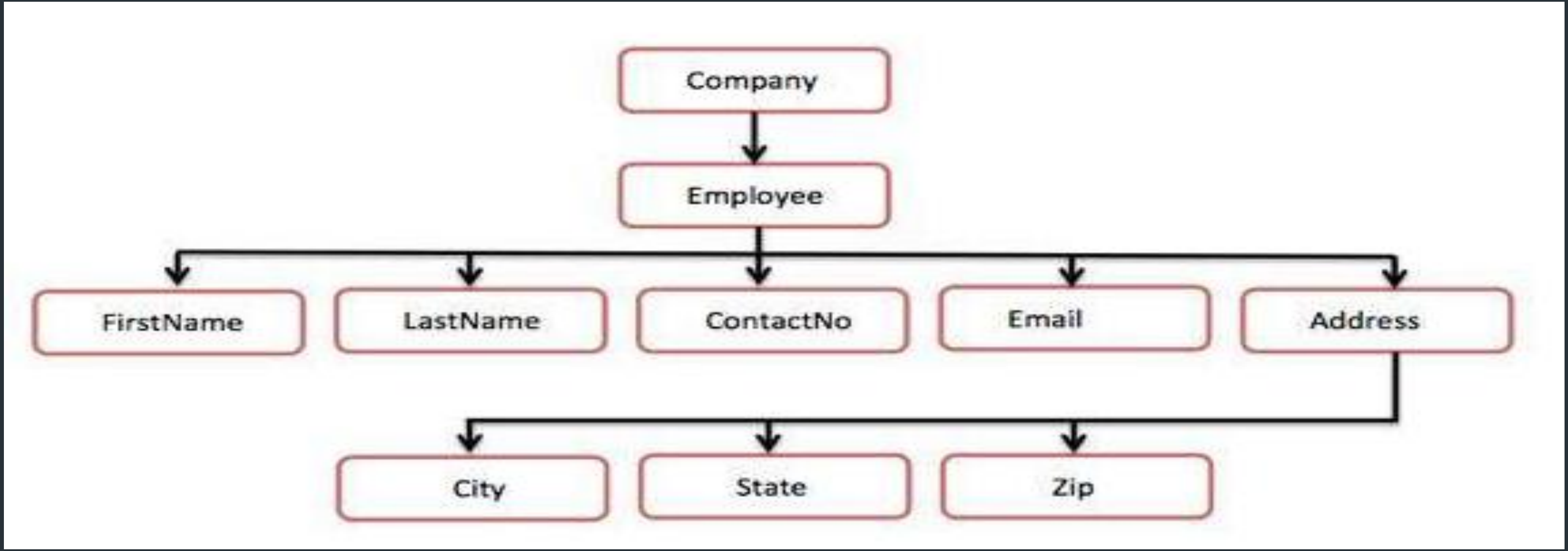
Address - Tree Representation



Example 1: XML document for the given tree

```
<?xml version = “1.0” ?>
<address>
  <name>
      <first>Jafflyn</first>
      <last>Leni</last>
    </name>
    <email>leni@gmail.com</email>
    <phone>+91-91414-14141</phone>
    <birthday>
      <year>2007</year>
      <month>04</month>
      <day>21</day>
  </birthday>
</address>
```

Company - Tree representation



Example 2: XML document for the given tree

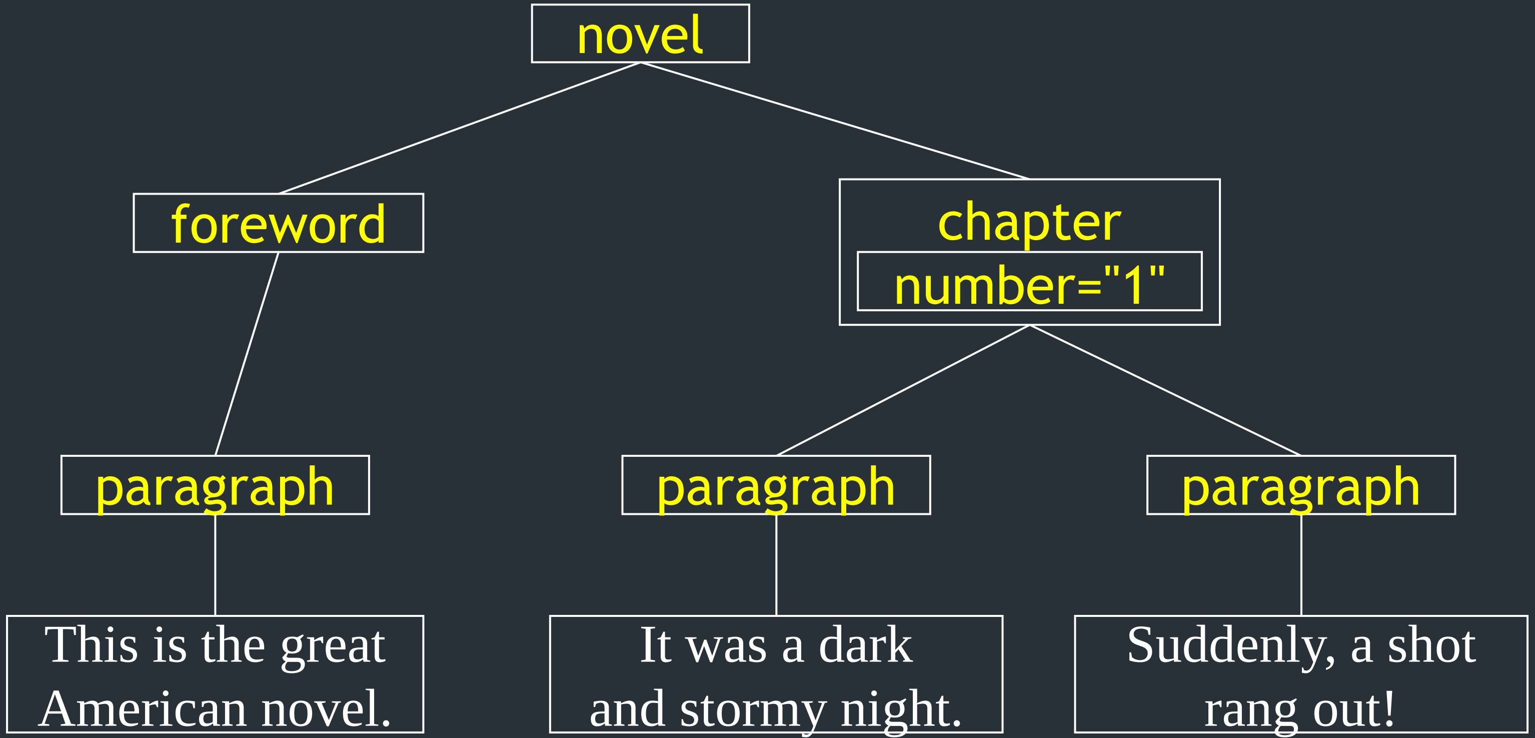
```
<?xml version = "1.0"?>
<Company>
    <Employee>
        <FirstName>Frank</FirstName>
        <LastName>Len</LastName>
        <ContactNo>1234567890</ContactNo>
        <Email>frank@xyz.com</Email>
        <Address>
            <City>Bangalore</City>
            <State>Karnataka</State>
            <Zip>560212</Zip>
        </Address>
    </Employee>
</Company>
```

Example 3: XML document

```
<?xml version="1.0“?>
<weatherReport>
  <date>7/1/2020</date>
  <city>Chennai</city>
  <state>TN</state>
  <country>India</country>
  High Temp: <high scale="F">103</high>
  Low Temp: <low scale="F">70</low>
  Morning: <morning>Partly cloudy, Foggy</morning>
  Afternoon: <afternoon>Sunny &amp; hot</afternoon>
  Evening: <evening>Clear and Cool</evening>
</weatherReport>
```

XML attribute

number=“1”


novel
foreword
chapter
number="1"
paragraph
paragraph
paragraph
This is the great American novel.
It was a dark and stormy night.
Suddenly, a shot rang out!

Example 4: XML Attributes

```
<novel>
  <foreword>
        <paragraph> This is the great American novel.
        </paragraph>
  </foreword>
  <chapter number="1">
        <paragraph>It was a dark and stormy night.
        </paragraph>
        <paragraph>Suddenly, a shot rang out!
        </paragraph>
  </chapter>
</novel>
```

Example 5: XML Attribute

```
<person gender="Female">
  <firstname>Jafflyn</firstname>
  <lastname>Leni</lastname>
</person>

<person>
  <gender>Female</gender>
  <firstname>Jafflyn</firstname>
  <lastname>Leni</lastname>
</person>
```

Document Type Definitions (DTD)

- A DTD describes the tree structure of a document and something about its data.
- There are two data types, PCDATA and CDATA.
  - PCDATA is parsed character data.
  - CDATA is character data, not usually parsed.
- A DTD determines how many times a node may appear, and how child nodes are ordered.

PCDATA

- PCDATA is text that will be parsed by a parser. Tags inside the text will be treated as markup and entities will be expanded.
- PCDATA is the only plain text content model.

CDATA

- CDATA is text that will *not* be parsed by a parser. Tags inside the text will *not* be treated as markup and entities will not be expanded.
- The predefined entities such as **&lt;, &gt;,** and **&amp; quote(')** and **double quote (")** require typing and are generally difficult to read in the markup. In such cases, CDATA section can be used.

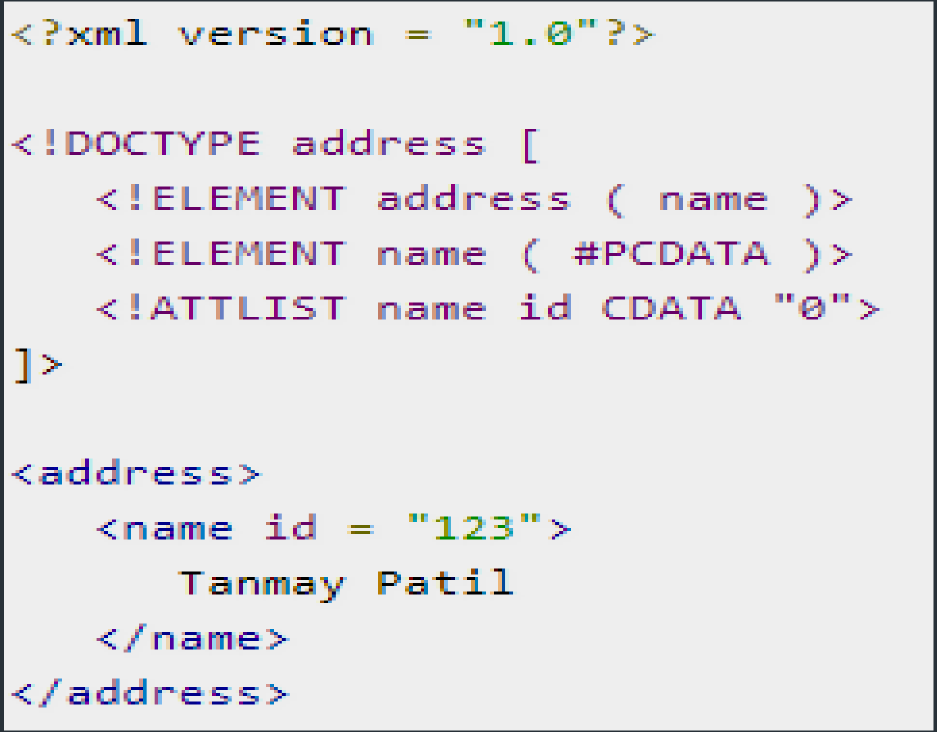
- <![CDATA[characters with markup]]>

Internal DTD

```
<?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> Frank</name>
   <company>GE</company>
   <phone>044-333335</phone>
</address>
```

CDATA

<!ATTLIST element-name attribute-name attribute-type "default-value">



```
<?xml version = "1.0"?>

<!DOCTYPE address [
   <!ELEMENT address ( name )>
   <!ELEMENT name ( #PCDATA )>
   <!ATTLIST name id CDATA "0">
]>

<address>
   <name id = "123">
      Tanmay Patil
   </name>
</address>
```

External DTD

- In external DTD elements are declared outside
- the XML file. They are accessed by specifying the
- system attributes which may be either the
- legal *.dtd file or a valid URL.*

External DTD – Example1

address.dtd

```
<!ELEMENT address (name,company,phone)>
<!ELEMENT name (#PCDATA)>
<!ELEMENT company (#PCDATA)>
<!ELEMENT phone (#PCDATA)>
```

External DTD – E.g. 1 contd..

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<!DOCTYPE address SYSTEM "address.dtd">
<address>
  <name> Frank</name>
  <company>GE</company>
  <phone>044-333335</phone>
</address>
```

External DTD for address Example 2

Address.dtd

```
<!ELEMENT address (name, email, phone, birthday)>
  <!ELEMENT name (first, last)>
          <!ELEMENT first (#PCDATA)>
          <!ELEMENT last (#PCDATA)>
  <!ELEMENT email (#PCDATA)>
  <!ELEMENT phone (#PCDATA)>
  <!ELEMENT birthday (year, month, day)>
          <!ELEMENT year (#PCDATA)>
          <!ELEMENT month (#PCDATA)>
          <!ELEMENT day (#PCDATA)>
```

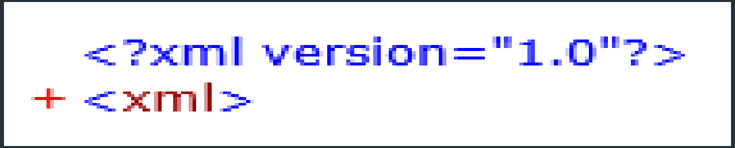
External DTD – E.g. 2 contd..

```
<?xml version="1.0" encoding="UTF-8" standalone="no" ?>
<!DOCTYPE address SYSTEM "address.dtd">
<?xml version = “1.0” ?>
<address>
        <name>
            <first>Jafflyn</first>
            <last>Leni</last>
        </name>
        <email>leni@gmail.com</email>
        <phone>+91-91414-14141</phone>
        <birthday>
            <year>2007</year>
            <month>04</month>
            <day>21</day>
        </birthday>
</address>
```

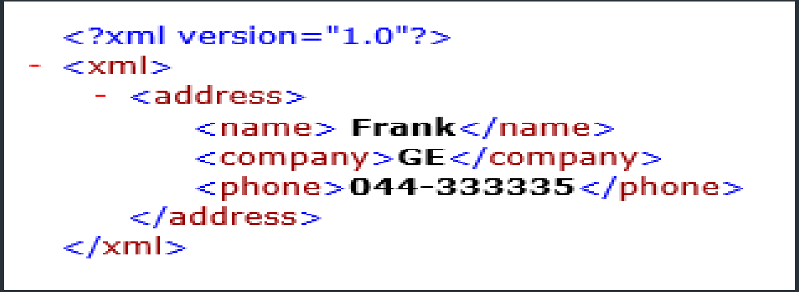
XML output in browser

Address.xml

```
<xml>
<address>
		<name> Frank</name>
		<company>GE</company>
		<phone>044-333335</phone>
</address>
</table>
</xml>
```



```
  <?xml version="1.0"?>
+ <xml>
```

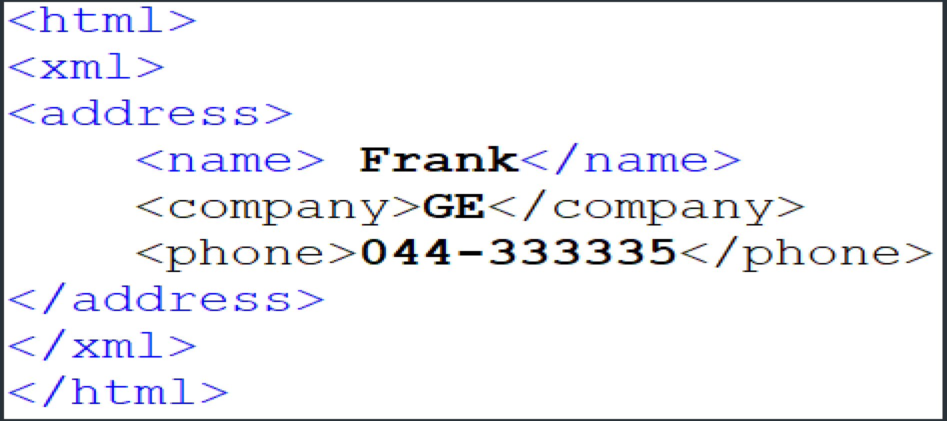


```
  <?xml version="1.0"?>
- <xml>
  - <address>
      <name> Frank</name>
      <company>GE</company>
      <phone>044-333335</phone>
    </address>
  </xml>
```

Data Islands

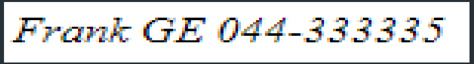
- XML can be used to store data inside HTML documents. XML data can be stored inside HTML pages as "**Data Islands**".

xmlbind.html



```
<html>
<xml>
<address>
    <name> Frank</name>
    <company>GE</company>
    <phone>044-333335</phone>
</address>
</xml>
</html>
```

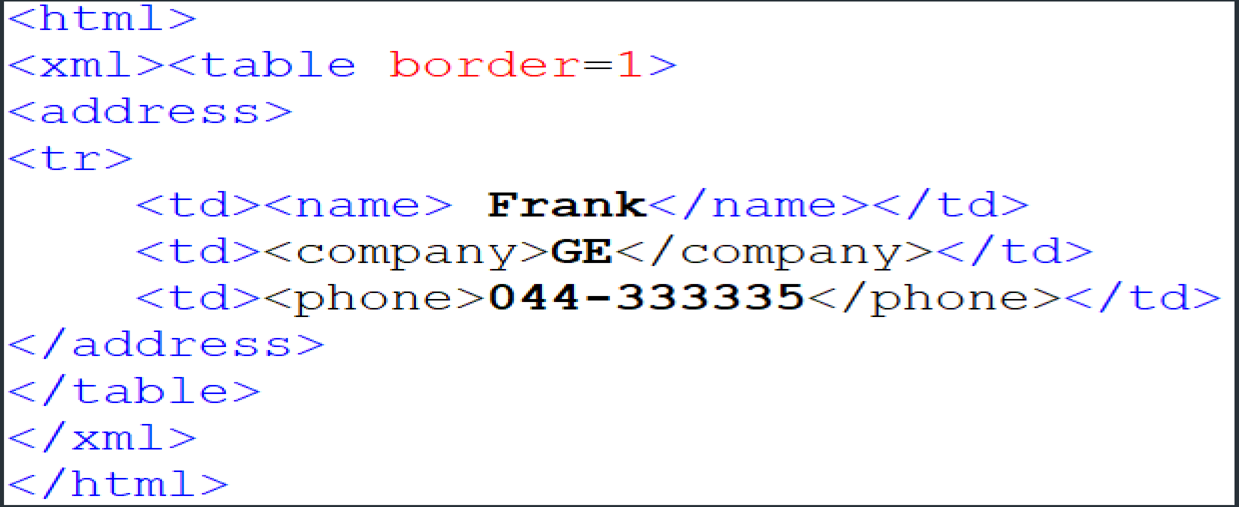
Output



```
Frank GE 044-333335
```

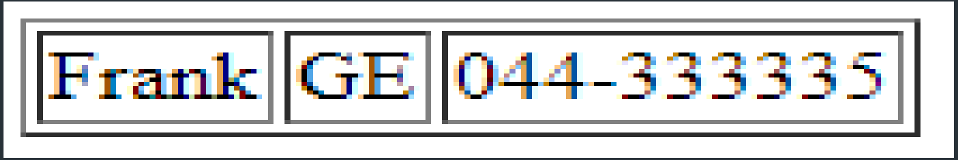
Data Islands

xmlbind2.html



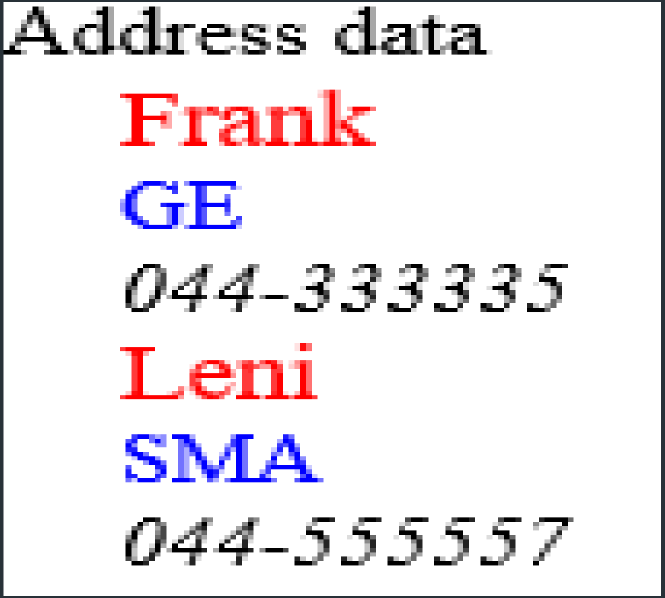
```
<html>
<xml><table border=1>
<address>
<tr>
    <td><name> Frank</name></td>
    <td><company>GE</company></td>
    <td><phone>044-333335</phone></td>
</address>
</table>
</xml>
</html>
```

Output:



```
Frank | GE | 044-333335
```

CSS Style



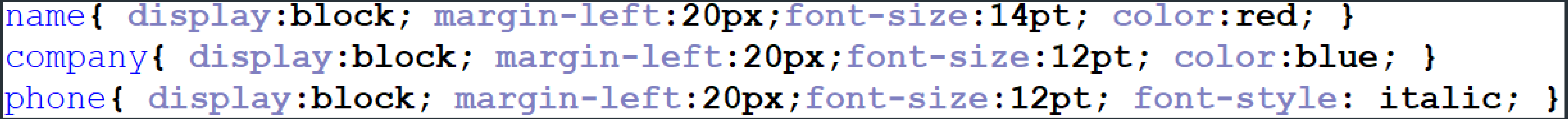
```
Address data
    Frank
    GE
    044-333335
    Leni
    SMA
    044-555557
```

CSS style

Steps:

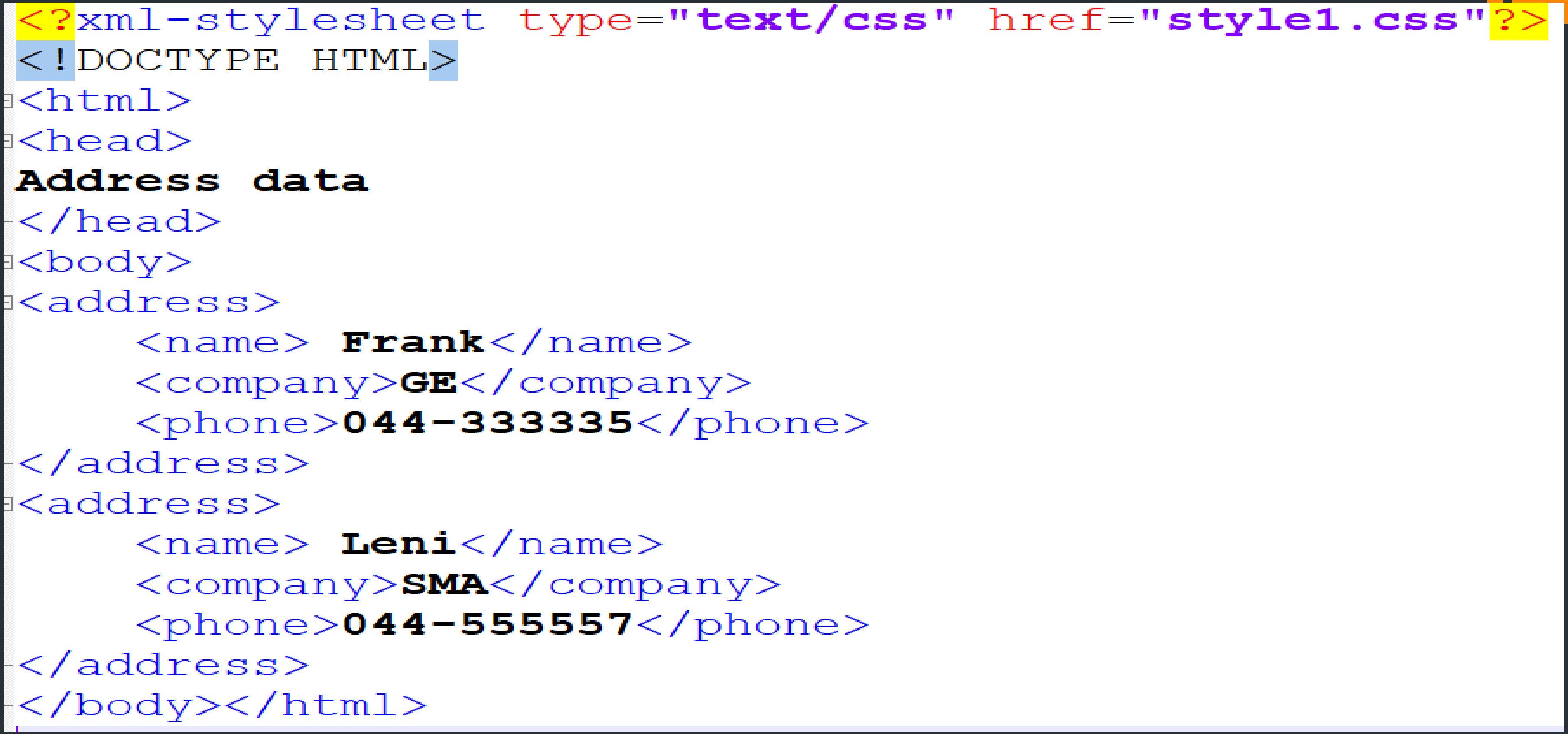
- Create a style document using CSS
- Create an XML document for Address DB

style.css



```
name{ display:block; margin-left:20px;font-size:14pt; color:red; }
company{ display:block; margin-left:20px;font-size:12pt; color:blue; }
phone{ display:block; margin-left:20px;font-size:12pt; font-style: italic; }
```

address.xml



```
<?xml-stylesheet type="text/css" href="style1.css"?>
<!DOCTYPE HTML>
<html>
<head>
Address data
</head>
<body>
<address>
    <name> Frank</name>
    <company>GE</company>
    <phone>044-333335</phone>
</address>
<address>
    <name> Leni</name>
    <company>SMA</company>
    <phone>044-555557</phone>
</address>
</body></html>
```

XML Schema Definition (XSD)

- Schemas are themselves XML documents.
- They were standardized after DTDs and provide more information about the document.
- They also determine the tree structure and how many children a node may have.
- A XSD file has a .xsd extension

Data Types

- There are many data types in XSD. Data types are classified into
  - XSD Strings
  - XSD Numeric
  - XSD Date

XSD Strings

- A String data types contains characters like alphabets, numbers and special characters, line feed, carriage returns and tab spaces

Data Types	Description
string	A string
name	A string which contains a valid name
normalizedString	A string that does not contain line feeds, carriage returns, or tabs

XSD Numeric

- These data types contains numbers which may be a whole number or decimal number.

Data types	Description
Integer	Contains integer value
Decimal	Contains decimal value
positiveInteger	Contains integer value which is only positive

XSD Date

- This data type contains date and time values.
- Format of the date is “YYYY-MM-DD”
- All are mandatory

Data types	Description
Date	Defines the date value (YYYY-MM-DD)
Time	Defines the time value (hh:mm:ss)
DateTime	Defines both data and time (yyyy-mm-ddThh:mm:ss)

XML Schema Definition (XSD)

- They divide elements into
  - Simple types and
  - Complex types

Simple Type

- Syntax –
  - <xs:element name="xxx" type="yyy"/>
  - ‘xxx’ is the name of the element and ‘yyy’ is the data type of the element.

```
<xs:element name="name" type="xs:string"/>

<xs:element name="email" type="xs:string"/>

<xs:element name="phone" type="xs:integer"/>

<xs:element name="birthday" type="xs:date"/>
```

- Simple type with Default value:

```
<xs:element name=“salary" type="xs:integer“ default: 5000/>
```

Complex Type

```
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">
<xs:element name="address">
  <xs:complexType>
          <xs:sequence>
                    <xs:element name="name" type="xs:string"/>
                    <xs:element name="email" type="xs:string"/>
                    <xs:element name="phone" type="xs:integer"/>
                    <xs:element name="birthday" type="xs:date"/>
          </xs:sequence>
  </xs:complexType>
</xs:element>
</xs:schema>
```

Schema and xmlns

- Syntax:

```
xs:schema xmlns:xs=uri
```

- xs:schema defines a schema
- The Namespace starts with the keyword **xmlns**.
- Uniform Resource Identifiers (URI) associated with this namespace is the Schema language definition, which has the standard value of http://www.w3.org/2001/XMLSchema

XML Schema(xs) and XML Schema Instance (xsi)

- URI references identifying namespaces are compared when determining whether a name belongs to a given namespace
- xmlns:xs="http://www.w3.org/2001/XMLSchema" indicates that the elements and data types used in the schema come from the

  http://www.w3.org/2001/XMLSchema" namespace.
- namespace should be prefixed with **xs:**
- xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" tells the XML parser that this document should be validated against a schema.
- xsi:noNamespaceSchemaLocation="**address.xsd**" specifies WHERE the schema resides (here it is in the same folder as "employee.xml")

Associating XML with XSD

- Define an XSD that contains address schema like name, email, phone and birthday.
- XML file to enter data
- xsi:noNamespaceSchemaLocation:
  - ✓ locates the schema for elements that are not in any namespace.

address.xsd

```
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">
<xs:element name="address">
  <xs:complexType>
        <xs:sequence>
              <xs:element name="name" type="xs:string"/>
              <xs:element name="email" type="xs:string"/>
              <xs:element name="phone" type="xs:integer"/>
              <xs:element name="birthday" type="xs:date"/>
        </xs:sequence>
  </xs:complexType>
</xs:element>
</xs:schema>
```

employee.xml

```
<?xml version="1.0"?>
<address xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:noNamespaceSchemaLocation=“address.xsd">

<name> Frank </name>
<email> frank@gmail.com </email>
<phone> 04422555567 </phone>
<birthday> 21/Aug/1993 </ birthday >

</address>
```

Thank
You!