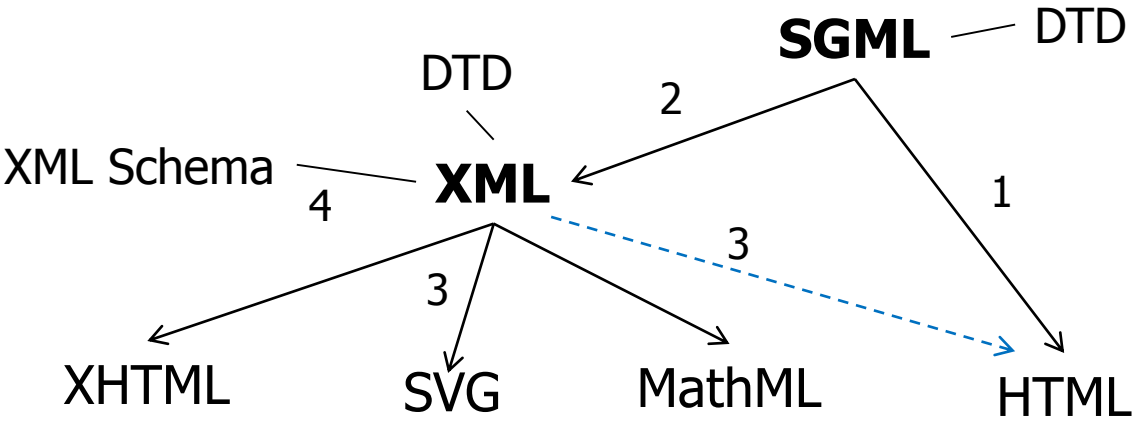
COMP 4021
Internet Computing

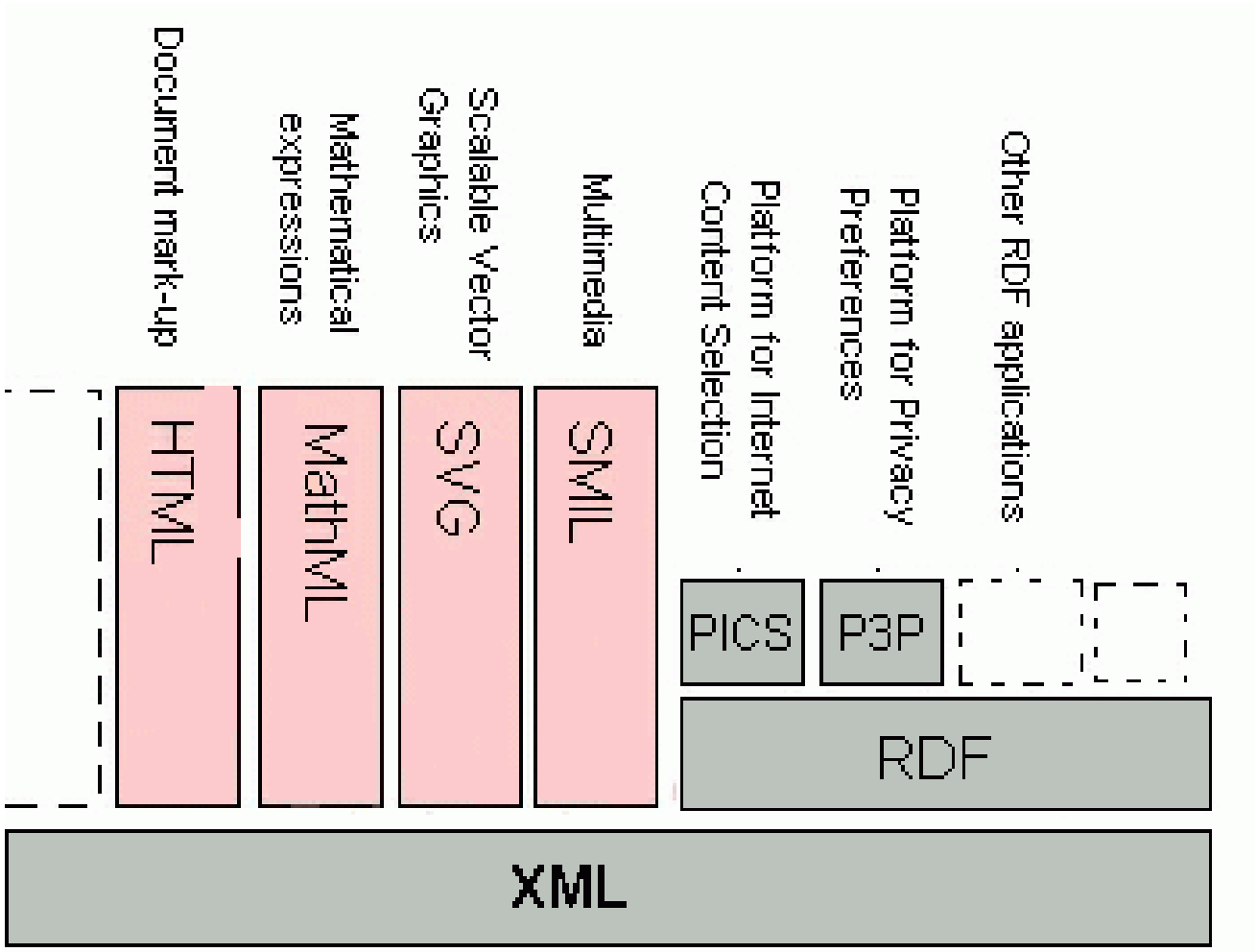
XML

History

- Mid 1980s: Information retrieval, library and publishing communities developed SGML (Standard Generalized Markup Language):
- Step 1 (Late 1980s): HTML taken some feature from SGML (e.g., tagging); DTD existed for HTML but not used very much
- Step 2 (Mid 1990s): XML as a simplified version of SGML
- Step 3 (Late 1990s till now): XHTML, SVG, MathML, etc., etc.
- Step 4 (Early 2000s): XML Schema



Many Standards are Built on XML



XML

- eXtensible Markup Language
- HTML is for markup of documents; XML can be used to mark up any kind of data (structured data as in database), strings, and supports nesting
- It is NOT a language that allows you to add more commands and functions to make up a more powerful (and hence bigger) language
- It is a language that allows you to define a new language
- You need a grammar to define a new language; XML allows you to define a grammar using Document Type Definition (DTD)

Briefly Speaking …

- XML describes the structure/ content of a document
- You can quickly define your own XML structure and let other application to parse the structure
  - Of course, defining a full grammar is very difficult (as we will see later)
- XML doesn't describe any visual appearance

Tags, Elements and Attributes

Element: The "address" element contains four sub-elements, name, street, city and postal-code

Tag: start tag

Attribute: Name=value pairs inside start tags

Tag: end tag

```
<address>
  <name>
    <title>Mrs.</title>
    <first-name>
      Mary
    </first-name>
    <last-name>
      McGoon
    </last-name>
  </name>
  <street>
    1401 Main Street
  </street>
  <city state="NC">Anytown</city>
  <postal-code>
    34829
  </postal-code>
</address>
```

Well-formed, Valid and Invalid XML

- Well-formed documents: Follow the XML syntax rules but don't have a DTD or schema
- Valid documents: Follow both the XML syntax rules and the rules defined in their DTD or schema
- Invalid documents: Don't follow the XML syntax rules or the DTD or schema, if available

XML Syntax Rules (I)

- The root element: An XML document must be contained in a single element called the root element

```
<?xml version="1.0"?>
<!-- A well-formed document -->
<greeting>
  Hello, World!
</greeting>
```

```
<?xml version="1.0"?>
<!-- An invalid document -->
<greeting>
  Hello, World!
</greeting>
<greeting>
  Hola, el Mundo!
</greeting>
```

- XML elements can't overlap.

```
<!-- NOT legal XML markup -->
<p>
  <b>I <i>really
  love</b> XML.
  </i>
</p>
```

XML Syntax Rules (II)

- End tags are required; note how empty elements are handled

```
<!-- NOT legal XML markup -->
<p>Yada yada yada...
<p>Yada yada yada...
<p>...
```

```
<!-- Two equivalent break elements -->
<br></br>
<br />

<!-- Two equivalent image elements -->
<img src="../img/c.gif"></img>
<img src="../img/c.gif" />
```

- Elements are case sensitive (convention is to use lower case as much as possible)

```
<!-- NOT legal XML markup -->
<h1>Elements are
  case sensitive</H1>

<!-- legal XML markup -->
<h1>Elements are
  case sensitive</h1>
```

XML Syntax Rules (III)

- An attribute, if specified, must have a value
- Attribute values must be double or single quoted

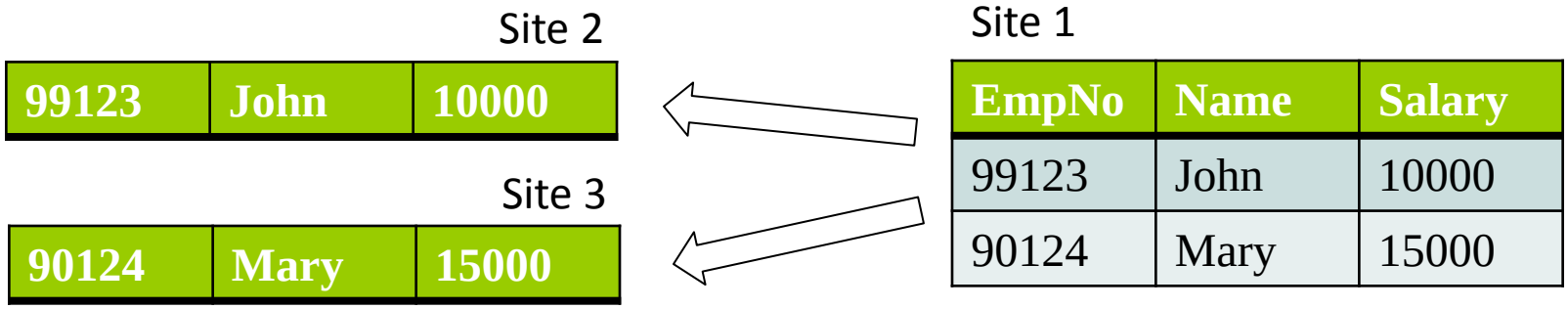
```
<!-- NOT legal XML markup -->
<ol compact>

<!-- legal XML markup -->
<ol compact="yes">
```

- Parameter values are enclosed in speech marks

  I.e. <circle id="face_outline" . . . />

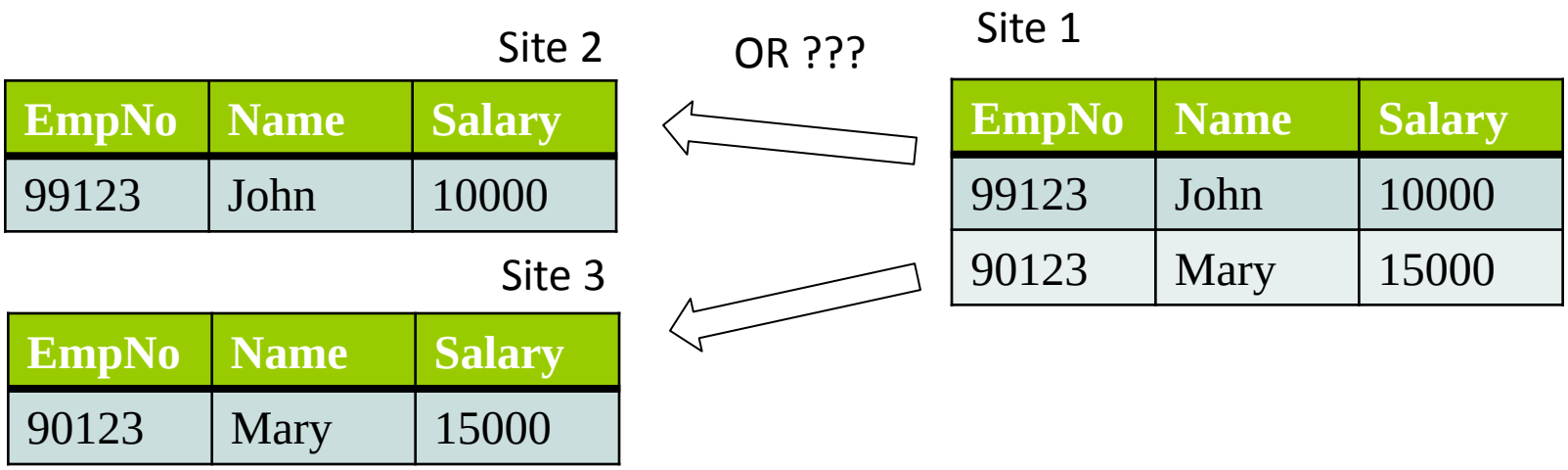
Why XML: Standard for Data Exchange

- XML is an standard for data exchange
- With DTD/XML Schema, an XML file can be validated
- XML data is self described



What do these values mean?

Why XML: Standard for Data Exchange



OR CSV, TXT, etc. ???

What if the data is binary?

If the table is stored in Oracle, can you simply send the table?

Why XML: Standard for Data Exchange

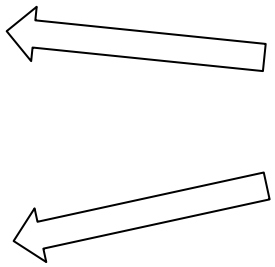
- XML data is self described

Site 2

```
<Employee>
<EmpNo>99123</EmpNo>
<Name>John</Name>
<Salary>10000</Salary>
</Employee>
```

Site 3

```
<Employee>
<EmpNo>90123</EmpNo>
<Name>Mary</Name>
<Salary>15000</Salary>
</Employee>
```

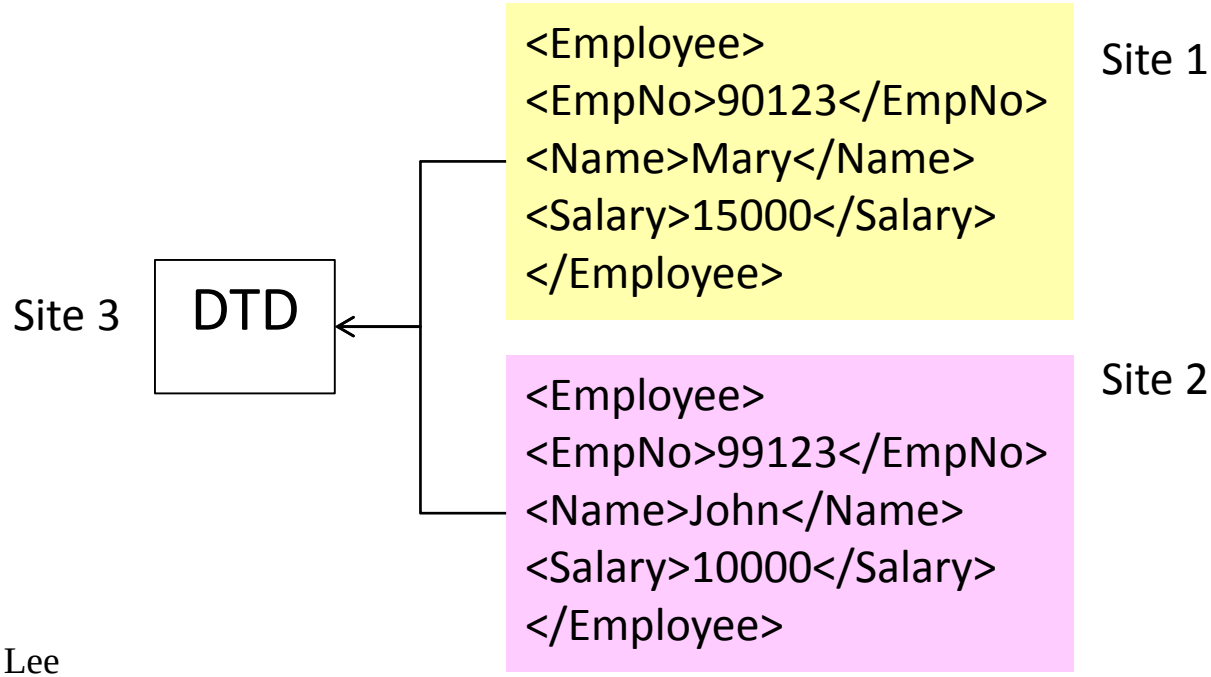


Site 1

EmpNo	Name	Salary
99123	John	10000
90123	Mary	15000

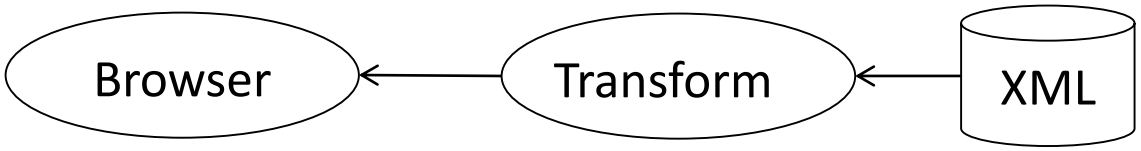
Why XML: Standard for Data Exchange

- XML data is Unicode based, thus supporting multiple languages in the same file
- By sharing the same DTD, a site can validate the XML data received from another site before using it



Why XML: Availability of XML tools

- Many tools are available for the processing of XML data: Java XML, XML DOM, XSLT, SAX, PHP-XML, etc.
- If XML data is generated by another program, you may want to validate it against the DTD

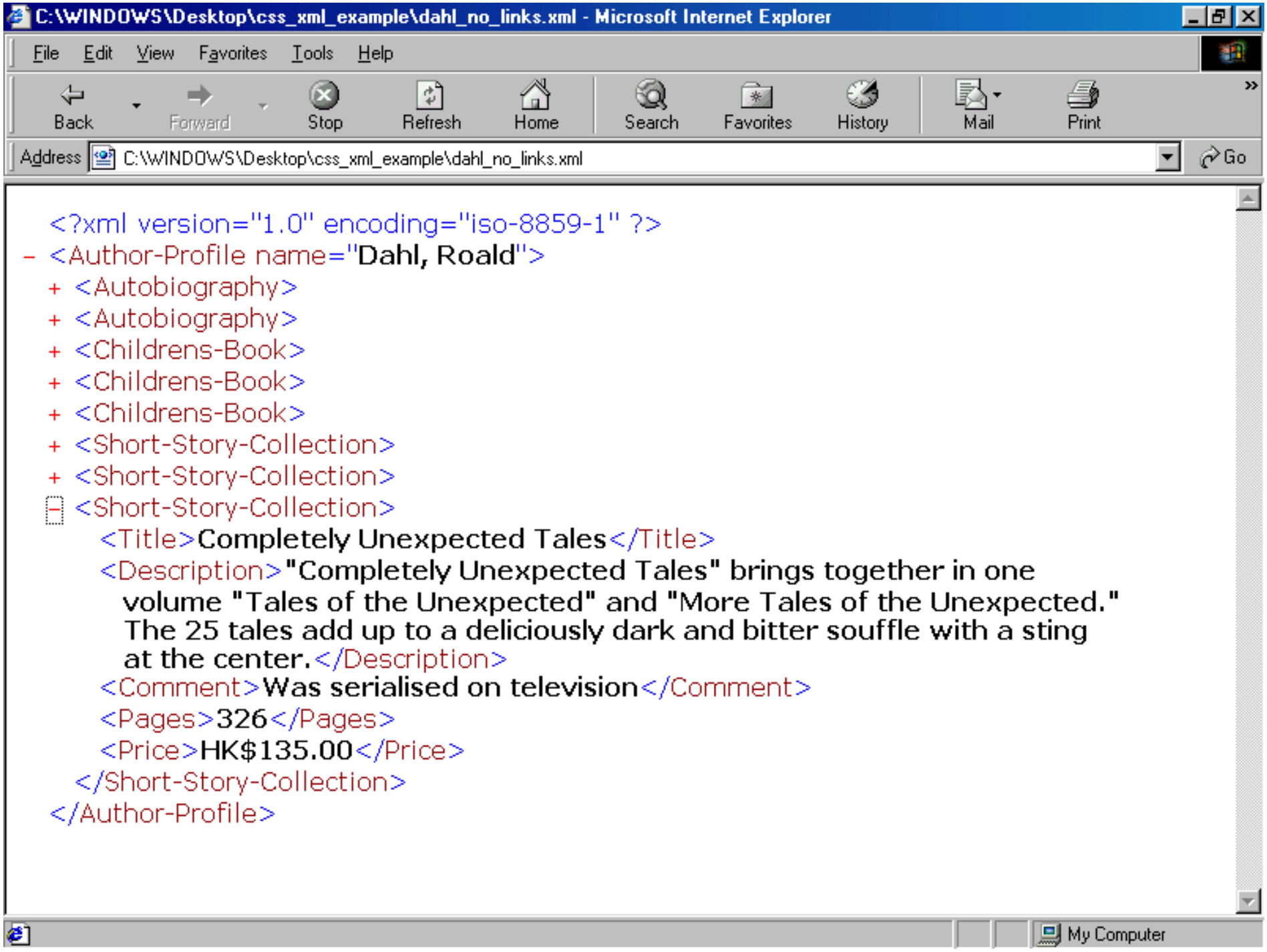


How To Render/Display XML?

- Some possibilities for handling XML:
  1) Give it to IE to display
  2) Use a CSS file to render the XML
  3) Use JavaScript to convert the XML
  4) Use a XSLT file to convert the XML

Method 1) IE Display of XML

- An XML file by itself has no display parameters
- If you give a pure XML file (which has no CSS or XSLT) to IE it will show the file using a tree structure display
  - Example on next page
  - Can hide branches by clicking on the '-'


C:\WINDOWS\Desktop\css_xml_example\dahl_no_links.xml - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Forward Stop Refresh Home Search Favorites History Mail Print
Address C:\WINDOWS\Desktop\css_xml_example\dahl_no_links.xml
Go
<?xml version="1.0" encoding="iso-8859-1" ?>
- <Author-Profile name="Dahl, Roald">
+ <Autobiography>
+ <Autobiography>
+ <Childrens-Book>
+ <Childrens-Book>
+ <Childrens-Book>
+ <Short-Story-Collection>
+ <Short-Story-Collection>
- <Short-Story-Collection>
<Title>Completely Unexpected Tales</Title>
<Description>"Completely Unexpected Tales" brings together in one volume "Tales of the Unexpected" and "More Tales of the Unexpected." The 25 tales add up to a deliciously dark and bitter souffle with a sting at the center.</Description>
<Comment>Was serialised on television</Comment>
<Pages>326</Pages>
<Price>HK$135.00</Price>
</Short-Story-Collection>
</Author-Profile>
My Computer

Method 2) XML and CSS

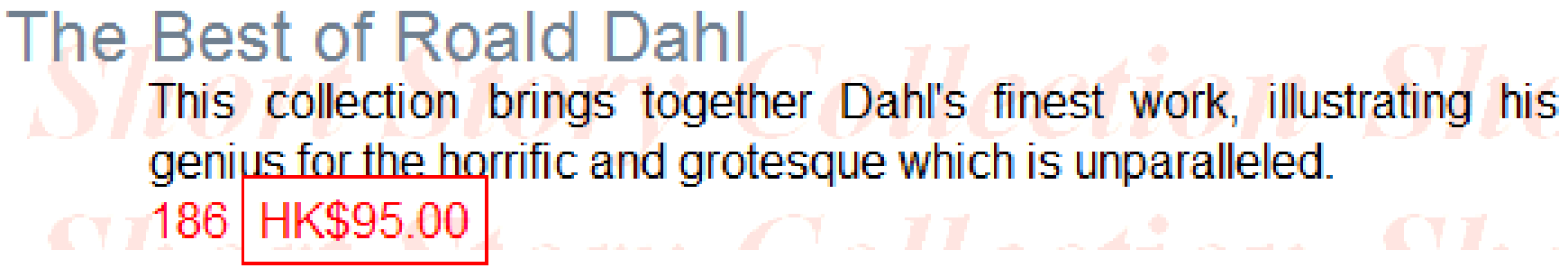
- Use a style sheet file to define the display style for each tag

```
<Short-Story-Collection>
    <Title>The Best of Roald Dahl</Title>
    <Description>
    This collection brings together Dahl's finest work, illustrating
    his genius for the horrific and grotesque which is unparalleled.
    </Description>
    <Pages>186</Pages>
    <Price>HK$95.00</Price>
</Short-Story-Collection>
. . .
```

Example XML+CSS - The CSS

```
Short-Story-Collection { background:url(short_story.png); }
Title {
    display:block; margin-top:1em;
    font-size: 18pt; color:slategray; }
Description {
    display:block; color:black; text-align:justify; margin-left: 3em; }
Pages {
    color:red; text-align:right; text-indent: 3em; }
Price {
    color:red; text-align:right; border:1px solid red; padding:5px; }
```

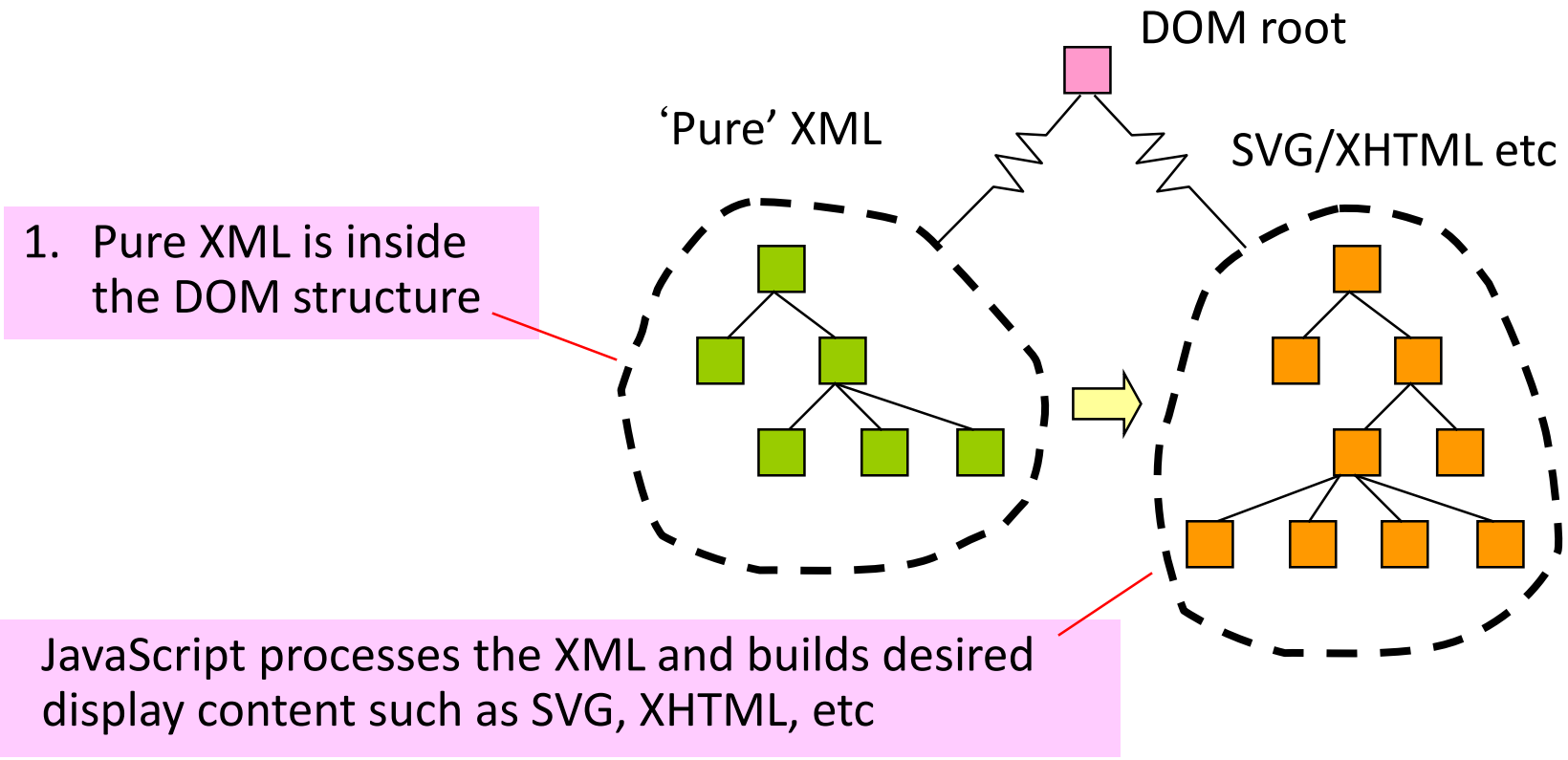
Example XML+CSS - The Result



CSS - Limitations

- So XML+CSS works well
- But what if you want more, for example:
  - You want ‘Pages:’ in front of the page count
  - You want ‘Price:’ in front of the price
  - You want the data sorted in alphabetical order
  - You want the XML displayed as SVG
- CSS can’t do any of these things
  - need method 3 or 4

Method 3) Use JavaScript



- This approach can also be used by VBScript (in a web page), ActionScript (in Flash), and Java (i.e. in an applet)

Conversion of XML to HTML Using JavaScript

```
var html = "";
var list = xmlDoc.getElementsByTagName("Short-Story-Collection");
for (var i = 0; i < list.length; i++) {
    var el = list.item(i);
    html += "<div class='Short-Story-Collection'>";
    . . .
    html += "<span class='Price'>";
    html += "Price: " +
        el.getElementsByTagName("Price").item(0).firstChild.nodeValue;
    html += "</div>";
    . . .
    html += "</div>";    }
. . .
document.body.innerHTML = html;
```

In this way you have total control over the output of the conversion

Example Result

The Best of Roald Dahl

This collection brings together Dahl's finest work, illustrating his genius for the horrific and grotesque which is unparalleled.

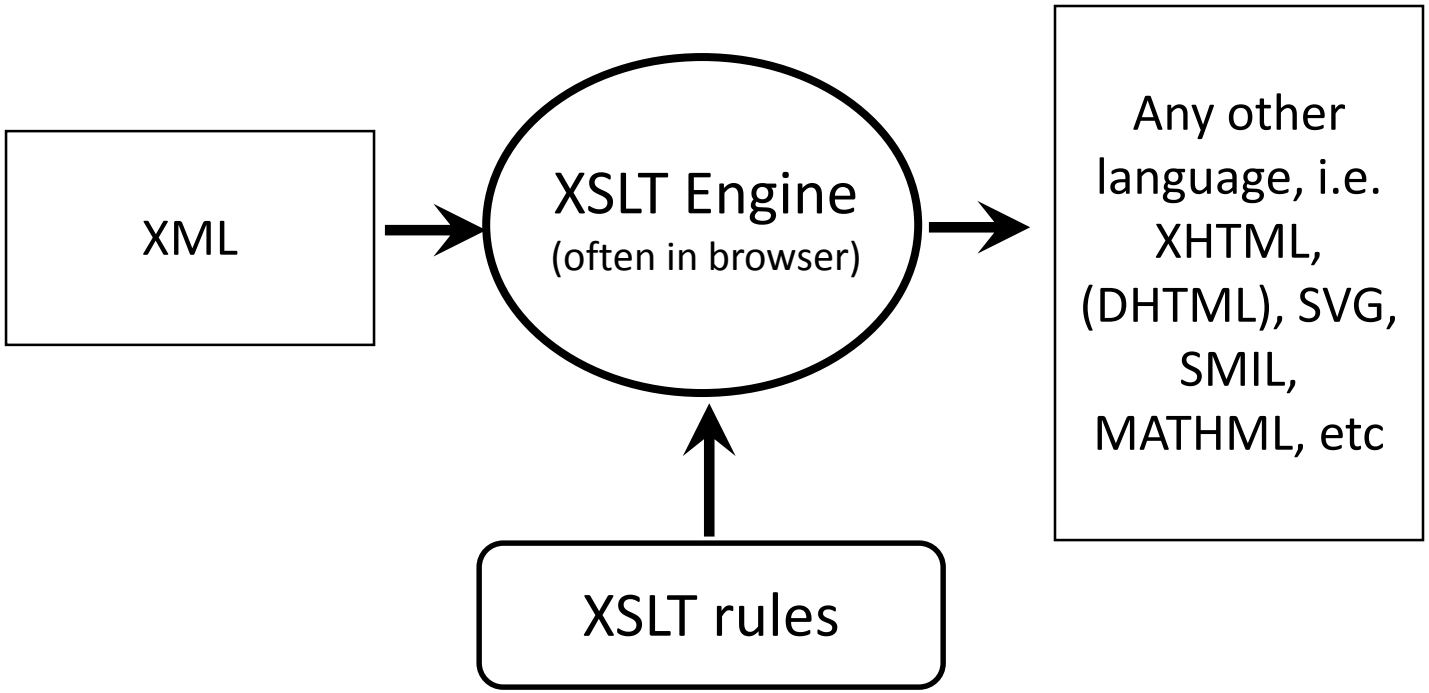
Pages: 186 Price: HK$95.00

Method 4) XSL/ XSLT

- XSL =Extensible Stylesheet Language
- XSL is a group of recommendations for handling XML
- XSLT=XSL Transformations
- XSLT is a language for converting XML into other XML documents

XSL/ XSLT

- You can use XSL to change XML into almost anything



Example 1

01_simple.xsl

```
<?xml version="1.0"?>
<xsl:stylesheet version="1.0" xmlns:xsl=
    "http://www.w3.org/1999/XSL/Transform">
    <xsl:output method="html"/>
    <xsl:template match="/">
        <html> <body>
            <xsl:apply-templates/>
        </body> </html>
    </xsl:template>
</xsl:stylesheet>
```

HTML will be generated

Generate this html at the start

Keep applying the rules for anything else found in the XML document

Example 1 Outputs

- XML input

```
<document>
This is a simple mapping of xml to html by
    using xsl transformation.
</document>
```

- HTML output

```
<html>
<body>
This is a simple mapping of xml to html by
    using xsl transformation.
</body>
</html>
```

Example 2

- The XML input

```
<?xml version="1.0" encoding="iso-8859-1"?>
<?xml-stylesheet type="text/xsl" href="02_two_levels.xsl"?>
<document>
    <title>02_two_levels</title>
    <content>This is a simple mapping of xml to html by using xsl
    transformation.</content>
</document>
```

Improved Rules

02_two_levels.xsl

```
<xsl:template match="document">
   <html> <head> <xsl:apply-templates select="title" /> </head>
      <body> <xsl:apply-templates select="content" /> </body>
   </html>
</xsl:template>

<xsl:template match="title">
        <title><xsl:apply-templates/></title>
</xsl:template>
<xsl:template match="content">
        <p><xsl:apply-templates/></p>
</xsl:template>
```

Example 2 HTML Output

```
<html>
<head>
    <meta http-equiv="Content-Type" content="text/html;
    charset=utf-8">
    <title>02_two_levels</title>
</head>
<body>
    <p>This is a simple mapping of xml to html by using xsl
    transformation.</p>
</body>
</html>
```

Example 3

- The XML input

```
<?xml version="1.0" encoding="iso-8859-1"?>
<?xml-stylesheet type="text/xsl" href="04_value_of.xsl"?>
<course>
    . . .
    <instructor>
        <surname>Rossiter</surname>
        <firstname>David</firstname>
    </instructor>
</course>
```

Improved Rules - Get XML Value

```
. . .
<xsl:template match="instructor">
    <b>Instructor:</b>
    <xsl:value-of select="surname" />
    ,
    <xsl:value-of select="firstname" />
</xsl:template>
. . .
```

04_value_of.xsl

Example 3 HTML Output

```
<html>
  <body>
    <hr>
                . . .
                <b>Instructor: </b>
                Rossiter, David
    <hr>
  </body>
</html>
```

Example 4

- The XML input

```
<?xml version="1.0" encoding="iso-8859-1"?>
<?xml-stylesheet type="text/xsl" href="05_foreach.xsl"?>
<department>
    <name>Computer Science</name>
    <course>
        <code>COMP303</code> <title>Internet Computing</title>
    </course>
    <course>
        <code>COMP336</code> <title>Information Retrieval</title>
    </course>
```

Improved Rules - For-each Loop

```
. . .
<xsl:template match="department">
    <h1><xsl:apply-templates select="name"/></h1>
    <table border="1">
        <tr><th>COURSE CODE</th><th>TITLE</th></tr>
        <xsl:for-each select="course">
         <tr>
         <td><xsl:apply-templates select="code"/></td>
         <td><xsl:apply-templates select="title"/></td>
         </tr>
        </xsl:for-each>
    </table>
</xsl:template>
. . .
```

05_foreach.xsl

Example 4 HTML Output

```
<html>  <body>
        <h1>Computer Science</h1>
        <table border="1">
         <tr><th>COURSE CODE</th><th>TITLE</th></tr>
         <tr>
                <td>COMP303</td>
                <td>Internet Computing</td>
         </tr>
         <tr>
                <td>COMP336</td>
                <td>Information Retrieval</td>
         </tr>
         . . .
```

XML Other Things

- An XML declaration is recommended, but not required
  - ISO-8859-1 character set includes all of the characters used by most Western European languages[default is UTF-8
  - Standalone specifies this document requires reading other files

```
<?xml version="1.0" encoding="ISO-8859-1" standalone="no"?>
```

- Comments
- Processing Instructions (PIs)
- Entities
  - &dw will be replaced with developmentWorks
  - Predefined entities: &lt &gt &quot &apos &amp

```
<!-- Here's a PI for Cocoon: -->
<?cocoon-process type="sql"?>
```

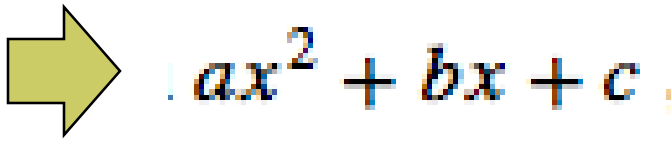
```
<!-- Here's an entity: -->
<!ENTITY dw "developerWorks">
```

Namespaces

- Different languages define their own names, e.g., HTML, SVG, MathML, etc.
- If you only use only one language on a web page, each name is unique defined by that language
- But if you use two or more of languages at the same time, a name may have conflicting definitions in those languages so confusion may arise
- Remember that there are many XML-based languages in this world that you are not aware of, so the tags you use may conflict with other languages

MathML Example

```
<?xml version="1.0" encoding="UTF-8"?>
 <!DOCTYPE math PUBLIC "-//W3C//DTD MathML 2.0//EN"
        "http://www.w3.org/Math/DTD/mathml2/mathml2.dtd">
 <math xmlns="http://www.w3.org/1998/Math/MathML">
  <mrow>
    <mi>a</mi>
    <mo>&#x2062;<!-- &InvisibleTimes; --></mo>
    <msup>
      <mi>x</mi>
      <mn>2</mn>
    </msup>
    <mo>+</mo>
                    <mi>b</mi>
                      <mo>&#x2062;<!-- &InvisibleTimes; ></mo>
                      <mi>x</mi>
                      <mo>+</mo>
                      <mi>c</mi>
                    </mrow>  </math>
```



Some More MathML Examples

$$\int\!\!\int_{\Delta S} (\nabla \times \mathbf{A}) \cdot \mathbf{n}\, dS \qquad \int\!\!\int\!\!\int_{V} \Psi \nabla^2 \phi \; dV \qquad \int \cdots \int f(x_1, \ldots, x_k)\, dx_1 \ldots dx_k$$

$$a_1 + a_2 + \cdots + a_n = \sum_{i=1}^{n} a_i \qquad L_1(x) \leq \cdots \leq f(x) \cdots \leq U_2(x) \leq U_1(x) \qquad B \xrightarrow[H^+]{130\,^\circ C} C$$

$$f'(a) = \lim_{h \to 0} \frac{f(a+h) - f(a)}{h} \qquad \underline{e}_i = \frac{\partial x^k}{\partial y^i} \underline{E}_k \qquad H(j\omega) = \begin{cases} e^{-j\omega t_0} & \text{for } |\omega| < \omega_c \\ 0 & \text{for } |\omega| > \omega_c \end{cases}$$

$$\left\langle \frac{\mathbf{p}^2}{m} \right\rangle = \langle \mathbf{r} \cdot \nabla V(r) \rangle \qquad \left| \frac{f(x + \delta_m) - f(x)}{\delta_m} \right| \geq \zeta \qquad 1 = \sum_n |u_n\rangle \langle u_n| \qquad \lim_{n \to \infty} p_n$$

$$\limsup_{n \to \infty} \sqrt[n]{c_n} \leq \alpha$$

$$\overrightarrow{AB} + \overrightarrow{BC} = \overrightarrow{AC}$$

$$\begin{pmatrix} a_{11} - \lambda & \cdots & a_{1n} \\ \vdots & \ddots & \vdots \\ a_{n1} & \cdots & a_{nn} - \lambda \end{pmatrix} \begin{pmatrix} x_1 \\ \vdots \\ x_n \end{pmatrix} = \mathbf{0}$$

$$\overline{(\Delta n)^2} \equiv \overline{(n - \overline{n})^2}$$

$$(\mathbf{A} \times \mathbf{B})_\alpha = \varepsilon_{\alpha\beta\gamma} A_\beta B_\gamma$$

Name Conflict and Namespaces

- HTML and MathML may have some identical tags so using them at the same time (i.e. in the same web page) may cause confusion
  - E.g., HTML has a div element for a rectangular area but MathML may have a div element for division
  - How does the browser know which language a 'div' in a web page corresponds to?
- The solution is for each language to have a namespace which defines the valid nodes for that language
- Web page content can say exactly which namespace it is using

XML Namespace

- Three namespace prefixes defined below: addr, books, mortgate, each defining it own set of valid elements (not shown in the example)
- Sub-elements inherit namespace from parent elements

```
<?xml version="1.0"?>
<customer_summary
  xmlns:addr="http://www.xyz.com/addresses/"
  xmlns:books="http://www.zyx.com/books/"
  xmlns:mortgage="http://www.yyz.com/title/"
>
... <addr:name><title>Mrs.</title> ... </addr:name> ...
... <books:title>Lord of the Rings</books:title> ...
... <mortgage:title>NC2948-388-1983</mortgage:title> ...
```

Example Using Two Namespaces

```
<my_web_page
   xmlns:html   ="http://www.w3.org/1999/xhtml"
   xmlns:mathml
   ="http://www.w3.org/1998/Math/MathML" >
   . . .
   <html:div>
   </html:div>
   . . .
   <mathml:div>
   </mathml:div>
   . . .
</my_web_page>
```

Uses the HTML namespace

Uses MATHML namespace

Some MathML Formula:

$$\frac{2}{x} = \frac{3}{y}$$

Take Home Message

- XML is the foundation of Web languages
- XML is a language that can be used to define new language (including HTML, SVG, etc.)
- XML appears to be bulky but it is good for data exchange across distributed websites (see next set of slides)
- There are many ways to render XML
- XLT means to be an XML language by itself, with specifications to manipulate XML data
  - No longer under development by W3C