



COMP 4021

Internet Computing

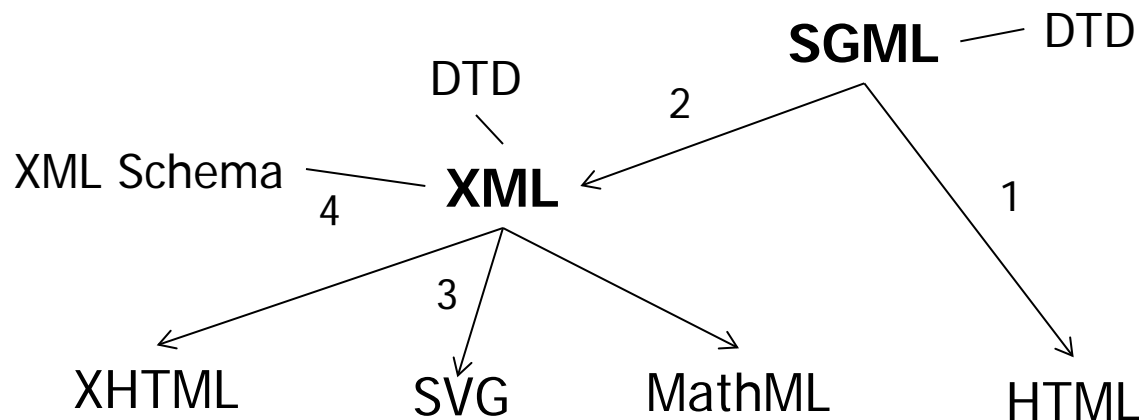
XML and XSL

Dr. Kenneth LEUNG

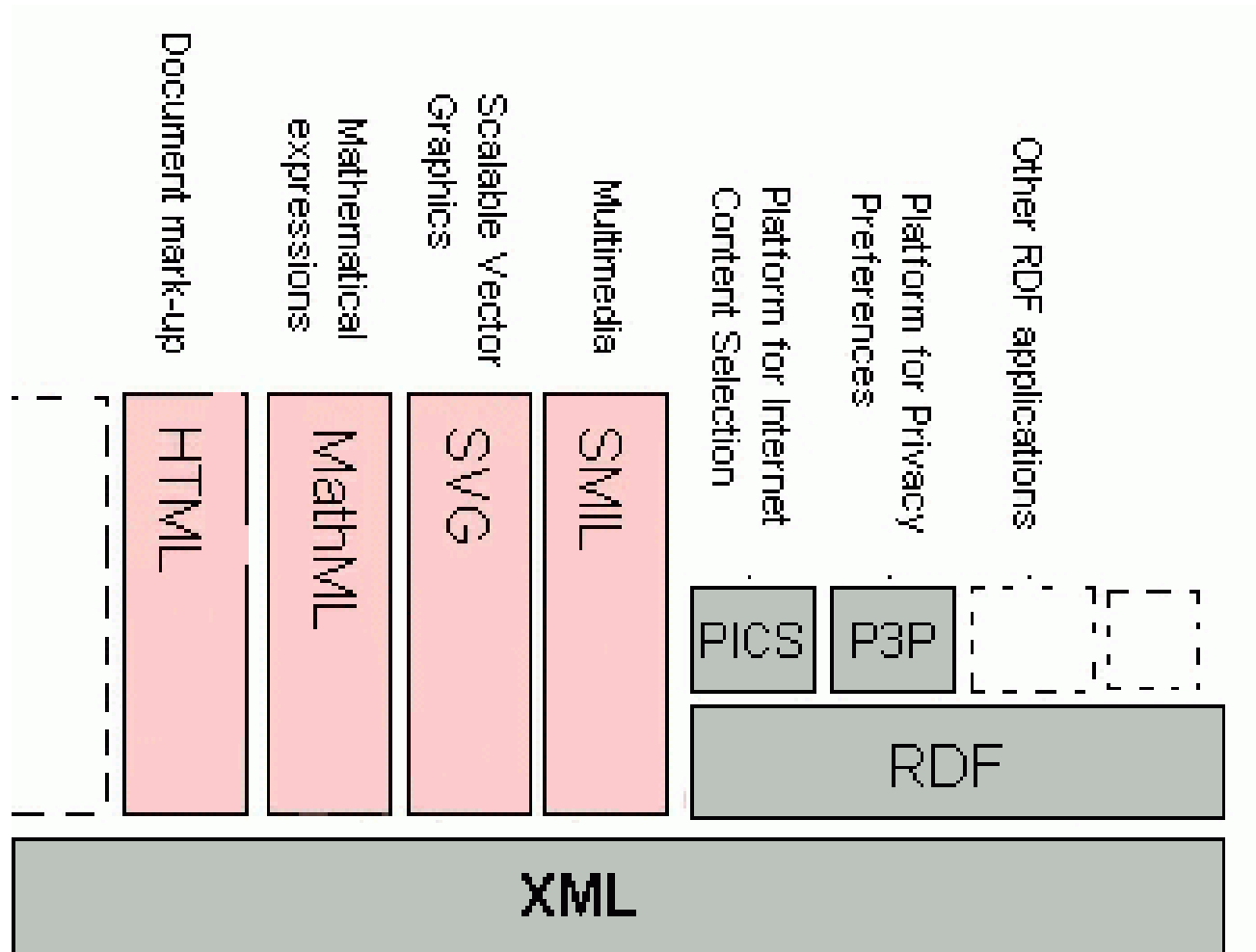
Slides created by Dr. David ROSSITER

History

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

```
<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>
```

Tag: end tag

Attribute:
Name=value
pairs inside
start tags

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>

```

- 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

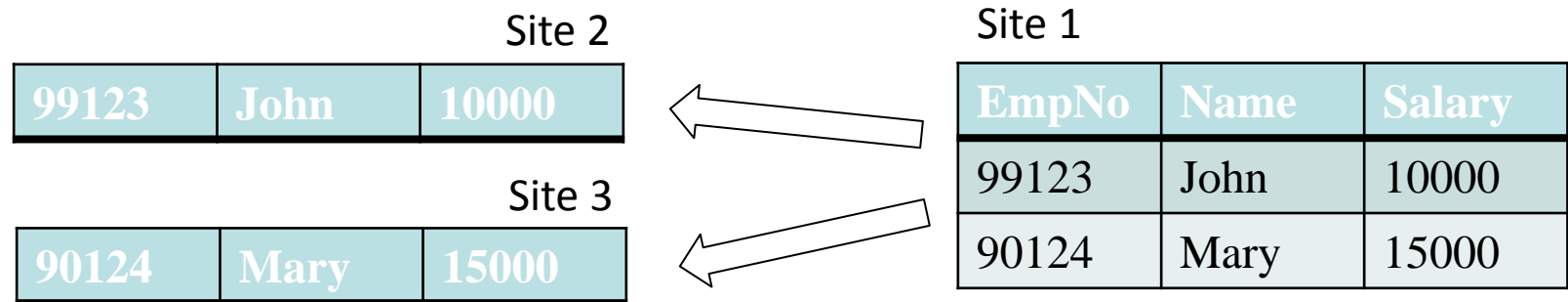
```
<!-- NOT legal XML markup -->  
<ol compact>  
  
<!-- legal XML markup -->  
<ol compact="yes">
```

Parameter values are enclosed in speech marks

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

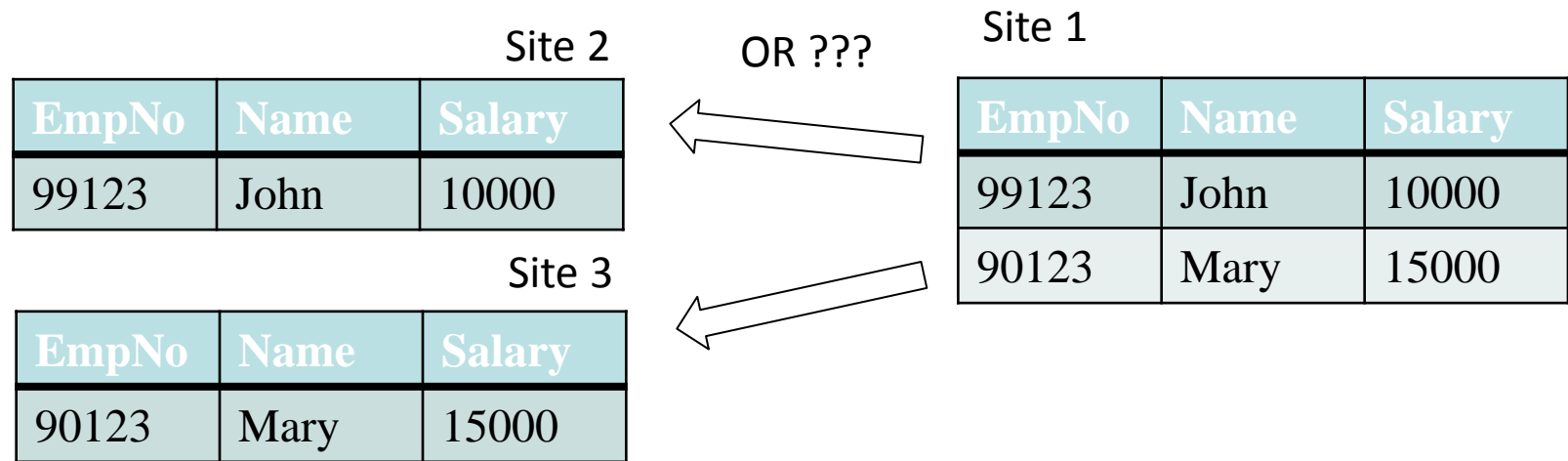
Why XML: Standard for Data Exchange

- 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

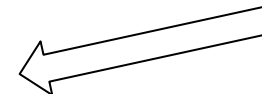
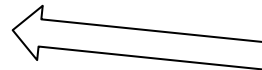
```
<Employee>
<EmpNo>99123</EmpNo>
<Name>John</John>
<Salary>10000</Salary>
</Employee>
```

Site 3

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

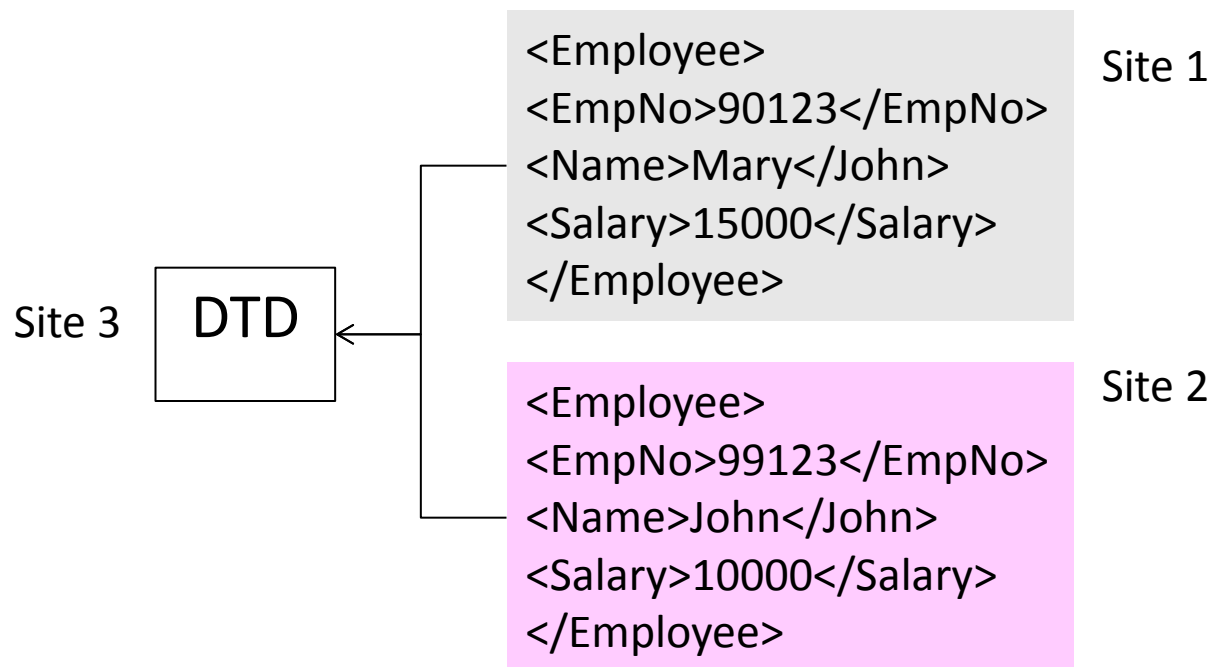
Site 1

EmpNo	Name	Salary
99123	John	10000
90123	Mary	15000



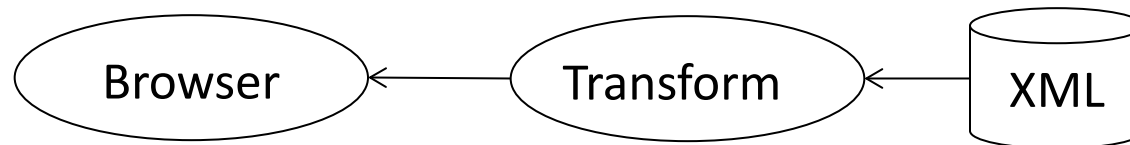
Why XML: Standard for Data Exchange

- 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



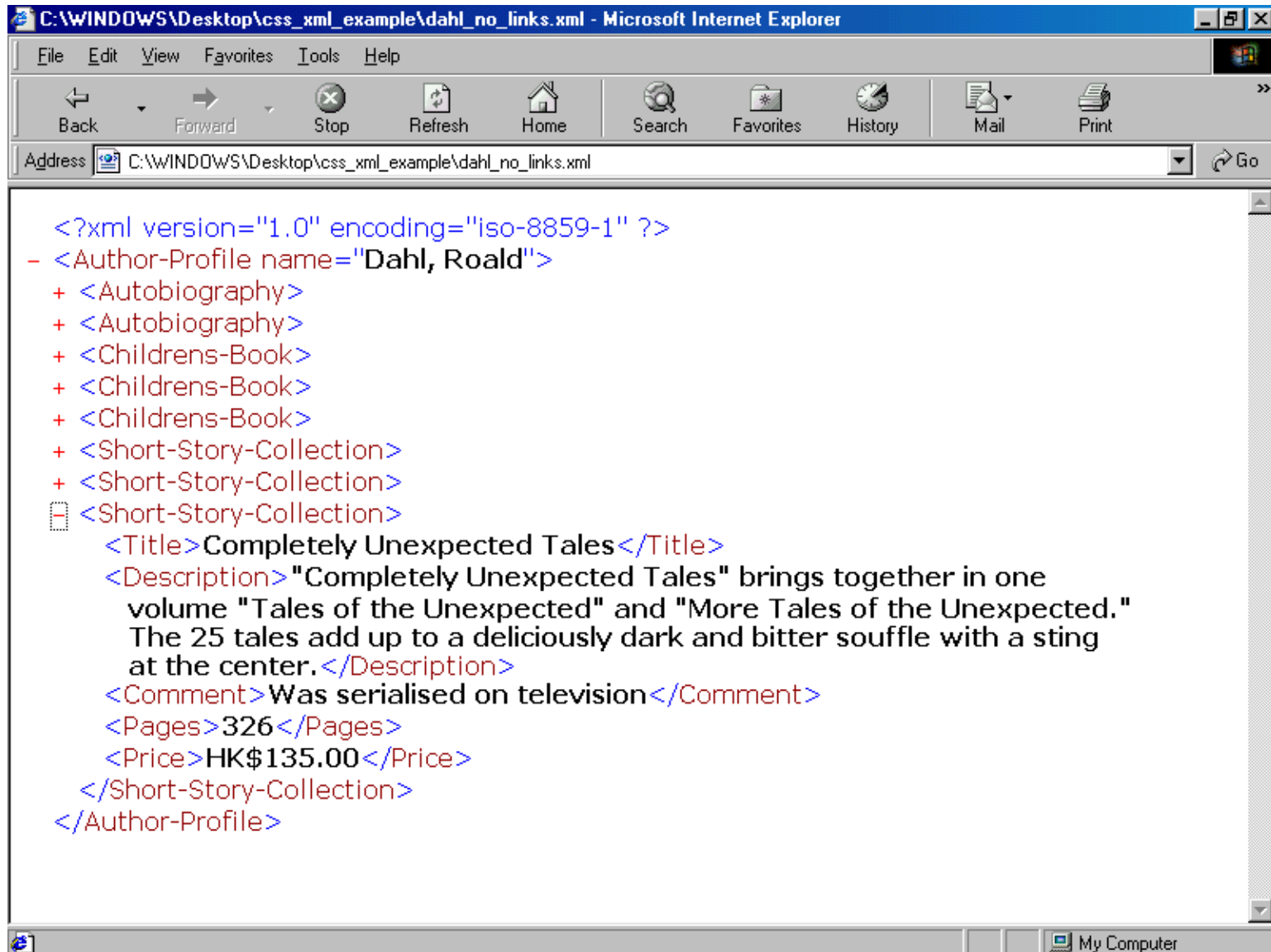
How To Render/Display XML?

- 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 '-'

Method 1) IE Display of XML



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

The Best of Roald Dahl

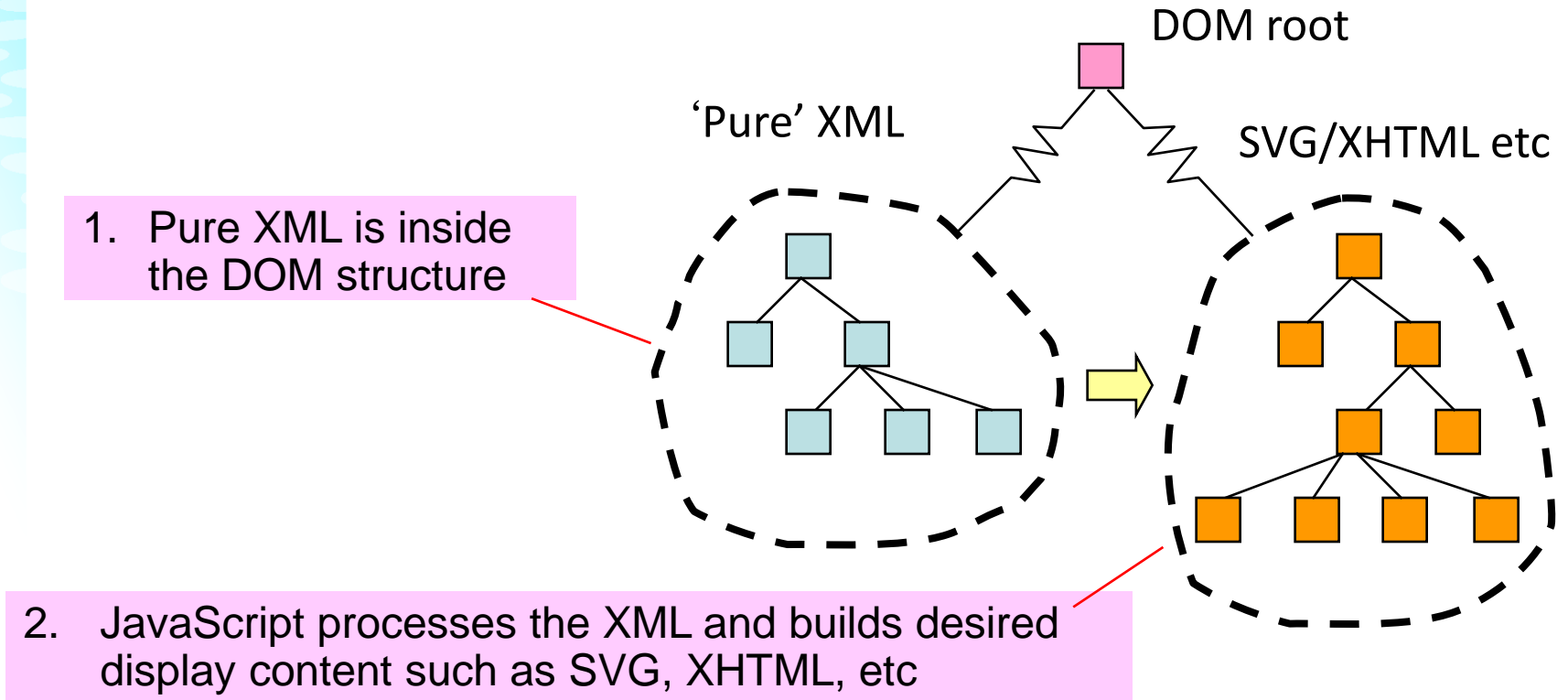
This collection brings together Dahl's finest work, illustrating his genius for the horrific and grotesque which is unparalleled.

186 HK\$95.00

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.

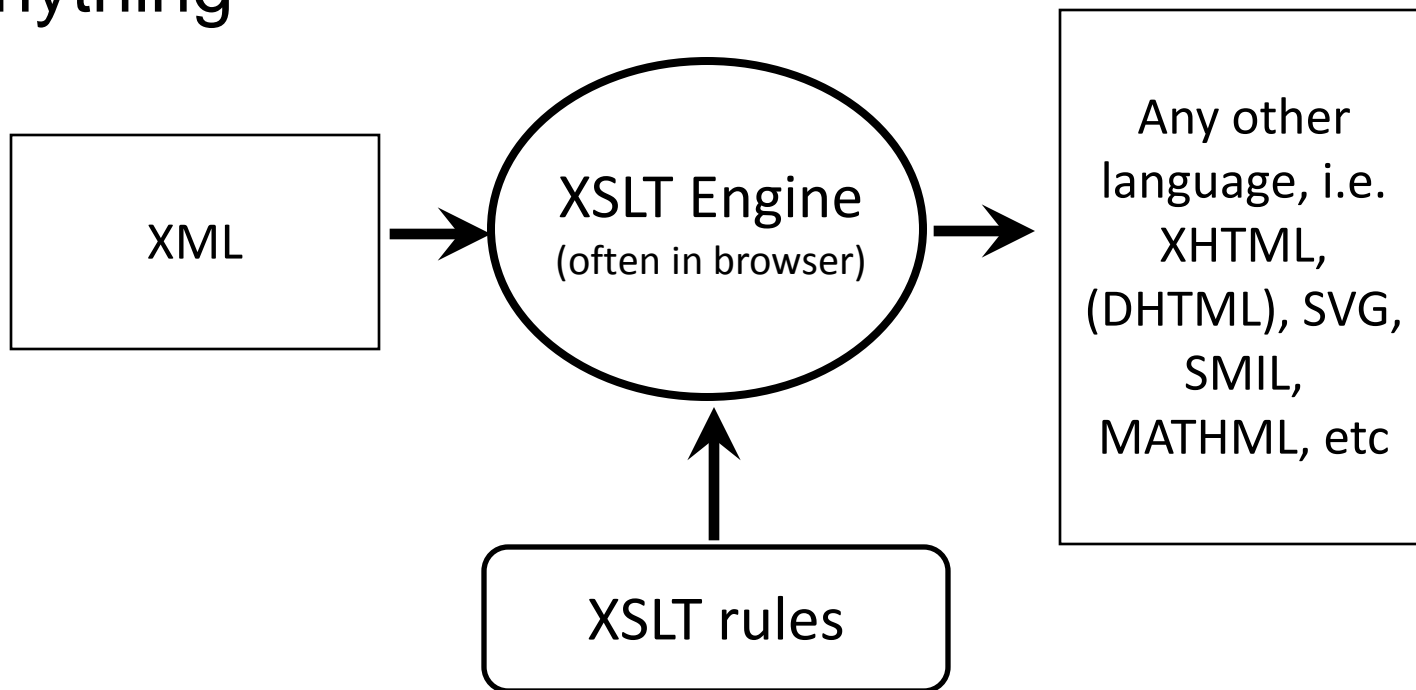
Pages: 186 Price: HK\$95.00

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"/> } HTML will be generated
```

```
<xsl:template match="/">
```

```
<html> <body> } Generate this
```

```
<xsl:apply-templates/>
```

```
</body> </html> }
```

```
</xsl:template>
```

```
</xsl:stylesheet>
```

*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="03_create_element.xsl"?>
```

```
<course>
```

```
  <code>COMP303</code>
```

```
  <title>Internet Computing</title>
```

```
  <url>http://www.cs.ust.hk/course/comp303</url>
```

```
</course>
```

Improved Rules - Adding Attributes

```
<xsl:template match="url">
  <xsl:element name="p">
    <xsl:element name="a">
      <xsl:attribute name="href">
        <xsl:apply-templates />
      </xsl:attribute>
      course information
    </xsl:element>
  </xsl:element>
</xsl:template>
...
```

03_create_element.xsl

Example 3 HTML Output

```
<html> <body>  
<hr>  
<pre>COMP303</pre>  
<h3>Internet Computing</h3>  
<p>  
  <a href="http://www.cs.ust.hk/course/comp303">  
    course information </a>  
</p>  
<hr>  
</body> </html>
```

Example 4

- 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 4 HTML Output

```
<html>
```

```
<body>
```

```
<hr>
```

...

```
<b>Instructor: </b>
```

```
Rossiter, David
```

```
<hr>
```

```
</body>
```

```
</html>
```

Example 5

- 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 5 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>
    ...
```

Example 6

- The XML input

```
<?xml version="1.0" encoding="iso-8859-1"?>
<?xml-stylesheet type="text/xsl" href="06_sorting.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 - Sorting

. . .

```
<xsl:for-each select="course">
```

```
  <xsl:sort select="title" />
```

```
  <tr>
```

```
    <xsl:apply-templates select="title" />
```

```
    <xsl:apply-templates select="code" />
```

```
  </tr>
```

```
</xsl:for-each>
```

. . .

06_sorting.xsl

Example 6 HTML Output

```
<table border="1">
```

```
  <tr>
```

```
    <th>TITLE (sorted)</th><th>COURSE CODE</th>
```

```
  </tr>
```

```
  <tr>
```

```
    <td>Computer Graphics</td>
```

```
    <td>COMP341</td>
```

```
  </tr>
```

```
  <tr>
```

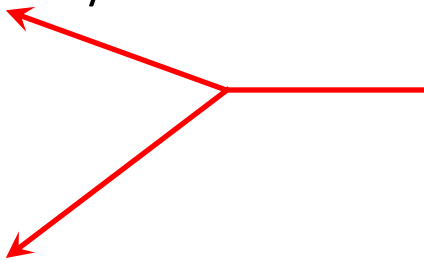
```
    <td>Fundamentals of Multimedia Computing</td>
```

```
    <td>COMP343</td>
```

```
  </tr>
```

```
  ...
```

*Sorted by the title
of the courses*



Example 7

- The XML input

```
<?xml version="1.0" encoding="iso-8859-1"?>
<?xml-stylesheet type="text/xsl" href="07_variable.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 - Use of Variables

...

```
<xsl:apply-templates select="department">
```

```
  <xsl:with-param name="sort-key">
```

```
    code
```

```
  </xsl:with-param>
```

```
</xsl:apply-templates>
```

```
<xsl:apply-templates select="department">
```

```
  <xsl:with-param name="sort-key">
```

```
    title
```

```
  </xsl:with-param>
```

```
</xsl:apply-templates>
```

...

07_variable.xsl

Improved Rules - Use of Variables

<xsl:choose>

<xsl:when test="\$sort-key='code'">

 <xsl:apply-templates select="course">

 <xsl:sort select="code"/>

 </xsl:apply-templates>

 </xsl:when>

<xsl:when test="\$sort-key='title'">

 <xsl:apply-templates select="course">

 <xsl:sort select="title"/>

 </xsl:apply-templates>

 </xsl:when>

</xsl:choose>

07_variable.xsl

Example 7 HTML Output

`<table border="1">` Sorted by course title

COMP341	Computer Graphics
COMP343	Fundamentals of Multimedia Computing
...	...

`</table>`

`<table border="1">` Sorted by course code

COMP303	Internet Computing
COMP336	Information Retrieval
...	...

`</table>`