CSC309 Tutorial

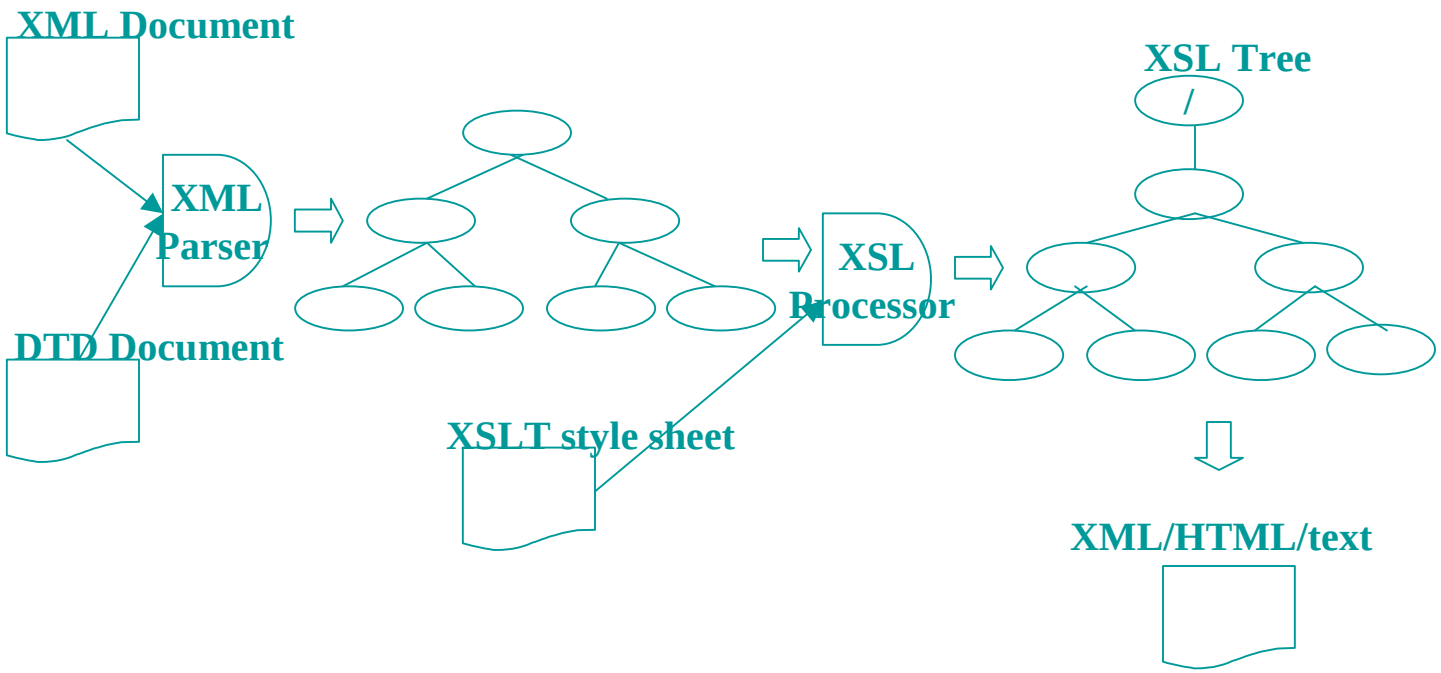
XSL

Lei Jiang

Feb. 3, 2003

XML Document as a Tree

- In an XSL transformation
  - the processor reads both an source XML document and an XSLT style sheet
  - and outputs a new XML/HTML/text document (or fragment of document).



XML Document as a Tree (cont'd)

- XSLT processors model an XML document as a tree that contains seven kinds of nodes:
  - The root (not the same as the root element!)
  - Elements
  - Attributes
  - Text
  - Namespaces
  - Processing instructions
  - Comments
- Document Type Definition (DTD) and document type declaration (DocType) are not included in this tree.

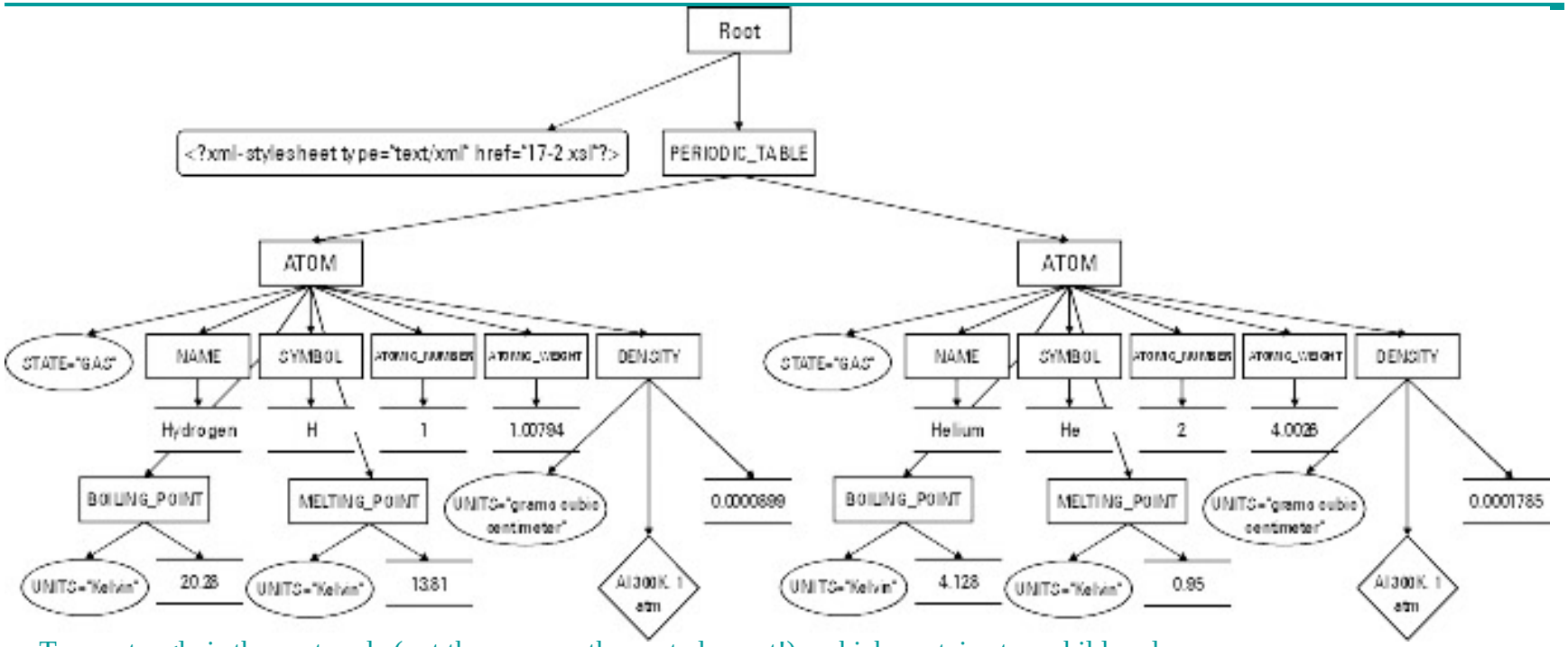
XML Document as a Tree (cont'd)

```
<?xml version="1.0"?>
<?xml-stylesheet type="text/xml" href="17-2.xsl"?>
<PERIODIC_TABLE>

 <ATOM STATE="GAS">
  <NAME>Hydrogen</NAME>
  <SYMBOL>H</SYMBOL>
  <ATOMIC_NUMBER>1</ATOMIC_NUMBER>
  <ATOMIC_WEIGHT>1.00794</ATOMIC_WEIGHT>
  <BOILING_POINT
      UNITS="Kelvin">20.28</BOILING_POINT>
  <MELTING_POINT
      UNITS="Kelvin">13.81</MELTING_POINT>
  <DENSITY UNITS="grams/cubic centimeter">
    <!-- At 300K, 1 atm -->
    0.0000899
  </DENSITY>
 </ATOM>
 <ATOM STATE="GAS">
  <NAME>Helium</NAME>
  <SYMBOL>He</SYMBOL>
  <ATOMIC_NUMBER>2</ATOMIC_NUMBER>
  <ATOMIC_WEIGHT>4.0026</ATOMIC_WEIGHT>
  <BOILING_POINT
      UNITS="Kelvin">4.216</BOILING_POINT>
  <MELTING_POINT
      UNITS="Kelvin">0.95</MELTING_POINT>
  <DENSITY UNITS="grams/cubic centimeter"><!-- At
      300K -->
    0.0001785
  </DENSITY>
 </ATOM>
</PERIODIC_TABLE>
```

- The root PERIODIC_TABLE element contains ATOM child elements.
- Each ATOM element contains several child elements providing the atomic number, atomic weight, symbol, boiling point, and so forth.
- A UNITS attribute specifies the units for those elements that have units.

XML Document as a Tree (cont'd)



- Top rectangle is the root node (not the same as the root element!), which contains two child node
  - the xml-stylesheet PI
  - the root element PERIODIC_TABL
- The PERIODIC_TABLE element contains two child nodes, both are ATOM elements.
- Each ATOM element has an attribute node for its STATE attribute.
- Each child element contains a node for its contents, as well as nodes for any attributes, comments and processing instructions it possesses.

XML Document as a Tree (cont'd)

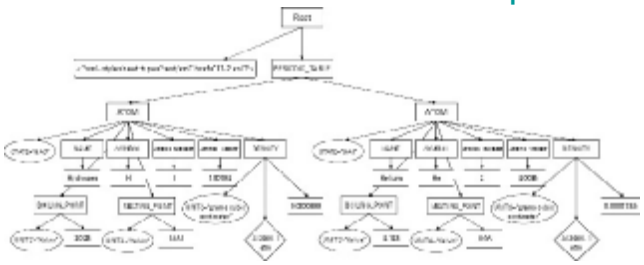
- XSLT operates by transforming one XML tree into another XML tree.

- The XSL transformation language contains operators for
  - selecting nodes from the tree
  - reordering the nodes
  - outputting nodes

- When an processor transforms an XML document,
  - it walks the document tree, looking at each node in turn.
  - As each node, the processor compares it with the pattern of each template rule.
  - When a match is found, it outputs the rule's template.

XSLT Templates

- **xsl:template** elements associate particular output with particular input.
  - **match** attribute specifies which nodes to choose
  - content of the element is the actual template to be instantiated when a match is found



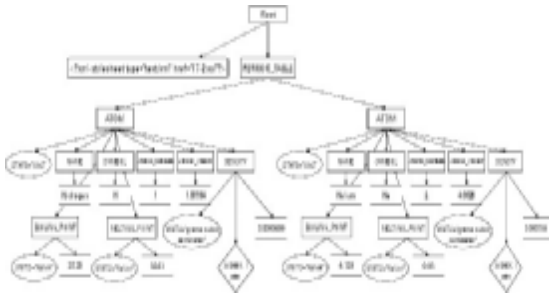
```
<html>
 <head>
 </head>
 <body>
 </body>
</html>
```

```
<xsl:template match="/">
 <html>
  <head>
  </head>
  <body>
  </body>
 </html>
</xsl:template>
```

XSLT Templates (cont'd)

- The **xsl:apply-templates** element
  - To get beyond the root, you have to tell the processor to process the children of the root.
  - **xsl:apply-templates** recursively process the nodes through the XML document.

XSLT Templates (cont'd)



```
<xsl:template match="/">
  <html>
    <xsl:apply-templates/>
  </html>
</xsl:template>

<xsl:template match="PERIODIC_TABLE">
  <body>
    <xsl:apply-templates/>
  </body>
</xsl:template>

<xsl:template match="ATOM">
  An Atom
</xsl:template>
```

1. The root node is compared with all template rules. It matches the first one.
2. The <html> tag is written out.
3. The **xsl:apply-templates** element causes processor to process the *child nodes* of the root node.
   a. The first child, the *xml-stylesheet processing* instruction, is compared with the template rules. But there is not matching rules.
   b. The second child, the root element PERIODIC_TABLE, is compared with the template rules. It matches the second template rule.
   c. The <body> tag is written out.
   d. The xsl:apply-templates element in the body element causes the processor to process the child nodes of PERIODIC_TABLE.
      a) The first child, the Hydrogen ATOM element, is compared with the template rules. It matches the third template rule.
      b) The text "An Atom" is output.
      c) The second, the Helium ATOM element, is compared with the template rules. It matches the third template rule.
      d) The text "An Atom" is output.
   e. The </body> tag is written out.
4. The </html> tag is written out.

XSLT Templates (**cont'**d)

The end result is:

```
<html>
<body>
   An Atom
   An Atom
</body>
</html>
```

XSLT Templates (**cont'**d)

- The **select** attribute
  - Used choose a particular set of children instead of all children.

```
<xsl:template match="ATOM">
 <xsl:apply-templates select="NAME"/>
</xsl:template>
```

Replace the text "An Atom" with the name of the ATOM element

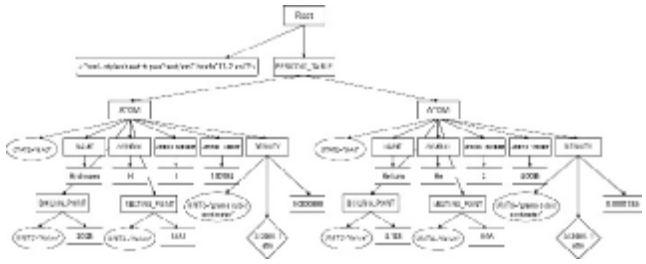
XSLT: value-of

- The **xsl:value-of** element computes the value of something in the input document and copies it into the output document.

```
<xsl:template match="/">
  <html>
    <xsl:apply-templates/>
  </html>
</xsl:template>

<xsl:template match="PERIODIC_TABLE">
  <body>
    <xsl:apply-templates/>
  </body>
</xsl:template>

<xsl:template match="ATOM">
   <xsl:value-of select="NAME"/>
</xsl:template> </xsl:template>
```



```
<ATOM STATE="GAS">
  <NAME>Hydrogen</NAME>
  ……
</ATOM>

<ATOM STATE="GAS">
  <NAME>Helium</NAME>
  ……
</ATOM>
```

XSLT: value-of (cont'd)

- The Result

```
<html>
<body>
 Hydrogen
 Helium
</body>
</html>
```

```
<xsl:template match="/">
  <html>
    <xsl:apply-templates/>
  </html>
</xsl:template>

<xsl:template match="PERIODIC_TABLE">
  <body>
    <xsl:apply-templates/>
  </body>
</xsl:template>

<xsl:template match="ATOM">
   <xsl:value-of select="NAME"/>
</xsl:template> </xsl:template>
```

XSLT: for-each

- Processing Multiple Elements with **xsl:for-each**
  - If there are multiple possible items that could be selected, the xsl:value-of element will choose the first one matched.
  - For example, it is a poor because typical PERIODIC_TABLE element contains more than one ATOM:

```
<xsl:template match="PERIODIC_TABLE">
 <xsl:value-of select="ATOM"/>
</xsl:template>
```

- There are two ways of processing multiple elements
  - **xsl:apply-templates** with a select attribute that chooses the particular elements:

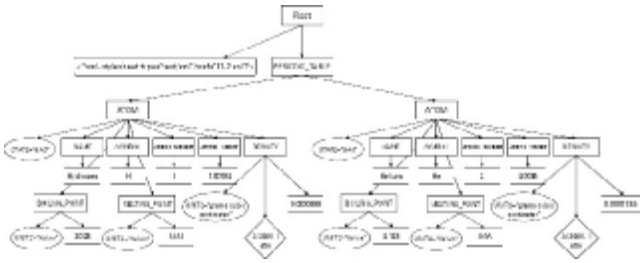
```
<xsl:template match="PERIODIC_TABLE">
  <xsl:apply-templates select="ATOM"/>
</xsl:template>
```

  - **xsl:for-each** processes each element in turn. However, **no additional template is required**.

```
<xsl:template match="PERIODIC_TABLE">
 <xsl:for-each select="ATOM">
   <xsl:value-of select="."/>
 </xsl:for-each>
</xsl:template>
```

XSLT: patterns

- Matching the root node: /



```
<xsl:template match="/">
  <html>
    <head>
      <title>Atomic Number vs. Atomic Weight</title>
    </head>
    <body>
      <table>
        Atom data will go here
      </table>
    </body>
  </html>
</xsl:template>
```

XSLT: patterns (cont'd)

- Wild cards: *
  - Indicate a template matches all elements

```
<xsl:template match="*">
    <P>
      <xsl:value-of select="."/>
    </P>
</xsl:template>
```

  - This template says that all elements should be wrapped in a P element.

XSLT: patterns (cont'd)

- Matching attributes with @
  - it matches against attributes and selects nodes according to attribute names.

```
<xsl:template match="@UNITS">
    <I><xsl:value-of select="."/></I>
</xsl:template>
```

```
<PERIODIC_TABLE>

 <ATOM STATE="GAS">
  ……
  <BOILING_POINT
       UNITS="Kelvin">20.28</BOILING_POINT>
  <MELTING_POINT
       UNITS="Kelvin">13.81</MELTING_POINT>
  <DENSITY UNITS="grams/cubic centimeter">
   <!-- At 300K, 1 atm -->
  ……
```

  - This template rule matches UNITS attributes, and wraps them in an I element.

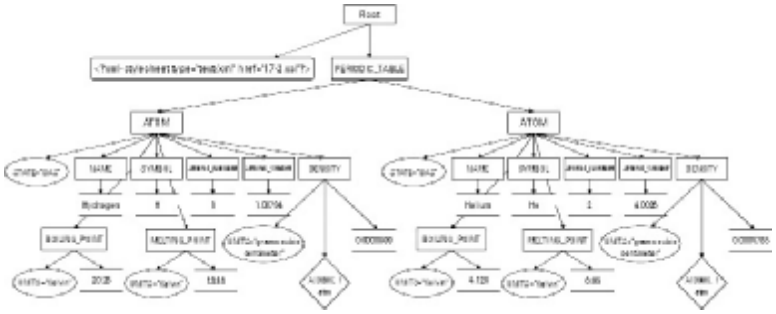
XSLT: patterns (cont'd)

```
<xsl:template match="/PERIODIC_TABLE">
  <html>
    <body>
      <h1>Atomic Number vs. Melting Point</h1>
      <table>
        <th>Element</th>
        <th>Atomic Number</th>
        <th>Melting Point</th>
        <xsl:apply-templates/>
      </table>
    </body>
  </html>
</xsl:template>

<xsl:template match="ATOM">
   <tr>
    <td><xsl:value-of select="NAME"/></td>
    <td><xsl:value-of select="ATOMIC_NUMBER"/></td>
    <td><xsl:apply-templates select="MELTING_POINT"/></td>
  </tr>
</xsl:template>

<xsl:template match="MELTING_POINT">
  <xsl:value-of select="."/>
  <xsl:apply-templates select="@UNITS"/>
</xsl:template>

<xsl:template match="@UNITS">
  <I><xsl:value-of select="."/></I>
</xsl:template>
```



```
<tr>
<td>Hydrogen</td><td>1</td><td>13.81<I>Kelvin</I></td>
</tr>

 <tr>
<td>Helium</td><td>2</td><td>0.95<I>Kelvin</I></td>
</tr>
```

XSLT: patterns (cont'd)

- Matching text nodes with text()
  - Text nodes are normally ignored as nodes
  - **text()** operator does enable you to select the text child of an element

```
<xsl:template match="text()">
    <b><xsl:value-of select="."/></b>
</xsl:template>
```

    - this rule emboldens all text.
  - The main reason this operator exists is for the default rules
  - XSLT processors must provide the following default rule

```
<xsl:template match="text()">
    <xsl:value-of select="."/>
</xsl:template>
```

XSLT: making choices

- **xsl:if** element
  - provides a simple facility for changing the output.

```
<xsl:template match="ATOM">
  <xsl:value-of select="NAME"/>
  <xsl:if test="position()!=last()">, </xsl:if>
</xsl:template>
```

    - In this example, we add a comma and a space to all ATOM elements except the last one.
  - There are no xsl:else or xsl:else-if elements.

XSLT: making choices

- **xsl:choose** element
  - Select one of several outputs depending on conditions
  - In the example, the rule changes the color of the output
    - based on whether the STATE attribute is SOLID, LIQUID, or GAS

```
<xsl:template match="ATOM">
 <xsl:choose>
  <xsl:when test="@STATE='SOLID'">
   <P style="color: black">
    <xsl:value-of select="."/>
   </P>
  </xsl:when>
  <xsl:when test="@STATE='LIQUID'">
   <P style="color: blue">
    <xsl:value-of select="."/>
   </P>
  </xsl:when>
  <xsl:when test="@STATE='GAS'">
   <P style="color: red">
    <xsl:value-of select="."/>
   </P>
  </xsl:when>
  <xsl:otherwise>
   <P style="color: green">
    <xsl:value-of select="."/>
   </P>
  </xsl:otherwise>
 </xsl:choose>
</xsl:template>
```

Example

```
.....
<BOOKLIST>
        <BOOKS>
                <ITEM CAT="S">
                        <TITLE>Number, the Language of Science</TITLE>
                        <AUTHOR>Danzig</AUTHOR>
                        <PRICE>5.95</PRICE>
                        <QUANTITY>3</QUANTITY>
                </ITEM>

                <ITEM CAT="F">
                        <TITLE>Tales of Grandpa Cat</TITLE>
                        <PUBLISHER>Associated Press</PUBLISHER>
                        <AUTHOR>Wardlaw, Lee</AUTHOR>
                        <PRICE>6.58</PRICE>
                        <QUANTITY>5</QUANTITY>
                </ITEM>

                <ITEM CAT="C">
                        <TITLE>Learn Java Now</TITLE>
                        <AUTHOR>Stephen R. Davis</AUTHOR>
                        <PUBLISHER>Microsoft Corporation</PUBLISHER>
                        <PRICE>9.95</PRICE>
                        <QUANTITY>12</QUANTITY>
                </ITEM>

                <ITEM CAT="C" TAX="12.5">
                        <TITLE>Design Patterns</TITLE>
                        <AUTHOR>Erich Gamma, Richard Helm, Ralph Johnson, John
                                Vlissides</AUTHOR>
                        <PUBLISHER>Addison Wesley</PUBLISHER>
                        <PRICE>49.95</PRICE>
                        <QUANTITY>2</QUANTITY>
                </ITEM>
        </BOOKS>

        <CATEGORIES DESC="Miscellaneous categories"> A list of categories
                <CATEGORY CODE="S" DESC="Science"/>
                <CATEGORY CODE="C" DESC="Computing"/>
                <CATEGORY CODE="F" DESC="Fiction"/>
        </CATEGORIES>
</BOOKLIST>
```

Example (cont'd)

```
.....
<xsl:template match="BOOKLIST">
      <xsl:apply-templates select="BOOKS"/>
</xsl:template>

<xsl:template match="BOOKS">
<xsl:text>
      Title,Author,Category
</xsl:text>
<xsl:for-each select="ITEM">
      <xsl:text>
            "<xsl:value-of select="TITLE"/>",
            "<xsl:value-of select="AUTHOR"/>",
            "<xsl:value-of select="@CAT"/>(
      </xsl:text>
      <xsl:choose>
            <xsl:when test='@CAT="F"'>Fiction</xsl:when>
            <xsl:when test='@CAT="S"'>Science</xsl:when>
            <xsl:when test='@CAT="C"'>Computing</xsl:when>
            <xsl:otherwise>Unclassified</xsl:otherwise>
      </xsl:choose>
      <xsl:text>
            )“
      </xsl:text>
</xsl:for-each>
</xsl:template>
.....
```

Example (cont'd)

```
Title,Author,Category
"Number, the Language of Science","Danzig","S(Science)"
"Tales of Grandpa Cat","Wardlaw, Lee","F(Fiction)"
"Learn Java Now","Stephen R. Davis","C(Computing)"
"Design Patterns","Erich Gamma, Richard Helm,JohnVlissides",“C(Computing)"
```