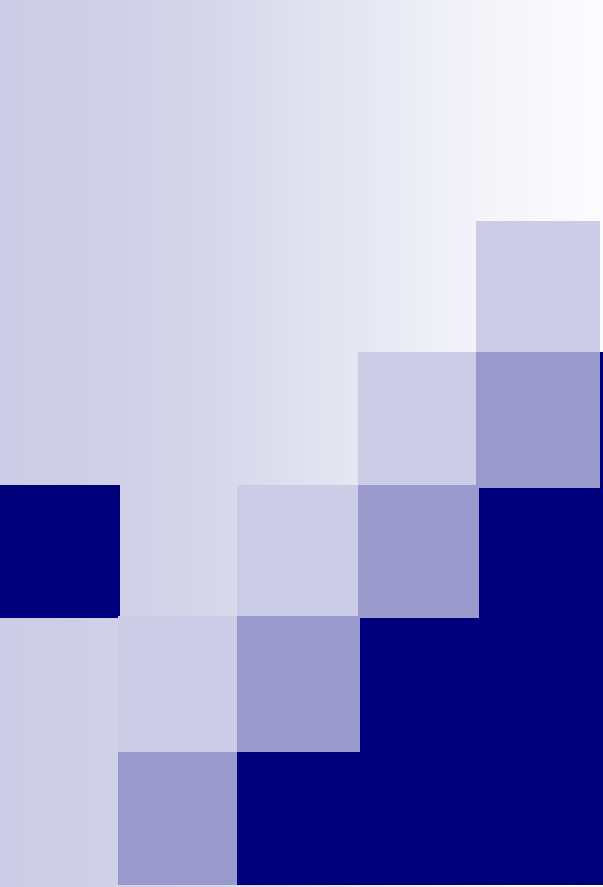




XML



XRay XML Editor

- Available from
 - ☐ <http://www.architag.com>
 - ☐ Download for free



XML ADVANTAGES

■ Extensible

- Add your own tags

```
<color>  
  <red></red>  
  <green></green>  
  <blue></blue>  
</color>
```



XML ADVANTAGES

- Interoperable

- XML has no dependencies on

- Operating System

- Language

- Data source

XML ADVANTAGES

■ Self Describing

- Data describes itself
- Structure is easily identified

```
<employee>  
  <name>Jake</name>  
  <salary>25000</salary>  
  <region>Ohio</region>  
</employee>
```



XML DISADVANTAGES

- Main Disadvantage

- Well structured, but no 'markup' or manipulation

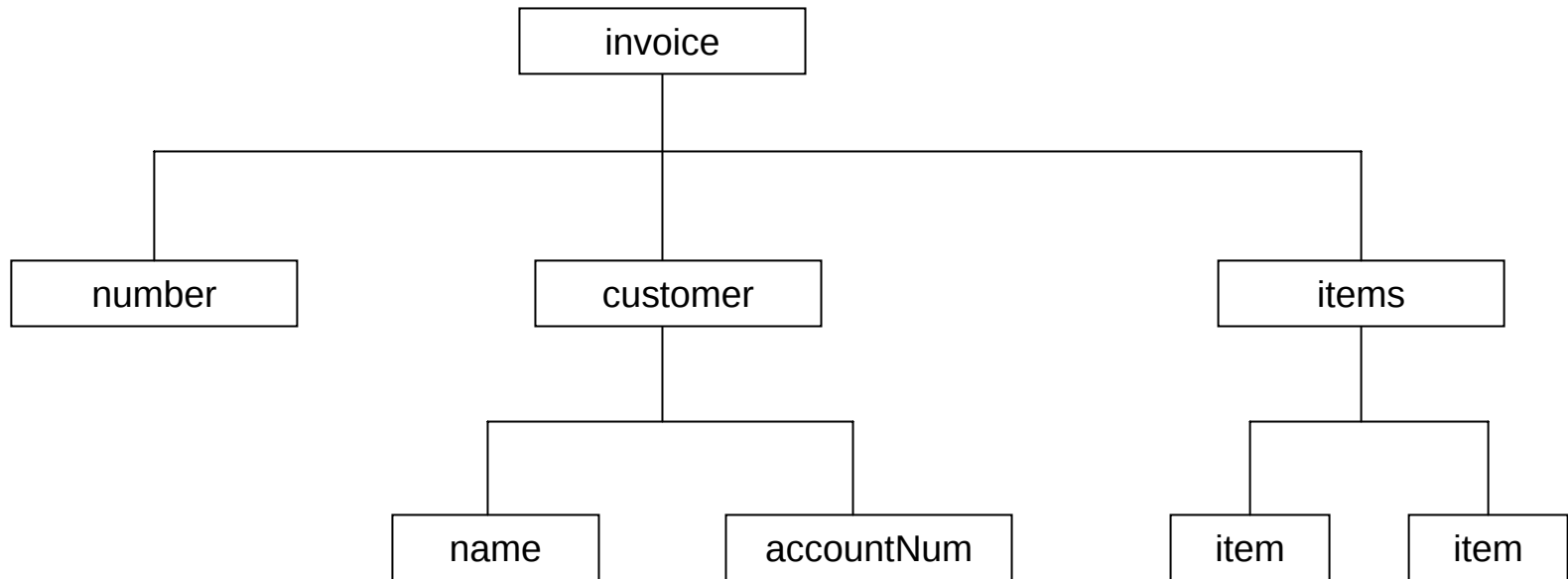
- This is where XSL comes in

XML

- Each box in the flow chart is a container
 - Containers are also known as ELEMENTS
 - What tags you are wrapping text in
 - What your data talks about

`<name>Jake</name>`

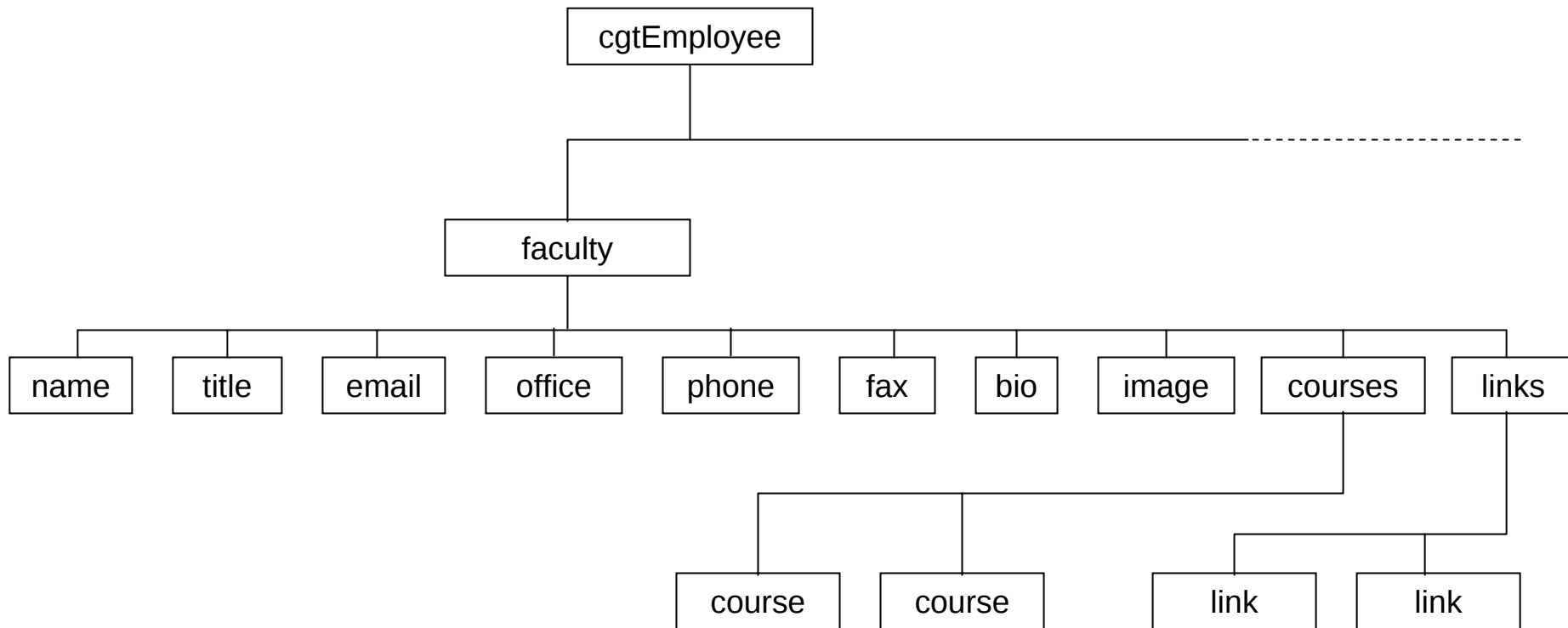
XML STRUCTURE



XML STRUCTURE

```
<invoice>
  <number>xxx</number>
  <customer>
    <name>yyy</name>
    <accountNum>ZZZ</accountNum>
  </customer>
  <items>
    <item>aaa</item>
    <item>bbb</item>
  </items>
</invoice>
```

CGT FACULTY DATA TREE



CGT FACULTY XML TREE

```
<cgtEmployee>
  <faculty>
    <name></name>
    <title></title>
    <email></email>
    <office></office>
    <phone></phone>
    <fax></fax>
    <bio></bio>
    <image></image>
    <courses>
      <course number="101" href="class.htm" />
      <course number="102" href="class2.asp" />
    </courses>
    <links>
      <link name="link1" href="link1.htm" />
      <link name="link2" href="link2.asp" />
    </links>
  </faculty>
</cgtEmployee>
```

- For the CGT Dept., the XML structure between the faculty tags would repeat ~20 times.

- Why?

- There are ~20 faculty members in CGT

Abridged Version

```
<cgtEmployee>
  <faculty>
    ...
  </faculty>
  <faculty>
    ...
  </faculty>
  <faculty>
    ...
  </faculty>
  <staff>
    ...
  </staff>
</cgtEmployee>
```

Or maybe
something
like this →

```
<cgtEmployee>
  <employee type="faculty">
    ...
  </employee>
  <employee type="faculty">
    ...
  </employee>
  <employee type="staff">
    ...
  </employee>
  <employee type="staff">
    ...
  </employee>
</cgtEmployee>
```



Faculty vs Staff

■ Staff

- May not have as many elements as faculty
 - courses
 - links
 - May or may not have a bio
 - publications

Big Picture

- Perhaps your root is technology:

```
<technology>
  <bcmEmployee>
    ...
  </bcmEmployee>
  <cgtEmployee>
    ...
  </cgtEmployee>
  <cptEmployee>
    ...
  </cptEmployee>
  ...
</technology>
```



HTML, CSS, XML, XSL

- HTML

- Formatting without structure

- CSS

- Added formatting without structure

- XML

- Structure without formatting

- XSL

- Add formatting to XML

FORMATTING XML WITH CSS

- XML can be formatted with
 - HTML
 - CSS
 - XSL (Extensible Stylesheet Language)
 - XSL is to XML what CSS is to HTML
 - Kind of...
 - CSS can be applied to an XML document

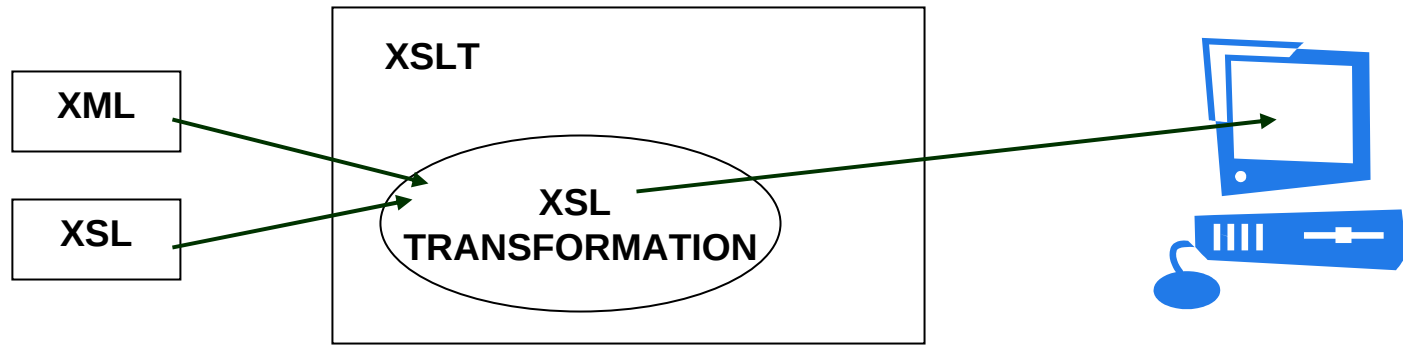
```
<?xml-stylesheet type="text/css" href="mystyles.css"?>
```



XML AND XSLT

- XML – Extensible Markup Language
- XSLT – Extensible Style Language Transformations
- Well-formed
 - Adheres to rules we described earlier
 - Same rules we have followed all semester
- Valid
 - Is well-formed, but adheres to something called a schema

XSL Transformation





XML and XSL

- XML CSS Example
- XSL Example 1
- XSL Example 2
- orgChart example

XSLT

- XSLT document has series of Templates
- Each template references a source document Element
 - (match)

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

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



XSL

- Extensible Stylesheet Language
 - XML based language
 - Used for manipulating XML data
 - Takes care of the Presentation Layer

XSL continued

- With this separation of Data & Presentation
 - Can have one set of Data
 - Can create many stylesheets for data
 - Can manipulate data differently each time
 - KEY
 - Without touching the data

DB

vs.

XML

Employee	
	EmployeeID
	LastName
	FirstName
	Title
	TitleOfCourtesy
	BirthDate
	HireDate
	Address
	City
	Region
	PostalCode
	Country
	HomePhone
	Extension
	Photo
	Notes
	ReportsTo

```
<Employee>
  <EmployeeID></EmployeeID>
  <LastName></LastName>
  <FirstName></FirstName>
  <Title></Title>
  <TitleOfCourtesy></TitleOfCourtesy>
  <BirthDate></BirthDate>
  <HireDate></HireDate>
  <Address></Address>
  <City></City>
  <Region></Region>
  <PostalCode></PostalCode>
  <Country></Country>
  <HomePhone></HomePhone>
  <Extension></Extension>
  <Photo></Photo>
  <Notes></Notes>
  <ReportsTo></ReportsTo>
</Employee>
```

If Statement

VBScript

```
If 7 <> 14 Then
    Response.Write("True")
End If
```

XSL

```
<xsl:if test="7 != 14">
    True
</xsl:if>
```



Choose

VBScript

```
If 7 <> 14 Then
    Response.Write("True")
ElseIf 7 = 14 Then
    Response.Write("False")
Else
    Response.Write("This will never happen.")
End If
```

Choose (cont.)

XSL

```
<xsl:choose>
  <xsl:when test="7 != 14">      <!-- Same as If above ----->
    True
  </xsl:when>
  <xsl:when test="7 = 14">      <!-- Same as ElseIf above -->
    False
  </xsl:when>
  <xsl:otherwise>               <!-- Same as Else above ---->
    This will never happen.
  </xsl:otherwise>
</xsl:choose>
```



For

Using this XML:

```
<CATALOG>  
  <PLANT>  
    <COMMON>Marigold</COMMON>  
  </PLANT>  
</CATALOG>
```



For (cont.)

VBScript

```
For Each child In oRootElement.childNodes  
    Response.Write(child.childNodes(0).text)  
Next
```



For (cont.)

XSL

```
<xsl:for-each select="PLANT">  
    <xsl:value-of select="COMMON">  
</xsl:for-each>
```

img tag Example

Using this XML

```
<manager>
  <employee>
    <alias>cabartlett</alias>
  </employee>
  <staff>
    <alias>jmfallen</alias>
  </staff>
  <booboobear>
    <alias>booboo</alias>
  </booboobear>
</manager>
```

img tag Example (cont.)

XSL

```
<!-- Starting from already inside manager ----->
<!-- The asterick is a wildcard ----->
<!-- Doesn't matter that one element is employee & --->
<!-- the next is staff & the next is booboobear ----->

<xsl:for-each select="*/alias">
  <!-- Create big image -->
  <img width="144" height="144" style="display:none">
    <!-- The following line should be all on one line, but it wraps here: -->
    <xsl:attribute name="src">images/bigTestPics/<xsl:value-of select="."
      />.jpg</xsl:attribute>

    <xsl:attribute name="id"><xsl:value-of select="." />bigPic</xsl:attribute>

    <xsl:attribute name="name"><xsl:value-of select="." />bigPic</xsl:attribute>
  </img>
</xsl:for-each>
```

img tag Example (cont.)

Output

```

```

```

```

```

```

XSL

■ XSL Parameters

```
<xsl:param>
```

```
<xsl:param name="phone">555-1234</xsl:param>
```

Looks the same as a variable



Variable vs. Param

■ Variables

- Constants (values cannot be changed)
- Only way they can be changed is by declaring it inside of a for-each loop, in which case its value is updated for each iteration.

Variable Example

```
<xsl:variable name="mailLink">
  <a href=mailto:{$email}>
    <xsl:value-of select="$email" />
  </a>
</xsl:variable>
```

Notice the use of { and }

- Used because XML will not allow you to nest angle brackets <>
- If you tried to use value-of instead of { } then you would get an error

Continued

Finally, notice that the above variable has tags in it. It has the anchor tag `<a>`

To copy this variable value, it would look like this

```
<p>Please contact us at  
  <xsl:copy-of select="$mailLink" />  
</p>
```

Notice the use of copy-of

Use copy-of when there are elements (tags) as part of the value of a variable.

XSL Attributes

```
<book id="1234567">  
  <author></author>  
</book>  
<book id="7654321">  
  <author></author>  
</book>
```

Select the ID of each book

```
<xsl:for-each select="book">  
  <b><xsl:value-of select="@id" /></b>  
</xsl:for-each>
```

Use the @ symbol to access attribute values

Create a Variable

■ Variable Creation

```
<xsl:variable name="email">rjglotzbach@tech</xsl:variable>
```

```
<xsl:variable name="subject">Wrox Press</xsl:variable>
```

```
<xsl:variable name="address">123 N. South St. </xsl:variable>
```

To get the value of the variable

```
<xsl:value-of select="$email" />
```

Precede the variable name with \$

Parameters

- Assigned a value from `<xsl:with-param>`
- Can be assigned just like a variable (as in the preceding example), in which case, it behaves just like a variable.