Introduction to XSLT Concepts

Deborah Aleyne Lapeyre and B. Tommie Usdin
Mulberry Technologies, Inc.
17 West Jefferson St., Suite 207
Rockville, MD 20850
Phone: 301/315-9631
Fax: 301/315-8285
info@mulberrytech.com
`http://www.mulberrytech.com`

January 2006
©2006 Mulberry Technologies, Inc.



Introduction to XSLT Concepts

Administrivia.....1

What is XSLT.....4

What XSLT Does is "Transform".....5

The Very Basics of XSLT Transforms.....6

Sample XSLT Transforms

Logical Components of an XSLT Application.....9

Component 1: XML Document.....10

Looking at an XML Document as a Tree.....11

Component 2: The XSLT Stylesheet (aka XSLT Transform).....11

An XSL Stylesheet / Transform.....13

Component 3: An XSLT Engine/Processor.....14

Component 4: The Output File(s).....16

Watching a Stylesheet in Operation

How Input-Driven Stylesheets Work.....16

Advice: What to Do *and Not Do* with XSLT.....17

Business Uses XSLT Because XML is Everywhere.....17

For the *Right Kind of Problems** ….....17

What's *Really* Easy in XSLT.....18

XSLT Easily Changes XML into Different XML.....18

XSLT Handles Markup Well.....18

XSLT is *Not* Good at Everything.....19

XSLT is Weak on Manipulating Text (Strings).....19

Really Big Files.....21

Making Flat Files into Hierarchies.....21

Where XSLT Fits in Processing.....22

How Organizations Use XSLT.....23

Simple Business Transforms.....23

Making HTML From Semantically Richer XML.....24

Single Source and Reuse Publishing.....25
Construct the Output for Publishing.....25
What You Want in the Order You Want It.....26
There is Not Just *One* Print Product.....26
Some of the Text is Added by the Transform.....27
Large Structures Can be Built and Inserted as Well.....27
XSLT is Also Useful During Production.....28
XML for Interchange and Archiving.....30
XSLT as the Middle Component in XSL-FO.....30
How XSL-FO Works.....31
Architecture of a Full XSL System (XSLT + XSL-FO).....32
Formatting Objects Describe Page Layout.....32
Applying Styles through XSL FOs.....33
XSL-FO is a Great Report Writer.....33
The Last Bits.....34
What is XPath.....34
XPath Has Two Main Uses.....35
You've Seen XPath in *match* Expressions 35
XPath Can Be Very Complex.....35
Another Complexity: Push-me Pull-you Stylesheets
What is a Pull Stylesheet?.....37
Why Pull Can Be a Problem.....40
Heads UP: XSLT and XPath 1.0, 1.1, 2.0.....41
What Was "Wrong" with XSLT 1.0.....41
XSLT 2.0: More Power; More Programmer Responsibility.....42
How to Deal with XSLT 1.0 and 2.0 (November 2005).....42
How to Make XSLT Programmers.....43
XSLT is Also Really Easy But.....43
How to Learn XSLT.....45
Debbie's XSLT Programming Pearls (Optional).....45

Now Let's Look at Some Real Stylesheets.....46

End Speech; Start References

For Further Information.....47

XSLT Technical Reference Book.....47

Useful XSLT Reference Website: Zvon.....48

XSLT Concept/Syntax Books.....48

XSLT Syntax+ for Programmers.....48

Colophon.....49

Appendixes

Appendix 1: Representative XSLT Tools.....49

Appendix 2: Acronyms Used in This Talk.....50

Introduction to XSLT Concepts

slide 1

Administrivia

- Start, end, break
- Ask questions any time (please!)
- Who we are
- Why this class
- Why more publishing examples
- Anything else?

slide 2

Where We Are *Not* Going in This Tutorial

- What is XML, why you should care, how XML works (element, attribute, DTD, schema, entity)
- How to solve your particular business problem(s)
- Programmer stuff like how to write stylesheets (although you *will* see some code)
- Syntax of the XSLT language (templates, functions, location paths)
- Detailed XPath syntax (location paths, functions, data types)
- XSLT tools
- XSL-FO in depth (that's this afternoon)

slide 3

Where We Are Going Today

The What and Why of XSLT

- What is transformation, what is XSLT
- How it works (logical components of an XSLT system)
- How to think about it (the XSLT processing model)
- How businesses are using XSLT
- What XSLT does not do well
- How should you learn/write XSLT

slide 4

WARNING!

We are going to show code!

You'll understand the examples even if you ignore the code

We are going to act as if you never heard of XSLT and start from scratch

A Quick Poll (Who You Are)

- Where in the process
  - content creators / editors / publishers
  - prepress / composition
  - printers
  - print / web / graphic design
  - fulfillment / distribution
  - System analysts / application programmers
  - Training
- What kind of publishing
  - Books (monographs, reference series, etc.)
  - Journals
  - Magazines and newspapers
  - Product documentation
  - Technical documentation
  - Course materials (CBT, course-packages, tests, textbooks plus, etc.)
- Non-publishing folks

slide 6

What Do You Know Now?

- Know HTML (even a little)
- XML
- SGML
- XSLT
- XSL-FO
- Microsoft Word, WordPerfect
- QuarkXPress, InDesign, other desktop publishing
- High-end composition systems

slide 7

What is XSLT

Extensible Stylesheet Language Transformation

- Name is misleading
- **Stylesheet**
  - implies it makes things *look* like something
  - not necessarily or usually true
- Name should have been
  "The XML Transformation Language"

slide 8

So What is XSLT Really?

- Provides transformation and manipulation functions for XML files
- Designed to make XML into *something else*
- 1.0 W3C Recommendation 1999
- 2.0 Candidate Recommendation November 2005

slide 9

What XSLT Does is "Transform"

Transform means change

Reads XML documents and writes

- HTML for browsers
- a different XML tag set
- typesetting driver file (InDesign, QuarkXPress, FrameMaker)
- interchange file (RTF, RDF, EDI, etc.)
- a flat ASCII file (plain text, comma separated etc.)

slide 10

The Very Basics of XSLT Transforms

- Transform
  - does *not* change the input file
  - creates one (or more) new output files
- Transform does *not* make something else into XML
- Two basic requirements
  - known XML source (tag set, schema, DTD)
  - known target

Sample XSLT Transforms

slide 11

Take in an XML document

```
<employee-record type="dog" empno="9">
<name>
<first>Sasparilla</first>
<last>Usdin</last></name>
<affiliation>
<title>Deputy in Charge of Chewables</title>
<company>Mulberry Technologies</company>
<location><city>Rockville</city>
<state>MD</state><zip>20850</zip></location>
<email-name>sassy</email-name>
</affiliation>
<height unit="in">36</height><weight unit="lb">70</weight>
</employee-record>
```

slide 12

Transform It into HTML

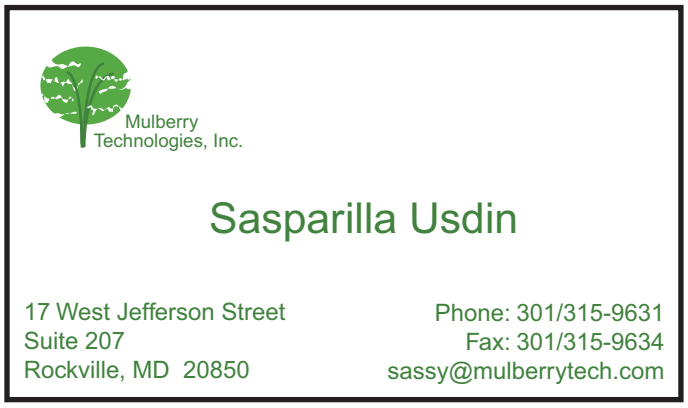
(convert to HTML and display in a browser)



slide 13

Transform It into PDF

(convert to PDF and display with Acrobat)



slide 14

Transform It into QuarkXPress



New Employee Announcements

Sasparilla Usdin
has recently joined Mulberry Technologies, Inc.'s Rockville staff as Deputy in Charge of Chewables.

Welcome to the team, Sassy!

- XML elements rolled into "form letter"
- Something (perhaps employee-id) linked to photo

slide 15

Transform It into a Database Load File

```
Key: 00095AUS
EMPNO: 009
001:USDIN
002:Sasparilla
008:36
014:70
020:Deputy in Charge of Chewables
```

slide 16

In Other Words: Tagging Changes Large and Small

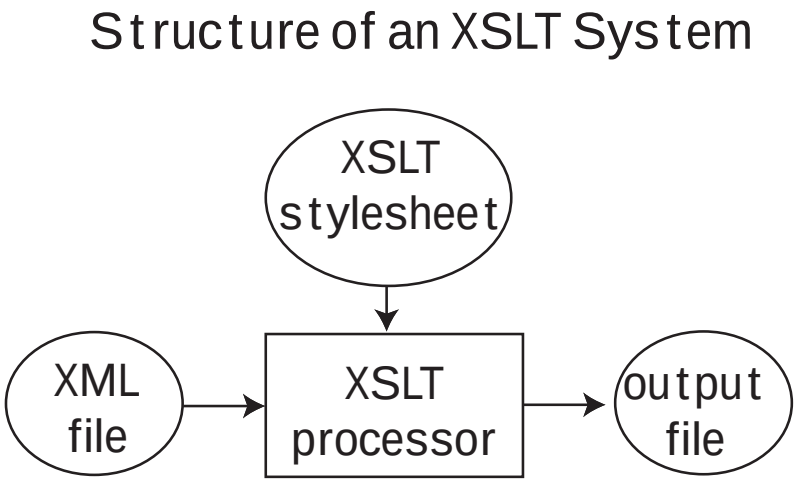
- Change the following
  ```
  <surname>Lapeyre</surname>
  <firstnames>Deborah A.</firstnames>
  ```
  INTO
  ```
  <contrib>Deborah A. Lapeyre</contrib>
  ```
- Change the following
  ```
  <chapter><title>Lawns and Gardens</title>
  ```
  INTO
  ```
  <h2>Lawns and Gardens</h2>
  ```
- `<bold>Tall</bold>`...
  INTO
  ```
  :G46Helvetica-ExtraBold;Tall
  ```

slide 17

Logical Components of an XSLT Application

(needs XSLT processing software (called an "XSLT Engine")

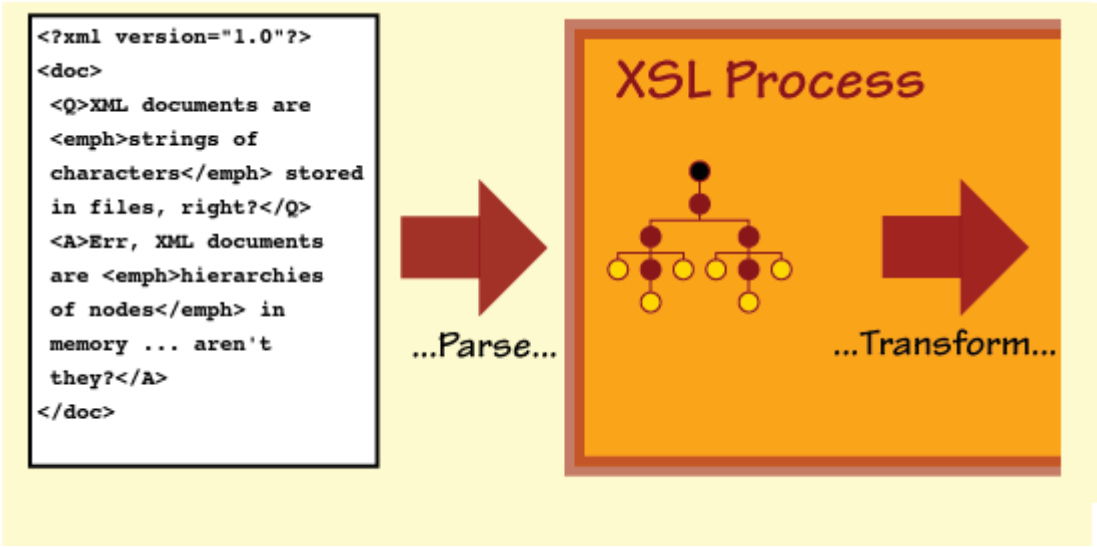
- Reads XML document(s) (tags and text)
- Uses an XSLT stylesheet/transform (the program)
- Runs using XSLT processing software (called an XSLT Engine)
- Produces output document(s)



slide 18

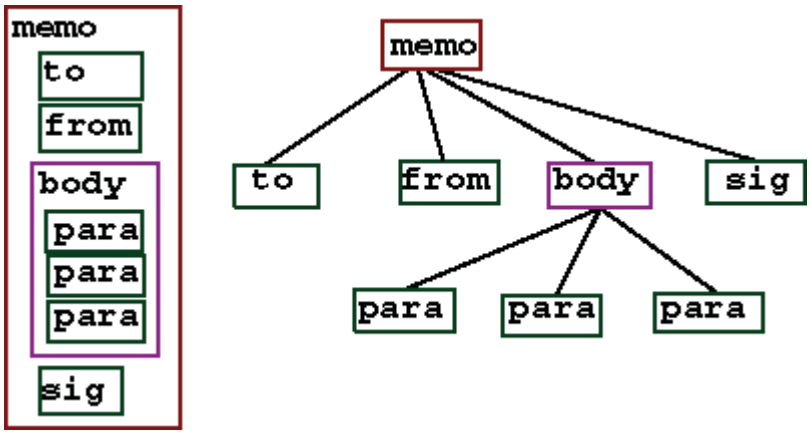
Component 1: XML Document

- XML documents
  - are sequences of data characters and markup
  - start-tag and end-tag markup delimits elements
- But XSLT does not work *directly* on XML documents
- Part of the XSLT processing (usually an XML parser) builds a tree
- XSLT works on trees (made from XML documents)



slide 19

Looking at an XML Document as a Tree



slide 20

Component 2: The XSLT Stylesheet (aka XSLT Transform)

- A computer program
- Transformation instructions
- Called a “stylesheet” (or a “transform”)
- A well-formed XML document!
- Commands in the XSLT language are
  - a tag set (elements and attributes)
  - defined *by the W3C XSLT recommendation*
  - that look like this (`<xsl:sort>` and `<xsl:number>`)

slide 21

An XSLT Stylesheet / Transform Is

- A series of rules (called template rules)
  - Each rule is a sequence of XSLT commands
  - Each command is an XML element with attributes
- A rule is executed when it
  - matches some condition
  - or is called by name

slide 22

"Matching a Condition" Means

- If you find a (_____) in the source XML,
  then do this (perform the template)
- Matching can mean finding in the XML
  - an element
  - an element/attribute combination
  - an element in a certain context
  - some special circumstance
    (words in the content, any element at all, etc.)

An XSL Stylesheet / Transform

(close your eyes, this is code)

- Here is a template rule
- This rule matches a `<paragraph>` element
- Notice that it is made up of XML elements (two kinds)
- The two kinds of XML elements
  - XSLT language tags (instructions)
  - HTML tags

```
1  <xsl:template match="paragraph">
2      <hr/>
3      <p>
4         <xsl:apply-templates/>
5      </p>
6  </xsl:template>
```

slide 24

Component 3: An XSLT Engine/Processor

- You need special software to run XSLT
  - But you don't have to buy them
  - Free open-source, shareware, as well as commercial
- New ones all the time
- Look for more at: *http://www.xml.com*
  - Saxon (*http://users.iclway.co.uk/mhkay/saxon/*)
  - Xalan XSLT (*http://xml.apache.org/xalan/index.html*)
  - Unicorn XSLT Processor (*http://www.unicorn-enterprises.com/*)
  - XSLT C library for Gnome (*http://xmlsoft.org/XSLT/*)

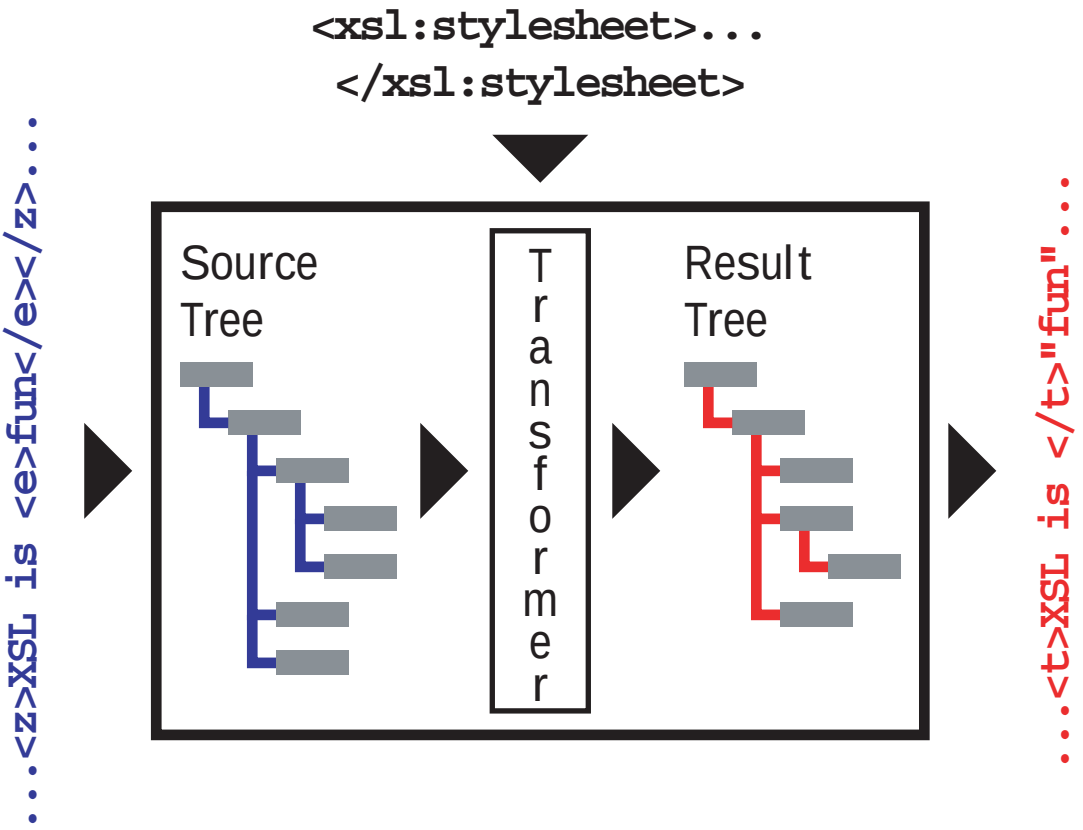
slide 25

XSLT Also Built Into/Can be Hooked Into

- XML programmers' developing environments
- XML-aware editors
- Content aggregation systems
- Other XML processors

In softwares like this XSLT comes built in and you still don't have to buy it!

How an XSLT Processor Works



The big dark rectangle above is the XSLT processor

slide 27

Component 4: The Output File(s)

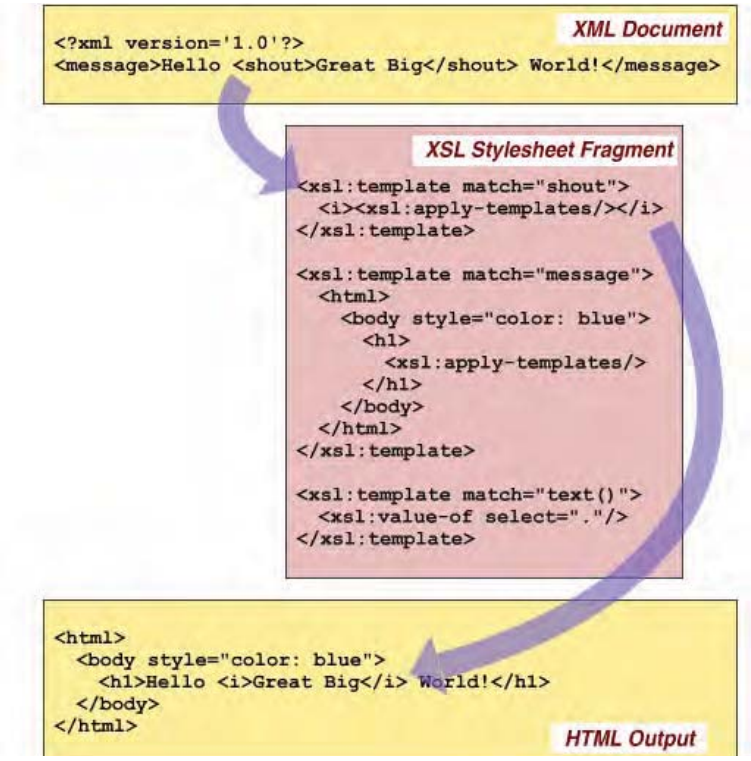
XSLT can make 3 syntaxes for output

- XML files
- HTML
- Text (untagged files)
  - ASCII email message
  - comma-separated file
  - desktop publishing system format (e.g., XTags for QuarkXPress)

Watching a Stylesheet in Operation

slide 28

How Input-Driven Stylesheets Work



slide 29

Advice: What to Do *and Not Do* with XSLT

slide 30

Business Uses XSLT Because XML is Everywhere

- XSLT was designed to process XML
  - Takes full advantage of the tree
  - XML constructs are built in (no special programming)
- Solves problems with
  - order of the material
  - document model/processing mismatch
  - interchange (mine different from yours different from ours)
  - personalization/localization
- Part of the XML family, so applications built to support

Makes content fluid, as XML and SGML have always promised

slide 31

For the *Right Kind of Problems** …

XSLT is

- faster
- better
- cheaper

*All three, but note the caveat

slide 32

What's *Really* Easy in XSLT

- Extract just *some* of the input
- Change sequence of elements (rearrange / sort)
- Remove material
- Use the same element / attribute in 5 places
- Add generated text

slide 33

XSLT Easily Changes XML into Different XML

- Rename an element or attribute
- Change element xxx into element yyy
- Make elements into attributes
- Make attributes into elements

slide 34

XSLT Handles Markup Well

XSLT works best when

- What you care about (want to process) is tagged!
- Hierarchy is explicit
- The most important relationships are tree relationships
  - containment (parent / child)
  - siblings
  - attributes

slide 35

XSLT is *Not* Good at Everything

- Not at all
  - conversion *into* XML
  - Non-XML data (Word, QuarkXPress, SGML)
- Not as good as most "programming languages"
  - number crunching (arithmetic and higher math)
  - string processing (parsing)
  - really big files
  - making structure where there was none
    (making flat files into hierarchies)

slide 36

XSLT is Weak on Manipulating Text (Strings)

- An XSLT processor expects to work on
  - a tree of nodes
  - not an XML file of tags and text
- If you have untagged files
  (comma delimited, space delimited, tab delimited)
  - there is no tree
  - strings must be "parsed" into pieces
  - XSLT does this awkwardly

(XSLT 2.0 has *better* string manipulation than XSLT 1.0, but…)

slide 37

What If You Need String Processing?

- Use a different programming language
- Preprocess to make the data into XML
  - add tags
  - add nesting (make hierarchy explicit)
  - add end tags

slide 38

Real World String Parsing Example

The original data looked like this:

```
<title>Large Animals</title>
<address>Dallas, TX 23071</address>
```

The Requirement was to put the name of the state before every section title

```
<title>Texas Large Animals</title>
```

One solution

- run a program (Perl, etc.) to make the following

```
<title>Large Animals</title>
<address><city>Dallas</city>, <state>TX</state>
 23071</address>
```

- Now it's a node in a tree, run XSLT

slide 39

Really Big Files

Files are sequences of characters; XSLT works on trees

- Many XSLT processors
  - make the input document into a *tree in memory*
  - make the stylesheet into another *tree in memory*
  - make the results into more *trees in memory*
- Document may not fit in memory
- Usual solution is “chunking”

slide 40

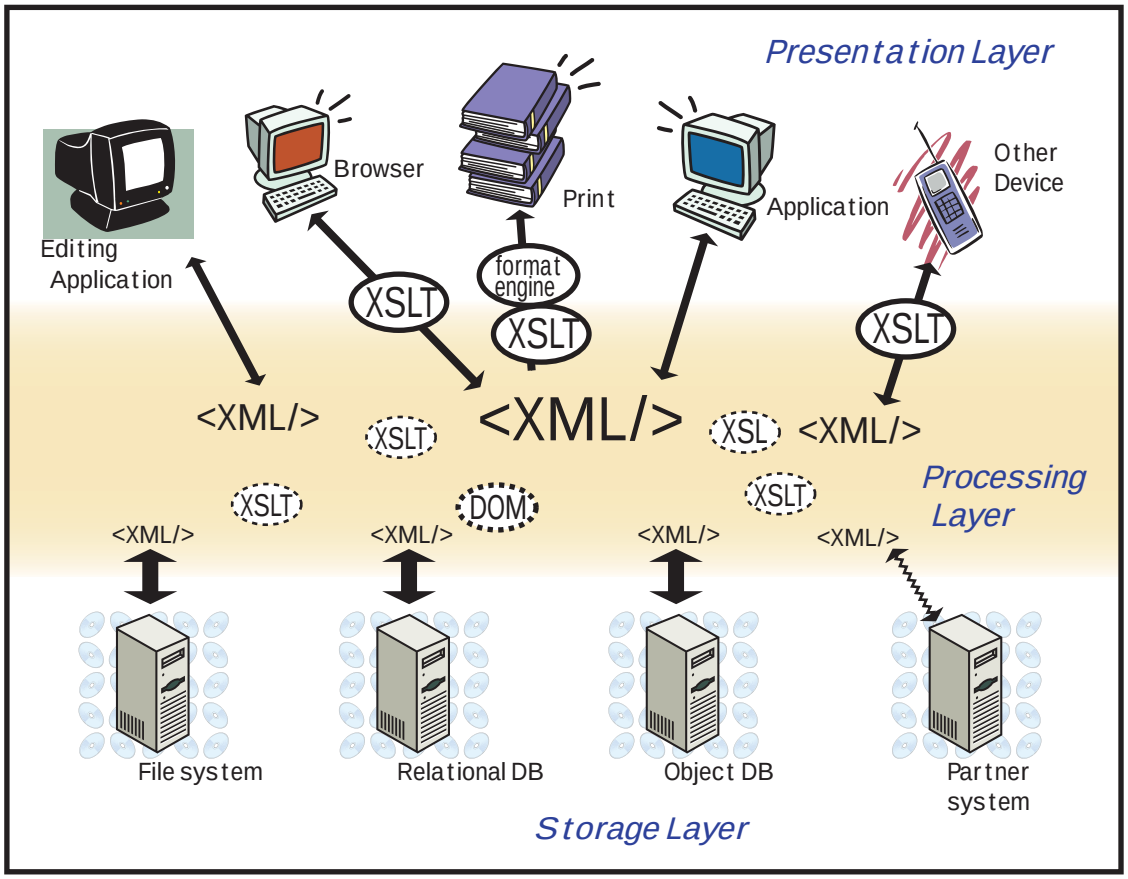
Making Flat Files into Hierarchies

- XSLT 1.0 was not designed to do this
- Sometimes you can do it anyway
  - using grouping techniques
  - using keys (an advanced technique)
- When it works (maybe 2/3) it is elegant, clever, and tricky
- Success depends on the data
  - information must be there
  - markup must be clean and consistent

(XSLT 2.0 much better at this, but still needs clear distinctions)

Where XSLT Fits in Processing

- XML used in any of the three tiers, *especially in the middle*
- XSL is used for any processing
  - *within* the middle tier (application to application)
  - *between* tiers
  - database to database



XSLT is often the mechanism represented by the arrow.

slide 42

How Organizations Use XSLT

- Simple business transforms
- Making HTML from semantically richer XML
- Single Source and Reuse Publishing
- Transforms for editorial QA
- XML to XML transforms
- XSLT as the middle component in XSL-FO

slide 43

Simple Business Transforms

- Data exchange between applications
  - you give me what you think I need
  - I take what I want in the order I want it
- E-Business / E-Commerce — Translate between transaction formats faster, easier, better than with EDI
- Portals / Web Services / Data Aggregation
  - grab just the data you want from a repository, database, files
  - rearrange it to suit
  - serve it forth

slide 44

Making HTML From Semantically Richer XML

Read in semantically rich tagging

```
<COMPUTER CLASS="Portable">
<MFR>GCA</MFR><FAMILY>Laptop</FAMILY>
<LINE>Thinkie</LINE><MODEL>520XL</MODEL>
<DISK UOM="GB">80</DISK>
<SPEED UOM="GHz">3.2</SPEED>
</COMPUTER>
```

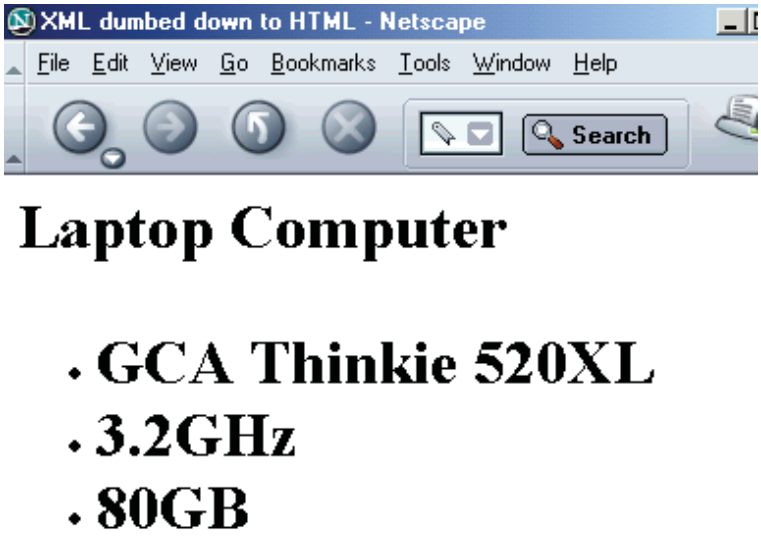
Simplify it to HTML for display in any browser

```
<H2>Laptop Computer</H2>
<UL>
<LI>GCA Thinkie 520XL</LI>
<LI>3.2GHz</LI>
<LI>80GB</LI>
</UL>
```

(Use CSS for look-and-feel; serve to even really old browsers)

slide 45

Which Displays As



slide 46

Single Source and Reuse Publishing

(XSLT fulfills the XML promise of multiple use)

- Making the output product
  - preparation for publishing (web and print)
  - Print on Demand and web serving
  - composition drivers
- QA and proofing
- XML to XML transform
- XSLT as the middle component in XSL-FO

slide 47

Construct the Output for Publishing

(transformations build products)

- Out of databases, rearranged for the web
- Customized printing = Different users get
  - different order
  - different text or content
  - same content different look-and-feel
- Print on Demand (with data up to *this* minute)

slide 48

What You Want in the Order You Want It

Select / Extract / List / Omit

- Pull out the metadata to put into the catalog
- Extract titles and abstracts of all articles for the advertising webpage
- Extract the CME material for a special site for nurses
- Get all the environmental impact material
- Publish this report with all the **SECRET** material removed
- Get me the citations to send to the link matching service
- My car has a sun-roof, manual transmission, and option package #4, make me *my* owners manual
- Get me all the dosage sections that mention pregnancy restrictions

slide 49

There is Not Just *One* Print Product

- Customization (change, assemble, or adapt based on customer or organization)
  - mix and match text and graphic components
  - target specific markets
- Personalization (tailor a product to an individual person)
  - based on purchase, profile, history
- Internationalization (multiple languages, script, writing directions, currency)
- Localization (adapting a print product to a specific locality/region)

slide 50

Some of the Text is Added by the Transform

(textual additions are called "generated" text)

Text that is not in the data, but is put in by the transform, based on the tagging

For example:

- numbers or bullets that prefix list items (1., 2., 3.) (based on `<list-item>` tag)
- mark a footnote reference (²) or a citation reference **[Lapeyre, 2006]** based on a cross-reference made with an attribute
- Adding words or phrases to titles (**Chapter VI** Sassy Poodles)
- Turning a cross reference into text
  - `<xref redid=:A123456"/>` into
  - "See Figure 6, Herpetologist Distribution Curve"

Less content maintenance!

slide 51

Large Structures Can be Built and Inserted as Well

- Table of Contents from chapter titles
- Subject index from embedded index terms
- List of Figures, Tables, Equations, Genus-species names
- Title Page from the metadata elements
- Leaning Objectives from embedded objectives

XSLT is Also Useful During Production

Transformations for Editorial QA and Proofing

- Make checklists for humans to examine
- Make files for automated authority checking
- Run galleys as often as you want
- Make useful displays that will never be printed
  - number things that won't be numbered on display
  - if the book will say"(See Section *4.3)*"
    put the section title into the reference
    "(See Section *4.3 My Life with Poodles*)"
  - make false color proofs
    - ferrous materials in red and non-ferrous in green
    - all skeletal system paragraphs in blue, circulatory system paragraphs in red
    - a citation with author name in green, journal name in pink, year in blue, paper title in yellow

False Color Proof

Water is blue (italic), land is yellow (bold), and "features" are purpley (display font in the print)

The rivers of the state flow in general from NW. to S.E., across the **Blue Ridge**, the **Piedmont** and the **Coastal Plain**, following courses which were established before erosion had produced much of the present topography. But in the **Newer Appalachians** the streams more often follow the trend of the structure until they empty into one of the larger, transverse streams. Thus the *Shenandoah* flows N.E. to the *Potomac*, the *Hoiston* S.W. toward the *Tennessee*. A part of this same province, in the S.W. part of the state, is drained by the *New* river, which flows N.W. across the ridges to the *Kanawha* and *Ohio* rivers in the **Appalachian Plateau**. In the limestone regions caverns and natural bridges occur, among which LURAY CAVERN and the NATURAL BRIDGE are well known. The drowned lower courses of the SE. flowing streams are navigable, and afford many excellent harbours. *Chesapeake Bay* covers much land that might otherwise be agriculturally valuable, but repays this loss, in part at least, by its excellent fisheries, including those for oysters. In the S.E., where the low, flat **Coastal Plain** is poorly drained, is the *Great Dismal Swamp*, a fresh-water marsh covering 700 sq. m., in the midst of which is *Lake Drummond*, 2 m. or more in diameter. Along the shores of *Chesapeake Bay* and the *Atlantic Ocean* are low, sandy beaches, often enclosing lagoons or salt marshes. ¶

slide 54

XML for Interchange and Archiving

XML to XML Transforms

- Corporate tagset into
  - client's tags
  - business partner's tags
- Company-specific tags into Industry Standard schema
- 5 Publisher tag sets into one repository / aggregator tag set
- Authoring DTD into publication DTD
- 50 articles to one RSS feed of the summaries

slide 55

XSLT as the Middle Component in XSL-FO

- XSL is a spec with two parts
  - XSLT (the transformation part)
  - XSL-FO (the formatting part)
    - XSL provides a tag set into which XML documents may be transformed (using XSLT)
    - describes page geometry
    - says how to put content on the page

XSL-FO used to make PDF (or RTF or MIF) directly from XML

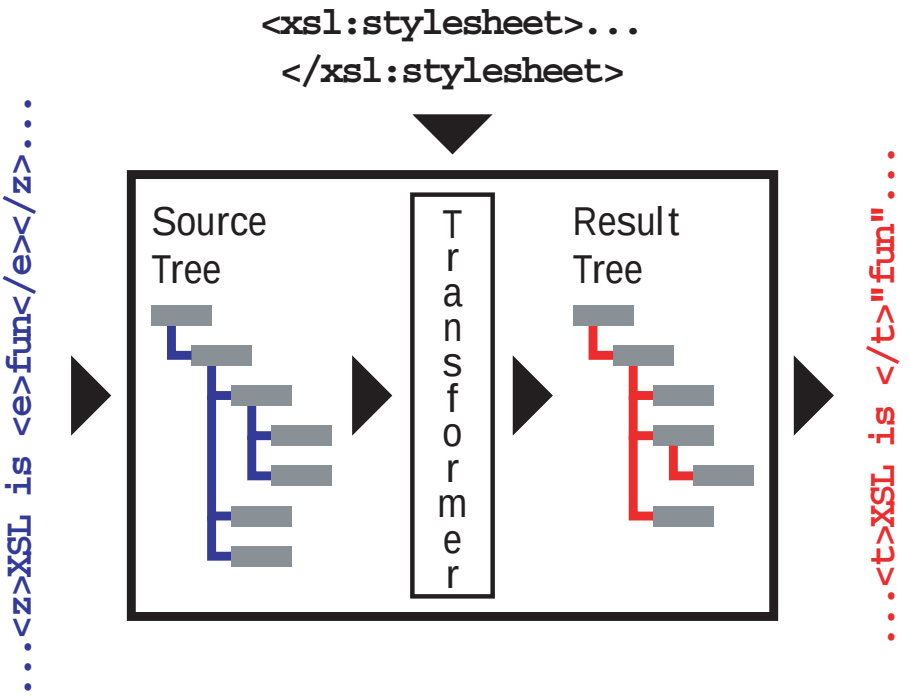
slide 56

How XSL-FO Works

- XSLT
  - transforms the input
  - makes a tree of formatting objects
- An XSL-FO document is
  - an XML document
  - with text and graphic content wrapped in formatting object tags
- XSL-FO (XSL Formatting Objects)
  - get processed by a *rendering engine* (software)
  - to make an output file
  - a display engine (like a browser or a printer) makes the pretty output

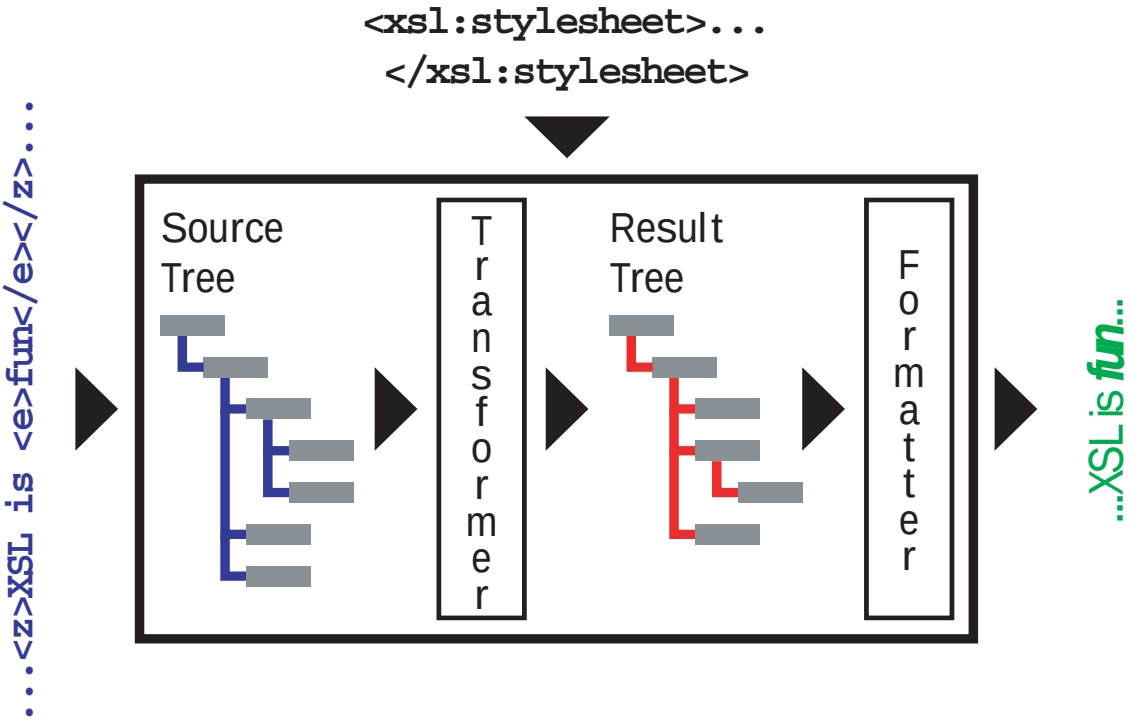
slide 57

Remember How an XSLT System Works



slide 58

Architecture of a Full XSL System (XSLT + XSL-FO)



slide 59

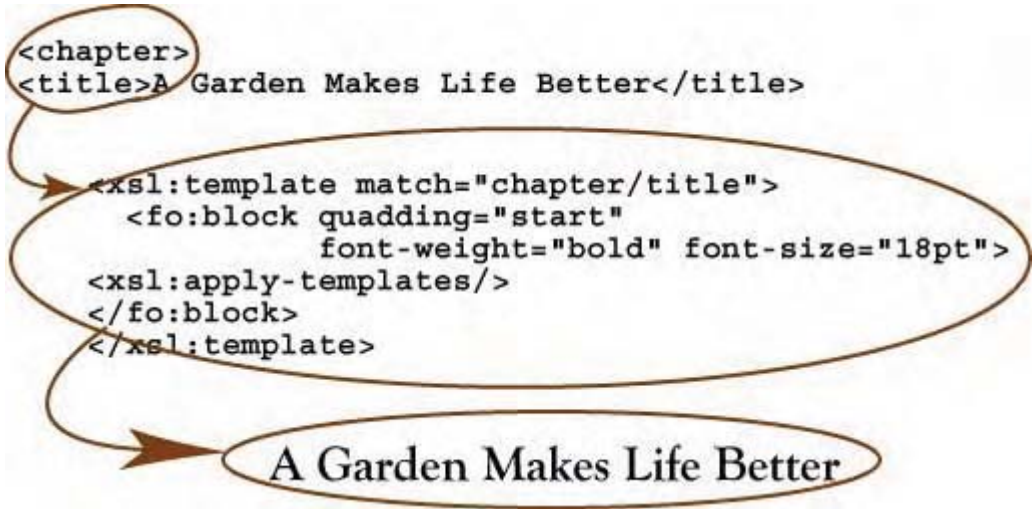
Formatting Objects Describe Page Layout

- Page layout:
  - page size, margins, columns
  - headers, footers, side-bars. etc.
- Different page layout templates (*masters*) can be sequenced, e.g.
  - first page followed by later pages
  - recto / verso alternating
- XSL-FO properties control hyphenation, widows / orphans, etc.

Mulberry Technologies, Inc.

slide 60

Applying Styles through XSL FOs



slide 61

XSL-FO is a Great Report Writer

(Pagination is not a problem)

- Credit card and bank statements
- Investment portfolios
- Hospital systems reports
- Insurance policies and claims
- Patient medical charts
- Directory products
  - product and editorial indicies
  - company personnel listing

slide 62

The Last Bits

- Other things you need to know about how XSLT works
  - XPath for tree-walking
  - Pull-style stylesheets
- XSLT 2.0 (with XPath 2.0)
- How can you make yourself (or your staff) into XSLT people

slide 63

There's Another Part of XSLT We Haven't Talked About XPath

Really powerful!

slide 64

What is XPath

- The tree-walking part of XSLT
- So named because it uses a path notation with slashes like UNIX directories and URLs

```
play/act/scecne/speech
invoice/customer_data/customer_name
```

- XPath 1.0 W3C Recommendation in 1999
- XPath 2.0 W3C Recommendation Draft 2005
  (more complex, more powerful, harder to learn)

slide 65

XPath Has Two Main Uses

- First use: Addressing
  - addresses (finds) part of an XML document
  - can address any part of the tree from any other
  - (ask for something in an XML document ("gimme my footnote!") ... and get it back)
- Second use in XSLT: for Testing/Matching
  - test whether a node in a tree matches a pattern
  - is this node a paragraph that is inside a footnote that has an attribute called "footnote-type" with the value "legal"?

slide 66

You've Seen XPath in *match* Expressions

- `<xsl:template match="title">`
  Matches `<title>` elements
- `<xsl:template match="scene/title">`
  Matches `<title>` elements that have a `<scene>` parent
- `<xsl:template match="para[@type='warning']">`
  Matches `<para>` elements that have a `type` attribute that has a value of "warning"

slide 67

XPath Can Be Very Complex

(all that power has a price)

```
child::slide[attribute::type="overview"]/child::list
[count(descendent::item) > 8]
preceding::slide[descendant::title
            [contains(self::node(),
'Business')]]
```

Thanks to Jeni Tennison for:

```
select="following-sibling::transaction
      [@type = $type and substring(@date, 4, 2) != $month] [1]"
```

Another Complexity: Push-me Pull-you Stylesheets

slide 68

XSLT is

- By design and default, driven *by the XML input file*
- That means you tell it *what* to do, not how or when
- Automatically recursive through the use of templates

slide 69

XSLT can be Written in Two Ways (1)

- Way # 1 = Input-driven (called Push)
  - walk the input tree
  - match elements in input tree
  - do something when you find a match
- This is what XSLT was designed to do
- This works the best (when it works for your data)

slide 70

XSLT can be Written in Two Ways (2)

- Way # 2 = Stylesheet driven (called Pull)
  - more like a typical computer program
  - walk the stylesheet
    (which specifies the order of the output document)
  - when it asks for data, go get it from the input tree
- Similar to some fill-in-the-blank programming languages

Pull stylesheets are most useful when you have *very regular data*

slide 71

What is a Pull Stylesheet?

Let's look at some XML for a menu

```
<specials-menu>
<menu-date>Friday, July 28, 2000</menu-date>
<spec-meat price="24.50">Pork chops with Chard
&amp;amp; Apples</spec-meat>
<spec-appetiz price="3.95">Seckel pears with
Gorgonzola and Walnuts</spec-appetiz>
<spec-soup price="6.95">Red and Yellow
Pepper</spec-soup>
<spec-fish price="18.50">Seared Achoo with Risotto
and Spinach</spec-fish>
<spec-pasta price="12.25">Wagon-wheels Alfredo
(with side salad)</spec-pasta>
<spec-sweet price="12.95">Strawberry and
Chocolate Tart</spec-sweet>
</specials-menu>
```

slide 72

Now Let's Look at the Stylesheet

Here's the XML file for that:

```
<html xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
      version="1.0">

<title>Today's Menu -
    <xsl:value-of select="//specials-menu/menu-date"/>
</title>

<h1> Specials --
    <xsl:value-of select="//menu-date"/>
</h1>

<h2>Appetizer</h2>
<ul>
  <li>
     <xsl:value-of select="//spec-appetiz"/>
  </li>
</ul>

<h2>Soup</h2>
<ul>
  <li>
     <xsl:value-of select="//spec-soup"/>
  </li>
</ul>
```

```
<h2>Fish</h2>
<ul>
  <li>
    <xsl:value-of select="//spec-fish"/>
  </li>
</ul>

<h2>Pasta</h2>
<ul>
  <li>
    <xsl:value-of select ="//spec-pasta"/>
  </li>
</ul>

<h2>Entree</h2>
<ul>
  <li>
    <xsl:value-of select ="//spec-meat"/>
  </li>
</ul>

<h2>Dessert</h2>
<ul>
  <li>
    <xsl:value-of select ="//spec-sweet"/>
  </li>
</ul>
<br/>

<h3>Specials are available every day from opening until
exhaustion.</h3>
<p>Come see us anytime!</p>

</html>
```

And That Produced HTML like this:

Specials -- Friday, July 28, 2000

Appetizer

- Seckel pears with Gorgonzola and Walnuts

Soup

- Red and Yellow Pepper

Fish

- Seared Achoo with Risotto and Spinach

Pasta

- Wagon-wheels Alfredo (with side salad)

Entree

- Pork chops with Chard & Apples

Dessert

- Strawberry and Chocolate Tart

Specials are available every day from opening until exhaustion.

Come see us anytime!

slide 74

Why Pull Can Be a Problem

- XSLT was designed as a "data filter" language (called side-effect free)
- Stylesheets tell the processor *what to do* when it finds something
- Processor controls the finding of things in the input tree (*when to do it*)
- Pull stylesheets
  - control both what and when
  - fight the design of the language

slide 75

Is Pull Always Bad?

- *Of course not*
  - might be the only way to do something
  - might be the best way
- Pull is useful when
  - output is in a specific order, not the order of input (many database and transaction applications)
  - data is very regular
- Not so good if
  - data is irregular
  - data order is unknown (like most text)
- hierarchy can change (like most text)

(Pull works best in a data environment)

slide 76

Heads UP: XSLT and XPath 1.0, 1.1, 2.0

- **XSL 1.0** — The current version
- **XSL 1.1** — Aborted, does not exist any more
- **XSL 2.0** —
  - The coming thing (Draft Recommendation as of Jan 2006)
  - *Not* backwards compatible with 1.0

slide 77

What Was "Wrong" with XSLT 1.0

(To simplify greatly the programmer-style reasons)

- Not really consistent with W3C XML Schema
- Weakly typed and cast types automagically
- Making multiple output files was an extension, not built in
- Could not pass trees into templates or query result trees
- Had to extend the language to write your own function
  (XSLT 1.0 has templates, not general functions)
- Grouping was difficult
- No regular expressions

slide 78

XSLT 2.0: More Power; More Programmer Responsibility

(this is all programmer stuff)

- Multiple result documents
- Strong data typing
- Temporary trees can be accessed; XPath can see them
- New serialization types and attributes to control things like Unicode normalization
- Grouping support
- Support for user-defined functions
- Regular expressions in match, replace, and tokenize functions
- Schema import including substitution groups

slide 79

How to Deal with XSLT 1.0 and 2.0 (November 2005)

- Processors will have a switch, user chooses
- Mission critical is done in 1.0 (now)
- Many processors **not** up to 2.0 (that's changing fast)
- Everyone learning and playing with 2.0
- **Recommendation:** If you aren't a declarative languages geek, learn XSLT 1.0 to get the template stuff down *then* learn XSLT 2.0

slide 80

How to Make XSLT Programmers

(out of yourself or your staff)

XSLT is

- Fast to learn
- Useful when you know even a little
- Fast to write (great for prototyping, reports)
- Powerful in XML problem space
- Use as a toolkit, use *in addition to* java, perl , python, C++, icon

slide 81

XSLT is Also Really Easy But...

- Very simple language (under 30 commands)
- Getting started is easy
- Anybody can learn this (secretary, editorial staff, lawyer)
- XSLT is a declarative, side-effect free language
  - processes “differently”
  - templates recurse all by themselves
- If programmers have a procedural background, they may have some habits to “unlearn”

slide 82

Pull: The Big Beginner Mistake

- Pull should be used sparingly
  - for special situations only
  - Best used on very regular, predictable structures (data)
- Beginners use it for everything
- How to spot this problem in your programmers' stylesheets
  - uses `<xsl:for-each>` to force recursion (instead of using templates, which recurse as designed)
  - uses `<xsl:value-of>` which
    - processes the *text* inside an element
    - (instead of `<xsl:apply-templates>`, which processes *both text the and embedded tags* inside an element)

slide 83

Why Programmers Will Like Pull

- They are used to controlling the order of execution
- They don't trust the stylesheet to "do it right"
- They need recursion, so they force it (instead of *using* it)
- They fight the template design
- Relational database folks forget about containment and mixed content
- Solution
  - training
  - at least one good book
  - training
  - write for different kinds of data
  - training

slide 84

How to Learn XSLT

- Start with a course
- Scan Jeni Tennison's beginner book
- Read some GOOD code (Norm Walsh's DocBook stylesheets)
- Buy Michael Kay's Programmer's Reference
  (and/or haunt Zvon)
- Read the XSL list and ask questions

* URLs for all the above are in the appendices

slide 85

Debbie's XSLT Programming Pearls (Optional)

(this is programmer stuff)

- Don't Fight XSLT; use the language for what it is good for.
- Not all data is trees or OHCO (Ordered Hierarchy of Content Objects).
- Don't curse it because it won't do YYY; reach for another language.
- In XSLT 1.0, variables are constants within scope and do not vary.
- 90% of all XSLT bugs are XPath bugs.
- What is your context? An XPath *to* nowhere, *goes* nowhere,
  however cleverly written.
- When all else fails, rewrite your code as simple templates,
  then debug those.

Now Let's Look at Some Real Stylesheets

```
<xsl:template match="address">
  <address>
    <xsl:apply-templates select="affil | subaffil"/>
    <xsl:apply-templates
      select="aline | city | cntry | email | fax |
              phone | postcode | province |
              state | web"/>
  </address>
</xsl:template>

<xsl:template match="affil">
  <affiliation>
    <xsl:apply-templates/>
  </affiliation>
</xsl:template>

<xsl:template match="subaffil">
  <subAffiliation>
    <xsl:apply-templates/>
  </subAffiliation>
</xsl:template>

<xsl:template match="aline | city | cntry | state |
                     province | postcode" >
  <line>
    <xsl:apply-templates/>
  </line>
</xsl:template>

<xsl:template match="phone|fax|email|web">
  <xsl:copy-of select="."/>
</xsl:template>
```

End Speech; Start References

slide 87

For Further Information

- XSL FAQ at **`http://www.dpawson.co.uk/xsl/xslfaq.html`**
- W3C XSL info: **`http://www.w3.org/Style/XSL`**
- XSL-List (Mailing List) Nice to Newbies!
  - See **`http://www.mulberrytech.com/xsl/xsl-list`**
- The W3C Recommendations (programmers only!)
  - XSLT 1.0 Recommendation: **`http://www.w3.org/TR/xslt`**
  - XSLT 2.0 Recommendation: **`http://www.w3.org/TR/xslt20/`** (Working Draft in Nov 2004 was the latest when this was written in January 2005)
  - XPath 1.0 Recommendation: **`http://www.w3.org/TR/xpath`**
  - XPath 2.0 Recommendation: **`http://www.w3.org/TR/xpath20/`** (Working Draft October 2004 was the latest when this was written in January 2005,. Committee members say this is pretty stable.)

slide 88

XSLT Technical Reference Book (XSLT programmer ought to have access to a copy)

- *XSLT (1st or 2nd Edition) Programmers Reference* by Michael Kay
  - ISBN: 1-861005-06-7 (WROX Press, but available Amazon)
  - [Not a tutorial; not a great XPath reference, other than that, really great! The 2nd edition has more material on XSLT tools. Don't read the W3C spec, read this book instead.]
- *XSLT 2.0 (3rd Edition) Programmers Reference* by Michael Kay
  - ISBN: 0-764-56909-0 (WROX Press 2004)
  - [Still a better read than the W3C Spec, but since the spec is still unfinished, so he's making intelligent, committee-member guesses on a few points. Has added case studies.]

slide 89

Useful XSLT Reference Website: Zvon

`http://www.zvon.org`

- If you feel you must read the XSLT 1.0 and XPath 1.0, at least read them indexed and hyperlinked on this site.**`http://www.zvon.org/xxl/XSLTreference/Output/index.html`**
- Also online tutorials for XPath 1.0, etc.

slide 90

XSLT Concept/Syntax Books

- *Beginning XSLT* by Jeni Tennison
  ISBN: 1-861005-94-6 (WROX Press, but it went into a second printing and Amazon still has new and used for cheap!) Her XSLT 2.0 Book is not available yet because she feels (as of Dec 2004) that the spec is still changing and she wants he book to be completely accurate.
- *The XSL Companion (2nd Edition better!)* by Neil Bradley
  - (Addison Wesley)
- *XML Bible* by Elliotte Rusty Harold
  - ISBN: 0-7645-4819-0 (Hungry Minds)
  - [Only a couple of chapters cover XSLT, but very good, and available online]

slide 91

XSLT Syntax+ for Programmers

- *XSLT and XPath: A Guide to XML Transformations* by John Robert Gardner and Zarella L. Rendon
  - ISBN: 0-13-040446-2 (Prentice Hall PTR) [tutorial. XSLT and XPath 1.0]
- *Professional XSL* by Kurt Cagle, Michael Corning, et al.
  - ISBN: 1-861003-57-9 (WROX Press) [Detailed studies plus tutorial for XSLT/XPath 1.0]
- *Practical Transformation using XSLT and XPath* by Ken Holman
  - Eleventh Edition - ISBN 1-894049-12-8 - 2004-02-03. Available from his website: **`http://www.CraneSoftwrights.com`** (tutorial)
- *XSLT Cookbook* by Sal Mangano
  - ISBN: 0-7645-4776-3 (O'Reilly) [XSLT 1.0 Code samples plus; good for parameters, variables, named templates; a bit fond of pull]

slide 92

Colophon

- Slides and handouts created from single XML source
- Slides projected from HTML which was created from XML using XSLT
- Handouts created from XML:
  - Source XML transformed to Open Office XML
  - Open Office XML opened in Open Office
  - Pagination normally adjusted
  - Saved as PDF
- Slideshow materials available at:
  `http://www.mulberrytech.com/slideshow`

Appendix 1

Representative XSLT Tools

XSLT processors

Commonly used free tools:

- Saxon (Michael Kay; Java, open source):
  `http://sourceforge.net/projects/saxon/`
- Xerces and Xalan (IBM; Apache; Java, open source):
  `http://xml.apache.org/`
- XT (Java): **`http://www.blnz.com/xt/index.html`**
- MacTransform (Mac interface for Xalan): **`http://catcode.com/macxslt/`**. This unglitzy, information-rich page provides a download and covers how to install and use the kit.
- MSXML (recent Microsoft platforms):
  `http://msdn.microsoft.com/xml/`

XSLT editing tools

- Stylus Studio from eXcelon:
  `http://www.stylusstudio.com/xml_product_index.html`
  Development environment for creating, validating, and debugging XSL stylesheets and XML-to-XML mappings. $395
- XML Cooktop (Windows): **`http://www.xmlcooktop.com/`**. A free editor and development environment for XML, DTD, and XSLT documents. Listed here because of its XSLT support which include an XPath console.

- XMLSpy from Altova: **`http://www.xmlspy.com/`**. This XML editing tool includes an XSLT debugging feature. See listing under XML editors for details.
- Xselerator from Marrowsoft: **`http://www.marrowsoft.com/`**. A dedicated XSLT editor. Stylesheet navigation, wizards, debugging tools.
- ActiveState's Komodo and other products

More XSLT processors described at **`http://www.xml.com/pub/rg/XSLT_Software`**; see also **`http://www.oasis-open.org/cover/xslSoftware.html`**

Appendix 2

Acronyms Used in This Talk

aka	also known as
CME	Continuing Medical Education
CSS	Cascading Style Sheets
DTD	Document Type Definition
EDI	Electronic Data Interchange
GUI	Graphical User Interface
HTML	Hypertext Markup Language
OO	Object Oriented (applied to database)
MIF	FrameMaker internal format
PDA	Personal Digital Assistant
PDF	Portable Document Format
RDF	Resource Description Framework
RTF	Rich Text Format (Microsoft interchange format for Word)
SGML	Standard Generalized Markup Language
SVG	Scalable Vector Graphics
URL	Uniform (Universal) Resource Indicator
W3C	World Wide Web Consortium
WWW	World Wide Web
WYSIROTS	What You See Is Representative Of The Structure
XML	Extensible Markup Language
XSL	Extensible Stylesheet Language
XSL-FO	XSL Formatting Objects
XSLT	XSL Transformations