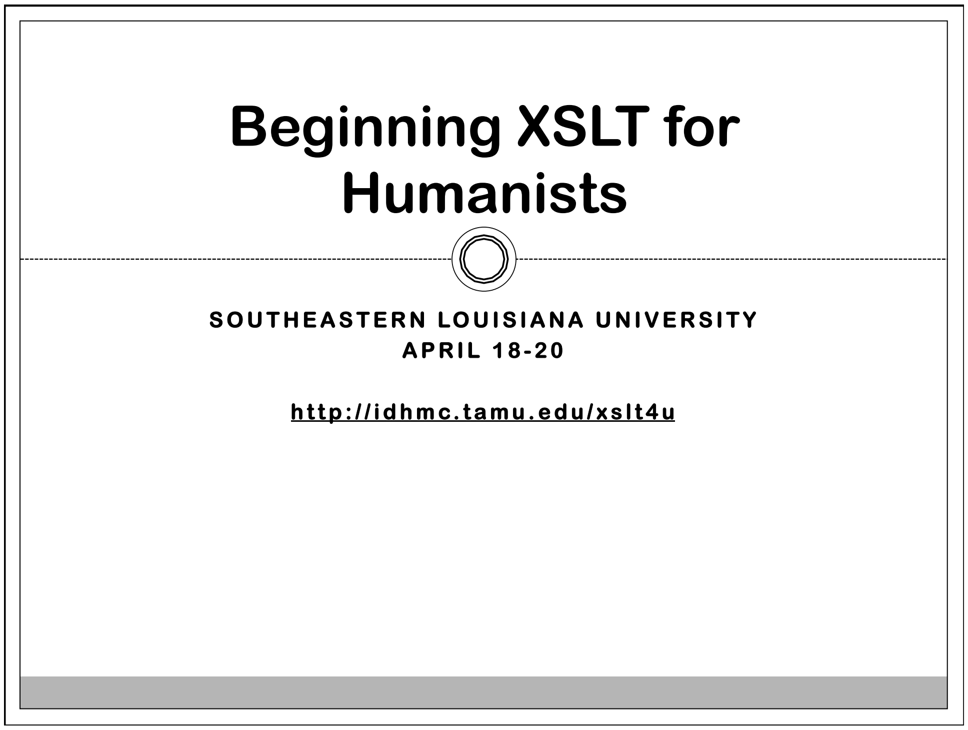
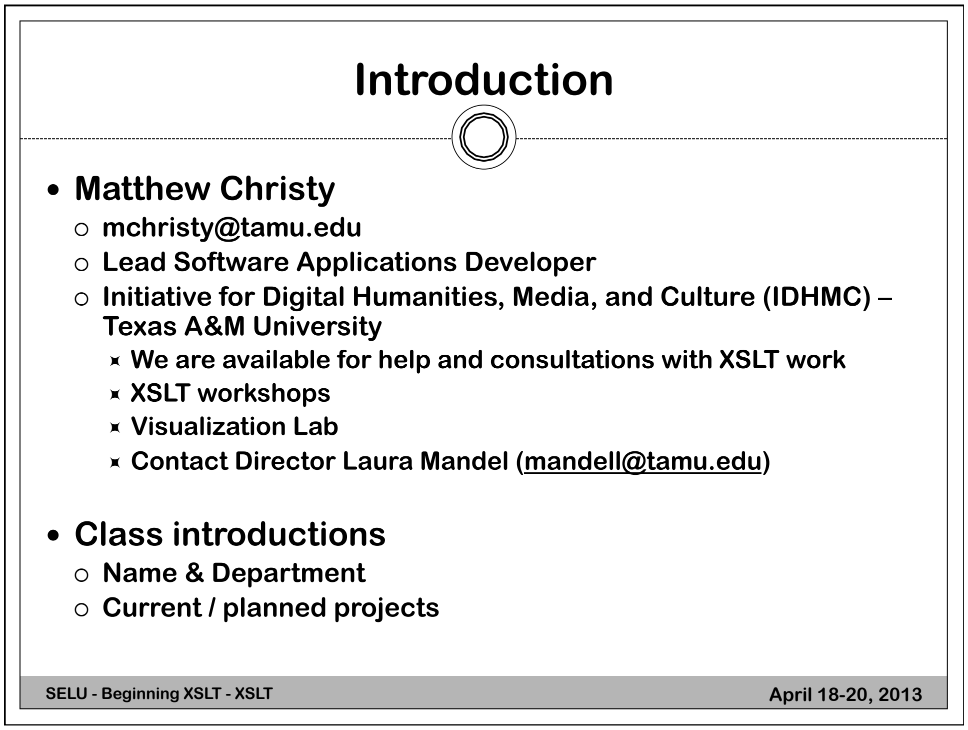


Beginning XSLT for Humanists

SOUTHEASTERN LOUISIANA UNIVERSITY
APRIL 18-20

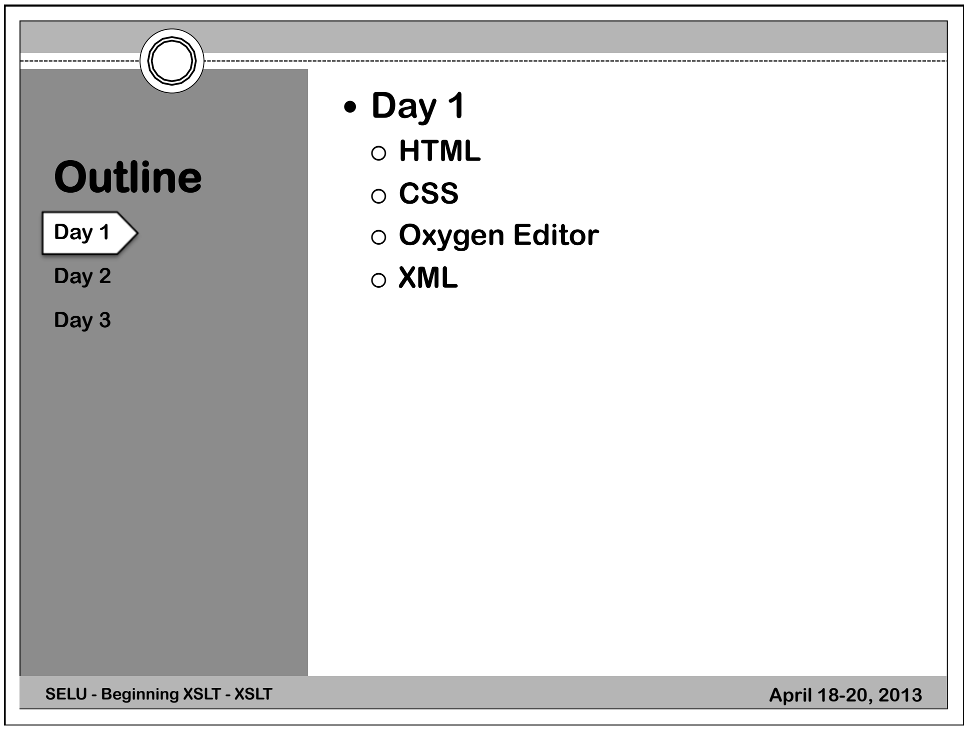
http://idhmc.tamu.edu/xslt4u



Introduction

- Matthew Christy
  - mchristy@tamu.edu
  - Lead Software Applications Developer
  - Initiative for Digital Humanities, Media, and Culture (IDHMC) – Texas A&M University
    - We are available for help and consultations with XSLT work
    - XSLT workshops
    - Visualization Lab
    - Contact Director Laura Mandel (mandell@tamu.edu)
- Class introductions
  - Name & Department
  - Current / planned projects

SELU - Beginning XSLT - XSLT April 18-20, 2013

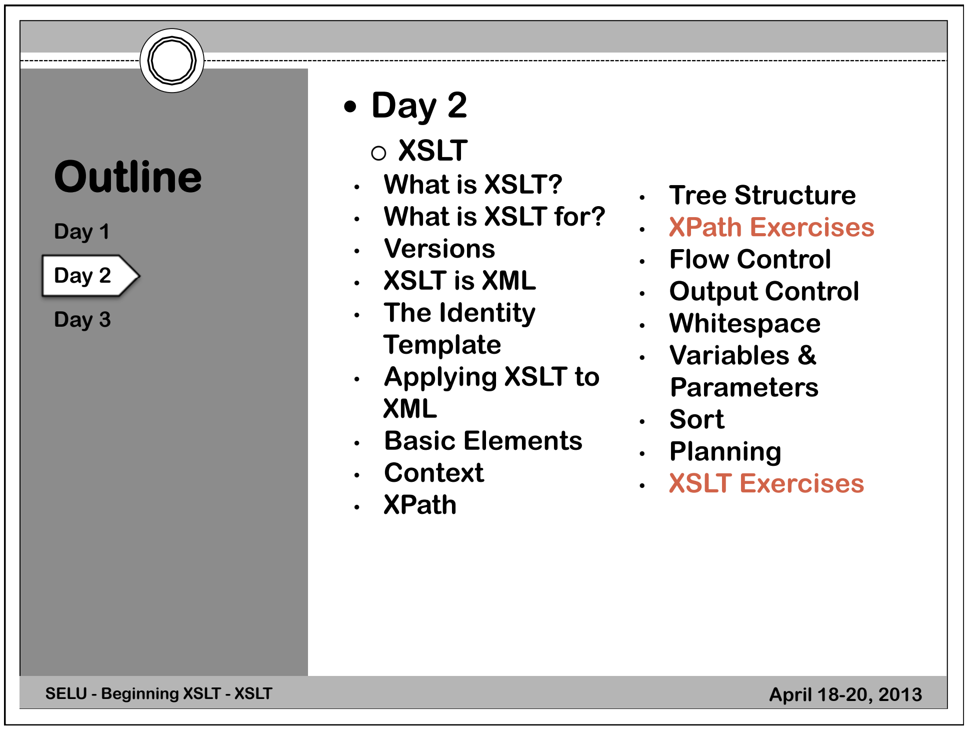


Outline

Day 1
Day 2
Day 3

- Day 1
  - HTML
  - CSS
  - Oxygen Editor
  - XML

SELU - Beginning XSLT - XSLT April 18-20, 2013

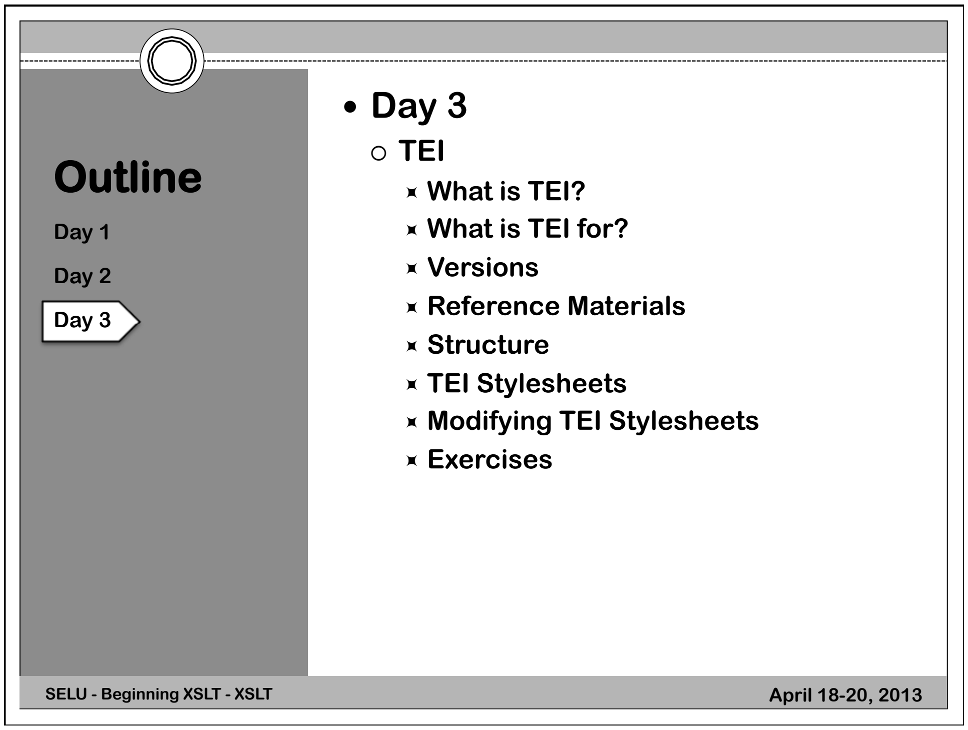


Outline

Day 1
Day 2
Day 3

- Day 2
  - XSLT
    - What is XSLT?
    - What is XSLT for?
    - Versions
    - XSLT is XML
    - The Identity Template
    - Applying XSLT to XML
    - Basic Elements
    - Context
    - XPath
    - Tree Structure
    - XPath Exercises
    - Flow Control
    - Output Control
    - Whitespace
    - Variables & Parameters
    - Sort
    - Planning
    - XSLT Exercises

SELU - Beginning XSLT - XSLT April 18-20, 2013



Outline

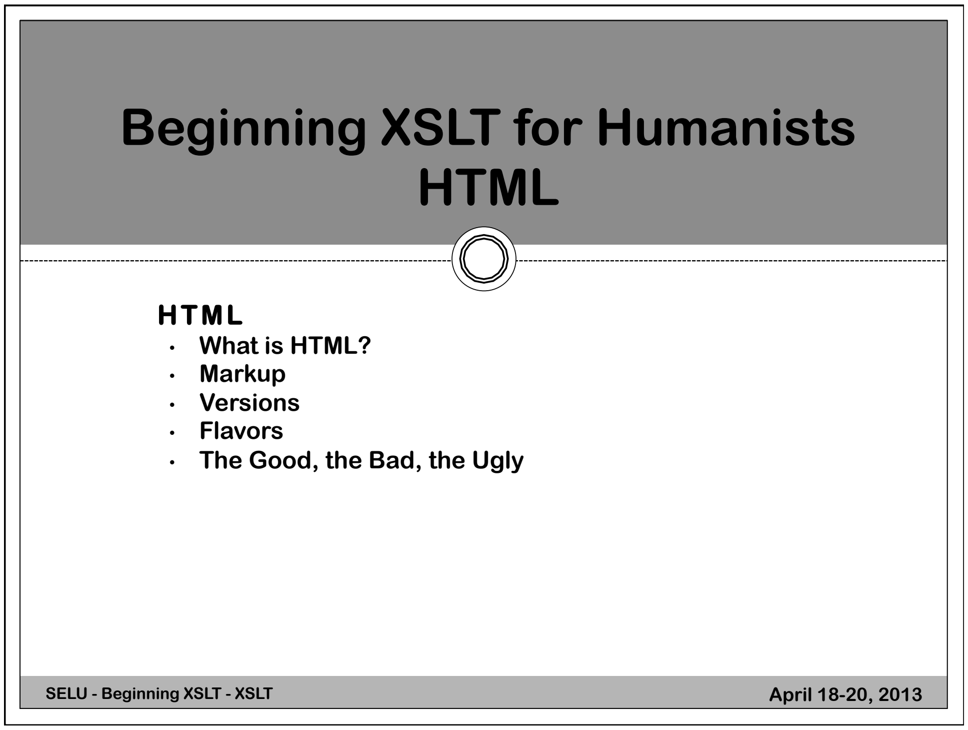
Day 1

Day 2

Day 3

- Day 3
  - TEI
    - What is TEI?
    - What is TEI for?
    - Versions
    - Reference Materials
    - Structure
    - TEI Stylesheets
    - Modifying TEI Stylesheets
    - Exercises

SELU - Beginning XSLT - XSLT April 18-20, 2013



Beginning XSLT for Humanists HTML

HTML

- What is HTML?
- Markup
- Versions
- Flavors
- The Good, the Bad, the Ugly

SELU - Beginning XSLT - XSLT April 18-20, 2013

What is HTML?

- Hyper Text Markup Language
  - *Hypertext*: text displayed on a computer with references (hyperlinks) to that provide access to other texts.
  - *Markup*: a mechanism that uses special characters or syntax to annotate a text.
  - Language: a protocol for communication. In this case the language is used to instruct the client (browser) how to render the text.

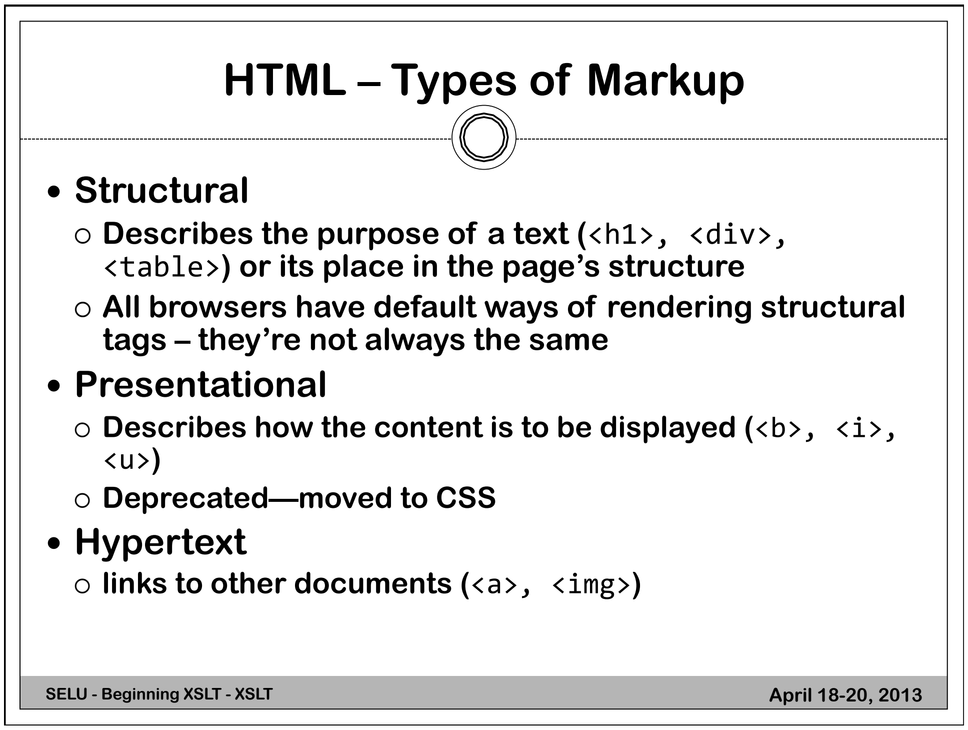
"HyperText Markup Language is a markup language that web browsers use to interpret and compose text, images and other material into visual or audible web pages."

-wikipedia [http://en.wikipedia.org/wiki/Html#Development]

HTML – Markup

- Elements
  - Tags
    - Denoted by <>: `<tagname>`
    - Tags open and close (start and end): `<tag>…</tag>`
    - Tags are nested
      - Yes: `<h1><b>Title</b>-Subtitle</h1>`
      - No: `<b><h1>Title</b>-Subtitle</h1>`
  - Attributes
    - A name-value pair within a start tag: `<tag att1="val1">`
    - Attribute names are unique within a tag
    - Attribute values are applied to the tag or the tag's contents
  - Content
    - Text between the start/end tags

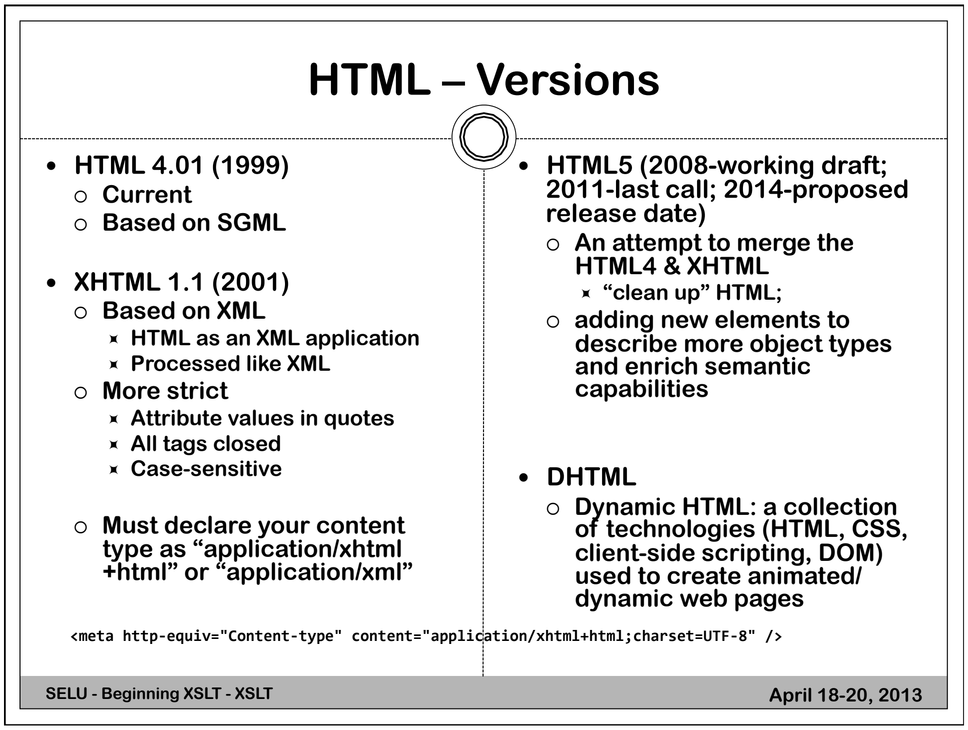
In HTML all tag and attribute names are pre-defined. You can't just make up your own.



HTML – Types of Markup

- Structural
  - Describes the purpose of a text (`<h1>, <div>, <table>`) or its place in the page's structure
  - All browsers have default ways of rendering structural tags – they're not always the same
- Presentational
  - Describes how the content is to be displayed (`<b>, <i>, <u>`)
  - Deprecated—moved to CSS
- Hypertext
  - links to other documents (`<a>, <img>`)

SELU - Beginning XSLT - XSLT April 18-20, 2013

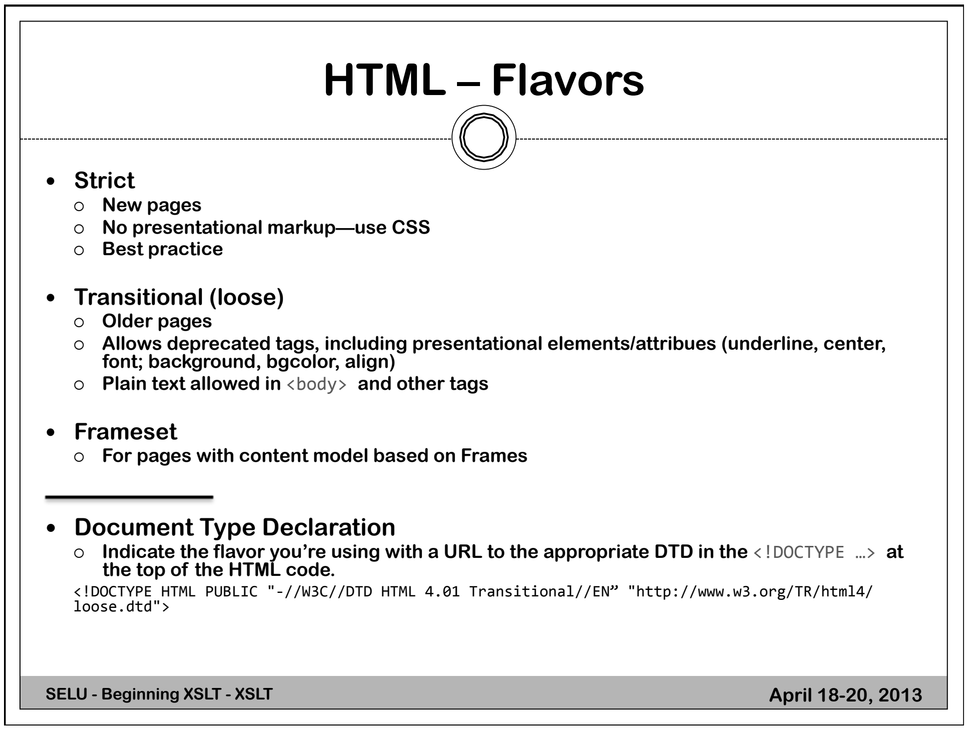


HTML – Versions

- HTML 4.01 (1999)
  - Current
  - Based on SGML
- XHTML 1.1 (2001)
  - Based on XML
    - HTML as an XML application
    - Processed like XML
  - More strict
    - Attribute values in quotes
    - All tags closed
    - Case-sensitive
  - Must declare your content type as "application/xhtml +html" or "application/xml"
- HTML5 (2008-working draft; 2011-last call; 2014-proposed release date)
  - An attempt to merge the HTML4 & XHTML
    - "clean up" HTML;
  - adding new elements to describe more object types and enrich semantic capabilities
- DHTML
  - Dynamic HTML: a collection of technologies (HTML, CSS, client-side scripting, DOM) used to create animated/ dynamic web pages

```
<meta http-equiv="Content-type" content="application/xhtml+html;charset=UTF-8" />
```

SELU - Beginning XSLT - XSLT April 18-20, 2013



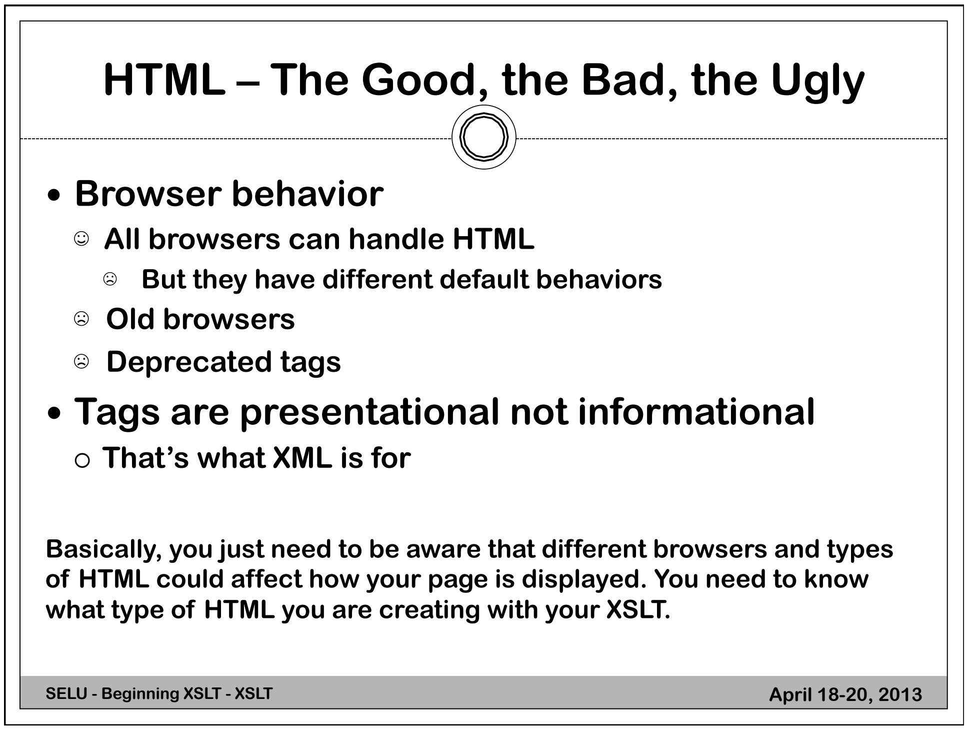
HTML – Flavors

- **Strict**
  - New pages
  - No presentational markup—use CSS
  - Best practice
- **Transitional (loose)**
  - Older pages
  - Allows deprecated tags, including presentational elements/attribues (underline, center, font; background, bgcolor, align)
  - Plain text allowed in `<body>` and other tags
- **Frameset**
  - For pages with content model based on Frames

- **Document Type Declaration**
  - Indicate the flavor you're using with a URL to the appropriate DTD in the `<!DOCTYPE …>` at the top of the HTML code.

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN" "http://www.w3.org/TR/html4/
loose.dtd">
```

SELU - Beginning XSLT - XSLT April 18-20, 2013

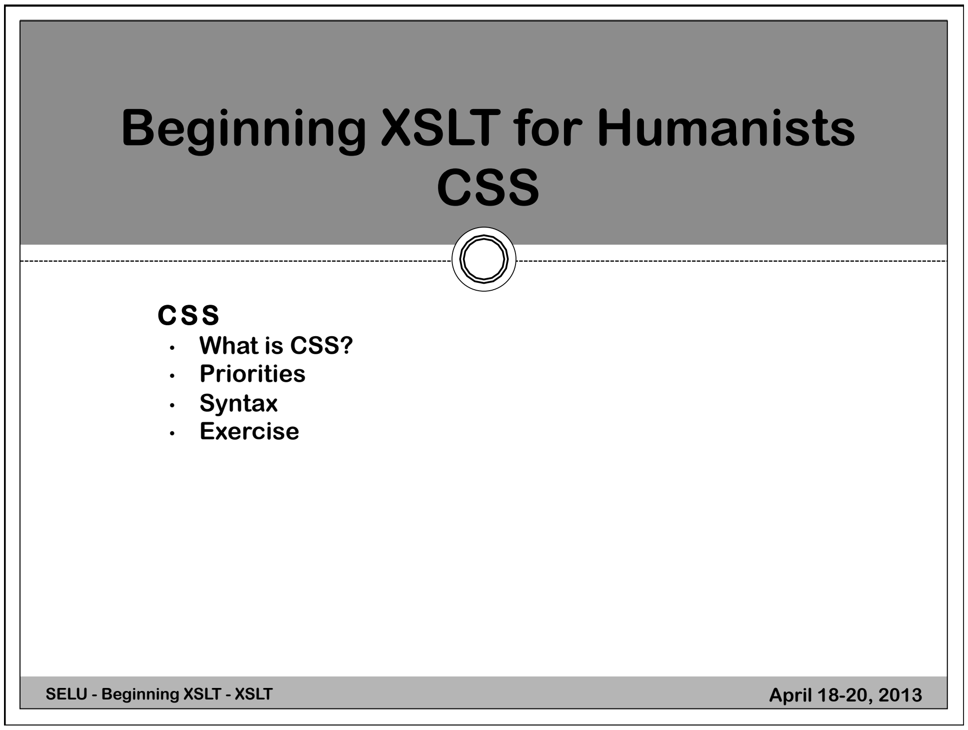


HTML – The Good, the Bad, the Ugly

- **Browser behavior**
  - ☺ All browsers can handle HTML
    - ☹ But they have different default behaviors
  - ☹ Old browsers
  - ☹ Deprecated tags
- **Tags are presentational not informational**
  - That's what XML is for

Basically, you just need to be aware that different browsers and types of HTML could affect how your page is displayed. You need to know what type of HTML you are creating with your XSLT.

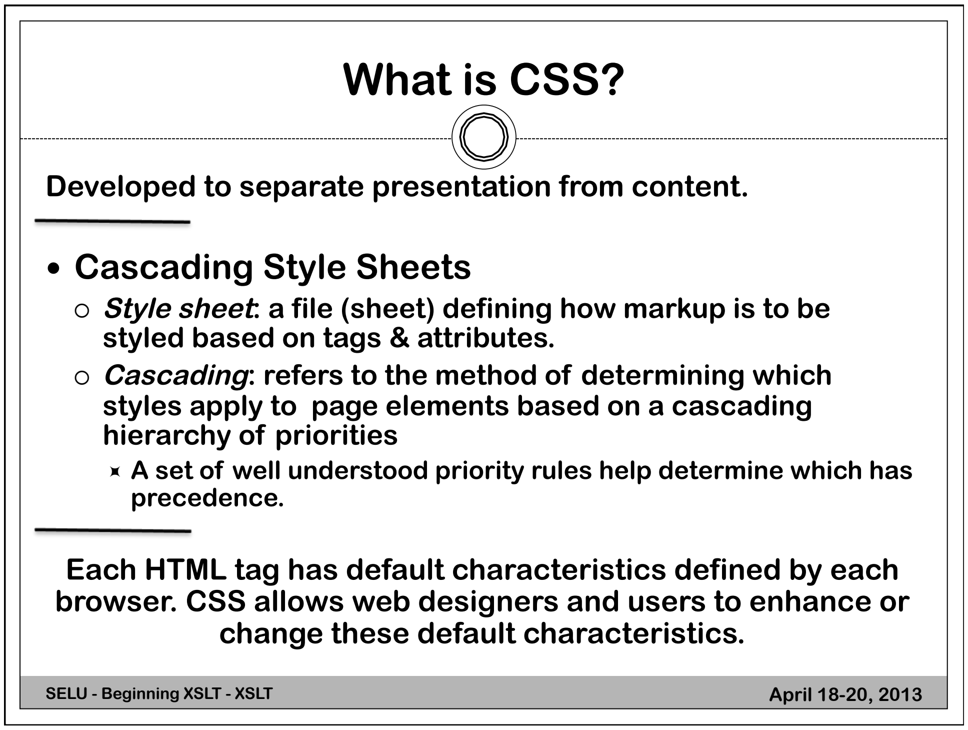
SELU - Beginning XSLT - XSLT April 18-20, 2013



Beginning XSLT for Humanists CSS

CSS

- What is CSS?
- Priorities
- Syntax
- Exercise



What is CSS?

Developed to separate presentation from content.

- Cascading Style Sheets
  - *Style sheet*: a file (sheet) defining how markup is to be styled based on tags & attributes.
  - *Cascading*: refers to the method of determining which styles apply to page elements based on a cascading hierarchy of priorities
    - A set of well understood priority rules help determine which has precedence.

Each HTML tag has default characteristics defined by each browser. CSS allows web designers and users to enhance or change these default characteristics.

CSS – Priorities

Lowest

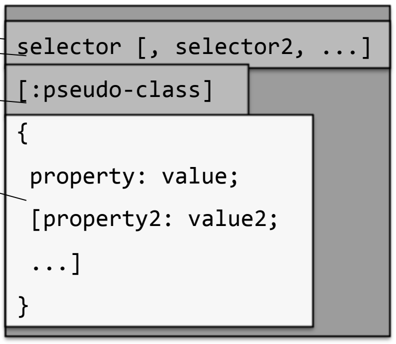
Highest

- Browser default
- External style sheet
- Internal style sheet (in `<head>` segment)
- Inline style (`<style>` attribute inside a tag)

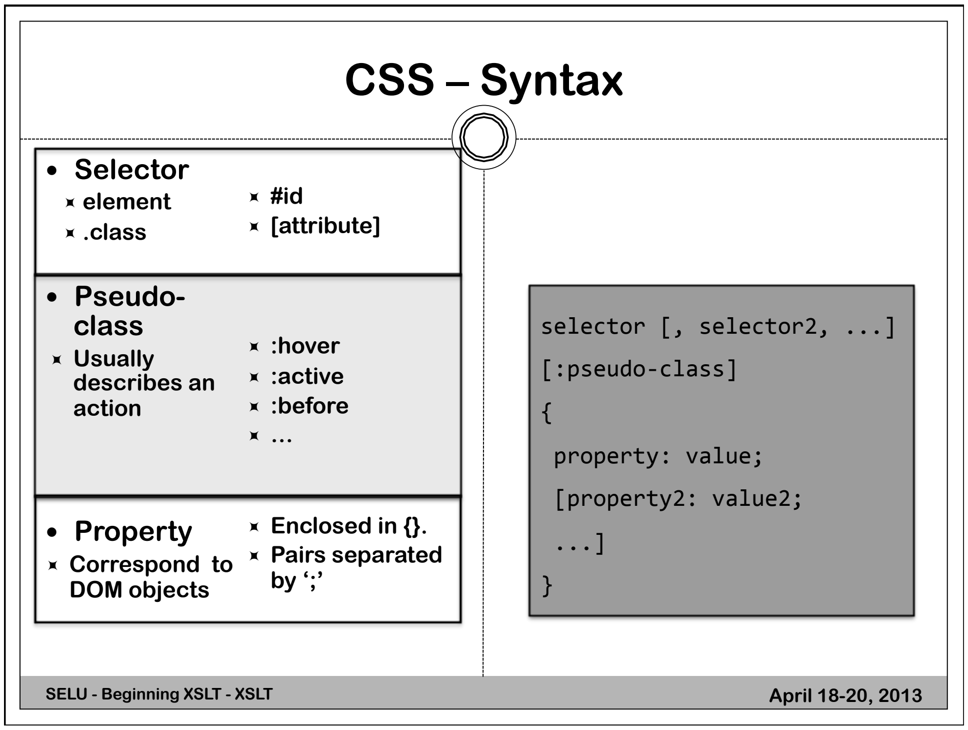
- In addition, some properties are inherited from parent tags.
- If two rules have the same priority level, the latter one wins.

CSS – Syntax

- CSS is made up of a series of rules
  - A rule consists of one or more selectors with an optional pseudo-class and a declaration block.
  - A declaration block consists of one or more sets of property-value pairs.



```
selector [, selector2, ...]
[:pseudo-class]
{
 property: value;
 [property2: value2;
 ...]
}
```

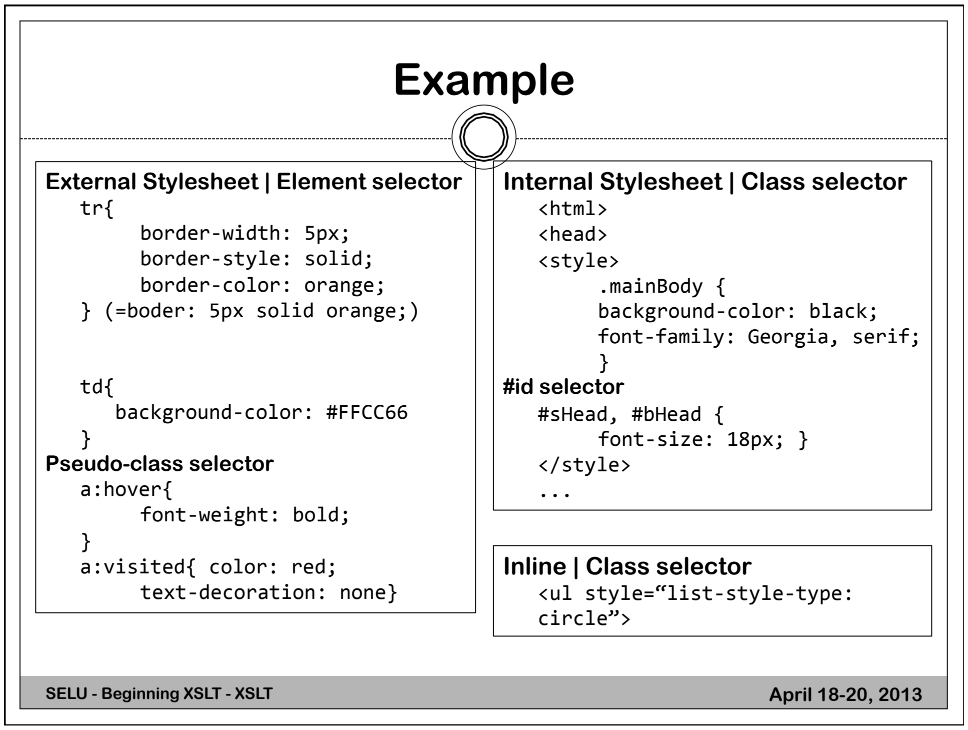


CSS – Syntax

- **Selector**
  - element
  - .class
  - #id
  - [attribute]
- **Pseudo-class**
  - Usually describes an action
  - :hover
  - :active
  - :before
  - …
- **Property**
  - Correspond to DOM objects
  - Enclosed in {}.
  - Pairs separated by ‘;’

```
selector [, selector2, ...]
[:pseudo-class]
{
 property: value;
 [property2: value2;
 ...]
}
```

SELU - Beginning XSLT - XSLT April 18-20, 2013



Example

External Stylesheet | Element selector

```
tr{
     border-width: 5px;
     border-style: solid;
     border-color: orange;
} (=boder: 5px solid orange;)


td{
   background-color: #FFCC66
}
```

Pseudo-class selector

```
a:hover{
     font-weight: bold;
}
a:visited{ color: red;
     text-decoration: none}
```

Internal Stylesheet | Class selector

```
<html>
<head>
<style>
     .mainBody {
     background-color: black;
     font-family: Georgia, serif;
     }
```

#id selector

```
#sHead, #bHead {
     font-size: 18px; }
</style>
...
```

Inline | Class selector

```
<ul style="list-style-type:
circle">
```

SELU - Beginning XSLT - XSLT April 18-20, 2013

Exercise 1

Start with example files Library-wCSS-2.html and libStyle-2.css and make the following modifications:

- ❑ Visually separate headers “Books” and “Serials” from the rest of the table – center the text, etc.
- ❑ Create more visual space between the Books and Serials sections
  - Try creating two separate tables, or formatting the empty row between them
- ❑ Visually connect the book and serial titles with the rest of the bibliographic data – remove the lines separating them.
- ❑ Fix the problem of authors & subtitles being on the line below the title.

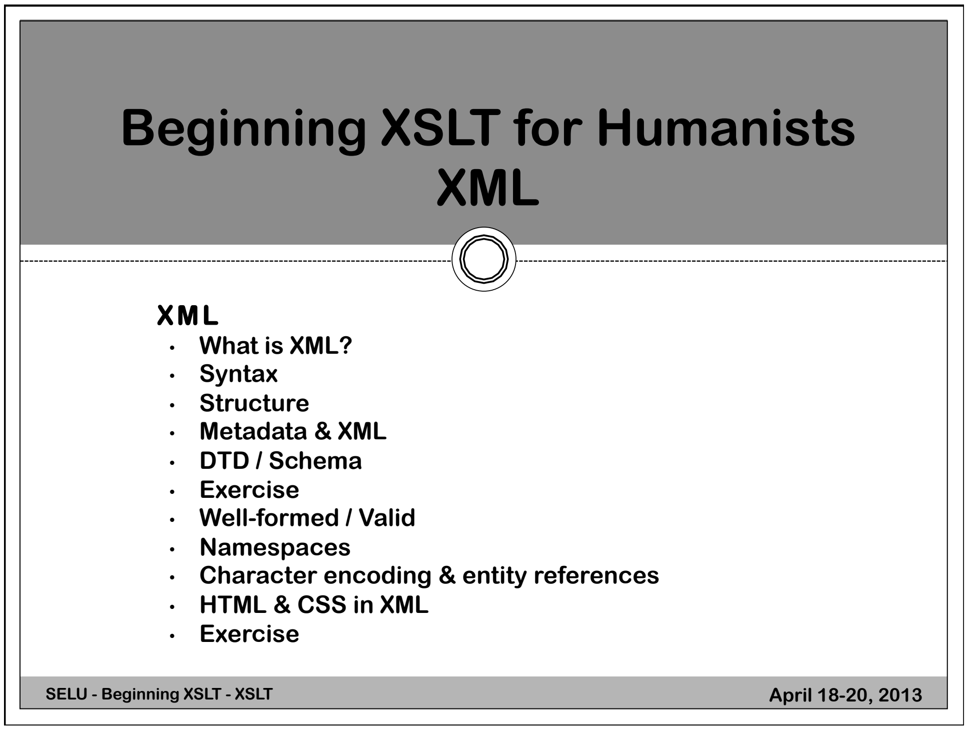
SELU - Beginning XSLT - XSLT April 18-20, 2013

Beginning XSLT for Humanists Oxygen Editor

OXYGEN

- Install
- Setup
  - Blank Spaces

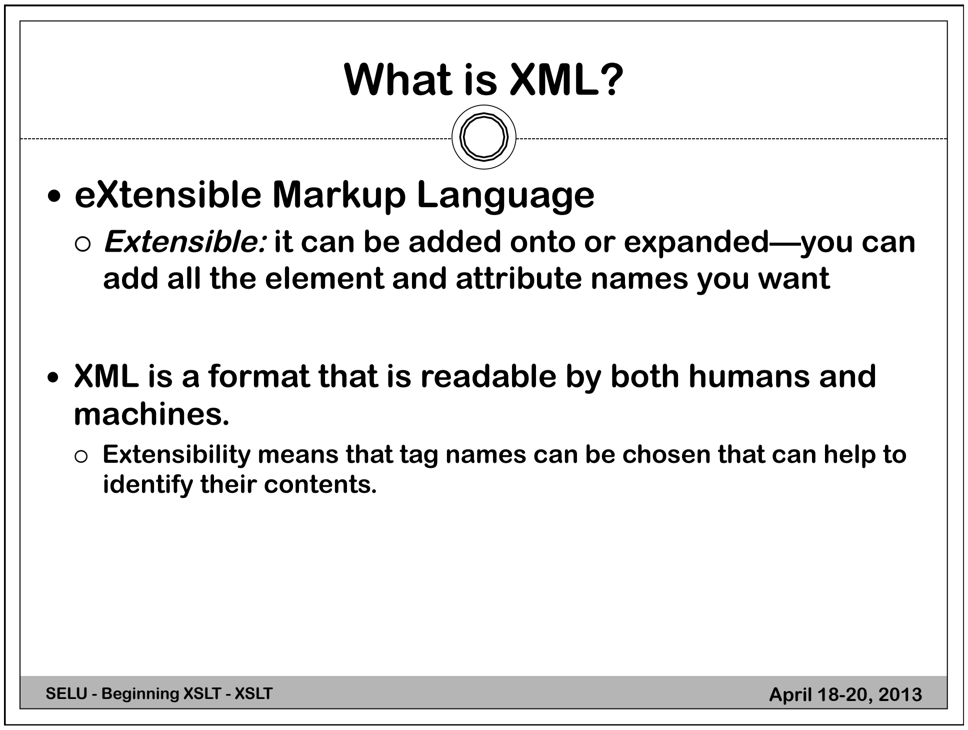
SELU - Beginning XSLT - XSLT April 18-20, 2013



Beginning XSLT for Humanists XML

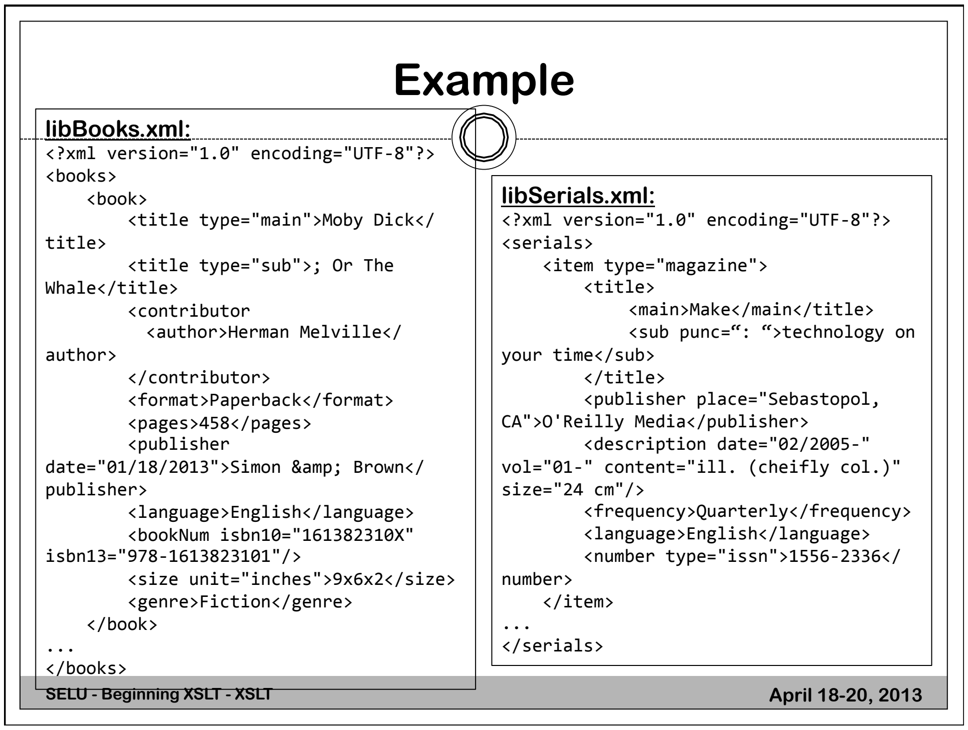
XML

- What is XML?
- Syntax
- Structure
- Metadata & XML
- DTD / Schema
- Exercise
- Well-formed / Valid
- Namespaces
- Character encoding & entity references
- HTML & CSS in XML
- Exercise



What is XML?

- eXtensible Markup Language
  - *Extensible:* it can be added onto or expanded—you can add all the element and attribute names you want
- XML is a format that is readable by both humans and machines.
  - Extensibility means that tag names can be chosen that can help to identify their contents.



Example

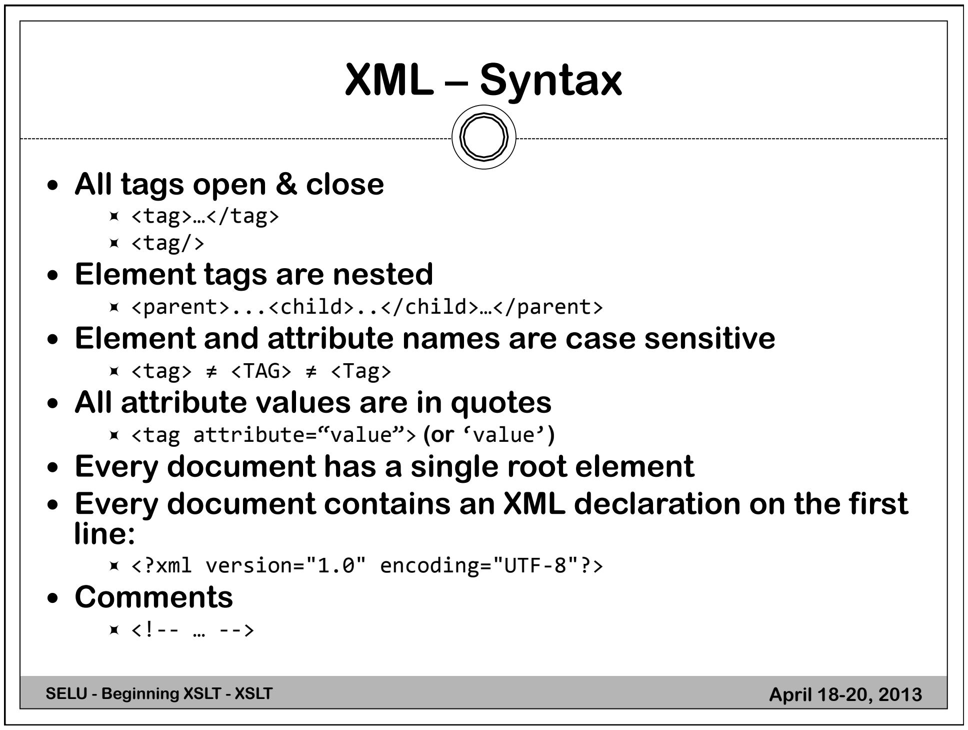
libBooks.xml:

```
<?xml version="1.0" encoding="UTF-8"?>
<books>
    <book>
        <title type="main">Moby Dick</
title>
        <title type="sub">; Or The
Whale</title>
        <contributor
          <author>Herman Melville</
author>
        </contributor>
        <format>Paperback</format>
        <pages>458</pages>
        <publisher
date="01/18/2013">Simon &amp; Brown</
publisher>
        <language>English</language>
        <bookNum isbn10="161382310X"
isbn13="978-1613823101"/>
        <size unit="inches">9x6x2</size>
        <genre>Fiction</genre>
    </book>
...
</books>
```

libSerials.xml:

```
<?xml version="1.0" encoding="UTF-8"?>
<serials>
    <item type="magazine">
        <title>
            <main>Make</main></title>
            <sub punc=“: “>technology on
your time</sub>
        </title>
        <publisher place="Sebastopol,
CA">O'Reilly Media</publisher>
        <description date="02/2005-"
vol="01-" content="ill. (cheifly col.)"
size="24 cm"/>
        <frequency>Quarterly</frequency>
        <language>English</language>
        <number type="issn">1556-2336</
number>
    </item>
...
</serials>
```

SELU - Beginning XSLT - XSLT April 18-20, 2013



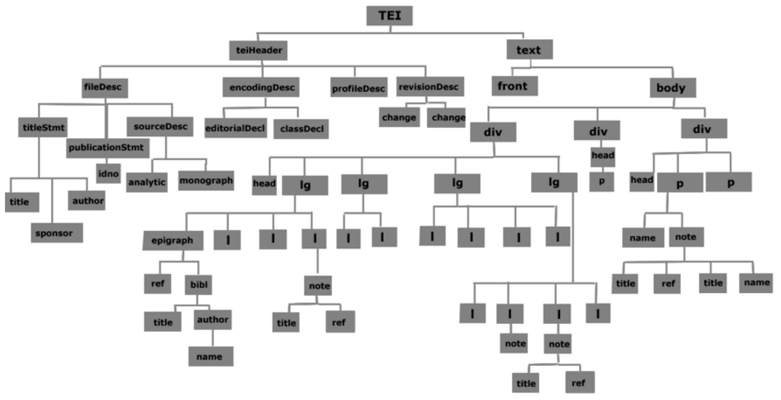
XML – Syntax

- **All tags open & close**
  - `<tag>…</tag>`
  - `<tag/>`
- **Element tags are nested**
  - `<parent>...<child>..</child>…</parent>`
- **Element and attribute names are case sensitive**
  - `<tag> ≠ <TAG> ≠ <Tag>`
- **All attribute values are in quotes**
  - `<tag attribute=“value”>` (or `‘value’`)
- **Every document has a single root element**
- **Every document contains an XML declaration on the first line:**
  - `<?xml version="1.0" encoding="UTF-8"?>`
- **Comments**
  - `<!-- … -->`

SELU - Beginning XSLT - XSLT April 18-20, 2013

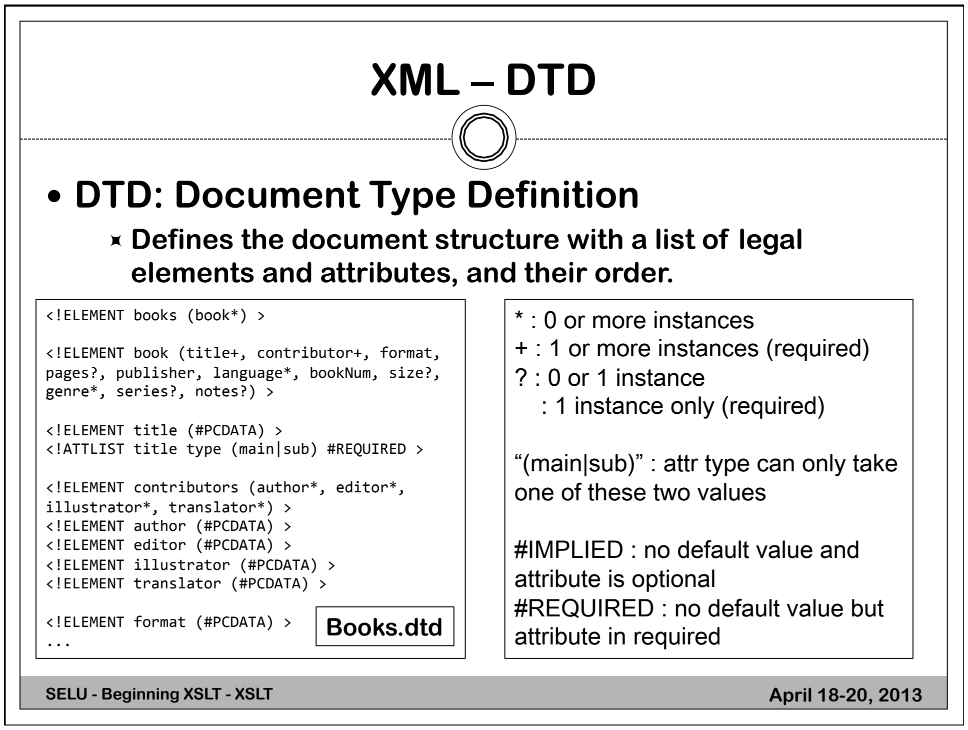
XML – Structure

- XML documents have a tree-like hierarchical structure



XML – Metadata & XML

- Metadata is data about data, or about objects
  - Objects can be physical or digital
  - XML tags
    - Meaningful element/attribute names constitute metadata about data in the XML DB
  - About a physical object
    - A physical description; like library catalog entry
  - About a digital object
    - File type; creation; creator; date created; can include physical object metadata as well
  - About a file
    - How is the file encoded; its source(s); who edited it; etc.



XML – DTD

- **DTD: Document Type Definition**
  - **Defines the document structure with a list of legal elements and attributes, and their order.**

```
<!ELEMENT books (book*) >

<!ELEMENT book (title+, contributor+, format,
pages?, publisher, language*, bookNum, size?,
genre*, series?, notes?) >

<!ELEMENT title (#PCDATA) >
<!ATTLIST title type (main|sub) #REQUIRED >

<!ELEMENT contributors (author*, editor*,
illustrator*, translator*) >
<!ELEMENT author (#PCDATA) >
<!ELEMENT editor (#PCDATA) >
<!ELEMENT illustrator (#PCDATA) >
<!ELEMENT translator (#PCDATA) >

<!ELEMENT format (#PCDATA) >
...
```

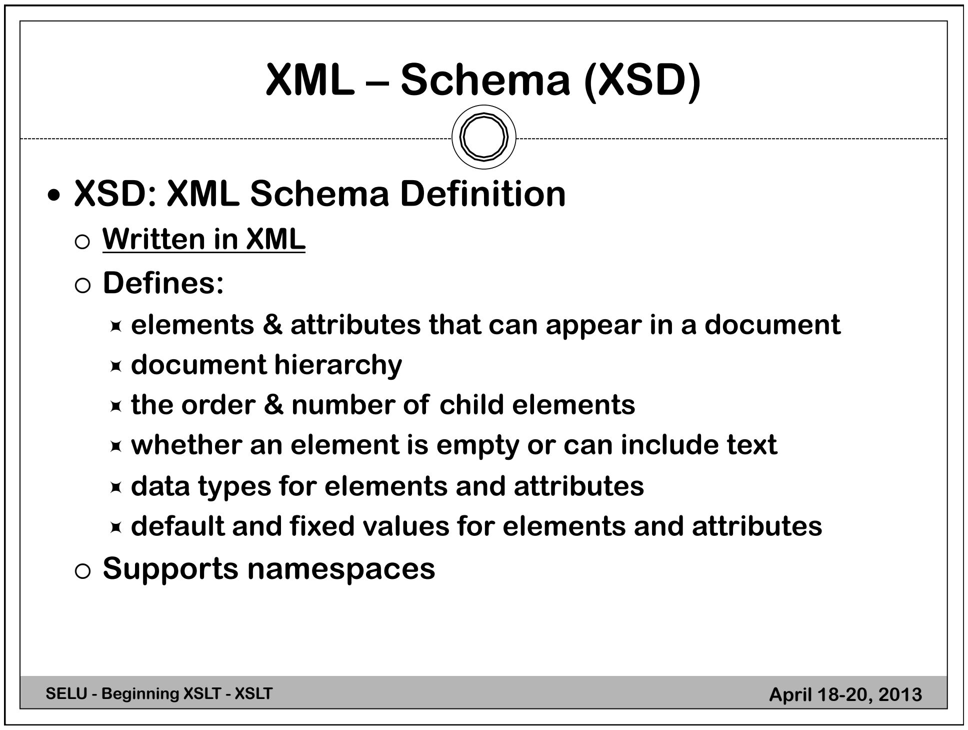
Books.dtd

* : 0 or more instances
+ : 1 or more instances (required)
? : 0 or 1 instance
  : 1 instance only (required)

"(main|sub)" : attr type can only take one of these two values

#IMPLIED : no default value and attribute is optional
#REQUIRED : no default value but attribute in required

SELU - Beginning XSLT - XSLT April 18-20, 2013



XML – Schema (XSD)

- **XSD: XML Schema Definition**
  - **<u>Written in XML</u>**
  - **Defines:**
    - **elements & attributes that can appear in a document**
    - **document hierarchy**
    - **the order & number of child elements**
    - **whether an element is empty or can include text**
    - **data types for elements and attributes**
    - **default and fixed values for elements and attributes**
  - **Supports namespaces**

SELU - Beginning XSLT - XSLT April 18-20, 2013

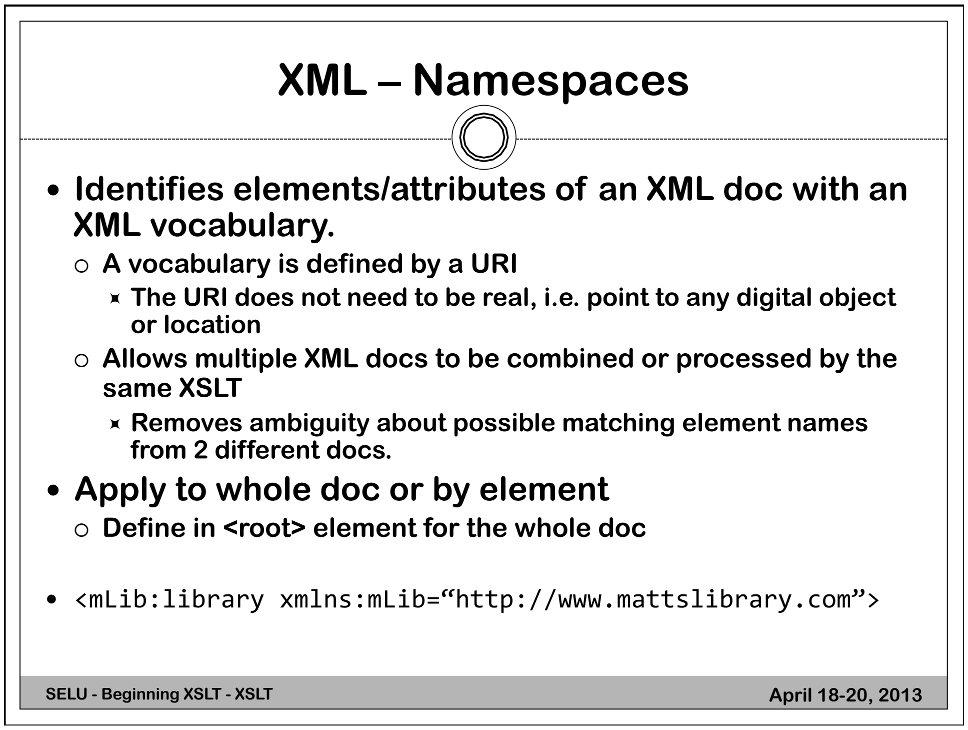
Exercise 2

Convert Books.dtd to libBooks.xsd

- ❑ Let’s just do the <books>, <book>, <title>, and <contributors> elements.

XML – Well-formed / Valid

- Well-formed XML
  - Obeys all the rules of XML syntax
- Valid XML
  - Is well-formed AND
  - Obeys all the rules established by the DTD/XSD

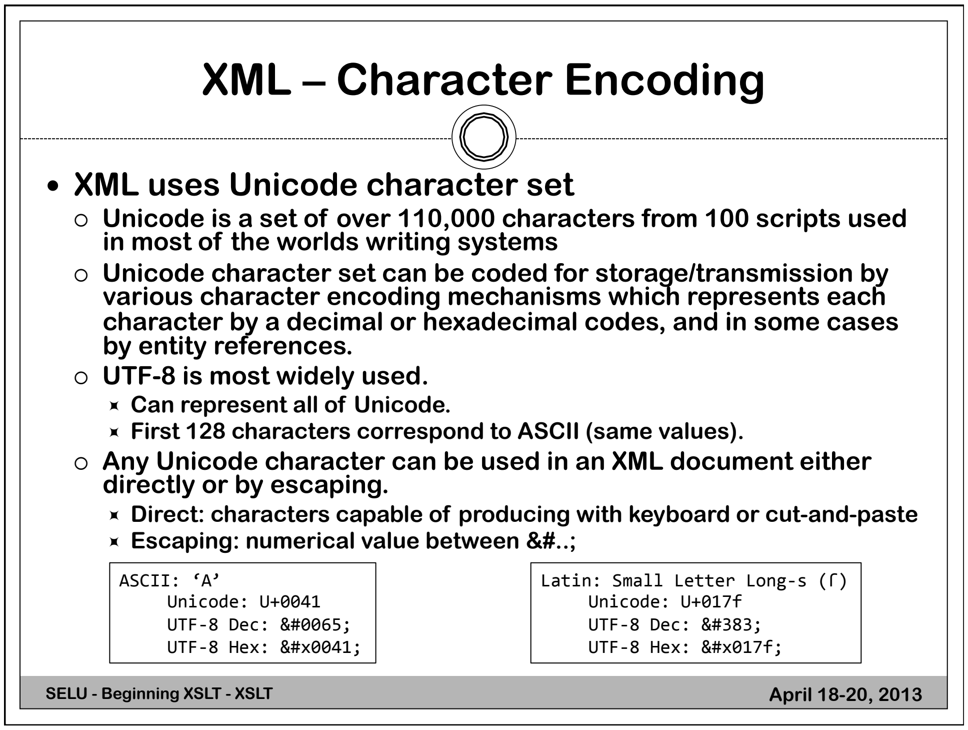


XML – Namespaces

- Identifies elements/attributes of an XML doc with an XML vocabulary.
  - A vocabulary is defined by a URI
    - The URI does not need to be real, i.e. point to any digital object or location
  - Allows multiple XML docs to be combined or processed by the same XSLT
    - Removes ambiguity about possible matching element names from 2 different docs.
- Apply to whole doc or by element
  - Define in <root> element for the whole doc

- `<mLib:library xmlns:mLib=“http://www.mattslibrary.com”>`

SELU - Beginning XSLT - XSLT April 18-20, 2013



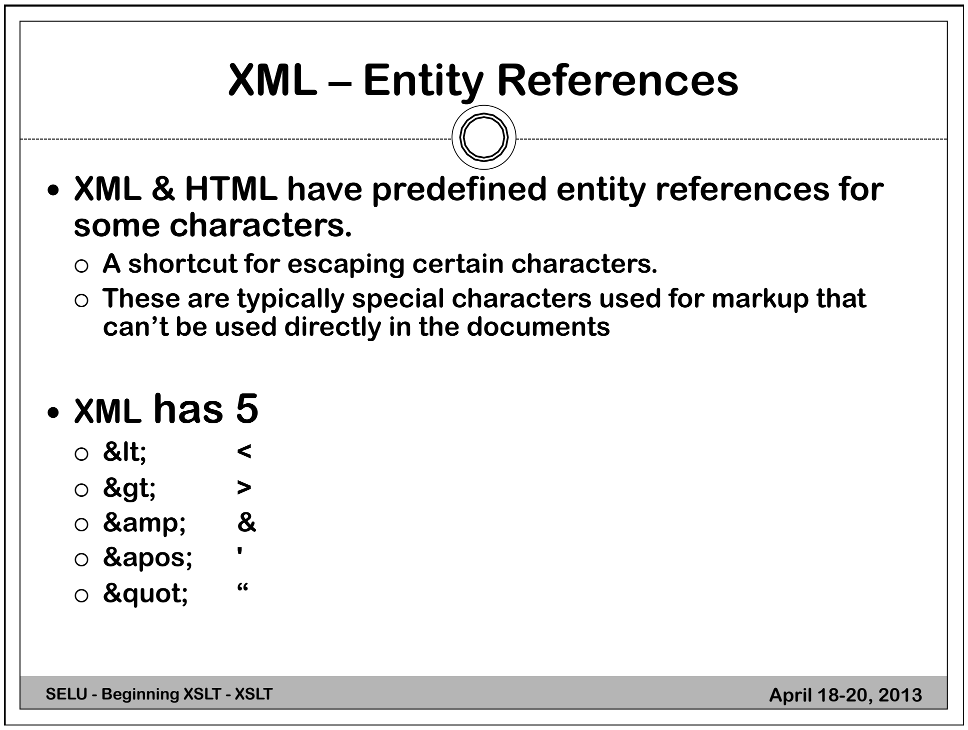
XML – Character Encoding

- XML uses Unicode character set
  - Unicode is a set of over 110,000 characters from 100 scripts used in most of the worlds writing systems
  - Unicode character set can be coded for storage/transmission by various character encoding mechanisms which represents each character by a decimal or hexadecimal codes, and in some cases by entity references.
  - UTF-8 is most widely used.
    - Can represent all of Unicode.
    - First 128 characters correspond to ASCII (same values).
  - Any Unicode character can be used in an XML document either directly or by escaping.
    - Direct: characters capable of producing with keyboard or cut-and-paste
    - Escaping: numerical value between &#..;

```
ASCII: ‘A’
     Unicode: U+0041
     UTF-8 Dec: &#0065;
     UTF-8 Hex: &#x0041;
```

```
Latin: Small Letter Long-s (ſ)
     Unicode: U+017f
     UTF-8 Dec: &#383;
     UTF-8 Hex: &#x017f;
```

SELU - Beginning XSLT - XSLT April 18-20, 2013

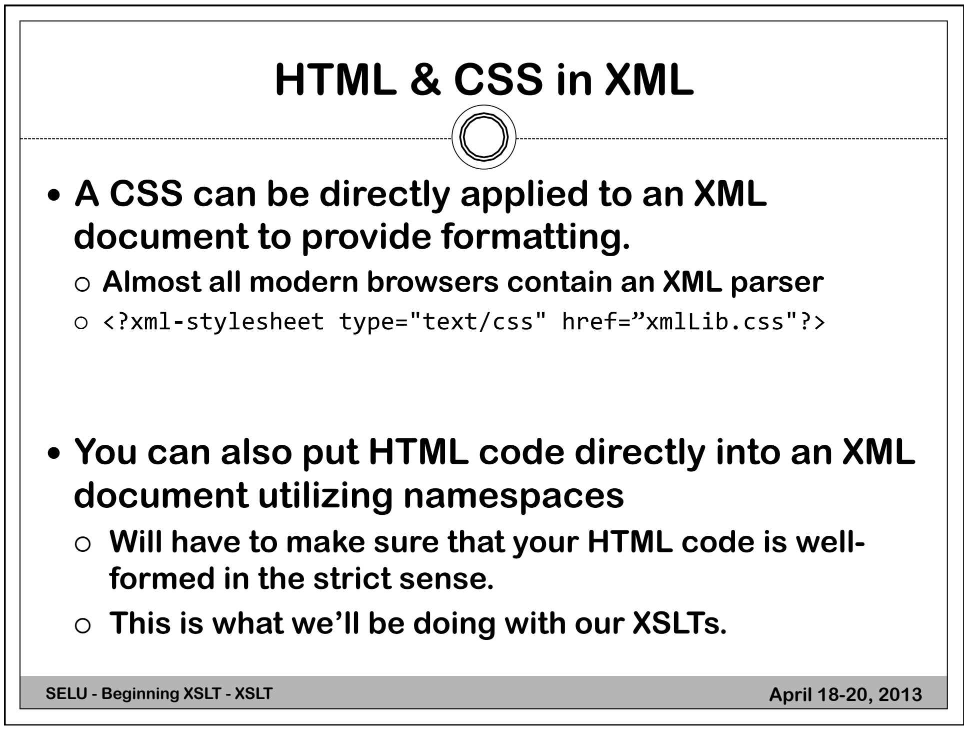


XML – Entity References

- XML & HTML have predefined entity references for some characters.
  - A shortcut for escaping certain characters.
  - These are typically special characters used for markup that can’t be used directly in the documents
- XML has 5

&lt;	<
&gt;	>
&amp;	&
&apos;	'
&quot;	“

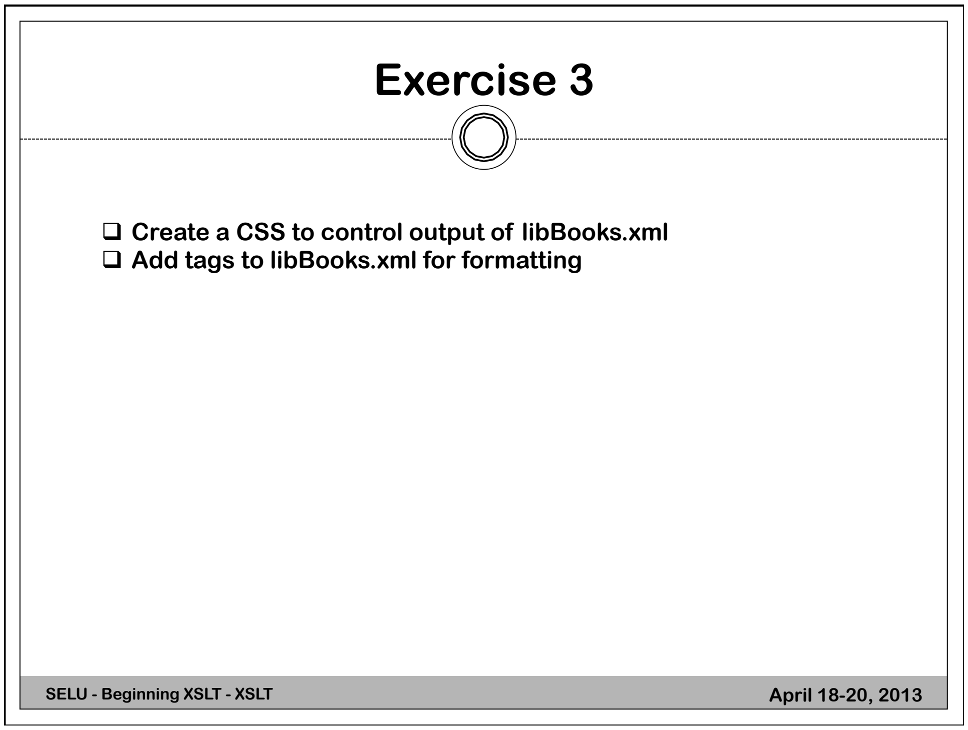
SELU - Beginning XSLT - XSLT April 18-20, 2013



HTML & CSS in XML

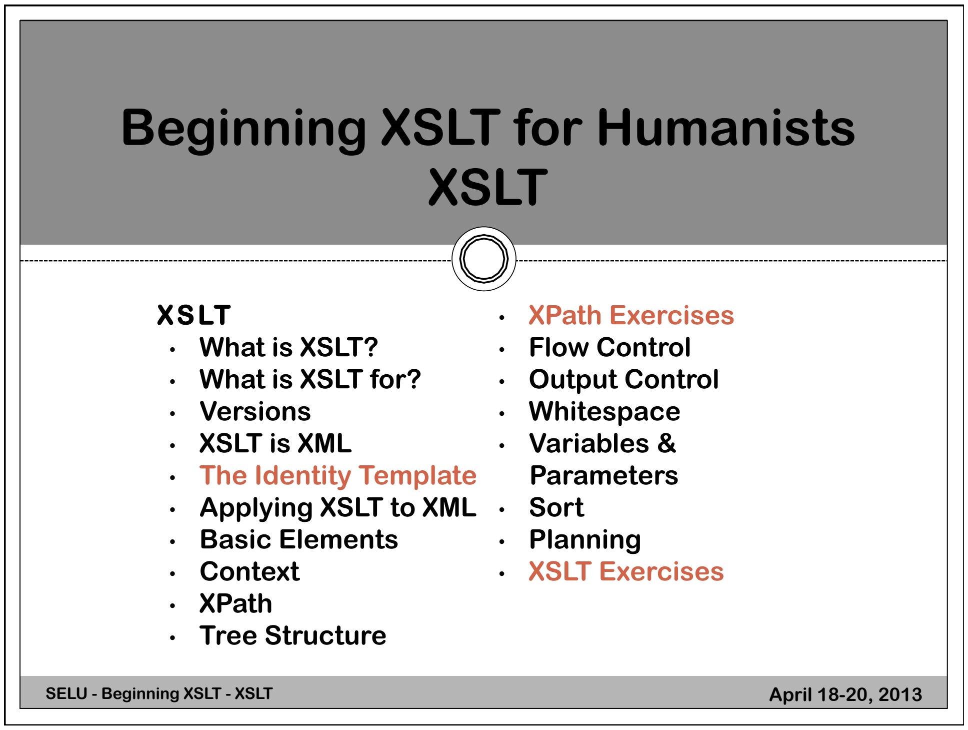
- A CSS can be directly applied to an XML document to provide formatting.
  - Almost all modern browsers contain an XML parser
  - `<?xml-stylesheet type="text/css" href=”xmlLib.css"?>`
- You can also put HTML code directly into an XML document utilizing namespaces
  - Will have to make sure that your HTML code is well-formed in the strict sense.
  - This is what we’ll be doing with our XSLTs.

SELU - Beginning XSLT - XSLT April 18-20, 2013



Exercise 3

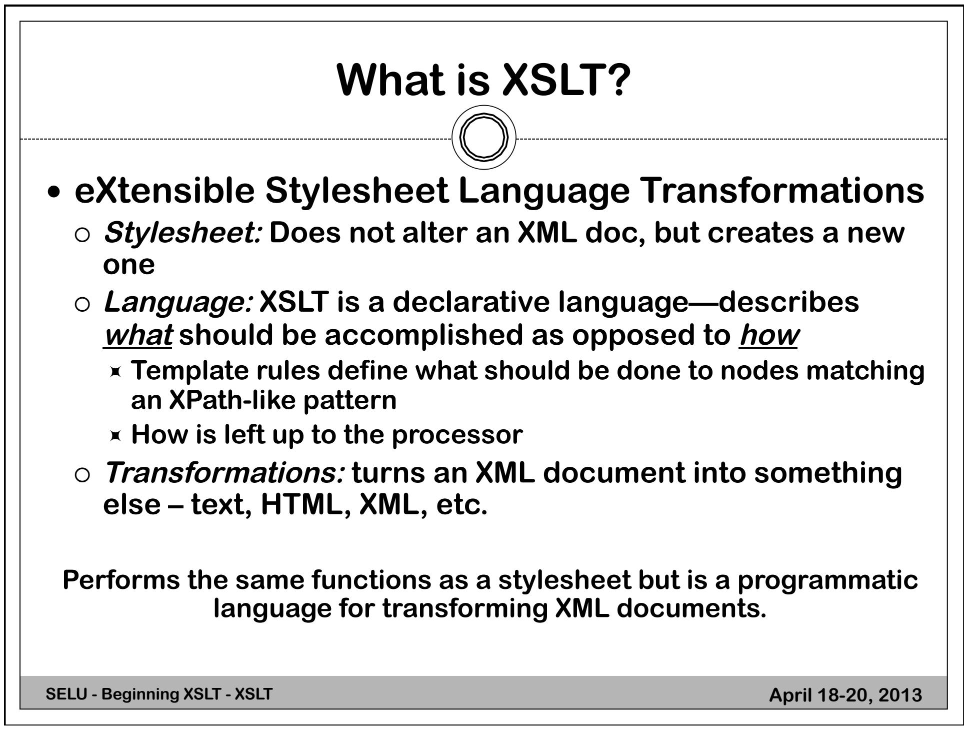
- ❑ Create a CSS to control output of libBooks.xml
- ❑ Add tags to libBooks.xml for formatting



Beginning XSLT for Humanists XSLT

XSLT

- What is XSLT?
- What is XSLT for?
- Versions
- XSLT is XML
- The Identity Template
- Applying XSLT to XML
- Basic Elements
- Context
- XPath
- Tree Structure
- XPath Exercises
- Flow Control
- Output Control
- Whitespace
- Variables & Parameters
- Sort
- Planning
- XSLT Exercises

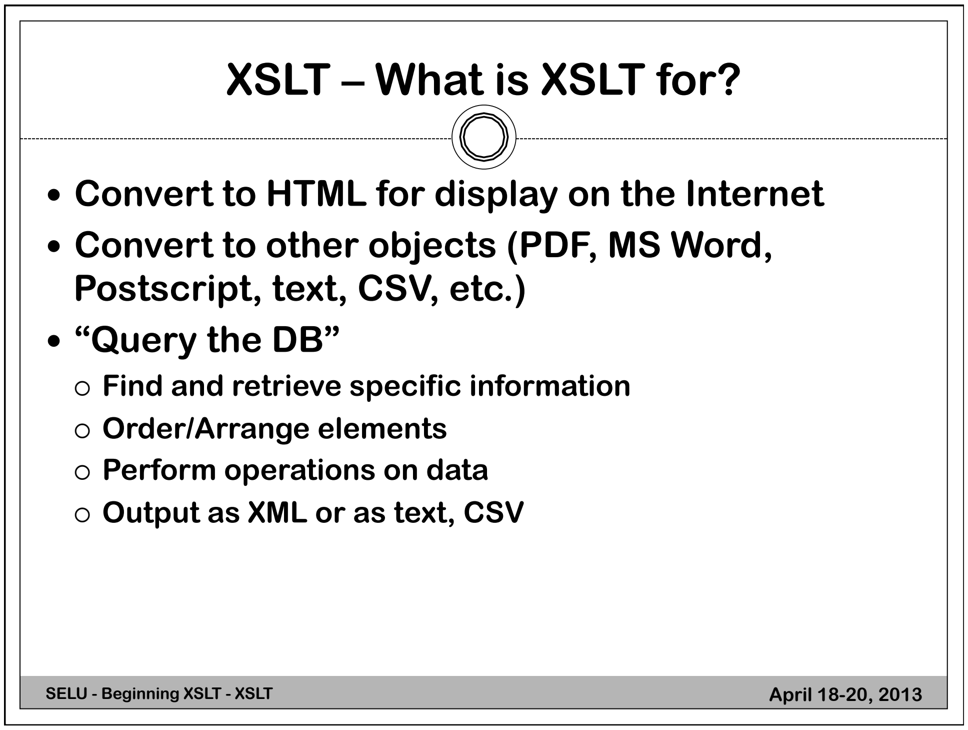


What is XSLT?

- eXtensible Stylesheet Language Transformations
  - *Stylesheet:* Does not alter an XML doc, but creates a new one
  - *Language:* XSLT is a declarative language—describes *__what__* should be accomplished as opposed to *__how__*
    - Template rules define what should be done to nodes matching an XPath-like pattern
    - How is left up to the processor
  - *Transformations:* turns an XML document into something else – text, HTML, XML, etc.

Performs the same functions as a stylesheet but is a programmatic language for transforming XML documents.

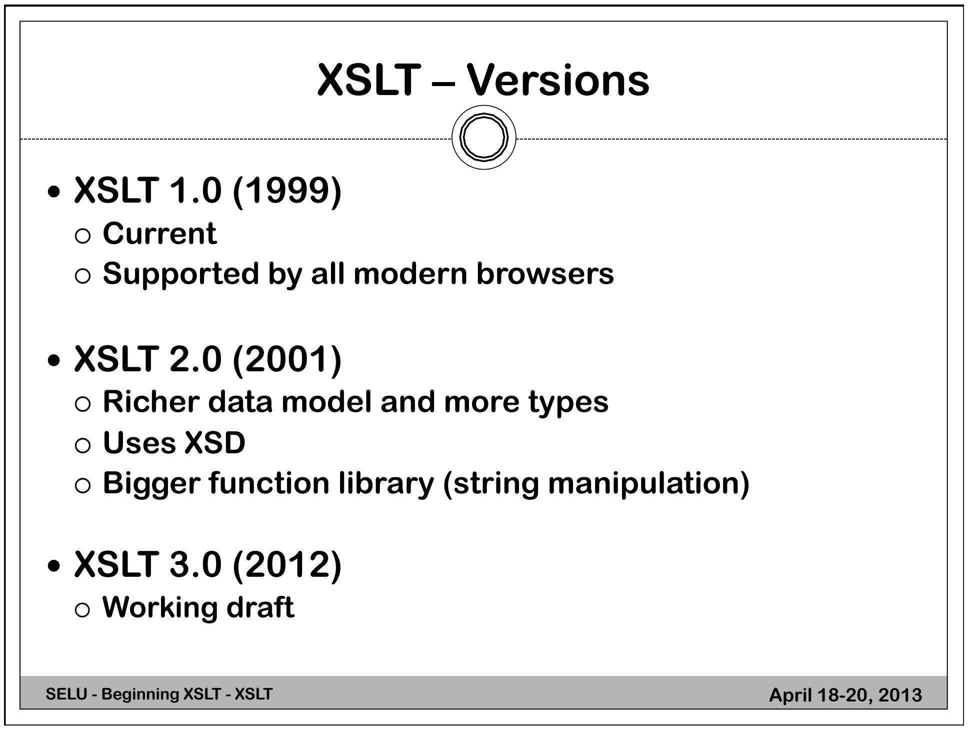
SELU - Beginning XSLT - XSLT April 18-20, 2013



XSLT – What is XSLT for?

- Convert to HTML for display on the Internet
- Convert to other objects (PDF, MS Word, Postscript, text, CSV, etc.)
- “Query the DB”
  - Find and retrieve specific information
  - Order/Arrange elements
  - Perform operations on data
  - Output as XML or as text, CSV

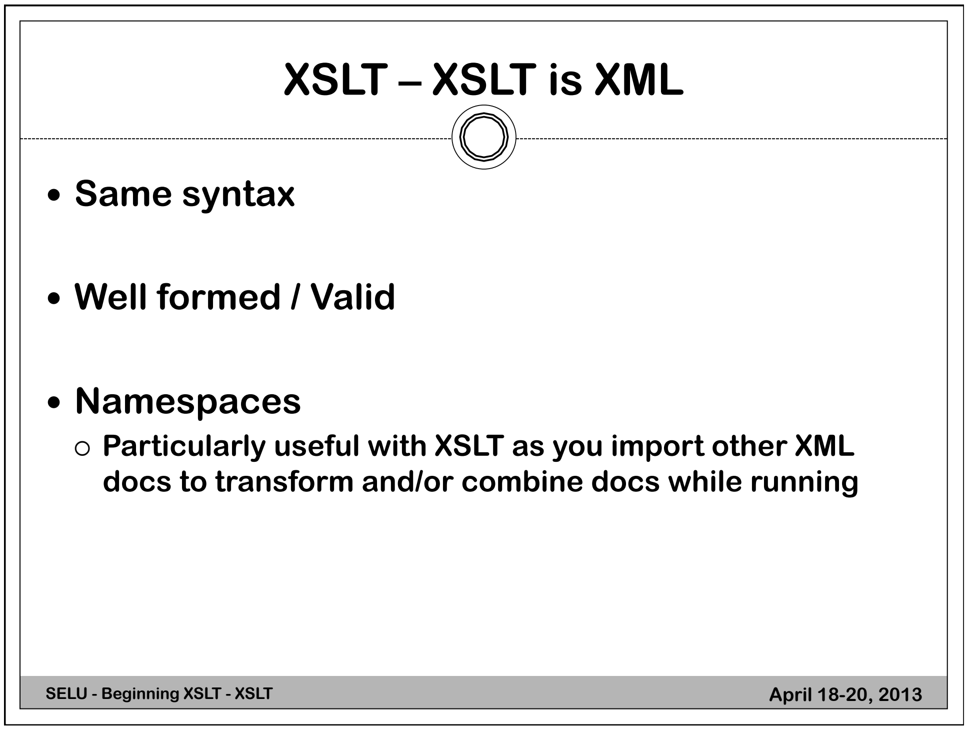
SELU - Beginning XSLT - XSLT April 18-20, 2013



XSLT – Versions

- XSLT 1.0 (1999)
  - Current
  - Supported by all modern browsers
- XSLT 2.0 (2001)
  - Richer data model and more types
  - Uses XSD
  - Bigger function library (string manipulation)
- XSLT 3.0 (2012)
  - Working draft

SELU - Beginning XSLT - XSLT April 18-20, 2013



XSLT – XSLT is XML

- Same syntax
- Well formed / Valid
- Namespaces
  - Particularly useful with XSLT as you import other XML docs to transform and/or combine docs while running

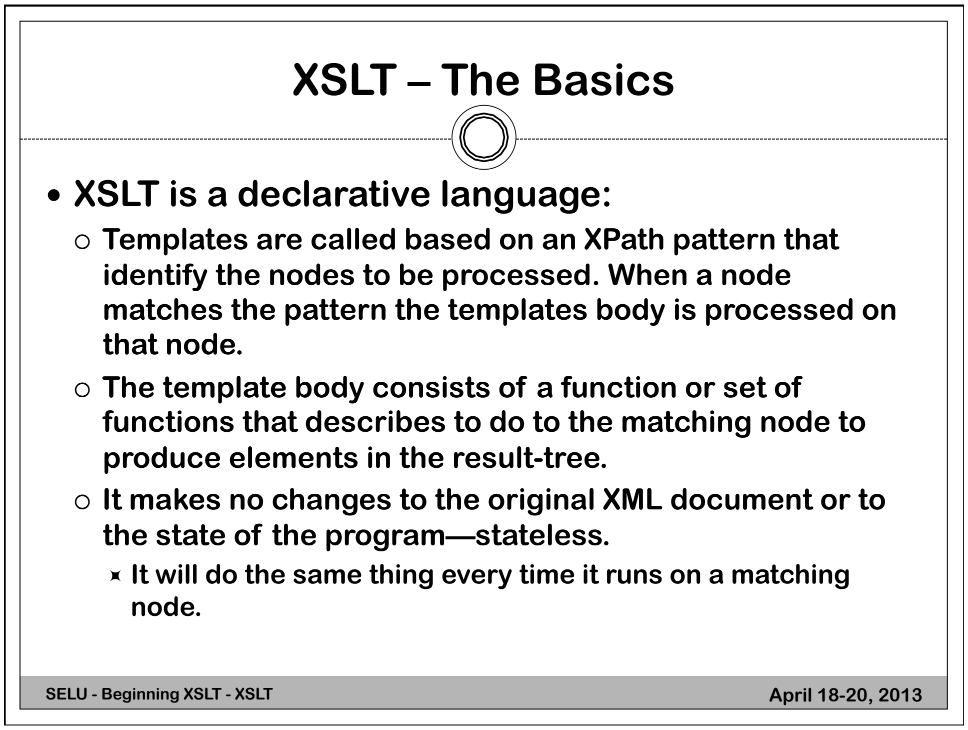
SELU - Beginning XSLT - XSLT April 18-20, 2013

Example – the Identity template

```
<?xml version="1.0" encoding="UTF-8"?>
<xsl:stylesheet
    xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
    version="2.0">

    <xsl:template match="@*|node()">
        <xsl:copy>
            <xsl:apply-templates select="@*|node()"/>
        </xsl:copy>
    </xsl:template>

</xsl:stylesheet>
```

XSLT – Applying XSLT to XML

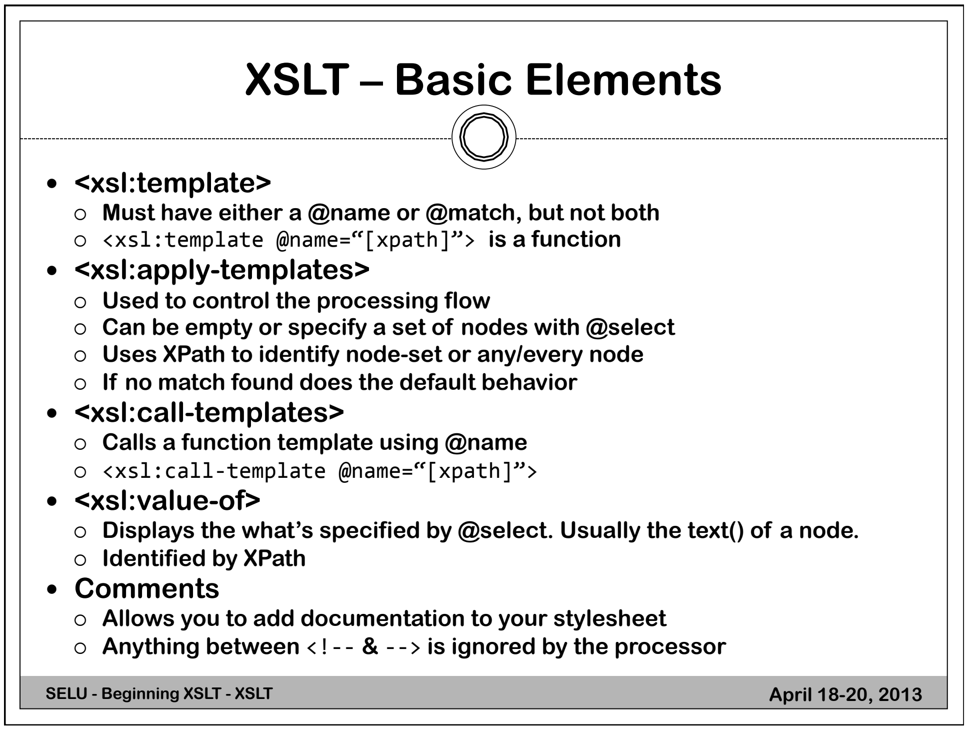
- **Oxygen**
  - **Input/Output**
- **Processor(s)**
  - **V1: Saxon6.5.5, Xalan, Xsltproc**
  - **V2: Saxon-xE 9.4.0.4 (Home, Professional, Extended)**
- **Result-tree**
  - **As the XSLT does its work it creates a result-tree. The output of the transformation is the combination of all result-trees produced.**
  - **In our case 1 tree.**
  - **Can’t be accessed by user during processing.**



XSLT – The Basics

- **XSLT is a declarative language:**
  - Templates are called based on an XPath pattern that identify the nodes to be processed. When a node matches the pattern the templates body is processed on that node.
  - The template body consists of a function or set of functions that describes to do to the matching node to produce elements in the result-tree.
  - It makes no changes to the original XML document or to the state of the program—stateless.
    - It will do the same thing every time it runs on a matching node.

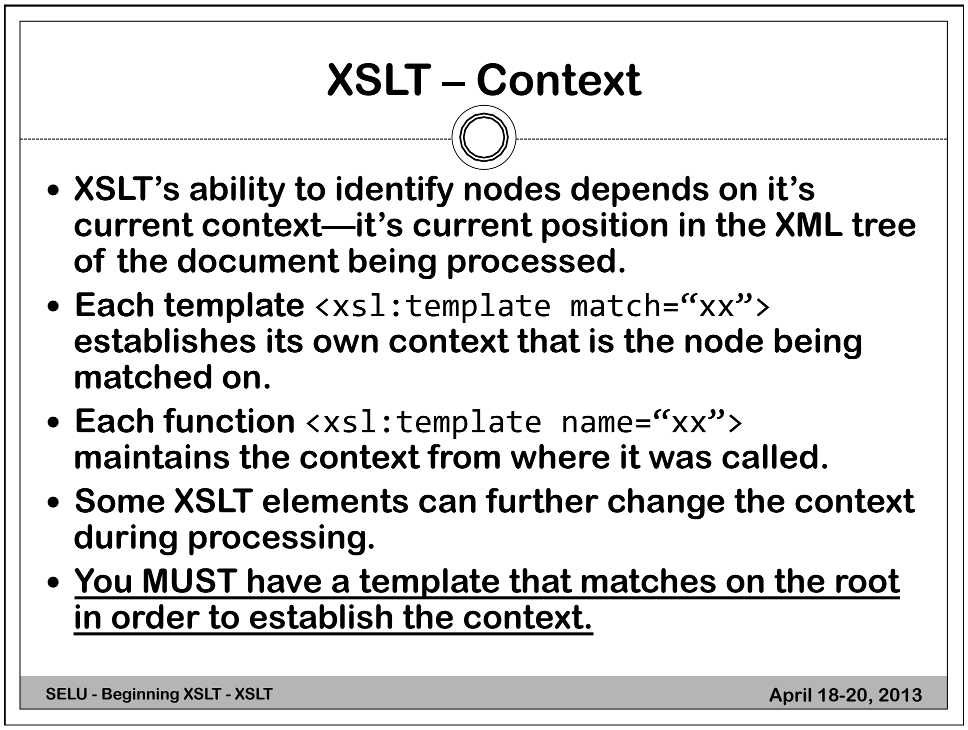
SELU - Beginning XSLT - XSLT April 18-20, 2013



XSLT – Basic Elements

- <xsl:template>
  - Must have either a @name or @match, but not both
  - `<xsl:template @name="[xpath]">` is a function
- <xsl:apply-templates>
  - Used to control the processing flow
  - Can be empty or specify a set of nodes with @select
  - Uses XPath to identify node-set or any/every node
  - If no match found does the default behavior
- <xsl:call-templates>
  - Calls a function template using @name
  - `<xsl:call-template @name="[xpath]">`
- <xsl:value-of>
  - Displays the what's specified by @select. Usually the text() of a node.
  - Identified by XPath
- Comments
  - Allows you to add documentation to your stylesheet
  - Anything between `<!--` & `-->` is ignored by the processor

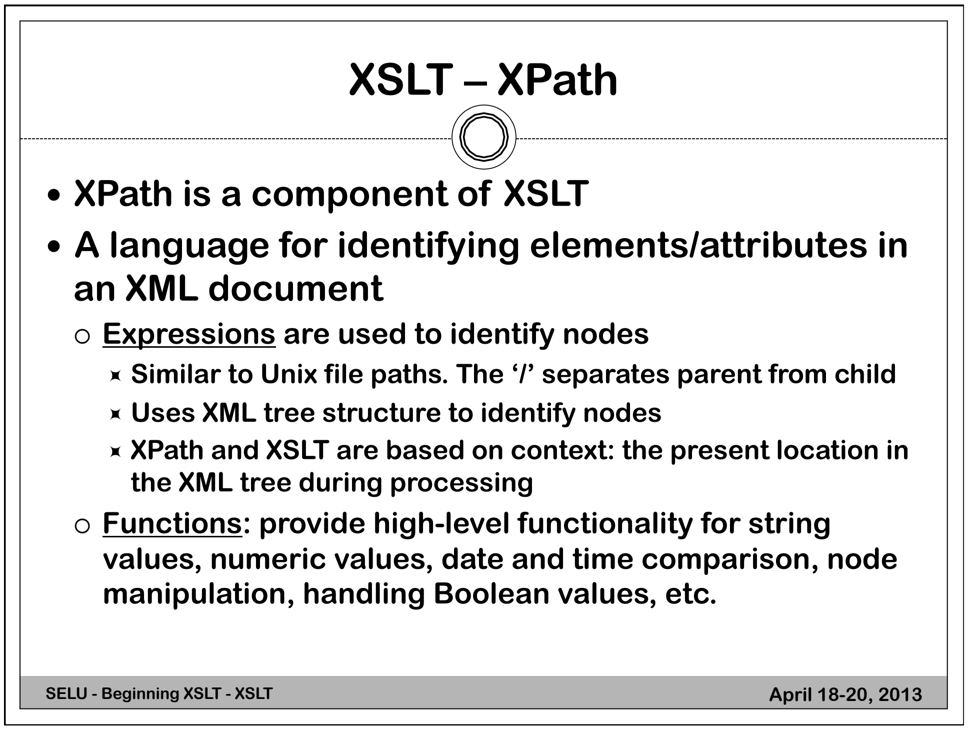
SELU - Beginning XSLT - XSLT April 18-20, 2013



XSLT – Context

- XSLT's ability to identify nodes depends on it's current context—it's current position in the XML tree of the document being processed.
- Each template `<xsl:template match="xx">` establishes its own context that is the node being matched on.
- Each function `<xsl:template name="xx">` maintains the context from where it was called.
- Some XSLT elements can further change the context during processing.
- <u>You MUST have a template that matches on the root in order to establish the context.</u>

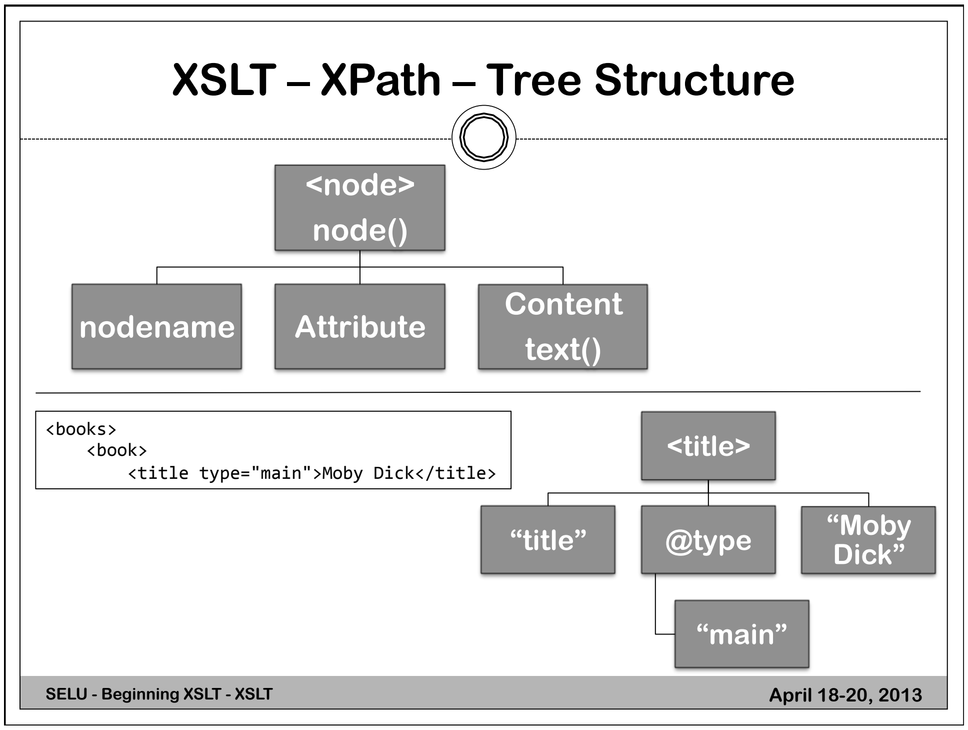
SELU - Beginning XSLT - XSLT April 18-20, 2013



XSLT – XPath

- XPath is a component of XSLT
- A language for identifying elements/attributes in an XML document
  - <u>Expressions</u> are used to identify nodes
    - Similar to Unix file paths. The '/' separates parent from child
    - Uses XML tree structure to identify nodes
    - XPath and XSLT are based on context: the present location in the XML tree during processing
  - <u>Functions</u>: provide high-level functionality for string values, numeric values, date and time comparison, node manipulation, handling Boolean values, etc.

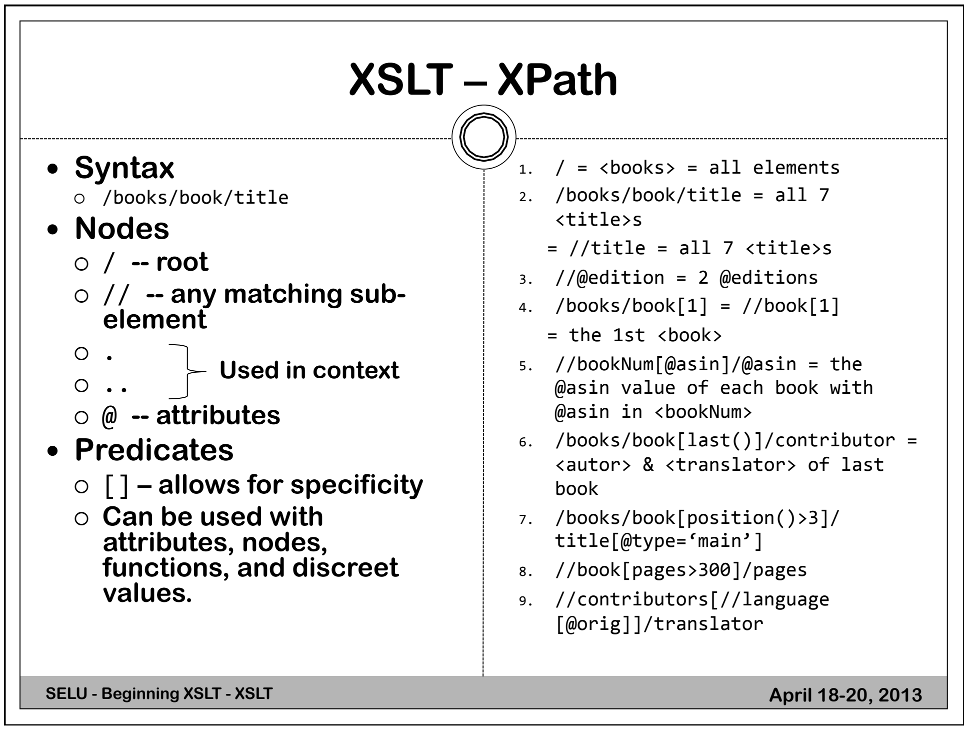
SELU - Beginning XSLT - XSLT April 18-20, 2013



XSLT – XPath – Tree Structure

```
<books>
    <book>
        <title type="main">Moby Dick</title>
```

SELU - Beginning XSLT - XSLT April 18-20, 2013

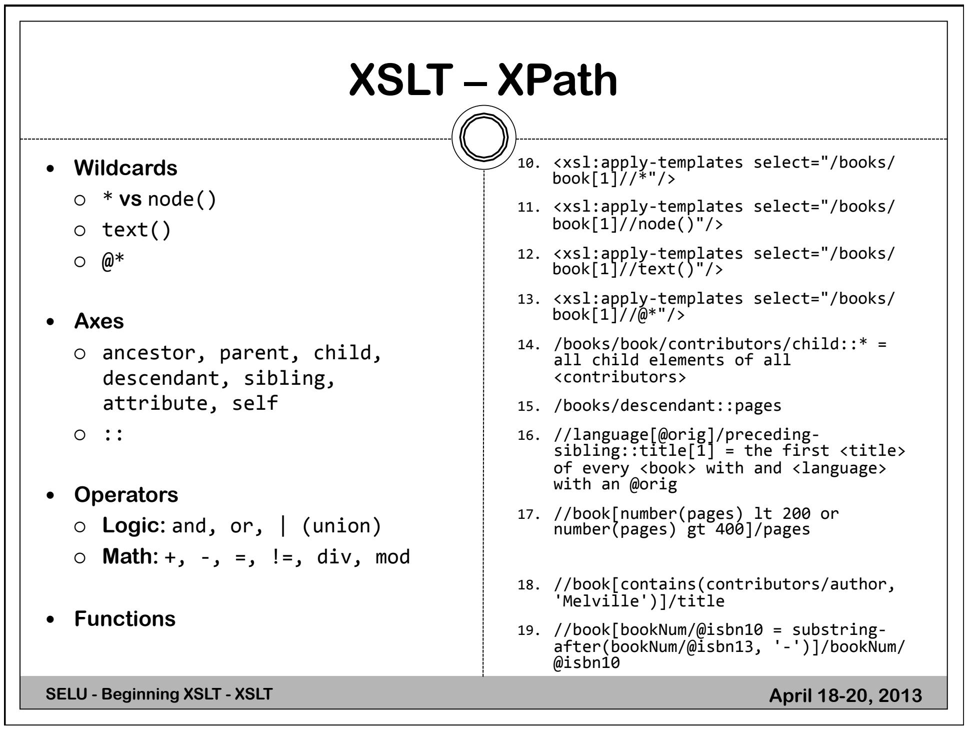


XSLT – XPath

- Syntax
  - /books/book/title
- Nodes
  - / -- root
  - // -- any matching sub-element
  - .
  - ..
    - Used in context
  - @ -- attributes
- Predicates
  - [ ] – allows for specificity
  - Can be used with attributes, nodes, functions, and discreet values.

1. / = <books> = all elements
2. /books/book/title = all 7 <title>s
   = //title = all 7 <title>s
3. //@edition = 2 @editions
4. /books/book[1] = //book[1]
   = the 1st <book>
5. //bookNum[@asin]/@asin = the @asin value of each book with @asin in <bookNum>
6. /books/book[last()]/contributor = <autor> & <translator> of last book
7. /books/book[position()>3]/title[@type='main']
8. //book[pages>300]/pages
9. //contributors[//language[@orig]]/translator

SELU - Beginning XSLT - XSLT April 18-20, 2013

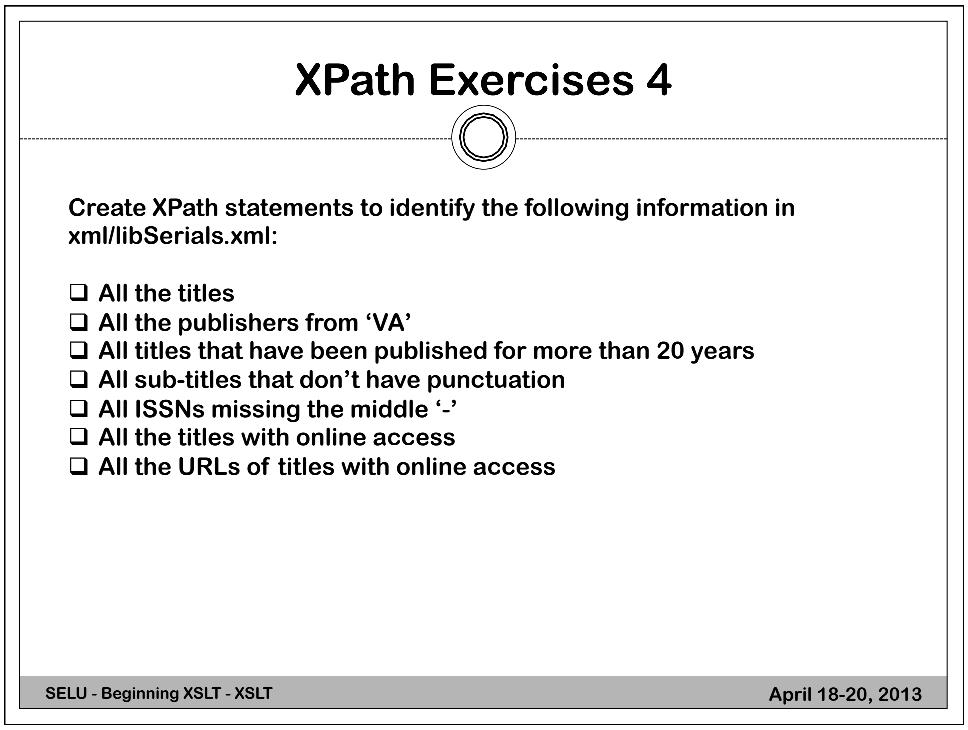


XSLT – XPath

- **Wildcards**
  - `*` **vs** `node()`
  - `text()`
  - `@*`
- **Axes**
  - `ancestor, parent, child, descendant, sibling, attribute, self`
  - `::`
- **Operators**
  - **Logic:** `and, or, | (union)`
  - **Math:** `+, -, =, !=, div, mod`
- **Functions**

10. `<xsl:apply-templates select="/books/book[1]//*"/>`
11. `<xsl:apply-templates select="/books/book[1]//node()"/>`
12. `<xsl:apply-templates select="/books/book[1]//text()"/>`
13. `<xsl:apply-templates select="/books/book[1]//@*"/>`
14. `/books/book/contributors/child::* = all child elements of all <contributors>`
15. `/books/descendant::pages`
16. `//language[@orig]/preceding-sibling::title[1] = the first <title> of every <book> with and <language> with an @orig`
17. `//book[number(pages) lt 200 or number(pages) gt 400]/pages`
18. `//book[contains(contributors/author, 'Melville')]/title`
19. `//book[bookNum/@isbn10 = substring-after(bookNum/@isbn13, '-')]/bookNum/@isbn10`

SELU - Beginning XSLT - XSLT April 18-20, 2013

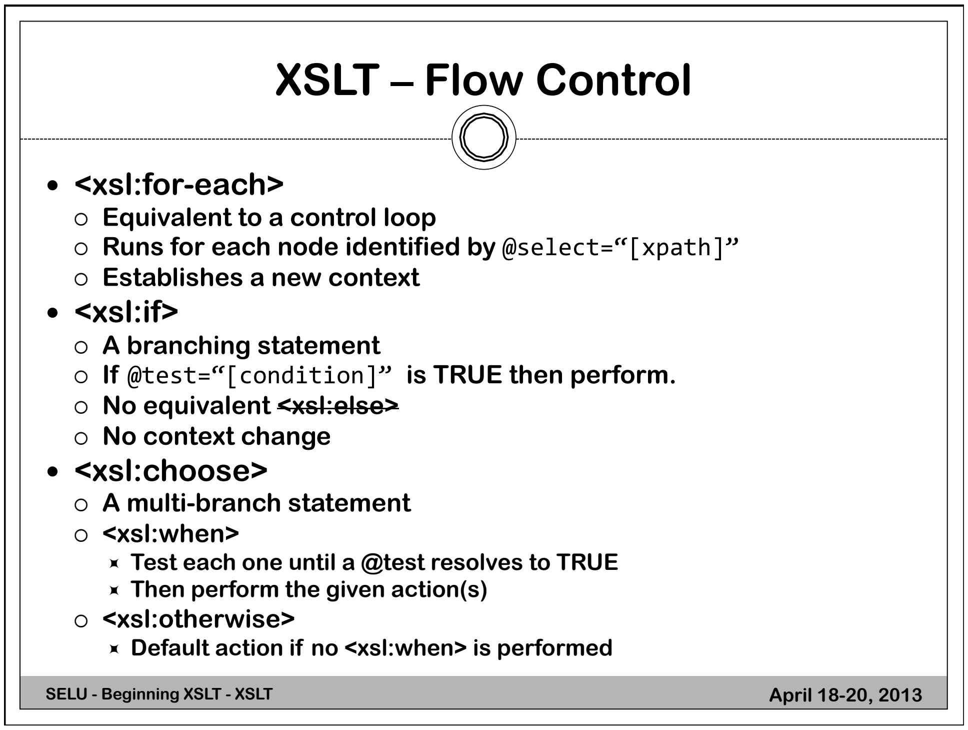


XPath Exercises 4

Create XPath statements to identify the following information in xml/libSerials.xml:

- ❑ **All the titles**
- ❑ **All the publishers from 'VA'**
- ❑ **All titles that have been published for more than 20 years**
- ❑ **All sub-titles that don't have punctuation**
- ❑ **All ISSNs missing the middle '-'**
- ❑ **All the titles with online access**
- ❑ **All the URLs of titles with online access**

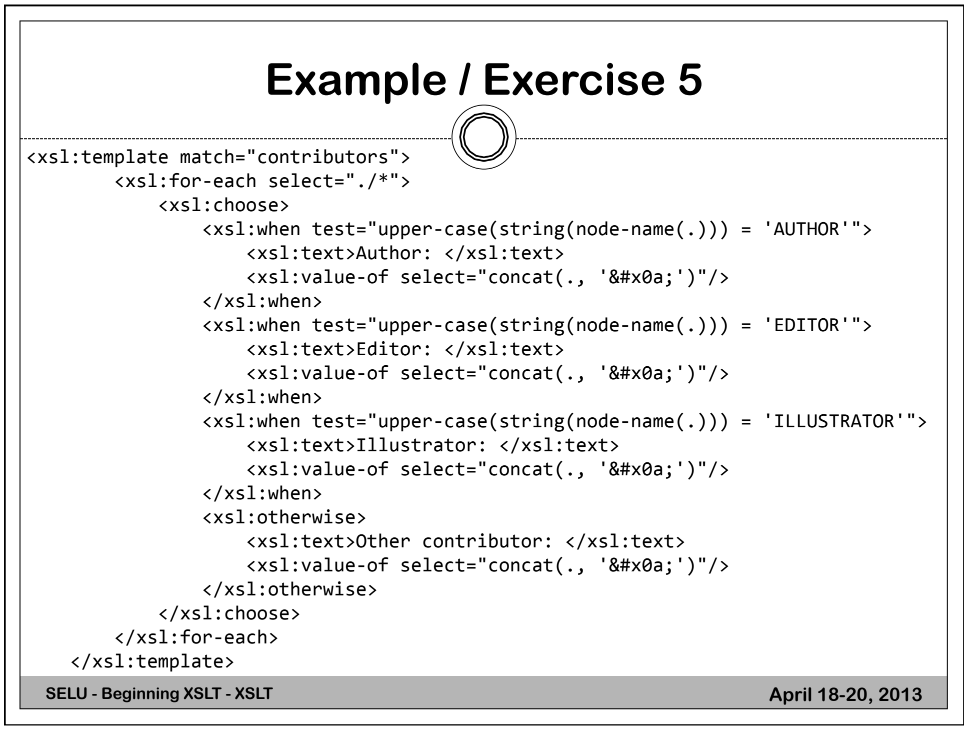
SELU - Beginning XSLT - XSLT April 18-20, 2013



XSLT – Flow Control

- <xsl:for-each>
  - Equivalent to a control loop
  - Runs for each node identified by `@select="[xpath]"`
  - Establishes a new context
- <xsl:if>
  - A branching statement
  - If `@test="[condition]"` is TRUE then perform.
  - No equivalent ~~<xsl:else>~~
  - No context change
- <xsl:choose>
  - A multi-branch statement
  - <xsl:when>
    - Test each one until a @test resolves to TRUE
    - Then perform the given action(s)
  - <xsl:otherwise>
    - Default action if no <xsl:when> is performed

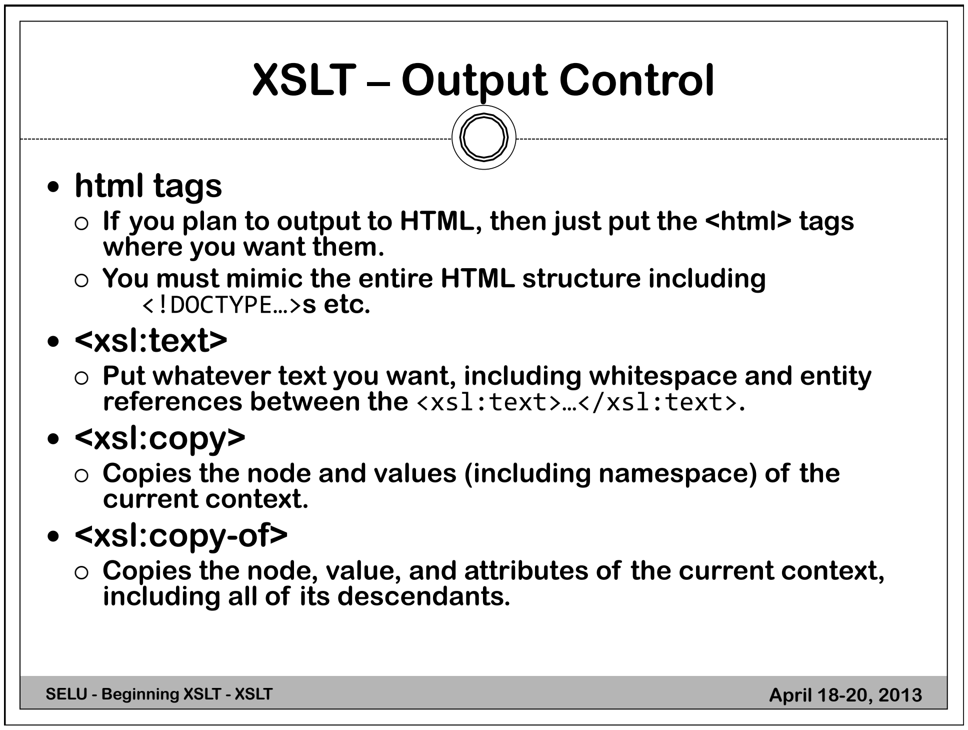
SELU - Beginning XSLT - XSLT April 18-20, 2013



Example / Exercise 5

```
<xsl:template match="contributors">
        <xsl:for-each select="./*">
            <xsl:choose>
                <xsl:when test="upper-case(string(node-name(.))) = 'AUTHOR'">
                    <xsl:text>Author: </xsl:text>
                    <xsl:value-of select="concat(., '&#x0a;')"/>
                </xsl:when>
                <xsl:when test="upper-case(string(node-name(.))) = 'EDITOR'">
                    <xsl:text>Editor: </xsl:text>
                    <xsl:value-of select="concat(., '&#x0a;')"/>
                </xsl:when>
                <xsl:when test="upper-case(string(node-name(.))) = 'ILLUSTRATOR'">
                    <xsl:text>Illustrator: </xsl:text>
                    <xsl:value-of select="concat(., '&#x0a;')"/>
                </xsl:when>
                <xsl:otherwise>
                    <xsl:text>Other contributor: </xsl:text>
                    <xsl:value-of select="concat(., '&#x0a;')"/>
                </xsl:otherwise>
            </xsl:choose>
        </xsl:for-each>
    </xsl:template>
```

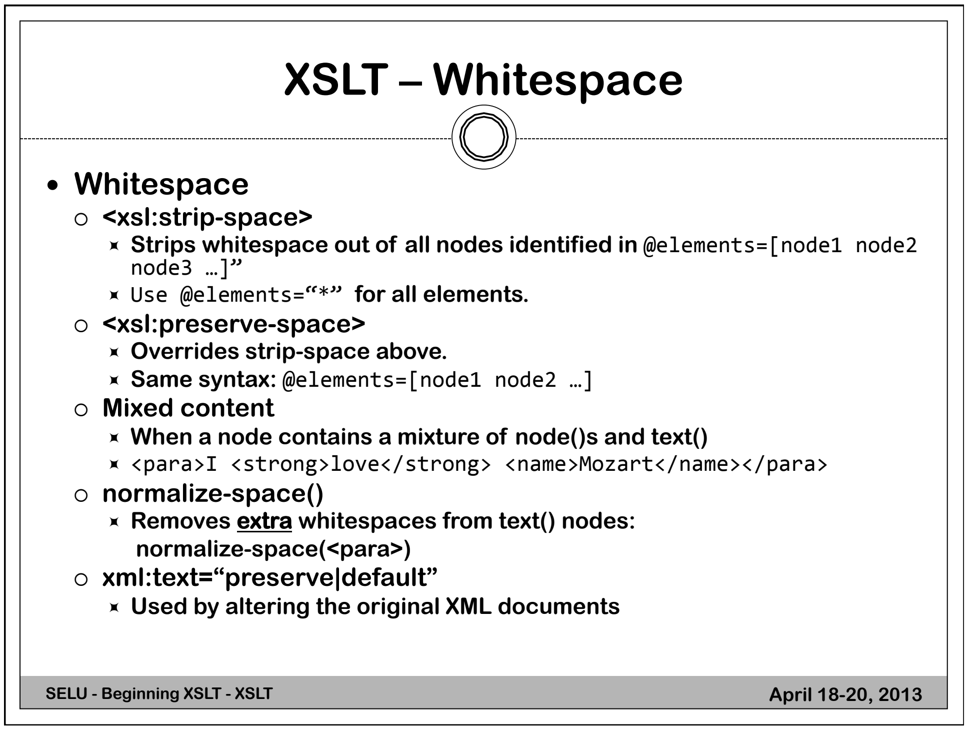
SELU - Beginning XSLT - XSLT April 18-20, 2013



XSLT – Output Control

- html tags
  - If you plan to output to HTML, then just put the <html> tags where you want them.
  - You must mimic the entire HTML structure including `<!DOCTYPE…>`s etc.
- <xsl:text>
  - Put whatever text you want, including whitespace and entity references between the `<xsl:text>…</xsl:text>`.
- <xsl:copy>
  - Copies the node and values (including namespace) of the current context.
- <xsl:copy-of>
  - Copies the node, value, and attributes of the current context, including all of its descendants.

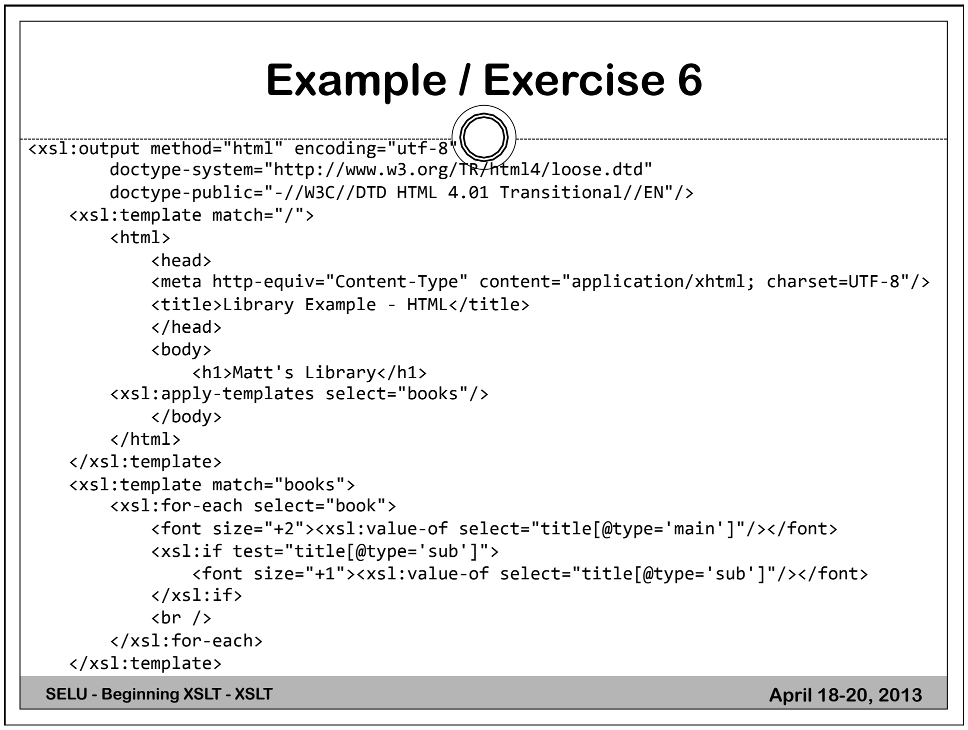
SELU - Beginning XSLT - XSLT April 18-20, 2013



XSLT – Whitespace

- Whitespace
  - <xsl:strip-space>
    - Strips whitespace out of all nodes identified in `@elements=[node1 node2 node3 …]`”
    - Use `@elements=“*”` for all elements.
  - <xsl:preserve-space>
    - Overrides strip-space above.
    - Same syntax: `@elements=[node1 node2 …]`
  - Mixed content
    - When a node contains a mixture of node()s and text()
    - `<para>I <strong>love</strong> <name>Mozart</name></para>`
  - normalize-space()
    - Removes <u>extra</u> whitespaces from text() nodes: normalize-space(<para>)
  - xml:text=“preserve|default”
    - Used by altering the original XML documents

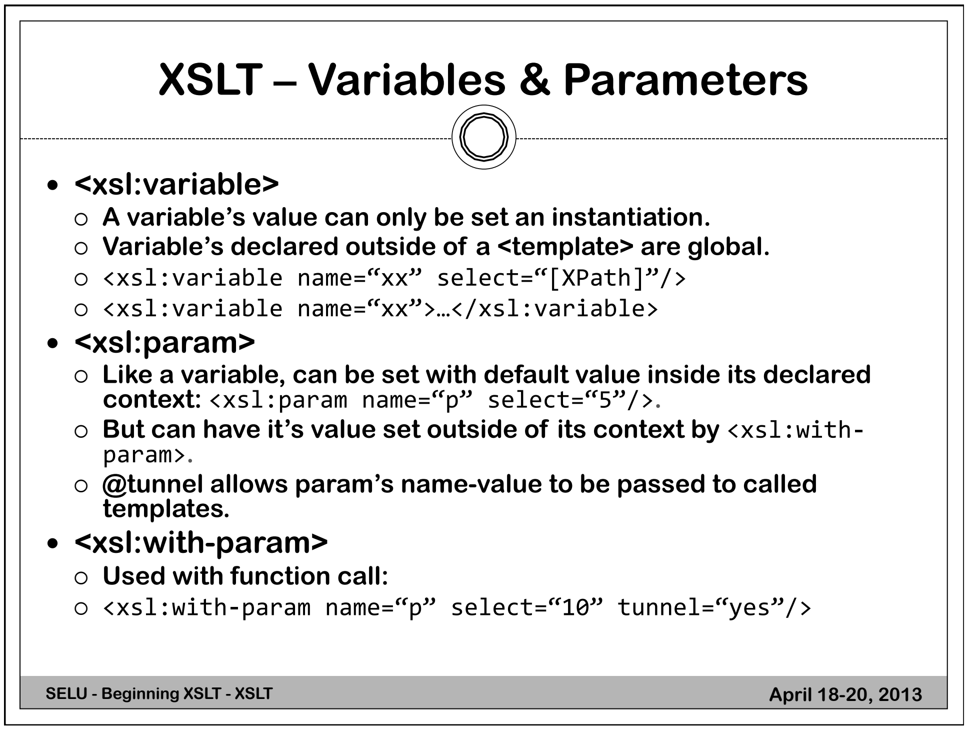
SELU - Beginning XSLT - XSLT April 18-20, 2013



Example / Exercise 6

```
<xsl:output method="html" encoding="utf-8"
        doctype-system="http://www.w3.org/TR/html4/loose.dtd"
        doctype-public="-//W3C//DTD HTML 4.01 Transitional//EN"/>
    <xsl:template match="/">
        <html>
            <head>
            <meta http-equiv="Content-Type" content="application/xhtml; charset=UTF-8"/>
            <title>Library Example - HTML</title>
            </head>
            <body>
                <h1>Matt's Library</h1>
        <xsl:apply-templates select="books"/>
            </body>
        </html>
    </xsl:template>
    <xsl:template match="books">
        <xsl:for-each select="book">
            <font size="+2"><xsl:value-of select="title[@type='main']"/></font>
            <xsl:if test="title[@type='sub']">
                <font size="+1"><xsl:value-of select="title[@type='sub']"/></font>
            </xsl:if>
            <br />
        </xsl:for-each>
    </xsl:template>
```

SELU - Beginning XSLT - XSLT April 18-20, 2013



XSLT – Variables & Parameters

- **<xsl:variable>**
  - **A variable's value can only be set an instantiation.**
  - **Variable's declared outside of a <template> are global.**
  - `<xsl:variable name="xx" select="[XPath]"/>`
  - `<xsl:variable name="xx">…</xsl:variable>`
- **<xsl:param>**
  - **Like a variable, can be set with default value inside its declared context:** `<xsl:param name="p" select="5"/>`.
  - **But can have it's value set outside of its context by** `<xsl:with-param>`.
  - **@tunnel allows param's name-value to be passed to called templates.**
- **<xsl:with-param>**
  - **Used with function call:**
  - `<xsl:with-param name="p" select="10" tunnel="yes"/>`

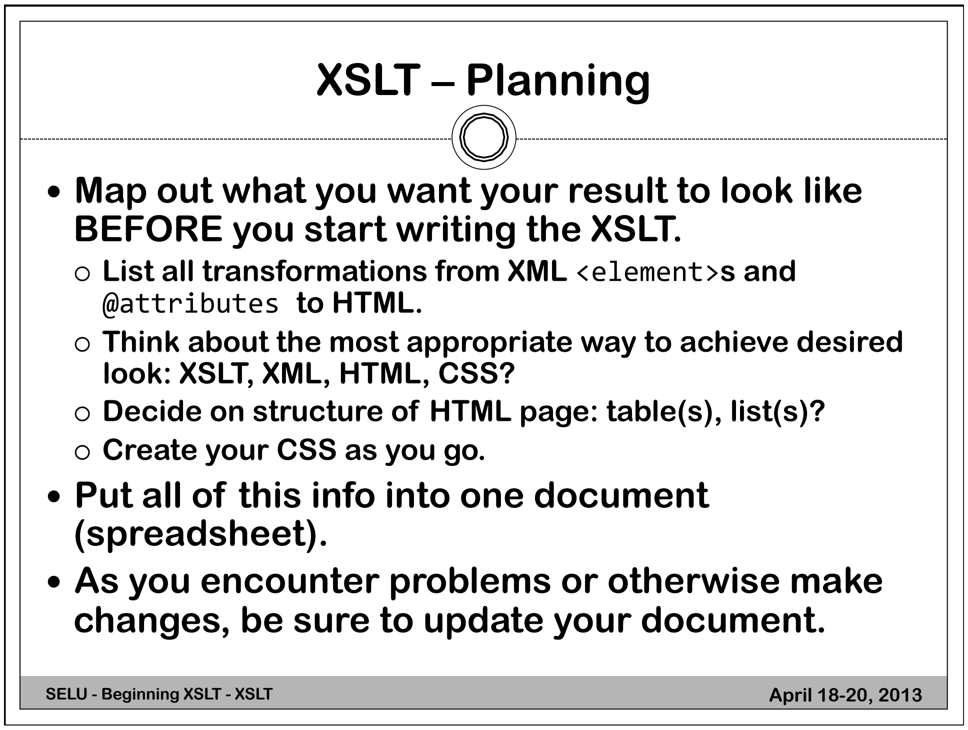
SELU - Beginning XSLT - XSLT April 18-20, 2013

XSLT – Sort

- **Built-in function to sort a tree based on criteria specified by attributes:**
  - **@select: identifies the nodes used to do the sorting**
  - **@order:**
    - **“ascending” (default) or “descending”**
  - **@order-case:**
    - **Sort firt on upper or lower-case letters**
    - **“upper-first” (default) or “lower-first”**
  - **@data-type:**
    - **The type of the data to be sorted**
    - **“text” (default), “number”, or “qname” (node names)**

Example / Exercise 7

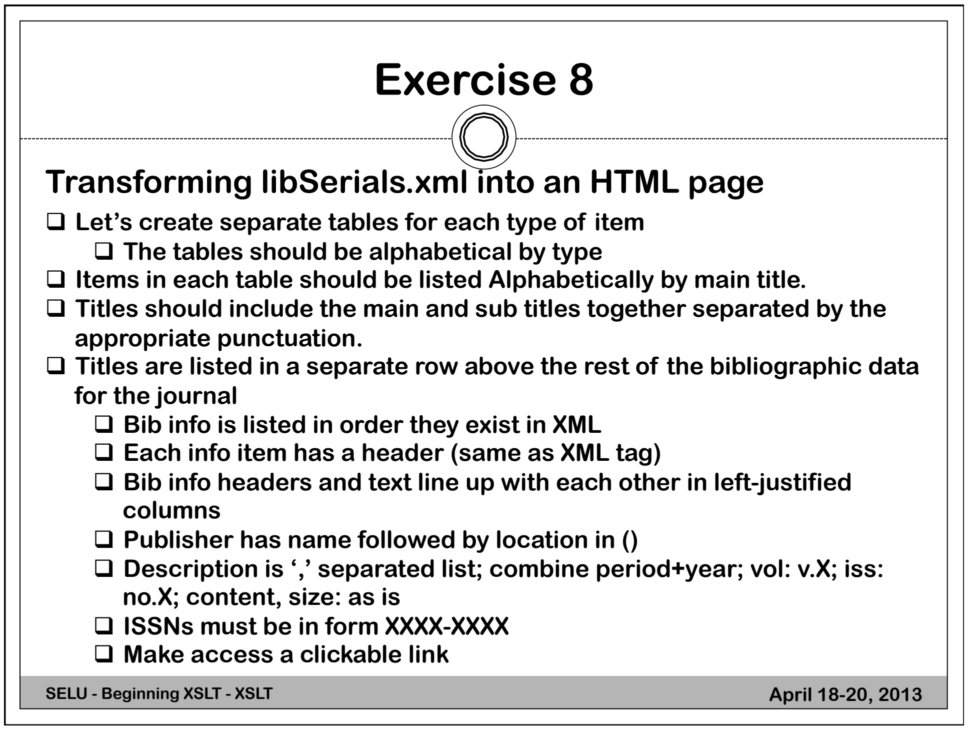
```
<xsl:template match="/">
       <xsl:apply-templates select="/books"/>
   </xsl:template>

   <xsl:template match="books">
       <html><body><table>
           <tr><td>Title</td><td>Bib</td></tr>
       <xsl:for-each select="book">
           <xsl:sort select="contributors/author"/>
           <xsl:sort select="title[@type='main']"/>
           <xsl:call-template name="outputBooks">
               <xsl:with-param name="sortedBks" select="."/>
           </xsl:call-template>
       </xsl:for-each>
       </table></body></html>
   </xsl:template>

   <xsl:template name="outputBooks">
       <xsl:param name="sortedBks" select="//books"/>
       <tr><td><xsl:value-of select="title"/></td>
           <td><xsl:value-of select="title/following-sibling::*"/></td></tr>
   </xsl:template>
```



XSLT – Planning

- Map out what you want your result to look like BEFORE you start writing the XSLT.
  - List all transformations from XML `<element>`s and `@attributes` to HTML.
  - Think about the most appropriate way to achieve desired look: XSLT, XML, HTML, CSS?
  - Decide on structure of HTML page: table(s), list(s)?
  - Create your CSS as you go.
- Put all of this info into one document (spreadsheet).
- As you encounter problems or otherwise make changes, be sure to update your document.

SELU - Beginning XSLT - XSLT April 18-20, 2013

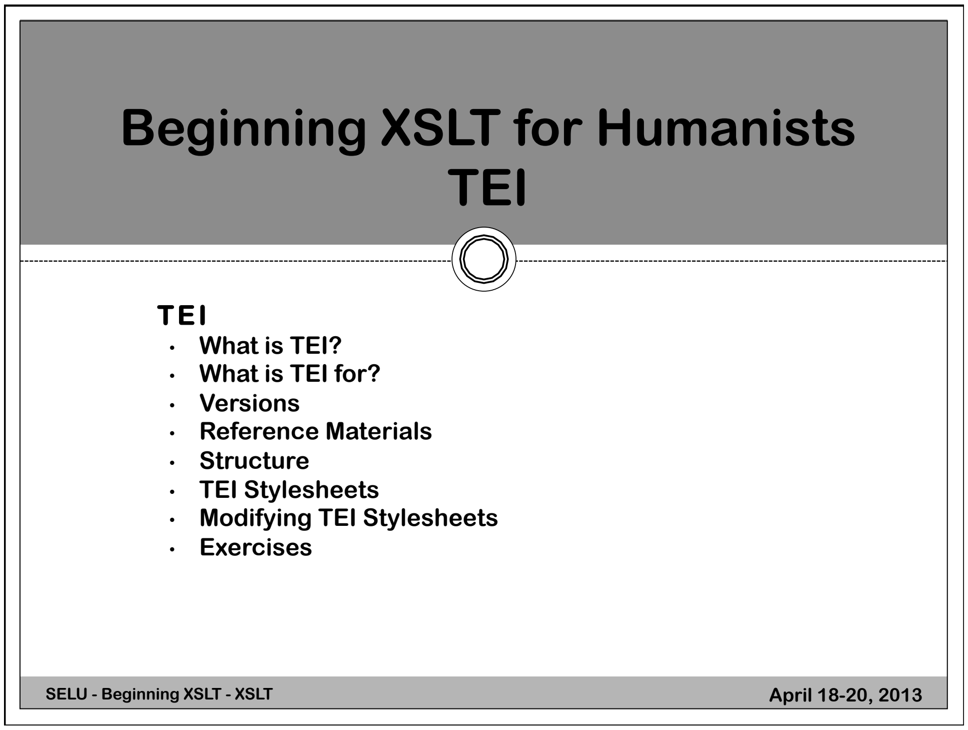


Exercise 8

Transforming libSerials.xml into an HTML page

- ❑ Let's create separate tables for each type of item
  - ❑ The tables should be alphabetical by type
- ❑ Items in each table should be listed Alphabetically by main title.
- ❑ Titles should include the main and sub titles together separated by the appropriate punctuation.
- ❑ Titles are listed in a separate row above the rest of the bibliographic data for the journal
  - ❑ Bib info is listed in order they exist in XML
  - ❑ Each info item has a header (same as XML tag)
  - ❑ Bib info headers and text line up with each other in left-justified columns
  - ❑ Publisher has name followed by location in ()
  - ❑ Description is ',' separated list; combine period+year; vol: v.X; iss: no.X; content, size: as is
  - ❑ ISSNs must be in form XXXX-XXXX
  - ❑ Make access a clickable link

SELU - Beginning XSLT - XSLT April 18-20, 2013

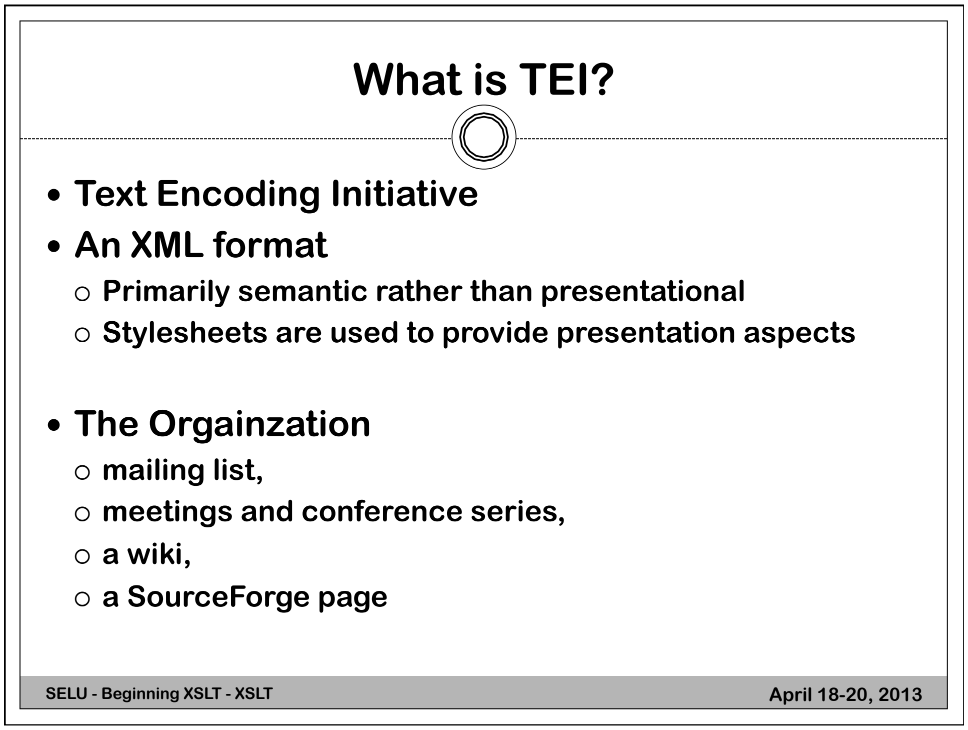


Beginning XSLT for Humanists TEI

TEI

- What is TEI?
- What is TEI for?
- Versions
- Reference Materials
- Structure
- TEI Stylesheets
- Modifying TEI Stylesheets
- Exercises

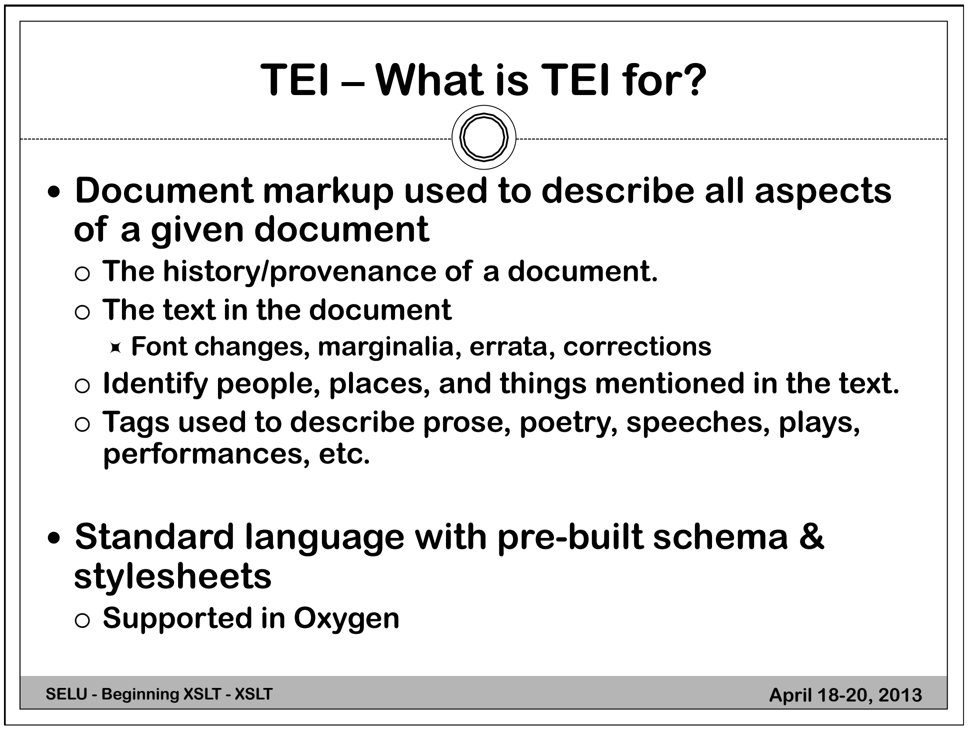
SELU - Beginning XSLT - XSLT April 18-20, 2013



What is TEI?

- Text Encoding Initiative
- An XML format
  - Primarily semantic rather than presentational
  - Stylesheets are used to provide presentation aspects
- The Orgainzation
  - mailing list,
  - meetings and conference series,
  - a wiki,
  - a SourceForge page

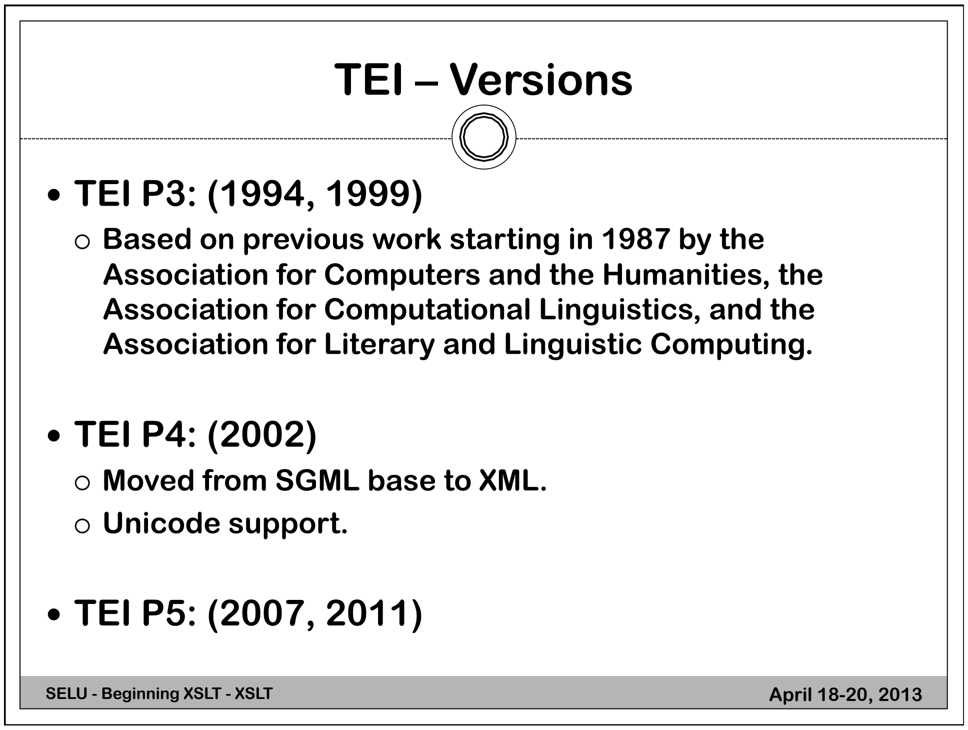
SELU - Beginning XSLT - XSLT April 18-20, 2013



TEI – What is TEI for?

- Document markup used to describe all aspects of a given document
  - The history/provenance of a document.
  - The text in the document
    - Font changes, marginalia, errata, corrections
  - Identify people, places, and things mentioned in the text.
  - Tags used to describe prose, poetry, speeches, plays, performances, etc.
- Standard language with pre-built schema & stylesheets
  - Supported in Oxygen

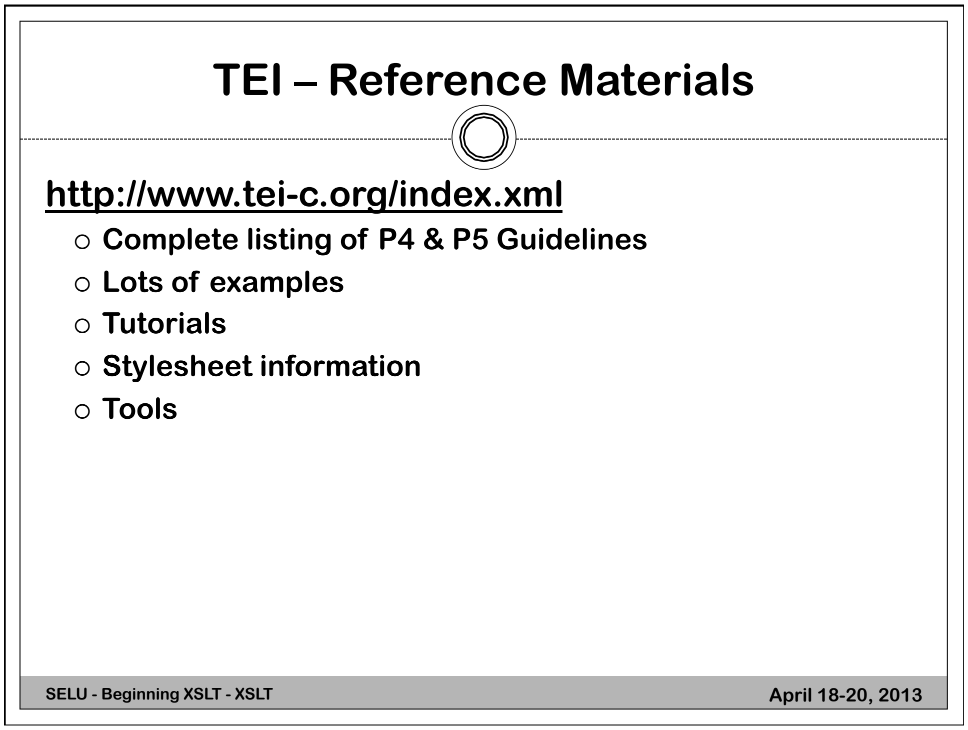
SELU - Beginning XSLT - XSLT April 18-20, 2013



TEI – Versions

- TEI P3: (1994, 1999)
  - Based on previous work starting in 1987 by the Association for Computers and the Humanities, the Association for Computational Linguistics, and the Association for Literary and Linguistic Computing.
- TEI P4: (2002)
  - Moved from SGML base to XML.
  - Unicode support.
- TEI P5: (2007, 2011)

SELU - Beginning XSLT - XSLT April 18-20, 2013

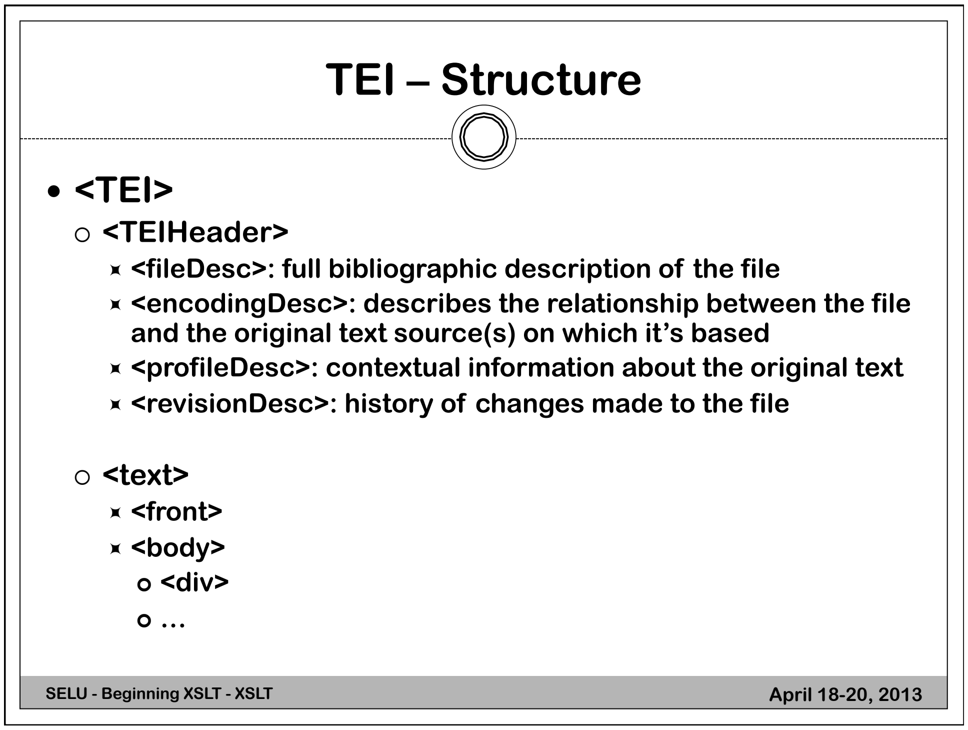


TEI – Reference Materials

http://www.tei-c.org/index.xml

- Complete listing of P4 & P5 Guidelines
- Lots of examples
- Tutorials
- Stylesheet information
- Tools

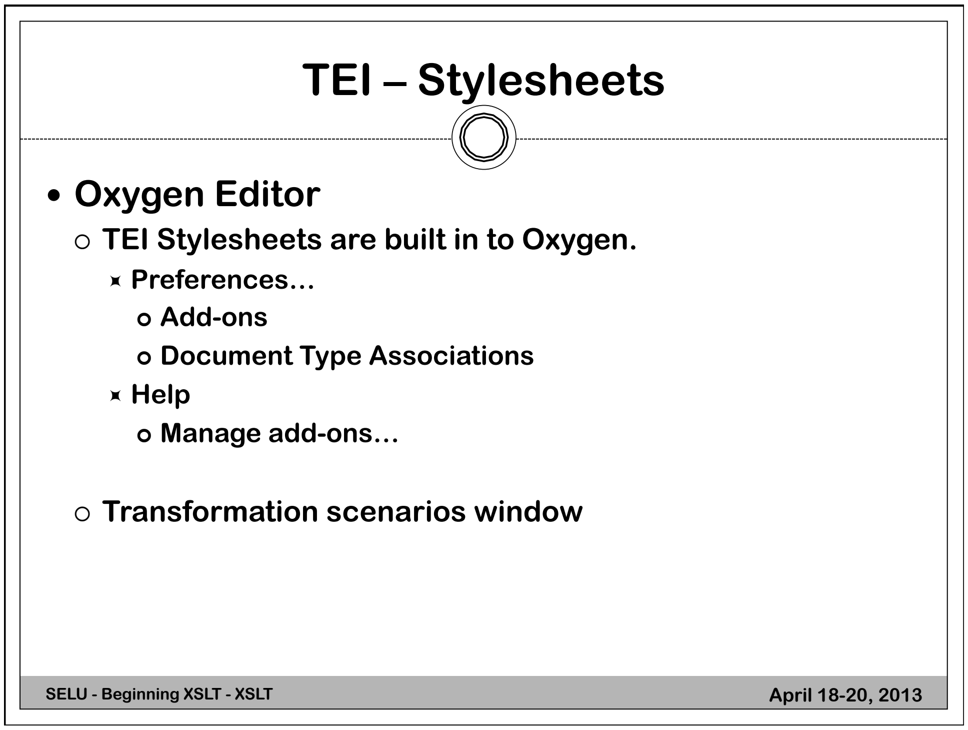
SELU - Beginning XSLT - XSLT April 18-20, 2013



TEI – Structure

- <TEI>
  - <TEIHeader>
    - <fileDesc>: full bibliographic description of the file
    - <encodingDesc>: describes the relationship between the file and the original text source(s) on which it's based
    - <profileDesc>: contextual information about the original text
    - <revisionDesc>: history of changes made to the file
  - <text>
    - <front>
    - <body>
      - <div>
      - …

SELU - Beginning XSLT - XSLT April 18-20, 2013

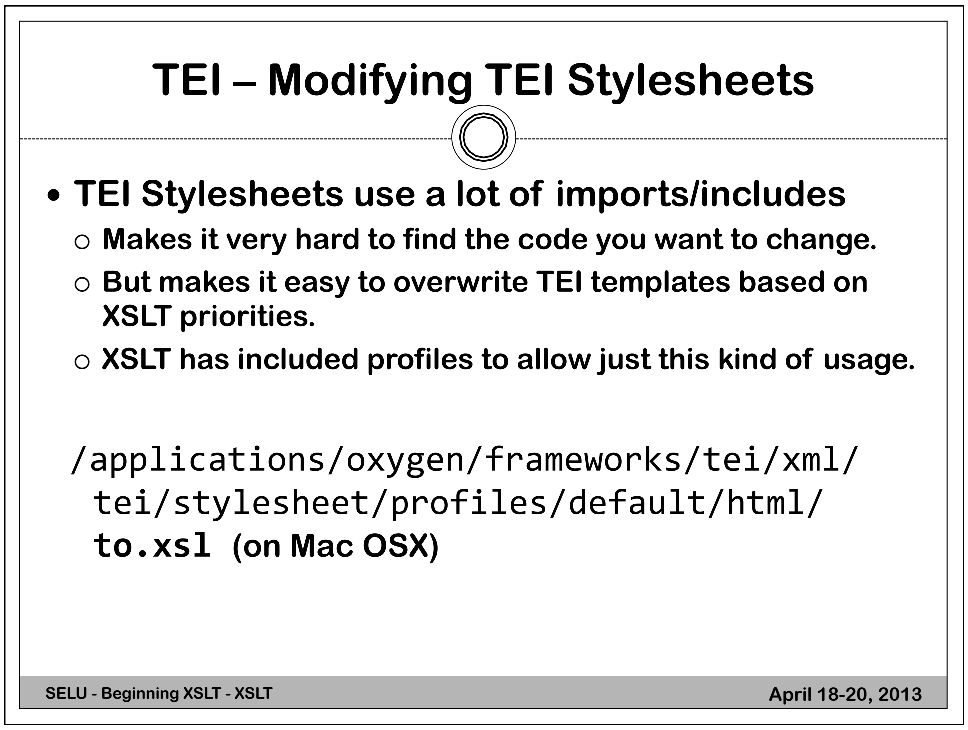


TEI – Stylesheets

- Oxygen Editor
  - TEI Stylesheets are built in to Oxygen.
    - Preferences…
      - Add-ons
      - Document Type Associations
    - Help
      - Manage add-ons…

  - Transformation scenarios window

SELU - Beginning XSLT - XSLT April 18-20, 2013



TEI – Modifying TEI Stylesheets

- TEI Stylesheets use a lot of imports/includes
  - Makes it very hard to find the code you want to change.
  - But makes it easy to overwrite TEI templates based on XSLT priorities.
  - XSLT has included profiles to allow just this kind of usage.

`/applications/oxygen/frameworks/tei/xml/tei/stylesheet/profiles/default/html/`**`to.xsl`** (on Mac OSX)

SELU - Beginning XSLT - XSLT April 18-20, 2013

Exercises

Feedback and Help

RDF

- What is RDF?
- What is RDF for?
- Syntax
- Types
- Vocabularies