

An Introduction to Extensible Stylesheet Language (XSL)

Simon Mahony

From an original document by Susan Hockey

This document is part of a collection of presentations and exercises on XML. For full details of this and the rest of the collection see the cover sheet at:

Stylesheets

- **CSS** for appearance and layout
 - Cascading Style Sheets
- **XSL-FO** for printer output
 - Extensible Stylesheet Language: Formatting Objects
- **XSLT** for Transformation – manipulation
 - Extensible Stylesheet Language: Transformations

XSL: Formatting Objects

- Used to describe XML documents for display (usually printed)
- Describe how page should look
- 56 Elements- enough detail to publish books
- E.g. fo:footnote, fo:page-number, fo:table, fo:external-graphic

XSLT

- Powerful transformation language
- W3C standard
- Transforms one tree structure to another
- Used for formatting as with CSS

- XSLT files are XML files and therefore follow XML rules - ie declaration / well-formed

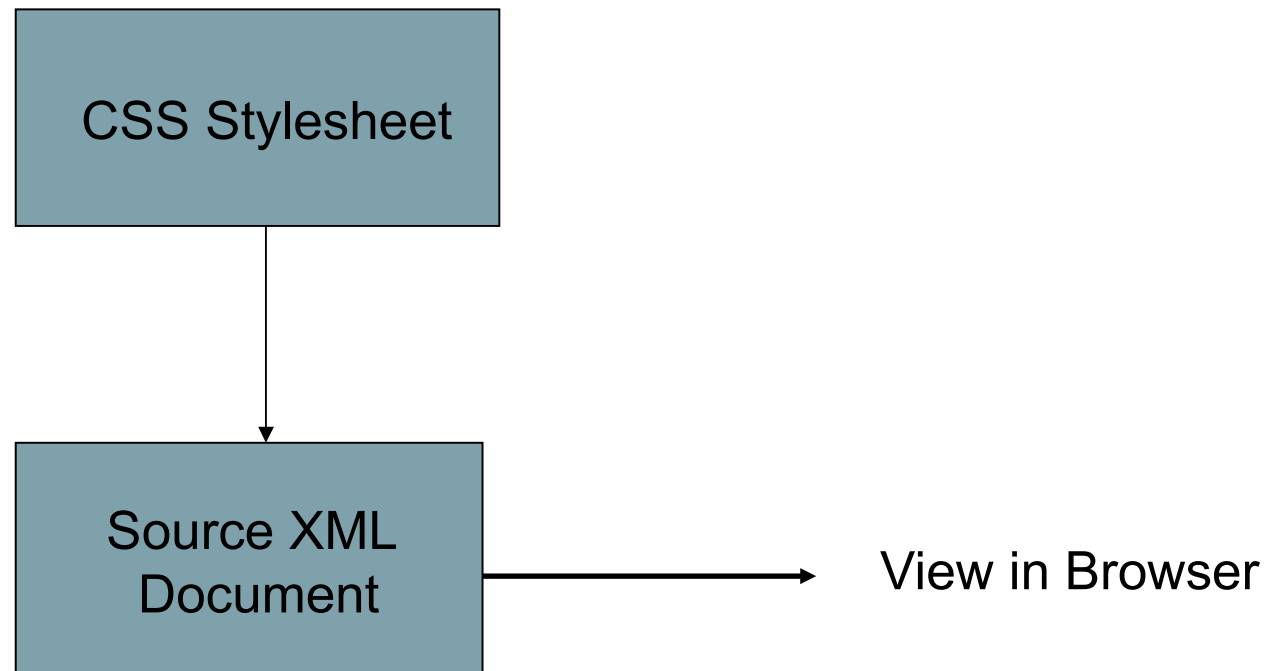
XSL and Display on the Web

- Transforms a source XML document to an output XHTML document for web display
- XSL "drills down" the XML tree and picks out elements and attributes to go into the XHTML document
- The XHTML can also have some CSS
- Not all elements need be transformed

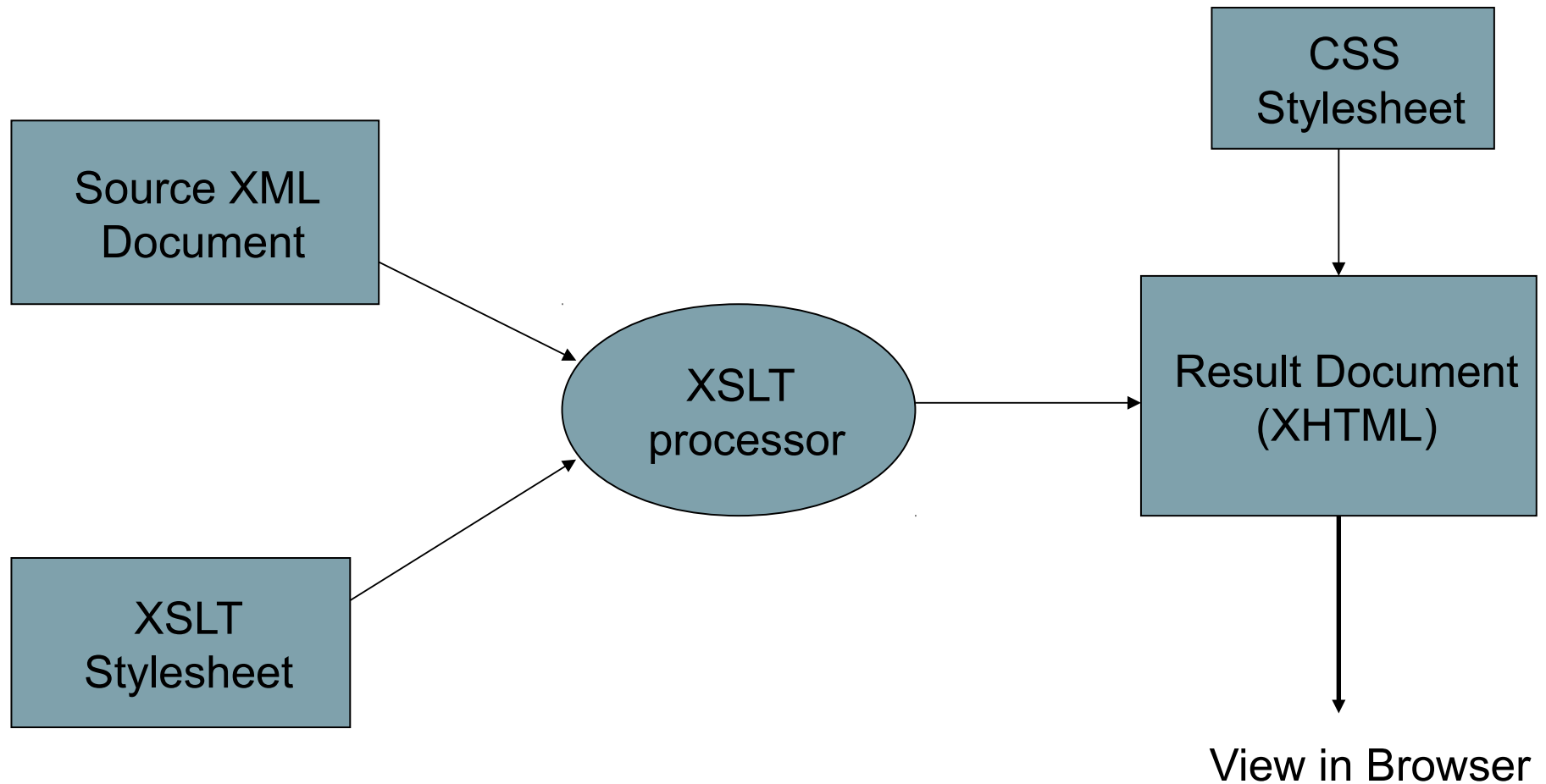
Publication Process

- Document analysis
- Create DTD (or other Schema)
- Create documents
- Well Formed and Validate against DTD/Schema
- Publish
 - Transform with XSLT Stylesheet
 - Style with CSS Stylesheet

CSS Styling



Transformation and Styling



Overview of Some XSLT Functions

- Present selected information for different readers
- Add elements for viewing, e.g. add address or logo
- Create new content from existing content, e.g. table of contents
- Re-order content
- Can also operate on attributes

XSLT Processors

- Incorporate Well-formedness check
- IE6 +, Firefox include CSS and XSLT processors
- Others are available
 - Saxon
 - Xalan
 - MSXML
- Transformation processing (Saxon) built in to Oxygen
 - Also debugging

XSL

- An XML Language
- Namespace xsl:
- Declaration

```
<?xml version="1.0" encoding="UTF-8"?>
```

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

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

XSLT Versions

- Version 1.0 November 1999
- Version 1.1 Working draft December 2000
 - Never formally released. Changes subsumed into ...
- Version 2.0 January 2007.
 - Still quite new
 - Backwards compatible with version 1.0
- Version 1.0 is the version used here

An XSL Stylesheet

- Well-formed XML document
 - Extension **.xsl**
- Describes and works on the document tree
- Top level element is **<xsl:stylesheet>**

How does it work?

- Selects part(s) of source document
- Defines 'templates' for selected parts to be transformed
- Creates new 'result document' containing transformed part(s) of source document
- Source document is XML
- Result document can be text, xml, xhtml,
- Our focus here is on xhtml for display in browser as part of a web site

Selecting from Source Document

- Uses **XPath** expression language
- Collects information from the source document by navigating through the document tree to **nodes** specified in the stylesheet
- Can also perform calculations – average, count, >, <, etc.

Templates

- Stylesheet is a set of templates

```
<xsl:stylesheet>  
  <xsl:template></xsl:template>  
  <xsl:template></xsl:template>  
</xsl:stylesheet>
```
- Each template consists of two parts:
 - a path (how to find the bit you want)
 - content (what you want to do with it)
- Use the path to find an element – using XPath language
- The content is what is done to the element as it is transferred to the result document

XPath (XML Path Language)

- Language used with XSLT
- Describes path to the element in the source tree to which the template applies
- Syntax is similar to file paths, starting from the root
 - `/article/section`
 - `<section>` as a child of `<article>`
 - `/section/title`
 - `<title>` as a child of `<section>`
- `/` points to the immediate children of the element

Xpath 'view' of your Document

- Specifying the right path depends on your start point.
- XSL and Xpath see your document like a tree.
- Each component of your document is called a 'node'
 - Element nodes
 - Attribute nodes
 - Text nodes
 - Comment nodes
 - Processing-instruction nodes

XSLT nodes

- **Node** is an individual piece of the XML document
- **Root**: the document itself - independent of any content
- **Element**: each element in the XML document
- **Attribute**: each attribute in the XML document
- **Text**: text content of an element
- **Comment**: comment in the XML doc
- **Processing instruction**: instructions in XML doc

Parent / child / sibling

- Every node is related in some way to another
- Relationship as in tree structure
- Root node has children
- To child nodes the root node is a parent node
- Children can have children or siblings
- Siblings have same parent node
- Descendants are a node's child nodes
- Ancestor nodes are a node's parent / grandparent / great-grandparent etc

The Document Root

- The XSL **root** is the whole document - **NOT** the root element of the document.
- '/' addresses the entire document including comments and processing instructions
- So to address <recipe> in XSL/Xpath use '/recipe'
- To address <author> use /recipe/author
- Similar to filesystem addressing

More Paths

- `//section`
`<section>` anywhere within the document
- `/article/*/title`
`<title>` when it is a grandchild of `<article>`

Attributes in Paths

book/@style

Finds the style attribute of **<book>**

XSL Template Content

- Content of the template gives the elements to insert in the resulting tree
- Convert to XHTML for display by a web browser
- Not all elements need to be transformed
- Add 'literal result elements' in the stylesheet
 - XHTML tags to give basic formatting

XSL Template Content

- To display the element `<title>` within `<section>` as a paragraph in italic, XSL must find instances of `<title>` within `<section>` and then convert `<title>` ... `</title>` to
`<p><i>` ... `</i></p>`
or to the appropriate CSS

Example

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

```
  <p><i>
```

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

```
</i></p>
```

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

`<xsl:value-of select=".">` takes the value (content) of the current element

Example

```
<section><title>Pride and Prejudice</title></section>
```

Is transformed to

```
<p><i>Pride and Prejudice</i></p>
```

[Uses `<i>` tag for simplicity]

XSL Selections

- Can include patterns
- Rather like database queries
- Using location paths to drill down the tree to the required element

Context Node

- Patterns require a context node from where search is begun
- Start with the document instance, not the outermost element which is inside the document instance

Building an XSL Stylesheet

- Define templates for each element (normally called nodes) to be processed
- Define template for the document instance
- Apply the templates wherever necessary

Repeating Information

- The template can deal with all instances of a particular element one by one using `<xsl:for-each>`
- Useful for documents with a repeating structure
- Will return each and every occurrence of the node
- `<xsl:value-of select="title">` finds an exact element with the content specified in the select attribute

XML

```
<poem>
  <title>Title of Poem</title>
  <author>Name of Author</author>
  <stanza>
    <line>line 1</line>
    <line>line 2 </line>
    <line>line 3</line>
    <line>line 4</line>
  </stanza>
  <stanza>
    <line>line 1</line>
    <line>line 2</line>
    <line>line 3</line>
    <line>line 4</line>
  </stanza>
</poem>
```

XHTML

```
<html>
<head>
  <title>Title of Poem by name of
  Author</title>
  </head>
<body>
  <h1>Title of Poem</h1>
  <p><i>Author of Poem</i></p>
  <div>line 1</div>
  <div>line 2</div>
  <div>line 3</div>
  <div>line 4</div>
  <br />
  <div>line 1</div>
  <div>line 2</div>
  <div>line 3</div>
  <div>line 4</div>
  <br />
</body>
</html>
```


From XML to XHTML

XML	XHTML
<pre> <poem> </poem> </pre>	<pre> <html> <head> </head> <body> </body> </html> </pre>

From XML to XHTML

XML	XHTML
<pre><title> </title></pre>	<pre><title> [combination of 2 elements + text inserted] </title></pre>
<pre><author> </author></pre>	<pre>AND <h1> </h1> <p><i></i></p></pre>

Create mandatory elements and add header title

```
<xsl:template match="poem">
  <html>
    <head>
      <title> <xsl:value-of select="//title"/>
      by <xsl:value-of select="//author"/> </title>
    </head>
    <body>
      <xsl:apply-templates/>
    </body>
  </html>
</xsl:template>
```

Add `<h1>title </h1>` and `<i>author </i>` in `<body>`

(note XPath now has changed – relative to position)

```
<xsl:template match="title">
  <h1>
    <xsl:apply-templates/>
  </h1>
</xsl:template>

<xsl:template match="author">
  <p>
    <i>
      <xsl:apply-templates/>
    </i>
  </p>
</xsl:template>
```

Example Document Structure

```
<catalog>
  <cd>
    <title>... </title>
    <artist>... </artist>
  </cd>
  <cd>
    <title>... </title>
    <artist>... </artist>
  </cd>
</catalog>
```

(From W3 Schools)

XML

```

<catalog>
  <cd>
    <title>      </title>
    <artist>    </artist>
  </cd>
  <cd>
    <title>      </title>
    <artist>    </artist>
  </cd>
  <cd>
    <title>      </title>
    <artist>    </artist>
  </cd>
  <cd>
    <title>      </title>
    <artist>    </artist>
  </cd>
</catalog>

```

XHTML

```

<html>
  <body>
    <h2>My CD Collection</h2>
    <table>
      <tr>
        <th>Title</th>
        <th>Artist</th>
      </tr>
      <tr>
        <td>      </td>
        <td>      </td>
      </tr>
    </table>
  </body>
</html>

```

Example 71.xsl

```
<xsl:for-each select="catalog/cd">
  <tr>
    <td> <xsl:value-of select="title"/> </td>
    <td> <xsl:value-of select="artist"/> </td>
  </tr>
</xsl:for-each>
```

For each `<cd>` outputs the content of each `<title>` and `<artist>` element in turn within table cells (`<td>`).

value-of starts at specified node and repeats action until another value specified.

Instance Document example71.xml

```
<catalog>  
  <cd>  
    <title>CD 1 title</title>  
    <artist>CD 1 artist</artist>  
  </cd>  
  <cd>  
    <title>CD 2 title </title>  
    <artist>CD 2 artist </artist>  
  </cd>  
</catalog>
```


Example XHTML Result document

```
<tr>  
  <td>CD 1 title</td>  
  <td>CD 1 artist</td>  
</tr>  
<tr>  
  <td>CD 2 title</td>  
  <td>CD 2 artist</td>  
</tr>
```

Example XSL for the same document

```
<xsl:for-each select="catalog/cd">  
  <p>  
    <xsl:value-of select="title"/>  
    is by  
    <xsl:value-of select="artist"/>  
  </p>  
</xsl:for-each>
```

(note the supplied text [**is by**] at transformation stage)

Example Output

`<p>CD 1 title is by CD artist 1</p>`

`<p>CD 2 title is by CD artist 2</p>`

Displaying XML Documents Using an XSL Stylesheet

- Create an XHTML shell as an output document
- Literal result elements
- Use templates to put content within the shell

Putting it all Together

- The XML document references the stylesheet in exactly the same way as for CSS.

```
<?xml-stylesheet type="text/xsl"  
  href="stylesheetfilename.xsl"?>
```

as the second line of the file after

```
<?xml version="1.0" encoding="UTF-8"?>
```

Putting it all Together

- The stylesheet file is a well-formed XML document and begins with

```
<?xml version="1.0" encoding="UTF-8"?>
```

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

It ends with

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

Some URLs for XSLT tutorials

Zvon XSLT Tutorial

<http://www.zvon.org/xxl/XSLTutorial/Books/Book1/index.html>

W3schools.com XSLT Tutorial

<http://www.w3schools.com/xsl/default.asp>