More XML Stylesheet Functions with 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:

OPEN EDUCATIONAL RESOURCES FOR THE DIGITAL HUMANITIES

More XML Stylesheet Functions with XSL

XSL stylesheet gives processing instructions

XSLT processor (in Oxygen) reads the XML document

Whenever it matches a node specified by the template, it applies the instruction contained in the template

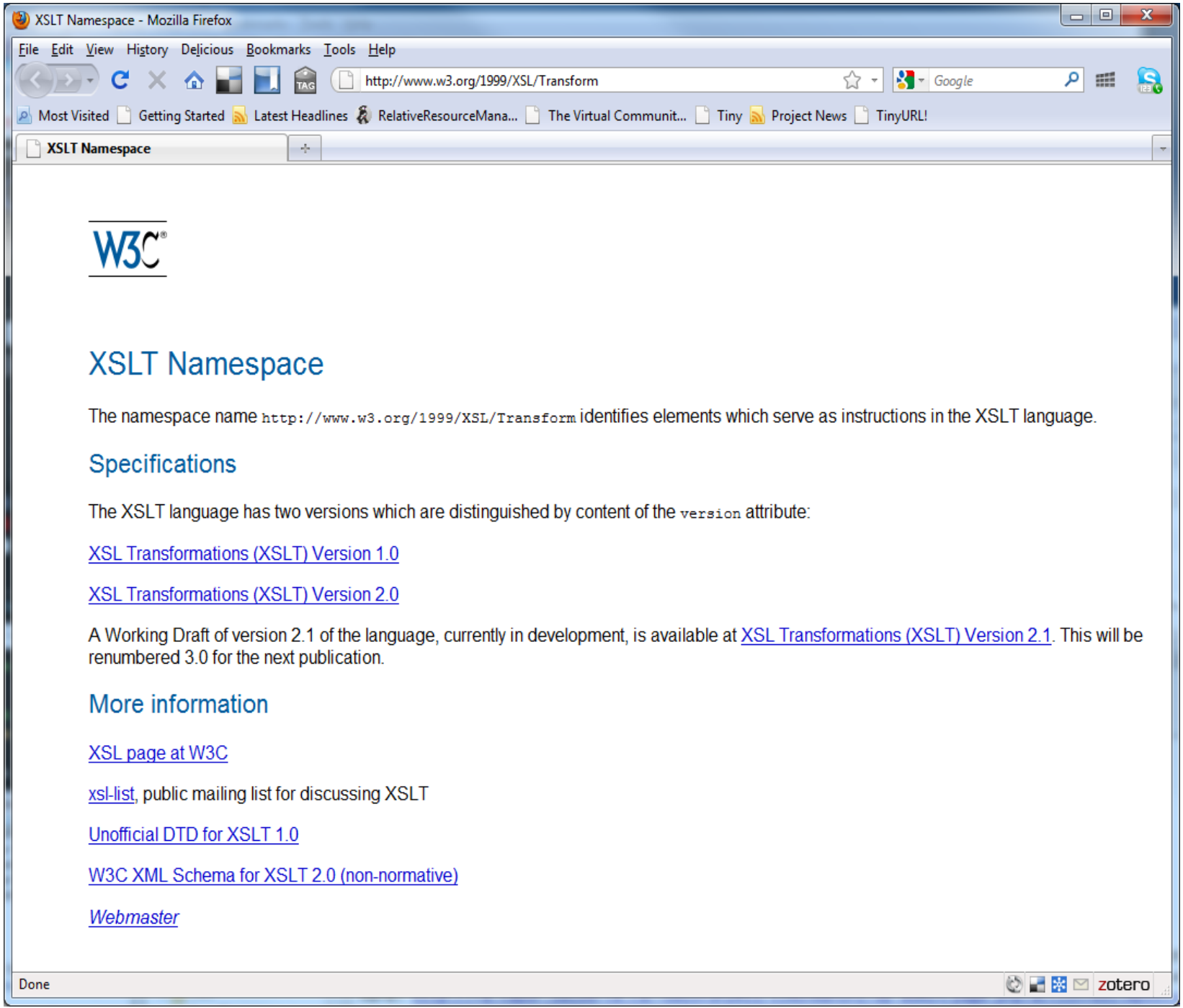
Namespace

- Let the parser know which DTD or schema vocab
- Permits sharing of vocabularies
- Eliminates possible confusion
- Value of attribute is URI
  - location of document about the namespace
  - or simply a unique identifier (as all URLs are unique)

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

ie all tags beginning xsl belong to the *namespace* that has that URI. So all come from the XSLT tagset as documented by w3 at that URL

Declaring the 'version' is mandatory


XSLT Namespace - Mozilla Firefox
File Edit View History Delicious Bookmarks Tools Help
http://www.w3.org/1999/XSL/Transform
Most Visited Getting Started Latest Headlines RelativeResourceMana... The Virtual Communit... Tiny Project News TinyURL!
XSLT Namespace
XSLT Namespace
The namespace name http://www.w3.org/1999/XSL/Transform identifies elements which serve as instructions in the XSLT language.
Specifications
The XSLT language has two versions which are distinguished by content of the version attribute:
XSL Transformations (XSLT) Version 1.0
XSL Transformations (XSLT) Version 2.0
A Working Draft of version 2.1 of the language, currently in development, is available at XSL Transformations (XSLT) Version 2.1. This will be renumbered 3.0 for the next publication.
More information
XSL page at W3C
xsl-list, public mailing list for discussing XSLT
Unofficial DTD for XSLT 1.0
W3C XML Schema for XSLT 2.0 (non-normative)
Webmaster
Done
zotero

Namespaces

Conventional but not mandatory to use xsl prefix

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

Here: all tags beginning trans belong to the *namespace* that has that URI.

```
<trans:template match=" … ">
   <p>
        <trans:apply-templates/>
   </p>
</trans:template>
```

Possible but rarely done. Why would you?

Also used in XHTML

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
    "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">

<head>
    <meta http-equiv="content-type" content="text/html; charset=UTF-8" />
    <meta http-equiv="Content-Language" content="en-gb" />
    <title>Introduction</title>

    <link rel="stylesheet" type="text/css" href="style.css" />
</head>

    <body>
        <p> EVERYTHING TO DISPLAY ON PAGE GOES HERE</p>
    </body>
</html>
```

XHTML namespace

The namespace name `http://www.w3.org/1999/xhtml` is intended for use in various specifications such as:

Recommendations:

- *XHTML™ 1.0: The Extensible HyperText Markup Language*
- *XHTML Modularization*
- *XHTML 1.1*
- *XHTML Basic*
- *XHTML Print*
- *XHTML+RDFa*

Working drafts:

- *HTML 5: A vocabulary and associated APIs for HTML and XHTML*

The charters of the following W3C Working Groups include work on HTML that may impact this namespace:

- Semantic Web Deployment Working Group, chartered July 2006 to work on RDFa
- WAI Protocols and Formats Working Group (PFWG), chartered Dec 2006 to work on Accessibility of Dynamic Web Content
- HTML Working Group, chartered March 2007
- XHTML2 Working Group, chartered March 2007

For more information about XML namespaces, please refer to *Namespaces in XML*.

Steven Pemberton, W3C HTML Activity Lead
Last edited: $Date: 2009/10/05 12:06:42 $

Specify the output <xsl:output>

- specify the type of document to be generated
- Instruct processor to generate XHTML

```
<xsl:output method="xhtml"
doctype-system="http://www.w3.org/TR/xhtml1/DTD/xhtml1-
                                                transitional.dtd"/>
```

Other possible values: xml / text / html

XPath

- Detailed technical description at w3/xpath (e.g.)
- Language to identify location of XML element
- Allows you to point to and select specific parts
  - ie XPath gives direction to the parser
  - enables drilling down to the node you want
- Nodes (various parts of the XML document)
  - root / element / attribute / text / comment / etc
- Context node: starting point

- Abbreviated and unabbreviated (full description)

W3C Recommendation

XML Path Language (XPath)
Version 1.0

W3C Recommendation 16 November 1999

This version:

http://www.w3.org/TR/1999/REC-xpath-19991116
(available in XML or HTML)

Latest version:

http://www.w3.org/TR/xpath

Previous versions:

http://www.w3.org/TR/1999/PR-xpath-19991008
http://www.w3.org/1999/08/WD-xpath-19990813
http://www.w3.org/1999/07/WD-xpath-19990709
http://www.w3.org/TR/1999/WD-xslt-19990421

Editors:

James Clark <jjc@jclark.com>
Steve DeRose (Inso Corp. and Brown University) <Steven_DeRose@Brown.edu>

Abstract

XPath is a language for addressing parts of an XML document, designed to be used by both XSLT and XPointer.

Status of this document

XPath

- Detailed technical description at w3/xpath
- Language to identify location of XML element
- Allows you to point to and select specific parts
  - ie XPath gives direction to the parser
  - enables drilling down to the node you want
- Nodes (various parts of the XML document)
  - root / element / attribute / text / comment / etc
- Context node: starting point

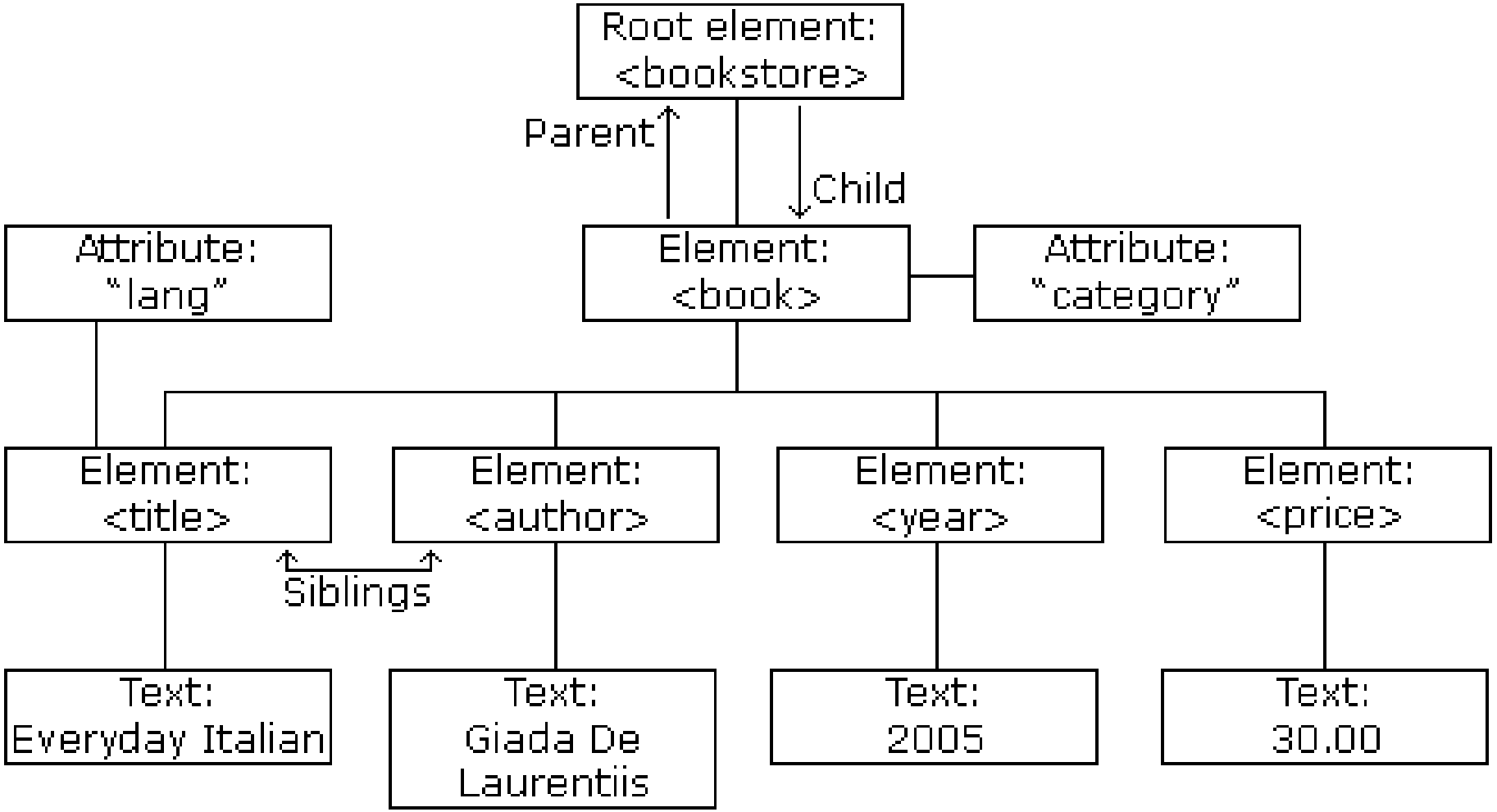
- Abbreviated and unabbreviated (full description)

Family tree

parent / child / sibling / ancestor / descendent

```
<root>
   <child>
   <subchild> … </subchild>
   <subchild> … </subchild>
   </child>
   <child>
    <subchild> … </subchild>
   </child>
</root>
```

XML tree structure



Source: <http://www.w3schools.com/xml/xml_tree.asp>

Schematic for XPath

- Possibilities (axes)
  - child/descendent (down the family tree)
  - (unlike humans all nodes [except one] have one parent)
  - sibling (across)
  - parent/ancestor/cousin (back up the family tree)
  - self (stand still)
  - genealogy does not apply to attributes

Wildcards to select unspecified XML elements

*	matches any **element** node
*@	matches any **attribute** node
node()	matches any **node**
text()	matches only **text**

XPath un/abbreviated

pathway cf filepath from which it was developed (uses step operator /)

<xsl:template match="element/child::title">

Matches title elements where its immediate parent is element

Usually abbreviated to :

<xsl:template match="element/title">

when processor finds this match will apply template to each in turn

To take value of the current node:

full - <xsl:template match="self::node()"> abbr - <xsl:template match=".">

To select an attribute:

full - "attribute::attributename" abbr - "@attributename"

/child::book/child::chapter::child::page abbr - /book/chapter/page

XPath- Axes

ancestor	matches ancestor(s) of the current node
ancestor-or-self	matches the current node and its ancestor(s)
attribute	matches attribute(s) of the current node
child	matches child(ren) of the current node
descendant	matches descendant(s) of the current node
descendant-or-self	matches the current node and its descendant(s)
following	matches node(s) after the closing tag of current node.
following-sibling	matches sibling(s) after the current node
namespace	matches namespace node(s) within the current node
parent	matches the parent of the current node
preceding	matches node(s) before the closing tag of current node
preceding-sibling	matches sibling(s) before the current node
self	matches the current node

XPath unabbreviated syntax

- Syntax: axis::node

eg: element/child::title

Abbreviated to: element/title

Combine with wildcards (see earlier)

self::* Selects the current node

usually abbreviated to .

self::* Selects the current node

child::name Selects all <name> nodes that are children of current node

attribute::type Selects the @type attribute of the current node

child::* Selects all children of the current node

attribute::* Selects all attributes of the current node

child::text() Selects all text child nodes of the current node

child::node() Selects all child nodes of the current node

descendant::email Selects all <email> descendants of the current node

ancestor::author Selects all <author> ancestors of the current node

child::*/child::last Selects all <last> grandchildren of current node

Can be combined eg:

ancestor-or-self::name Selects all <name> ancestors of current node – and the current as well if it is a <name> node

Abbreviations

.	Selects the current node
./name	Selects all <name> nodes that are children of current node
@type	Selects the @type attribute of the current node
./*	Selects all children of the current node
@*	Selects all attributes of the current node
./text()	Selects all text child nodes of the current node
./node()	Selects all child nodes of the current node
.//::email	Selects all <email> descendants of the current node
//author	Selects all <author> ancestors of the current node
./*/last	Selects all <last> grandchildren of the current node

Note: no abbreviated syntax for ancestor (or ancestor-or-self)

Existing DTDs and Schemas

- DTDs - best for document or text-intensive content
- Schema - best for data-intensive content (where you need to control your content more precisely)

Lists

```
<xsl:template match="ingredients">
  <ul>
  <xsl:for-each select="ingredient">
  <li>
  <xsl:apply-templates/>
  </li>
  </xsl:for-each>
  </ul>
</xsl:template>
```

URL: XML to XHTML

XML

<url> http://www.ucl.ac.uk/ </url>

<url target="http://www.ucl.ac.uk/" />

XHTML

http://www.kcl.ac.uk

Adding URLs (generating attributes)

If XML: <url>http://www.ucl.ac.uk</url>

```
<xsl:template match="url">
    <a>
    <xsl:attribute name="href">
    <xsl:value-of select="."/>
    </xsl:attribute>
    <xsl:apply-templates/>
    </a>
</xsl:template>
```

output:

```
 <a href="http://www.ucl.ac.uk">http://www.ucl.ac.uk</a>
```

Alternatively

If XML specifies URL as an attribute:

```
<url target="http://www.ucl.ac.uk"/>
<xsl:template match="url">
   <a>
   <xsl:attribute href="url">
   <xsl:value-of select="@target"/>
   </xsl:attribute>
   <xsl:value-of select="@target"/>
   </a>
</xsl:template>
```

output:

```
 <a href="http://www.ucl.ac.uk">http://www.ucl.ac.uk</a>
```

Working with Images

Need to access the attributes and generate attributes

```
<xsl:template match="image">
<img>
<xsl:attribute name="src">
<xsl:value-of        select="image/@imagename"/>
        </xsl:attribute>
<xsl:attribute name="alt">
        <xsl:value-of select="image/@caption"/>
</xsl:attribute>
</img>
</xsl:template>
```

Puts imagename as the value of the src attribute and caption as the alt attribute of <img>

CSS

- Zen Garden (see bibliography)
  - The Beauty of CSS Design: http://www.csszengarden.com/
- Combine CSS with XSL
  - <link rel="stylesheet" type="text/css" href="filename.css" />

- Test in different browsers (ie Firefox, IE, Safari)

- W3C CSS: http://www.w3.org/Style/CSS/
- W3schools CSS tutorial: http://www.w3schools.com/css/

Create link from html file to your CSS stylesheet

```
<head>
    <meta http-equiv="content-type" content="text/html;
    charset=UTF-8" />
    <meta http-equiv="Content-Language" content="en-gb" />
    <title>Page Title</title>

    <link rel="stylesheet" type="text/css" href="style.css" />
</head>
```

Where style.css is the filename of the css (assumed here to be in the same folder)

Adding comments

Use comments to document your work as you go

Syntax for XHTML / XML / XSL / DTD / Schema

<!-- this is a comment and will not be parsed -->

For CSS

/* this is a comment */