
Using Style Sheets to Publish XML to the Web

by Alan Houser

About the Speaker: Alan Houser is principal partner in Group Wellesley, a Pittsburgh PA-based company that provides documentation and content management consulting services to technology-based businesses. Alan holds degrees in electrical engineering and professional writing from Carnegie Mellon University, Pittsburgh, PA. He has more than 10 years of technical writing experience, including six years of experience with FrameMaker+SGML. Alan has presented seminars and corporate training courses in SGML- and XML-related topics. He has designed both SGML- and XML-based publishing solutions to support both single-source publishing for multiple audiences and personalized content for specific reader profiles. You can reach Alan at arh@groupwellesley.com.

Introduction

XML allows you to publish data to arbitrary output devices and to manipulate, select, and transform the elements you publish — allowing you to create customized documents from a single documentation source file. Two mechanisms exist for publishing XML to the World Wide Web: cascading style sheets (CSS) and the Extensible Style Language Transformations (XSLT). We will discuss the capabilities of the two style sheet languages, as well as the strengths and limitations of each.

Why Style Sheets are useful for HTML

The first of our two style sheet formats was finalized in 1996 by the World Wide Web Consortium (W3C). The W3C is the standards body that controls the development of a number of Internet-related standards, including HTML and XML. Before we look at style sheets and XML, let's look at what style sheets originally brought to HTML Web publishing.

If you have looked at HTML source files, you have probably seen something like this:

```
<font color="red" "ptsize"="12" family="serif" face="Times-roman">Troubleshooting your Acme Widget</font>
```

Note that the `` tag specifies the color, size, font family, and font face. If you were to look at the entire document, you might see that each block of text is wrapped in a `` tag. This is problematic for several reasons, including:

- It is difficult for human maintainers to read this HTML source code.
- To change the font characteristics of this page, you must edit each `` instance. This could be an arduous task, especially if changing the look of an entire web site!

There is a better way. A cascading style sheet allows you to apply formatting rules to specific sections of your HTML document. The text of the cascading style sheet can reside within the header of your HTML files, or within an external style sheet file. Cascading style sheets provide several benefits to Web site content authors, designers, and maintainers, including:

- simplification of the HTML source code required to display a Web page.
- the ability to change the format of an entire Web site by changing a few lines of the style sheet, instead of making changes throughout the HTML source code.

Why Style Sheets are Essential for XML

While style sheets are useful when working with HTML, they are essential when displaying XML to a Web browser. HTML is a fixed vocabulary. Browsers know exactly what tags can comprise an HTML document, and therefore know how to render (format) the contents of each HTML tag in a browser window. Browsers know to expect tags like `<p>`, `<h1>`, `<table>`, `
`, and browsers know how to display text when they encounter each of these tags in an HTML source file.

XML, however, is not a fixed vocabulary. An XML source file looks a lot like an HTML source file, with one major exception — the tag names are not predefined. The document author, or a company, or an industry, or perhaps a consortium

defines the set of XML tags used. XML is “extensible” because it allows authors (or organizations, or consortiums) to define sets of tags, or XML vocabularies, that describes the content of their XML source files.

Because XML is not a fixed set of tags, a Web browser cannot possibly know in advance what tags it will encounter in XML source documents, let alone know how to render each XML element in the browser window. The browser needs explicit instructions for how to render each XML tag. A style sheet provides instructions that map XML tags to formats that your Web browser can use to display an XML document.

How Cascading Style Sheets work with XML

If you examine a cascading style sheet, you will see a series of lines that look something like the following:

```
h1 {font-family:'Helvetica, sans-serif'; font-weight: bold; font-size:100%; color:#000000;}
p {font-family:'Helvetica, sans-serif'; font-size:90%; color:#000000;}
ul {font-family:'Helvetica, sans-serif'; font-size:90%; color:#000000;}
```

Each line forms a style sheet rule, which consists of a selector component (in the example above, h1, p, and ul), followed by a properties component (the text within the curly braces).

To create a cascading style sheet for an XML document, you use XML tags that appear in the document in the rule section of the style sheet. For example, you may have a fragment of an XML document that looks like the following:

```
<procedure>
<procedureTitle>Creating an XML Style Sheet</procedureTitle>
<procedureBody>Before you create an XML Style Sheet, it is helpful to have the Acme Style Sheet Widget <partNumber>5115-6112</partNumber> on hand.</procedureBody>
</procedure>
```

In the example above, the XML tags <procedureTitle>, <procedureBody>, and <partNumber> denote the title, body text, and part number, respectively. You might imagine using these tags in an application to, for example, create a list of procedure titles with links to the appropriate procedure text. However, these tags provide no information about how their content should be displayed by a Web browser.

To tell a Web browser how to display these XML tags, we need a style sheet. A style sheet for the above example might look like this.

```
procedureTitle {font-family:'Helvetica, sans-serif'; font-weight: bold; font-size:140%; color:#000000;}
procedureBody {font-family:'Helvetica, sans-serif'; font-size:100%; color:#000000;}
partNumber {font-family:'Helvetica, sans-serif'; font-size:90%; color:#000000;}
```

Note that this style sheet specifies the font family, weight, size, and color for the elements <procedureTitle>, <procedureBody>, and <partNumber>.

Attaching a Cascading Style Sheet to an XML Document

Unlike HTML, which permits style sheet directives within the HTML source file, an XML style sheet must be created in a separate file. An XML document references a cascading style sheet through a special XML construct called a “processing instruction.” If the style sheet for an XML document is in the file “document.css,” the processing instruction for referencing the style sheet will look like this:

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

Limitations of Using Cascading Style Sheets with XML

Cascading style sheets provide a relatively easy way to display XML content to Web browsers. If you use cascading style sheets with XML, however, you should be aware of the following limitations:

- Your browser must be XML-aware (at the time of this writing, Microsoft Internet Explorer version 5.0 or later) to properly display XML with a cascading style sheet. Results with other browsers will be unpredictable.
- Cascading style sheets do not allow you manipulate your XML document in any manner. You cannot select, sort, filter, count, or re-arrange your XML elements with a cascading style sheet.

Of course, you can’t manipulate HTML documents, either. There is no easy, standard way to select, sort, filter, count, or re-arranging the content of an HTML document.

Herein lies the true capabilities of XML — that you can perform database-like operations on your XML documents. To unlock this capability, you need the second type of XML style sheet, the XSL Transformation Language, or XSLT.

Publishing XML with XSLT Style Sheets

The XSL Transformation language (XSLT) is another specification of the World Wide Web Consortium. XSLT, along with the related XPath (XML Path language) specification, provide the ability to perform database-like operations on XML documents. Using XSLT/XPath, you can:

- select specific elements in an XML document
- sort elements
- filter (remove) specific elements
- count or perform other arithmetic operations on elements or the contents of elements (for example, an XSLT style sheet can total prices in an XML invoice)
- rearrange (reorder) elements
- transform XML elements to other XML elements or to HTML elements (for example, you can transform the <procedureTitle> element in the previous example to <h1> for display in a Web browser)

These capabilities provide a powerful mechanism, not only for customizing the publishing of XML documents, but for transforming XML documents into HTML that can be displayed by any Web browser.

Comparing CSS and XSLT

Unlike CSS, which maps formatting information to specific XML elements, XSLT actually transforms an XML document to HTML for display by a Web browser. An XSLT transformation has three components:

- the XML source file
- the XSLT file, which contains rules for transforming XML elements to HTML elements
- the HTML result file, which the browser displays

This may seem unnecessarily complex. If you are publishing the content of your XML files to the Web as-is, it may actually be unnecessarily complex. XML and XSLT are most useful for publishing multiple documents to multiple audiences from a single set of XML source files. To change the HTML that you publish, you only need to change your XSLT file. You may choose to maintain different XSLT files to create different HTML documents.

(XSLT is actually not limited to creating HTML output files. The result of an XSLT transformation can be another XML document, or even a text file.)

Performing an XSLT Transformation

An XSLT transformation is typically performed by an XSLT processor, which is a stand-alone program that parses the input XML file, applies the XSLT transformation rules, and writes an output file. Several free XSLT processors are available in a number of different languages, including Java, C++, and Perl. Several easy-to-use Win32 executable versions are also available. My favorite XSLT processors are Saxon (<http://users.iclway.co.uk/mhkay/saxon/>) and XT (<http://www.jclark.com/xml/xt.html>). Both are available as Win32 executable files.

Microsoft Internet Explorer version 5.0 or greater includes a built-in XSLT processor. However, Microsoft Internet Explorer supports an older version of XSLT. To upgrade IE5 to handle the final XSLT specification, you must install the MSXML updater that is available from Microsoft's Web site, at <http://msdn.microsoft.com/xml>.

A Simple XSLT Transformation

Consider the following XML file. This file represents a list of books, with category, title, and price information, all denoted by XML tags.

```
<?xml version="1.0" ?>
<BOOKLIST>
<ITEM>
<CATEGORY>XML</CATEGORY>
<TITLE>HyperPublishing with XML</TITLE>
<PRICE>$44.95</PRICE>
</ITEM>
<ITEM>
<CATEGORY>XML</CATEGORY>
<TITLE>Pete's Guide to XML</TITLE>
<PRICE>$32.95</PRICE>
</ITEM>
<ITEM>
<CATEGORY>XML</CATEGORY>
<TITLE>Mastering XML with Alan</TITLE>
<PRICE>$35.95</PRICE>
</ITEM>
</BOOKLIST>
```

In this example, our goal is to use an XSLT style sheet convert the previous XML file to HTML for display in a Web browser. This style sheet we will use builds an HTML table, with one row for each item. Then, for each item (row), the style sheet:

- 1 Inserts the title of each book into the first cell.
- 2 Inserts the category of each book into the second cell.
- 3 Inserts the price of each book into the third cell.

In the XSLT file example that follows, tags (text within angle brackets) that begin with “xsl:” are style sheet directives. All other tags form HTML “templates” into which XML data from the previous file is inserted.

```
<?xml version="1.0" ?>
<xsl:stylesheet xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
version="1.0">
<xsl:output method="html" />
<xsl:template match="/">
<html>
<head>
<title>The book catalog listed in a table</title>
</head>
<body>
<table border="1" cellspacing="0" cellpadding="5">
<tbody>
<xsl:for-each select="BOOKLIST/ITEM">
<tr>
<th align="left">
<xsl:value-of select="./TITLE" />
</th>
<td>
<xsl:value-of select="./CATEGORY" />
</td>
<td>
<xsl:value-of select="./PRICE" />
</td>
</tr>
</xsl:for-each>
</tbody>
</table>
</body>
</html>
</xsl:template>
</xsl:stylesheet>
```

The result HTML file follows. Note that the order of the data in the HTML file is not the order of the data in the XML source file. Using an XSLT style sheet, we were able to re-order the data, then transform it to an HTML format that we could display in a Web browser.

```
<html>
<head>
<title>The book catalog listed in a table</title>
</head>
<body>
<table border="1" cellspacing="0" cellpadding="5">
<tbody>
<tr>
<th align="left">HyperPublishing with
XML</th><td>XML</td><td>$44.95</td>
</tr>
<tr>
<th align="left">Pete's Guide to
XML</th><td>XML</td><td>$32.95</td>
</tr>
<tr>
<th align="left">Mastering XML with
Alan</th><td>XML</td><td>$35.95</td>
</tr>
</tbody>
</table>
</body>
</html>
```

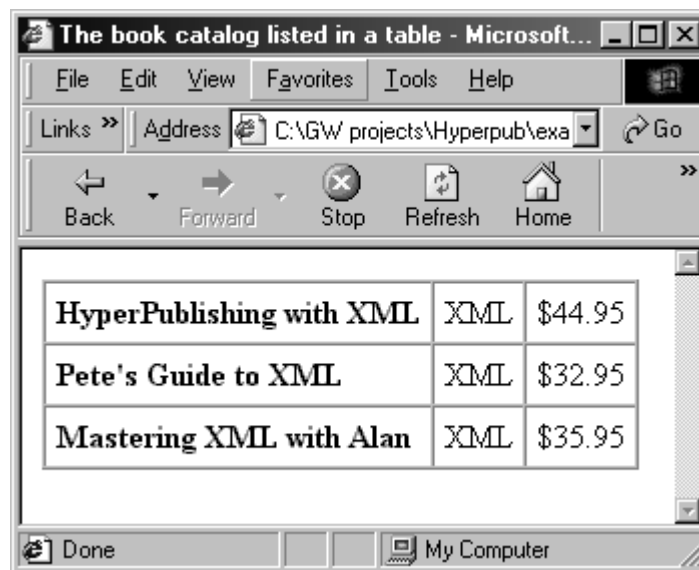


Figure 1 Result of XSLT Transformation

Advantages of XSLT

- XSLT transformations allow us to select, manipulate, and transform our XML documents just as we could if the information were stored in a database.
- XSLT transformations allow us to create HTML from our XML documents. We can design this HTML to be viewable in any Web browser.

- We can create a different HTML result file by changing our XSLT transformation. We have completely separated content and format — we don't have to edit our XML content files to change the way we display them.
- We can create different renditions of our XML data by applying different XSLT transformations. We might choose to maintain multiple XSLT files to perform different transformations on our XML source files and produce different HTML results.

Limitations of XSLT

- XSLT transformations are normally performed on the Web server, with the resulting HTML files served to the client browsers. Only Microsoft Internet Explorer version 5 or greater will perform XSLT transformations directly. Furthermore, you must install an updater from Microsoft to use the final version of XSLT with Internet Explorer 5 or greater.
- XSLT, and the accompanying XPath language, are quite powerful, and therefore quite complex. Writing XSLT style sheets is a task for somebody with, at minimum, experience with scripting languages like Perl.

Conclusions

Cascading style sheets (CSS) provide a mechanism for displaying XML documents in Web browsers that provide CSS support. Cascading style sheets allow you to specify formatting information for the XML tags that comprise your document. Cascading style sheets do not allow you to select or transform content from within your XML document for display — CSS provides “all or nothing” display to a Web browser.

XSLT style sheets provide a powerful mechanism for manipulating, transforming, and selecting the content of XML documents for display to a Web browser. XSLT and its accompanying language, XPath, comprise a rich programming language for manipulating XML documents. Organizations that wish to take full advantage of the capabilities of XML documents are likely to use XSLT as a production tool.

Resources

- W3C CSS Specifications and Information
<http://www.w3.org/Style/CSS/>
- W3C XML Area
<http://www.w3.org/XML/>

- W3C XSLT Specification
<http://www.w3.org/TR/xslt>
- *XSLT Programmer's Reference* by Michael Kay (Wrox Press, 2000, ISBN 1-861003-12-9)
- *Cascading Style Sheets: The Definitive Guide* by Eric Meyer (O'Reilly & Associates, 2000, ISBN 1-56592-622-6)
- *Core CSS* by Keith Schengili-Roberts (Prentice Hall, 2000, ISBN 0-13-083456-4)
- *Cascading Style Sheets: Designing for the Web* by Håkon Wium Lie and Bert Bos (2nd edition, Addison-Wesley 1999, ISBN 0-201-59625-3)