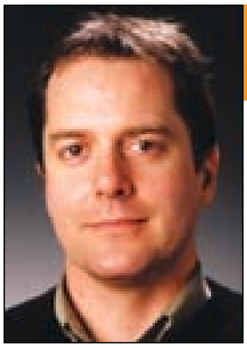

Evolving with XML

This column covers the rapidly evolving world of XML languages and how they affect the future of Web publishing. If you have comments or questions, please contact Doug McLaughlin at ***dmclaughlin@mindspring.com.***

Douglas J. McLaughlin, *Column Editor*

Styling XML: An Opinionated Guide

By Douglas J. McLaughlin

XML has arrived! While many of us have heard the steady stream of accolades for this technology over the past couple of years, we have perceived XML as a structured data format: human-readable, cross-platform, able to solve entire classes of problems in the world of information architecture. This perception, however, related mostly to back-end data systems and middle-tier software programming. In the presentation layer of the Web—publishing and user interface design—XML hardly seemed to matter.

Until now. Support for XML is now available in three major browsers: Microsoft *Internet Explorer 5*, Netscape *Navigator 6* (still in preview release at press time), and *Opera 4*. This support shows the potential of XML as the new dialect of the Web, but there are still some key issues to resolve. The most important issue is that of style languages, which provide the presentational instructions necessary to make XML visible in a browser.

Currently, the options for rendering XML content in the browser are XSLT (eXtenstible Style Language for Transformation) and CSS (Cascading Style Sheets). While *Internet Explorer 5* has additional data rendering capabilities (e.g., Data Islands, which allow the developer to bind XML data to HTML elements), XSLT and CSS are crucial to this discussion because they are standards-based, are nonproprietary, and have the vendor support necessary to succeed.

Unfortunately, there is no uniform approach to rendering XML data in the browsers listed above. At this time, only *Internet Explorer 5* provides anything resembling XSLT support in addition to support for CSS; *Navigator* and *Opera* provide only CSS support.

So, let's have some fun. Let's examine the strengths and weaknesses of XSLT and CSS in bringing XML to the browser.

CSS and HTML: Good!

Cascading Style Sheets have been used in Web development for several years now. Because CSS allows developers to separate content from presentation, it is a powerful tool in the development and management of Web sites. CSS is also very simple and easy to learn. It consists primarily of **selectors** and **declarations**. A selector is the element of a Web page whose appearance you want to control; a declaration is the description of the appearance. For instance:

H2 {margin-left: 10px; color: red}

This code tells the browser to apply a ten-pixel left margin and color the text red for every H2 element. H2 is the selector, and the portion inside the curly braces is the declaration. CSS has additional functionality that relates to positioning on the screen, use of multiple style sheets and alternate media types, and contextual style rules, but it really doesn't get much more complicated than the selector-declaration syntax illustrated above.

CSS and XML: Well… Hmmm…

From this, it can be inferred that CSS can also render XML documents in the browser. For example, suppose that, among the XML elements, you have <name> and <hobbies>. Your CSS code could be:

name {font-family: Arial; font-size: 10pt}
hobbies {font-family: Arial; font-size: 10pt; font-style: italic}

Since CSS is a tool for separating content from presentation, the selector-declaration syntax works quite naturally with XML. Instead of HTML elements, which contain presentational instructions, we now bind CSS style declarations to XML elements, which describe data in an intelligible way. CSS2, the latest incarnation of Cascading Style Sheets, also has some declarations that substitute nicely for HTML:

name {display: table-row}
hobbies {display: list-item}

But while CSS allows us to view XML documents in a Web browser, isn't this approach a bit limited? Norman Walsh, a key contributor to the XSLT specification, thinks so: "In order to effectively render content, you need…a facility for transforming the source document into the layout document (in XSL, the transformation language). Decorating the source tree is not sufficient for most applications." (To read Walsh's entire article, visit ***www.xml.com/xml/pub/1999/06/xsl_edit.html***)

Decorating the source tree? The presumption here is that the XML data, which have a tree-like structure, are the source of some sort of transformation. Whether through XSLT or through programming interfaces such as SAX (Simple API for XML) or DOM (Document Object Model), XML works best when it is fluid and mutable; because it serves a variety of purposes, it must undergo a variety of transformations, the most common being from XML to HTML.

The result of transformations to the source tree is called the "result tree." The source tree and the result tree may well be one and the same for simple Web documents, but for Web applications, the source tree and the result tree can—and should—be quite different. Otherwise, reusability, one of the key advantages of XML, is lost.

For example, what if the XML data are best rendered in a form control, such as a select box? What if the source tree contains address information and the user only wants to view addresses in a particular city or sort addresses alphabetically by state? What if the source tree is an XML version of *Hamlet* and one user wants to view only the speeches by Lord Polonius and another wants to read only the speeches of King Claudius?

Because the result tree of XML processed with CSS is identical to the source tree, CSS is very limited in addressing these needs. CSS makes the source tree visible in a browser, but it does little to transform the source tree in response to user input. It locates XML elements and, if it recognizes them, renders the information they delimit based on a style declaration.

This limitation is not only browser-based. What if you want to transform XML to another data configuration? For example, what if the data need to be converted to WML (Wireless Markup Language) to accommodate the profusion of wireless information appliances currently hitting the Web? Or, perhaps, a trading partner requests XML data, but the partner has a different vocabulary for describing the data ("You call them *customers*, but we call them *buyers*."). How do you convert XML into a format that can be used in your partner's own systems?

XML and XSLT

Extensible Style Language (XSL) consists of two parts: a language for transforming XML documents (XSLT) and a language of formatting semantics (FO). The formatting semantics specification, which is somewhat similar to CSS and appears to target print media more than Web publishing, is still under development. At this time, it enjoys little or no browser support. XSLT became a formal World Wide Web Consortium (W3C) recommendation in November 1999 and is enjoying enthusiastic support. We will refer to the transformation language as XSLT, but please note that the terms XSL and XSLT are often used interchangeably.

With XSLT, you can:

- Transform XML data from one document type, or schema, to another
- Transform XML data to HTML, which, if done on a Web server, is viewable by any browser
- Deploy a number of native functions that perform arithmetic calculations; format numbers, times, and dates; and call inline scripts
- Transform XML data to a number of text formats, including comma- or tab-delimited files, rich text files, plain text files, and Structured Query Language (SQL) statements
- Embed script blocks that control both how XSLT processes the XML data at runtime and the behavior of HTML after the transformation has occurred

In addition to the capabilities listed above, XSLT incorporates an additional W3C specification: XML Path Language (XPath). XPath, also released in November 1999, is a powerful pattern-matching language that enables the user to sort, filter, and query XML data. This capability allows XML data to be output as an entire set or as precisely targeted subsets. XPath is a separate specification from XSLT because other XML languages, such as XPointer, can use it.

While XSLT and XPath will provide a powerful set of tools for XML developers, this doesn't completely preclude the use of CSS with XML. For certain purposes, CSS works just fine. The Mozilla developers obviously think so, since CSS is used with XML-based User Interface Language (XUL) to customize the browser's user interface.

In deciding whether to use CSS or XSLT to style XML data, you should first ask yourself whether the data will need to change based on user input or simply need to be readable inside the browser. This decision gets right to the heart of a dichotomy in the XML movement. Is XML a language for marking up documents, or is it a reusable data container? Before you answer this question, I would like to point out that using CSS and XSLT to display XML data might not be an either/or proposition. In a recent project, I used both of them together. Code samples will follow in my next column.