Review: Using XML to Separate Content from Prensentation

By Steve Hoenisch
Last updated on June 1, 2002

Table of Contents

---

1 Introduction

You've probably heard the propaganda by now: XML blesses you with a way to separate content from presentation. Separation in turn yields productive gains over HTML and other data formats used to manage content: In a process sometimes called *single-sourcing*, the content of an XML document can be formatted for display in a web browser, reformatted for delivery to such devices as mobile phones and handheld computers, and converted into a PDF file suitable for printing. Meantime, parts of an XML document, such as paragraphs that supply background information about a given subject, can be reused in other XML-encoded documents. What makes all this possible? XSLT.

Extensible Stylesheet Language for Transformations, or XSLT, is a functional programming language that enables us to bridge the gap between content and presentation by providing a means of specifying how a content-based XML document is transformed into a presentation-oriented document or data format.

---

2 XSLT and CSS

2.1 Cascading Style Sheets

XSL has a related standard, though unlike XSL it is not based in XML: Cascading Style Sheets (CSS). In the context of web publishing, Cascading Style Sheets (CSS) can also be used, at least theoretically, to statically format XML documents for display, but it cannot be used to transform them in any meaningful way.Until there is much stronger web browser support for displaying XML documents with CSS, it is not a practical option for web publishing.

2.1.1 Separating Formatting from Styling

Using CSS to complement XSLT, however, is a powerful strategy for building web pages -- a strategy that splits presentation into what I call formatting and styling. Formatting can be seen to include basic HTML markup like headings, horizontal rules, lists and the like. Styling, meantime, defines the visual properties of markup: Its colors, sizes, widths, margins, bullet types, and so forth. Although the distinction between the two is not always clear-cut, formatting typically appears in the form of elements, styling information in the form of attribute-value pairs. For example, in `<h1 style="color: olive;">` the h1 element formats the text as a first-level heading and the values of the style attribute, used for specifying inline CSS styles, reflect how the text should be styled.

2.1.1.1 One Keystroke Changes Everything

You can gain a great deal of utility from separating as much styling information as possible from the formatting and

placing it a Cascading Style Sheet. The separation gives you a way to make wholesale design changes to a web site without having to change the formatting code in every HTML document; if you've set up your web pages properly, with all of them linking to a single CSS, you merely make the stylistic changes in one file, the Cascading Style Sheet. *Cascading Style Sheets: The Definitive Guide*, published by O'Reilly, provides a detailed account of how to use CSS. The W3C CSS specifications are available at http://www.w3c.org/Style/CSS/.

2.2 Principles of Separation

There are other principles of separation that can be immensely useful in building well-engineered, text-based web pages with XML, XSLT, CSS, and HTML. An overarching objective of separation is to use XML to structure content and XSLT and CSS to format it in a way that minimizes redundancy and maximizes flexibility, including the capability to repurpose content and publish it in various formats. Most of the principles listed below are best applied to narrative-oriented documents that will be published as white papers, technical manuals, help files, essays, and so forth. Keep in mind that these principles are merely a guide and that the list of exceptions is long: Your data, purpose, audience, delivery format, and other factors will influence how you engineer your system's own matrix of structure, metainformation, parameters, content, and presentation.

2.2.1 XML Documents

Strive to maximize the content that exists as text, not as tags, in your XML documents. In other words, content that you expect end users to see should usually be set as content, not as elements or attributes. Avoid setting content as tags, either as elements or attributes, even if doing so comes at the cost of some redundancy; it's easier for content authors to work with content that is out in the open, not hidden as the value of attributes or as elements themselves. Instead, use elements to describe the structure or content of your material; use attributes to capture metainformation about the content or its structure and to encode parameters that are used to process the content. Place recurring content, especially content that may change, such as the name of a product under development, in entities. Shy away from placing formatting or styling information in your XML documents, though it may be expedient to include some table and image formatting instructions.

2.2.2 XSLT Stylesheets

Place as much formatting information as possible in your XSLT stylesheet and eschew using it as a container for standard content. Instead, place content out in the open in your XML documents even if doing so comes as the expense of a little duplication. However, highly variable content, such as the date of publication and the version number, may be best placed in the XSLT stylesheet as entities. It may also benefit you to make exceptions for content that varies by audience, as in the case of parameterized settings used in internationalization.

2.2.3 Using Cascading Style Sheets

As I mentioned above, Cascading Style Sheets should complement the XSLT stylesheet by containing as many as possible of the formatting code's visual styling properties.

Note: In the examples that accompany this tutorial, you'll see some of these principles at work.