Lurching Toward Babel: HTML, CSS, and XML

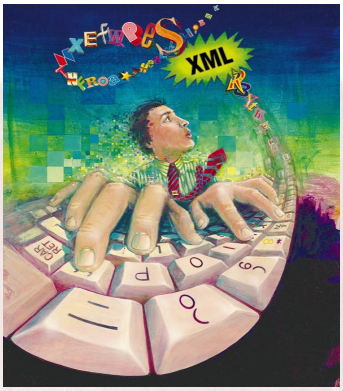
Jukka Korpela, Helsinki University of Technology

When HTML was young, it was described as a scalable document format that could be used for information exchange on virtually any platform. HTML's designers specifically mentioned that HTML documents could be presented not only in GUIs but for text-only systems, text-to-speech devices, and even for Braille renderings.

Such enthusiastic descriptions of HTML used to appear regularly on the World Wide Web Consortium's (W3C) Web site (http://www.w3.org/pub/MarkUp/). But lately the tone has changed. The W3C no longer believes HTML can achieve its original goals.

HTML's underlying idea was very simple: Define a simple language that described the structure of a document and expect companies to develop software that can present such documents in different environments and according to various user options. Using HTML, authors could create something that could easily be presented in any visible or audible format—on the fly—without doing anything extra.

As it turns out, however, this idea, though disarmingly simple, deviated from well-established publishing practices. In traditional publishing, graphic designers and layout artists consider the specific features of the presentation medium, including the paper size and quality, the color palette, and so on. It has been very difficult to switch from this approach to a simpler one, in which the author provides the content and specifies the logical structure, leaving the presentation to user agents.



Hold tight with HTML. The XML/CSS model is heading into a land of confusion.

It is not surprising, then, that when browser vendors started adding methods for specifying presentation details—such as font size and colors—authors started using them. But they weren't satisfied with this early technology. They kept asking for publishing-quality typographic control on the Web. Designers and authors got some of what they wanted, and we got this mess.

SEPARATING STRUCTURE AND PRESENTATION

Style sheets in general—and cascading style sheets (CSS) in particular—separate structure and content from presentation. Applied to the Web and HTML, this means that HTML doesn't contain presentational features. Instead, separate tools specify presentation.

For example, instead of using HTML markup like

```
<H1 ALIGN=center><FONT COLOR=
 red>My Heading</FONT></H1>
```

for all first-level headings, you would write them as

```
<H1>My Heading</H1>
```

and attach a separate style sheet. The style sheet would contain content like

```
H1{text-align: center; color:
 red}
```

which specifies presentation to be applied to all first-level heading (H1) elements.

The CSS approach makes documents simpler to write and maintain. You could write a single style sheet and use it for hundreds of documents. Even the first CSS version, CSS1, addresses presentation issues—such as physical font sizes and dimensions—that cannot be handled in HTML at all.

While CSS1 is simple and intuitive, it is still yet another language to learn. (There is a very good CSS1 tutorial by the Web Design Group at http://www.stack.nl/htmlhelp/reference/css/.) When writing CSS1, it is easy to make typos that make browsers behave strangely, but it is also very easy to check documents using a style sheet checker.

In theory, a browser can apply several style sheets to a document as a cascade, building sophisticated document presentation. This means, for example, that a company could have its own company-wide style rules while individual authors within the company could add features to these style rules through their own personal presentation rules.

Moreover, any reader could have a personal style sheet, for example, for setting font sizes large enough for easy reading. A browser that supports CSS takes all

Editor: Ron Vetter, University of North Carolina at Wilmington, Mathematical Sciences Dept., 601 South College Rd., Wilmington, NC 28403; voice (910) 962-3671, fax (910) 962-7107; vetter@cms. uncwil.edu

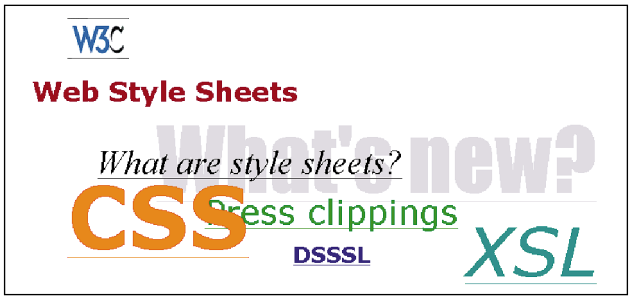


Figure 1.W3C's CSS page on Internet Explorer 4.0 with default settings intact.

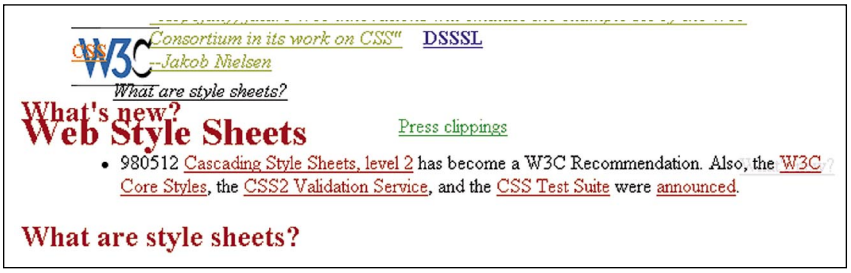


Figure 2. W3C's CSS page on IE 4.0 set to "ignore font styles and sizes."

these styles into account, not by calculating averages, but by strict preference rules.

DOES IT REALLY WORK?

Style sheets sometimes work, but more often they don't. Authors who try to use style sheets encounter serious problems in current implementations.

To begin with, the style sheet support in Internet Explorer 3.0, a browser that is still widely used, is worse than nonexistent: It is hopelessly incomplete and buggy. IE 4.0 and Navigator 4.0 have better CSS1 code.

Some authors have been able to make their style sheets work on both browser families. But it takes serious effort and a lot of help to circumvent browser bugs. (See "Tips and Workarounds" in a list of CSS resources at http://home.att.net/%7Esjacct/.)

The question is not whether it is possible to implement CSS well, but whether there are inherent problems in CSS caused by the very ideas that gave birth to it. In practice, if you want to maximize the probability of having your headings displayed red, you'll use `FONT COLOR=red` in HTML. Any browser that is capable of supporting style sheets also supports the `FONT` alternative. So why bother with CSS?

Many authors don't. They continue to do their markup in HTML. They might consider adding CSS later to do things that can't be done in HTML, but many see CSS just as a new way of achieving the same results.

COULD CSS WORK?

The most obvious problem with CSS is the cascade itself, which allows you to connect multiple style sheets to a document. The cascade is difficult to implement, but its difficulty isn't the major problem. If a browser implements the cascade correctly, it is still practically impossible to guarantee good or even tolerable presentation. Effective presentation consists of several ingredients, such as color, positioning, dimension, and font property, which must be compatible with each other.

Figure 1 shows the official CSS page produced by the W3C (http:// www.w3.org/Style/). With CSS support turned on—using just the style sheet provided with the page—the site is indeed legible. (It is important to note that it looks just fine without CSS support. Another way of putting this is that the page *degrades gracefully*, which is one of the supposed benefits of CSS.)

But even an extremely simple cascade can turn it into a total mess. Use, say, IE 4.0 and set the browser options to ignore the font styles and sizes specified on a page. Doing so is comparable to using a user style sheet. Then reload the page and see what happens. The resulting mess, shown in Figure 2, is indicative of CSS's major problem.

With CSS, you can present hypertext links in boxes, with colored underlines, or with unique text colors, among other stylistic characteristics. You can even combine these methods. But what happens if you want to present links using blue only? In a worst-case scenario, another style sheet might specify the same color for the background, for normal text, or for emphasized text. This means, in effect, that in order to avoid massive confusion a style sheet must specify colors for everything that can have color. It must set colors for background, normal text, visited links, headings, and so forth.

Similarly, if a style sheet sets any colors at all, it should set all link colors. But if you use this technique, some users who see the document using your style sheet will not see links in regular, default colors. And they may not recognize them as links at all.

If you specify colors for everything, your style sheet could safely participate in a cascade: The browser would either use all those color specifications or none of them. However, a user's style sheet might still override your color specification for some particular element and perhaps ruin your entire layout. This wouldn't happen if the user style sheet were designed as carefully as yours—that is, if the user's style sheet accounted for all heading levels, link colors, and so forth. But isn't it a bit too much to rely on two human beings doing things perfectly?

Graceful degradation is, at best, something you can achieve using style sheets. But it isn't automatic. If presentation really matters, CSS is not a reliable tool; if presentation is not that important, why bother writing specifications for it?

Even if your cascade works well technically—and doesn't make the document illegible—can cascading style sheets produce style? Ultimately, no. Basing a presentation on a combination of features from varying independent sources is like letting one person select a suit for you, letting someone else choose the shirt, defaulting your shoes to whatever you happen to be wearing, and then carefully picking the socks yourself, unaware of what else you'll be wearing later in the day.

STYLE SHEETS WITHOUT CASCADE?

Should we deduce, then, that only one style sheet should be applied to a document? Doing so would simplify things technically and would probably speed

Continued on page 106

Continued from page 104

them up, too. With one style sheet, a document could be presented using either a user's or an author's style sheet, or perhaps just the browser's default settings.

But what do we gain by doing so versus making a document available both in HTML and in PostScript? With the latter methods, the user could view the document on a browser, according to his or her browser settings, or in a PostScript reader, exactly as the author intended it to be seen.

According to W3C, the CSS is supposed to engender a balance between the author and the reader. The question is whether the author or the reader is in command of document presentation.

There are a huge number of Web-savvy publishing methods for controlling presentation. Nothing prevents us from distributing documents in, say, MS Word, PDF, PostScript, or Autocad. HTML was designed to be a fundamentally different publishing alternative. If you, as an author, find this acceptable, perhaps even desirable, you'll use HTML; if not, there are a large number of alternatives.

XML: TOWARD BABEL

Currently, many designers advocate the use of CSS together with the Extensible Markup Language (XML) as a replacement for HTML. Generally, W3C promotes XML as if it were an extension to HTML. In fact, XML is a simplified form of Standard Generalized Markup Language. SGML, in turn, is used to define the syntax of markup languages like HTML.

In other words, XML is basically a dialect of SGML. But by advocating the use of XML on the Web, W3C is essentially suggesting that everyone design a personal language for personal hypertext documents and different languages for different documents.

The XML metalanguage can define the formal syntax of a language, such as nesting rules for elements. The semantics could of course be described in plain English. But this doesn't seem to be of interest to XML evangelists. They are more interested in just specifying presentation with CSS. Naturally, this means that they do not use CSS as a presentation suggestion only, since (with the XML/CSS model) there is no default or user-defined presentation.

Quite probably there will soon be browsers that support XML and CSS to some technically satisfactory degree. The XML/CSS approach to Web publication might well be adequate in special cases. But switching from HTML to XML/CSS as a general solution would be a huge "devolutionary" leap.

As a publishing method, XML/CSS is comparable to using text processing software with styles or macros: XML/CSS structures documents, but the structure has no independent, public semantics. The XML/CSS approach means that instead of developing HTML so that it can adequately express the structure of a wider range of documents, we must create new markup languages.

What this means in practice is illustrated by MathML (http://www.w3.org/TR/REC-MathML/), the mathematical markup language. MathML is, to put it mildly, complex and difficult compared to the old proposals for adding basic mathematical markup into HTML (like the long-expired proposal in the HTML 3.0 draft at http://www.w3.org/MarkUp/html3/).

The Web needs a Renaissance. It must return to its classical roots. One of its classical roots is HTML as a simple, scalable, document format that can be used for information exchange on virtually any platform. Coming back to this model means a return to the original principles of HTML and very carefully extending the language in the spirit of those principles.

It will take time before we all realize that the original HTML proposals are still much stronger than the latest XML/CSS developments.

If you have been wondering whether you should hurry to catch the train and start learning XML and CSS, stop wondering. There is no need to run to the station. The XML/CSS train is leaving, but it's headed into a land of confusion. Hold tight. There is still a lot of good work to be done with that simple, scalable document format we call HTML. ❖

Jukka Korpela is a systems analyst in the Computing Centre at Helsinki University of Technology, Finland. Contact him at Jukka.Korpela@hut.fi or http://www.hut.fi/u/jkorpela/.