BY JEFF HOLTZMAN
COMPUTER EDITOR

XML: Who Controls The DTD?

YOU MAY HAVE CAUGHT THE BUZZ ABOUT XML (EXTENDED MARKUP LANGUAGE). WHAT IS IT? WHAT IS IT GOOD FOR? HOW DOES IT DIFFER FROM HTML? WHAT DIFFERENCE WILL IT MAKE—IF ANY? THOSE ARE ALL GOOD QUESTIONS, AND WE'RE GLAD you asked. To answer them, we'll take a kind of "outside-in" approach, by comparing HTML, SGML, and XML.

If you've had any exposure to the Internet, you probably have an intuitive notion of what HTML (Hyper Text Markup Language) is. Basically, HTML is a language for specifying the layout of Web pages. Any given HTML document contains two broad categories of information: content and markup. In HTML, markup specifies how content appears. For example, you can specify various levels of headings, emphasize (with bold or italic) portions of the content, or display several content items as a list. In HTML, markup is contained between angle brackets, <like this>; everything else is content.

The problem with HTML is its simplicity. It can only do a limited range of things, and as it evolves, different products interpret its markup differently, or in some cases, not at all. Eventually, one might suspect, a set of standards would be reached, and vendors would be able to create compatible applications that conform to those standards. From that point of view, the problem is not just the rapid rate of evolution, but the idea that there are probably an infinite variety of ways it could evolve to meet different needs.

There are deeper problems as well. HTML only concerns itself with appearance and presentation. HTML doesn't touch the question of structure. It turns out that there are lots of interesting and useful things that can be done when you concern yourself with both.

FIG. 1—COMING SOON to a browser near you: XML.

SGML

In the larger scheme of things, HTML is an application of SGML (Standard Generalized Markup Language). SGML is not a markup language, like HTML. Rather, SGML is a *metalanguage*, that is, a language for specifying other markup languages, like HTML. In other words, SGML is markup language language.

SGML is an ISO standard; it grew out of research dating back to the 1960s. Ordinary people don't use SGML; large publishing organizations do. Typically, it is used to organize large publishing projects such as airplane maintenance manuals and encyclopedias.

The way you do anything useful with SGML is by creating a Document Type Definition (DTD). The DTD specifies what the elements of a document may be, and how they relate to one another. HTML, in fact, is nothing more than an SGML DTD; an often very loosely interpreted DTD, but a DTD nonetheless. The HTML DTD does nothing but specify the set of tags, such as <h1>, , and so forth, that make up a Web page. Microsoft and Netscape may choose to render (present) <h1> in different ways, but the fact is that they're both working off the same set of tags, that is, the same DTD.

Sort of. Actually, they're not, and that's why different Web pages display differently—or not at all—in different browsers.

So, what's the solution? You could add more tags, but that is a poor choice. More tags mean continuing political battles over standards, *ergo* continuing incompatibilities. A more general solution is the real answer. Enter XML.

XML

So where does XML fit in? XML is not a DTD like HTML. XML does not specify the ultimate tag set to end all tag sets.

XML is like SGML on a severe weight-loss program. As the XML FAQ states, XML is more like SGML−− than HTML++. SGML is an 18-wheeler; XML is a Ferrari. That's enough analogies; you get the idea.

XML is a simplified subset of SGML with the following characteristics.

- It greatly increases the (potential) power and variety of Web (and other) documents.
- It disentangles evolution from the standardization process.
- It "democratizes" the DTD creation process. If you need a specialized capability, create a DTD for it, and write some specialized code to handle it. You needn't be afraid that you'll break someone else's browser or plug-in.
- It is extensible. The earliest extensions concern themselves with more varied hyper-linking and better link man-

LISTING 1—PRESENTATION TAGS

```
<pre>
  <mrow>
    <mrow>
      <msup>
        <mi>x</mi>
        <mn>2</mn>
      </msup>
      <mo>+</mo>
      <mrow>
        <mn>4</mn>
        <mo>&invisibletimes;</mo>
        <mi>x</mi>
      </mrow>
      <mo>+</mo>
      <mn>4</mn>
    </mrow>
    <mo>=</mo>
    <mn>0</mn>
  </mrow>
</pre>
```

LISTING 2—CONTENT TAGS

```
<pre>
  <apply>
    <plus/>
    <apply>
      <power/>
      <ci>x</ci>
      <cn>2</cn>
    </apply>
    <apply>
      <times/>
      <cn>4</cn>
      <ci>x</ci>
    </apply>
    <cn>4</cn>
  </apply>
</pre>
```

agement (using XLL, a developing Extended Link Language), and with portable style sheets (using XSL, a developing Extensible Stylesheet Language).

• Code to write XML parsers and processors can be much simpler than corresponding SGML code, which greatly lowers the barrier to entry for smaller companies and specialized applications. It also gives established SGML tool vendors a leg up on the competition as XML is a strict subset of SGML.

In a sense, it can be useful to think of XML as a portable, self-documenting database format. Any XML processor can read any conforming XML document and be able to decode the schema or structure of that document. It may not be able to do anything with that structure, but it will be able to understand it.

Continuing the database analogy, the DTD is really the schema. The DTD can be contained within the document, or referenced externally. An HTML DTD reference usually looks something like this, <!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 3.2//EN">, and appears as the first line of an HTML file.

On the other hand, it is not required that an XML document have a DTD. In that case, it has an *implicit* DTD, which is simply the structure defined by the tags in the document. If it does have a DTD, the DTD may be contained within the document or externally referenced (or both).

By the way, the XML spec uses two terms that seem to confuse lots of people: *valid* and *well-formed*. A well-formed XML document basically has properly formed and nested tags. A valid XML document is simply a well-formed document that has an explicit DTD. An invalid document is not necessarily *incorrect*; it just doesn't have an explicit DTD.

What can you do with XML?

The answer is easy; the ramifications subtle. Using XML you can create your own specialized markup languages to handle your own special needs, without worrying about breaking the other guy's stuff. Does that mean Microsoft and Netscape can achieve a lasting peace? In and of itself, not really.

An example can help flesh out the issues. One of the first applications of XML is a language called MathML. Like XML itself, MathML is a product of the World Wide Web Consortium; as such it is vendor independent. As of this writing, MathML is not an official standard, but it's close.

At the most basic level, the purpose of MathML is to provide a standard means of embedding mathematical expressions in Web pages so that scientists and engineers can communicate without converting everything to GIF images.

GIFs are bad; they take up space and use bandwidth, but even worse, the conversion process discards information. MathML provides a means of specifying presentation, without discarding the semantic content, *i.e.*, the *meaning* of the expression. That in turn provides maintainability, reusability, cut-and-pasteability, and other benefits.

In that sense, MathML is "two, two, two mints in one." The MathML DTD in fact defines two orthogonal sets of tags, one specifying presentation, the other specifying content. Listing 1 shows the presentation tags, and Listing 2 shows the content tags for this equation: x^2 + 4x + 4 = 0. Both sets of tags would be contained in a single document.

Except in very simple cases, people will not edit those tags by hand. Graphical equation editors will provide a user interface; behind the scenes, MathML gets generated, stored, transferred, and possibly converted, say to PostScript or TeX for hardcopy, or to Braille, or to audio format for hearing-impaired users. (Perhaps a new market for equations on tape will emerge.)

All those output conversions are both possible and practical because of the robust, unambiguous nature of MathML.

Getting there didn't happen overnight. Over the years, there have been several attempts at producing something that could do what MathML can (or will shortly be able to) do. Those attempts provided fodder for the current activities. These are the organizations officially listed as participating in development of the current spec: Adobe, the American Mathematical Society, Design Science, Elsevier Science, The Geometry Center at the University of Minnesota, HP, IBM, INRIA, SoftQuad, Waterloo Maple Inc., and Wolfram.

That's some pretty distinguished company. Does this mean we'll be able to take MathML models out of Mathematica and plug them directly into Maple? That would be nice. The companies would distinguish the products by the quality of their editors and rendering engines.

But what happens when Wolfram wants to implement something that's not covered by the spec? Does the company have the reserve to wait for standardization to occur, or does it create its own extension and forge ahead?

Conclusions

So, what *can* you do with XML? You could define your own markup DTD. Then, to do anything with it, you would have to create your own edit engine, and your own rendering engine. That's just great if your XML dialect is something nobody else really has much of an interest in.

But what if there are interested parties? Say, for example, that you want to create an XML dialect for electronics symbols. **Electronics Now** could create its own for publication in this magazine, but what about all the other magazines? What about companies that design and build electronic devices? What about

(Continued on page 30)

continued from page 26

the educational establishment?

The point is that we could quickly end up embroiled in a standards war. And it's really the same kind of war as the Microsoft/Netscape browser wars. The issue is who controls the DTD.

Despite the hype, I think XML is a valuable technology. I don't think it's going to be a user-level panacea, because there are still too many opportunities for market wars to take place.

XML's value is more infrastructural. It's like a very low-level API that everyone can write to. Even if (when) differences emerge, the facilities of the environment itself provide a means for working out those differences.

I sure hope so, anyway.

That's all the room we have for now, so we'll see you next time. Until then, you can stay in touch via e-mail at jeff@ingeninc.com. EN