Standards and choice

Ben Klemens

22 May 2009

[Part five of six]

The World Wide Web consortium (the W3C) maintains the standards for what is a valid web page, and they provide a validator for web authors to use to check the validity of our own pages, at `http://validator.w3.org`.

Most authors could care less about validation. They figure that if it looks OK on the browser they're using, and maybe one other, then they're done. For example, try validating the home page of the World Bank (265 errors).[1]

Even as esteemed an organization as the Library of Congress (whose front page validates perfectly) has considered building web pages that violate standards to the point of only working in one brand of browser, but at least they were polite enough to float the possibility with a request for comments first. Tim Berners-Lee, the author of the original HTML standard and frequently credited as the founder of the Internet, submitted a comment that explained the importance of documents written around standards instead of programs:

> At the outset, we would like to stress that nothing in this letter should be construed as a criticism of Microsoft's Internet Explorer [...]. We would write the same letter if the choice was to offer support solely for Mozilla Firefox, Safari, or any other product. [...]
>
> While a large proportion of the marketplace uses the Microsoft Internet Explorer to browse the Web, certain classes of users will find it either impossible or extremely inconvenient to do so. [...] Users with disabilities often must augment their browsing software with special assistive software and/or hardware ("assistive technology"). [...] In addition, some individuals with disabilities rely on alternative browsers (for instance, "talking browsers") that are designed to meet their specific needs. Users with disabilities rely on a standards-based Web to ensure that services they access on the Web will be usable through the variety of mainstream software and specialized assistive technologies that they use.

He also points out that when a security flaw is found in a product, people or institutions will often switch to a competitor until the security flaw is patched. That is, even we of decent eyesight would do well to keep a variety of readers on our hard drives (I

[1] Stats are from validation attempts in late 2005. I sincerely hope they do better if you try them today.

use three). This is obviously only possible if a variety of readers can all understand the same document format.

Extending the standards So standards are good. But despite the obviousness of that statement, folks still insist on not complying.

Surely, the most common reason for ignoring a standard is that it does not allow for some form of expression that the author eagerly wants to use. But the author needs to bear in mind that freer expression bears all the costs of broken standards. My favorite Thai restaurant near work, Thaiphoon,[2] has a website that I sometimes check so I can order ahead. When I open with my usual browser, I get a notice that I need to get the Flash plugin to view the site. Since I'm checking from a heavily restricted work computer, I can't install Flash, and often wind up eating at the Chinese place instead. But what happens when I visit the website in a Flash-enabled browser? I get a menu. A plain English text menu.

Or consider the sad state of email. Like a restaurant menu, about 100% of email is also plain text. You tell people things, using words. For about 40 years, there has been a standard (ASCII) that allows different programs to interpret text correctly. Ah, what Nirvana: all the information we need to get across can be gotten across with an easy and supremely well-supported standard. If the UN worked this well, we would have world peace. In fact, now that the computing world is increasingly international, there are more character sets than English-centric ASCII, but nearly every known language is supported by the Unicode standard (yes, Ogham, Ugaritic, Deseret, and Limbu are in there.) Yet people increasingly throw the standard out and encode the text into a word processor document in a proprietary format. If you're lucky, you have a word processor that can read the proprietary-format documents your colleague emailed. For example, if the sender has Word 2000 and the recipient has Word 95, communication won't happen. Putting plain text in a word processor document—even with a bit of extra formatting—is exactly on par with putting a plain old menu in a Flash plugin: yeah, there's a little more glitz, but it comes at the price of potentially excluding, imposing work upon, or alienating the reader.

Of course, word processor documents are nice because they do provide extensions on top of plain text. They let you control the font and layout that the recipient sees in ways that plain text can only approximate. Flash certainly does things that HTML will never even think of supporting. But there is a trade-off that many people ignore, under the presumption that everybody is just like them. "Well, *I* have a copy of Word 2000 and an email client that displays web pages, so everybody else must too. My eyesight and dexterity with mouse and keyboard is fine, so my recipient's must be too." In a social context, the presumption that everybody is like you is the source of a great deal of impoliteness, offense, and general unhappiness, and we teach people from early childhood to understand that others are not like them and that they should maintain standards of decorum until they know that the other party is OK with breaking them. Sure, we can wear the risquè t-shirt to work and maybe make some people smile, but we know that such free expression carries a trade-off in the form of a risk of offending some. We should do the same when writing documents: stick to the basic standards

[2]CT & S, NW DC. Try the Panang tofu.

unless we have a reason to do otherwise and we know that the recipient is OK with our new-fangled alternative.

There do exist valid reasons to ignore standards or set out to establish new ones; e.g., the correct response to a spoken "thank you" is "you're welcome", but it is accepted custom to send a "thank you" email but not a "you're welcome" email, because that sort of thing just sort of clutters up the in box. But those who ignore the standards for no reason or for lousy reasons ("I don't have to say thanks—he owed me.") are just rude.

Bringing it back to the subject at hand, Word establishes its own standard, when it doesn't have to. First, users often write a Word document when a simple plain text file will do. An email with no text in the body but a Word attachment with a single paragraph of plain text is a waste in every sense.

Second, there are standards that do approximately everything a Word document does, such as HTML. You can probably think of a few things that you can do in Word that you can't do in HTML. You can also probably live your entire professional life not using them.

Alternative tools Microsoft goes out of its way to make its DOC format opaque, because users are better locked-in if they can only edit their colleagues' documents with Microsoft tools. But I promised you a paper that does not discuss Microsoft's business strategy, but how Word's design hurts your efficiency. The closed-format design means that, by definition, the only way to edit a Word document is in Word.

There are literally hundreds of editors for a LaTeX or HTML document. You can use anything that can read ASCII-formatted files—even including Word. That means that a market has sprung up that eagerly attempts to appease the needs and skills of different users. As above, EMACS and vi are specialized text editors and therefore have dozens of commands to just edit text, but there are hundreds of other text editors that I didn't mention; pick the one that most fits your lifestyle and run with it. For Word documents, you have no choice but to edit them in Word.

On the output end, there are a wide variety of programs that read LaTeX-formatted documents and display them via formats like HTML, PDF, or plain text. Because the file format is open, many people have implemented programs to process LaTeX-marked text to produce interesting new output. I'll have more such options in the sequel.

Meanwhile, the only thing you can do with a Word document is open it in Word. If Word is not to your liking for any reason, you are stuck. If you need to output something besides Word DOC format, you had better hope that Word allows you to do the conversion.[3]

XML The more tech-savvy readers know that the latest version of Word uses the extensible markup language (XML), which is a commonly-accepted standard for semantic markup. However, this is slightly misleading. First, there is not yet a mechanism

[3] Yes, many people try eagerly to write Word-document compatible extensions, with varying success. But the market for such extensions is absolutely miniscule compared to the market around plain text.

OpenOffice.org will save DOC files as PDFs, by the way. Even if you are married to Word, you may want to download OpenOffice.org and keep it around exclusively as a PDF converter.

to write your own style sheets as I described above. Markup like this is valid XML, but it's just an elaborate way to say boldface. That is, Word takes a system designed for semantic markup and uses it for literal markup.

But more importantly, using the XML standard does not yet mean easy interoperability. XML is a format for writing down data in a tree structure using plain text, so that it can be easily parsed by readers in any system. XML parsers are common in most coding languages, your browser, Word, and many telephones. But once you've got the XML tree read in, what can you do with it?

An XML file depends on a companion document type definition (DTD) file that explains what headings and types and modifiers are available. There are many, depending on your purpose. An address book will define structures for people and organizations, while XHTML defines headers and tables. Two XML systems that read different DTDs are, in the end, incompatible. If one system marks paragraphs with <p> and another with <par>, then the two won't be able to do anything with each other's data, even though parsing the XML structure will be a non-issue.

Word's XML is Word-specific. Politics: although there exist open DTDs for text documents, including DocBook and OpenDoc, Microsoft is insisting on supporting one and only one XML schema: its own. It has applied for patents on that schema in the U.S. and Europe, and although it has stated that it will allow others to use its soon-to-be-patented technology for free, many are wary of whether the format will remain open.

So Word's XML is a near-miss: it solves the problem of parsing the bits on the hard drive in a standard manner, but it doesn't take advantage of the possibility of semantic markup, or of using any of the myriad existing formats that are well-supported by others.

I've argued for the value of decoupling the interface from the document, so that if you don't like how Word does its thing, you can use another tool to edit your document, and then send your finished product to a Word-using colleague who doesn't care what you used to produce the document. But for Word's format such options are limited. There are many tools that will do certain limited operations; there are a handful of competing word processors that try to look like Word, which get within spitting distance of fully supporting Word documents; and that's about it for handling Word's format. I am not aware of a single non-Microsoft product that claims 100% compatibility with Word's XML format.

So, as long as you're using Word's format, you're more-or-less stuck using Word. So next time I'll present some cleaner breaks that use recognized standards.