Mako Hill

Stephen Harris

Origins of Reading

19 April 2002

# Procedural and Descriptive Mark-Up: DocBook and HTML

A printed page contains much more information than the words and characters represented on its surface. The difference between Adobe PageMaker or Microsoft Word, programs which costs hundreds of dollars and take up many megabytes of hard disk space, and Notepad, which Microsoft gives away for free and which only takes up a few kilobytes, is that the first two attempt to give users the ability to represent, control, and store more of this non-textual–or *meta*–data. This data, which includes typefaces, margins, colors, spacing, font sizes and much more, is often referred to "mark-up[1]." While Microsoft Word is certainly more expensive and powerful than Notepad, it is also much more complex and more difficult to learn and use. In a white paper released by the Abortext group, the authors state that, "in conventional word processing and desktop publishing systems, your authors spend up to 30% of their time searching for information, and another 30% of their time applying styles and squeezing paragraphs so that each printed page looks nice" (Abortext). Word processors like LyX try to tackle the problem of unnecessary complexity by reducing users' ability to micro-manage markup in their documents. They encourage users to classify information by *type* and then let the word processor handle the translation onto the printed page through a *descriptive*, as opposed to *procedural,* approach to markup.

However, LyX–like Word, PageMaker or any "What You See Is What You Get" (WYSIWYG) editor–classifies data in terms of how it will appear *on printed pages.* It provides descriptive shortcuts to macros, or memorized lists, of procedural markup manipulations.

[1] *Markup* is a term that originally referred to notes scrawled in manuscript margins by designers for the typesetters. These notes described the typefaces, font sizes, and other design information for the finished printed pages. In the technical sense, *markup* simply refers to this body of *metadata.*

This allows LyX to remain accessible to people accustomed to WYSIWYG interfaces who *do care* how their document looks while providing a much more data-oriented (and consequently more simple) interface for its manipulation. However, it falls short of realizing the possibilities of a system that more purely classifies text as data. In theory, text in a purely descriptive markup system exist *as data* and irrespective of a single material form. In just one example, problems of accessibility and transportability created by a text's existence in a single medium vanish as a single text, mediated through any number of different renderings, can be quickly produced in the newest digital or printed form. In papers classified as data, the text itself, as opposed to just the markup, can be manipulated in relation to its medium. This type of flexibility is the precise goal of projects like the Standard Generalized Markup Language (SGML).

Although SGML may be unknown to most, it acts a superset of a large number of markup languages which include the Hyper Text Markup Language (HTML), the format used for the bulk of data on the World Wide Web. In SGML source code, markup instructions are enclosed in greater-than and less-than brackets which are read as descriptive instructions or guidelines–exactly like a designer's markup instructions. In HTML, an author uses bold markup instructions, or *tags,* to create emphasis: the string "this is <b>bold text</b>" in HTML will be rendered as "this is **bold text"** in a viewer's web browser. Even complex pages are done using a very simple set of text markup instructions.[2] Notices like, "best viewed with Internet Explorer 4.0" refer to the fact that while no web browser, or even web browser *version*, renders webpages exactly as any other, subtle differences in how each browser will render HTML often result in huge difference in how web pages look. Web designers will quickly point to this as one of the greatest problems with HTML. These problems however lies in their fundamental misuse of the standard.

[2]HTML was not originally designed to be used in the ways it is being used today. Precise layout is achieved by using (some might argue by misusing) borderless tables, Cascading Style Sheets (CSS), and other creative uses and additions to the standard to help achieve precise layout across platforms and web browsers.

HTML, as a subset of SGML, was designed as a way of classifying data–a type of simple text database. When an author uses a <table> tag, she is not referring to a *specific* table or one of a certain size, shape, or border. She can expect that that each <th> tag would contain a table header and each <td> would contain a table cell; but that is all she can safely assume. Viewed on a monitor, these tables might be displayed using colors or contrasts designed to increase readability on a computer screen. As rendering software improved, these tables might look better or more stylized; they might look as professional or minimalistic as necessary at any given point. HTML's inventors never promised, and in fact specifically avoided, one standard way to render an HTML document (or hypertext). In fact, it is amazing that HTML has been reigned in and standardized as effectively as it has.

However, while HTML provides a poor typesetting and layout language, the standard's attempts to grant concessions to standardization sacrifice much of its flexibility as a data classification method. While program like *lynx* and *links* demonstrate some of HTML's flexibility, many tags correspond directly to procedural markup methods (<b> means bold, <font size=15> means a 15 pt. font) and much of this information can not be represented when HTML is removed from a full-featured graphical web browser.[3] HTML is also extremely literal; text typed between tags in a HTML document is *always* displayed in exactly the order it is entered in the document.

While HTML rarely takes advantage of SGML's power as a data classification method, DocBook, another popular subset of SGML (DTD), is specifically designed to do exactly that. DocBook is used largely in the production of technical documentation but is useful for producing many other types of literature. While superficially DocBook looks like HTML, its tags are different in that they are almost purely descriptive. To emphasize a phrase or word in HTML, authors use the <i> (italics) tag; in DocBook they enclose the text in <emphasis> tags instead. If an author were to mention the book A History of Reading

[3] *lynx* and *links* are purely text-based web browsers. They each can be run to turn HTML into pure text formats although much of the formatting is simply tossed out: the <u> (underline), <i> (italics), <b> (bold) are some simple examples.

in an HTML document, they would enclose it in the <u> (underline tags); in DocBook they would substitute <citetitle>. However, <emphasis> and <citetitle> are not merely DocBook equivalents of <i> and <u>. DocBook is specifically designed so that it can be rendered into a number of different formats and mediums (including HTML).[4]. Rendered into HTML, these tags might correspond to each other. However, rendered into plain text format, <emphasis> may show up as "*text*" and Citetitle as "_A History of Reading_," or any number of possibilities.

DocBook is more flexible than HTML because it parses document source as raw data instead of "cooked," or literal, text. For example, <header> tags might include <title>, <subtitle> and <author> tags; the <author> tag might include a <firstname>, <surname>, and <othername> tag. Depending on the instructions given to rendering software (in the form of what are called stylesheets), this data might be displayed in a number of a different ways. The rendering software might be instructed to create a title page according to APA or MLA standards. If the rendering instructions do not call for a title page, it might simple create a heading, or toss the data out completely. If one renders a document in a culture that traditionally puts surnames first, the rendering software might display the author's name differently than if it an audience were Western. However, each of these style and textual changes would be executed with no change to the source text itself. If you have written a book in DocBook and want to publish a chapter on the web or create a plaintext version to quote in an email, no conversion is necessary; simply render it using a different stylesheet.

The power of treating text as data is especially visible in the creation of bibliographies. Authors writing in DocBook create bibliographies by creating and filling a database with their reference's information. Users employ <author>, <title>, <publisher>, and a large number of sub-tags to provide as much information as possible. Then, according to the

[4] *sgmltools*, a free program I use that takes advantage of *OpenJade* (also free software) and can effortlessly render DocBook SGML into two types of HTML, postscript, DVI, Rich Text Format (RTF), plain text, ld2db, $\TeX$, and pdb.

style the user has specified, the DocBook processing software will output an appropriately formatted bibliography.

As the need for documents to exist in both digital and printed form increases, the role for DocBook and other descriptive markup formats expands. The limits of digital formats like PDF that simply reproduce printed pages become apparent quickly. However, DocBook is based on a complex list of tag attributes and an equally complex relationship of parent and children tags which are difficult to remember and use. It is also difficult for some authors to visualize and manipulate the markup in their documents while completely divorced from a representation in a material medium. However, DocBook, still relatively young, has quickly gained widespread popularity and the number of freely available stylesheets has quickly increased. Similarly, tools to make producing DocBook easier have begun to appear as well. New and novel ways of displaying and manipulating DocBook text are created constantly. The power of DocBook is exhibited in the fact that while new formats are created and novel ways to render or manipulate data are forwarded, no change to DocBook documents will be necessary to update them to the new standard.

As an example, the same DocBook source code, available at (http://yukidoke.org/~mako/projects/howt
HOWTO.sgml.gz), renders into the following forms:

- plain text - Available: http://yukidoke.org/~mako/projects/howto/FreeSoftwareProjectManagement
  HOWTO.sgml.gz
- PDF - Available: http://yukidoke.org/~mako/projects/howto/FreeSoftwareProjectManagement-HOWTO.pdf
- HTML - Available: http://www.tldp.org/HOWTO/Software-Proj-Mgmt-HOWTO/index.html

## References

[1] Abortext. SGML: Getting Started, A Guide to SGML (Standard Generalized Markup Language) and
1995. 16 April 2002. Available: <http://www.arbortext.com/data/getting_started_with_SGML/getti

[2] Hill, Benjamin C. LYX: Challenging the Word Processor as Typewriter. 5 April 2002. 16 April 2002. Available: <http://yukidoke.org/~mako/writing/origins/OR-Lyx.pdf>.