



A Quick Introduction to Markup Languages and An Overview of the Braille2000 XML Strategy

by Robert Stepp
Computer Application Specialties Company
October 2001

Introduction

By 1968, engineers had figured out that characters could be represented by numbers and ASCII (American Standard Code for Information Interchange) became an American National Standards Institute registered standard code for the digital representation of text. When the first personal computer was created in 1974, it was built using ASCII code. Now almost all text is stored digitally using the ASCII code.

A few years later, computer output devices became elaborate enough to have selectable fonts and display screens became advanced enough to have multiple colors. Until then, a character was just a character, in one standard shape and one standard color. Now there were millions of variations in characters and an enumeration of all the possibilities just wouldn't do. So the notion of text attributes came into use, for typeface name, for color, for size, for style (e.g., boldness), and for positioning. Now schemes were needed to encode not only the characters but also the attributes of the text. Because of the ease that had developed for manipulating text, now using display screen terminals, it was natural that text attributes would be represented by text. Some text would be just text (the content) and other text would designate attributes of the text (the markup). A syntax was needed to distinguish one from the other. A markup language is just such a syntax.

Markup language: Presentation control

The term "markup language" is used to describe a syntactic strategy by which arbitrary text (the content) can be described by a system of attributes (also given as text), which usually determine how the content text should be displayed (the font, the color, the size, etc.). For simplicity, markup is often just words, such as "font" or "size" followed by values that precisely determine that attribute, such as the name of a font typeface, or its size in points or in some other size units. To keep the markup from being lost in the content, special punctuation is introduced to identify what is markup and what is content.

Note that plain ASCII-coded text provides for the concept of "lines". That is, there are some numbers in the ASCII code definition that indicate certain non-printed conditions such as "end of line". For example, standard ASCII-braille consists of the 64 numbers representing the 64 cells (these numbers happen to be the upper-case two-thirds of the regular printable ASCII definition) as well as some additional numbers for line-ending codes ("Carriage-Return" and "Line-Feed"). Lines

of plain text (no font, color, size, style changes), for print or for braille, need only ASCII without any markup.

HTML

Markup languages of all kinds have been around since the mid 1970's. The most ubiquitous markup language is HTML (Hypertext Markup Language), used on Web pages. If you see HTML text (in your Internet browser, click on Source in the View menu to see the HTML text itself), you will notice the pattern of descriptive words (called tags) which are enclosed in angle brackets, such as <p> which denotes the beginning of a new paragraph. The angle brackets are the punctuation used to separate markup from content. You should immediately ask yourself "what happens if an angle bracket is used in the content part of the file, such as the text of this paragraph?" The answer is that there is an alternate way to designate an angle bracket as content, so that < and > are reserved exclusively as markup punctuation to surround elements of the markup.

One more very special thing happens in markup languages: the line boundaries no longer mean anything. When marked-up text is stored in a file, it is convenient to divide the text into lines of reasonable length, otherwise it is awkward to read and edit the file. But the line boundaries are merely for convenience and are ignored when it comes to painting the text on the screen (according to the markup directives). In a markup language, it is only the markup directives (tags) that determine how the text appears, i.e., tags determine where content lines and paragraphs divide, and what fonts, colors, sizes, and positions are used. For most situations, you should image all the lines in a markup file as being concatenated into one very long single line. And then the tags are processed to arrange the content text for viewing.

Example

The following brief excerpt shows this paragraph and the above centered title in HTML markup code. If you read only the parts in angle brackets, you can tell that there is a centered item and a paragraph. If you read only the parts not in angle brackets, you can read the content. A display system (for example your Internet browser or an HTML editor) displays the text in the way the markup directs. That's all there is to it: ASCII-coded content text, with ASCII-coded markup text set off by a consistent punctuation.

```
<p align="center">Example</p>
<p> The following brief excerpt shows this paragraph and the above centered title
in HTML markup code. If you read only the parts in angle brackets, you can
tell that there is a heading and a paragraph. If you read only the parts not in
angle brackets, you can read the content. A printing or display system (for
example your Internet browser or an HTML editor) displays or prints the text
in the way the markup directs. That's all there is to it: ASCII-coded content
text, with ASCII-coded markup text set off by a consistent punctuation.</p>
```

Note that the way content text is arranged in a markup file has nothing to do with how the text will be displayed. The presentation is controlled solely by tags, and here only the "p" tag is used. "align" is the name of an attribute that further directs how a tag manages text. Tags normally come in pairs, one before and one after the content text that they control. Because of this, nesting of tag-controlled regions is possible. In markup files of this type, the ending tags are always written with a / followed by the same tag spelling as that which started the markup unit.

Presentation versus structure

In marked-up text (such as the HTML example above), most of the tags say something about how some part of the content text should be displayed. For example, "centered" or "at point size 24". These statements say nothing about the purpose or role of the text. For example the markup above indicated that "Example" is centered, but not why it is centered (it is actually a minor heading). Some aspects of print presentation would appear to directly relate to a braille format concept, such as "centered". But this is not totally accurate: in a textbook, centered headings in print may be used in a variety of ways, some of which might best be transcribed in other ways, such as putting minor headings out as cell-5 headings. The presentation concept "centered" does not carry enough meaning. The structure concept "subheading" tells us much more.

Markup tags can convey both presentation and structure. For example, in an HTML file, there is a "title" element that uses the <title> and </title> tags to surround the text that is the title. In HTML the title is administrative data and there is no directive as to what font or color to use nor how large the text should be. Most browsers merely copy the title text to the title bar on the browser's window, and there are standard text attributes for window title bars. Web search engines build indexes of web pages by plucking out the title text (by finding the <title> tags within each HTML file they find as they crawl through all Web pages).

Think now about a textbook. Markup that provides presentation will talk about fonts and text sizes and how the text should be positioned. Markup that provides document structure will talk about units, chapters, subchapters, headings, subheadings, paragraphs, and page numbers, without telling us just how or where those things are located on the printed page.

HTML is mostly a presentation mechanism. So is the very different markup language called "rich text format" and all of the proprietary markup languages of the various word processors such as Word and WordPerfect. XML is a markup language that looks a lot like HTML, but is much more flexible. XML can be used to declare presentation or document structure or both. As proposed for use by publishers, XML will be used with tags that denote textbook structure (in elaborate detail) but not much about how the textbook text is presented on each page. Definitive empirical testing is not yet complete, but it appears that document structure (e.g., heading, subheading) provides better braille formatting cues than does presentation (e.g., 20 point text, centered; 12 point text, left justified, blue). This is because document structure says how the text is used (its role in the whole document) whereas presentation merely describes how it appears, and elements with vastly different roles (and different braille format requirements) may happen to be typeset or positioned in the same ways when in fact they are very different things.

XML

The eXtensible Markup Language is a sibling to HTML and both are children of SGML (Standard General Markup Language). They all use tags set off by angle brackets, but the tag sets (the words that can be used for form tags) are determined in totally different ways. HTML is a standard with predefined tag names. Web browsers are programmed to understand HTML tags implicitly, and to render the text in the prescribed way on the screen. XML has no predefined tags, and thus no built-in purpose, and thus no a priori bias to tags used to describe presentation versus tags used to describe structure. To use XML effectively, one must first define the tags that will be used. The statement of that definition is called a DTD (document type definition).

If you have a DTD (which is itself an XML file, i.e, text), then you can create XML markup files that conform to that DTD. The proposed publishers' XML standard comes with a DTD file that defines the tags that can be used by publishers and how they can be nested to describe the structure of a textbook. A publishers' XML file describes the structure of the textbook because that is how the standards committee defined the DTD that publishers use.

The following example shows an excerpt of text from a history textbook recently encoded in the proposed publishers' XML standard:

```
<level5>
  <h5>The Renaissance spirit.</h5>
  <p>Increased trade and travel made Europeans curious about the wider world. Scholars translated the works of ancient Greeks, Romans, and Arabs. They then made discoveries of their own in fields such as medicine, astronomy, and chemistry. This burst of learning was called the <strong><em>Renaissance</em></strong> (REHN uh sahns), a French word meaning rebirth. It started in the late 1300s and continued until about 1600.</p>
  <p>One invention that helped spread the spirit of the Renaissance was the printing press. It was invented during the mid-1400s by Johannes Gutenberg (GOOT uhn berg) of Germany. Before Gutenberg's invention, monks wrote out books by hand. As a result, only a few copies were available. With the printing press,
  <pagenum page="normal">64</pagenum>
  large numbers of books could be printed at a low cost. As more books became available, more people learned to read. The more people read, the more they learned about the world.</p>
  <sidebar class="HistoryAndYou">
    <hd class="sidebarTitle">History and You</hd>
    <p>During the Renaissance, a new ideal person emerged. To meet the ideal, a person had to master every area of learning and be expert in a wide range of skills Who do you know today who might be considered a "Renaissance person"?</p>
  </sidebar>
</level5>
```

The excerpt is a set of paragraphs in structure level 5. The level relates this part of the book to the whole, i.e., level 5 material is part of level 4 which is part of level 3, etc., denoting subsection, section, and chapter parts of the book. The tag "h5" denotes a subheading of level 5, i.e., a subsusbubsubheading. What? No such thing in braille? Transcribers deal with this all the time: the lower numbered levels of headings (headings, subheadings, subsubheadings) usually all become centered headings in braille, while the higher numbered levels of headings become cell-5 headings in braille. This is a subjective determination, and so when XML is translated to braille, it is the transcriber who will decide how the five or more levels of headings map into the three kinds of braille headings. In the first paragraph there is bold italics text (via for bold and for italics). Notice how the boundaries of the strong and em markup units nest one within the other. This is typical of XML and HTML.

There is a print page turn noted in the middle of the second paragraph.

Close to the second paragraph is a sidebar (text on the page that is not part of the linear flow of the prose). This particular sidebar is of type "HistoryAndYou", a name give to it by the XML encoder-person because this kind of sidebar occurs again and again in the textbook. This particular sidebar instance has a subheading and one paragraph of text. Notice the ending tag for the sidebar. Then this part of the book (a small level 5 chunk) ends. After this in the file will be other level 5 chunks, like this one, and perhaps some level 5 text that has multiple level 6 parts. Eventually all the parts at all levels are completed, closing the last level 1 chunk, ultimately completing the structure of the entire book. The XML tags provide many cues as to what braille format could be used. These cues will be used by an automated translation system, not usually by the transcriber directly.

If we were to look only at the "level" and "h" (heading) tags over the entire book, we would see a nested structure like this:

```
<book>
  <level1 class="unit">
    <h1 class="unit">UNIT 1: A Meeting of Different Worlds</h1>
    ...other level 2 tag units...
    <level2 class="chapter">
      <h2 class="chapter">CHAPTER 3 Europeans Reach the Americas (1000–1650)</h2>
      ...other level 3 tag units...
      <level3 class="section">
        <level4 class="subsection">
          <h4 class="subsection">A Changing World</h4>
          <level5 class="subsection">
            <h5>The Renaissance spirit.</h5>
            ...paragraphs...
          </level5>
          ...other level 5 tag units...
        </level4>
        ...other level 4 tag units...
      </level3>
      ...other level 3 tag units...
    </level2>
    ...other level 2 tag units...
  </level1>
  ...other level 1 tag units...
</book>
```

The indenting in the above markup elements is for clarity of presentation: indenting of markup elements is permitted in XML but means nothing. Paragraphs and page numbers and sidebars can be found throughout the above structure, according to where those elements show in the print copy of the textbook. Notice that nothing tells us exactly what the print page looks like. We are told (in great detail) how the text is subdivided into a hierarchy of structural parts: book, unit, chapter, section, subsection, plus sidebars, how those parts form the whole, and the various categories of parts, such as section, subsection, and sidebar. This is not everything, but it is a good set of cues from which mostly correct braille format may be derived.

Getting braille from XML

A textbook in the proposed publishers' XML format will be like the excerpt above, only fleshed out with thousands of lines of intervening paragraph and sidebar (and other kinds of) text. The idea is that a relatively standard markup translation tool will convert patterns of the publisher's tags into carefully selected patterns in some other markup language, namely a presentation language for braille, a markup language that describes how the text should be arranged on the braille page. The markup language used with Braille2000 is BML (braille markup language, an XML type of markup language that is unique to Braille2000). BML is to Braille2000 what HTML is to a web-browser. The sample text from above would become the following markup in BML:

```
<BRL2000a>
  ...preceding text...
  <p li="5" > ,! ,R5AISS.E _S4<p>
  <p li="3" ro="1">,9CR1S$ TRADE & amp; TRAVEL MADE ,EUROP1NS CUR1S AB ! WID] _W4 ,S*OL&gt;S
  TRANSLAT$ ! "WS ( ANCI5T ,GREEKS1 ,ROMANS1 & amp; ,&gt;ABS4 ,!Y !N MADE 4COV]IES ( _! [N 9
  FIELDS S* Z M$IC9E1 A/RONOMY1 & amp; *EMI/R Y4 ,? BUR/ ( LE&gt;N+ 0 CALLS ! <i> ,R5AISS.E </i>
```

```

7,,REHN UH SAHNS71 A ,FR5* ^W M1N+ REBIR?4 ,X /&gt;T$ 9 ! LATE #ACJJ'S &amp; 3T9U$ UNTIL AB
#AFJJ4</p>
<p li="3" ro="1"> ,O 9V5;N T HELPS SPR1D ! _S (! ,R5AISS.E 0 ! PR9T+ PRESS4 ,X 0 9V5T$ DUR+ ! MID-
#ADJJ'S 0,JOHANNES ,GUT5BJG 7,,GOOT UHN B]G7 ( ,G]_M4 ,2F ,GUT5BJG'S 9V5;N1 MONKS WROTE \
BOOKS 0H&amp;4 ,Z A RESULT1 ONLY A FEW COPIES 7 AVAILA#4 ,)! PR9T+ PRESS1
<a cn="PrintNum">64</a >
L&gt;GE NUMB]S ( BOOKS CD 2 PR9T$ AT A L[ CO/4 ,Z M BOOKS 2CAME AVAILA#1 M P LE&gt;N$
6R1D4 ,! M P R1D1 ! M !Y LE&gt;N$ AB ! _W4</p>
...subsequent test...
</BRL2000a>

```

The translation process for XML files will derive the above from the XML. The markup tags are transformed into BML tags, and the words are changed from print spellings to ASCII-braille. Those of you that think about it will immediately worry about such things as the contractions for gh and ar, which in ASCII-braille are < and > characters. There are some ar contractions in the excerpt above. They are encoded, by standard XML rules, as ">". In general, &name; denotes, by special name, those characters that are reserved for markup punctuation. This includes < (<), > (>) and & (&). Thus the only < > characters above are those that set off markup tags.

Note that in BML the paragraph tags <p> include attributes for indent (li=) and runover (ro=). The level 5 heading has thus become a cell-5 heading. The page number is now encoded as a Braille2000 "printnum" annotation (which makes a yellow page turn line in the braille text). The bold italics has become braille italics, as designated by the <i> tag. This example omits many details. The BML markup language is native to Braille2000 and thus the above text can be processed by Braille2000 to yield braille pages formatted as per the presentation directives in BML. Transcribers will not see the above kind of text, only its effects.

In the XML world, the tool that derives presentation (such as BML) from an XML document is called a "stylesheet" specification. Like the DTD, the stylesheet is itself a document written in XML. There is a stylesheet being built for Braille2000 to transform publishers' XML markup into BML, according to BANA formats rules.

Transcriber Interaction With the Translator

Although built from a number of simple elements, markup language documents, including those done in XML can be rather complex. The goal is to hide from the transcriber as much of the detail as possible. Thus, in the design of the XML mechanisms for Braille2000, no provision has been made to show XML or BML to the transcriber or to provide an XML editing environment. There will need to be some summarizing report by which the transcriber can inspect the hierarchical nature of the book, especially to decide how to handle all the various levels of headings that appear in print.

The transcriber will perform an automatic translation of the XML file, getting formatted braille pages as the result, as outlined above. The generated pages of braille will then be edited by hand (perhaps during proofreading) using WYSIWYG and generalized format replacement techniques (such as changing all the level 4 headings from cell 5 type to centered type, done as one operation to all the braille headings derived from XML statements for that level).