XSL-FO meets the Tower of Babel

Tony Graham
Mentea
13 Kelly’s Bay Beach
Skerries, Co Dublin, Ireland
info@mentea.net
http://www.mentea.net

Version 1.0 – MultilingualWeb Workshop – 15 March 2012
© 2012 Mentea
XSL-FO meets the Tower of Babel

Page 5
I18n in XSL 1.0 Requirements 14
Japanese layout 17
1948 and All That 20
What to do? 23
How hard can it be? 25
Further Information 25
I am here as a member of the W3C Working Group that is developing XSL-FO. XSL-FO is an XML vocabulary for specifying formatting semantics. Since some of you may not be familiar with XSL-FO, I'll start with a history of XSL-FO before getting to the Tower of Babel.

In the beginning, there was the page. And the page was without objects and void, and not a whole lot was upon the face of the page.
XSL said, let there be text, and there was text.
XSL is Fun

And XSL saw the text, and they divided the text from the page. And XSL called the text the region-body, and the space they called the margins. And the region-body and the margins were the first formatting.
And XSL said, let there be blocks in the midst of the text, and let them divide the text from the text. And it was so. And the text with blocks was the second formatting.
And XSL said, let there be regions in the edges of the page to hold static content, and let them be for page numbers and running heads, and it was so.
And XSL made the two page sides, the recto for the right hand side and the verso for the left. And XSL set them in the page masters to apply to all pages. And the recto and the verso were the third formatting.
XSL is Fun


Donec sed odio diam. Etiam molestie leo non mi condictum sed est rutrum auctor lobortis. Proin curae sed dictum

And XSL said, let the blocks bring forth abundantly inline objects and properties that apply to them.

And XSL created fo:character and every inline FO after their kind, and every property after its kind, and it was good.

And the inline FOs and properties were the fourth formatting.


<table>
<thead>
<tr>
<th>Graphics, Tables, Columns</th>
</tr>
</thead>
</table>

**XSL is Fun**

```
Lorem ipsum dolor sit amet, consectetur adipiscing elit. Donec ipsum justo, auctor ut
elementum et, dignissim id arcu. Praecavit odio urna, quis adipiscing justo. Aliquam sit
amet magna augue. Morbi lectus neque, condimentum ut amet mattis vitae, ornare
aliquam velpurus lectus. Donec semper nulla id ante venenatis
lorem. Donec a quam ligula, id dignissim lacus, sed dictum, libero. Curabitur
porttitor odio in ante, vel lacinia nunc eros. Aliquam erat volutpat.
```

```
Donec at sodales diam. Etiam mollis sed lorem nec felis.
```

```
Aliquam augue velit, auctor a dui vitae, faucibus euismod dolor.
```

```
Mentea © 2012 Mentea
```

And XSL said, let users make graphics, and tables, and columns, and side floats and other areas.
And the graphics and tables and columns and side floats were the fifth formatting.
XSL is Fun

Then XSL did errata until XSL 1.1, which added change bars, table markers, indexing FOs and much that was good.

I promised you a history, not the history, of XSL-FO. I’m now going to just slide past the whole temptation and fratricide thing and go straight to the Tower part of the story.
Unlike in the original Tower of Babel, the designers of XSL-FO knew it was coming, and XSL-FO has from the beginning been designed to support multiple writing modes, etc.

This is the “bird’s eye” view of the requirements for XSL 1.0 from 1998 with the i18n section highlighted. As you can see, it’s 8 out of the 21 pages.

Writing modes

XSL-FO has always been good at the “big picture” of multilingualism. For example, XSL-FO defines multiple writing modes to cover the requirements of just about every known script.
Writing-mode-neutral names

XSL-FO also has margin-*, border-*, padding-*, and other properties much like CSS, but they are defined in terms of -before, -after, -start, and -end rather than fixed -top, -bottom, -left, and -right. XSL-FO also has the -top, -bottom, -left, and -right properties for compatibility with CSS, but they are defined as mapping to -before, -after, -start, or -end depending on the current writing mode, reference orientation, etc.

Font baselines

Common with CSS

XSL-FO and CSS share the same concepts of different baselines for different styles of text.
In contrast to what you saw from the XSL 1.0 requirements, this is the section on non-Western processing for the 2008 XSL 2.0 requirements.

It’s not that we gave up on i18n, it’s that we recognised that there wasn’t much point repeating what remained from the XSL 1.0 requirements if we still weren’t able to implement them. The other thing that changed since 1998 was that the W3C was working on its first layout task force.
The Requirements for Japanese Layout was based on JIS-X-4051 and on the collected experience of the Task Force members. It’s useful to: authors and designers producing Japanese content; to spec writers who try to capture its recommendations in terms of objects and properties; and to implementers, who try to take care of the rest of the details so you don’t have to. It’s also more accessible than the JIS standard since it’s in English as well as Japanese.

Ruby

- “small-sized, supplementary text attached to a character or a group of characters in the main text”
- Used with:
  - Japanese
  - Chinese + Bopomofo
  - Chinese + Pinyin
  - Other languages

To illustrate the benefit of the Japanese Layout Requirements, I’m going to talk a bit about Ruby, hopefully without stealing too much of Richard and Jirka’s thunder. Ruby is small annotations attached to a base text, and I’m talking about it because it’s used not just with Japanese but with Chinese and bopomofo tone marks, with Chinese with pinyin pronunciations, and, this being the World Wide Web, with other languages as well.
What I know about Japanese Ruby

This is a bird’s eye view of the 40 pages about Ruby in the Japanese Layout Requirements document. If Felix or Richard squint a little, they may even be able to recognise a few of the graphics.

What I know about Bopomofo Ruby

Twelve diagrams like this:

In contrast, this is an excerpt of the only details that I know about formatting bopomofo as Ruby. Neither XSL-FO nor CSS has this level of detail. Bopomofo Ruby is mostly used in children’s textbooks, so it’s a problem if someone wants to produce material for children, a web page for children, or even an EPUB for children.
If you thought I didn't know much about bopomofo as Ruby, this is all I know about Pinyin as Ruby. I can tell you this is a poem about silkworms and candles, but I can't tell you anything about the finer points of formatting it.
1948 and All That

- UDHR adopted 10 December 1948
- Designed to give meaning to “fundamental freedoms” and “human rights” phrases in UN Charter
- Not universally liked
- Guinness World Record as “Most Translated Document”

If you fast-forward an indeterminate number of years to 1948, you come across a Tower of Babel of our own making. The Universal Declaration of Human Rights may have started out as a way to fill in the holes in the UN Charter, but according to the Guinness Book of World Records, it’s been translated into more languages than any other document.
Every written language has its own typographic conventions, and these must be respected if printed material is to appear natural to its intended audience. Such rules include differences in punctuation, spacing, and diacritics; standards for hyphenation and justification; existence or nonexistence of upper- and lowercase letterforms; traditions regarding format; and many other major and minor conventions of typographic styling.

This, of course, makes the UDHR ideal for people doing comparisons between languages or scripts. The oldest of these that I know of is the Typographia Polyglotta, published in 1991 with a second edition in 1997. The 1997 edition has the preamble of the UDHR in 22 languages, including Arabic, Hebrew, Chinese, Japanese, and Korean. To do that in 1991, when Unicode was little more than a good idea, or even in 1997 was an achievement, even if we fully expect our web browsers and even word processors to handle it today.

So it’s interesting that the book includes some comments by its typesetter about the need to adhere to the conventions of the script that you’re formatting.
“UDHR in Unicode”

- Demonstrates the use of Unicode for a wide variety of languages
- See http://www.unicode.org/udhr/index.html
- XML source for nearly 400 languages/scripts
- Over 2,000 pages when formatted

Another one of the uses with which you may be more familiar is the “UDHR in Unicode” project that has the dual purpose of promoting both the UDHR and Unicode. It suits my purposes that its files are available as XML, and when you format them in one go, as some of us like to do, they come out to over 2,000 pages, although different formatters will do better or worse with some of the languages.

UDHR in Khmer

“Nice try, shame about the superscripts and leading”

I was contacted by a guy last year about formatting Khmer, but I was away from the office that week so the best I could do was forward the UDHR in Khmer, though I hadn’t looked at it previously. He emailed back that the lines were too close together and the subscripts and superscripts (as he put it) were wrong. I didn’t hear from him again, so I couldn’t try improving the Khmer formatting and getting another review.
What to do?

- More Task Forces?
- Multilingual Layout Community Group?
- Both?

So now I’ve demonstrated some of the gaps in my knowledge and, by extension, some of the gaps in the ability of current stylesheet languages and current formatters to properly represent non-Western scripts. The next question is what to do about it. The two possibilities that have occurred to me are for the W3C to create more task forces along the lines of the Japanese Layout Task Force or to create a Multilingual Layout Community Group to capture knowledge as it becomes available.

More Task Forces?

- A lot of work by experts
- JLReq influencing XSL-FO and CSS
  - Also useful to authors, designers, and implementers
- TF visible and co-ordinated
- Easier to hold/justify F2F reviews

The Japanese Layout Requirements document represents a mammoth amount of work (that is still going on) by a group of experts. As you would expect, the document is packed with useful information. If the W3C was able to put together more task forces, it would be a major effort to find the right groups of people and major efforts by those people to produce sets of layout requirements. But the stature of the group and the quality of the deliverables can make it easier for others to justify reviewing the work and easier to include it in other standards.
Multilingual Layout CG?

- New facet of W3C
- Per-CG wiki, mailing lists, blog, & IRC channel
- Anyone able to join
- Easy progression to standards track
- Dilute existing i18n mailing lists?
- Could it last beyond initial enthusiasm?

Community and business groups are a new venture for the W3C where there’s a very low overhead for creating a community of interest around any particular topic. A Multilingual Layout Community Group could be a crowd-sourcing of the task of collecting information about best practices for laying out different scripts, but if it happened it would come with very real concerns about its effect on other, existing W3C i18n efforts and about whether it would start with a lot of enthusiasm but not achieve much.

Task Forces and Community Group?

- The cathedral and the bizarre?
- Task Forces for major efforts
- CG wiki accretes information about lesser-known traditions
- CG as incubator for Task Forceable efforts
- Contributor Agreement makes it easy to use CG output

Then there’s the possible middle ground of task forces and a Multilingual Layout Community Group co-existing. In the best of all possible worlds, there could be task forces to do the heavy lifting on defined areas and a Community Group as the incubator for potential task forces and the place to collect information about lesser-known scripts, but that best of all possible worlds assumes that the W3C can form new task forces and that it’s possible to have a viable Community Group.
How hard can it be?

The Plantin Polyglot Bible was published just up the road in Antwerp between 1568 and 1573. It contains five languages, so they handled the Tower of Babel five times. However, we just need to solve it once. How hard can it be?

Further Information

- Extensible Stylesheet Language (XSL) 2.0
  http://www.w3.org/TR/xslfo20
- Extensible Stylesheet Language (XSL) Requirements Version 2.0
  http://www.w3.org/TR/xslfo20-req/
- XSL Requirements Summary
  http://www.w3.org/TR/WD-XSLReq
- Requirements for Japanese Text Layout
  http://www.w3.org/TR/jlreq/
- Summary of Japanese Layout Note
  http://www.tcworld.info/index.php?id=64
UDHR References

- UDHR in Unicode
  http://www.unicode.org/udhr/
- Wikipedia on UDHR
  Universal_Declaration_of_Human_Rights
- UDHR record
  http://www.guinnessworldrecords.com/records-1000/most-
  translated-document/

Credits

- slide 16
  http://www.user.uni-hannover.de/nhtcapri/ruby-annotation.html
- slide 26
  http://www.w3.org/community/
- slide 28
  http://en.wikipedia.org/wiki/
  File:Heritage_day_2011_gnangarra-68.jpg
  Photograph by Gnangarra, Creative Commons Attribution 2.5 Australia licence