Formatting JATS

as easy as 1-2-3

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

Version 1.0 – 2 April 2014
© 2014 Mentea
All rights reserved.
Formatting JATS

as easy as 1-2-3

JATS Preview stylesheets 5
Aside: GitHub 11
XSLT 1.0: Government body 11
XSLT 2.0: PLOS ONE 15
XSLT 3.0: xslt3testbed 21
References 24
Appendix A – About 25
Formatting JATS: as easy as 1-2-3

- JATS Preview stylesheets
- XSLT 1.0
- XSLT 2.0
- XSLT 3.0

The 1-2-3 comes from using JATS with three versions of XSLT.

JATS Preview stylesheets

https://github.com/NCBITools/JATSPreviewStylesheets

- XSLT 1.0
- Public domain
- No copyright issues
- Developed for NCBI by Mulberry Technologies

JATS Preview with “selfie”

Paper for this talk formatted using JATS Preview stylesheet with picture of paper formatted using JATS preview stylesheets.

(’selfie’ was added to the OED in 2013 so maybe it doesn’t need quotes)
Reconstructed timeline

Reconstructed from comments in code, downloads, and emails with Kim Tryka and Tommie Usdin.

Why still XSLT 1.0 in 2012?

- XSLT 1.0 still dominant on some platforms
  - .NET
  - Linux/Unix
- Also tested with XSLT 2.0
- NLM stylesheets developed circa 2006/2007
  - One well-known XSLT 2.0 processor
  - Java only

What does it do?

- Preprocessing
  - Convert OASIS tables to HTML tables
  - Massage citation format
  - Some require XSLT 2.0
- Formatting
  - XML to HTML
  - XML to XSL-FO for formatting as PDF
- Post-processing
  - HTML to XHTML for MathML

The only part that I’ve needed to use, and the only part being covered, is the transformation to XSL-FO and formatting to PDF.
**Customizability**

“These stylesheets are provided as a point of entry for JATS users who may not have the resources to create them from scratch. Because there are many varied implementations of JATS, you should have no expectation that these stylesheets will create production ready files in any arbitrary system. Instead, the stylesheets should be customized for your particular needs.”

“Because we view these stylesheets as a template for a customized solution, not the solution itself, we will accept changes that fix an actual bug, but we will not merge in changes that we view as “customization”. For example, we will accept changes that fix a problem which otherwise leads to failure in creating a final output file, but we will not accept changes that focus on presentational aspects of the final output (such as font changes, margin changes, graphics sizing, etc).”

---

**XSLT features supporting customizability**

- Templates
- Modular stylesheets
- Named attribute sets

---

**Templates**

- `match` matches a context in source XML
- Content of `xsl:template` instantiated when template is applied

```xml
<xsl:template match="td">
  <fo:table-cell xsl:use-attribute-sets="td">
    <xsl:call-template name="process-table-cell"/>
  </fo:table-cell>
</xsl:template>
```

Elements in the body of the template not in the XSLT namespace are copied to the result, and elements and attributes in the XSLT namespace are acted on by the XSLT processor.
Modular stylesheets

```xml
<xsl:include
  href = uri-reference />
```

- `href` refers to other stylesheet
- Children of other `xsl:stylesheet` replace `xsl:include`

```xml
<xsl:import
  href = uri-reference />
```

- `href` refers to other stylesheet
- Imported definitions and template rules not part of importing stylesheet
- Have lower import precedence

Imports in JATS XSL-FO preview

```
jats-xslfo.xsl  →  xhtml-tables-fo.xsl
```

There are more interesting block diagrams later.

Overriding templates

- Template in importing stylesheet overrides same context in imported
- Good when overriding complete function of template
- Extra overhead if you just want to change one little thing
Attribute sets

- Named set of attribute definitions
- Use in multiple places
- Definitions evaluated in each context where used

```xml
<xsl:attribute-set name="fig">
  <xsl:attribute name="keep-together.within-page">
    always
  </xsl:attribute>
  <xsl:attribute name="id">
    <xsl:value-of select="generate-id()" />
  </xsl:attribute>
</xsl:attribute-set>
```

Since attribute definitions in attribute sets are evaluated each time the attribute set is used, the value of the id attribute will be unique to each context.

JATS Preview supporting customizability

- Global variables
- Attribute sets
- Named templates
Example customization

- Add to attribute set from JATS stylesheets

```xml
<xsl:attribute-set name="td">
  <xsl:attribute name="line-stacking-strategy">max-height</xsl:attribute>
</xsl:attribute-set>
```

- New attribute set reusing merged td attribute set

```xml
<xsl:attribute-set name="td-small" use-attribute-sets="td">
  <xsl:attribute name="line-height">10pt</xsl:attribute>
  <xsl:attribute name="border">none</xsl:attribute>
  <xsl:attribute name="padding-top">0pt</xsl:attribute>
  <xsl:attribute name="padding-bottom">0pt</xsl:attribute>
</xsl:attribute-set>
```

- Override JATS stylesheets in more-specific context

```xml
<xsl:template match="td[ancestor::table[@style = 'small']]">
  <fo:table-cell xsl:use-attribute-sets="td-small">
    <xsl:call-template name="process-table-cell"/>
  </fo:table-cell>
</xsl:template>
```

The `xsl:attribute-set` extends the 'td' defined in the JATS Preview stylesheet.
The new 'td-small' attribute set includes the attribute definitions from all declarations for the 'td' attribute set plus the definitions contained in its definition.
The template matches on a more-specific context than the general-purpose template for td in the JATS Preview stylesheets, so in those particular contexts, the XSLT processor uses this template, which adds a different set of attributes to the generated `fo:table-cell` but which still uses the 'process-table-cell' named template from the JATS Preview stylesheets as is used in the original template for td.
This illustrates in a nutshell how a customisation is able to extend, override, and reuse the constructs in the core JATS Preview stylesheets.

Summary: JATS Preview

- XSLT 1.0
- Not accepting customisations into core
- Stylesheet structure facilitates customisations
Aside: GitHub

- "World’s largest open source community"
- Git distributed version control system
- Easy to “fork” – make your own version of projects
- Easy to “pull” merge requests from other projects

XSLT 1.0: Government body

The paper for this talk formatted using XSLT 1.0 stylesheets

© 2014 Mentea
**Project details**

- Source: variation on JATS Blue with custom metadata
- Result: similar page design to JATS preview stylesheets
- XSLT 1.0 because...
  - Client preference
  - Body and back content unchanged from JATS
  - Page design similar to JATS preview
- Customisation...
  - Changes in new modules
  - Import JATS Preview stylesheets

**Import structure**

```
project.xsl  -->  project-mathml.xsl

               ↓
format-mathml.xsl

               ↓
project-xslfo.xsl

               ↓
jats-xslfo.xsl  -->  xhtml-tables-fo.xsl
```

```
xsl:include
__________
xsl:import
```
MathML fix-up modules

- Separate modules that can be dropped when problems solved
- project-mathml.xsl – add parentheses around display equation number
- format-mathml.xsl – workaround too-high accented characters

\[
\frac{SE(\hat{\rho})}{\hat{\rho}} > .175 \text{ when } \hat{\rho} \leq .5
\]

becomes

\[
\frac{SE(\hat{\rho})}{\hat{\rho}} > .175 \text{ when } \hat{\rho} \leq .5
\]

(Latest formatter has rewritten MathML support)
**What’s in `project-xslfo.xsl`?**

- Initial comments: 7%
- Variable & attribute set overrides: 18%
- New attribute sets: 5%
- Top-level element templates: 15%
- Cover page templates: 15%
- PDF bookmarks: 5%
- Lower-level element templates: 35%
- Initial comments: 7%

**Summary: XSLT 1.0**

- Customisation on top of JATS Preview stylesheets
- Preview stylesheets provided sufficient hooks
Sample PLOS ONE pages.

**Project details**

- Peer-reviewed, open-access, online publication
- Public Library of Science
- JATS/NLM markup
- Lights-out batch formatting with XSL-FO
- Previously produced use 3B2 and (presumably) manual fix-up
- XSLT 2.0 because...
  - Big differences in metadata, figure, table handling
  - Needed vendor extensions
- Customisation...
  - Modified version of jats-xslfo.xsl
  - Additional XSLT modules
**PONE “features”**

- Figures and tables float to top (or bottom) of page
- Figures column-wide or page-wide
  - No size information in XML
- Figure graphic+caption can’t overflow page
- Tables column-wide, page-wide, or page-high
  - Page-high may be single column
  - May be multiple pages
  - No width indication in XML
  - No row spanning (thank goodness!)
  - No figures or tables allowed after start of back matter

**XSLT/XSL-FO “features”**

- Page-wide floats
  - Vendor extension for column-wide
- Floats don’t break
- Floats only at top of page
  - Bottom-float extension available but unused
- Graphic size not available to XSLT
- *Fire-and-forget* processing

**Table handling**

- “Pre-format” tables in three widths on long pages
  - Column-wide, page-wide, (width of) page-high
  - Prefix table IDs with string indicating width
- Format to area tree XML
- Compare area trees for each table
- Use width with least area and no overflow
- Recreate as multiple `fo:float` if overflows page
  - Re-use table column widths from area tree to remain consistent
Picking “Best” Tables

Three tables formatted in each of three widths, with preferred versions highlighted.

Sized and placed tables

Column-wide and page-wide tables placed on pages.
Usual processing model

The conventional XSLT–XSL-FO processing model.

Table-handling processing model

The processing model including preprocessing tables to generate an area tree from which to determine the preferred width for each table.

Graphics handling

- Get TIFF graphics
- ImageMagick identify gives graphic size and resolution
- “Pre-format” caption at both widths to get exact size
- Choose best width
- (Possibly) scale down graphic so caption also fits on page
Figure-handling processing model

Processing model when graphics handling added.

Floats after back matter

Figures and tables are required to not appear after the start of the back matter.
**Splitting at back matter**

- Format "final" FO with right-width tables and figures to area tree
- Compare positions of first “back” content and last float
  - back plus bits from front, body
- Generate new FO with either one or two `fo:page-sequence`
- If second `fo:sequence`, it contains only back matter so floats in first appear before back matter

**Putting It All Together**

![Diagram of the full processing model](image)

The full processing model.

**Import structure**

```
splitter.xsl → size-chooser.xsl → size-functions.xsl
```

All the top-level stylesheets use `plos-xslfo.xsl` for basic formatting.

`splitter.xsl` does everything `size-chooser.xsl` does, and more, so it imports that file rather than importing `plos-xslfo.xsl` directly.
Summary: XSLT 2.0

- It shouldn’t be this hard
- Column-wide floats require vendor extension
- Navigating area tree isn’t easy
- No standard for area tree XML made it harder and even less portable
- Creating new FO and reprocessing easier than rewriting area tree
- EXPath Binary Module (and a TIFF-handling library!) could avoid using ImageMagick
  - Or use vendor extension

XSLT 3.0: xslt3testbed

https://github.com/MenteaXML/xslt3testbed

- Trying out new XSLT 3.0 features
- Converting existing JATS stylesheets to XSLT 3.0

Why?

“...the design process does not include enough feedback; by the time people start reporting their usability experiences, the decisions are difficult to change.”

- Early start on patterns and idioms to help adoption
- Find infelicities in spec (and implementations)
- The time is right
  - Project started November 2013
  - XSLT 3.0 Last Call WD – 12 December 2013


Motivation comes from looking for a better way to get people using the new version:

- 1997: Wanted to discuss DSSSL so started DSSSLList
- 1998: XSL-List started – people tried every new XSL feature as it came out
- 2004–2007++: People had working XSLT 1.0 systems and there weren’t many XSLT 2.0 processors, so adoption slow
- 2013–2014: Looking for a quicker win than mailing lists, and people now used to working with GitHub projects

W3C Process

- End game for a W3C spec:
  - Last Call
  - Candidate Recommendation
  - Proposed Recommendation
  - Recommendation
- Changes after “Last Call” require more documentation and substantiation
Why JATS?
- Simpler than, e.g., DocBook or TEI
- Not a toy
- Potentially useful to authors and archives
- Existing XSLT stylesheets available

Why JATSPreviewStylesheets?
https://github.com/NCBITools/JATSPreviewStylesheets
- XSLT 1.0
  - Easy for new contributors to add XSLT 2.0-isms
- Public domain
  - No copyright issues
  - XSLT 3.0 stylesheets also public domain
- Explicitly not supporting gazillion customisation parameters, PIs, etc.
  - Simpler processing
  - Fewer user expectations

xslt3testbed goals
- Trial different techniques
- Open for dipping into to try random ideas
- Develop patterns and idioms
- Develop XSLT 3.0 package for XHTML tables
  - xsl:package new in XSLT 3.0
  - XHTML tables used in many document types

xslt3testbed non-goals
- Single best way of doing anything
  - Multiple ways to solve the same problem are okay
- Definitive XSLT 3.0 testbed
  - It’s easy to fork and make your own version
- Complete stylesheet for all of JATS
  - Existing stylesheets don’t cover everything yet either

Results so far
- Trying out maps, anonymous functions, and xsl:iterate
- Small advances in multiple areas
- Both XSL-FO and XHTML stylesheets
- More details in XML Prague 2014 talk
  http://www.mentea.net/resources/xslt30testbed-slides.pdf
6 W3C Bugzilla bug tickets so far

<table>
<thead>
<tr>
<th>ID</th>
<th>Product / Component</th>
<th>Assignee</th>
<th>Status</th>
<th>Resolution</th>
<th>Summary</th>
</tr>
</thead>
<tbody>
<tr>
<td>4641</td>
<td>XPath / XSLT 3.0 melee</td>
<td>GW</td>
<td>New</td>
<td></td>
<td>XPath level element and attribute constructors for use in aggregate functions</td>
</tr>
<tr>
<td>4649</td>
<td>XPath / XPath 3.0, jonathan mike ASSI</td>
<td></td>
<td></td>
<td></td>
<td>XPath doc FunctionBuild production in body of spec</td>
</tr>
<tr>
<td>4640</td>
<td>XPath / XPath 3.0, jonathan mike RESO</td>
<td></td>
<td></td>
<td></td>
<td>XPath/SegueTicket vs TypesDeclarationIn XPath20Query 3.0</td>
</tr>
<tr>
<td>4610</td>
<td>XPath / Function miko RESO</td>
<td></td>
<td></td>
<td></td>
<td>XPath/SegueTicket vs TypesDeclarationIn XPath20Query 3.0</td>
</tr>
<tr>
<td>4609</td>
<td>XPath / XPath 3.0 melee</td>
<td></td>
<td></td>
<td></td>
<td>XPath/SegueTicket vs TypesDeclarationIn XPath20Query 3.0</td>
</tr>
<tr>
<td>4608</td>
<td>XPath / XPath 3.0 melee</td>
<td></td>
<td></td>
<td></td>
<td>XPath/SegueTicket vs TypesDeclarationIn XPath20Query 3.0</td>
</tr>
</tbody>
</table>

5 JATSPreviewStylesheets patches so far

The JATSPreviewStylesheets network graph

All branches in the network using MenteaXML/JATSPreviewStylesheets as the reference point.

Other results

- One XSLT processor bug
- One change to Wendell Piez’s JATS Oxygen plug-in
- Technique for hosting Oxygen plugins on GitHub
Summary: XSLT 3.0

https://github.com/MenteaXML/xslt3testbed

- The time is right
- Useful in multiple arenas
- Results summarised on project wiki and http://inasmuch.as/
- Well suited for trying things out
- Go fork and multiply

Conclusion

- JATS Preview stylesheets:
  - Explicitly don’t support customisation
  - Good basis for your own customization
- Customise by:
  - Layer on top of existing style sheets
  - Modify your copy of the stylesheets
- Usable with XSLT 1.0, 2.0, or 3.0

References

- slide 41 – Micheal Kay
- slide 42 – W3C Process Document
  http://www.w3.org/2005/10/Process-20051014/tr.html
- slide 48 – Bugs so far
  https://www.w3.org/Bugs/Public/buglist.cgi?email1=tgraham%40mentea.net&emailreporter1=1&emailtype1=substring&product=XPath%20%2FXQuery%20%2FXSLT&query_format=advanced
Appendix A
About
Tony Graham 25
Mentea 25

Tony Graham

Tony Graham has been working with markup since 1991, with XML since 1996, and with XSLT/XSL-FO since 1998. He is Chair of the Print and Page Layout Community Group at the W3C and previously an invited expert on the W3C XML Print and Page Layout Working Group (XPPL) defining the XSL-FO specification, as well as an acknowledged expert in XSLT, developer of the open source xmlroff XSL formatter, a committer to both the XSpec and Juxy XSLT testing frameworks, the author of “Unicode: A Primer”, a member of the XML Guild, and a qualified trainer.

Tony’s career in XML and SGML spans Japan, USA, UK, and Ireland, working with data in English, Chinese, Japanese, and Korean, and with academic, automotive, publishing, software, and telecommunications applications. He has also spoken about XML, XSLT, XSL-FO, EPUB, and related technologies to clients and conferences in North America, Europe, and Australia.

Mentea

Mentea specialises in consulting and training in XML, XSL-FO, & XSLT. We are available for on-site meetings and classes, worldwide, but as well as on-site meetings and classes, we routinely keep in touch with clients though email, Skype, instant messaging, and telephone and through a secure, per-client or per-project wiki, revision-control, and issue-tracking system.

Our staff have been working with markup since 1991, with XML since 1996, and with XSLT/XSL-FO since 1998. Based in Dublin, Ireland, Mentea has a global reach: in recent projects, we have helped companies and organisations in the USA, Ireland, England, and France with their XSLT, XSL, and XML, including:

- Writing Schematron for a professional body
- Augmenting a XSLT-based automated schema documentation system that produces both HTML and PDF
- Extending FOP for a software company
- Training in XML, oXygen, DocBook, XSLT 2.0, and XSL-FO
- Formatting JATS to PDF for a scientific journal
- Writing XSLT stylesheets to convert non-XML into XML then into EPUB
- Writing XSLT to convert Excel into XML for a commercial bank

Mentea presents a unique range of skills extending beyond XML and XSL-FO/XSLT into Unicode, SGML, DSSSL, and programming in C, Java, Perl, Lisp, and other languages.

We understand how markup works. Our staff has worked with markup in Japan, USA, UK, and Ireland as user, consultant, and developer, with data in English, French, Chinese, Japanese, and Korean, with academic, automotive, publishing, software, and telecommunications applications, and in the Web Services and document processing arenas.

We are also interested in applying the tools for ensuring software quality – unit testing, code coverage, profiling, and other tools – to XML and XSLT/XSL-FO processing.

Through our associations and affiliations with other consultants around the world, we can call on extra help for large or specialised projects.