XSLT 3.0 Testbed

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

Version 1.0 – 15 February 2014

XSLT 3.0 Testbed

XSLT 3.0 testbed 1

- What?
- How?
- Why?
- Results so far
- Next steps

What? How? 2

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

Why? 3

- 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

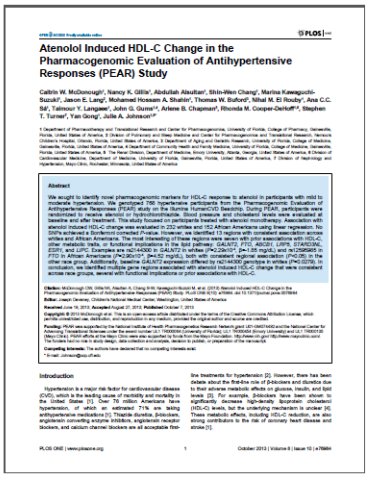
W3C Process 4

- End game for a W3C spec:
  - Last Call
  - Candidate Recommendation
  - Proposed Recommendation
  - Recommendation
- Changes after “Last Call” require more documentation and substantiation

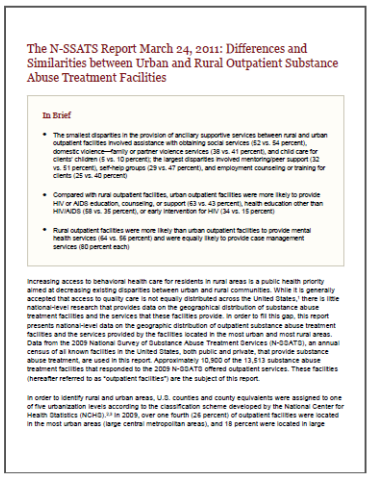
What is JATS? 5

- Journal Article Tag Suite
- Successor to “National Library of Medicine (NLM) Journal Archiving and Interchange Tag Suite”
- Over 2 million articles in PubMed Central (in 2011)
- Used by journal publishers and archives around the world
- Nature announced two Open Access journals in January 2014
- Used by Public Library of Science (PLOS)

JATS Samples 6



Atenolol Induced HDL-C Change in the Pharmacogenomic Evaluation of Antihypertensive Responses (PEAR) Study



The N-SSATS Report March 24, 2011: Differences and Similarities between Urban and Rural Outpatient Substance Abuse Treatment Facilities

In Brief

- The smallest disparities in the provision of ancillary supportive services between rural and urban outpatient facilities involved assistance with obtaining social services (52 vs. 54 percent), domestic violence—family or partner violence services (38 vs. 41 percent), and child care for clients' children (5 vs. 10 percent); the largest disparities involved mentoring/peer support (32 vs. 51 percent), self-help groups (29 vs. 47 percent), and employment counseling or training for clients (25 vs. 40 percent)
- Compared with rural outpatient facilities, urban outpatient facilities were more likely to provide HIV or AIDS education, counseling, or support (63 vs. 43 percent), health education other than HIV/AIDS (68 vs. 36 percent), or early intervention for HIV (34 vs. 15 percent)
- Rural outpatient facilities were more likely than urban outpatient facilities to provide mental health services (64 vs. 56 percent) and were equally likely to provide case management services (60 percent each)

Why JATS? 7

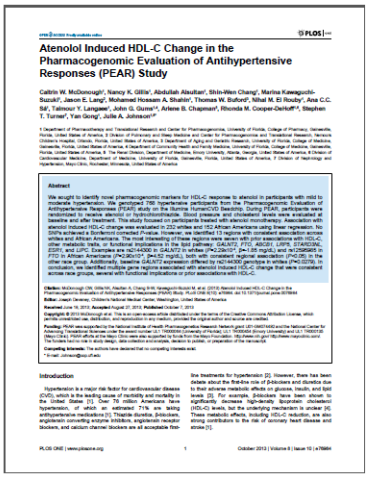
- Simpler than, e.g., DocBook or TEI
- Not a toy
- Potentially useful to authors and archives
- Existing XSLT stylesheets available

JATSPreviewStylesheets 8

`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

Explicitly not supporting customisation 9


Atenolol Induced HDL-C Change in the Pharmacogenomic Evaluation of Antihypertensive Responses (PEAR) Study


Atenolol Induced HDL-C Change in the Pharmacogenomic Evaluation of Antihypertensive Responses (PEAR) Study

xslt3testbed goals 10

- Trial different techniques
- 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

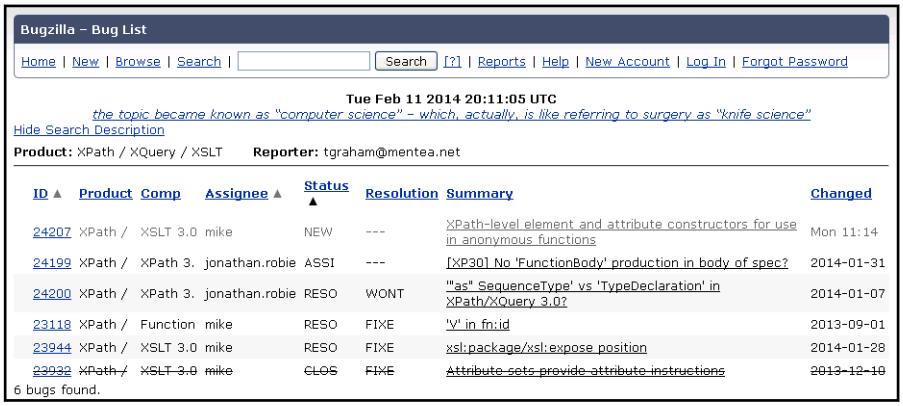
xslt3testbed non-goals 11

- 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 12

- Trying out maps, anonymous functions, and `xsl:iterate`
- Small advances in multiple areas
- Not expecting to find a meteorite

6 W3C Bugzilla bu^H^Htickets so far 13



Bugzilla – Bug List

Home | New | Browse | Search | Search | [?] | Reports | Help | New Account | Log In | Forgot Password

Tue Feb 11 2014 20:11:05 UTC

the topic became known as "computer science" – which, actually, is like referring to surgery as "knife science"

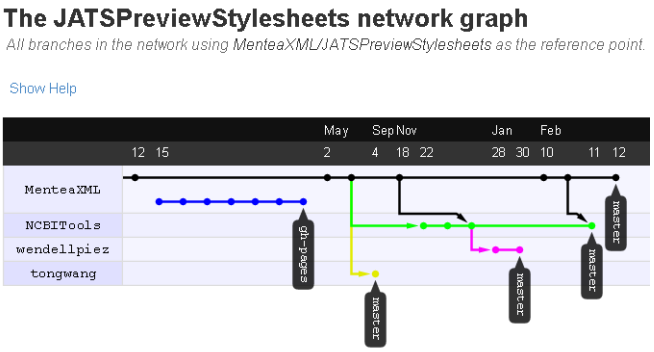
Hide Search Description

Product: XPath / XQuery / XSLT **Reporter:** tgraham@mentea.net

ID ▲	Product	Comp	Assignee ▲	Status ▲	Resolution	Summary	Changed
24207	XPath /	XSLT 3.0	mike	NEW	---	XPath-level element and attribute constructors for use in anonymous functions	Mon 11:14
24199	XPath /	XPath 3.	jonathan.robie	ASSI	---	[XP30] No 'FunctionBody' production in body of spec?	2014-01-31
24200	XPath /	XPath 3.	jonathan.robie	RESO	WONT	"as" SequenceType' vs 'TypeDeclaration' in XPath/XQuery 3.0?	2014-01-07
23118	XPath /	Function	mike	RESO	FIXE	'V' in fn:id	2013-09-01
23944	XPath /	XSLT 3.0	mike	RESO	FIXE	xsl:package/xsl:expose position	2014-01-28
~~23932~~	~~XPath /~~	~~XSLT 3.0~~	~~mike~~	~~CLOS~~	~~FIXE~~	~~Attribute sets provide attribute instructions~~	~~2013-12-10~~

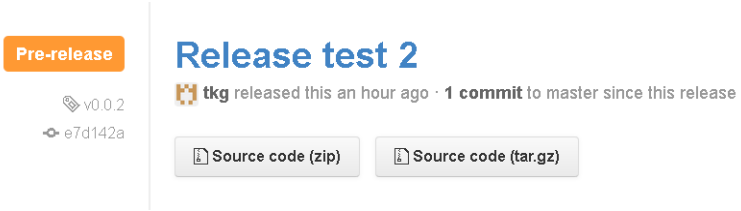
6 bugs found.

4 JATSPreviewStylesheets patches so far 14



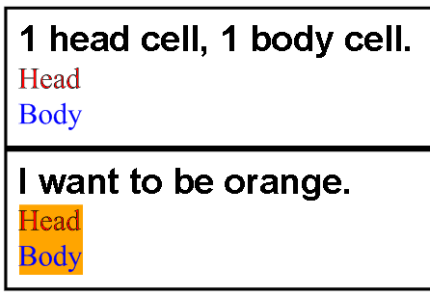
Other results 15

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



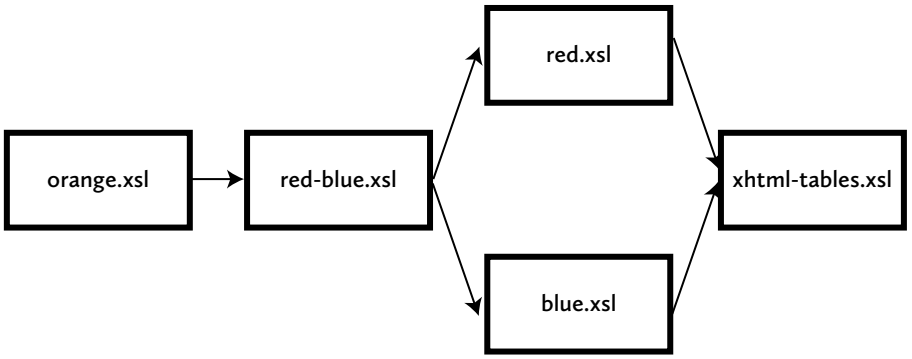
`xsl:attribute-set` vs map of functions 16

- Compare `xsl:attribute-set` to map of functions
- Two tables with red table head text and blue table body text
- One with `<table style="orange">`



Stylesheets 17

- `orange.xsl` – Handles `table[@style eq 'orange']`
- `red-blue.xsl` – Imports others
- `red.xsl` – Red header text
- `blue.xsl` – Blue body text
- `xhtml-tables.xsl` – Base table stylesheet



`xsl:attribute-set` 18

```
<xsl:attribute-set name="thead">
  <xsl:attribute name="color" select="'red'" />
</xsl:attribute-set>
```

- Since XSLT 1.0
- Generate named sets of attributes
- Evaluate `xsl:attribute` declaration in current context
- Multiple `xsl:attribute-set` with same name aggregate
- Refer to with `@use-attribute-sets` or `@xsl:use-attribute-sets`

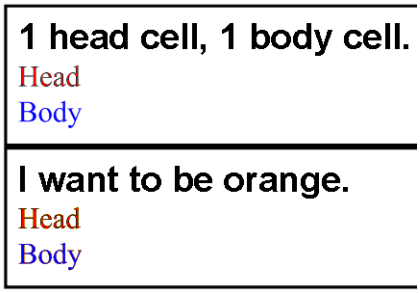
`xsl:attribute-set` has current context 19

```
<xsl:attribute-set name="fig">
  <xsl:attribute name="id" select="generate-id()" />
</xsl:attribute-set>
```

- Different value for every context
- Can also call templates and functions inside `xsl:attribute`

`xsl:attribute-set` and the tables 20

- Attribute set for each element type, e.g., `thead`
- Using *only* those attribute sets, can't do one table with orange style



Map of functions 21

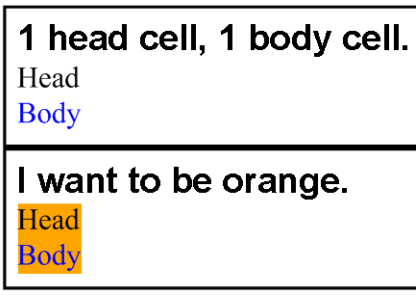
- Element names as key, function to apply as value
- Context passed as function parameter

```
<xsl:template match="thead">
  <xsl:next-match>
    <xsl:with-param
        name="table-functions"
        as="map(xs:string, function(element()) as attribute()*)"
        select="map {
'thead' := function($context as element()) as attribute()* {
x3tb:attribute('color', 'red')
}
}"
        tunnel="yes" />
  </xsl:next-match>
</xsl:template>
```

(`x3tb:attribute()` is XSLT function containing `xsl:attribute`)

Map of functions and the tables 22

- Import precedence and using `xsl:next-match` misses effect from `red.xsl`
- Could work around with this example, but harder in large system
- Possible work with map of sequence of functions or modifying of result of lower-precedence templates

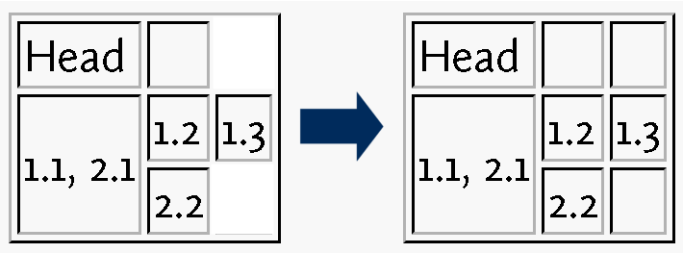


`xsl:attribute-set` vs map of functions 23

- `xsl:attribute-set`:
  - Aggregating `xsl:attribute-set` across modules "just works"
  - Need other mechanism for per-element customisations
  - Can't "turn off" an attribute, only override
- Map of functions:
  - Can do per-element overrides
  - More work for XSLT developer than `xsl:attribute-set`
  - Aggregating across modules and overriding templates needs more thought
  - Map of sequence of functions could be interesting
  - Is it just reinventing `typeswitch`?

XHTML tables and `xsl:iterator` 24

- Insert `<td/>` in gaps in table
- Consider rowspans
- Use `xsl:iterate`



Three column table, some cells missing 25

```
<table border="1">
  <colgroup>
    <col align="left" span="1"/>
    <col align="left" span="1"/>
    <col align="left" span="1"/>
  </colgroup>
  <thead>
    <tr>
      <td rowspan="1" colspan="1">Head</td>
      <td rowspan="1" colspan="1"></td>
      ????
    </tr>
  </thead>
  <tbody>
    <tr>
      <td rowspan="2" colspan="1">1.1, 2.1</td>
      <td rowspan="1" colspan="1">1.2</td>
      <td rowspan="1" colspan="1">1.3</td>
    </tr>
    <tr>
      <td rowspan="1" colspan="1">2.2</td>
      ????
    </tr>
  </tbody>
</table>
```

xsl:iterate 26

- Sequential processing of a sequence
- “many XSLT users find writing recursive functions to be a difficult skill, and this construct promises to be easier to learn”
- “[S]hould be more amenable to optimization”

xsl:iterate 27

```
<xsl:iterate
  select = expression >
  <!-- Content: (xsl:param*, sequence-constructor,
                 xsl:on-completion?) -->
</xsl:iterate>
```

- Parameters just like `xsl:for-each`/`xsl:param`

xsl:on-completion 28

```
<xsl:on-completion
  select? = expression >
  <!-- Content: (sequence-constructor) -->
</xsl:on-completion>
```

- Last within `xsl:iterate`
- Additional (or only) result from `xsl:iterate`

xsl:break, xsl:next-iteration 29

```
<xsl:break
  select? = expression >
  <!-- Content: (sequence-constructor) -->
</xsl:break>
```

- "Last" within xsl:iterate
- Break out of xsl:iterate

```
<xsl:next-iteration>
  <!-- Content: (xsl:with-param*) -->
</xsl:next-iteration>
```

- "Last" within xsl:iterate
- Modify some or all parameters for next iteration
- If no xsl:iterate, parameters not changed

When is "last"? 30

- Last in xsl:iterate
- Last in last:
  - xsl:if
  - xsl:when
  - xsl:otherwise
  - xsl:catch
  - xsl:try
  - I.e., in "tail position" of element in tail position of ...
- So, conditionally set/change parameters in a "last" xsl:if or xsl:choose

Applying xsl:iterate to table 31

- xsl:iterate over <tr> in <thead, <tbody>, <tfoot>
- Inner xsl:iterate over <th> and <td> in <tr>
- Keep map of columns and remaining rowspan for the column
- Update map in inner xsl:iterate for each <th> or <td>
- Emit <td/> if more columns than cells

`xsl:iterate` in action


```
<xsl:template match="thead | tbody | tfoot">
  <xsl:param name="col-map" as="map(xs:integer, element(col))" />
  <xsl:variable name="cell-map"
                select="map:new(for $i in 1 to count(map:keys($col-map))
                                return map:entry($i, 0))"
                as="map(xs:integer, xs:integer)" />
  <xsl:copy>
    <xsl:apply-templates select="@*" mode="table-copy"/>
    <xsl:iterate select="tr">
      <xsl:param name="cell-map" select="$cell-map"
                 as="map(xs:integer, xs:integer)" />
      <xsl:message select="('row', count(map:keys($cell-map)))"/>
      <xsl:variable name="content" as="item()+">
        <xsl:iterate select="1 to count(map:keys($cell-map))">
          <xsl:param name="colnum"
                     select="1"
                     as="xs:integer" />
          <xsl:param name="cell-map"
                     select="$cell-map"
                     as="map(xs:integer, xs:integer)" />
          <xsl:param name="cells" select="*" as="element()*" />

          <xsl:message select="map:get($cell-map, .)"/>
```

Code, continued


```
          <xsl:choose>
            <xsl:when test="map:get($cell-map,.) > 0">
              <xsl:message select="'rowspanned'"/>
              <xsl:next-iteration>
                <xsl:with-param
                    name="colnum" select="$colnum + 1" as="xs:integer" />
                <xsl:with-param
                    name="cell-map"
                    select="map:new(($cell-map,
                                     map:entry(., map:get($cell-map,.) - 1)))"
                    as="map(xs:integer, xs:integer)" />
              </xsl:next-iteration>
            </xsl:when>
            <xsl:when test="empty($cells)">
              <xsl:message select="'no cell'"/>
              <td/>
              <xsl:next-iteration>
                <xsl:with-param name="colnum" select="$colnum + 1"/>
              </xsl:next-iteration>
            </xsl:when>
```

Code, continued

34

```
            <xsl:otherwise>
              <xsl:message select="concat('cell: ''',
                                          $cells[1]),
                                          '''; rowspan: ',
                                          string($cells[1]/@rowspan)"/>
              <xsl:apply-templates select="$cells[1]" />
              <xsl:next-iteration>
                <xsl:with-param
                    name="colnum" select="$colnum + 1" as="xs:integer" />
                <xsl:with-param name="cell-map"
                                select="map:new(($cell-map,
                                                 map:entry(., xs:integer(($cells[1]/@rowspan, 1)
[1]) - 1)))"/>
                <xsl:with-param name="cells" select="$cells[position() > 1]"/>
              </xsl:next-iteration>
            </xsl:otherwise>
          </xsl:choose>
          <xsl:on-completion>
            <xsl:sequence select="$cell-map" />
          </xsl:on-completion>
        </xsl:iterate>
        </xsl:variable>
```

Code, continued

35

```
      <xsl:copy>
        <xsl:apply-templates select="@*" mode="table-copy"/>
        <xsl:sequence select="$content[position() != last()]" />
      </xsl:copy>
      <xsl:next-iteration>
        <xsl:with-param name="cell-map"
                        select="$content[last()]" />
      </xsl:next-iteration>
    </xsl:iterate>
  </xsl:copy>
</xsl:template>
```

`xsl:iterate` issues

36

- Less readable (IMO) than recursive templates or functions
- Returning map in `xsl:on-completion` of inner `xsl:iterate`
  - Buffering result of `<tr>` just to use `last()` to get map
  - Maybe split map calculation and `<tr>` processing or use `xsl:accumulator`?
- Not sure why there's no `xsl:sort` with `xsl:iterate`

Idioms

37

- "manner of expression characteristic of or peculiar to a language"
- Every language, human or computer, has them
- Different in different versions of XSLT

XSLT 1.0 – Meunchian grouping 38

- Developed by Steve Meunch
- Because XSLT 1.0 didn't do grouping ... now forgotten?

```
<xsl:key name="contacts-by-surname"
         match="contact" use="surname" />
<xsl:template match="records">
  <xsl:for-each select="contact[count(. |
                                      key('contacts-by-surname',
                                          surname)[1]) = 1]">
    <xsl:sort select="surname" />
    <xsl:value-of select="surname" />,<br />
    <xsl:for-each select="key('contacts-by-surname',
                              surname)">
      <xsl:sort select="forename" />
      <xsl:value-of select="forename" /> (
      <xsl:value-of select="title" />)<br />
    </xsl:for-each>
  </xsl:for-each>
</xsl:template>
```

XSLT 2.0 – `xsl:choose` shortcuts 39

- XSLT 1.0

```
<fo:table-cell>
  <xsl:attribute name="text-align">
    <xsl:choose>
      <xsl:when test="@align">
        <xsl:value-of select="@align"/>
      </xsl:when>
      <xsl:otherwise>from-table-column()</xsl:otherwise>
    </xsl:choose>
  </xsl:attribute>
</fo:table-cell>
```

- XPath 2.0 `if`

```
<fo:table-cell
    text-align="{if (exists(@align))
                   then @align
                 else 'from-table-column()'}">
```

- XPath 2.0 sequence constructor

```
<fo:table-cell text-align="{(@align, 'from-table-column()')[1]}">
```

XSLT 3.0 – ???? 40

Predicting...

- Common, safe streaming constructs
- Higher-order functions and named function references
  - XSLT 2.0:

```
if (@firstpage eq 0)
  then floor($totalPagesDecimal)
else ceiling($totalPagesDecimal)
```

  - XSLT 3.0 (and validating processor only?):

```
(floor#1,ceiling#1)[@firstpage + 1]($totalPagesDecimal)
```

Conclusion 41

https://github.com/MenteaXML/xslt3testbed

- The time is right
- Useful in multple arenas
- Go fork and multiply

References 42

- slide 4 – W3C Process Document
  http://www.w3.org/2005/10/Process-20051014/tr.html
- slide 5 – NISO Z39.96 The Journal Article Tag Suite (JATS): What Happened to the NLM DTDs?
  http://quod.lib.umich.edu/j/jep/3336451.0014.106/--niso-z3996-the-journal-article-tag-suite-jats-what-happened?rgn=main;view=fulltext
- slide 13 – Bugs so far
  https://www.w3.org/Bugs/Public/buglist.cgi?email1=tgraham%40mentea.net&emailreporter1=1&emailtype1=substring&product=XPath%20%2F%20XQuery%20%2F%20XSLT&query_format=advanced
- slide 18 – xsl:attribute-set
  http://www.w3.org/TR/xslt-30/#attribute-sets

References 43

- slide 26 – xsl:iterate
  http://www.w3.org/TR/xslt-30/#iterate
- slide 37 – idiom
  http://www.thefreedictionary.com/idiom
- slide 40 – Functional XPath commands
  http://x-query.com/pipermail/talk/2014-January/004379.html

Appendix A
About

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.

MENTEA