

Print & Page Layout Community Group

@ W3C

Tony Graham

Chair

Print & Page Layout Community Group

<http://www.w3.org/community/ppl/> public-ppl@w3.org

@pplcg

<irc://irc.w3.org:6665/#ppl>

Version 1.0 – 14 February 2014

© 2014 Mentea



Print & Page Layout Community Group

@ W₃C

Print & Page Layout Community Group 5

Survey 8

Decision making in XSL-FO processing 10

Area Tree As XSLT Extension Function 13

Adapt Saxon-CE Event Model? 16

References 17

Print & Page Layout Community Group

1

- Introducing the Community Group
- Survey and results
- Feedback in XSL-FO processing
- Feedback extension
- Possible future work

Print & Page Layout Community Group

2

“The Print and Page Layout Community Group is open to all aspects of page layout theory and practice. We can and will cover everything from the Crystal Goblet through to specifications and on to the nitty-gritty of writing stylesheets. You will find XSL-FO discussed here, but you will also find other stylesheet languages, and all are equally welcome.”

History

3

- Started after XML Prague 2012
- Interest in feedback in processing after XML Prague 2013
 - Presentation by Patrick Gundlach of Speedata
- Survey and revised description January 2014
- ????? after XML Prague 2014

Who we are

4



Member involvements

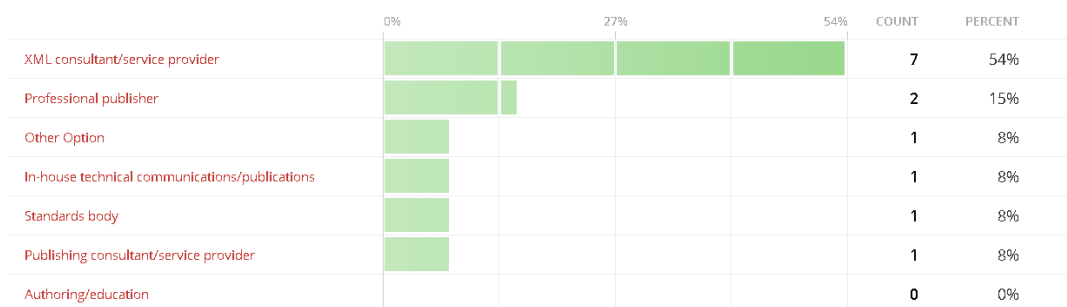
5

Question
01

My involvement is with:

Answers
13
100%

Skips
0
0%



Member interests

6

Question
05

My interest in ... is as ...

Answers
13
100%

Skips
0
0%

	SPECIALIST	GENERALIST	NOVICE	INTERESTED BYSTANDER	UNINTERESTED
Page layout	10	3	0	0	0
Single source/multiple outputs	9	2	0	1	1
XML	13	0	0	0	0
HTML/XHTML/HTML5	4	7	2	0	0
CSS	3	8	2	0	0
XSL-FO	8	4	0	0	0
Digital Publishing	7	5	0	0	0
Writing software	7	4	0	1	0
Standards development	7	4	0	2	0

PPL WG Members' Technologies

7

- AGFA/Xerox CAPS
- ArborText
- Cognos
- DITA Open Toolkit
- DSSSL
- Developed in-house
- DynaText
- EPUB/EPUB3
- FrameMaker
- HTML+CSS
- InDesign
- LogiXML
- Omnimark
- Panorama
- Perl POD
- Quark
- speedata
- TeX/LaTeX
- Troff
- WordPerfect SGML/XML tools
- XSL-FO

Survey

8

- Original group description was dated
 - Mentioned XML Prague 2012 as “recent”
- Group discussed what was important
- Survey found priorities
- 64 responses
- Member and non-member responses very different

XSL-FO should be...

9

Question
06

XSL-FO discussion at the PPL CG should be:

Answers

13

100%

Skips

0

0%

	0%	19.5%	39%	COUNT	PERCENT
Something that just happens as necessary				5	38%
Secondary objective				5	38%
Primary focus				3	23%
Off-topic				0	0%

Question
06

XSL-FO discussion at the PPL CG should be:

Answers

45

100%

Skips

0

0%

	0%	24.5%	49%	COUNT	PERCENT
Primary focus				22	49%
Something that just happens as necessary				12	27%
Secondary objective				8	18%
Off-topic				3	7%

PPL CG should develop...

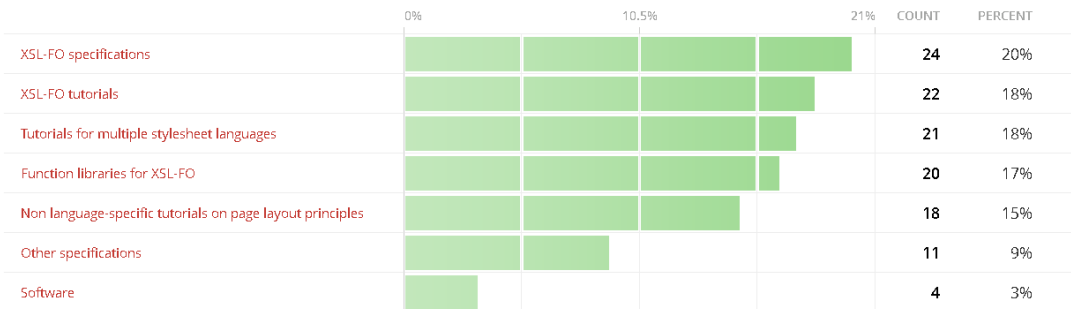
10

Question
07

The PPL CG should develop:

Answers
36
277%Skips
0
0%Question
07

The PPL CG should develop:

Answers
120
267%Skips
0
0%

Specifications should be...

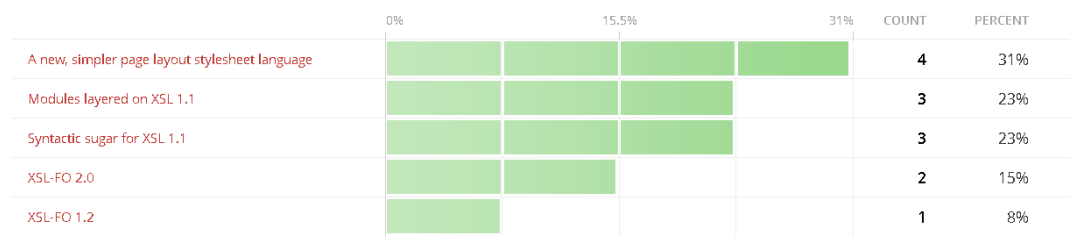
11

Question
08

If the PPL CG works on a styling specification, it should be:

Answers
13
100%

Skips
0
0%

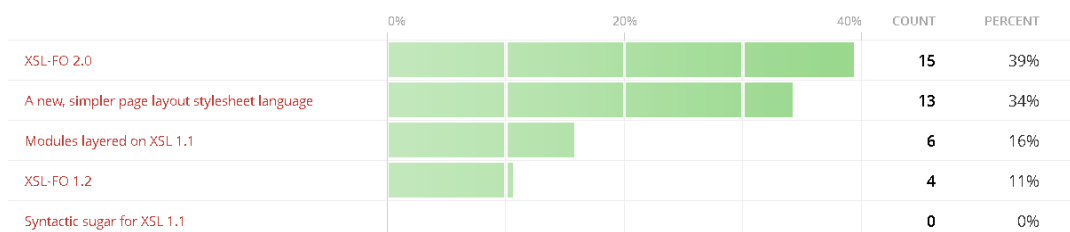


Question
08

If the PPL CG works on a styling specification, it should be:

Answers
38
84%

Skips
7
16%



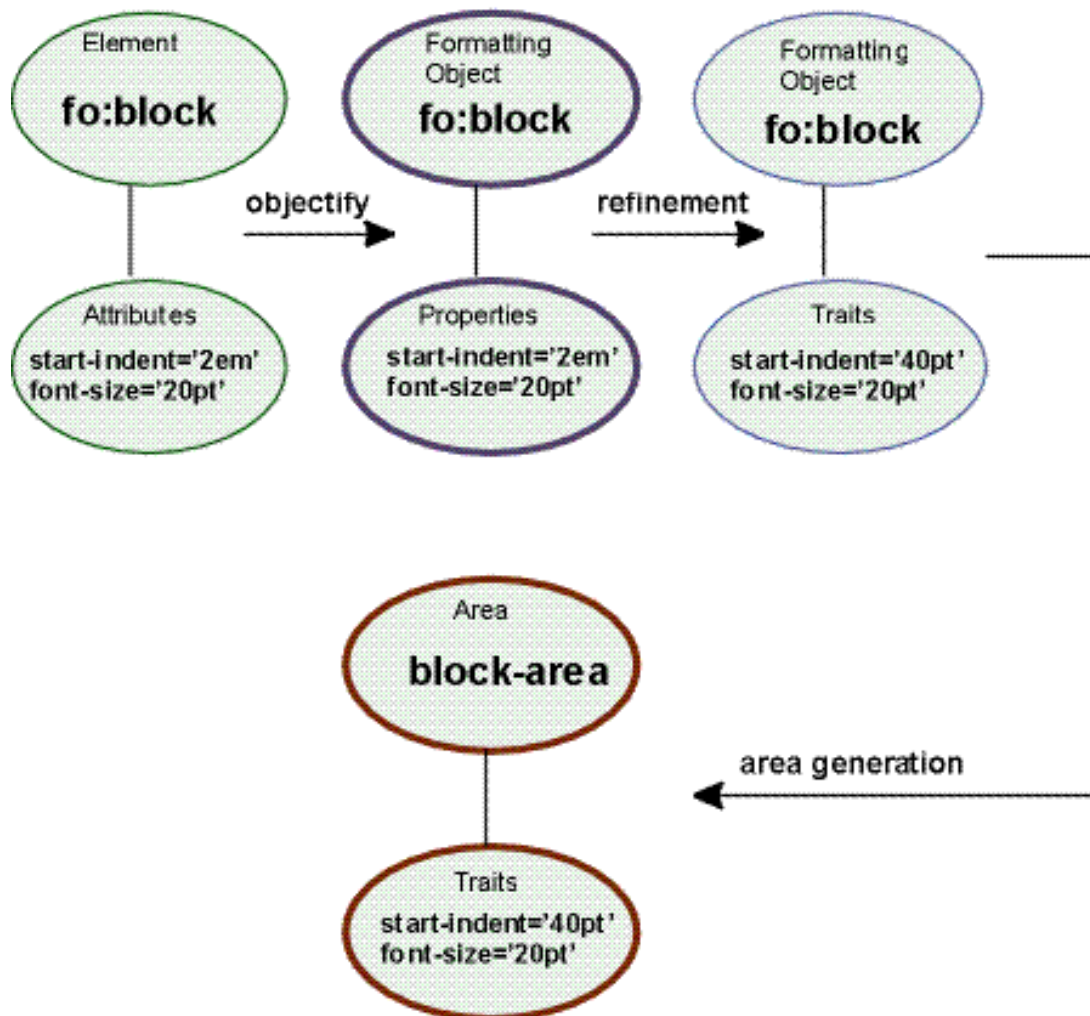
Decision making in XSL-FO processing

12

- Interesting to PPL CG members
 - Spurred by Patrick Gundlach of Speedata
- Decision making in XSL-FO processing model
- PPL CG extension for running formatter inside XSLT
- Possible future work

XSL-FO Processing Model

13



What Decision Making?

14

- Making a choice between alternatives
- Deciding, e.g.:
 - Where to break lines
 - Where to break blocks
 - Where to place floats
 - What to use as marker content
 - Scale of graphics
 - Space-before and space-after of nested blocks
- No control once FO produced
- “Fire-and-forget” processing

“Fire-and-forget”

15

“**Fire-and-forget** is a term for a type of missile guidance which does not require further guidance after launch such as illumination of the target or wire guidance, and can hit its target without the launcher being in line-of-sight of the target...

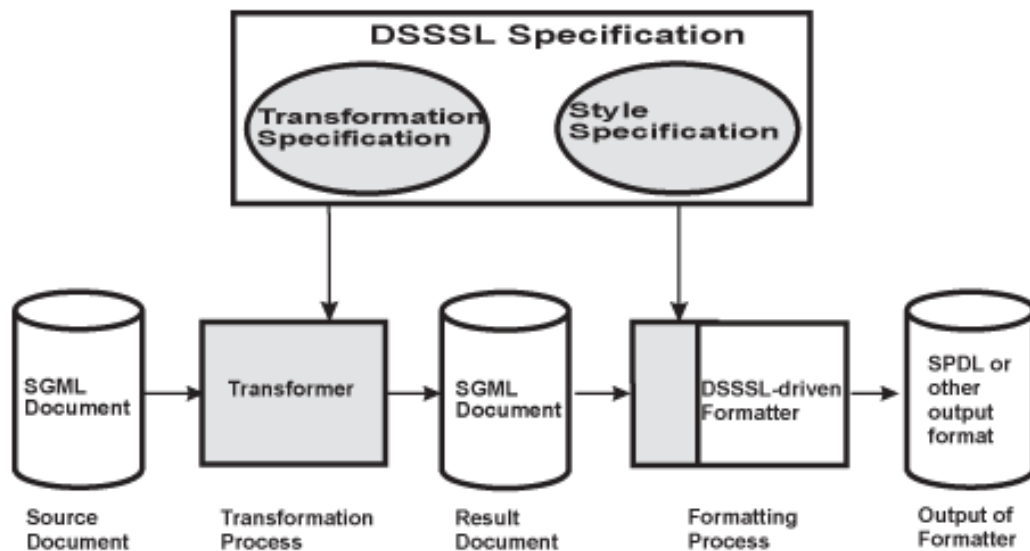
Generally, information about the target is programmed into the missile just prior to launch. ... *After it is fired, the missile guides itself* by some combination of gyroscopes and accelerometers, GPS, organic radar, and infrared optics.”

<https://en.wikipedia.org/wiki/Fire-and-forget>
(Italics added)

XSL-FO Follows DSSSL

16

“XSL builds on the prior work on Cascading Style Sheets [CSS2] and the Document Style Semantics and Specification Language [DSSSL].”



XSL-FO 2.0 Requirements

17

- Section 2.3, Including information from formatting time
 - “to allow expressions that include information that’s only available at formatting time.”
- Section 3.2, Pagination information
 - “to compute expressions that are based on information that is only available after the pagination stage”
- Section 2.1.4, Copyfitting
 - “shrink or grow content (change properties of text, line-spacing, ...) to make it constrain to a certain area.”
 - “multiple instances of alternative content can be provided to determine best fit”
 - act “across a given number of pages, regions, columns etc, for example to constrain the number of pages to 5 pages.”

Area Tree As XSLT Extension Function

18

<http://www.w3.org/community/ppl/wiki/XSLTExtensions>

- Updated distribution before the end of today!
- Saxon and Xerces extension functions
- Runs FOP or Antenna House on input and returns area tree
- First implemented by Arved Sandstrom of MagicLamp Software Solutions

Example: List Label Length

19

The Print and Page Layout Community Group is:
 1.1 the "virtual water cooler"
 1.45678 where you can hang out

The Print and Page Layout Community Group is:
 1.1 the "virtual water cooler"
 1.456 where you can hang out

Adjusted List Label Length

20

The Print and Page Layout Community Group is:
 1.1 the "virtual water cooler"
 1.45678 where you can hang out

The Print and Page Layout Community Group is:
 1.1 the "virtual water cooler"
 1.456 where you can hang out

Transform With Overrides

21

```
<xsl:variable name="overrides">
  <overrides>
    <!-- Find the maximum label width for each list and convert to pt. -->
    <xsl:for-each select="key('lists', true())">
      <xsl:variable name="id" select="@id" as="xs:string" />
      <xsl:variable name="block"
        select="key('blocks', $id, $area-tree)[1]" />
      <override
        id="{ $id }"
        label-width="{ max($block//text/@ipd) div 1000 }pt" />
    </xsl:for-each>
  </overrides>
</xsl:variable>

<xsl:apply-templates select="/">
  <xsl:with-param name="overrides" select="$overrides" as="document-node()"
  tunnel="yes" />
</xsl:apply-templates>
```

Example: Fill Box

22

The Print and Page Layout Community Group is open to all aspects of page layout theory and practice. We can and will cover everything from the Crystal Goblet through to specifications and on to the nitty-gritty of writing stylesheets. You will find XSL-FO discussed here, but you will also find other stylesheet languages, and all are equally welcome.

Iteration

23

```

<xsl:choose>
  <xsl:when test="$iteration eq $iteration-max">
    <!-- Format final output -->
  </xsl:when>
  <xsl:when test="$bpd div 1000 > $target-height">
    <xsl:call-template name="do-box">
      <xsl:with-param
        name="font-size"
        select="($font-size + $font-size.minimum) div 2"
        as="xs:double" />
      <xsl:with-param
        name="font-size.maximum"
        select="$font-size"
        as="xs:double"
        tunnel="yes" />
      <xsl:with-param name="iteration" select="$iteration + 1" as="xs:integer" />
    </xsl:call-template>
  </xsl:when>
  <xsl:when test="$target-height - ($bpd div 1000) <=
    $target-height * $tolerance div $target-height">
    <xsl:message>It fits.</xsl:message>
    <!-- Format final output -->
  </xsl:when>

```

Iteration (continued)

24

```

<xsl:otherwise>
  <xsl:call-template name="do-box">
    <xsl:with-param
      name="font-size"
      select="($font-size + $font-size.maximum) div 2"
      as="xs:double" />
    <xsl:with-param
      name="font-size.minimum"
      select="$font-size"
      as="xs:double"
      tunnel="yes" />
    <xsl:with-param name="iteration" select="$iteration + 1" as="xs:integer" />
  </xsl:call-template>
</xsl:otherwise>
</xsl:choose>

```

Using the extension

25

```
<xsl:variable
  name="area-tree"
  select="ppl:area-tree($fo_tree)"
  as="document-node()" />

<xsl:variable
  name="block"
  select="ppl:block-by-id($area-tree, key('boxes', true())[1]/
@id)"
  as="element()" />

<xsl:variable
  name="bpd"
  select="ppl:block-bpd($block)"
  as="xs:double" />
```

Summary: Extension Function

26

<http://www.w3.org/community/ppl/wiki/XSLTExtensions>

- It shouldn't be this hard
- Different formatter: different area tree
 - Different elements, attributes, namespace
 - Different length format
- Common area tree format and/or accessor library would make more portable
 - `ppl:block-bpd()` example of what could be done
 - Accessors plus common format would be better

Adapt Saxon-CE Event Model?

27

- From XSL-FO 2.0 requirements:
 - "... information that's only available at formatting time."
 - "... based on information that is only available after the pagination stage"
 - "multiple instances of alternative content can be provided to determine best fit"
 - act "across a given number of pages, regions, columns etc, for example to constrain the number of pages to 5 pages."
- How to put decision making inside the formatter?
- How to do it without inventing a complicated new language?

Saxon-CE and XSL-FO Events

28

- Saxon-CE: Template for context in mode corresponding to UI event

```
<xsl:template match="button[@id='reset']" mode="ixsl:onclick">
  <xsl:for-each select="//div[starts-with(@id, 'square')]">
    <xsl:result-document href="#{@id}" method="replace-content">
      <xsl:text>&#xa0;</xsl:text>
    </xsl:result-document>
  </xsl:for-each>
</xsl:template>
```

- XSL-FO: Template for context in mode corresponding to formatter event?

```
<xsl:template match="BlockArea[key('fig', @id, $src-doc)]"
  mode="ppl:overflow">
  <xsl:result-document href="#{@id}/area:external-graphic"
    method="replace-content">
    <xsl:copy>
      <xsl:apply-templates select="*" />
      <xsl:attribute name="width"
        select="ppl:scale(area:external-graphic/@width,
          0.8)" />
    </xsl:copy>
  </xsl:template>
```

Summary: Event Handling

29

- Not (yet) implemented
- Solves expression language question
- What are the useful event types?
- Should it modify FO tree, area tree, or both?

What you can do

30

- Use the extension
<http://www.w3.org/community/ppl/wiki/XSLTExtensions>
- Join the Print and Page Layout Community Group
- Contribute to PPL CG work

References

31

- slide 2 – Print & Page Layout Community Group
<http://www.w3.org/community/ppl/>
- slide 7 – Technologies
<http://lists.w3.org/Archives/Public/public-ppl/2013Dec/0032.html>

