

XSLT and XSL FO Toolbox of Tips and Tricks

Tony Graham
Antenna House, Inc.
tgraham@antenna.co.jp
tony@antennahouse.com
<http://www.antennahouse.com>

Version 1.2 – 5 June 2015

Licensed under a Creative Commons Attribution-Noncommercial-
Share Alike 3.0 Unported License

XSLT and XSL FO Toolbox of Tips and Tricks

Extended Example: UDHR in Unicode 5

Discovery 9

Mapping 16

Implementation 17

When Are You Done? 52

Summary 54

Appendix A – **About** 55

Exercises 56

Learning Objectives

1

- Work through a small FO project
- See ways of doing things

We'll be working through a complete sample project to illustrate the process of developing a stylesheet for formatting XML using XSL FO.

Outline

2

- Introducing the extended example
- Discovery
- Mapping
- Implementation

Purely for the sake of convenience, the project has been divided into three stages – discovery, mapping, and implementation. Real-life projects are seldom as clean-cut as that.

Extended Example: “UDHR in Unicode”

3

- Working through making FO for “UDHR in Unicode”
- UDHR = UN “Universal Declaration of Human Rights”
- Unicode = big character set
- UDHR in Unicode = text in lots of languages
- See <http://www.unicode.org/udhr/>

Universal Declaration of Human Rights - English

© 1996 – 2009 The Office of the High Commissioner for Human Rights

This HTML version prepared by the *UDHR in Unicode* project, <http://www.unicode.org/udhr>.

Universal Declaration of Human Rights

Preamble

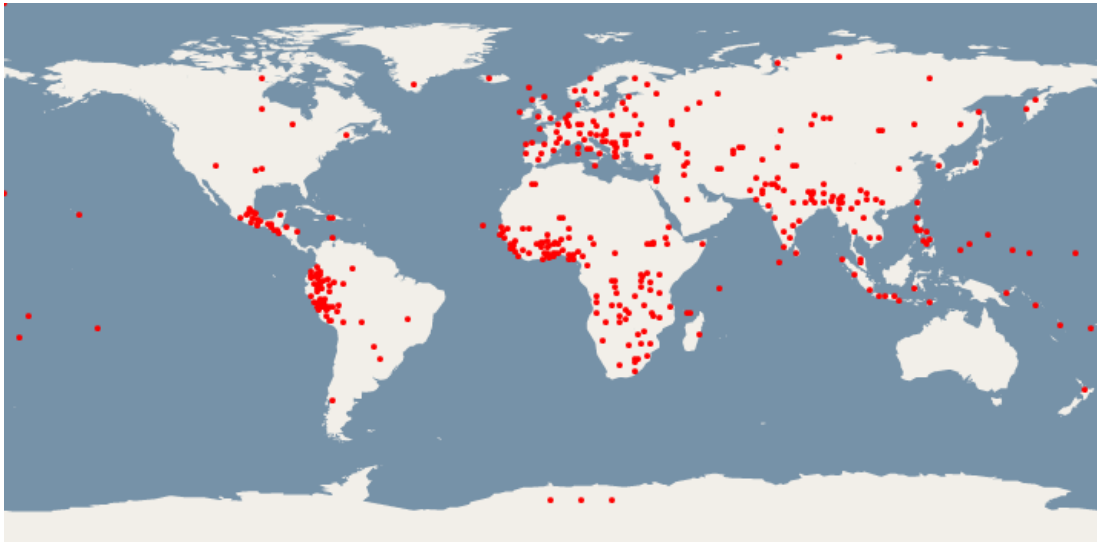
Whereas recognition of the inherent dignity and of the equal and inalienable rights of all members of the human family is the foundation of freedom, justice and peace in the world,

The sample project is formatting “UDHR in Unicode”.

Why “UDHR in Unicode”?

4

- Simple structure
- Largish set of samples
- Example PDFs available
- HTML stylesheet available



“UDHR in Unicode”, with its about 400 translations of the UDHR, makes a useful sample because it is large enough to be realistic but simple enough that we can cover the important features in the time available.

The measles on the map represent locations for the scripts covered by “UDHR in Unicode”. If the map showed areas rather than points, a lot more of it would be covered since, for example, Australia would be shaded for English. The dots in Antarctica represent languages, such as Esperanto, that aren't specific to any region.

1948 and All That

5

- 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”

The Universal Declaration of Human Rights was adopted in 1948 to back up some of the phrases in the UN charter.

“UDHR in Unicode” XML Files

6

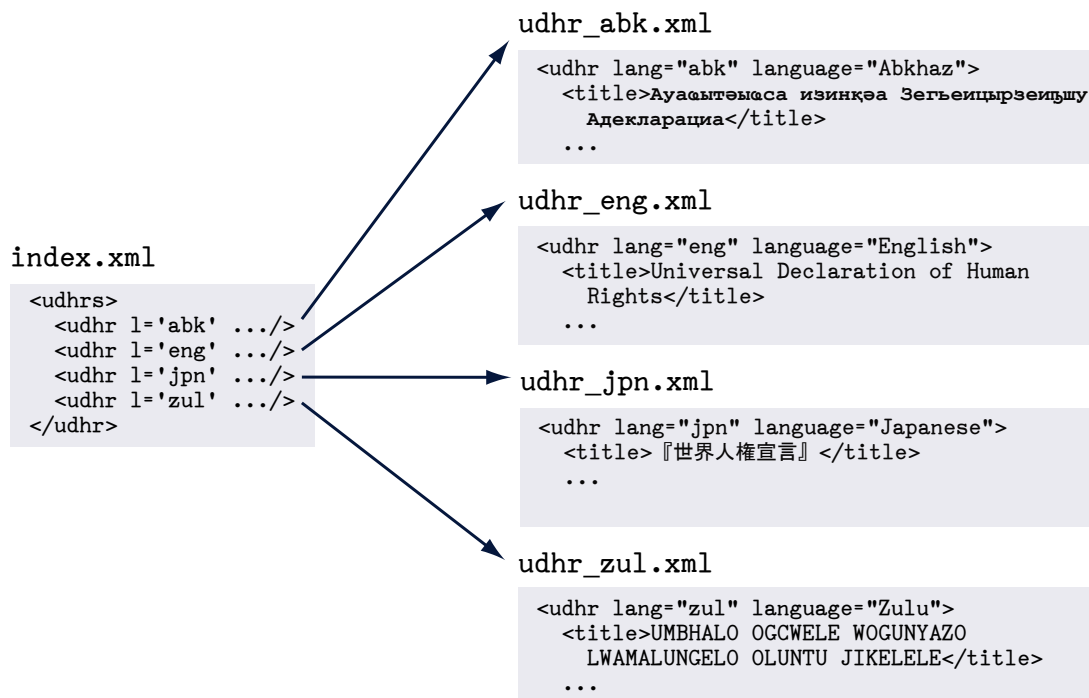
- index.xml
 - Metadata about translations
 - References to translation files
- udhr_*.xml
 - Text of UDHR in one language and script

There are two sorts of files in the XML produced by the “UDHR in Unicode” project:

- The single index.xml file contains metadata about all of the other per-language/script files.
- There’s one XML for every language/script covered by “UDHR in Unicode”. Most contain the full text of the UDHR, but some are only partially complete.

index.xml and UDHR XML Files

7



This illustrates the correspondence between elements in index.xml and the per-language/script XML files.

Per-translation Metadata in index.xml

8

```
<udhrs
...
  <udhr
    l='zul' iso639-3='zul'
    uli='zu' bcp47='en'
    ohchr='zuu' stage='4'
    pdf='y' notes='y'
    loc='-28,29' country='ZA'
    region='Africa' demo='n'
    n='Zulu' />
  </udhr>
</udhrs>
```

The index.xml metadata for Zulu.

<udhr> Attributes

9

Attribute	Description
l='zul'	Ethnologue (15th Ed.) language identifier
iso639-3='zul'	ISO 639-2 Alpha-3 language code
uli='zu'	Unicode language identifier
bcp47='zu'	IETF BCP 47 language tag
ohchr='zuu'	UN OHCHR language identifier
stage='4'	"Degree of progress" (1–5)
pdf='y'	Whether "UDHR in Unicode" has produced PDF
notes='y'	Whether there's additional notes about the translation
loc='-28,29'	Map reference
country='ZA'	ISO 3166 country code
region='Africa'	Geographic region
demo='n'	???
n='Zulu'	Language name

udhr_zul.html

10

```

<udhr lang="zul" language="Zulu"
  xmlns="http://www.unhchr.ch/udhr">
  <title>UMBHALO OGCWELE WOGUNYAZO LWAMALUNGELO OLUNTU
  JIKELELE</title>
  <preamble>
    <title>Isandulelo</title>
    <para>Ngokunjalo ukwamukelwa ngokuzuzwa kwesithunzi
    samalungelo alinganayo najwayelekile awowonke amalunga
    omndeni wesintu kuyisisekelo senkululeko, sobulungiswa
    noxolo emhlabeni,</para>
    ...
  </preamble>
  <article number="1">
    <title>Isigaba 1</title>
    <para>Bonke abantu bazalwa bekhululekile belingana
    ngesithunzi nangamalungelo. Bahlanganiswe wumcabango
    nangunembeza futhi kufanele baphathane ngomoya
    wobunye.</para>
  </article>
  ...
</udhr>

```

An excerpt of the UDHR in Zulu with "UDHR in Unicode" markup.

Discovery

11

- Finding what's in the source documents
- What contains what
- What attribute values are used

The first phase in a typical XSL FO project is discovering what you have to work with. Even when you have a schema, it's helpful to have a number of sample documents so you can see what's actually used. There's little point implementing everything in the schema if only 10% of it is actually used.

The schema may allow nested structures – e.g., lists within lists, tables within tables, lists within paragraphs within lists within paragraphs – and it's helpful to know how many of those you have to support, and just as importantly, which combinations you're unlikely to have to support.

The schema may define attributes or elements as containing only text, but the actual documents may contain only certain values that you will have to handle properly.

And, of course, you may be dealing with well-formed XML for which there is no schema.

How To Find?

12

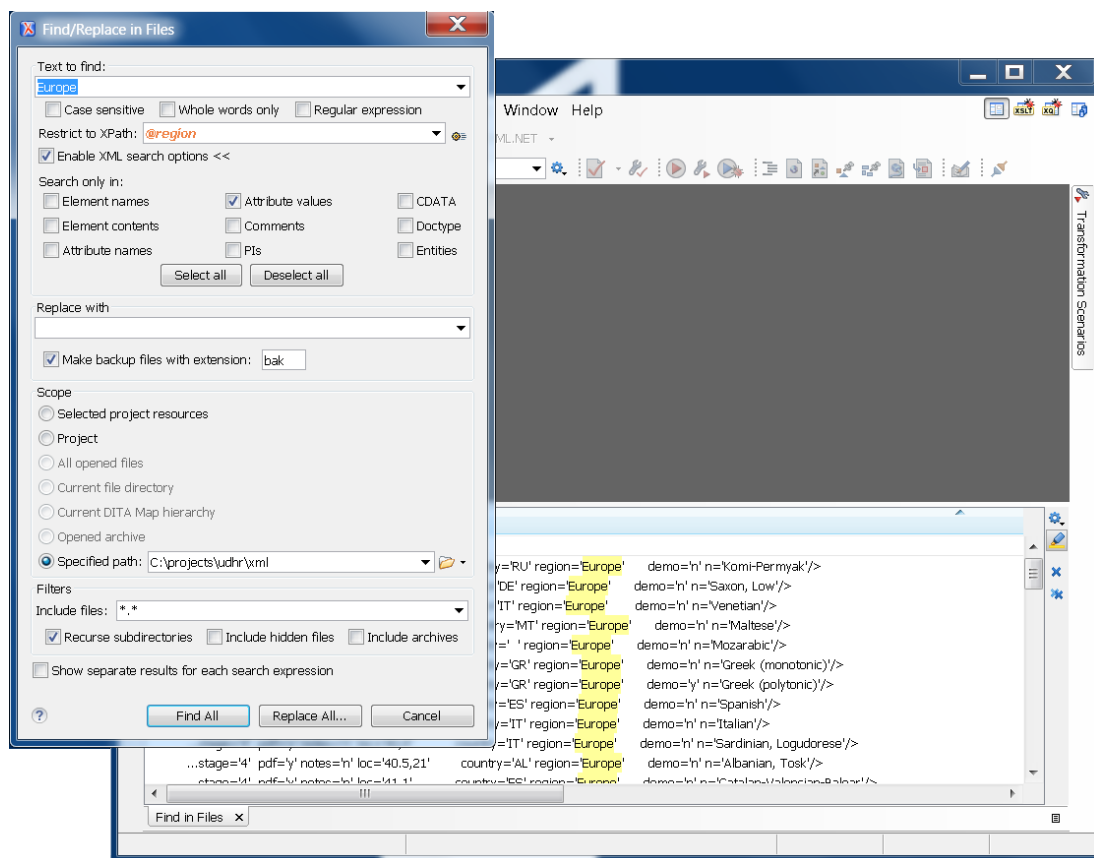
- Inspect the samples
- Search using grep or similar
- Generate a schema
- Use XQuery

The simplest way to discover what's in your source documents is just to look at them, particularly if you have samples of a style that you have to reproduce. You may, indeed, find details that you won't find any other way, but it's hard to properly examine a lot of samples just by looking at them.

You can, for example, use grep to search for strings in the source documents, but grep isn't XML-aware.

Multi-file Search in XML Editor

13



You may be able to do XML-aware searches across multiple files from within your XML editor.

Generate a Schema

14

- Let a schema generator do the work for you
- May find more than you can by inspection
- Real schema says what's allowed
- Sometimes you just want what's used
 - May plan to throw error message for unexpected

Your XML source may already conform to a schema, but the XML that you'll be working with may use only a subset of what's allowed. If you can find out what's actually used, then you can avoid implementing styles that will never be used.

trang

15

```
trang file.xml schema.rnc
```

```
trang *.xml schema.dtd
```

- Written by James Clark
- One *or more* input files
- Output as DTD, W3C XML Schema, or RelaxNG
- Run using oXygen schema generator when too many input files for command-line

trang is probably the most commonly used tool for generating a schema from a set of source documents.

Generated RNC Schema for index.xml

16

```

start =
  element udhrs {
    element udhr {
      attribute bcp47 { text },
      attribute country { text },
      attribute demo { xsd:NCName },
      attribute iso639-3 { text },
      attribute l { xsd:NMTOKEN },
      attribute loc { text },
      attribute n { text },
      attribute notes { xsd:NCName },
      attribute nv { xsd:NMTOKEN }?,
      attribute ohchr { text },
      attribute pdf { xsd:NCName },
      attribute region { text },
      attribute stage { xsd:integer },
      attribute uli { text },
      attribute v { xsd:NMTOKEN }?
    }+
  }

```

This is a good start. We now know that @nv and @v are not always present. trang was able to detect, for example, that @l has a restricted range of values, but we already know just by looking that some of the attribute have more specific values than trang shows.

Exercise 1 – Using Trang

17

Use Trang to generate a schema for all the udhr_*.xml files.

XQuery

18

When you have many sample documents:

- Load samples into XML database
- Query to see what you've got
- Often simpler than doing it in XSLT and/or grep

XQuery is useful for using XPath's over multiple documents to find out exactly what's in them.

Sample XQuery Processors

19

- Saxon
 - Open source Saxon HE works okay
 - Reads all the files every time
 - Queries in external files
- MarkLogic
 - Not open source
 - Need to have server running
 - Ad-hoc queries through cq web form
- eXist-db
 - Open source
 - Need to have server running
 - Ad-hoc queries through eXide shell
- BaseX
 - Open source
 - Don't always need to have server running
 - Ad-hoc queries through GUI

A very incomplete list of available XQuery processors.

What to XQuery?

20

- Counts of occurrences
- Distinct values of an attribute
- Whether things always occur together
- Base URI of interesting files so you can eyeball the XML

What you look for using XQuery will obviously depend on your particular XML. It's often useful to get the base URI of files that exhibit the particular features that you have to handle. By also looking at the XML for those files, you may notice patterns in the markup that you would have trouble noticing if you only fish through the files using XQuery.

Saxon XQuery Example

21

Find the region names used in index.xml

```
for $region in distinct-values(/udhrs/udhr/@region)
order by $region
return concat("'", $region, "' &#10;")
```

```
java -cp saxon9he.jar net.sf.saxon.Query -s:index.xml regions.xq
```

```
' '
'Africa'
'America'
'Americas'
'Asia'
'Europe'
'Other'
'Pacific'
```

XQuery, command line to run, and output from Saxon.

XQuery Example

22

Find the UDHRs with multiple <note>

```
for $udhr in collection()/
udhr:udhr[count(udhr:note) > 1] return base-uri($udhr)
```

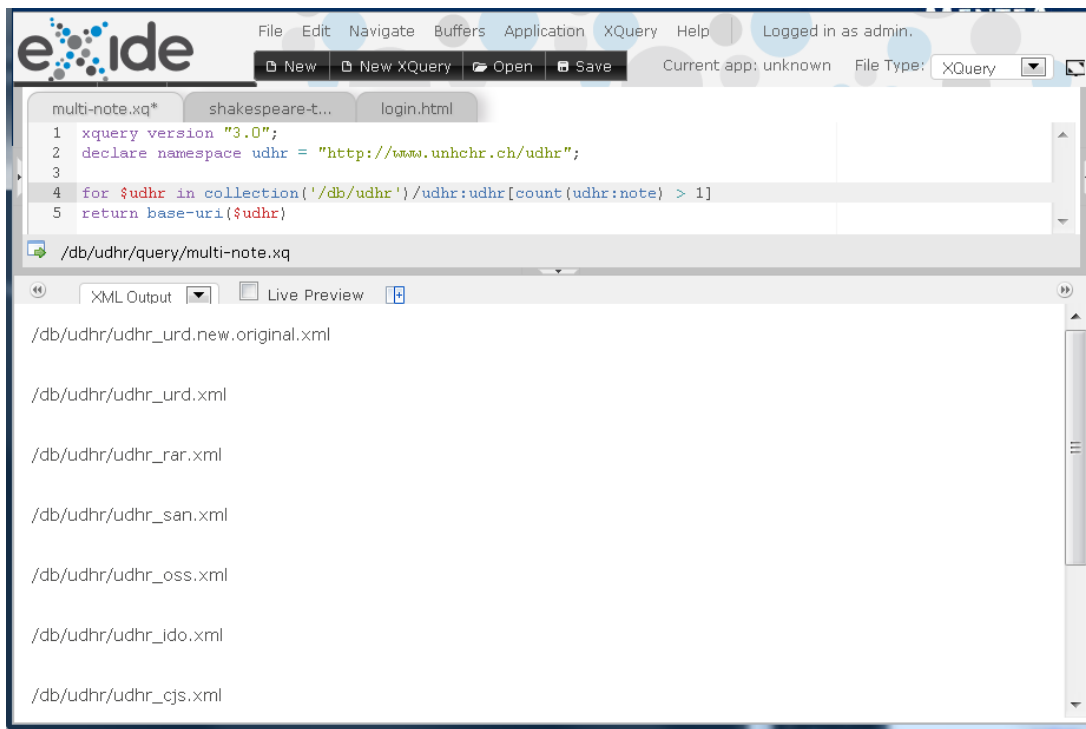
```
/udhrs/udhr_cjs.xml
/udhrs/udhr_gla.xml
/udhrs/udhr_ido.xml
/udhrs/udhr_mai.xml
/udhrs/udhr_oss.xml
/udhrs/udhr_rar.xml
/udhrs/udhr_san.xml
/udhrs/udhr_urd.new.original.xml
/udhrs/udhr_urd.xml
/udhrs/udhr_ydd.xml
```

Sample XQuery and the result from running the XQuery in an XML database.

eXist XQuery Example

23

Find the UDHRs with multiple <note>:



The result of running an XQuery in the eXide application that comes with eXist.

Exercise 2 – XQuery

24

Find the length of the longest `udhr:item/@tag` value.

Mapping

25

- Specification of how input maps to output
- Useful for:
 - Telling what to write
 - Checking completeness of transform
 - Writing tests
- Complexity depends on size of project
- Get someone else to do it
- Needs to be kept up to date!
- XML → XML usually simpler, fewer details than XML → styles

When the project is large enough to warrant the effort, particularly when there's multiple people working on the project, it can be useful to document the mapping between the source XML and the desired output.

However, the mapping needs to be kept up to date as decisions change.

How to Write a Mapping?

26

- Narrative text
- Narrative text in XML
 - Markup for input and output contexts
 - Run transforms to check everything covered
- Spreadsheet
 - Input context and output specification in different columns in same row

How you write the mapping depends on the conventions of your organization and on the complexity of the mapping.

Sample Mapping

27

- `udhrs` becomes `fo:root`
- `udhr:udhr` becomes `fo:page-sequence`
- `udhr:udhr/udhr:title` becomes `fo:block`
 - `font-size: 2em`
 - `font-weight: bold`
 - `space-after: 12pt`
- Other `udhr:title` become `fo:block`:
 - `font-weight: bold`
 - `space-before: 3pt`
 - `space-after: 6pt`
- `udhr:orderedlist` becomes `fo:list-block`:
 - `provisional-distance-between-starts: 15pt`
 - `provisional-label-separation: 5pt`
 - `space-before: 6pt`
 - `space-after: 6pt`

A partial mapping of "UDHR in Unicode" XML to FO styles.

Implementation

28

- "Successive approximation" or "progressive refinement"
- Re-use
- Parameterise

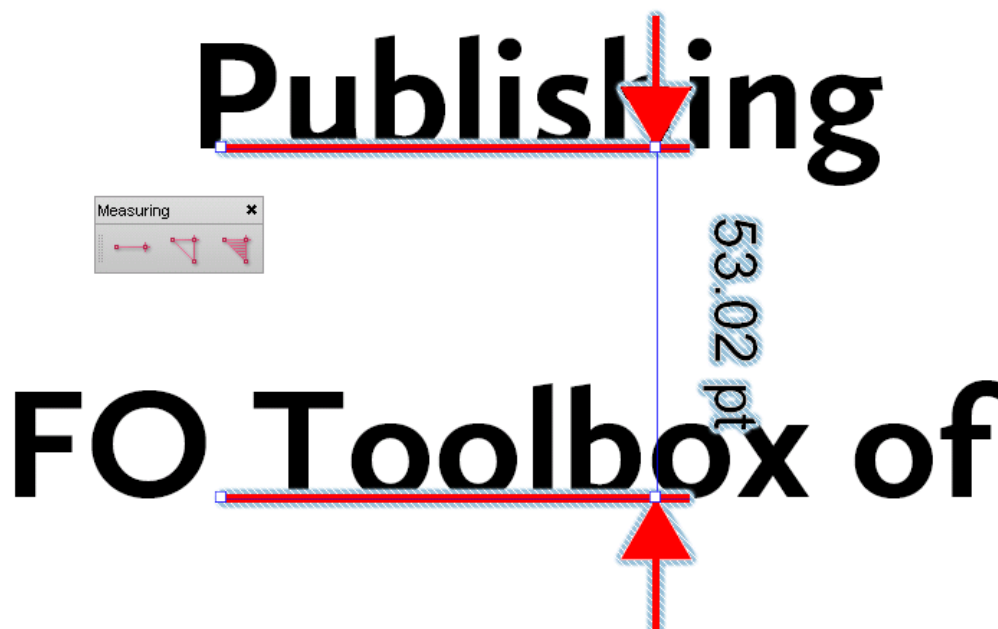
Even when you have a comprehensive mapping, it's more usual to start with some basic styles and build on that than it is to write the entire stylesheet before running any of it.

As you build up your stylesheet, look for ways to reuse templates, often by replacing hard-coded property values with parameter references and by passing different parameter values to the templates each time they are used.

Working From Designer's Design

29

- Probably won't cover every context
- Try to use same application as used by designer
- Use ruler in application or PDF viewer



If you have them, the design documents or mockup samples produced by the designer can be very useful for finding specific details about font sizes, leading, etc. However, the documentation and samples probably won't cover every context that your stylesheet will have to handle. Also, samples often show the ideal case and may provide few clues about how to handle text that is either much shorter or much longer than in the sample.

The designer might not have documented every aspect of the design that can be represented using FOs and properties. You may, however, be able to open the sample in an application that will let you find some of the values that you need. When all else fails, you can still use a ruler and pencil!

Stylesheet Stage 1

30

- Plain title page
- One fo:page-sequence per translation
- One fo:block per translation
- No table of contents
- Reused existing page layout
- 1175 pages for all translations

When developing the sample application, the first iteration just reused the page masters from some previous work and simply produced one fo:page-sequence per translation. The result wasn't at all pretty, but at this point it wasn't expected to be pretty.

Initial Output

[illegible]

Stylesheet Stage 2

- fo:block for every title and paragraph
- Writing mode depends on language
- 1520 pages for all translations

In the next iteration, after a bit more development time, the text is broken into separate blocks and the stylesheet sets the writing mode depending on the current language.

Text Direction

34


- English is written left-to-right
- Some other languages aren't (or aren't always)


English Left to right

 Universal Declaration of Human Rights

Hebrew Right to left

 הכרזה לכל באי עולם בדבר זכויות האדם

Japanese Left to right

 『世界人権宣言』

Japanese
 『世界人権宣言』
 Top to bottom


Mixed Text Direction

35

- Right-to-left text in left-to-right
- Left-to-right text in right-to-left
- Works according to Unicode BIDI algorithm
 - Used by (nearly) everything that uses Unicode
 - Including web browsers
 - Including XSL FO
- Override using markup or special characters when BIDI algorithm isn't what you want
- Overall text direction still set by overall writing mode
 - E.g., right-to-left text in left-aligned paragraph

הכרה לכל באי עולם בדבר זכויות האדם הואיל והכרה בכבוד הטבעי אשר לכל בני משפחה האדם ובזכויותיהם השוות והבלתי נפקעות הוא יסוד החופש, הצדק והשלום בעולם. הואיל והזולז בזכויות האדם ובזיון הבשילו מעשים פראיים שפגעו קשה במצפונה של האנושות; ובנין עולם, שבו ייהנו כל יצורי אנוש מחירות הדיבור והאמונה ומן החירות מפחד וממחסור, הוכרז כראש שאיפותיו של כל אדם. הואיל והכרח חיוני הוא שזכויות האדם תהיינה מוגנות בכוח שלטון של החוק, שלא יהא האדם אנוס, כמפלט אחרון, להשליך את יהבו על מרידה בעריצות ובדיכוי. הואיל והכרח חיוני הוא לקדם את התפתחותם של יחסי ידידות בין האומות. הואיל והעמים המאוגדים בארגון האומות המאוחדות חזרו ואישרו במגילה את אמונתם בזכויות היסוד של

Text direction isn't always entirely one way or entirely the other. XSL FO follows the rules of the Unicode BIDI algorithm for how to handle a run of text in one direction that is nested in a run of text in the other direction.

When the Unicode BIDI algorithm does not give the desired result, it's possible to use XSL FO markup to override the automatic behavior.

Writing Mode in XSL 1.1

36

- Settable on FOs that always generate a reference area
 - And `fo:table` to set row and column order

Applies to:

- `fo:page-sequence`
- `fo:simple-page-master`
- `fo:region-body`
- `fo:region-before`
- `fo:region-after`
- `fo:region-start`
- `fo:region-end`
- `fo:block-container`
- `fo:inline-container`
- `fo:table`

In XSL FO, the writing mode can be set only on the FOs that generate a reference area and also on `fo:table`, since table columns are read right-to-left in right-to-left scripts.

Writing Mode: The Story So Far

37

Initial stylesheet:

- Default initial writing mode: lr-tb

Current stylesheet:

- fo:page-sequence/@writing-mode from udhr:udhr/@lang
- Three letter codes not widely used
- May be some unimplemented by current browsers, etc.
- Codes with variants even less likely to be understood
- Reworked code from udhrs2html.xsl

There may be “UDHR in Unicode” files for languages and scripts that the FO processor does not natively support, so the writing mode is being set explicitly by using a lookup on the three-letter language code that “UDHR in Unicode” uses as the primary language identifier. The lookup is based on similar code in the “UDHR in Unicode” stylesheet for producing HTML.

Setting Writing Mode

38

```
<xsl:template match="udhr:udhr">
  <fo:page-sequence master-reference="PageMaster"
    writing-mode="{m:writing-mode(@lang)}"
    initial-page-number="auto-odd">
    ...
  </fo:page-sequence>
</xsl:template>

<xsl:function name="m:writing-mode" as="xs:string">
  <xsl:param name="lang" />

  <xsl:sequence
    select="if ($lang = ('heb', 'arb', 'pnb', 'skr',
                        'ydd', 'pes', 'urd', 'pbu',
                        'mly_arab', 'uig_arab',
                        'aii', 'div'))
    then 'rl'
    else 'lr'" />
</xsl:function>
```

Stylesheet except showing the writing-mode lookup.

Stylesheet Stage 3

39

- Bold titles
 - May be culturally inappropriate
 - Bold fonts not always available
 - Official "UDHR in Unicode" uses only medium weight
 - But makes the exercise more interesting
- Space between blocks
- Lists
- 1767 pages for all translations

The third iteration added bold titles and more space around the blocks of text. It's possible that there's languages/scripts for which bold titles are culturally inappropriate or simply unavailable, but the purpose of the example project is to demonstrate FO processing.

Titles and List Items in Zulu

40

Isigaba 12

Akekho okuyogaxekwa, ngokungemthetho ezindabeni zakhe zangasese, emndenini, ekhaya noma ezincwadini azithumelayo nazitholayo, noma ukuhlaselwa kwesithunzi nokuhlonipheka kwakhe. Wonke umuntu unelungelo lokuvikelwa kulokho kuhlaselwa nokugxambukela.

Isigaba 13

1. Wonke umuntu unelungelo lokuhamba ngenkululeko nokuhlala phakathi kwemingcele ezungeze imibuso ehlukene.
2. Wonke umuntu unelungelo lokushiya izwe lakhe, futhi abuye abuyele kulo.

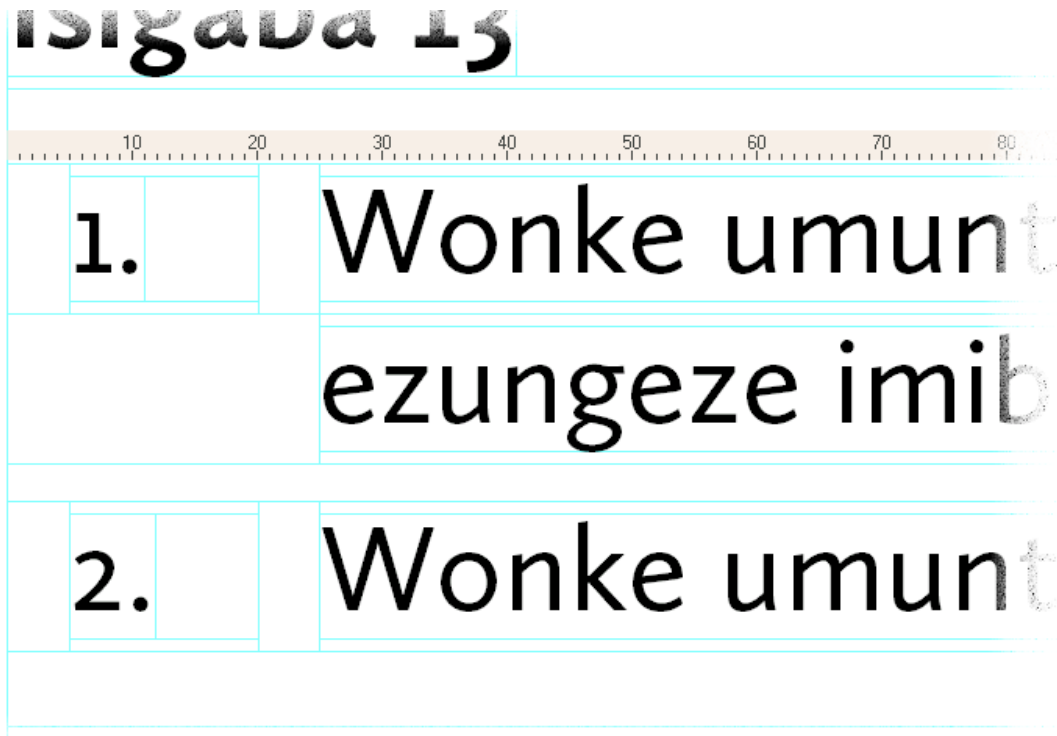
Isigaba 14

1. Wonke umuntu unelungelo lokucela ukuhlala akhosele kwamanye amazwe ebalekela ukuphathwa kabuhlungu.
2. Lelilungelo alinakusetshenziselwa ukubalekela ukushushiswa okufanele okuvela ngamacala angahlangene nezombangazwe noma ngezenzo ezingqubuzana nezinhloso nemigomo yeNhlango yeZizwe.

The next few slides cover formatting lists and use this as their example.

List Detail

41



A close-up of two list items. This is a screenshot from AH Formatter GUI with both area outlines and rulers enabled.

fo:list-block

42

```

<xsl:param name="ordered-list-separation" select="' 15pt' "/>
<xsl:param name="ordered-label-separation" select="' 5pt' "/>

<xsl:template match="udhr:orderedlist">
  <fo:list-block
    space-before="6pt"
    space-after="6pt"
    provisional-distance-between-starts=
      "{$ordered-list-separation} + {$ordered-label-separation}"
    provisional-label-separation=
      "{$ordered-label-separation}">
    <xsl:apply-templates/>
  </fo:list-block>
</xsl:template>

```

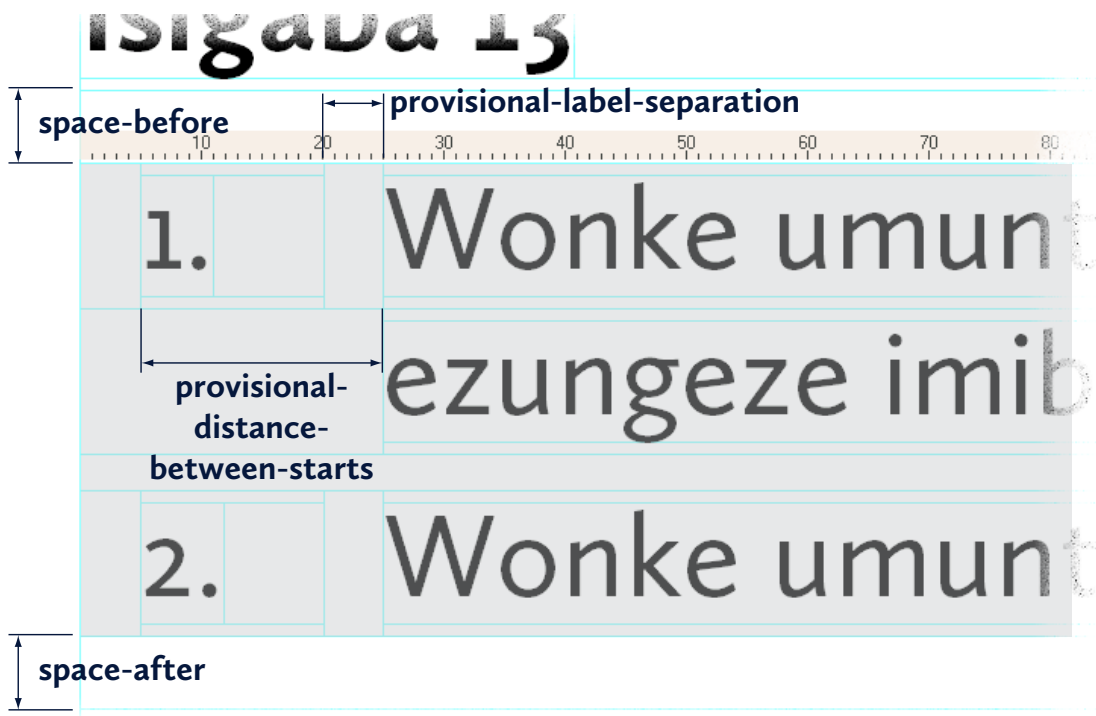
A list is formatted in XSL FO using an `fo:list-block`. The `fo:list-block` contains one or more `fo:list-item`, and each `fo:list-item` contains an `fo:list-item-label` and an `fo:list-item-body`.

This is the template producing the `fo:list-block`.

This is also an example of using parameters and calculations rather than hard-coding values.

Formatted fo:list-block

43



The same screenshot with the area generated by the `fo:list-block` highlighted and with properties indicated. Note that `provisional-label-separation` and `provisional-distance-between-starts` do not set the lengths that they indicate. Instead, the properties' values can be used by descendant `fo:list-item-label` and `fo:list-item-body` to produce areas with the dimensions indicated.

fo:list-item

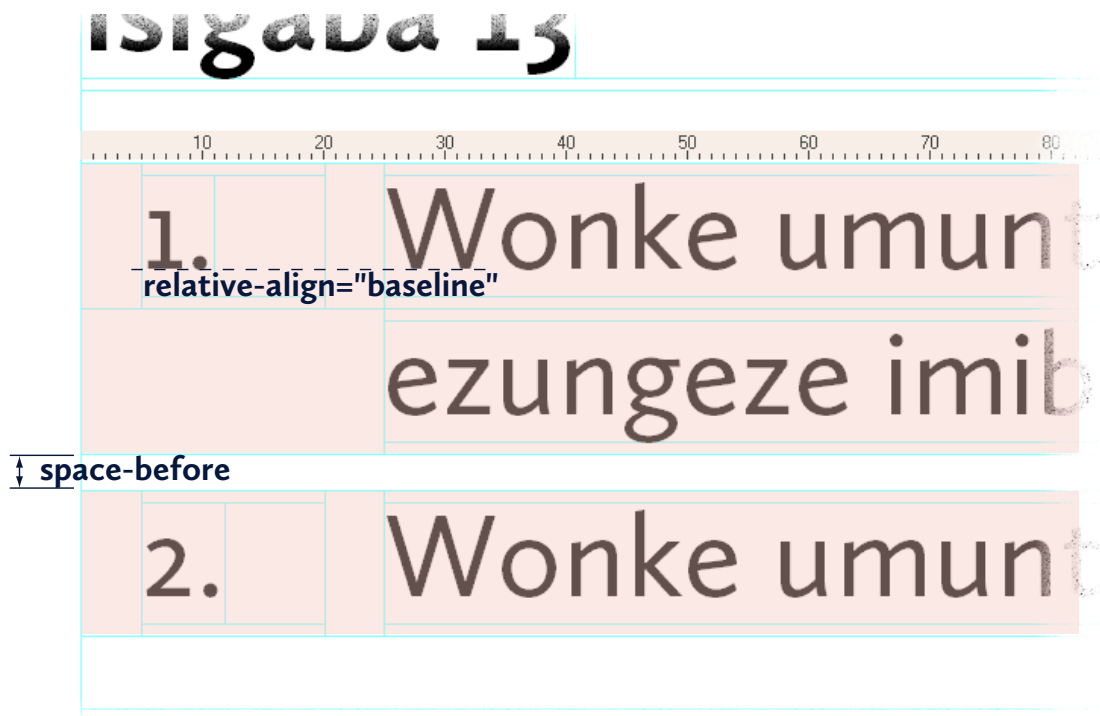
44

```
<xsl:template match="udhr:listitem">
  <fo:list-item space-before="3pt"
    relative-align="baseline">
    <fo:list-item-label ...>
      ...
    </fo:list-item-label>
    <fo:list-item-body ...>
      ...
    </fo:list-item-body>
  </fo:list-item>
</xsl:template>
```

An abbreviated version of the template for `udhr:listitem`.

Formatted fo:list-item

45



The screenshot with the areas generated by the two fo:list-item highlighted.

fo:list-item-label

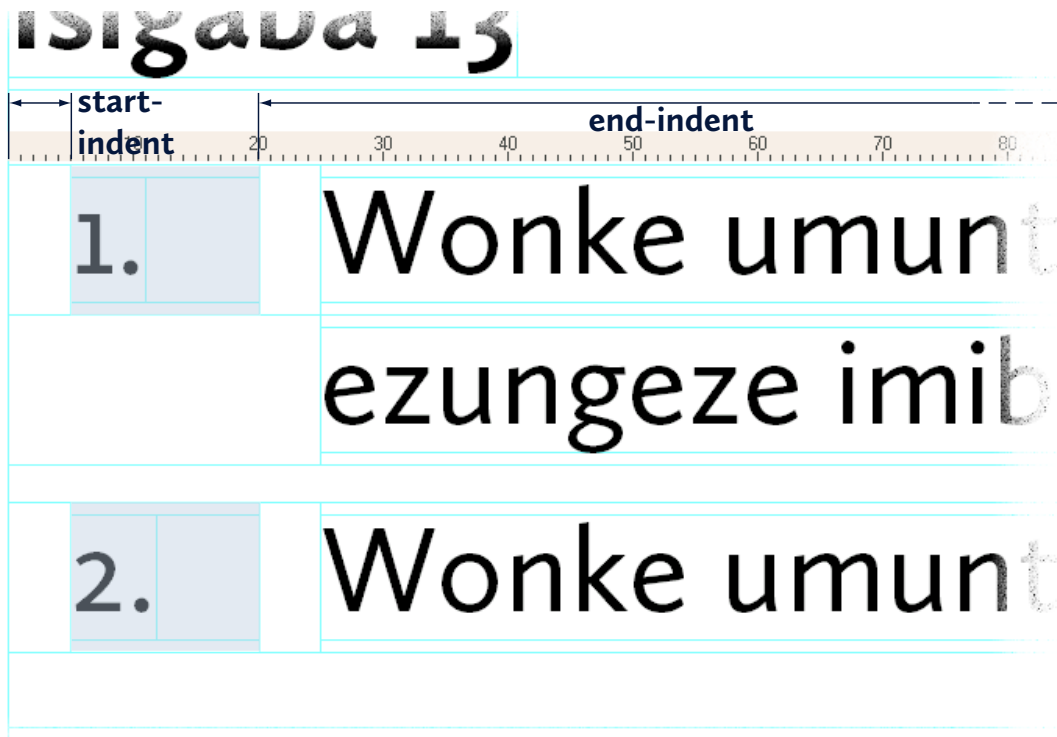
46

```
<fo:list-item-label
  end-indent="label-end()"
  start-indent="{$ordered-label-separation}"
  <fo:block>
    <xsl:choose>
      <xsl:when test="exists(@tag)">
        <xsl:value-of select="@tag"/>
      </xsl:when>
      <xsl:otherwise>
        <xsl:number format='1.' />
      </xsl:otherwise>
    </xsl:choose>
  </fo:block>
</fo:list-item-label>
```

The portion of the previous xsl:template that produces the fo:list-item-label.

Formatted `fo:list-item-label`

47



The screenshot with the areas generated by the `fo:list-item-label` highlighted.

`fo:list-item-body`

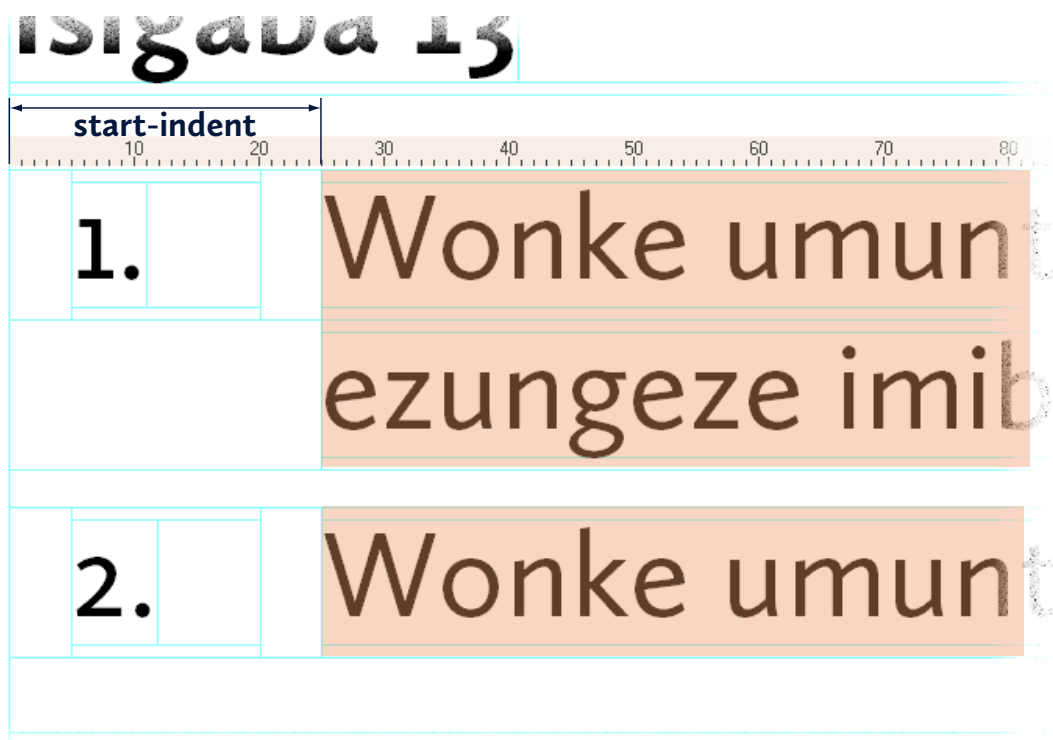
48

```
<fo:list-item-body
  start-indent="body-start()"
  end-indent="{ $ordered-list-separation }">
  <fo:block>
    <xsl:apply-templates/>
  </fo:block>
</fo:list-item-body>
```

The portion of the previous `xsl:template` that produces the `fo:list-item-body`.

Formatted fo:list-item-body

49



The screenshot with the areas generated by the fo:list-item-body highlighted.

body-start() and label-end()

50

- fo:list-item-label/@end-indent and fo:list-item-body/@start-indent could be any value
- label-end() and body-start() inherit from nearest ancestor fo:list-block
- end-indent="label-end()" and start-indent="body-start()" are just "conveniences"
- Can set properties to any value
 - E.g., start-indent="2 * body-start()"

label-end() and body-start() are provided so you can easily use the provisional-label-separation and provisional-distance-between-starts properties from the fo:list-item.

How They Work

51

```

provisional-distance-between-starts
    = 20pt

provisional-label-separation
    = 5pt

body-start() = start-indent + start-intrusion-adjustment +
               provisional-distance-between-starts
               = 5pt + 0 + 20pt
               = 25pt

label-end() = width -
             (provisional-distance-between-starts +
              start-indent +
              start-intrusion-adjustment -
              provisional-label-separation)
             = 100% - (20pt + 5pt + 0 - 5pt)
             = 100% - 20pt

```

The formulas relating provisional-label-separation and provisional-distance-between-starts to label-end() and body-start(). (The dimmed portions apply only when the list item is intruded on by a float.)

Exercise 3 – Lists of Problems

52

- Some scripts use more than single numbers for list markers
- Chamula Tzotzil uses “Jun sloilal.”, “Chib sloilal.” and “Oxib sloilal.”
- How would you handle the changing size of the list markers in the one document?

The range of values used in list item labels in “UDHR in Unicode” requires something more than just fixed label separation if the result is to look good for all languages. How would you handle it?

Working Backwards

53

- When you can’t produce the right FO markup using XSLT...
- Edit sample FO until it’s right
- Then modify the XSLT to match it

You may reach a point where you’re not sure what FOs and properties to generate to produce a correct result. If so, you may need to ‘work backwards’ by editing a sample FO file until it produces the correct result and then modifying the XSLT to recreate the FO markup.

Checking Output

54

- Work with PDF
 - Ruler in PDF reader
 - Export as RTF (YMMV)
 - Paste into word processor or Illustrator to check fonts and sizes (YMMV)
- Open in FO processor application
- xmlroff testing module
- Antenna House Regression Testing Framework (AHRTS)
- Validate with focheck
- Review page masters with fopages

Checking your output isn't a separate phase, since you should be doing this all the way through.

There are multiple ways to check your formatted output, as well as different aspects that can be checked.

Working with the PDF or opening the PDF in some other application to check font sizes, etc., is one option, but there are other alternatives.

xmlroff Testing Module

55

- Works with any command-line XSL processor
- Compares current results against reference
- Summary and individual HTML reports
- "Stereo" view of differences

inherited-property-value()

The value of font-size property specified on ancestor page-sequence is 10pt.

1. No font-size specified. Should be 10pt.

2. font-size="20pt". Should be 20pt.

3. font-size="inherited-property-value()". Should be 10pt.

4. font-size="inherited-property-value(font-size)". Should be 10pt.

5. font-size="inherited-property-value('font-size')". Should be 10pt.

6. font-size="inherited-property-value(provisional-distance-between-starts) div 2". Should be 12pt.

Antenna House Regression Testing System

56

The screenshot displays the Antenna House Regression Testing System interface, which compares two versions of a document (PDFs or directories of PDFs) and produces a PDF report of differences. The interface is divided into three main panels:

- Original Version Page:** Shows the original document content. It includes a title "Long and Complicated Scientific, Technical, or Medical Title" and a body of text with various sections like "Abstract", "Introduction", "Materials and Methods", and "Discussion".
- Difference Page:** Shows the differences between the two versions. Changes are highlighted in yellow. For example, in the "Introduction" section, the sentence "The high level of the gene expression was observed in the brain" is highlighted. In the "Materials and Methods" section, the sentence "The high level of the gene expression was observed in the brain" is highlighted. In the "Discussion" section, the sentence "The high level of the gene expression was observed in the brain" is highlighted.
- New Version Page:** Shows the new document content. It includes the same title and body of text as the original version, but with the changes from the difference page applied.

The interface also includes a sidebar on the left with navigation links and a footer with the Antenna House logo and contact information.

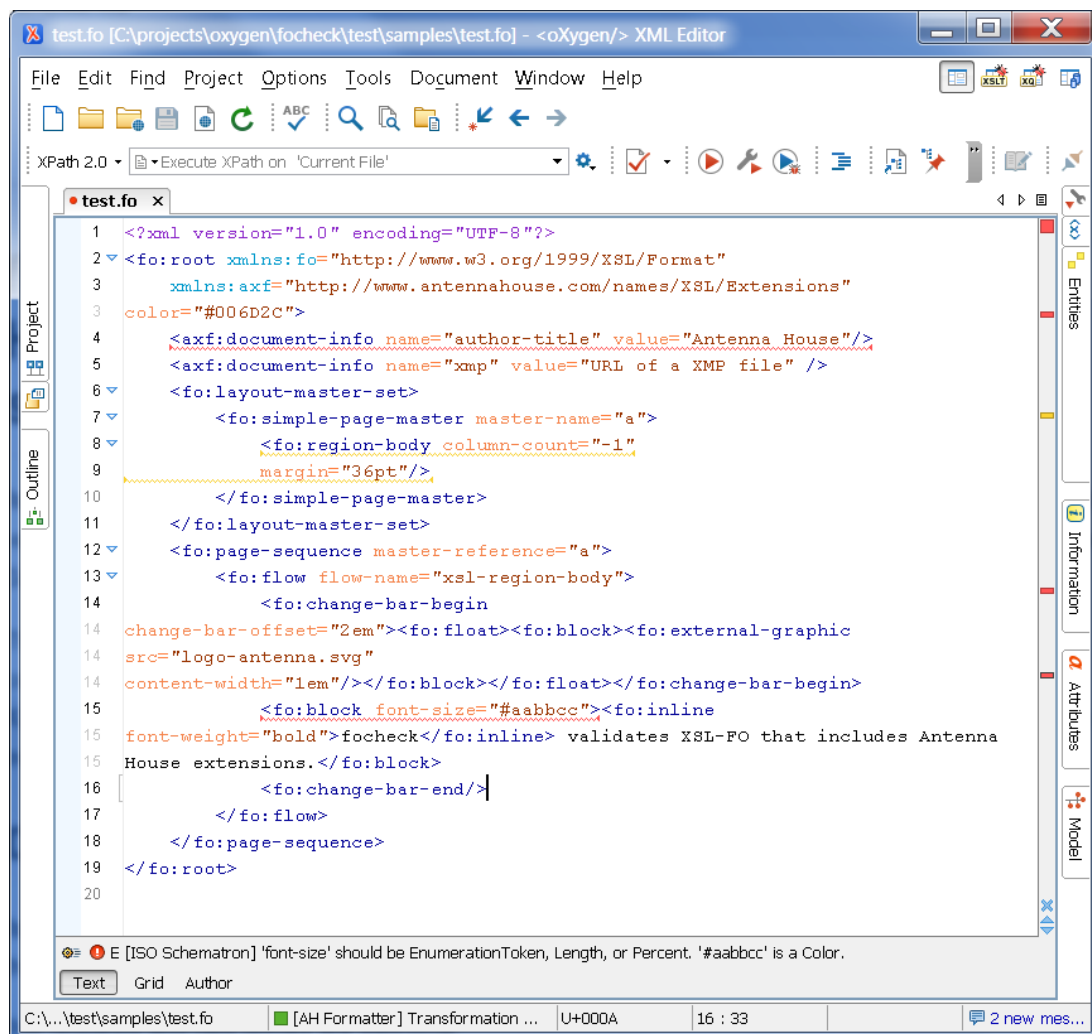
AHRTS is able to run on two PDFs or two directories of PDFs and produce a PDF report of differences between the files.

Validate with focheck

57

<https://github.com/AntennaHouse/focheck>

- Validate XSL-FO with Relax NG and Schematron
- oXygen plug-in or command-line



focheck, from Antenna House and available on GitHub, allows you to validate your generated XSL FO, including XSL FO with Antenna House extensions. You can install it as an oXygen plugin or you can use the Relax NG and Schematron as you would use any schema.

fopages

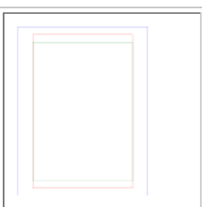
58

<https://github.com/MenteaXML/fopages>

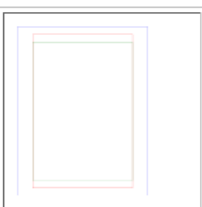
- Report on page masters used in FO file
- Work in progress

Page Masters

- back-even

Width	8.5in	
Height	11in	

- back-even-draft

Width	8.5in	
Height	11in	

- back-first

Page Sequence Masters

- back

Page master	Blank?	Odd or even?
blank	blank	
back-first		
back-odd		odd
back-odd		even

- back-draft

Page master	Blank?	Odd or even?
blank-draft	blank	
back-first-draft		
back-odd-draft		odd
back-odd-draft		even

- back

focheck produces a report of the page masters and page sequences masters used in an FO file.

Stylesheet Stage 4

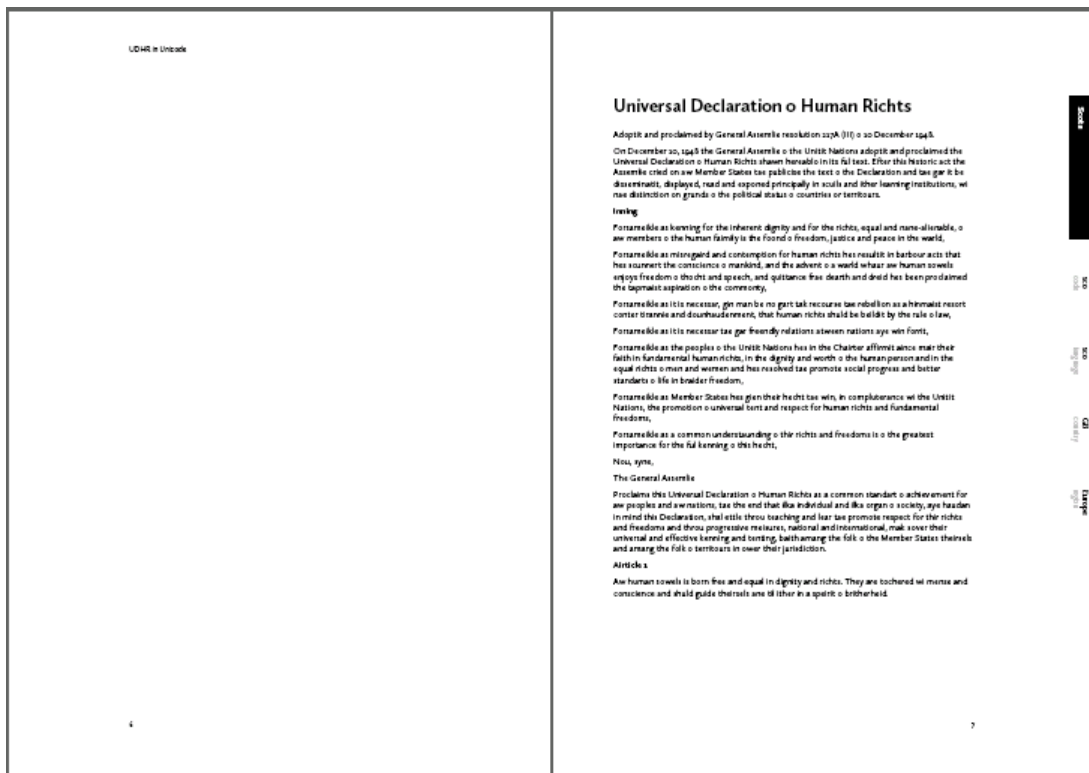
59

- Asymmetric page design
 - After Villard de Honnecourt, Paris, c. 1280
- Metadata in side region
- Lists item labels in margin
- 2248 pages for all translations

This is an example project without a specified design, so anything is possible if it improves the output. This next iteration changes the page layout, moves the list item labels into the margins, and puts some of the metadata for each translation in the side regions.

Asymmetric Page Design in Scots

60

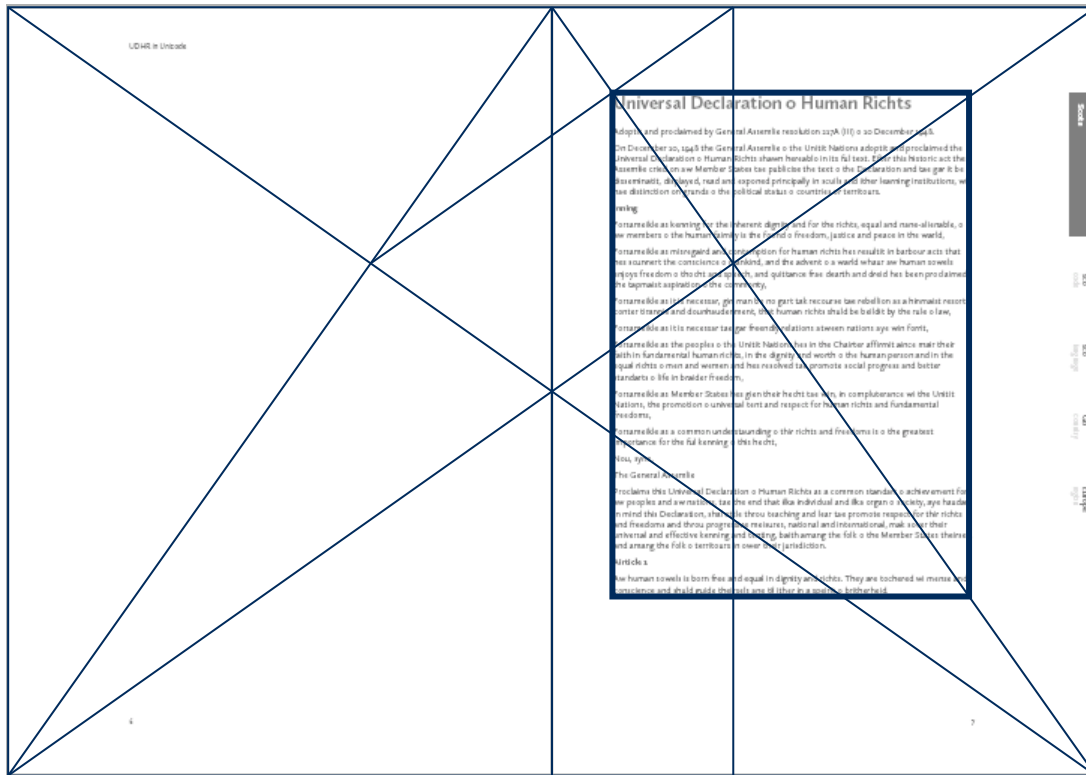


There are many classic, geometric page designs that are visually pleasing. Putting the text block 1in in from every edge of the page may be simple, but it's also very plain.

Villard de Honnecourt's Page Design

61

- Geometrically divide page
- From when straight lines were more accurate/repeatable than using measurements



All of these lines simply divide the page height and width into ninths. The inner and top margins are one-ninth, and the outer and bottom margins are two-ninths, of the page width and height.

Stuff In The Margins

62

- Language name, etc., from index.xml
- Could you tell every language without it?

on o Human Richts

mlie resolution 217A (III) o 10 December 1948.

mlie o the Unitit Nations adoptit and proclaimed the
nawn hereablo in its ful text. Efter this historic act the
e publicise the text o the Declaration and tae gar it be
d principally in scuils and ither learning institutions, wi nae
as o countries or territours.

dignity and for the richts, equal and nane-alienable, o aw
nd o freedom, justice and peace in the world,

tion for human richts hes resultit in barbour acts that hes
the advent o a world whaur aw human sowels enjoys
ance frae dearth and dreid hes been proclaimed the

Scots

SCO
code

To make the page more interesting, and because few people could tell every language/
script just by looking at them, some of the metadata from index.xml is placed on the
outer edge of each recto page.

Where Did We Get It?

63

- Metadata from index.xml
- You've seen this before
- Typical of metadata about some text

```
<udhrs
...
<udhr
  l='sco' iso639-3='sco'
  uli='' bcp47=''
  ohchr='sco' stage='4'
  pdf='y' notes='n'
  loc='59,-2' country='GB'
  region='Europe' demo='n'
  n='Scots' />
...
</udhrs>
```

The metadata on the outer edge is taken from the `udhr` element for the current language.

How Does It Get On The Page?

64

```
<fo:static-content flow-name="First-Outside">
  <fo:block-container reference-orientation="270">
    <fo:table>
      <fo:table-body>
        <fo:table-cell
          width="{ $body-before - $before-extent } in - 1em" />
        <fo:table-cell color="white"
          background-color="black" padding="3pt" padding-start="1em"
          font-weight="bold" width="2in" display-align="center">
          <fo:block><xsl:value-of select="$name" /></fo:block>
        </fo:table-cell>
        <fo:table-cell width="0.5in" />
        <fo:table-cell width="1in" padding="3pt">
          <fo:block><xsl:value-of select="$code" /></fo:block>
          <fo:block color="silver"
            font-size="small">code</fo:block>
        </fo:table-cell>
      </fo:table-body>
    </fo:table>
  </fo:block-container>
</fo:static-content>
```

The metadata appears in a table that is rotated so its top edge is along the outside edge of the page. The table is inside an `fo:static-content` in the FO file, and the FO processor uses the `fo:static-content` on pages that have a region named "First-Outside".

Page Regions In XSL FO

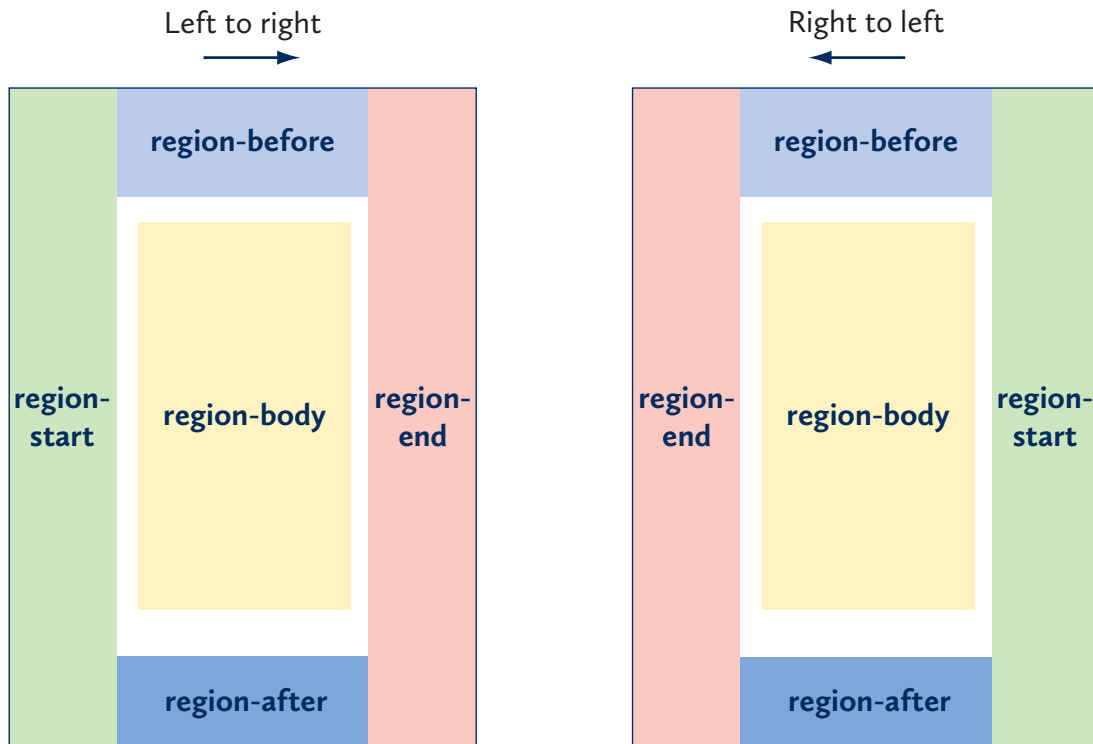
65



The `fo:simple-page-master` FO in XSL 1.1 defines the dimensions of a page. A document may have more than one `fo:simple-page-master`, and the same `fo:simple-page-master` may be used in multiple contexts. An `fo:simple-page-master` has up to five regions: `fo:region-body`, `fo:region-start`, `fo:region-end`, `fo:region-start`, and `fo:region-end`. Only `fo:region-body` is required.

Effect of Writing Mode

66



The FO names for the outer regions include “-before”, “-after”, “-start”, and “-end” rather than “-top”, “-bottom”, “-left”, and “-right” since the relative position of the regions depends on the writing mode.

This figure shows the arrangement of the regions for both “lr” (short for “tb-lr” for “top-to-bottom, right-to-left”) and “rl” (you can work it out) writing modes. As you can imagine, the arrangements would be different again for “tb-rl” and “bt-rl”, etc.

Regions Have Default Names

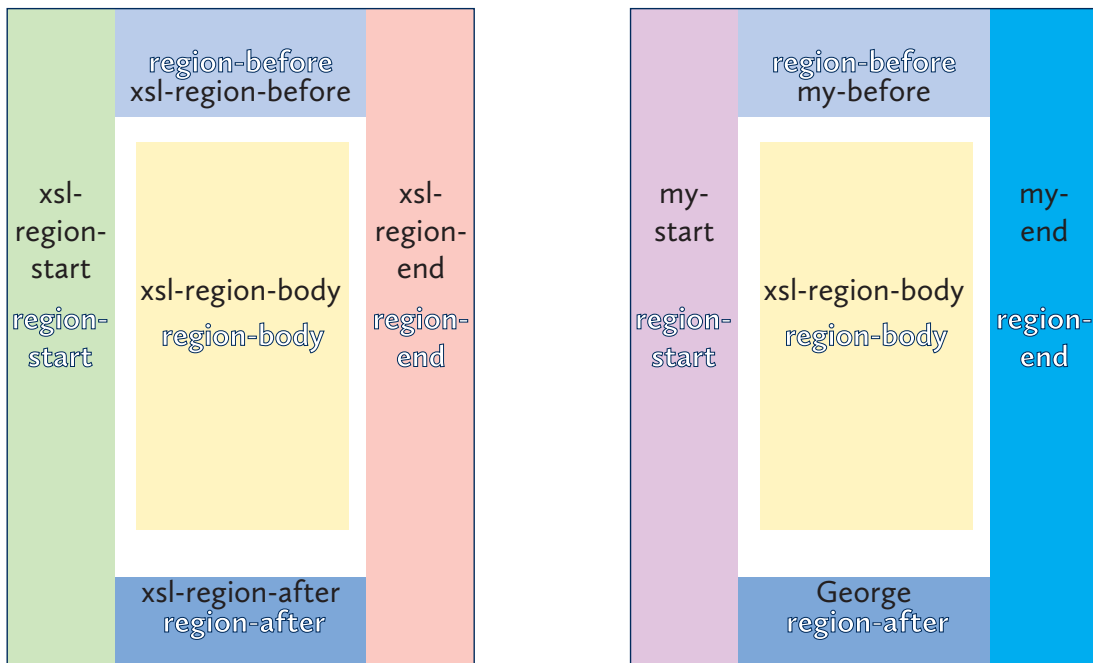
67



Regions have names, and content is directed to the page regions based on the region's name, not on its FO type (a.k.a. its "class"). This time, the figure shows the default, initial name for each page region FO, which just happens to look a lot like the FO's name, such as "xsl-region-body" for `fo:region-body`.

Or Use Your Own Names

68



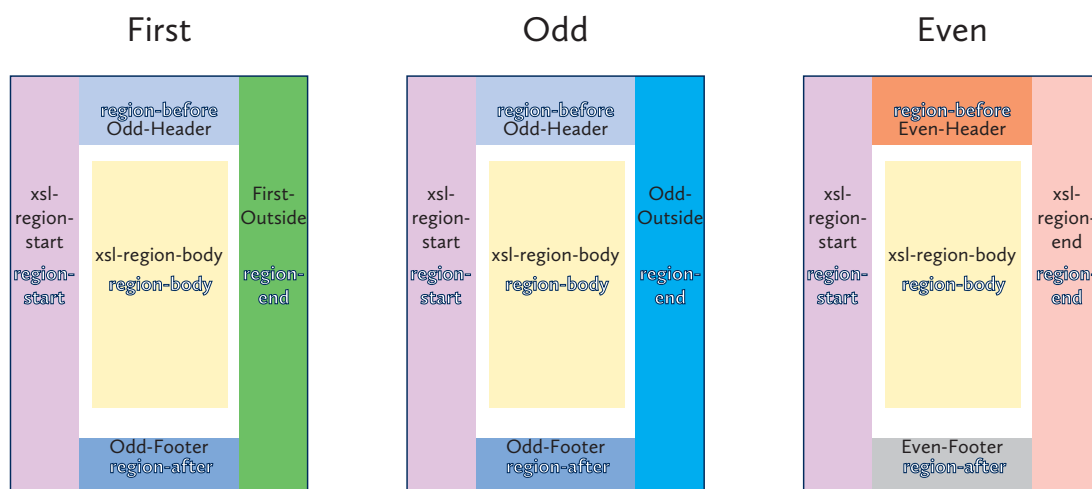
But you don't have to stick with the initial names. You can define the regions with any name, such as "my-before", "my-end", or even "George", provided it's:

- A valid name. It has to be, in XML terminology, a NCName, which means, e.g., it can't contain a colon and has to start with a letter (or other allowed character).
- Unique within the region names for the page master
- Not the initial name for a different class of region: i.e., you can't call your fo:region-before "xsl-region-after" and expect things to work

An additional proviso that works across page masters is that you can't use the same name on different region types on different page masters. So "George" can't be a fo:region-before on one page master and fo:region-start on another. Sorry, George.

Reusing Region Names Across Page Masters

69



The flip-side of the proviso above is that you can use the same region name for the same region on different page masters. This figure shows three page masters that will separately be used for:

- The first page in a page sequence;
- The odd pages; and
- The even pages.

Since (for conventional, “tb-lr” documents, anyway) the first page of the first page sequence is numbered “1” and appears on the right-hand (recto) side of a spread and (conventionally, again) following page sequences also start on an odd-numbered, recto page, the page master for the first pages often has the same header and/or footer as other pages or other odd-numbered pages. In this case, the “First” page master and the “Odd” page master have the same names for the same regions except `xsl:region-end`, which is named “First-Outside” on the “First” page master and “Odd-Outside” on the “Odd” page master. (Note that repetition of “First” and “Odd” in region names is for convenience only and isn’t a requirement of XSL-FO.)

Arrange Page Masters Into Sequence

70



When the multiple page masters are used to make a sequence of pages (using `fo:page-sequence-master`, as you do), you can see the repetition of the regions across the multiple pages. When overlaid over pages from the example, you can see the same footer in use on the odd pages, a different footer on the even pages, the different `fo:region-end` content on the first and the later odd pages, and (in this case) nothing at all in the `fo:region-start` of any page.

Content From fo:page-sequence

71

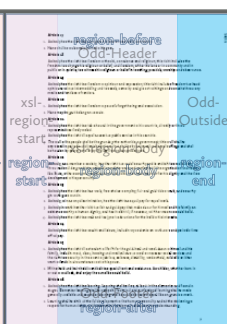
First



Even



Odd



Even



```

<fo:static-content flow-name="Even-Header">
  <fo:block>UDHR in Unicode</fo:block>
</fo:static-content>
<fo:static-content flow-name="First-Outside">
  <fo:block-container reference-orientation="270">
    ...
  </fo:block-container>
</fo:static-content>
<fo:static-content flow-name="Odd-Outside">
  <fo:block-container reference-orientation="270">
    ...
  </fo:block-container>
</fo:static-content>

```

```

<fo:static-content flow-name="Odd-Header">
  <fo:block>...</fo:block>
</fo:static-content>
<fo:static-content flow-name="Even-Footer">
  <fo:block><fo:page-number/></fo:block>
</fo:static-content>
<fo:static-content flow-name="Odd-Footer">
  <fo:block><fo:page-number/></fo:block>
</fo:static-content>
<fo:flow flow-name="xsl-region-body" id="sco">
  ...
</fo:flow>

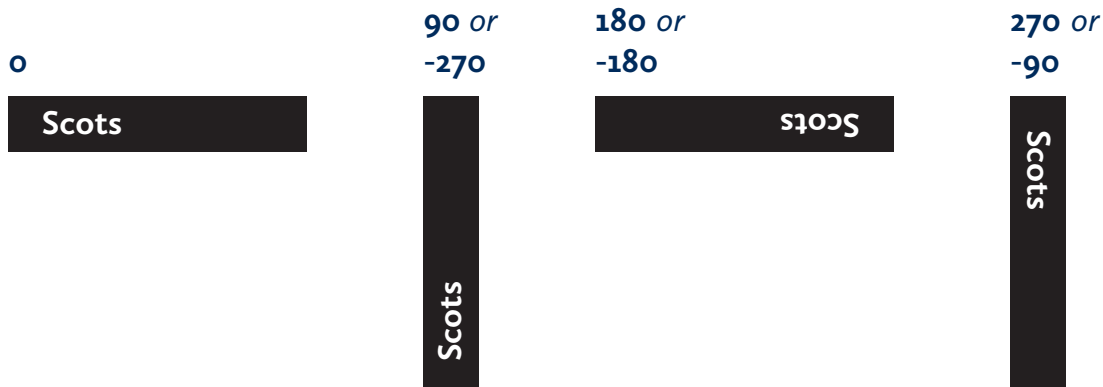
```

The content of the page regions, including the content of the `fo:region-body`, comes from the `fo:page-sequence`: (conventionally) the content of the outer regions are defined in `fo:static-content`, and the content of the `fo:region-body`, in the `fo:flow`. The correspondence between `fo:static-content` and an outer region is by name: put simply, when a page master for a page has a region with the same name as the "flow-name" property of an `fo:static-content` of the current `fo:page-sequence`, the areas from the formatting objects within the `fo:static-content` are repeated within that region on that page. (The not put simply version could see things rearranged through a XSL 1.1 `fo:flow-map` and multiple `fo:flow` with content directed to multiple regions on the page, but that's something for another time.)

Effect of reference-orientation

72

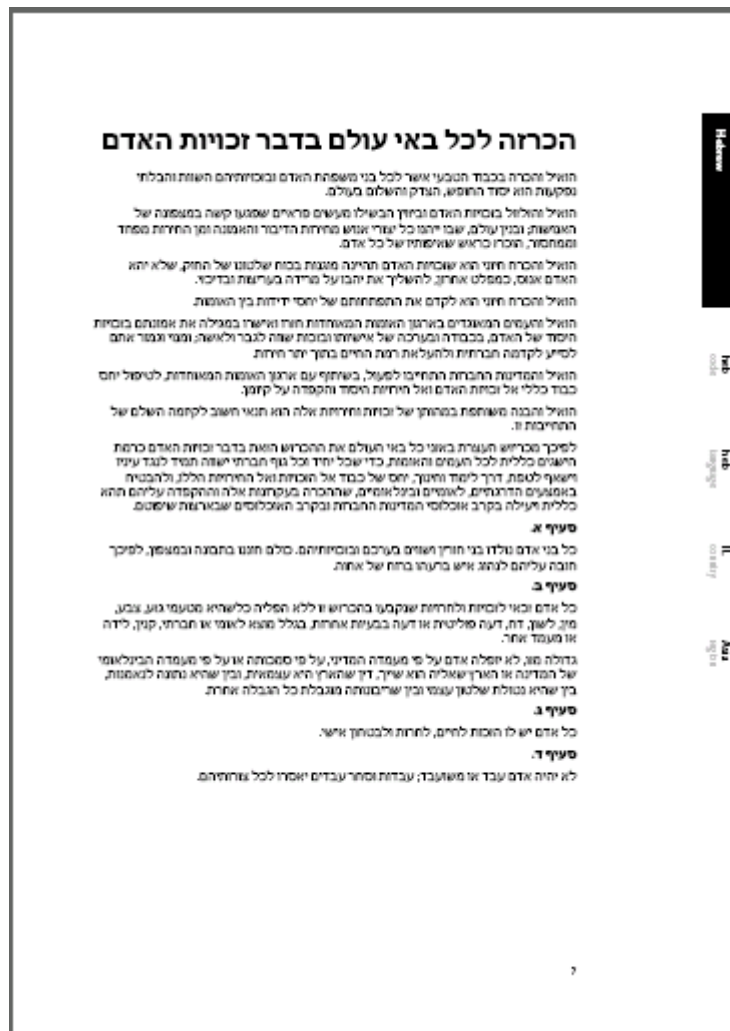
- Specifies direction of "top", "bottom", "left", and "right"
- Applies to all FOs that generate reference areas
- Change is relative to current orientation



“Outer” Edge vs. r | Writing Mode

73

- “start” and “end” reversed between l r and r | writing modes
- No “outer” region in XSL FO
- Changing writing mode in fo:block-container



Since “start” and “end” change with the writing mode, it became necessary to set the writing mode on the fo:region-body only.

Title Page and Copyright Notice

74

- Boilerplate
- Could add logo, etc.



Making Title and Copyright Pages

75

```
<fo:page-sequence master-reference="CoverFrontMaster"
  writing-mode="from-page-master-region()">
  <fo:flow flow-name="xsl-region-body">
    <fo:block space-after="72pt" space-after.precedence="1"
      font-size="4em" font-weight="bolder"
      text-align="center">UDHR in Unicode</fo:block>
    <fo:block-container height="100%" display-align="after">
      <fo:block>This is the full text of the Universal
      Declaration of Human Rights (UDHR), in one or more
      languages.</fo:block>
      <fo:block>© 1996-2007 The Office of the High
      Commissioner for Human Rights.</fo:block>
      <fo:block>This PDF version prepared from XML from the
      UDHR in Unicode project, <fo:basic-link
      external-destination="http://www.unicode.org/udhr"
      >http://www.unicode.org/udhr</fo:basic-link>.</fo:block>
    </fo:block-container>
  </fo:flow>
</fo:page-sequence>
```

The title page is generated from text that is hard-coded in the XSLT file.

Tables of Contents

76

- Language name, sorted alphabetically
- Title, sorted in default collation

Languages	
Arabic, Standard	7
Bengali	22
English	25
Hebrew	26
Japanese	27
Scots	27

Titles	
Universal Declaration of Human Rights	27
Universal Declaration of Human Rights	27
הכרזה לכל בני האדם בדבר זכויותיהם	27
الإعلان العالمي لحقوق الإنسان	27
世界人権宣言	27

Two tables of contents are generated.

Making “Languages” Table Of Contents

77

```
<fo:page-sequence master-reference="PageMaster"
  initial-page-number="auto-odd">
  <xsl:call-template name="PageMaster-static-content"/>
  <fo:flow flow-name="xsl-region-body">
    <fo:block space-after="24pt" space-after.precedence="1">
      <xsl:call-template name="toc-title">
        <xsl:with-param name="title">Languages</xsl:with-param>
      </xsl:call-template>
    </fo:block>
    <xsl:apply-templates
      select="udhr[@stage!=2]
        [$lang eq '#all' or
        (tokenize($lang,
          '¥s*') =
          string-join((@l, @v),
            '_'))]"
      mode="toc">
      <xsl:sort select="@n"/>
    </xsl:apply-templates>
  </fo:flow>
</fo:page-sequence>
```

“Languages” Table Of Contents Entry

78

```

<xsl:template name="toc-page-number">
  <xsl:param name="id" select="generate-id()"/>
  <xsl:text>&#xA0;&#xA0;</xsl:text>
  <fo:basic-link internal-destination="{ $id }">
    <fo:page-number-citation ref-id="{ $id }">
      xsl:use-attribute-sets="toc-number-attr"/>
    </fo:basic-link>
  </xsl:template>

<xsl:template match="udhr" mode="toc" priority="2">
  <fo:block xsl:use-attribute-sets="toc-block-attr">
    <fo:inline xsl:use-attribute-sets="toc-attr">
      <xsl:value-of select="@n"/>
    </fo:inline>
    <xsl:call-template name="toc-page-number">
      <xsl:with-param
        name="id"
        select="string-join((@l, @v), ' _ ')"
        as="xs:string" />
    </xsl:call-template>
  </fo:block>
</xsl:template>

```

Making “Titles” Tables Of Contents

79

```

<fo:flow flow-name="xsl-region-body">
  ...
  <fo:table>
    <fo:table-body>
      <xsl:apply-templates
        select="udhr[@stage!=2]
          [$lang eq '#all' or
            (tokenize($lang,
              ' ¥s*') =
              string-join((@l, @v),
                ' _ '))]"]
        mode="toc2">
      <xsl:sort
        select="document(concat(' udhr_',
          string-join((@l, @v),
            ' _ '),
            '.xml'))/
          udhr:udhr/udhr:title"/>
      </xsl:apply-templates>
    </fo:table-body>
  </fo:table>
</fo:flow>

```

“Titles” Table Of Contents Entry

80

```

<xsl:template match="udhr:udhr" mode="toc2">
  ...
  <fo:table-row>
    <fo:table-cell>
      <xsl:if test="$lr">
        <xsl:apply-templates select="udhr:title" mode="#current" />
      </xsl:if>
    </fo:table-cell>
    <fo:table-cell display-align="after">
      <fo:block text-align="center">
        <fo:basic-link internal-destination="{ $id }">
          <fo:page-number-citation ref-id="{ $id }">
            xsl:use-attribute-sets="toc-number-attr"/>
          </fo:basic-link>
        </fo:block>
      </fo:table-cell>
      <fo:table-cell>
        <xsl:if test="not($lr)">
          <xsl:apply-templates select="udhr:title" mode="#current" />
        </xsl:if>
      </fo:table-cell>
    </fo:table-row>
  </xsl:template>

```

When Are You Done?

81

- You may be done when:
 - There's no unhandled elements
 - There's no unhandled contexts
 - The pages look good

The definition of “done” depends on the particular project.

Are There Unhandled Elements?

82

- Catch-all template that highlights unhandled elements
- Make sure it has lowest import precedence!

```

<xsl:template match="*" mode="#all">
  <fo:wrapper color="red">
    <xsl:apply-templates mode="#current" />
  </fo:wrapper>
</xsl:template>

```

It is quite common to include a template in the XSLT transformation that makes unhandled text appear red (or draws attention to itself in some other way) in the formatted output.

More Bells, More Whistles

83

```
<xsl:param name="debug" select="' no' "/>

<xsl:param name="no-rule" select="' article,preamble' "/>

<xsl:variable name="no-rule-strings"
  select="if ($no-rule ne '')
    then tokenize($no-rule, ', *')
    else ()"/>

<xsl:template
  match="*[$debug eq 'yes']
    [not(local-name() = $no-rule-strings)]"
  mode="#all" priority="-0.5">
  <xsl:message>
    <xsl:text>No rule for '</xsl:text>
    <xsl:value-of select="local-name()" />
    <xsl:text>'</xsl:text>
  </xsl:message>
  <fo:wrapper color="red">
    <xsl:apply-templates mode="#current"/>
  </fo:wrapper>
</xsl:template>
```

You can make this as fancy as you want.

Have You Handled All Contexts?

84

- Behaviour that depends on:
 - Specific attribute values
 - Combinations of multiple attributes
 - Several levels of ancestry
 - Space before or after depending on type of siblings
- Starting from a good specification will help!

You may find that new unhandled contexts are appearing even after you've delivered the stylesheet.

Do The Pages Look Good?

85

You're on your own here.

Exercise 4 – Finishing Up

86

Add a template rule to find unimplemented elements.

Handle any problems that you find.

Summary

87

- Discovery
 - Finding what you're dealing with
- Mapping
 - Working out what to do
- Implementation
 - Doing it

References

88

- UDHR in Unicode – <http://www.unicode.org/udhr/>
- Wikipedia on UDHR – https://secure.wikimedia.org/wikipedia/en/wiki/Universal_Declaration_of_Human_Rights
- UDHR record – <http://www.guinnessworldrecords.com/records-1000/most-translated-document/>

Tools

89

- AH Formatter – <http://www.antennahouse.com/antenna1/formatter/>
- Antenna House Regression Testing System – <http://www.antennahouse.com/antenna1/antenna-house-regression-testing-system/>
- BaseX – <http://basex.org/>
- focheck – <https://github.com/AntennaHouse/focheck>
- fopages – <https://github.com/MenteaXML/fopages>
- Saxon XQuery from the command line – <http://www.saxonica.com/documentation/index.html#!using-xquery/commandline>
- Trang – <http://www.thaiopensource.com/relaxng/trang.html>

Appendix A

About

Tony Graham 55

Antenna House 55

Tony Graham

Tony Graham is a Senior Architect with Antenna House, where he works on their XSL-FO and CSS formatter, cloud-based authoring solution, and related products. He also provides XSL-FO and XSLT consulting and training services on behalf of Antenna House.

Tony 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. Tony is the developer of the 'stf' Schematron testing framework and also Antenna House's 'focheck' XSL-FO validation tool, a committer to both the XSpec and Juxy XSLT testing frameworks, the author of "Unicode: A Primer", and a qualified trainer.

Tony's career in XML and SGML spans Japan, USA, UK, and Ireland. Before joining Antenna House, he had previously been an independent consultant, a Staff Engineer with Sun Microsystems, a Senior Consultant with Mulberry Technologies, and a Document Analyst with Uniscope. He has worked 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, Japan, and Australia.

Antenna House

Antenna House is a global software company that works to make your documentation development and processing easier every single day. We were founded in 1984 in Tokyo but have spread our document formatting software throughout the world thanks to our growing staff in Beijing, Delaware and Maryland.

With 30 years of experience, we can offer you the best professional typesetting software and help you test all of these documents to ensure you're sending the right document every time.

The Antenna House Formatter is the most powerful XSL-FO software and CSS document formatting software on the market. To meet your needs, we have expanded AH Formatter to support more than 60 languages. This ensures that you'll always have the best and the only professional typesetting software you'll ever need. Fill the gaps in your processes with AH Formatter.

Exercise 1

Generate Schema for UDHR XML Files

Use Trang to generate a schema for all the udhr_*.xml files.

- Before running Trang, look at one or two of the UDHR XML files to get an idea of their structure.
- Run Trang on all the UDHR XML files to produce a schema in your preferred format
 - The extension of the output file – .dtd, .xsd, .rng, or .rnc – will determine the schema language
- Review the generated schema
- Did Trang find more than you did?
- Compare the generated schema with the official schema in schema.rnc
- Which schema will be more useful when writing a stylesheet? Why?
- Will the attribute types in the generated schema help you to know what you would have to handle?
- Does it tell you enough?

Exercise 2

Longest `udhr:listitem/@tag`

Find the length of the longest `udhr:listitem/@tag` value.

Exercise 3

Variable for list-item-label Width

- Some scripts use more than single numbers for list markers
- Chamula Tzotzil uses "Jun sloilal.", "Chib sloilal." and "Oxib sloilal."
- How would you handle the changing size of the list markers in the one document?

Exercise 4

Finishing Up

Add a template rule to find unimplemented elements.

Run the languages `eng`, `sco`, and `tzc`

Handle any problems that you find.

