6 XSL: Extensible Stylesheet Language

- An advanced style language for XML documents:
  1. Language for transforming XML documents: XSLT
  2. XML vocabulary (of formatting objects) for specifying formatting semantics:
     XSL version 1.0, W3C Rec. (15 October, 2001)

- 6.1 Introduction and Overview
- 6.2 Using XSL Formatting Objects

What is it?

- An XSL style sheet specifies the presentation of a class of XML documents
  - by describing an XSLT transformation of the XML document into an XML document that uses the formatting vocabulary
  - XSL FO: a markup language to describe formatting
- XSL builds on CSS2 and DSSSL
  - DSSSL a standardised but mainly unimplemented SGML style language

Example of XSL syntax

- Formatting paragraph elements (p):
  - NB: An incomplete style sheet!

```xml
<?xml version='1.0'?>
<xsl:stylesheet version='1.0'
xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
xmlns:fo="http://www.w3.org/1999/XSL/Format">
  <xsl:template match="p">
    <fo:block>
      <fo:initial-property-set
        font-variant="small-caps"/>
      <xsl:apply-templates/>
    </fo:block>
  </xsl:template>
</xsl:stylesheet>
```

6.1 Overview of XSL Formatting

- A style sheet processor accepts an XML document and an XSL style sheet, and produces a formatted presentation
- Two steps:
  1. tree transformation:
     XML source tree -> result tree (using XSLT)
  2. formatting
     - interpreting the result tree to produce formatted presentation

Transformation & Formatting

- Tree transformation adds information needed to format the result tree
- Formatting semantics expressed using a formatting vocabulary, of:
  - formatting objects (FOs), nodes of the result tree
  - for typographic abstractions like page-sequence, block, in-line text, page reference, ...
  - XSL 1.0 defines 56 formatting object classes
- formatting properties control the presentation of formatting objects (indents, spacing, fonts, ...)
Formatting

- Formatting-object tree interpreted to produce the representation
- Each FO specifies a part of pagination, layout and styling applied to its content
- Properties control the formatting of a FO
  - some directly, e.g., color
  - some through constraints, e.g., space-before, minimum
    -> rendered form not uniquely defined by XSL

Areas and Area Tree

- Formatting generates an area tree consisting of nested rectangular areas
  - inline areas (e.g., glyph areas) within line areas
  - lines within block areas
  - blocks within regions of a page
- Rendering causes the area tree to appear on a medium
  - areas printed on a sequence of sheets
    (or displayed on a single scroll in a browser)

Generating the Area Tree (1/3)

- Formatting a gradual and complex process
- Conceptual process of XSL formatting:
  - Element and attribute tree
    - target of transformation, source of formatting
    - consists of element, attribute, and text nodes
    - transformed into a …
  - Formatting object tree
    - consists of formatting objects with properties
    - more detailed: each character its own object

Generating the Area Tree (2/3)

Generating the Area Tree (3/3)

- Properties of the formatting object tree refined into traits
  - e.g., by propagating inherited properties, and computing absolute values for relative properties
    - font-size="12pt", start-indent="2em"
      become traits
    - font-size="12pt", start-indent="24pt"
    - traits control generation of areas out of formatting objects
    - some traits only available as a result of formatting, e.g., page numbers

Benefits of XSL

- an extensive model and vocabulary for expressing XML style sheets
- pagination and layout model extend existing ones
  - area model a superset of the CSS2 box model
    - e.g., different writing directions; footnotes, page number roles.
- support of non-western-language directions
  - distances specified in terms of before, after, start and end, relative to "writing-mode"
- powerful source selection and manipulation
  (with XPath/XSLT)
XSL Area Model

- Formatting objects generate areas
  - each 0 or more
    - page breaks -> additional block areas
    - line breaks -> additional line areas
- Each area tree node (except root) is associated to a rectangular portion of the output medium
- An area has a content-rectangle
  - portion for child areas
  - optionally surrounded by a border and padding

Two area types

- block-areas
  - generated in block-progression-direction (normally top-to-bottom)
  - paragraphs and titles normally rendered using fo:block, which creates block areas
  - line-area a special case: no borders or padding
- inline-areas
  - generated in inline-progression-direction (normally left-to-right)
  - characters rendered using fo:character, which generates glyph-area inline-areas
  - no child areas, a single glyph image as content

Formatting objects and properties

- XSL 1.0 defines 56 formatting objects ...
  - page-sequence, simple-page-master, block, inline, list-block, list-item, list-item-label, list-item-body, external-graphic, basic-link, first, footnote, ...
  - and 246 properties
    - master-reference, background-color, font-family, font-size, space-before, end-indent, text-align, text-indent, ...
    - many common with CSS2

Some central formatting objects 1/3

- fo:root
  - top node of the formatting object tree
  - a wrapper for all the rest
- fo:simple-page-master
  - used as a template for creating pages
  - specifies the geometry of pages
    - region-body (for page content)
    - region-before, region-after, region-start and region-end
      (for header, footer, and left and right sidebar)

Page regions

- A simple page can contain 1-5 regions, specified by child elements of the simple-page-master
Top-level formatting objects

- Slightly simplified:

  \[
  \text{fo:root} \quad \text{fo:layout-master-set} \quad \text{fo:page-sequence-set} \\
  (\text{fo:simple-page-master} \mid \text{fo:page-sequence-master})+ \\
  \text{fo:region-body} \quad \text{fo:region-before?} \quad \text{fo:region-after?} \\
  \text{fo:region-start?} \quad \text{fo:region-end?} \\
  \text{specify masters for page sequences by referring to simple-page-masters}
  \]

Some central formatting objects 2/3

- **fo:page-sequence**
  - specifies the creation of page sequences
  - possibly different page-sequence (and page-sequence-master) for, say, each chapter
- **fo:flow**
  - child objects of page-sequences
  - flows attached to regions of a page-master
- **NB**: No ‘page’-formatting objects
  - pages created by the formatter

Some central formatting objects 3/3

- **fo:block**
  - commonly used for paragraphs, titles, ...
  - may contain text, other blocks, or
  
  \[
  \text{fo:inline} \\
  \text{(to change properties, e.g.,} \text{font-style}\text{ of inline text)}
  \]
- **fo:table** for formatting tabular material
- **fo:list-block** to format lists of
  - **fo:list-item**s of
  
  \[
  \text{fo:list-item-label} \quad \text{and} \quad \text{fo:list-item-body}
  \]

“Hello world” result tree as XSL document

```
<fo:layout-master-set>
  <fo:simple-page-master master-name="page">
    <fo:region-body/>
  </fo:simple-page-master>
</fo:layout-master-set>
<fo:page-sequence master-reference="page"> <!--use 'page' master-->
  <fo:flow flow-name="xsl-region-body">
    <fo:block>Hello World</fo:block>
  </fo:flow>
</fo:page-sequence>
</fo:root>
```

Implementations?

- W3C XSL Recommendation rather recent
- What is the state of implementations?
- Some promising/interesting ones:
  - XEP by RenderX
    - Java-based XSL-FO to PS/PDF formatter
    - commercial (~$5000, April 2001); evaluation version free
  - Passive TeX
    - set of TeX macros to process XSL-FO by Sebastian Rahtz
  - Apache FOP
Apache FOP

- FOP (Formatting object to PDF) by J. Tauber
  - Top: a man who pays too much attention to his appearance
  - donated to XML Apache project
    (http://xml.apache.org/fop/)
  - open-source freeware
  - Java-based XML/XSL-FO to PDF (or MIF/PCL/TXT/...) processor
- Implements a useful subset of XSL 1.0 Rec;
  Version 0.20.3:
  - 41 formatting objects (out of 56)
  - 111 formatting properties (out of 246, or 228 w/o aural)

6.2 An XSL-FO Example

- From J. David Eisenberg: Using XSL Formatting Objects.
  XML.com, January 17, 2001, (acknowledging the loan of some graphics)
- XSL FOs for a version of a handbook of Spanish
  - Tedious to manually mark-up document instances with
    XSL formatting objects; Think of this as the result of an
    XSLT transformation
- Overall structure of: 
  - fo:root: specification of
    - page masters, followed by
    - the content of the pages

Example: Page dimensions and margins

```xml
<fo:layout-master-set>
  <fo:simple-page-master master-name="cover"
    page-height="12cm"
    page-width="12cm"
    margin-top="0.5cm"
    margin-bottom="0.5cm"
    margin-left="1cm"
    margin-right="0.5cm">
  </fo:simple-page-master>
</fo:layout-master-set>
```

Intended layout of pages

```
Cover       Left-hand Content Pages       Right-hand Content Pages

Page 1

Page 2
```

Page regions

- A simple page can contain 1-5 regions, specified by child
  elements of the simple-page-master

```
Page 1

Region-Top

Region-Body

Region-Bottom

Region-Edge

```

Let us refine the page masters by specifying regions

```
<fo:simple-page-master master-name="cover"
  page-height="12cm"
  page-width="12cm"
  margin-top="0.5cm"
  margin-bottom="0.5cm"
  margin-left="1cm"
  margin-right="0.5cm">
    <fo:region-body
      margin-top="3cm"
      margin-bottom="3cm">
    </fo:region-body>
</fo:simple-page-master>
```

Example: Region dimensions

```
<fo:simple-page-master master-name="leftPage"
  page-height="12cm"
  page-width="12cm"
  margin-top="0.5cm"
  margin-bottom="0.5cm"
  margin-left="1cm"
  margin-right="0.5cm">
  <fo:region-before extent="1cm"/>
  <fo:region-after extent="1cm"/>
  <fo:region-body
    margin-top="1.1cm"
    margin-bottom="1.1cm">
  </fo:region-body>
</fo:simple-page-master>
```

- NB: body uses all space inside page margins
  -> margins of region-body have to accommodate other regions!
Example: Page Sequences

- Next: masters for sequences of content pages, using the defined simple-page-masters – repeatedly alternate masters for left and right pages:

```xml
<fo:repeatable-page-master-alternatives>
  <fo:conditional-page-master-reference
      master-reference="leftPage"
      odd-or-even="even"/>
  <fo:conditional-page-master-reference
      master-reference="rightPage"
      odd-or-even="odd"/>
</fo:repeatable-page-master-alternatives>
```

- Other attributes of conditional-page-master-reference to select the page master to be used:
  - page-position="first" or "last", or "rest" (neither first or last), or "any"
  - blank-or-not-blank="blank"/"not-blank"

- Next: Specifying the sequences of content pages – by naming masters to be used, and attaching content flows to regions

Example: Contents of the Cover Page

```xml
<fo:flow flow-name="xsl-region-body">
  <fo:block font-family="Helvetica" font-size="18pt" text-align="end">
    Spanish Review Handbook
  </fo:block>
  <fo:block font-family="Helvetica" font-size="12pt" text-align="end" space-after="36pt">
    Copyright © 2001 J. David Eisenberg
  </fo:block>
  <fo:block text-align="end">
    A Catcode Production
  </fo:block>
</fo:flow>
```

Example: Cover Page Formatted

Formatting the first page-sequence gives ...

Example: Content Pages

- Finally, a page-sequence for content pages – with static-content for the header and footer, and a flow for page bodies:

```xml
<fo:flow flow-name="xsl-region-body">
  <fo:block font-family="Arial" font-size="10pt" text-align="center">
    Página
    <fo:page-number />
  </fo:block>
</fo:flow>
```

- Content for page footers:

```xml
<!-- static-content is repeated on every page -->
<fo:block font-family="Arial" font-size="10pt" text-align="center">
  Página
  <fo:page-number />
</fo:block>
```

- Finally, specify the content of page body:
Example: Content Pages Continue

- Assign a flow of blocks to region-body:

```xml
<fo:flow flow-name="xsl-region-body">
  <fo:block font-size="14pt">
    Watch this space!
  </fo:block>
</fo:flow>
```

- Formatting and rendering this gives ...

Example: Content Pages Formatted

Using FOs in Practise

- No one should write XSL FO document instances by hand
- Instead, use XSLT style rules to create formatting objects
  - root with layout masters for match="/"  
  - page-sequences with flows for major parts (like chapters, or the entire document):

```xml
<xsl:template match="chapter">
  <fo:page-sequence master-reference="...">
    <fo:flow flow-name="xsl-region-body"...
      <xsl:apply-templates/>
    </fo:flow>
  </fo:page-sequence>
</xsl:template>
```

Mapping content elements

- content elements would be mapped to blocks, inlines, list-blocks, tables, ... as appropriate
- For example, headers:

```xml
<xsl:template match="header">
  <fo:block font-size="14pt" font-family="sans-serif" font-weight="bold" color="green" space-before="6pt" space-after="6pt">
    <xsl:apply-templates/>
  </fo:block>
</xsl:template>
```

Examples of mapping content elements

- Formatting in-line emphasis:

```xml
<xsl:template match="strong">
  <fo:inline font-weight="bold">
    <xsl:apply-templates/>
  </fo:inline>
</xsl:template>
```

- More examples in the exercises

Summary

- XSL is a powerful (and complex) style language for XML documents  
  - allows arbitrary transformations of input documents  
  - allows fine-tuned specification of formatted representation
- It is a standard!  
  - well, almost: a W3C Recommendation  
  - emerging implementations seem promising