An advanced style language for XML documents:
1. Language for transforming XML documents: XSLT
2. XML vocabulary (= markup language) for specifying formatting: XSL 1.0. W3C Rec. (Oct 2001)
   » written for implementers of XSL processors
   » (XSL 1.1 Candidate Rec., Feb 2005)

6.1 Introduction and Overview
6.2 XSL Formatting by Example

Formatting paragraphs:
– NB: A fragment of a complete style sheet!

<xsl:stylesheet version='1.0' xmlns:xsl='http://www.w3.org/1999/XSL/Transform'>
xsl:template match='para[position() > 1]'>
  <fo:block text-indent='Tab'>
    <xsl:apply-templates/>
  </fo:block>
</xsl:template>
</xsl:stylesheet>

A style sheet processor accepts an XML document and an XSL style sheet, and produces a formatted presentation

Two steps:
1. (XSLT) transformation: XML source tree -> result tree
2. (XSL FO) formatting
   » interprets the result tree to produce formatted presentation

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, ...)
   » XSL 1.0 defines 248 formatting properties; many common with CSS

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 (→ final rendering is not unique)

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 as a single scroll in a browser)
Generating the Area Tree (1/3)

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

Generating the Area Tree (2/3)

XSL formatting (Area Tree Generation)

- additional block areas
- additional line areas

Generating the Area Tree (3/3)

- Properties of the formatting object tree refined into traits (muutolinjapuuri, piirre)
  - e.g., by propagating inherited properties, and computing absolute values for relative properties
  - e.g., properties
    - font-size="12pt", text-indent="2em"
    - become traits
    - font-size="12pt", text-indent="2em"
  - 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

- Rich model and vocabulary for XML stylesheets
- Powerful selection and manipulation (→ XSLT)
- Pagination and layout extend existing ones
  - area model a superset of the CSS2 box model
  - e.g., different writing directions: footnotes, page number refs.
- Support of non-western writing directions
  - distances expressed in terms of before/after (for block-progression-direction), and start/end (for inline-progression-direction)

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) 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

Content, Padding and Border

- For CSS-compatibility also
  - margin-top, -right, -bottom and -left
  - space-start/end not supported by FOP, and margins only for pages and their regions

Two area types

- block-areas
  - generated in block-progression-direction (normally top-to-bottom)
  - paragraphs and titles normally rendered using foblock, 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, single-page-master, block, inline, list-block, list-item, list-item-label, list-item-body, external-graphic, basic-link, float, footnote, table, table-row, table-column, …
- and 248 properties
  - master-reference, background-color, font-family, font-size, space-before, start-indent, 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**
  - model of the geometry of pages
    - region-body (for page content)
    - region-before (for header), region-after (for footer), region-start and region-end (for left and right sidebar)

Top-level formatting objects

- Slightly simplified:
  - **fo:root**
  - **fo:layout-master-set**
  - **fo:page-sequence**
  - **fo:region-body**
  - **fo:region-before**
  - **fo:region-after**
  - **fo:region-start**
  - **fo:region-end**
- **fo:flow**
  - child of a page-sequence
  - Attribute flow-name connects to a region with a matching region-name
  - > the contents is distributed to that region of pages
- **NB:** There are no ‘page’-formatting objects
  - pages created by the formatter

Page Regions

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

Content objects for pages

- Slightly simplified:
  - **fo:page-sequence**
  - **fo:static-content**
    - repeated on every page
  - **fo:flow**
    - (distributed to pages)
- **Block-level object**

Some block-level objects

- **fo:block**
  - commonly used for paragraphs, titles, ...
  - may contain text, other blocks, or
    - **fo:inline**
      - to change properties, e.g., font-style of inline text
- **fo:table**
  - for formatting tabular material
- **fo:list-block**
  - to format lists of
    - **fo:list-item**
      - with **fo:list-item-label**
      - and **fo:list-item-body**

“Hello world” result tree as an XSL document

```xml
<fo:root xmlns:fo="http://www.w3.org/1999/XSL/Format">
  <fo:layout-master-set>
    <fo:simple-page-master-master-name="page">
      <fo:region-body/>
    </fo:simple-page-master-master-name>
    <fo:flow name="xsl-region-body">
      Hello World</fo:block>
    </fo:flow>
  </fo:layout-master-set>
  <fo:flow name="xsl-page-body">
    <fo:block>Hello World</fo:block>
  </fo:flow>
</fo:root>
```

Implementations?

- W3C XSL Rec rather recent (10/2001)
- What is the state of implementations?
- Some promising/interesting ones (2004):
  - XEP by RenderX (XSL-FO to PS/PDF formatter), XSL Formatter by Antenna House
    - $0 to $5,000 (evaluation, ... server versions; April 2004)
  - Adobe Document Server
  - Passive TeX
    - set of TeX macros to process XSL-FO by Sebastian Rahtz
  - Apache FOP
Apache FOP

- FOP (Formatting Objects Processor) by J. Tauber
  - "fop: 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
  - Not complete, but implements a useful subset of XSL 1.0

---

FOP 0.20.5 XSL-FO-Compliance

<table>
<thead>
<tr>
<th>Implemented</th>
<th>fully</th>
<th>partially</th>
<th>no</th>
<th>total</th>
</tr>
</thead>
<tbody>
<tr>
<td>formatting</td>
<td>36</td>
<td>17</td>
<td>3</td>
<td>56</td>
</tr>
<tr>
<td>objects</td>
<td>(64%)</td>
<td>(30%)</td>
<td>(5%)</td>
<td></td>
</tr>
<tr>
<td>formatting</td>
<td>95</td>
<td>110</td>
<td>43</td>
<td>248</td>
</tr>
<tr>
<td>properties</td>
<td>(38%)</td>
<td>(44%)</td>
<td>(17%)</td>
<td></td>
</tr>
<tr>
<td>non-aural</td>
<td>95</td>
<td>110</td>
<td>25</td>
<td>230</td>
</tr>
<tr>
<td>properties</td>
<td>(41%)</td>
<td>(48%)</td>
<td>(11%)</td>
<td></td>
</tr>
</tbody>
</table>

---

6.2 XSL-FO by Example

- From J. David Eisenberg: Using XSL Formatting Objects. XML.com, January 17, 2001, (acknowledging the loan of some graphics)
- XSL FO instance for a handbook of Spanish
  - NB: XSL FO is not designed to be hand-authored
  - Consider this as a machine-generated result
  - (of an XSLT transformation)
- Overall structure of for:root: specification of
  - page masters, followed by
  - the content of the pages

---

Example: Region 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="0.5cm"
    margin-right="0.5cm">
    <fo:region-body
      margin-top="1.1cm"
      margin-bottom="1.1cm"/>
  </fo:simple-page-master>
</fo:layout-master-set>
```

---

Intended layout of pages

- Page regions

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

  - Let’s refine the page masters with regions

---

Example: Region dimensions

```xml
<fo:simple-page-master master-name="cover"
  ... dimensions and margins as above ...>
  <fo:region-body margin-top="1.1cm"/>
</fo:simple-page-master>
```

---

Layout of Page Regions

- 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:
    \[
    \begin{align*}
    \text{\textless page-sequence-master master-name=``contents``} & \text{\textgreater} \\
    \text{\textless conditional-page-master-reference master-reference=``leftPage`` odd-or-even=``even``} & \text{\textgreater} \\
    \text{\textless conditional-page-master-reference master-reference=``rightPage`` odd-or-even=``odd``} & \text{\textgreater} \\
    \end{align*}
    \]
  - /page-sequence-master

Example: Contents of the Cover Page

\[
\begin{align*}
\text{\textless page-sequence master-reference=``cover``} \\
\text{\textless forflow flow-name=``xsl-region-body``} \\
\text{\textless forblock font-family=``Helvetica`` font-size=``10pt`` text-align=end} \\
\text{\textless page after=``30pt`` space-after=``30pt``} \\
\text{\textless forblock font-family=``Helvetica`` font-size=``10pt`` text-align=end} \\
\text{\textless forblock text-align=end} \\
\text{\textless forflow} \\
\text{\textless /page-sequence}
\end{align*}
\]

Example: Content Pages Continue

- Finally, a page-sequence for content pages:
  - with static-content for the header and footer, and a flow for contents of pages:
    \[
    \begin{align*}
    \text{\textless page-sequence master-reference=``contents`` initial-page-number=``2``} \\
    \text{\textless static-content} \\
    \text{\textless forstatic-content flow-name=``xsl-region-before``} \\
    \text{\textless forblock font-family=``Helvetica`` font-size=``10pt`` text-align=center} \\
    \text{\textless forblock text-align=center} \\
    \text{\textless /forstatic-content}
    \end{align*}
    \]
  - Content for page footers:
    \[
    \begin{align*}
    \text{\textless static-content is repeated on every page} \longrightarrow \\
    \text{\textless forstatic-content flow-name=``xsl-region-after``} \\
    \text{\textless forblock font-family=``Helvetica`` font-size=``10pt`` text-align=center} \\
    \text{\textless forblock} \\
    \text{\textless /forstatic-content}
    \end{align*}
    \]
  - Finally, specify the content of the page body:

Example: Content Pages Formatted

Formatting and rendering this gives ...

Example: Cover Page Formatted

Formatting the first page-sequence gives ...

Example: Content Pages Continue

Assign a flow of blocks to region-body:

\[
\begin{align*}
\text{\textless forflow flow-name=``xsl-region-body``} \\
\text{\textless forblock font-size=``10pt``} \\
\text{Watch this space!} \\
\text{\textless /forblock} \\
\text{\textless /forflow}
\end{align*}
\]

Formatting and rendering this gives ...

Example: Cover Page Formatted

Watch this space!
Using FOs in Practise

- XSL FO instances should not be created manually
- Instead, use XSLT rules to create formatting objects
  - for:root with layout masters for match="/" (e.g., pages)
  - page-sequences with a flow for major parts (like chapters, or the entire document):
    <xsl:template match="chapter">
      <fo:page sequence-ref="...">
        <xsl:apply-templates/> </fo:page>
    </xsl:template>

Examples of mapping content elements

- Formatting in-line emphasis:
  <xsl:template match="emphasis">
    <fo:inline font-style="italic" style="italic">
      <xsl:apply-templates/>
    </fo:inline>
  </xsl:template>
- More 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
  - currently used mainly for producing PDF
    - browser support being expected ...

Expert Views on XSL

- "What is XSL-FO and When Should I Use It" in Seybold Report 2(17) (Dec. 02) by S. Deach, an XSL 1.0 co-author and computer scientist at Adobe:
  - "XSL-FO is now in the 'early-adopter' phase"
  - "It is expected that a wide variety of authoring tools become available [...] I expect a significant adoption [...] over a three-to-five year time frame"
  - "XSL-FO is best [...] in generating content-driven documents in response to individual customer requests"
  - "Today, XSL is most useful if you need to produce customer-tailored, paginated documents on a server."