Introduction to XSL

Max Froumentin - W3C

Introduction to XSL

Max Froumentin - W3C

- Formatting Objects:
- Properties
- Example: Play to FO
- Top-level Template
- I18N Formatting Objects and Properties
- Other Formatting Objects
- Example: mixed writing modes
- If you are still interested...

Introduction to XSL

In a nutshell: XSL is a W3C specification that describes a method for visually presenting XML documents.

This tutorial will cover:

- An overview of the XSL spec (including XSLT and XPath)
- Examples of various use cases
- Relationship with other XML technologies
- A detailed example

These slides are available at
http://www.w3.org/People/maxf/XSLideMaker/

XML Documents

- XML (eXtensible Markup Language) adds information to text files, using **tags** and **attributes** [example1], [example2]
- Tag names are defined for a specific document type.
- Uses the Unicode character set
- Designed to be easily processed by machine while remaining readable.

Styling XML Documents

- XML documents are ideally semantic. For example, this bit of HTML is wrong:

```
Do not <strong>smoke</strong>, <it>eat</it> or
 <blink>drink</blink> in this room.
```

- Presentation data should be separate
- Two solutions for styling: CSS (Cascading Style Sheets) and XSL (eXtensible Stylesheet Language).

CSS

```
TITLE {
   display: block;
   font-family: Helvetica;
   font-size: 18pt
 }
```

Simple model: properties are associated to tags or attributes.

XSL

XSL is an alternative to CSS that allows greater control over the presentation of the XML data.

What can it do?

- [like CSS] allow changing presentation without changing the XML source, and display documents on various media,
- also: I18N features (writing modes, text alignment, hyphenation), complex page layout, footnotes, automatic generation of content (index)

Who is it for?

Applications that require high-level quality formatting:

- Publishing industry (books, technical documentation)

- Publication on different media: paper, web, mobile devices .

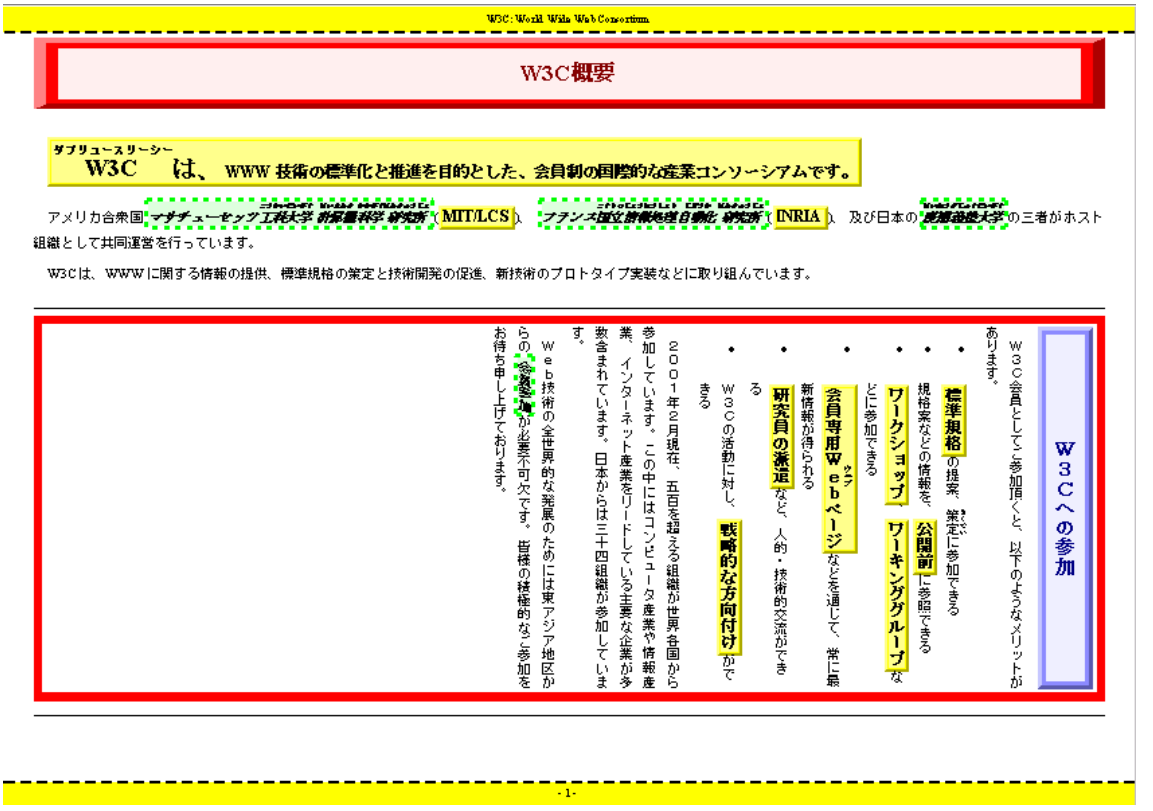
But is it not meant to be used where presentation is deeply tied to the contents (like graphic design).

Example I: Hamlet

```
<ACT>
   <SCENE>
     <TITLE>A room in the castle.</TITLE>
     <STAGEDIR>
       Enter KING CLAUDIUS, QUEEN GERTRUDE,
       POLONIUS, OPHELIA, ROSENCRANTZ, and
       GUILDENSTERN
     </STAGEDIR>
   <SPEECH speaker="King Claudius">
       <LINE>And can you, by no drift of circumstance,</LINE>
       <LINE>Get from him why he puts on this confusion,</LINE>
       <LINE>Grating so harshly all his days of quiet</LINE>
       <LINE>With turbulent and dangerous lunacy?</LINE>
   </SPEECH>
 ...
```

Formatted for paper output (PDF), formatted for the Web (XHTML)

Example II: Mixed Writing Modes



W3C: World Wide Web Consortium

W3C概要

ダブリュースリーシー
W3C は、 WWW 技術の標準化と推進を目的とした、会員制の国際的な産業コンソーシアムです。

アメリカ合衆国 マサチューセッツ工科大学 計算機科学 研究所（MIT/LCS）、フランス国立情報処理自動化 研究所（INRIA）、及び日本の 慶應義塾大学 の三者がホスト組織として共同運営を行っています。

W3Cは、WWWに関する情報の提供、標準規格の策定と技術開発の促進、新技術のプロトタイプ実装などに取り組んでいます。

W3Cへの参加

W3C会員としてご参加頂くと、以下のようなメリットがあります。

- 標準規格の提案、策定に参加できる
- 規格案などの情報を、公開前に参照できる
- ワークショップ ワーキンググループなどに参加できる
- 会員専用Webページなどを通じて、常に最新情報が得られる
- 研究員の派遣など、人的・技術的交流ができる
- W3Cの活動に対し、戦略的な方向付けができる

２００１年２月現在、五百を超える組織が世界各国から参加しています。この中にはコンピュータ産業や情報産業、インターネット産業をリードしている主要な企業が多数含まれています。日本からは二十四組織が参加しています。

Web技術の全世界的な発展のためには東アジア地区からの参加が必要不可欠です。皆様の積極的なご参加をお待ち申し上げております。

- 1-

Example III: database

```
...
   <record year="1992">
     <artist>Sundays, The</artist>
     <title>Blind</title>
   </record>
   <record year="1994">
     <artist>(Various)</artist>
     <title>The Glory of Gershwin</title>
     <note>Compilation</note>
   </record>
   <record type="soundtrack" year="1992">
     <artist>Kamen, Michael</artist>
     <title>Brazil</title>
     <location folder="3" page="20"/>
   </record>
 ...
```

PDF

Other Examples

- W3C specs (one DTD, output formats: HTML, sliced HTML, text, PDF), e.g MathML 2.0, XML 1.0
- This slideshow

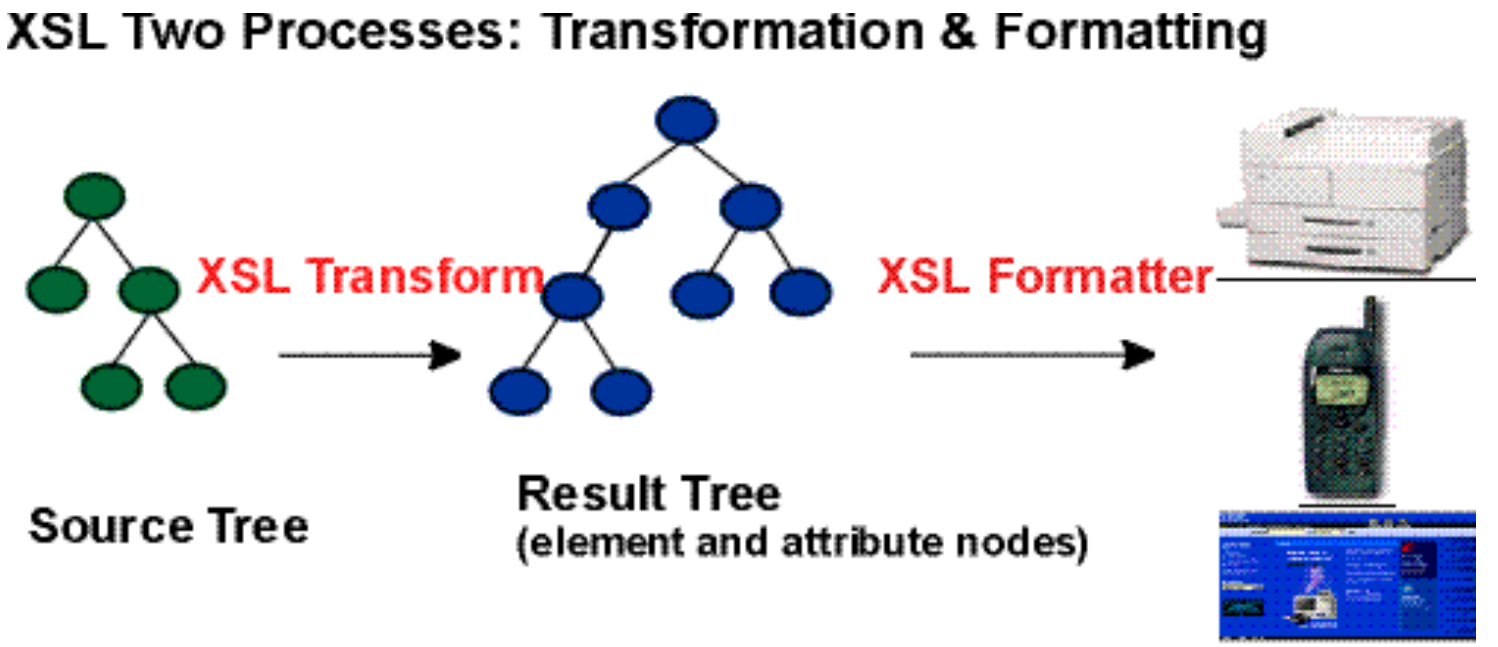
How do they do that?

- An XSL stylesheet is an XML File
- It is associated to an XML document with a **Stylesheet Processing Instruction** (like CSS)

```
<?xml-stylesheet ref="shakespeare.xsl" type="text/xsl"?>
 <ACT>
    <SCENE>
      <TITLE>A room in the castle.</TITLE>
```

- The actual formatting is performed either off-line or on the browser

The XSL Process(es)



The result tree is an XML document in which the markup has information about how to display the document: what font to use, the size of a page, etc. This markup is called **Formatting Objects**

(elements) and **Properties** (attributes). For example:

```
<block font-family="Helvetica">ACT III</block>
 <block font-size="10pt">Scene 1: A room in the castle</block>
 <block space-before="10mm" font-style="italic">
   Enter KING CLAUDIUS, QUEEN GERTRUDE, POLONIUS,
   OPHELIA, ROSENCRANTZ, and GUILDENSTERN
 </block>
 ...
```

Generated from:

```
<ACT>
   <SCENE>
     <TITLE>A room in the castle.</TITLE>
     <STAGEDIR>
       Enter KING CLAUDIUS, QUEEN GERTRUDE,
       POLONIUS, OPHELIA, ROSENCRANTZ, and
       GUILDENSTERN
     </STAGEDIR>
 ...
```

Server-Side/Client-Side XSL

- Off-line (e.g. for printing)
- Server-side:

  server transforms, client renders (not recommended)

- Client-side:

  client transforms and renders (allows user styles)

XSL and other W3C specs

XSL uses CSS properties to express formatting information, and uses the CSS inheritance model.

- CSS:

```
TITLE {
   display: block;
   font-family: Helvetica;
   font-size: 14pt;
 }
```

- XSL:

```
<xsl:template match="TITLE">
   <fo:block font-family="Helvetica" font-size="14pt">
     [...]
   </fo:block>
 </xsl:template>
```

XSL and SVG, MathML

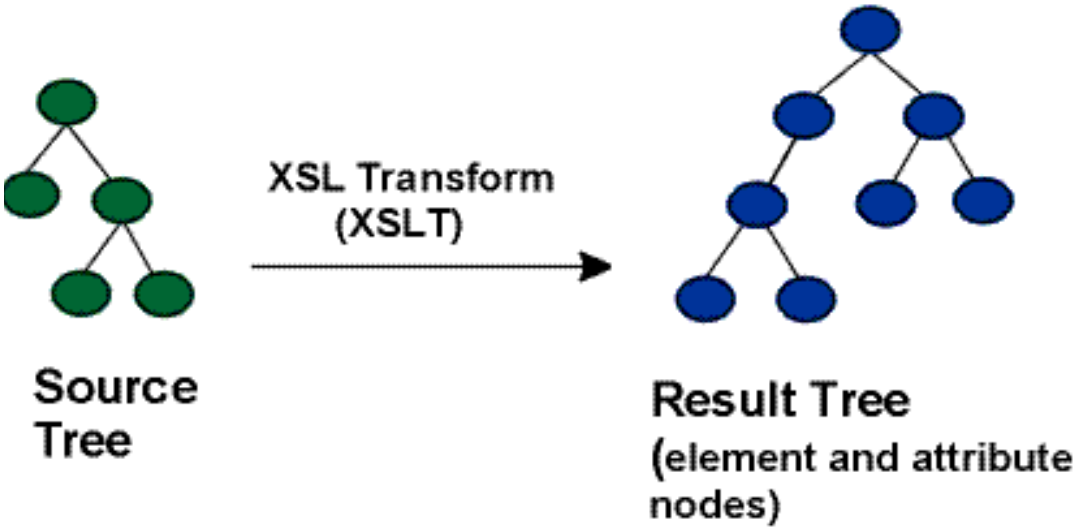
- XSL can import images and other types of known XML

documents: SVG and MathML.

- Up to the renderer to handle other namespaces

Transformations: XSLT

XSLT is a transformation language originally designed to transform any XML document into another XML document containing formatting objects: pages, blocks, graphics, text, etc.



General-purpose XSLT

XSLT has evolved to become a general-purpose transformation language from XML to XML.

Many users use it to transform their own XML document type to HTML for viewing within a browser

- XSLT stylesheets use XML syntax
- A stylesheet is a list of **templates**

```
<?xml version="1.0" encoding="utf-8"?>
 <xsl:stylesheet xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
                 version="1.0">
   <xsl:template match="/">...</xsl:template>
   <xsl:template match="/html">...</xsl:template>
 </xsl:stylesheet>
```

Templates

- Each template applies to a type of nodes in the input document
- When matches are made, the templates contains desired output.

```
<xsl:template match="TITLE">
   <fo:block font-family="Helvetica" font-size="14pt">
     <xsl:apply-templates/>
   </fo:block>
 </xsl:templates>
```

So this will transform:

```
<TITLE>Hamlet</TITLE>
```

into

```
<fo:block font-family="Helvetica" font-size="14pt">
   Hamlet
 </fo:block>
```

HTML can also be generated very simply in the template, using for instance `<h1>` instead of `<fo:block>`

`<xsl:apply-templates/>` means: apply other templates to contents.

Implicit rule: text is copied from input to output: a style sheet with no rules will only return the character data of the input.

XSLT statements

Allow navigation and iteration within the input document tree

- `<xsl:value-of select="..."/>`

  Gets a value (node contents or attribute) from the input tree.

- `<xsl:for-each select="...">`

  Loops over the nodes in the select expression

- `<xsl:if test="...">...</xsl:if>`

  Conditional

"Play" to HTML

Very simple one-template example using the 'pull' method:

```
<?xml version="1.0" encoding="utf-8"?>
 <html xmlns="http://www.w3.org/1999/xhtml"
       xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
       xsl:version="1.0">
   <head>
     <title><xsl:value-of select="PLAY/TITLE"/></title>
   </head>
   <body>
     <h1><xsl:value-of select="PLAY/TITLE"/></h1>
     <xsl:for-each select="PLAY/ACT">
       <xsl:for-each select="SCENE">
         <xsl:if test="TITLE">
           <h2><xsl:value-of select="TITLE"/></h2>
         </xsl:if>
         <xsl:for-each select="SPEECH">
           <h3 style="color: red"><xsl:value-of select="SPEAKER"/></h3>
           <xsl:for-each select="LINE">
             <p><xsl:value-of select="."/></p>
           </xsl:for-each>
         </xsl:for-each>
       </xsl:for-each>
     </xsl:for-each>
```

```
   </body>
 </html>
```

Result:

```
<?xml version="1.0" encoding="utf-8"?>
   <html xmlns="http://www.w3.org/1999/xhtml">
     <head>
       <title>The Tragedy of Hamlet, Prince of Denmark</title>
     </head>
     <body>
       <h1>The Tragedy of Hamlet, Prince of Denmark</h1>
       <h2>Elsinore. A platform beforethe castle.</h2>
       <h3 style="color: red">BERNARDO</h3>
       <p>Who's there?</p>
       <h3 style="color: red">FRANCISCO</h3>
       <p>Nay, answer me: stand, and unfold yourself.</p>
 ...
```

Extended, output: numbering, TOC, etc.

This uses the 'push' method where structure follows the input.

"Play" to HTML

Roughly there is one template for each tag type in the input

XPath

- ... is another W3C specification;
- an expression language to selects parts of an XML document tree;
- is used in `<xsl:template match="...">` or `<xsl:value-of select="...">`, etc.;
- and can be as simple as `TITLE`, or as complex as

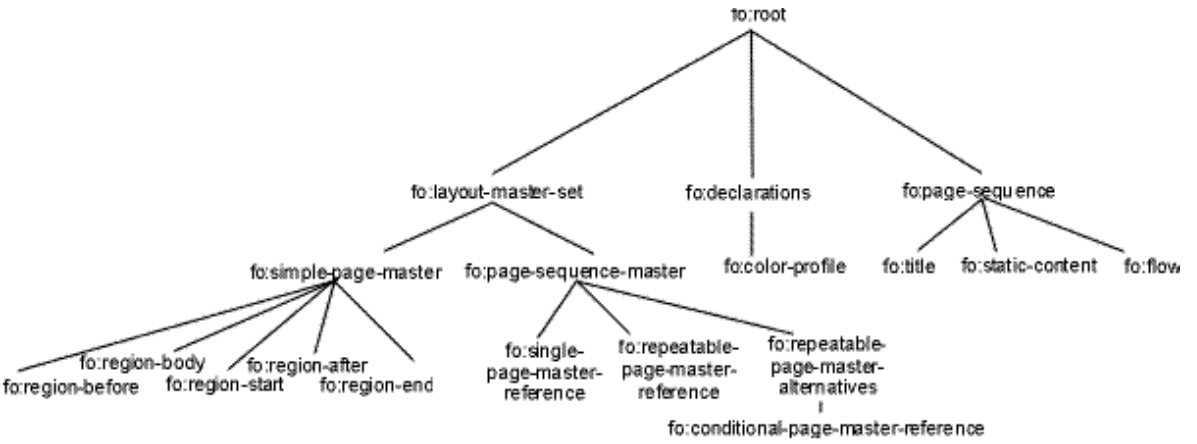
```
/ACT[3]/SCENE[position() &lt; 5

and position() &gt;
2]/SPEAKER[@name="Hamlet"]/

LINE[contains(.,"shoe box")]
```

Formatting Objects basics

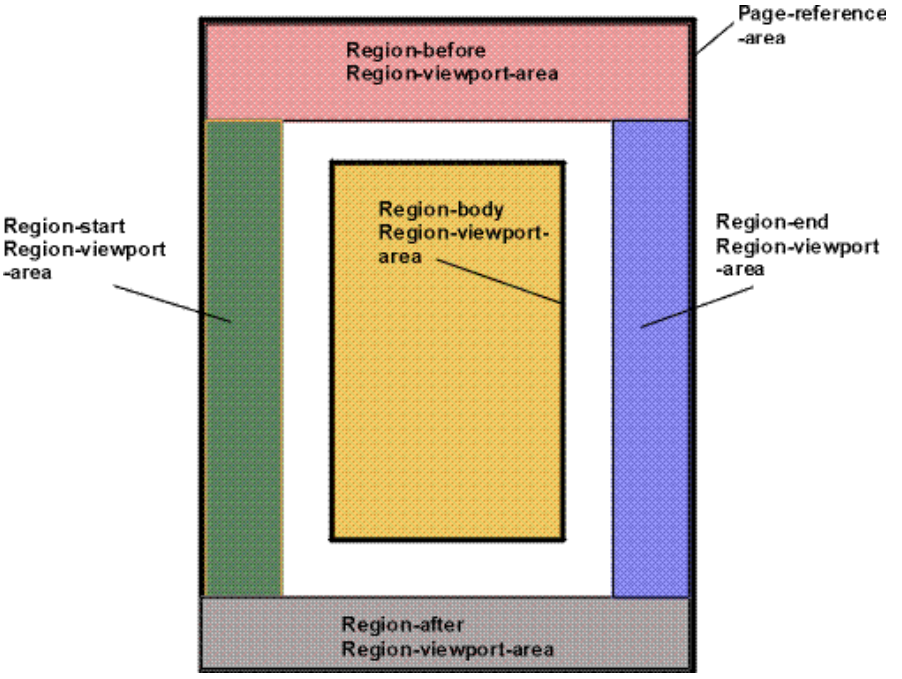
- the FO vocabulary is one special type of output from XSLT
- FOs are organized as an XML tree:



- Each node has associated **properties**, either directly specified (by attributes) or inherited.

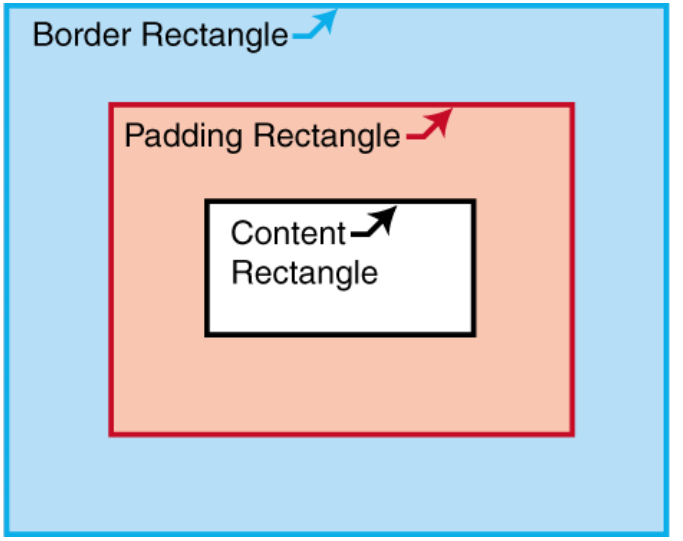
Pages

- A page is divided in 5 regions: body, before, after, start and end



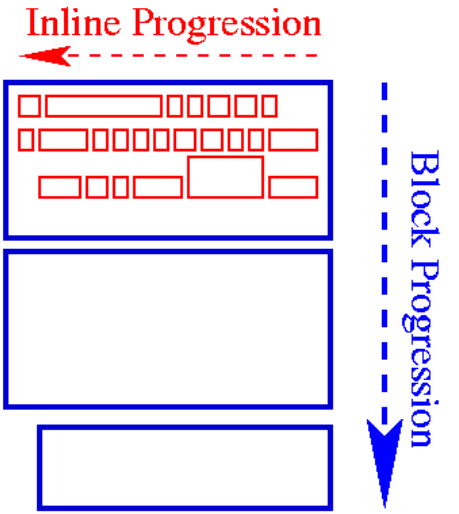
The area model

On the page will be layed out areas, that contain text, images and other areas. An area is a rectangle, with padding and border:



Block/inline areas

The concept of relative orientation and writing-modes. Where CSS defines top, bottom, left, right, XSL adds before, after, start and end. Areas can be of type: block or inline. Blocks are stacked from the 'before' side to the 'after' side, inlines are stacked orthogonally.



Formatting Objects:

- Define the layout
  fo:layout-master-set
  fo:page-master
  fo:page-sequence
- Generate areas
  fo:block
  fo:inline
  fo:character
- Other
  fo:page-number
  fo:external-graphics

Properties

- Each area has a set of traits: color, background, font-size, etc.
- Areas are inherited down the FO tree using the CSS inheritance model
- They are specified in the source as attributes associated to Formatting Objects.

```
<fo:block font-family="Helvetica" font-size="14pt">
   This is Helvetica 14pt text.
   <fo:block font-size="200%">
     This is Helvetica 28pt text.
   </fo:block>
 </fo:block>
```

Example: Play to FO

```
<?xml version="1.0" encoding="utf-8"?>
 <xsl:stylesheet
   xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
   xmlns:fo="http://www.w3.org/1999/XSL/Format"
   version="1.0">
 <!-- ************************************************** -->
   <xsl:output method="xml" indent="yes"/>
   <xsl:template match="/">
     <xsl:apply-templates/>
   </xsl:template>
   <xsl:template match="*">
     <xsl:apply-templates/>
   </xsl:template>
   <xsl:template match="PLAY">
     <fo:root>
       <fo:layout-master-set>
         <fo:simple-page-master master-name="title-page"
           page-width="210mm" page-height="297mm"
           margin-top="2cm" margin-bottom="2cm"
           margin-left="2cm" margin-right="2cm">
           <fo:region-body region-name="body"/>
         </fo:simple-page-master>
         <fo:simple-page-master master-name="act-page"
           page-width="210mm" page-height="297mm"
           margin-top="2cm" margin-bottom="2cm"
```

```
          margin-left="2cm" margin-right="2cm">
          <fo:region-body region-name="body" margin-top="1cm"
margin-bottom="1cm"/>
          <fo:region-before extent="1cm" region-name="header"/>
          <fo:region-after extent="1cm" region-name="footer"/>
        </fo:simple-page-master>
      </fo:layout-master-set>
      <fo:page-sequence master-name="title-page">
        <fo:flow flow-name="body">
          <fo:block display-align="center">
            <xsl:apply-templates select="TITLE"/>
            <xsl:apply-templates select="FM"/>
          </fo:block>
        </fo:flow>
      </fo:page-sequence>
      <fo:page-sequence master-name="title-page">
        <fo:flow flow-name="body">
          <fo:block space-before="5cm">
            <fo:block font-size="16pt" space-after="3em"
text-align="center">
              <xsl:text>Table of Contents</xsl:text>
            </fo:block>
            <fo:block start-indent="3cm" font-size="14pt">
               <xsl:apply-templates select="ACT" mode="toc"/>
            </fo:block>
```

Example: Play to FO

```
        </fo:block>
      </fo:flow>
    </fo:page-sequence>
    <fo:page-sequence master-name="title-page">
      <fo:flow flow-name="body">
        <xsl:apply-templates select="PERSONAE"/>
      </fo:flow>
    </fo:page-sequence>
    <fo:page-sequence master-name="title-page">
      <fo:flow flow-name="body">
        <xsl:apply-templates select="SCNDESCR"/>
      </fo:flow>
    </fo:page-sequence>
    <xsl:apply-templates select="ACT"/>
  </fo:root>
</xsl:template>
<xsl:template match="PLAY/TITLE">
  <fo:block text-align="center"
            font-size="30pt"
            space-before="1em"
            space-after="1em">
    <xsl:apply-templates/>
  </fo:block>
</xsl:template>
<xsl:template match="TITLE">
```

```
    <fo:block text-align="center"
              font-size="20pt"
              space-before="1em"
              space-after="1em">
      <xsl:apply-templates/>
    </fo:block>
  </xsl:template>
  <xsl:template match="ACT/TITLE">
    <fo:block id="{generate-id()}"
              text-align="center"
              font-size="20pt"
              space-before="1em"
              space-after="1em">
      <xsl:apply-templates/>
    </fo:block>
  </xsl:template>
  <xsl:template match="SCENE/TITLE">
    <fo:block text-align="center"
              font-size="16pt"
              space-before="1em"
              space-after="1em">
      <xsl:apply-templates/>
    </fo:block>
  </xsl:template>
  <xsl:template match="FM">
```

Example: Play to FO

```
    <fo:block text-align="center"
              font-size="10pt"
              space-before="1em"
              space-after="1em">
      <xsl:apply-templates/>
    </fo:block>
  </xsl:template>
  <xsl:template match="PERSONAE/PERSONA | PERSONAE/PGROUP">
    <fo:block space-after=".5em"><xsl:apply-templates/></fo:block>
  </xsl:template>
  <xsl:template match="PERSONAE/PGROUP/PERSONA">
    <fo:block><xsl:apply-templates/></fo:block>
  </xsl:template>
  <xsl:template match="GRPDESCR">
    <fo:block start-indent="5mm"><xsl:apply-templates/></fo:block>
  </xsl:template>
  <xsl:template match="SCNDESCR">
    <fo:block text-align="center"
              font-size="20pt">
      <xsl:apply-templates/>
    </fo:block>
  </xsl:template>
  <xsl:template match="SCENE">
    <fo:block
      id="{generate-id()}"
```

```
      font-size="20pt"
      space-before.optimum="10pt" space-after.optimum="5pt"
      text-align="center">
      <xsl:text>Scene </xsl:text>
      <xsl:number/>
    </fo:block>
    <xsl:apply-templates/>
  </xsl:template>
  <xsl:template match="ACT">
    <fo:page-sequence master-name="act-page">
      <fo:static-content flow-name="header">
        <fo:block text-align="end">
          <xsl:value-of select="/PLAY/PLAYSUBT"/>
          <xsl:text> - Act </xsl:text>
          <xsl:number format="I"/>
        </fo:block>
      </fo:static-content>
      <fo:static-content flow-name="footer">
        <fo:block text-align="end">
          <fo:page-number/>
        </fo:block>
      </fo:static-content>
      <fo:flow flow-name="body">
        <fo:block id="{generate-id()}"
          font-size="24pt"
```

Example: Play to FO

```
          space-before.optimum="10pt" space-after.optimum="5pt"
          text-align="center">
          <xsl:text>Act </xsl:text>
          <xsl:number format="I"/>
        </fo:block>
        <xsl:apply-templates/>
      </fo:flow>
    </fo:page-sequence>
  </xsl:template>
  <xsl:template match="ACT" mode="toc">
        <fo:block>
          <fo:basic-link internal-destination="{generate-id()}">
            <xsl:text>Act </xsl:text>
            <xsl:number/>
          </fo:basic-link>
          <fo:leader leader-length="5cm" leader-pattern="dots"
leader-alignment="reference-area"/>
          p. <fo:page-number-citation ref-id="{generate-id()}"/>
        </fo:block>
    <xsl:apply-templates mode="toc"/>
  </xsl:template>
  <xsl:template match="SCENE" mode="toc">
    <fo:block text-indent="2em">
    <fo:basic-link internal-destination="{generate-id()}">
      <xsl:text>Scene </xsl:text>
```

```
    <xsl:number/>
  </fo:basic-link>
  <fo:leader leader-length="5cm" leader-pattern="dots"/>
  p. <fo:page-number-citation ref-id="{generate-id()}"/>
</fo:block>
</xsl:template>
<xsl:template match="STAGEDIR">
  <fo:block text-align="center"
            font-size="10pt"
            font-style="italic"
            space-before=".5em">
    <xsl:apply-templates/>
  </fo:block>
</xsl:template>
<xsl:template match="SPEAKER">
  <fo:block text-align="center"
            font-size="10pt"
            space-before="1em"
            space-after=".5em">
    <xsl:apply-templates/>
  </fo:block>
</xsl:template>
<xsl:template match="LINE">
  <fo:block>
    <xsl:apply-templates/>
```

Example: Play to FO

```
    </fo:block>
  </xsl:template>
</xsl:stylesheet>
```

Top-level Template

Here we set the page format, running headers and footers, and columns

Page Format

```
<xsl:template match="/">
   <fo:root>
    <fo:layout-master-set>
      <fo:simple-page-master master-name="article-page"
        page-height="297mm" page-width="210mm"
        margin-top="20mm"  margin-bottom="10mm"
        margin-left="10mm" margin-right="10mm">
        <fo:region-body region-name="main" column-count="2"/>
        <fo:region-before region-name="header" extent="10pt"/>
        <fo:region-after region-name="header" extent="10pt"/>
      </fo:simple-page-master>
```

Page Sequence Master

```
 <fo:page-sequence-master master-name="article-sequence">
        <fo:single-page-master-reference master-name="article-page">
      </fo:page-sequence-master>
```

Page Sequence

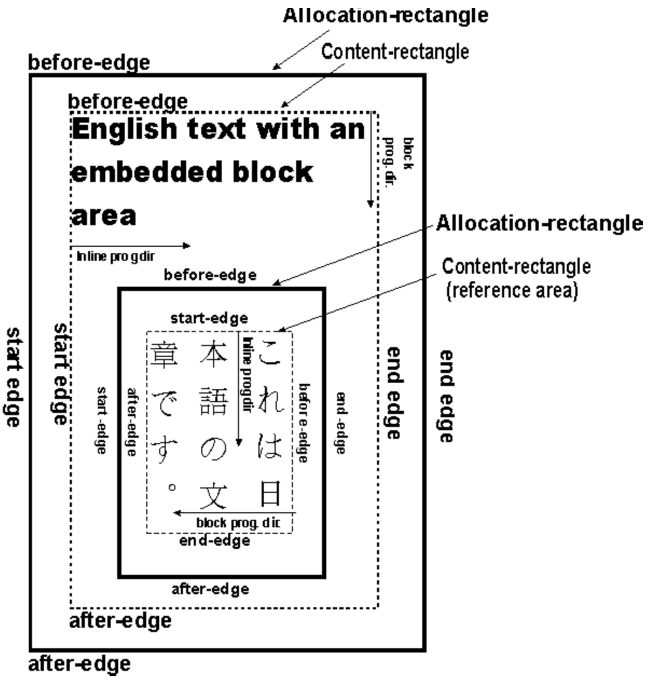
- Contains static-content (headers, footers, siders)
- And main text flow

Flow

Contains blocks, which contains text and inlines

I18N Formatting Objects and Properties

- fonts and Unicode character sets: [example1], [example2]
- writing-mode



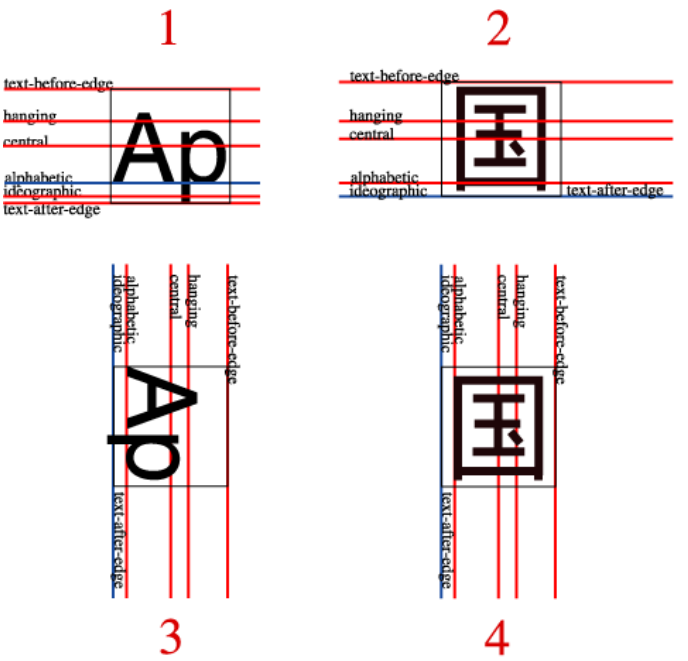
- baseline control

  `dominant-baseline`(of the principal font)

Modified font size. Baseline table is scaled and realigned on the dominant baseline:

```
<fo:inline>Apguru
   <fo:inline font-size=".75em" dominant-baseline="reset-size"
              alignment-baseline="hanging">Exji</fo:inline>
 </fo:inline>
```

I18N Formatting Objects and Properties


1
text-before-edge
hanging
central
alphabetic
ideographic
text-after-edge
Ap
2
text-before-edge
hanging
central
alphabetic
ideographic
text-after-edge
国
3
alphabetic
ideographic
central
hanging
text-before-edge
text-after-edge
Ap
4
alphabetic
ideographic
central
hanging
text-before-edge
text-after-edge
国

Other Formatting Objects

- fo:leader
- fo:external-graphic, fo:instream-foreign-object
- fo:footnote
- fo:page-number, fo:page-number-reference
- fo:list, fo:table, fo:float
- Dynamic properties (e.g. links)

And Properties

- Aural Properties (pitch, azimuth, etc.)
- Hyphenation control (country, hyphenation-character)
- Keeps and Breaks

Example: mixed writing modes

If you are still interested...

Status of the specifications

- XSLT 1.0 and XPath 1.0 are W3C recommendations
- Requirement documents for XSLT2.0 and XPath2.0 are available
- XPath2.0 is now being developed (with XML Query)
- XSL 1.0 (FO) is a Candidate Recommendation

Implementations

- Many implementations of XSLT1.0 exist: xt, Saxon, Oracle, Sun, Mozilla, (client side), MSXML (client side), Lotus, Unicorn, libxml, most of them free
- XSL 1.0: 7+ implementations: RenderX, Antenna House, FOP

(does SVG), PassiveTeX (does MathML), etc.

The Future

- XSL 1.0 will move to Recommendation
- Interoperability: include SVG and MathML in XSL.
- Applications: publishers will be able to put publications on the web as easily as printing them