XSL Formatting Objects Tutorial

Table of Contents

1. Introduction

This document gives a quick, learn-by-example introduction to XSL Formatting Objects. I don't discuss subtle details of implementation, but rather provide a series of examples of how to perform routine

tasks with XEP — an XSL formatter developed by RenderX, Inc. It is not a manual of XSL FO in general, and some examples given here may not work in other XSL FO formatters, or give different results.

This tutorial was conceived as a means to facilitate reading of *XSL 1.0 Recommendation of October 15, 2001*. The normative text is available from W3C site: http://www.w3.org/TR/2001/REC-xsl-20011015/. You should obtain a copy of XSL 1.0 Recommendation, and refer to it for a complete description of objects and properties mentioned here.

XEP 4.9 also implements several extensions to the XSL 1.0 Recommendation: they add support for useful functionality that cannot be expressed by standard XSL Formatting Objects. The last chapters of this document discuss available extensions in more detail. Needless to say, the extensions are proprietary and incompatible with other XSL formatters; please avoid their use whenever possible.

Readers who aren't XEP users can download an evaluation copy of XEP from http://shop.xattic.com, and run the proposed examples to see how the formatter works. This manual is also available in XML format with an associated XSL FO stylesheet; browsing its source code may also be interesting (and instructive).

XEP covers more of the spec than was needed for this manual. Please refer to XEP product documentation for a full list of supported elements and properties.

2. “Hello, World!”

Let us create the shortest XSL FO document.

```
<?xml version="1.0" encoding="iso-8859-1"?>❶

<fo:root xmlns:fo="http://www.w3.org/1999/XSL/Format">❷
  <fo:layout-master-set>❸
    <fo:simple-page-master master-name="my-page">
      <fo:region-body margin="1in"/>
    </fo:simple-page-master>
  </fo:layout-master-set>

  <fo:page-sequence master-reference="my-page">❹
    <fo:flow flow-name="xsl-region-body">❺
      <fo:block>Hello, world!</fo:block>❻
```

```
    </fo:flow>
  </fo:page-sequence>
</fo:root>
```

❶ This is an *XML declaration*. XSL FO belongs to XML family, so this is obligatory.

❷ *Root element*. The obligatory namespace attribute declares the XSL Formatting Objects namespace.

❸ *Layout master set*. This element contains one or more declarations of *page masters* and *page sequence masters* — elements that define layouts of single pages and page sequences. In the example, I have defined a rudimentary page master, with only one area in it. The area should have a 1 inch margin from all sides of the page.

❹ *Page sequence*. Pages in the document are grouped into sequences; each sequence starts from a new page. `Master-reference` attribute selects an appropriate layout scheme from masters listed inside `<fo:layout-master-set>`. Setting `master-reference` to a page master name means that all pages in this sequence will be formatted using this page master.

❺ *Flow*. This is the container object for all user text in the document. Everything contained in the flow will be formatted into regions on pages generated inside the page sequence. Flow name links the flow to a specific region on the page (defined in the page master); in our example, it is the *body region*.

❻ *Block*. This object roughly corresponds to `<DIV>` in HTML, and normally includes a paragraph of text. I need it here, because text cannot be placed directly into a flow.

Now we can save this document into a file and compile it using XEP 4.9. to produce a PDF file. Open it with Acrobat Reader, and enjoy :-).

3. Font and Text Attributes

Let us now enrich the text with character-level formatting. Several properties control font styles — family, size, color, weight, etc. Let's look at some examples:

```
<fo:block font-family="Times" font-size="14pt">
  Hello, world!
</fo:block>
```

Font family is Times, and font size is 14 points.

```
<fo:block font-family="Times" font-size="14pt" font-style="italic">
  <fo:inline color="red">H</fo:inline>ello,
  <fo:inline font-weight="bold">world!</fo:inline>
</fo:block>
```

Same as above, plus:

- the whole text is italicized (`font-style="italic"`);
- the first letter of the first word is written in red (`color="red"`);
- the second word is written in bold font (`font-weight="bold"`).

Note a new formatting object — `<fo:inline>`. It corresponds to `<SPAN>` in HTML, and ascribes formatting to chunks of text within a block.

Font properties are *inheritable*. It means that, once defined for a formatting object, they apply to all formatting objects inside it. That's why the first inline sequence affects only the color of the font, leaving its family, size, and slant unmodified. Inheritable properties can be put almost everywhere on the formatting objects tree; as a rule, you specify default font for a document by applying these properties to `<fo:flow>`, `<fo:page-sequence>` or even `<fo:root>`.

To reduce typing, you can use a shorthand notation for setting font attributes as a group. For example, the above example can be rewritten as follows:

```
<fo:block font="italic 14pt Times">
  <fo:inline color="red">H</fo:inline>ello,
  <fo:inline font-weight="bold">world!</fo:inline>
</fo:block>
```

The `font` property has the following syntax:

```
[<style, weight, and/or variant>] <size>[/<line height>] <family>
```

It sets all mentioned attributes to specified values, and resets all other font-related attributes to their default values, overriding inherited values. Be careful when using this feature: `font="14pt Times"` is not equivalent to a conjunction of `font-size="14pt"` & `font-family="Times"`!

Let's now build a full XSL FO example with font attributes introduced above, and some new ones:

```
<?xml version="1.0" encoding="iso-8859-1"?>

<fo:root xmlns:fo="http://www.w3.org/1999/XSL/Format">
  <fo:layout-master-set>
    <fo:simple-page-master master-name="my-page">
      <fo:region-body margin="1in"/>
    </fo:simple-page-master>
  </fo:layout-master-set>
  <fo:page-sequence master-reference="my-page">
     <fo:flow flow-name="xsl-region-body" font="12pt Times"❶>
       <fo:block font="italic 24pt Helvetica">
         <fo:inline color="red">F</fo:inline>ont
         <fo:inline color="red">A</fo:inline>ttributes
       </fo:block>
       <fo:block>❷
         The inherited font for this block is 12pt Times.
       </fo:block>
       <fo:block>
         Font attributes:
         <fo:inline color="red">colored</fo:inline>,
         <fo:inline font-weight="bold">bold</fo:inline>,
         <fo:inline font-style="italic">italic</fo:inline>,
         <fo:inline font-size="75%">small</fo:inline>,
         <fo:inline font-size="133%">large</fo:inline>.
       </fo:block>
      <fo:block>
         Text attributes:❸
         <fo:inline text-decoration="underline">underlined</fo:inline>,
         <fo:inline letter-spacing="3pt"> expanded </fo:inline>,
         <fo:inline word-spacing="6pt">
           text with extra spacing between words
         </fo:inline>,
         <fo:inline text-transform="uppercase">all capitals</fo:inline>,
         <fo:inline text-transform="capitalize">capitalized</fo:inline>,
```

```
          text with <fo:inline baseline-shift="sub"
          font-size="smaller">subscripts</fo:inline>
          and <fo:inline baseline-shift="super"
          font-size="smaller">superscripts</fo:inline>.
        </fo:block>
      </fo:flow>
    </fo:page-sequence>
</fo:root>
```

❶ A common font for the whole flow is specified.

❷ This block inherits font attributes from the flow.

❸ In this block, I introduce several other text-level properties:

text decoration

: underline/overline/strikethrough;

letter* and *word spacing

: a positive value expands text, a negative value condenses it;

text transformations

: upper/lower case, capitalize;

shifted text

: subscripts and superscripts.

4. Blocks

4.1. Text Alignment, Line Height

Let's consider this piece:

```
<fo:block line-height="1.5" text-align="justify">
  This is an example of double-justified text.
  The space between lines is 1.5 of the nominal font height.
</fo:block>
```

The `line-height` property specifies the line height: it can be expressed as a length, as a numeric value, or as a percent. Numbers and percents are interpreted as multiples to the nominal font height.

`Text-align` property defines the alignment of the text within the block. XSL FO uses a specific co-ordinate system for referring to block edges: instead of ‘left’ and ‘right’, side alignment is expressed in terms of *inline progression direction*. For Western scripts, glyphs on the line are placed from left to right; therefore, left-aligned text will have `text-align="start"`, and right-aligned text will have `text-align="end"`. Two other values are `"center"` and `"justify"` (same as in CSS). You can also use other CSS values for this property — `"left"` is a synonym for `start`, and `"right"` is a synonym for `end`.

Let's look into a more complicated example:

```
<fo:block text-align="justify" text-indent="1in"
          text-align-last="end" last-line-end-indent="1in">
  This is an example of double-justified text with an indented first line.
  The last line of the text is aligned to the right, and indented
  by 1 inch from the right.
</fo:block>
```

This fragment should be formatted as follows:

- text is double justified (`text-align`);
- the first line is indented by 1 inch from the left (`text-indent`);
- the last line is aligned to the right (`text-align-last`);
- the last line is indented by 1 inch from the right (`last-line-end-indent`).

By specifying a negative value for `text-indent` / `last-line-end-indent`, it is possible to create outdents. To make the text stay within the limits, two more properties are used that control indentation of the text as a whole:

```
<fo:block text-align="start" text-indent="-1in"
          text-align-last="end" last-line-end-indent="-1in"
          start-indent="1in" end-indent="1in">
  This is an example of left-aligned text with an outdented first line.
```

```
  The last line of the text is aligned to the right, and outdented
  by 1 inch from the right.
</fo:block>
```

The names of properties `start-indent` and `end-indent` follow the same logic as the values for `text-indent`: `start-indent` controls white space added at the beginning of every line, and `end-indent` adds a margin at the end of every line.

To complete this chapter, let's introduce attributes to position blocks vertically with respect to each other: `space-before` and `space-after`. Their names also derive from the writing-mode approach: '`before`' means "before the first line", and '`after`' implies "after the last line".

Spaces are more complicated than indents: they aren't specified as a single value, but rather as a vector of several components — minimum, optimum, and maximum value for the space. Components can be assigned separately: an attribute name will consist of a property name followed by a component name, separated by a dot. You can also assign all numeric components the same value by using the attribute name without a component qualifier; this is what you normally do in most cases.

An important property of spaces is that *they aren't additive:* if there are several space specifiers between two blocks (e.g. `space-after` on a preceding block and `space-before` on a following block), a single space value is chosen so as to satisfy all applicable constraints. Apparently, spaces merge and don't sum up. (The real rules for space resolution are more complicate; please refer to the XSL FO specs).

The fragment below illustrates the use of spaces:

```
<fo:flow flow-name="xsl-region-body" font="14pt Times">
  <fo:block font-size="24pt"
            text-align="center"
            space-before="30pt"
            space-before.conditionality="retain"❶
            space-after="12pt">The Jabberwocky</fo:block>
  <fo:block font-style="italic"
            text-align="end"
            space-before="12pt"
            space-after="9pt">Lewis Carroll</fo:block>
  <fo:block start-indent="1.5in" space-after="9pt">
```

```
    <fo:block>&#8217;❷Twas brillig, and the slithy toves</fo:block>
    <fo:block>Did gyre and gimble in the wabe:</fo:block>
    <fo:block>All mimsy were the borogoves,</fo:block>
    <fo:block>And the mome raths outgrabe.</fo:block>
  </fo:block>
  <fo:block start-indent="1.5in" space-after="9pt">
    <fo:block>&#8220;❸Beware the Jabberwock, my son!</fo:block>
    <fo:block>The jaws that bite, the claws that catch!</fo:block>
    <fo:block>Beware the Jubjub bird, and shun</fo:block>
    <fo:block>The frumious Bandersnatch!&#8221;❹</fo:block>
  </fo:block>
</fo:flow>
```

❶ By default, `space-before` at the top of the page and `space-after` at the bottom of the page are suppressed. To force them, specify `.conditionality="retain"`.

❷ `&#8217;` is a UCS code for right single quote. XEP addresses all characters by their Unicode values.

❸❹ `&#8220;` and `&#8221;` are UCS codes for left and right double quotes.

4.2. Borders, Padding, and Background

Blocks may have *borders* from either side. Sides can be addressed either in an absolute orientation scheme (`left`, `right`, `top`, and `bottom`), or in a writing-mode relative scheme (resp. `start`, `end`, `before`, and `after`).

Every border has the following properties, that may get the following values in XSL FO:

`color`

one of 16 predefined HTML system colors, or an RGB value;

`style`

`solid`, *`dashed`*, *`dotted`*, *`double`*, *`inset`*, *`outset`*, *`groove`*, *`ridge`*, or *`none`*;

`width`

`thin`, *`medium`*, *`thick`*, or an explicit width specification.

You can specify each property for each border separately by writing several attributes of the form `border-{side}-{property}`:

```
<fo:block border-top-color="black"
          border-top-style="solid"
          border-top-width="thick"
          text-align="center">
  Thick black border at the top
</fo:block>
```

You can also specify properties for all the four sides as a whole, using a shorthand notation `border-{property}`:

```
<fo:block border-color="gray"
          border-style="groove"
          border-width="medium"
          text-align="center">
  Medium gray groove around the whole block
</fo:block>
```

You can also group properties that refer to one side into a single shorthand attribute `border-{side}`. However, only absolutely oriented side names (`top`, `bottom`, `left`, and `right`) are permitted in this position. An example:

```
<fo:block text-align="center"
          border-top="dashed 1pt #C00000"❶
          border-bottom="1pt dashed #C00000">
  1pt dashed red border at the top and bottom
</fo:block>
```

❶ Elements inside the attribute can be specified in any order, separated by spaces.

Finally, a single `border` attribute can accumulate properties that are ascribed to all the four sides:

```
<fo:block border="thin silver ridge"
          text-align="center">
```

```
  Thin silver ridge around the whole block
</fo:block>
```

When printed, a block may be split by a page break or a column break. What happens to the borders adjacent to the breakline? For some block types, you may want to box every part of the original block separately, i.e. draw a border line where the break occurs; this is the default behaviour in XSL FO. For some other blocks, you may prefer to “keep the box open” suppressing borders at column/page breaks. This behavior is controlled by a special component of the border's width — `border-{side}-width.conditionality`:

```
<fo:block border="thin blue groove"
          border-before-width.conditionality="discard"
          border-after-width.conditionality="discard">
  If this block happens to be split by a page break,
  no line will be drawn on either side of the break.
</fo:block>
```

Only writing-mode oriented sides (`before`, `after`, `start`, and `end`) are permitted in the conditional border expressions.

Once you have set a border around an object, you normally want to specify a *padding* between the text and the border. This is done by `padding-{side}` attributes:

```
<fo:block border="thin solid navy"
          text-align="center"
          padding-before="18pt"
          padding-bottom="18pt">
  <fo:block border="thin solid maroon">
    The outer block has a 18 pt padding from top and bottom
  </fo:block>
</fo:block>
```

There also exists a shorthand `padding` attribute:

```
<fo:block border="thin solid navy"
          text-align="center"
          padding="2cm">
  <fo:block border="thin solid maroon">
    The outer block has a 2 cm padding from all sides
  </fo:block>
</fo:block>
```

You can also specify several numbers as a value for `padding` attributes:

- if there are two values, the top and bottom paddings are set to the first value and the right and left paddings are set to the second;
- if there are three values, the top is set to the first value, the left and right are set to the second, and the bottom one is set to the third;
- if there are four values, they apply to the top, right, bottom, and left, respectively.

Example:

```
<fo:block border="thin solid navy"
          text-align="center"
          padding="1cm 3cm">
  <fo:block border="thin solid maroon">
    The outer block has 1 cm padding from top and bottom,
    and 3 cm padding from right and left.
  </fo:block>
</fo:block>
```

Like borders, padding may be conditional at page breaks: specifying `padding-`*`{side}`*`.conditionality="discard"` suppresses padding before or after a page break. Like for borders, only writing-mode oriented sides (`before`, `after`, `start`, and `end`) are permitted in this position.

To complete the picture, let's learn how to set *backgrounds* for blocks.

Blocks may have different backgrounds — colored or even decorated with a graphic. To specify a color for the background, use `background-color` property:

```
<fo:block color="yellow"
          background-color="red"
          padding="12pt"
          text-align="center">
  Yellow on red
</fo:block>
```

To add a background image to the block, you should

- specify image source — `background-image`;
- specify image position — `background-position`;
- specify whether the image should be repeated along any of the axes — `background-repeat`.

The `background-image` attribute specifies an URI of the bitmap image file. XEP handles HTTP, FTP, and file system resource locators in URIs. An unqualified URI is treated as a path to a file in the local file system; if the path is relative, it is calculated from the location of the source XSL FO document (rather than from the current directory where XEP is run). Note also the `url('…')` function-like wrapper around the file name: this is required by the XSL 1.0 Recommendation. (XEP recognizes unwrapped URLs, too).

☞ Under Win32, absolute pathnames may contain a colon after the disk letter. These colons confuse Java URI resolver: the disk letter is treated as protocol name. Use an unabridged file URI:

```
file:/C:/XEP/myimage.jpg
```

Example:

```
<fo:block border="0.5pt solid silver"
          background-image="url('spots.jpg')"
          padding="18pt"
          text-align="center">
  The whole background tiled with colored spots
```

```
</fo:block>
```

By default, an image specified in `background-image` tiles the whole block. To insert only a single instance, disable tiling by setting `background-repeat="no-repeat"`:

```
<fo:block border="0.5pt solid silver"
          background-image="url('spots.jpg')"
          background-repeat="no-repeat"
          background-position-horizontal="center"
          background-position-vertical="center"
          padding="18pt"
          text-align="center">
  A single image placed 18pt to the right
  and 6pt below the upper left corner.
</fo:block>
```

The position of the image within the block is controlled by two properties, `background-position-horizontal` and `background-position-vertical`. The `background-position` shorthand property from CSS2 can also be used:

```
<fo:block border="0.5pt solid silver"
          background-image="url('spots.jpg')"
          background-repeat="repeat-y"
          background-position="left center"
          background-color="silver"
          padding="18pt"
          text-align="center">
  A stripe made of coloured spots along the left edge of the block;
  the rest of the block has a silver background.
</fo:block>
```

XSL Recommendation defines no method to scale the background image. To do it, you have to recur to RenderX add-ons; see description of `rx:background-content-width`, `rx:background-content-height`, and `rx:background-scaling` extension properties in the documentation for XEP 4.9.

5. Page Layout

5.1. Page Sequence Masters

So far, I have used only single page masters in examples. In this section, more complex cases will be analyzed. To start, let's design a page sequence with two page masters: one for the first page, the other one for the rest of the document.

```
<?xml version="1.0" encoding="iso-8859-1"?>

<fo:root xmlns:fo="http://www.w3.org/1999/XSL/Format">
  <fo:layout-master-set>
    <fo:simple-page-master master-name="first-page">
      <fo:region-body margin="1in" border="thin silver ridge"❶
                                      padding="6pt"/>
    </fo:simple-page-master>
    <fo:simple-page-master master-name="all-pages">
      <fo:region-body margin="1in"/>
    </fo:simple-page-master>
    <fo:page-sequence-master master-name="my-sequence">❷
      <fo:single-page-master-reference master-reference="first-page"/>❸
      <fo:repeatable-page-master-reference master-reference="all-pages"/>❹
    </fo:page-sequence-master>
  </fo:layout-master-set>
  <fo:page-sequence master-reference="my-sequence"❺>
    <fo:flow flow-name="xsl-region-body" font="72pt Times">
       <fo:block space-before="2in" space-after="2in">❻
         First block
       </fo:block>
       <fo:block space-before="2in" space-after="2in">❼
         Second block
       </fo:block>
       <fo:block space-before="2in" space-after="2in">
         Third block
       </fo:block>
    </fo:flow>
```

```
  </fo:page-sequence>
</fo:root>
```

❶ In XSL FO, you can specify borders, padding, and background on regions in exactly the same way as you do it on blocks. The first page in this example will have a border around it, while others will remain borderless.

❷ The *page sequence master* defines the chain of page masters to use for a page sequence.

❸ `<fo:single-page-master-reference>` inserts a single page master in the chain.

❹ `<fo:repeatable-page-master-reference>` makes the specified page masters repeat up to the end of the chain.

❺ Note that `master-reference` attribute of a `<fo:page-sequence>` can refer to either a `<fo:page-sequence-master>` or a `<fo:simple-page-master>`. In the latter, all pages generated by this `<fo:page-sequence>` will use the same page master.

❻❼ Spaces *are not inheritable*: you cannot specify them on a surrounding block. There's no alternative to specifying them explicitly on every block involved.

You can also specify different page masters for odd and even pages, blank pages, first/last pages, etc. This is achieved by using a more complex sequence specifier — `<fo:repeatable-page-master-alternatives>`. It contains one or more `<fo:conditional-page-master-reference>` elements; each of these elements specifies a name of a page master and a set of conditions that should be satisfied for this page master to apply. When generating a page chain, the alternatives inside `<fo:conditional-page-master-reference>` are looked from left to right, and the first one for which all the conditions hold will be chosen.

In the example below, the first page will have a thin silver border around it, and all other pages will have a border along the inside edge of the body area (left for even pages, right for odd ones).

```
<fo:layout-master-set>
  <fo:simple-page-master master-name="first-page">
    <fo:region-body margin="1in" border="thin solid silver"
                                 padding="6pt"/>
  </fo:simple-page-master>
  <fo:simple-page-master master-name="odd-page">
    <fo:region-body margin="1in" border-right="medium gray ridge"
                                 padding-right="6pt"/>
  </fo:simple-page-master>
```

```
    <fo:simple-page-master master-name="even-page">
      <fo:region-body margin="1in" border-left="medium gray ridge"
                                   padding-left="6pt"/>
    </fo:simple-page-master>
    <fo:page-sequence-master master-name="my-sequence">
      <fo:repeatable-page-master-alternatives>
        <fo:conditional-page-master-reference page-position="first"
                                              master-reference="first-page"/>
        <fo:conditional-page-master-reference odd-or-even="odd"
                                              master-reference="odd-page"/>
        <fo:conditional-page-master-reference odd-or-even="even"
                                              master-reference="even-page"/>
      </fo:repeatable-page-master-alternatives>
    </fo:page-sequence-master>
  </fo:layout-master-set>
  <fo:page-sequence master-reference="my-sequence">
    …
    …
    …
```

5.2. Areas, Flows, Static Contents

So far, I only placed the contents into the body region of the page master. There are other regions on the page, used to display static elements of page layout — headers, footers, and sidebars. Inside page master, there may be up to five region specifications:

`<fo:region-body>`

: the central part of the page; required

`<fo:region-before>`

: header region

`<fo:region-after>`

: footer region

`<fo:region-start>`

: left sidebar region

<fo:region-end>

right sidebar region

The dimensions of the regions are calculated by the following algorithm:

1. The page size is controlled by `page-height` and `page-width` properties of the `<fo:simple-page-master>`. There also exists a `size` shorthand property to set the page size. Worth noting are *"portrait"* and *"landscape"* values to set the default page size with different orientation.

2. The size of region-body is determined by margins around it (calculated from page bounding rectangle).

3. All other regions stick to the correspondent edge on the page. Their width/height is given by the `extent` property. Note that side regions does not influence the position of the region-body; so, the correspondent margin of the region-body must be wide enough for the side regions to fit — not less than the `extent` of the correspondent side region.

4. To control the allocation of page corners to either sidebars or headers/footers, a special `precedence` attribute is used on side regions: if set to *"true"*, its bearer captures the corner. In case of equal precedences, headers/footers win.

All regions may have borders, padding and background. However, the XSL 1.0 Recommendation is contradictory with respect to borders and padding on all region areas but `<fo:region-body>`: the respective properties are listed, but there is a notice in the text that requires borders and padding to be 0. XEP implements borders and padding on all regions, but you are warned.

Regions are named using a special `region-name` property. This property has a different default value for each of the regions:

- *"xsl-region-body"* for `<fo:region-body>`,
- *"xsl-region-before"* for `<fo:region-before>`,
- *"xsl-region-after"* for `<fo:region-after>`,
- *"xsl-region-start"* for `<fo:region-start>`,
- *"xsl-region-end"* for `<fo:region-end>`.

To put contents into side regions, a special formatting object — `<fo:static-content>` — is placed inside `<fo:page-sequence>`. It is bound to a specific region by the `flow-name` property that should

match the `region-name` of a region in a page-master. The contents of a `<fo:static-content>` is formatted in the respective region on every page produced by this page-master.

Let us now look at the examples. The simplest example just adds a header to “Hello, world!”:

```
<fo:layout-master-set>
  <fo:simple-page-master master-name="my-page">
    <fo:region-body margin="1.5in 1in"❶/>
    <fo:region-before extent="1.5in"
                      padding="6pt 1in"
                      border-bottom="0.5pt silver solid"
                      display-align="after"❷/>
  </fo:simple-page-master>
</fo:layout-master-set>
<fo:page-sequence master-reference="my-page">
  <fo:static-content flow-name="xsl-region-before"❸
                     font="italic 10pt Times">
     <fo:block>"Hello, world!" Example</fo:block>
  </fo:static-content>
  <fo:flow flow-name="xsl-region-body">
     <fo:block>Hello, world!</fo:block>
  </fo:flow>
</fo:page-sequence>
```

❶ Margins can be set by multiple values inside the `margin` shorthand: assignment of values to sides is the same as for `padding`.

❷ `display-align` specifies the alignment of the contents inside the area. It is a common technique to set it to *`"before"`* for headers.

❸ The default name for the header region is used.

Another example uses sidebars. It draws a right sidebar on odd pages, and a left sidebar on even pages:

```
<fo:layout-master-set>
  <fo:simple-page-master master-name="even-page">
    <fo:region-body margin="1in 1.5in"
```

```
                    column-count="2"❶ column-gap="0.5in"/>
    <fo:region-start extent="1.5in"
                     region-name="my-left-sidebar"
                     reference-orientation="90"❷
                     padding="6pt 1in"
                     border-right="0.5pt silver solid"❸
                     display-align="after"/>
  </fo:simple-page-master>
  <fo:simple-page-master master-name="odd-page">
    <fo:region-body margin="1in 1.5in"
                    column-count="3"❹ column-gap="0.5in"/>
    <fo:region-end   extent="1.5in"
                     region-name="my-right-sidebar"
                     reference-orientation="-90"
                     padding="6pt 1in"
                     border-left="0.5pt silver solid"
                     display-align="after"/>
  </fo:simple-page-master>
  <fo:page-sequence-master master-name="my-sequence">
    <fo:repeatable-page-master-alternatives>
      <fo:conditional-page-master-reference odd-or-even="odd"
                                            master-reference="odd-page"/>
      <fo:conditional-page-master-reference odd-or-even="even"
                                            master-reference="even-page"/>
    </fo:repeatable-page-master-alternatives>
  </fo:page-sequence-master>
</fo:layout-master-set>
<fo:page-sequence master-reference="my-sequence">
  <fo:static-content flow-name="my-left-sidebar"
                        font="italic 10pt Times">
     <fo:block text-align="end">
       Left sidebar on an even page
     </fo:block>
  </fo:static-content>
  <fo:static-content flow-name="my-right-sidebar"
                        font="italic 10pt Times">
     <fo:block text-align="start">
```

```
          Right sidebar on an odd page
        </fo:block>
    </fo:static-content>
    …
    …
    …
```

❶❹ Body region may have *multiple columns*; other regions may not. The last attribute specifies the gap between columns.

> ☞ Number of columns may differ across pages within a single flow. In this example, all odd pages will have three columns while all even pages will have two.

❷ `reference-orientation` attribute specifies rotation of coordinate axes for the region; its value is an angle of rotation counterclockwise (in degrees; must be multiple of 90; negative values rotate clockwise).

❸ Note that sides of the region are referenced with respect to the original (not rotated) orientation!

6. More Complex Structures

6.1. Lists

Lists in XSL FO are much more than just a bulleted sequence of paragraphs: it is a general-purpose mechanism to align two blocks adjacent to each other. It may be used to format ordinary lists, footnotes, image lists, and even to produce some table-like layout patterns.

A list is created by a `<fo:list-block>` object. Inside it, there are one or more `<fo:list-item>` elements. Each list item contains one `<fo:list-item-label>` followed by one `<fo:list-item-body>`. These two elements contain blocks that are aligned vertically and placed side-by-side.

Let's start with an ordinary bulleted list:

```
<fo:list-block provisional-distance-between-starts="18pt"❶
               provisional-label-separation="3pt"❷>
  <fo:list-item>
    <fo:list-item-label end-indent="label-end()"❸>
      <fo:block>&#x2022;❹</fo:block>
```

```
    </fo:list-item-label>
    <fo:list-item-body start-indent="body-start()"❺>
       <fo:block>First item</fo:block>
    </fo:list-item-body>
  </fo:list-item>
  <fo:list-item>
    <fo:list-item-label end-indent="label-end()">
      <fo:block>&#x2022;</fo:block>
    </fo:list-item-label>
    <fo:list-item-body start-indent="body-start()">
       <fo:block>Second item</fo:block>
    </fo:list-item-body>
  </fo:list-item>
</fo:list-block>
```

❶ This property specifies how far the left side of the label is distant from the left side of the body.

❷ This property specifies the separation between the right side of the label and the left edge of the body.

❸ `end-indent` attribute specifies the offset of the right edge of `<fo:list-item-label>` from the right edge of the reference area (i.e. page). A special `label-end()` function sets it to the value calculated from `provisional-distance-between-starts` and `provisional-label-separation` values. However, this is not a default value: you have to specify `end-indent="label-end()"` on each `<fo:list-item-label>` in the list. Alternatively, you can use an explicit value of `end-indent`.

❹ This is a Unicode for a round bullet.

❺ `start-indent` attribute specifies the left offset of the `<fo:list-item-body>` from the left. A special `body-start()` function sets it to the value calculated from `provisional-distance-between-starts`. Like for the `<fo:list-item-label>`, this is not a default value; don't forget to specify it on each `<fo:list-item-body>`.

Another example — using `<fo:list-block>` to align text in a header:

```
<fo:static-content flow-name="xsl-region-before">
  <fo:list-block border-bottom="1pt gray ridge"❶ padding-after="6pt">
    <fo:list-item>
```

```
      <fo:list-item-label end-indent="3in">❷
        <fo:block text-align="start">RenderX XSL FO Manual</fo:block>
      </fo:list-item-label>
      <fo:list-item-body start-indent="3in">❸
        <fo:block text-align="end">List Example</fo:block>
      </fo:list-item-body>
    </fo:list-item>
  </fo:list-block>
</fo:static-content>
```

❶ `<fo:list-block>` and `<fo:list-item>` are block-level elements, and can get margins (including `space-before`/`space-after` and indents), keep & break constraints, borders, padding, and background. Please note that `<fo:list-item-body>` and `<fo:list-item-label>` aren't considered blocks in the spec, and cannot have borders or background.

❷❸ Instead of using `label-end()`/`body-start()` technique, you may also specify indents explicitly like in this example. To use this method, you should know the width of the parent reference area to avoid overlapping of item labels to item bodies.

6.2. Tables

Tables in XSL FO resemble HTML ones: they are made of cells grouped into rows; rows are further grouped into row groups — table header, table footer, and table bodies (one or more). There are also column descriptors. Table model is rather evolved: please refer to the spec for further details. Here, I limit myself to a couple of core examples.

A basic 2×2 table:

```
<fo:table border="0.5pt solid black" text-align="center">
  <fo:table-body>
    <fo:table-row>
      <fo:table-cell padding="6pt" border="0.5pt solid black">❶
        <fo:block> upper left </fo:block>
      </fo:table-cell>
      <fo:table-cell padding="6pt" border="0.5pt solid black">
        <fo:block> upper right </fo:block>
      </fo:table-cell>
```

```
    </fo:table-row>
    <fo:table-row>
      <fo:table-cell padding="6pt" border="0.5pt solid black">
        <fo:block> lower left </fo:block>
      </fo:table-cell>
      <fo:table-cell padding="6pt" border="0.5pt solid black">
        <fo:block> lower right </fo:block>
      </fo:table-cell>
    </fo:table-row>
  </fo:table-body>
</fo:table>
```

❶ Table cells can have borders, padding, and background; they cannot have margins. The table itself may have all attributes a normal block may have, except for padding.

A more complicated example includes explicit column width assignment and cells spanning multiple grid units:

```
<fo:table border="0.5pt solid black"
          text-align="center"
          border-spacing="3pt"❶>
  <fo:table-column column-width="1in"/>
  <fo:table-column column-width="0.5in" number-columns-repeated="2"❷/>
  <fo:table-header>❸
    <fo:table-row>
      <fo:table-cell padding="6pt"
                     border="1pt solid blue"
                     background-color="silver"
                     number-columns-spanned="3"❹>
        <fo:block text-align="center" font-weight="bold">
          Header
        </fo:block>
      </fo:table-cell>
    </fo:table-row>
  </fo:table-header>
  <fo:table-body>
```

```
    <fo:table-row>
      <fo:table-cell padding="6pt"
                     border="1pt solid blue"
                     background-color="silver"
                     number-rows-spanned="2"❺>
        <fo:block text-align="end" font-weight="bold">
          Items:
        </fo:block>
      </fo:table-cell>
      <fo:table-cell padding="6pt" border="0.5pt solid black">
        <fo:block> 1 : 1 </fo:block>
      </fo:table-cell>
      <fo:table-cell padding="6pt" border="0.5pt solid black">
        <fo:block> 1 : 2 </fo:block>
      </fo:table-cell>
    </fo:table-row>
    <fo:table-row>❻
      <fo:table-cell padding="6pt" border="0.5pt solid black">
        <fo:block> 2 : 1 </fo:block>
      </fo:table-cell>
      <fo:table-cell padding="6pt" border="0.5pt solid black">
        <fo:block> 2 : 2 </fo:block>
      </fo:table-cell>
    </fo:table-row>
  </fo:table-body>
</fo:table>
```

❶ `border-spacing` corresponds to `CELLSPACING` attribute of the HTML table model.

❷ This is a column specifier that sets the width for the second and the third column. Instead of repeating the same specifier for two consecutive columns, I have used an `number-columns-repeated` attribute: it just clones the description the specified number of times.

❸ *Table header* is a row group like `<fo:table-body>`. It is repeated at the top of page after page breaks unless you specify `table-omit-header-at-break="true"` on the embracing `<fo:table>`.

❹ This cell will span three columns horizontally.

❺ This cell will span two rows vertically.

❻ This row contains only two cells; the first grid unit is already occupied by the cell spanning from the previous row.

7. Graphics

There is a special inline element for including graphics into XSL FO — `<fo:external-graphic>`. The source image is specified by the `src` attribute whose value is a URI. XEP handles HTTP, FTP, `data` and filesystem resource locators in URIs. An unqualified URI is treated as a path to a file in the local file system; if the path is relative, it is calculated from the location of the source XSL FO document.

Here's an example:

```
<fo:block>
  This text includes a picture:
  <fo:external-graphic src="url('smile.gif')"❶
                       content-height="1em"❷ content-width="1em"❸/>
</fo:block>
```

❶ Note the `url('…')` function-like wrapper around the file name: this is required by the XSL 1.0 Recommendation. (XEP recognizes unwrapped URLs, too).

❷❸ In this example, the height and the width of the image are expressed in units relative to the nominal font size. This is a convenient technique to scale small inlined images proportionally to the text height.

If you need a block-level graphic, you should wrap the element in a `<fo:block>`, like in the example below:

```
<fo:block font-weight="bold"
          keep-with-next.within-column="always">❶
  Figure 1: A Smiling Face
</fo:block>
<fo:block>
    <fo:external-graphic src="url('smile.gif')"
                         content-height="200%" content-width="200%"/>❷
```

```
</fo:block>
```

❶ This property “glues” the figure caption to the figure itself.

❷ `content-width` and `content-height` expressed as percents denote scaling the image from its original (intrinsic) size.

It is also possible to create an image directly in the XSL-FO, using the embedded SVG feature:

```
<fo:block>
  Here is the image of a typical roadsign:
  <fo:instream-foreign-object content-height="1em">❶
    <svg:svg xmlns:svg="http://www.w3.org/2000/svg"❷
             height="100" width="100" viewBox="-50 -50 100 100">
      <svg:circle r="50" style="fill:red; stroke:none"/>
      <svg:rect x="-40" y="-10" width="80" height="20"
                style="fill:white; stroke:none"/>
    </svg:svg>
  </fo:instream-foreign-object>
</fo:block>
```

❶ Embedded images can be scaled with the same attributes as external ones.

❷ SVG elements reside in a separate namespace that has to be declared; otherwise, SVG images will not be recognized.

8. Advanced Features

8.1. Containers and Reference Orientation

`<fo:block-container>` objects group blocks in a separate area that can have a different orientation of coordinate axes or writing direction. A special `reference-orientation` property sets the orientation of the "top" direction for the area. Its value is the rotation angle, measured in degrees: *`0`*, *`90`*, *`180`*, *`270`*, *`-90`*, *`-180`*, *`-270`*. Positive values rotate the coordinate system counterclockwise, and negative ones turn it clockwise.

Let us consider a simple example:

```
<fo:block-container width="250pt" height="20pt"
                        border="1pt solid black"
                        reference-orientation="0">❶
  <fo:block text-align="left">❷
 Regular text.
  </fo:block>
</fo:block-container>
<fo:block-container width="250pt" height="20pt"
                        border="1pt solid black"
                        reference-orientation="180">❸
  <fo:block text-align="left">❹
 Text shown upside down.
  </fo:block>
</fo:block-container>
```

❶ This container has a default `reference-orientation` of 0. Note that the size of the rectangular area occupied by the container is explicitly specified.

❸ The text in this block should be shown upside down.

❷❹ Blocks in both containers have `text-align="left"`. Since "left" is determined with respect to the "top", the text in this block should be aligned to the opposite side than in the previous one.

Besides containers, `reference-orientation` property can apply to `<fo:simple-page-master>` and `<fo:region-*>` formatting objects. In the following example, I create a page master with a single `<fo:region-body>` whose contents is rotated 90° clockwise:

```
<?xml version="1.0" encoding="iso-8859-1"?>
<fo:root xmlns:fo="http://www.w3.org/1999/XSL/Format">
  <fo:layout-master-set>
    <fo:simple-page-master master-name="my-page">
      <fo:region-body border="0.25pt solid silver"
                        margin="0.5in"
                        padding="12pt"
                        reference-orientation="-90"/>
```

```
    </fo:simple-page-master>
  </fo:layout-master-set>
  <fo:page-sequence master-reference="my-page">
    <fo:flow flow-name="xsl-region-body">
      <fo:block>
        Text and images on the page
       <fo:external-graphic src="url('smile.gif')"
                            content-height="100%"
                            content-width="100%"/>
      </fo:block>
      <fo:block-container width="250pt" height="20pt"
                          border="1pt solid black"
                          reference-orientation="180">❶
        <fo:block>
   Text flips upside down.
        </fo:block>
      </fo:block-container>
    </fo:flow>
  </fo:page-sequence>
</fo:root>
```

❶ The rotation angle specified by `reference-orientation` is measured from the orientation of the parent area. In this case, 180° specified on the `<fo:block-container>` will be added to -90° already on the `<fo:region-body>`; the resulting text rotation will be 90° counter-clockwise.

8.2. Writing Mode and Bidirectionality

XSL FO has powerful means to deal with non-Western writing systems. Lines can be made horizontal or vertical; character can be ordered from any side; line stacking order on the page can also be varied. To define ordering of characters within lines and stacking direction of lines into paragraphs, we use `writing-mode` property. It can be specified on `<fo:simple-page-master>`, `<fo:region-*>`, `<fo:table>`, `<fo:block-container>`, and `<fo:inline-container>` elements. Its primary values are:

- `"lr-tb"`: left-to-right, top-to-bottom. This is the default writing mode in XSL-FO; it is used by the majority of world languages, including English;

- `"rl-tb"`: right-to-left, top-to-bottom. This mode is used in Arabic writing system (adopted by many languages of the Middle East), Hebrew, and Syriac alphabets.
- `"tb-rl"`: top-to-bottom, right-to-left. This way of writing is widely used for Japanese, but also for Chinese and other languages of the East Asia.

☞ As of version 4.9, XEP supports only horizontal writing modes: `"lr-tb"` and `"rl-tb"`.

`writing-mode` property defines every aspect of the document organization: binding edge, column ordering in tables, text alignment in blocks, etc. It also sets the correspondence between relative directions (`before` – `after` – `start` – `end`)and absolutely oriented ones (`top` – `bottom` – `left` – `right`).

However, internationalization issues are too complex to be described by a single property. In right-to-left writing systems (Arabic and Hebrew), it is not uncommon to include fragments of text written in Latin alphabet; such text progresses left-to-right. The mechanism that permits mixing writing directions within the same fragment of text is called *bidirectionality*, or bidi for short. The Unicode standard defines rules to calculate ordering of characters in bidirectional text; the respective part of the Unicode, Annex #9: *The Bidirectional Algorithm* [http://www.unicode.org/reports/tr9/], is adopted by XSL FO.

However, in certain cases the Unicode bidi algorithm is not enough to determine character ordering. For this purpose, XSL defines a special element `<fo:bidi-override>` that permit to alter the bidirectional behaviour of the whole text or its parts. It has two properties:

`direction`

Sets the dominant direction for a span of text. Possible values are:

- `"ltr"` — from left to right;
- `rtl` from right to left.

`unicode-bidi`

Specifies behaviour of a text span with respect to the Unicode bidi algorithm. Possible values are the following:

- `"normal"` — order characters by Unicode bidi;
- `"embed"` — open a new level of embedding;

- `"bidi-override"` — ignore directionality of the text and arrange characters in the order specified by the `direction` property.

The following example shows two block containers with different writing modes. Both contain a mixture of English (left-to-right) and Hebrew (right-to-left) text. The first container has `writing-mode="lr-tb"` (default): its content will be treated as an English phrase with inclusions of Hebrew words. The second container has `writing-mode="rl-tb"`, so its content will be considered a Hebrew phrase with some English words in it.

```
<fo:block-container writing-mode="lr-tb">❶
   <fo:block> The words of Queen Esther,
      &#x5D5;&#x5DB;&#x5D0;&#xFB2A;&#x5E8;
      &#x5D0;&#x5D1;&#x5D3;&#xFB4A;&#x5D9;
      &#x5D0;&#x5D1;&#x5D3;&#xFB4A;&#x5D9;
      "If I perish [in trying to save my people], I perish"
   </fo:block>
</fo:block-container>
<fo:block-container writing-mode="rl-tb">❷
   <fo:block> The words of Queen Esther,
      &#x5D5;&#x5DB;&#x5D0;&#xFB2A;&#x5E8;
      &#x5D0;&#x5D1;&#x5D3;&#xFB4A;&#x5D9;
      &#x5D0;&#x5D1;&#x5D3;&#xFB4A;&#x5D9;
      "If I perish [in trying to save my people], I perish"
   </fo:block>
</fo:block-container>
```

❶ Left-to-right, top-to-bottom writing mode.

❷ Right-to-left, top-to-bottom writing mode.

In the following small code snippet, `<fo:bidi-override>` formatting object is used to turn English text inside out, writing Latin symbols right-to-left (as if they were Hebrew characters):

```
…
<fo:block>
  <fo:bidi-override unicode-bidi="bidi-override" direction="rtl">❶
```

```
      This text ought to be turned inside out.
    </fo:bidi-override>
  </fo:block>
  …
```

❶ `direction` property sets writing direction, and `unicode-bidi="bidi-override"` cancels the effects of the Unicode BIDI algorithm, forcing all characters in the text to go from right to left.

8.3. Links

There are two kinds of links in XSL FO:

- links to locations inside the document;
- links to external entities/locations.

Both are achieved using the same formatting object — `<fo:basic-link>`. To make an *internal link*, the referenced object must have an `id` attribute that is cited in the `internal-destination` attribute of the link object:

```
<fo:basic-link internal-destination="smiley"
               text-decoration="underline">Click here</fo:basic-link>
```

An *external link* must have an URI specified in the `external-destination` attribute:

```
<fo:basic-link external-destination="url('http://www.RenderX.com/')"
               text-decoration="underline"
               color="blue">RenderX Home</fo:basic-link>
```

Unlike HTML, no default formatting is applied to links in XSL FO; you should provide character-level properties on `<fo:basic-link>` to distinguish it from the rest of the text (in the example above, `color` and `text-decoration` make the link text blue and underlined, as in a browser).

Note the `url('…')` notation inside the `external-destination` attribute; this is required by the XSL 1.0 Recommendation. (XEP handles unwrapped URLs, too).

In XEP, URLs starting with explicit `"file:"` protocol specification are rendered as PDF inter-document links. All other links are treated as Internet URIs, and open in a browser.

8.4. Leaders

A leader is an object used in XSL FO to create horizontal rules, lengthy white spaces, dot-filled tabs etc. Here is an example of a horizontal inline rule used to form a nice separator:

```
<fo:block text-align="center">
  <fo:leader leader-length="2in"
             leader-pattern="rule"
             alignment-baseline="middle"❶
             rule-thickness="0.5pt" color="black"/>
  <fo:inline font="16pt ZapfDingbats"
             color="#E00000">&#x274B;❷</fo:inline>
  <fo:leader leader-length="2in"
             leader-pattern="rule"
             alignment-baseline="middle"
             rule-thickness="0.5pt" color="black"/>
</fo:block>
```

❶ This aligns the leader vertically to the middle baseline — more or less at the level where strokes of a small x character cross.

❷ This draws a red eight-lobe asterisk. All dingbats should be referenced by their Unicode values; see XEP documentation for a list of codes assigned to ZapfDingbats glyphs.

See also an example of leader usage in a section about page numbers.

8.5. Footnotes and Floats

To insert a *footnote* at the bottom of the page, you should use a `<fo:footnote>` formatting object. It contains two formatting objects as its children:

- `<fo:inline>` contains an inline content used as a footnote anchor;
- `<fo:footnote-body>` object stores the text of the footnote body; its content will be placed at the bottom of the page.

Lists are often used to format footnote bodies. The example below shows a typical case:

```
<fo:block>
  This text contains a footnote<fo:footnote>❶
    <fo:inline baseline-shift="super"
               font-size="smaller">(1)</fo:inline>
    <fo:footnote-body>
      <fo:list-block provisional-label-separation="0pt"
                     provisional-distance-between-starts="18pt"
                     space-after.optimum="6pt"❷>
        <fo:list-item>
          <fo:list-item-label end-indent="label-end()">
            <fo:block>(1)</fo:block>
          </fo:list-item-label>
          <fo:list-item-body start-indent="body-start()">
            <fo:block>Footnote text</fo:block>
          </fo:list-item-body>
        </fo:list-item>
      </fo:list-block>
    </fo:footnote-body>
  </fo:footnote>
  after the word "footnote".
</fo:block>
```

❶ Footnote opening tag is placed immediately after the word “footnote”; breaking a line here would cause the footnote citation to detach from the word.

❷ This serves to separate adjacent footnote bodies.

Normally, footnotes are divided from the rest of the text by a *separator*. Separators are created in a special region named `xsl-footnote-separator`. You can insert content into it using a `<fo:static-content>` element. The example below shows a separator consisting of a solid line:

```
<fo:page-sequence>
  <fo:static-content flow-name="xsl-footnote-separator">
    <fo:block>
```

```
      <fo:leader leader-pattern="rule"
                 leader-length="100%"
                 rule-style="solid"
                 rule-thickness="0.5pt"/>
    </fo:block>
  </fo:static-content>
  …
  …
  …
```

Floats are similar to footnotes: they define a block that drifts to the top/left/right side of the page while the text flows around it. A typical use for floats is to put a picture, a table, etc. aside so that it does not disrupt the flow of the text. Here is an example of a top-float:

```
<fo:block>
  This text includes a floating picture.
  <fo:float float="before">
    <fo:block text-align="center"
              border="1pt solid gray"
              font="bold italic 9pt Helvetica">
      <fo:block>
        <fo:external-graphic src="url('smile.gif')"/>
      </fo:block>
      <fo:block>
        Fig. 1: A Smiling Face
      </fo:block>
    </fo:block>
  </fo:float>
  This text follows the float anchor.
</fo:block>
```

☞ Due to implementation restrictions of XEP 4.9, top floats (`float="before"`) appear on the top of the column *next to the current* one. For details, refer to *XSL Formatting Objects in XEP 4.9* (`doc/spec.pdf` in the XEP package).

The next example shows how to create a dropcap using a side float:

```
<fo:block intrusion-displace="line"❶>
  <fo:float float="start"❷>
    <fo:block font="bold 50pt/38pt❸ Helvetica" color="red">T</fo:block>
  </fo:float>
   his text starts with a big red letter T that hangs beneath the
   baseline. Few initial lines of text are shortened to make room
   for the dropcap.
</fo:block>
```

❶ Property `intrusion-displace` controls interaction of formatting objects with side floats. *`"line"`* is the default value for `<fo:block>` (in this case, it could be omitted): it forces lines of text to shrink, leaving room for the float.

❷ As in many places before, *`"start"`* means “start of line”; in the Western writing mode, this float will drift to the left.

❸ Notice that in the `font` shorthand attribute, the `line-height` is set to a smaller value than the `font-size` (38 pt vs. 50 pt). This serves to remove blank area (also known as *leading*) before and after the character on the line.

8.6. Page Numbering and Page Number References

To insert the *current page number*, use `<fo:page-number>` element:

```
<fo:static-content flow-name="xsl-region-before">
  <fo:block text-align="end">Page <fo:page-number/></fo:block>
</fo:static-content>
```

To insert a reference to a page where a certain element resides, that element must have a unique `id` property. Then you can reference it by a `<fo:page-number-citation>` element:

```
<fo:external-graphic id="smiley" src="url('smile.gif')"/>
  …
  …
  …
As shown on the "Smiling Face" diagram
```

```
(see page <fo:page-number-citation ref-id="smiley"/>), …
```

Page number citation can be used to obtain the total number of pages in the document: just place an empty block at the end of the text and refer to its page number, like in the example below:

```
<fo:static-content flow-name="xsl-region-before">
  <fo:block text-align="end">
    Page <fo:page-number/>
    of <fo:page-number-citation ref-id="terminator"/>
  </fo:block>
</fo:static-content>
<fo:flow>
  …
  …
  …
  <fo:block id="terminator"/>
</fo:flow>
```

Another important use of page number citations is to create tables of contents and indices. The example below shows a typical TOC entry in XEP:

```
<fo:block text-align-last="justify"❶>
  1. Introduction
  <fo:leader leader-pattern="dots"/>❷
  <fo:page-number-citation ref-id="intro"/>
</fo:block>
```

❶ The `text-align-last` attribute makes the last line extend up to the right edge of the text. If a `<fo:leader>` element is present on the line, it will grow so as to absorb all the remaining free space on the line (but not more that its `leader-length.maximum`).

❷ This element creates a dotted fill. The default value for `leader-length.maximum` is *`"100%"`* (i.e. equal to the width of the surrounding block); that's why we need not explicitly specify a length here.

8.7. Markers

Markers are used to change the contents of a side region according to the contents of the body region. A typical task performed with markers is to create *running headers* — to put the division title at the header of the page.

Actual use of markers involves two formatting objects:

`<fo:marker>`

creates a fragment of the document tree and associates it with a span of text in the flow;

`<fo:retrieve-marker>`

picks up a fragment created by an `<fo:marker>` and pastes it into a side region.

The contents of a`<fo:marker>` can be any block-level or inline-level formatting objects. `<fo:marker>` elements (one or more) can be the initial children of any formatting object that produces areas (`<fo:block>`, `<fo:inline>`, etc., including `<fo:wrapper>`). A `<fo:marker>` can appear inside `<fo:flow>` only. It possesses an only attribute — `marker-class-name`, used as a key when retrieving it.

`<fo:retrieve-marker>` is an empty element that can occur inside `<fo:static-content>` only. When the formatter builds a page instance, it replaces occurrences of `<fo:retrieve-markers>` by matching `<fo:markers>` found in the text flow.

Here is a basic example of using markers:

```
<fo:static-content flow-name="xsl-region-before">
  <fo:block text-align="center">
    <fo:retrieve-marker retrieve-class-name="division"/>
  </fo:block>
</fo:static-content>
…
…
…
<fo:block>❶
  <fo:marker marker-class-name="division">❷
    Introduction
  </fo:marker>
  <fo:block font-weight="bold" text-align="center">
```

```
    1. Introduction
  </fo:block>
  <fo:block text-indent="0.5in">
    Let me introduce you to something …
    … … …
  </fo:block>
</fo:block>
```

❶ This block should reside somewhere inside <fo:flow>.

❷ Remember that <fo:markers> should always be initial children of their parent elements.

9. RenderX Extensions

9.1. Document Info

PDF documents can have a number of information fields associated to them; these are displayed in Acrobat Reader when you choose Document Info / General from the File menu. Standard fields are 'Author', 'Title', 'Creator', 'Subject', and 'Keywords'. XSL FO provides no means to set these values.

XEP 4.9 provides a mechanism to set these fields from within the XSL FO document, using additional formatting objects. To trace a clear distinction between objects comprised in the XSL 1.0 Recommendation and those added by RenderX, the namespace mechanism is used: any and all additional elements or properties will have a different namespace prefix, associated to another namespace:

```
xmlns:rx="http://www.renderx.com/XSL/Extensions"
```

We need two extension elements to express document information fields: <rx:meta-info> and <rx:meta-field>.

<rx:meta-info>

This element is merely a container for one or more <rx:meta-field> elements. It should be the first child of <fo:root>.

<rx:meta-field>

This element specifies a single name/value pair. It has two mandatory attributes: name and value. Current implementation of the PDF and PostScript generators recognizes four possible values for name:

- `name="author"` fills the ‘Author’ field in the resulting PDF file with a string specified by the `value` property;
- `name="title"` fills the ‘Title’ field;
- `name="subject"` fills the ‘Subject’ field;
- `name="keywords"` fills the ‘Keywords’ field.

All other values for `name` are ignored. The ‘Creator’ field in the PDF file is set to the name of the application that created the document — `"XEP 4.9"`. There is no method to control it from the source file.

Let's see a code example containing this extension:

```
<?xml version="1.0" encoding="iso-8859-1"?>
<fo:root xmlns:fo="http://www.w3.org/1999/XSL/Format"
         xmlns:rx="http://www.renderx.com/XSL/Extensions"❶>
  <rx:meta-info>❷
    <rx:meta-field name="author" value="John Doe"/>
    <rx:meta-field name="title" value="Sample Document"/>
    <rx:meta-field name="subject" value="PDF Generation Test"/>
    <rx:meta-field name="keywords" value="XSL, PDF, FO Extensions"/>
  </rx:meta-info>
  <fo:layout-master-set>
    …
    …
    …
```

❶ Here's the namespace declaration for RenderX additional elements. It is obligatory if you use extensions.

❷ Note the different namespace prefix.

9.2. PDF Bookmarks

PDF documents can have an associated outline of their structure, commonly referred to as *bookmarks*. Being specific to a particular output format, bookmarks aren't covered by the XSL 1.0 Recommendation. XEP 4.9 implements three extension elements that enable bookmarks in PDF output:

`<rx:outline>`

Top-level element of the document outline tree. It should be located before any `<fo:page-sequence>` elements, but after `<fo:layout-master-set>` and `<fo:declarations>` (if present). Contains one or more `<rx:bookmark>` elements.

`<rx:bookmark>`

This element holds information about a single bookmark. It contains a mandatory `<rx:bookmark-label>` element as its first child. Bookmark destination is expressed either by `internal-destination` property (for internal navigation), or by `external-destination` (for extra-document links).

This element can also contain nested `<rx:bookmark>` elements that produce subordinate bookmarks.

`<rx:bookmark-label>`

This element contains text of a bookmark label; it must be the first child of its parent `<fo:bookmark>`. Contents of this element is plain text.

Shown below is a typical markup of bookmarks for a document consisting of two chapters, each of them having two subchapters:

```
    …
    …
    …
  </fo:layout-master-set>
  <rx:outline>
    <rx:bookmark internal-destination="chap_1">❶
      <rx:bookmark-label>Chapter 1</rx:bookmark-label>
      <rx:bookmark internal-destination="subchap_1_1">❷
        <rx:bookmark-label>Subchapter 1.1</rx:bookmark-label>
      </rx:bookmark>
      <rx:bookmark internal-destination="subchap_1_2">❸
```

```
        <rx:bookmark-label>Subchapter 1.2</rx:bookmark-label>
      </rx:bookmark>
    </rx:bookmark>
    <rx:bookmark internal-destination="chap_2">❹
      <rx:bookmark-label>Chapter 2</rx:bookmark-label>
      <rx:bookmark internal-destination="subchap_2_1">
        <rx:bookmark-label>Subchapter 2.1</rx:bookmark-label>
      </rx:bookmark>
      <rx:bookmark internal-destination="subchap_2_2">
        <rx:bookmark-label>Subchapter 2.2</rx:bookmark-label>
      </rx:bookmark>
    </rx:bookmark>
  </rx:outline>
  <fo:page-sequence master-reference="page">❺
    …
    …
    …
```

❶ Bookmark for first chapter.

❷ Bookmark for first subchapter. Note that is nested into the bookmark for chapter. In Acrobat Reader, the chapter bookmark will be shown as a parent of the subchapter one.

❸ Bookmark for second subchapter of the first chapter.

❹ Bookmark for second chapter.

❺ Remember that `<rx:outline>` should be located immediately before the first page sequence.

9.3. Indexes

Building page number lists for back-of-the-book indexes is a common task. It is relatively easy to collect a list of references to locations of index terms in the text; but then, to turn it into a real index entry, one should exclude repeated page numbers and merge adjacent numbers into ranges. Neither of these two operations can be done by pure XSLT/XSL-FO means. In this situation, introducing an extension looks inevitable.

The task of building an index can be split in two fairly independent subtasks:

- mark up occurrences of index terms in the main text;

- specify composition and formatting of page number lists in the index.

For the first task, XEP introduces a special extension attribute: `rx:key`. It can be specified on any element that can carry an `id`; unlike the latter, it need not be unique across the document. Its value is used as a key to select elements for the page number list.

If the element bearing `rx:key` completely fits onto one page, it will be represented as a single page number in the page number list. If it spans multiple pages, its entry in the page number list will be formatted as a range from the first to the last of the spanned pages.

There is also a mechanism to specify ranges explicitly. Two extension elements serve for the purpose:

`<rx:begin-index-range>`

Starts a range. Has two attributes, both required:

`id`

a unique identifier used to establish coupling;

`rx:key`

index key used to select the range into a page number list.

`<rx:end-index-range>`

Ends a range. Has one attribute, required:

`ref-id`

a reference to a `<rx:begin-index-range>` that started the range.

These two elements always form a pair: `<rx:begin-index-range>` is required to have an `id` attribute, and `<rx:end-index-range>` should have a `ref-id` with the same value. In the page number list, the pair is represented by a range from the page where `<rx:begin-index-range>` is located to the page where its matching `<rx:end-index-range>` resides. These elements may be located anywhere inside `<fo:flow>`: there are no constraints on their nesting with respect to other elements.

The actual index entry is created by another extension element, `<rx:page-index>`. It picks elements from the text by their `rx:key` properties, and produces a sorted list of unique page numbers. Items in the list are separated by an intercalary text, specified by the `list-separator` property; default is comma+space (`", "`).

`<rx:page-index>` should contain one or more `<rx:index-item>` elements as children. Each `<rx:index-item>` has a required `ref-key` attribute, and selects elements that have an `rx:key` with the same value.

☞ In XEP 4.9, `<rx:index-item>` only searches elements preceding it in the document; no forward references are implemented. This limitation is likely to persist in future versions of XEP.

A basic entry in an index will look like this:

```
<fo:inline rx:key="key.elephant">Elephants</fo:inline> live in Africa. …
<fo:inline rx:key="key.elephant">African elephants</fo:inline> have big ears …
…
<fo:block text-align="center" font="bold 16pt Futura">INDEX</fo:block>
<fo:block>
    Elephants <rx:page-index>
                  <rx:index-item ref-key="key.elephant"/>
              </rx:page-index>
</fo:block>
```

There are more attributes of `<rx:index-item>` to control the look of the index entry:

`range-separator`

String used to separate page numbers that form a continuous range. Default is en dash: *`"–"`* (`U+2013`).

`merge-subsequent-page-numbers`

Controls whether sequences of adjacent page numbers should be merged into ranges. Default is *`"false"`*.

`link-back`

If set to *`"true"`*, page numbers are formatted as hyperlinks back to the location of their correspondent index terms. Default is *`"false"`*.

Besides that, `<rx:index-item>` can bear additional inline properties, applied individually to each page number generated from this element. This gives a possibility to differentiate presentation styles across the list, e.g. make references to primary definitions bold. The following example illustrates it:

```
<fo:inline rx:key="key.elephant">Elephants</fo:inline> live
                                                in Africa. …
<fo:inline rx:key="key.elephant.primary">Elephant</fo:inline> is
                                                a quadruped mammal. …
```

```
<fo:inline rx:key="key.elephant">African elephants</fo:inline> have big
                                                ears …
…
<fo:block text-align="center" font="bold 16pt Futura">INDEX</fo:block>
<fo:block>
    Elephants <rx:page-index>
                  <rx:index-item ref-key="key.elephant.primary"
                                 font-weight="bold"
                                 link-back="true"/>❶
                  <rx:index-item ref-key="key.elephant"/>
              </rx:page-index>
</fo:block>
```

❶ When duplicates are removed, initial `<rx:index-item>` entries take precedence. Therefore, references to primary `rx:key`s should be placed first inside `<rx:page-index>`.

9.4. Flow Sections

`<rx:flow-section>` is a generalization of a `<fo:block>` with `span="all"` attribute: it changes the column count for a part of `<fo:flow>`. Unlike the case of `span="all"`, the number of columns inside `<rx:flow-section>` can be set to any value.

The syntax for this extension is straightforward:

- `<rx:flow-section>` may only occur as a direct child of `<fo:flow>`;
- only block-level elements can be descendants of an `<rx:flow-section>`;
- `<rx:flow-section>` can take `column-count` and `column-gap` attributes to control the number of columns and the separation between them. Columns in a `<rx:flow-section>` are always balanced.

Here is a short example. A text is located on a two-column page; it starts with a title that spans all columns, and then continues with an abstract formatted into three columns. After the abstract, the text turns back to its primordial two-column state.

```
<fo:root xmlns:fo="http://www.w3.org/1999/XSL/Format"
         xmlns:rx="http://www.renderx.com/XSL/Extensions">
```

```
<fo:layout-master-set>
  <fo:simple-page-master>
    <fo:region-body margin="1in" column-count="2"/>
  </fo:simple-page-master>
</fo:layout-master-set>
<fo:page-sequence master-reference="xsl-region-body">
  <fo:flow flow-name="xsl-region-body">
    <fo:block font-size="16pt" span="all">TITLE</fo:block>
    <rx:flow-section column-count="3" column-gap="18pt">
     <fo:block font-size="10pt">
        Abstract occupies three columns …
      </fo:block>
    </rx:flow-section>
    <fo:block>
      Normal text continues in two columns as specified in the page master …
```

10. Conclusion

Congratulations! If you have read this far and understood how the above examples work, you are prepared to write your own XSL FO stylesheets. You can find more examples of XSL FO usage in the *RenderX Test Suite*, available at http://www.renderx.com/testcases.html. There, you can find several dozens of test documents that illustrate and test virtually every aspect of XEP's functionality. Sample stylesheets for XSL FO are also available from RenderX site [http://www.renderx.com].