

XSL-FO Tutorial

Copyright© [ECRION Software](http://www.ecrion.com) 2003-2005. All Rights Reserved.

This document is designed to help XML programmers to develop XSL-FO documents to be rendered using XF Rendering Server 2005. Please contact Technical Support at support@ecrion.com if you need additional information about this product.

Table of Contents

About XF Rendering Server 2005	3
Product Features	3
What is XSL-FO	4
How hard it is to learn?	4
Hello World	5
Paragraphs	6
Text Alignment	7
Fonts	7
Borders	8
Backgrounds	9
Flow Layout	12
Inline Text Formatting	14
Subscripts and Superscripts	14
Graphics	16
SVG	16
XChart	17
External Graphics	18
Floats	19
Absolute Positioning	19
Tables	21
Lists	23
Numbered Lists	23
Pagination	25
Footnotes	27
Markers	29
Bookmarks	31
Miscellaneous Inline Elements	32
Page Numbers	32
Hyperlinks	33
Leaders	33
Extensions	35
Index Entries	35
Marking	35
Page Indexes	35
Encryption	37
Metadata	38
Digital Signatures	39
Barcodes	41
Appendix A - Colors	42

Last updated on: November 2004

Important Notice: This document and the information within is furnished "as is" and is subject to change without notice. In no event shall the author be liable for any damages whatsoever (including, without limitation, damages for loss of business profits, business interruption, loss of business information, or any other pecuniary loss) arising out of the use of or inability to use this product, even if the author has been advised of the possibility of such damages.

This PDF document was generated using XF Rendering Server 2005. For the latest version, visit [Technical Resources](#) section on our web site.

Additional Resources

W3C XML Recommendation
<http://www.w3.org/TR/REC-xml>

W3C XSLT Recommendation
<http://www.w3.org/TR/1999/REC-xslt-19991116>

W3C XML Namespace Specifications
<http://www.w3.org/tr/rec-xml-names>

W3C XSL-FO Recommendation
<http://www.w3.org/TR/xsl>

W3C SVG Recommendation
<http://www.w3.org/TR/SVG11>

ECRION XChart 1.0 Language Reference
<http://www.ecrion.com/XF/PDF/xChart 1.0 Language Reference.pdf>

W3C RDF/XML Specifications
<http://www.w3.org/TR/2004/REC-rdf-syntax-grammar-20040210>

Adobe XMP Specifications
<http://www.adobe.com/products/xmp/pdfs/xmpspec.pdf>

About XF Rendering Server 2005

XF Rendering Server 2005 is a highly scalable, enterprise class rendering product. It can be used to automate the creation of electronic documents like technical manuals, brochures, proposals, business reports containing charts and graphs, by dynamically generating them from XML.

XF Rendering Server 2005 supports two major industry standards: XSL-FO (Extensible Style Language Formatting Objects) describing how an XML document should be formatted for a variety of media as well as SVG (Scalable Vector Graphics) used to describe two-dimensional vector and mixed vector/raster graphics in XML.

In addition high quality, all-purpose charts can be generated directly from XML using **XChart** XML language. More information can be found in [XChart 1.0 Language Reference](#).

Product Features

- Supports XSL-FO, SVG, XChart as input.
- Produces PDF, HTML, GIF, JPEG, PNG, BMP and other formats.
- Supports TrueType font embedding.
- Scalable server architecture that can run across multiple CPUs, meeting the high-performance needs of your applications.
- Is accessible from a multitude of development environments: C++, VB, ASP, .NET, Java.
- Includes XF Designer 2004 XSL-FO authoring tool.

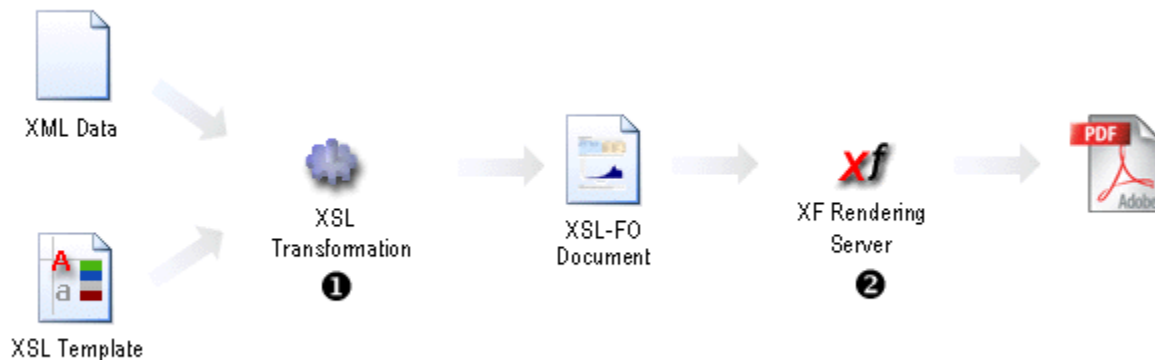
What is XSL-FO

XSL-FO is an XML language designed for describing all visual aspects of paginated documents. The well know HTML is another language for specifying formatting semantics, but is more usable for documents that are presented on screen, and less for materials destined for printing, because it does not support pagination elements like headers and footers, page size specifications, footnotes, etc.

XSL-FO is part of XSL language family:

- XSLT - (XSL Transformations) a language for transforming XML.
- XSL-FO - (XSL Formatting Objects) a language that can be used in XSLT for the purpose of "presenting" the XML

The following image depicts the steps required to produce a PDF document (or any other supported output format) using XSL:



- ❶ The XML data is transformed together with the XSL stylesheet to produce an XSL-FO document.
- ❷ The document is then converted to PDF.

How hard it is to learn?

The XSL-FO language uses CSS to describe formatting attributes like fonts, colors and borders, so from this point of view, it should be easy to learn by HTML developers. This manual will help you understand the language and accomplish more complicated tasks. The samples presented in this document are included when you install the product. The location of the samples has a path similar to the following. Here, it is assumed that C: is the XF Rendering Server 2005 installation drive:

```
C:\Program Files\Ecrion Software\XF Rendering Server 2005\XML Samples\XSL-FO\Tutorial
```

Hello World

Here is the traditional Hello World, XSL-FO style:

```
<?xml version="1.0" encoding="utf-8"?>
<fo:root xmlns:fo="http://www.w3.org/1999/XSL/Format">❶
  <fo:layout-master-set>❷
    <fo:simple-page-master master-name="LetterPage" page-width="8.5in"
      page-height="11in">❸
      <fo:region-body region-name="PageBody" margin="0.7in"/>
    </fo:simple-page-master>
  </fo:layout-master-set>
  <fo:page-sequence master-reference="LetterPage">
    <fo:flow flow-name="PageBody">
      <fo:block>Hello World</fo:block>❹
    </fo:flow>
  </fo:page-sequence>
</fo:root>
```



For the complete source code for this code example see "Tutorial/Hello World.fo" located under XML Documents Samples/Tutorial folder.

There are several things to notice:

❶ Any XML document must have only one root, and XSL-FO makes no exception. The root element for an XSL-FO document is *fo:root*.

The word "fo" before column character ":" is called a *namespace prefix*. An XML namespace is collection of names identified by an unique URL. It's main role is to avoid collisions when a single XML document contains elements and attributes defined by multiple software modules. The "fo" namespace prefix is linked with an unique URL, in this case "http://www.w3.org/1999/XSL/Format" using *xmlns* attribute. This syntax is based on [W3C XML Namespace Spec](#).

❷ ❸ The pages structure is defined using *fo:layout-master-set*; more about this in the chapter [Pagination](#). For now, is enough to say that it declares one type of page, 11.5 x 8 inches (US Letter).

❹ The "Hello World" paragraph is added into the page.

The result of rendering should be identical with the following figure.

Hello World

Figure 1

To convert this document into PDF, you can use XF Designer 2004. Open the document and generate the PDF from the tools menu. Or you can use *render.exe*, a console program located in "C:\Program Files\Ecrion Software\XF Rendering Server 2005\bin". The command line is:

```
render.exe -fo HelloWorld.fo -pdf C:\Temp\HelloWorld.pdf
```

The command line flag *-pdf* is optional; if not present render will generate a PDF file with a name identical with the input file's name and *.pdf* extension.

Paragraphs

In XSL-FO, paragraphs are created using *fo:block* elements. Various attributes can be set on a paragraph:

- Horizontal alignment is controlled by *text-align* attribute.
- Borders are set using *borders* attribute.
- Font is set using *font-family*, *font-size*, *font-weight*, etc.
- Spacing between two adjacent paragraphs is set using *space-before* and *space-after*.

```
<?xml version="1.0" encoding="utf-8"?>
<fo:root xmlns:fo="http://www.w3.org/1999/XSL/Format">
  <fo:layout-master-set>
    <fo:simple-page-master master-name="LetterPage"
      page-width="6in" page-height="5in">
      <fo:region-body region-name="PageBody" margin="0.7in"/>
    </fo:simple-page-master>
  </fo:layout-master-set>
  <fo:page-sequence master-reference="LetterPage">
    <fo:flow flow-name="PageBody" font-family="Arial" font-size="12pt"> ❶
      <fo:block text-align="justify" space-after="0.5cm"
        border="0.5pt solid green">❷
        This is the first paragraph of justified text. Notice how text
        fills all available space. The surrounding border is 0.5 points
        wide, solid, green color. This paragraph has a space-after
        equal to 0.5 cm.
      </fo:block>
      <fo:block text-align="justify" space-before="2cm"
        border="0.5pt dotted red">❸
        This is the second paragraph of justified text. This time
        the border is dotted and red. This paragraph has a
        space-before equal to 2 cm.
      </fo:block>
    </fo:flow>
  </fo:page-sequence>
</fo:root>
```



For the complete source code for this code example see "Tutorial/Simple Paragraphs.fo" located under XML Documents Samples/Tutorial folder.

In this example, we have two paragraphs of justified text, surrounded by borders, with 2 centimeters distance between them. The result of rendering should be identical with the following figure.

This is the first paragraph of justified text. Notice how text fills all available space for all lines except the last one. The alignment of the last line is controlled by *text-align-last* property.

This is the second paragraph. This block is left aligned.

Figure 2

There are several things to notice:

- ❶ We set the font to Arial, 12 points height on the parent *fo:flow* element.
- ❷ ❸ The distance between paragraphs is not additive; instead, the largest value will prevail. If a *fo:block* is the first or the last element in a page, you will also notice that the space is not present anymore! This behavior is controlled by *space-before.conditionality* and *space-*

after.conditionality attributes. If set to "retain", the corresponding space will not be discarded.

Text Alignment

Horizontal alignment of text is controlled by two attributes: *text-align* which will set the alignment for all lines of text, except the last one, which is set to *text-align-last*.

This is important to remember, because if your paragraph has only one line of text, you have to use *text-align-last* to set the alignment.

Possible values for *text-align* and *text-align-last* are:

- left (the default) to perform alignment to the right.
- right, to perform alignment to the right.
- center, to center the paragraph.
- justify, to justify the text, so that it fills the whole line.

Fonts

There are six properties that control the aspect of text: *font-family*, *font-style*, *font-variant*, *font-weight*, *font-size*, *line-height*.

To set the font face, you use *font-family* attribute. For example *font-family="Arial"* will specify that Arial font must be used.

If multiple font families are specified, the renderer will pick the first available, so you should list the fonts from the most specific to the most generic. For example, *font-family="Arial, Helvetica"* will select "Helvetica" if "Arial" is not present in the system.

The weight of the font can be specified using *font-weight* attribute. You can set it to either an absolute value ("bold" or "normal"), or to a value relative to parent element's font weight ("bolder" or "lighter").

To specify the font size, use *font-size* attribute. This size can be a length (1cm, 0.5in, 10pt, etc) or a percentage (0.5, 50%) from parent element's font.

A very important, and often misused property is *line-height*. This property determines the minimum line-height for a block element. The default value for *line-height* is 120%, that is, the line will be 20% taller than the text. For example, if the text is 10 points height, the line height will be of 12 points. The text will be centered on the line, 1 point from top, and 1 points from bottom. In the next example we set the line-height to 200%:

```
<?xml version="1.0" encoding="utf-8"?>
<fo:root xmlns:fo="http://www.w3.org/1999/XSL/Format">
  <fo:layout-master-set>
    <fo:simple-page-master master-name="LetterPage"
      page-width="6in" page-height="5in">
      <fo:region-body region-name="PageBody" margin="0.7in"/>
    </fo:simple-page-master>
  </fo:layout-master-set>
  <fo:page-sequence master-reference="LetterPage">
    <fo:flow flow-name="PageBody">
      <fo:block font-family="Arial" font-size="12pt" line-height="200%"
        border="1px outset blue">
        For this paragraph, the line-height is set to 200% ...
      </fo:block>
    </fo:flow>
  </fo:page-sequence>
</fo:root>
```



For the complete source code for this code example see "Tutorial/Line Height.fo" located under XML Documents Samples/Tutorial folder.

The result of rendering is displayed in following figure.

For this paragraph, the line-height is set to 200%. Because the font is 12 points height, an extra 12 points will be added to each line. The border is 3 points, solid inset.

Figure 3

We have mentioned before that this is the minimum line height: if a line contains an image of let's say 100 points in height, the total line height (only for that line) will be of 102 points.

To set all font properties at once you can use *font* shortcut attribute. A shortcut attribute will set the values for multiple attributes at once. The shortcut attribute *font* has the following syntax:

```
font="{style} {variant} {weight} {size}/{line-height} family".
```

For example, to achieve the same effect as the first example in this chapter, you could use `font="10pt Arial"`. Please note that by using this shortcut attribute, instead of specifying each font property individually, you will reset the values of non specified attributes:

```
<fo:flow flow-name="PageBody">
  <fo:block font="10pt bold Arial" border="1px solid blue">
    This is the parent block element with a 12 points, bold,
    Arial font.
  <fo:block font="10pt Verdana" border="1px solid red">
    This is the child block element; the same shortcut is
    used for changing the font, but because the weight
    was omitted, the font is no longer bold.
  </fo:block>
  When the child block ends, the remaining of parent's
  content is displayed.
</fo:block>
</fo:flow>
```



For the complete source code for this code example see "Tutorial/Inheritance.fo" located under XML Documents Samples/Tutorial folder.

The result of rendering is displayed in following figure.

This is the parent block element with a 10 points, bold, Arial font.

This is the child block element; the same shortcut is used for changing the font, but because the weight was omitted, the font is no longer bold.

When the child block ends, the remaining of parent's content is displayed.

Figure 4

XF Rendering Server 2005 can use both Type1 Postscript fonts and TrueType fonts. Fonts must be installed in Windows before use. To install a new font, simply drag and drop the font files in Control Panel/Fonts.

Borders

The border properties specify the width, color, and style of the borders a object. These properties apply to all XSL-FO elements.

- Border widths are set using *border-top-width*, *border-right-width*, *border-bottom-width*, *border-left-width* attributes. The value can be specified as a length (1pt, 0.5cm, etc). To set all four widths at once you can use *border-width* shorthand. For example, to set all borders to 0.1 inches thick, you use `border-width="0.1in"`.
- The border color properties are *border-top-color*, *border-right-color*, *border-bottom-color*, *border-left-color*. The shorthand for setting the color for all four borders at once is *border-color*.
- The border style properties specify the line style of a box's border (solid, double, dashed, etc.). The properties defined in this section refer to the *border-style* value type, which may take one of the following:

none, hidden	No border.
dotted	The border is a series of dots.
dashed	The border is a series of short line segments.
solid	The border is a single line segment.
double	The border is two solid lines. The sum of the two lines and the space between them equals the value of 'border-width'.
groove	The border looks as though it were carved into the canvas.
ridge	The opposite of 'grove': the border looks as though it were coming out of the canvas.
inset	The border makes the entire box look as though it were embedded in the canvas.
outset	The opposite of 'inset': the border makes the entire box look as though it were coming out of the canvas.

All borders are drawn on top of the box's background.

To set all attributes for a given border, you can use *border-top*, *border-right*, *border-bottom*, *border-left* shorthand attributes. The format is:

```
border-top="{width} {style} {color}".
```

The following notations are equivalent:

```
<fo:block border-top-style="solid" border-top-color="red" border-top-width="1mm">
...
</fo:block>
<fo:block border-top="1mm solid red">
...
</fo:block>
```

The *border* property is a shorthand property for setting the same width, color, and style for all four borders of a box. The *border* property cannot set different values on the four borders. To do so, one or more of the other border properties must be used. The format is:

```
border="{width} {style} {color}".
```

Backgrounds

XSL-FO object's backgrounds may be colors or images. Background properties allow authors to position a background image, repeat it, etc.

- To set the background color of an element use *background-color* attribute. It can be set to either a color value or the keyword "transparent".
- *background-image* sets the background image of an element. When setting a background image, authors should also specify a background color that will be used when the image is unavailable. When the image is available, it is rendered on top of the background color. (Thus, the color is visible in the transparent parts of the image). Values for this property are either an URL, to specify the image, or 'none', when no image is used.
- If a background image is specified, *background-repeat* attribute specifies whether the image is repeated (tiled), and how. It may take one of the following values:

repeat	The image is repeated both horizontally and vertically.
repeat-x	The image is repeated horizontally only.
repeat-y	The image is repeated vertically only.
no-repeat	The image is not repeated: only one copy of the image is drawn.

- If a background image has been specified, *background-position* property specifies its initial position. Values have the following meanings:

percentage percentage	With a value pair of '0% 0%', the upper left corner of the image is aligned with the upper left corner of the box's padding edge. A value pair of '100% 100%' places the lower right corner of the image in the lower right corner of padding area. With a value pair of '14% 84%', the point 14% across and 84% down the image is to be placed at the point 14% across and 84% down the padding area.
length length	With a value pair of '2cm 2cm', the upper left corner of the image is placed 2cm to the right and 2cm below the upper left corner of the padding area.
top left and left top	Same as '0% 0%'.
top, top center and center top	Same as '50% 0%'.
right top and top right	Same as '100% 0%'.
left, left center and center left	Same as '0% 50%'.
center and center center	Same as '50% 50%'.
right right center and center right	Same as '100% 50%'.
bottom left and left bottom	Same as '0% 100%'.
bottom bottom center and	Same as '50% 100%'.

center
bottom

bottom right Same as '100% 100%'.
and **right**
bottom

If only one percentage or length value is given, it sets the horizontal position only, the vertical position will be 50%. If two values are given, the horizontal position comes first. Combinations of length and percentage values are allowed, (e.g., '50% 2cm'). Negative positions are allowed. Keywords cannot be combined with percentage values or length values (all possible combinations are given above).

To set all attributes for one element's background, you can use *background* shortcut attribute. The format is:

```
background="{color} {image} {repeat} {position}".
```

Flow Layout

XSL-FO documents have flow layout, that is, content "flows" from one page to the next one:

```
<fo:flow flow-name="PageBody" font="bold 12pt Arial">
  <fo:block border="0.5pt solid blue" space-after="5pt">
    This is a first block element.
  </fo:block>
  <fo:block border="0.5pt solid blue" space-after="5pt">❶
    The content of this block is split across multiple pages.
    ...
  </fo:block>
  <fo:block border="0.5pt solid blue" keep-together="always">❷
    The content of this block is not split because
    keep-together is set.
    ...
  </fo:block>
  <fo:block border="0.5pt solid blue" space-after="5pt"
    keep-with-next="always">❸
    A block element that still fits on the second page.
  </fo:block>
  <fo:block border="0.5pt solid blue">
    A block on the third page. The previous block will be
    displayed on the second page because it has
    keep-with-next flag set.
  </fo:block>
</fo:flow>
```



For the complete source code for this code example see "Tutorial/Flow Layout.fo" located under XML Documents Samples/Tutorial folder.

The result of rendering is displayed in following figure.

The content of this block is splitted across multiple pages. The content of this block is splitted across multiple pages. The content of this block is splitted across multiple pages. The content of this block is splitted across multiple pages. The content of this block is splitted across multiple pages. The content of this

block is splitted across multiple pages. The content of this block is splitted across multiple pages. The content of this block is splitted across multiple pages.

This block has `keep-together` set to "always". Because of this flag, the block will be displayed on a new page as the renderer tries to prevent the block from splitting.

A block element that still fits on the previous page.

A block on the last page. The previous block will be displayed on the same page because it has `keep-with-next` flag set.

Figure 5

There are several properties that control how and when a block of text is split across multiple pages. They are:

- *break-before* and *break-after* attributes will force a page break before or after a block element. By block element we mean elements that fill all available horizontal space like [paragraphs](#), [tables](#) and [lists](#). For example, you might want to use this to start chapters on a new page.

- *keep-together* attribute prevents splitting of a block element. If there is not enough room to display the block on the current page, the block will be displayed on the next one (block ❷).
- *keep-with-next* and *keep-with-previous* will link a block element with the previous/next sibling block. This is useful to prevent page breaks occur between two closely related elements, like chapter title and chapter contents (block ❸).
- *widows* and *orphans* attributes are useful to control contextual information. The default values for this properties is "2", preventing the display of last line of a paragraph by itself at the top of a page (a widow) or the first line of a paragraph by itself at the bottom of a page (an orphan). You can see in the example above the fo:block ❶ will display the two line on the second page.

Inline Text Formatting

Inline elements allow XSL-FO developers to specify attributes for individual pieces of inline content (text and images), instead of the whole block.

In the example bellow, a fragment of text is filled with red, and it's font weight is set to bold:

```
<?xml version="1.0" encoding="utf-8"?>
<fo:root xmlns:fo="http://www.w3.org/1999/XSL/Format">
  <fo:layout-master-set>
    <fo:simple-page-master master-name="LetterPage" page-width="6in"
      page-height="3in">
      <fo:region-body region-name="PageBody" margin="0.7in"
        background-color="rgb(245,245,245)"/>
    </fo:simple-page-master>
  </fo:layout-master-set>
  <fo:page-sequence master-reference="LetterPage">
    <fo:flow flow-name="PageBody">
      <fo:block font="12pt Arial">
        Some
        <fo:inline font-weight="bold" color="red">inline text</fo:inline>
        formatting.
      </fo:block>
    </fo:flow>
  </fo:page-sequence>
</fo:root>
```



For the complete source code for this code example see "Tutorial/Inline Formatting.fo" located under XML Documents Samples/Tutorial folder.

The result of rendering is displayed in following figure.

Some **inline text** formatting.

Figure 6

Things to notice:

❶ The *fo:inline* element wraps the fragment "inline text" and sets *font-weight* to bold. The text color is set to red using *color* attribute.

Any color can be described using either a standard color value (see [Colors](#)) or by using it's red, green and blue components. The following notations are equivalent:

```
<fo:inline color="red">Hello</fo:inline>
<fo:inline color="rgb(255,0,0)">Hello</fo:inline>
```

Subscripts and Superscripts

Inline elements also allow creation of sub-scripts of super-scripts:

```
<?xml version="1.0" encoding="utf-8"?>
<fo:root xmlns:fo="http://www.w3.org/1999/XSL/Format">
  <fo:layout-master-set>
    <fo:simple-page-master master-name="LetterPage" page-width="6in"
      page-height="3in">
      <fo:region-body region-name="PageBody" margin="0.7in"
        background-color="rgb(245,245,245)"/>
    </fo:simple-page-master>
  </fo:layout-master-set>
  <fo:page-sequence master-reference="LetterPage">
```

```

<fo:flow flow-name="PageBody" font="12pt Arial">
  <fo:block>
    Normal text
    <fo:inline baseline-shift="sub">sub-script</fo:inline>❶
    normal text
    <fo:inline baseline-shift="super">super-script</fo:inline>❷
    normal text.
  </fo:block>
  <fo:block>
    Normal text
    <fo:inline baseline-shift="-50%">-50%</fo:inline>❸
    normal text
    <fo:inline baseline-shift="50%">+50%</fo:inline>
    normal text.
  </fo:block>
  <fo:block>
    Normal text
    <fo:inline baseline-shift="-5pt">-5pt</fo:inline>❹
    normal text
    <fo:inline baseline-shift="5pt">5pt</fo:inline>
    normal text.
  </fo:block>
</fo:flow>
</fo:page-sequence>
</fo:root>

```



For the complete source code for this code example see "Tutorial/Subscripts and Superscripts.fo" located under XML Documents Samples/Tutorial folder.

The rendering result is displayed in the next figure.

Normal text **sub-script** normal text **super-script** normal text.
 Normal text **-50%** normal text **+50%** normal text.
 Normal text **-5pt** normal text **5pt** normal text.

Figure 7

The property that controls the alignment of an inline element vertically within its parent line is *baseline-shift*. As you can see in this example, the text can be shifted vertically using either "sub" ❶ or "super" ❷ which will use font metrics to determine the subscript or superscript positions. You can also use a percentual ❸ or absolute ❹ value.

Graphics

XSL-FO provides the means to display images and vectorial graphics through two elements: *fo:instream-foreign-object* when you have the content embedded in the XSL-FO document and *fo:external-graphic* when the image resides in an external file.

SVG

One of the supported formats for *fo:instream-foreign-object* is SVG (Scalable Vector Graphics):

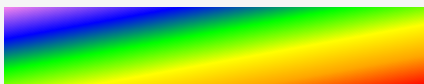
```
<fo:flow flow-name="xsl-region-body">
  <fo:block font-family="Arial" font-size="14pt">SVG Graphics Example</fo:block>
  <fo:block>
    <fo:instream-foreign-object> ❶
      <svg xmlns="http://www.w3.org/2000/svg" width="480" height="280">
        <linearGradient id="Grad1" gradientUnits="objectBoundingBox"
          x1="0" y1="0" x2="1" y2="1">
          <stop stop-color="rgb(238,130,238)" offset="0"/>
          <stop stop-color="blue" offset="0.2"/>
          ...
        </linearGradient>
        <!-- Linear gradient on the stroke of a rectangle -->
        <rect x="20" y="20" width="440" height="80" fill="url(#Grad1)" /> ❷
        <text font-family="Arial" font-size="14" x="20" y="130">
          Multi-color linear gradient.
        </text>
        <!-- Radial gradient on the stroke of a rectangle -->
        <radialGradient id="Grad2" gradientUnits="userSpaceOnUse"
          cx="240" cy="210" r="220" fx="240" fy="210">
          <stop stop-color="black" offset="0"/>
          <stop stop-color="yellow" offset="0.2"/>
          ...
        </radialGradient>
        <rect x="20" y="150" width="440" height="80" fill="url(#Grad2)" /> ❸
        <text font-family="Arial" font-size="14" x="20" y="260">
          Multi-color radial gradient.
        </text>
      </svg>
    </fo:instream-foreign-object>
  </fo:block>
</fo:flow>
```



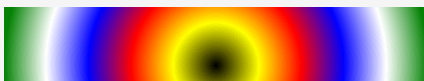
For the complete source code for this code example see "Tutorial/SVG Graphics.fo" located under XML Documents Samples/Tutorial folder.

The rendering result is displayed in the next figure.

SVG Graphics Example



Multi-color linear gradient.



Multi-color radial gradient.

Figure 8

The important points in example above are:

- ❶ *fo:instream-foreign-object* is used to wrap the SVG graphic.

② ③ Inside SVG, we fill two rectangles with gradients.

XChart

In addition to SVG, XF Rendering Server 2005 supports XChart, an XML language developed by Ecrion Software for the purpose of describing generic charts. The next example shows a mountain chart embedded in XSL-FO:

```
<fo:flow flow-name="PageBody">
  <fo:block font="bold 18pt Arial">
    Area Chart Example
  </fo:block>
  <fo:block>
    <fo:instream-foreign-object>
      <xc:root width="500pt" height="290pt"
        xmlns:xc="http://www.ecrion.com/xc"> ①
        <xc:graph x="5pt" y="5pt" width="450pt" height="280pt">
          <xc:plot-area>
            <xc:serie type="area" stroke-color="black" fill-color="red"
              stroke-width="1.5pt"> ②
              <xc:data-point category="1988-07-31" value="10000.00"/>
              <xc:data-point category="1988-08-31" value="9854.79"/>
              ...
            </xc:serie>
            <xc:serie type="area" stroke-color="black"
              fill-color="navy" stroke-width="1.5pt">
              <xc:data-point category="1988-07-31" value="9899.39"/>
              ...
            </xc:serie>
          </xc:plot-area>
          <xc:value-axis orientation="vertical"> ③
            <xc:title font-weight="bold">Value</xc:title>
            <xc:grid-lines stroke-color="silver"/>
            <xc:axis-labels offset="4pt" format="$#,##0"/>
          </xc:value-axis>
          <xc:category-axis orientation="horizontal"> ④
            <xc:major-tick-marks style="outside"/>
            <xc:title font-weight="bold">Date</xc:title>
            <xc:grid-lines stroke-color="silver"
              stroke-dash-array="2px 2px"/>
            <xc:axis-labels offset="3pt"/>
          </xc:category-axis>
        </xc:graph>
      </fo:instream-foreign-object>
    </fo:block>
  </fo:flow>
```



For the complete source code for this code example see "Tutorial/XChart.fo" located under XML Documents Samples/Tutorial folder.

The rendering result is displayed in the next figure.

Area Chart Example

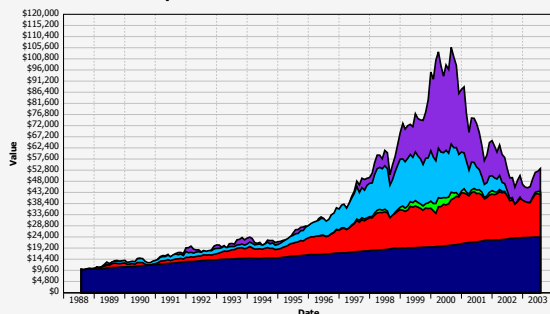


Figure 9

- ❶ Any XChart document has `xc:root` element.
 - ❷ ❸ ❹ We add several data series and define horizontal and vertical axis.
- For more information about XChart, please visit [XChart 1.0 Language Reference](#).

External Graphics

To display an image from an external file, use `fo:external-graphic`. All majors formats are supported, including BMP, JPEG, GIF, PNG, WMF, POSTSCRIPT, TIFF, etc. Unisys U.S. LZW Patent No. 4,558,302 used for GIF image compression expired on June 20, 2003, the counterpart patents in the United Kingdom, France, Germany and Italy expired on June 18, 2004, the Japanese counterpart patents expired on June 20, 2004 and the counterpart Canadian patent expired on July 7, 2004. For more information see [Unisys Web Site](#).

```
<fo:root xmlns:fo="http://www.w3.org/1999/XSL/Format">
  <fo:layout-master-set>
    <fo:simple-page-master master-name="LetterPage" page-width="3in" page-
height="1.8in">
      <fo:region-body region-name="PageBody" margin="0.1in"/>
    </fo:simple-page-master>
  </fo:layout-master-set>
  <fo:page-sequence master-reference="LetterPage">
    <fo:flow flow-name="PageBody" font-family="Arial" font-size="10pt">
      <fo:block font-weight="bold">
        External Graphics Example
      </fo:block>
      <fo:block>
        Text Before
        <fo:external-graphic src="ecrion-logo.png" ❶
content-height="0.7in" ❷
vertical-align="middle" ❸/>
        Text After
      </fo:block>
    </fo:flow>
  </fo:page-sequence>
</fo:root>
```



For the complete source code for this code example see "Tutorial/External Graphics.fo" located under XML Documents Samples/Tutorial folder.

The rendering result is displayed in the next figure.

External Graphics Example

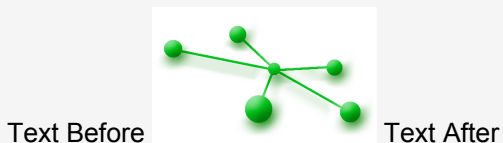


Figure 10

There are several things to notice in this example:

- ❶ Image urls can be either absolute or relative. When relative, the location of the XSL-FO document is used to compute the full path to the image. You can use `baseUrl` property (see [XF Rendering Server 2005 Programmers Reference](#)) to override this location.
- ❷ Image can be scaled using `content-width` and `content-height` properties. In this example we specify only the desired height and the width is computed automatically by the renderer, preserving the aspect ratio.
- ❸ Inline graphics can be shifted vertically using `vertical-align` attribute.

Floats

fo:float element inserts an out-of-line block-level element such as a figure or a pull quote onto the page. The float property determines which side of the page it floats on and the clear property determines whether and where other elements are allowed to float around it.

```
<fo:block text-align="justify">
  <fo:float float="start">❶
    <fo:block font-size="72pt" line-height="1" margin="5pt" text-depth="0"
  >L</fo:block>
</fo:float>orem ipsum dolor sit amet, consetetur sadipscing elitr ...
</fo:block>
```



For the complete source code for this code example see "Tutorial/DropCap.fo" located under XML Documents Samples/Tutorial folder.

The rendering result is displayed in the next figure.

Lorem ipsum dolor sit amet, consetetur sadipscing elitr, sed diam nonumy eirmod tempor invidunt ut labore et dolore magna aliquyam erat, sed diam voluptua. At vero eos et accusam et justo duo dolores et ea rebum. Stet clita kasd gubergren, no sea takimata sanctus est Lorem ipsum dolor sit amet. Lorem ipsum dolor sit amet, consetetur sadipscing elitr, sed diam nonumy eirmod tempor invidunt ut labore et dolore magna aliquyam erat, sed diam voluptua. At vero eos et accusam et justo duo dolores et ea rebum. Stet clita kasd gubergren, no sea takimata sanctus est Lorem ipsum dolor sit amet. Lorem ipsum dolor sit amet, consetetur sadipscing elitr, sed diam nonumy eirmod tempor invidunt ut labore et dolore magna aliquyam erat, sed diam voluptua. At vero eos et accusam et justo duo dolores et ea rebum. Stet clita kasd gubergren, no sea takimata sanctus est Lorem ipsum dolor sit amet.

Figure 11

There are several things to notice in this example:

- ❶ Each fo:float element should specify the floating side as either "start" or "end".
- ❷ Depending on the text you want to display, you may wish to eliminate the descent portion by setting the text-depth to 0.

Absolute Positioning

We have seen that XSL-FO documents have flow layout, that is, the content flows from one page to the next one, according to the rules imposed by page breaks, spacing, widows and orphans properties. However, sometimes it may be useful to position elements at absolute coordinates. You can achieve this by using *fo:block-container* element. In the example below, we have two fragments of text positioned under and over the main flow text.

```
<fo:flow flow-name="xsl-region-body" font-family="Verdana" font-size="10pt">
  <fo:block-container position="absolute" ❶
    top="10pt" left="30pt" height="14pt" width="100%" ❷>
    <fo:block font="72pt Arial" color="silver">Under</fo:block>
  </fo:block-container>
  <fo:block>
    <fo:block>
      Text Text Text Text Text Text Text Text Text Text ...
    </fo:block>
    <fo:block-container position="absolute"
      top="20pt" left="40pt" height="14pt" width="100%">
```

```
<fo:block font="72pt Arial" color="red">Over</fo:block>
</fo:block-container>
</fo:block>
<fo:block break-before="page"/>
<fo:block-container position="absolute"
  top="10pt" left="30pt" height="14pt" width="100%">
  <fo:block font="72pt Arial" color="silver">Under</fo:block>
</fo:block-container>
<fo:block>
  <fo:block>
    Text Text Text Text Text Text Text Text Text Text Text Text Text Text Text Text Text Text ...
  </fo:block>
  <fo:block-container position="absolute"
    top="10pt" left="30pt" height="14pt" width="100%">
    <fo:block font="72pt Arial" color="red">Over</fo:block>
  </fo:block-container>
</fo:block>
</fo:flow>
```



For the complete source code for this code example see "Tutorial/Absolute Positioning.fo" located under XML Documents Samples/Tutorial folder.

Text Text Text Text Text Text Text Text Text Text Text Text Text Text Text Text Text Text
Text Text Text Text Text Text Text Text Text Text Text Text Text Text Text Text Text Text
Text Text Text Text Text Text Text Text Text Text Text Text Text Text Text Text Text Text
Text Text Text Text Text Text Text Text Text Text Text Text Text Text Text Text Text Text
Text Text Text Text Text Text Text Text Text Text Text Text Text Text Text Text Text Text
Text Text Text Text Text Text Text Text Text Text Text Text Text Text Text Text Text Text
Text Text Text Text Text Text Text Text Text Text Text Text Text Text Text Text Text Text
Text Text Text Text Text Text Text Text Text Text Text Text Text Text Text Text Text Text
Text Text Text Text Text Text Text Text Text Text Text Text Text Text Text Text Text Text
Text Text Text Text Text Text Text Text Text Text Text Text Text Text Text Text Text Text

Text Text Text Text Text Text Text Text Text Text Text Text Text Text Text Text Text Text
Text Text Text Text Text Text Text Text Text Text Text Text Text Text Text Text Text Text
Text Text Text Text Text Text Text Text Text Text Text Text Text Text Text Text Text Text
Text Text Text Text Text Text Text Text Text Text Text Text Text Text Text Text Text Text
Text Text Text Text Text Text Text Text Text Text Text Text Text Text Text Text Text Text
Text Text Text Text Text Text Text Text Text Text Text Text Text Text Text Text Text Text
Text Text Text Text Text Text Text Text Text Text Text Text Text Text Text Text Text Text
Text Text Text Text Text Text Text Text Text Text Text Text Text Text Text Text Text Text
Text Text Text Text Text Text Text Text Text Text Text Text Text Text Text Text Text Text
Text Text Text Text Text Text Text Text Text Text Text Text Text Text Text Text Text Text

Figure 12

The important points in this document are:

- 1 *position* attribute is set to *absolute*
- 2 *top*, *bottom*, *left* and *right* coordinates are specified explicitly.

Tables

Tables are described in XSL-FO using *fo:table* element. A table can have a header (*fo:table-header*), a body (*fo:table-body*) and a footer (*fo:table-footer*). Each of these groups contain rows (*fo:table-row*), which in turn contain cells (*fo:table-cell*). The columns are described using *fo:table-column* elements.

```
<fo:table border-collapse="collapse" font-size="12pt" font-family="Arial"❶
  font-style="italic">
  <fo:table-column column-width="3in" background-color="rgb(255,246,206)"/>❷
  <fo:table-column column-width="50%"/>❸
  <fo:table-column column-width="50%"/>
  <fo:table-header color="rgb(255,255,255)" background-color="rgb(125,73,2)"❹
    font-weight="bold">
    <fo:table-row>
      <fo:table-cell padding="2pt" border="1pt solid black">
        <fo:block>Name</fo:block>
      </fo:table-cell>
      <fo:table-cell padding="2pt" border="1pt solid black">
        <fo:block>Quantity</fo:block>
      </fo:table-cell>
      <fo:table-cell padding="2pt" border="1pt solid black">
        <fo:block>Price</fo:block>
      </fo:table-cell>
    </fo:table-row>
  </fo:table-header>
  <fo:table-body>❺
    <fo:table-row>
      <fo:table-cell padding="2pt" border="1pt solid black">
        <fo:block>Cohiba red dot Corona Especial Cigars</fo:block>
      </fo:table-cell>
      <fo:table-cell padding="2pt" border="1pt solid black">
        <fo:block>25</fo:block>
      </fo:table-cell>
      <fo:table-cell padding="2pt" border="1pt solid black">
        <fo:block>$226.95</fo:block>
      </fo:table-cell>
    </fo:table-row>
    <fo:table-row>
      <fo:table-cell padding="2pt" border="1pt solid black"
        number-rows-spanned="2">❻
        <fo:block>Fuente Fuente Opus X</fo:block>
      </fo:table-cell>
      <fo:table-cell padding="2pt" border="1pt solid black">
        <fo:block>single</fo:block>
      </fo:table-cell>
      <fo:table-cell padding="2pt" border="1pt solid black">
        <fo:block>$28.95</fo:block>
      </fo:table-cell>
    </fo:table-row>
    <fo:table-row>
      <fo:table-cell padding="2pt" border="1pt solid black">
        <fo:block>25</fo:block>
      </fo:table-cell>
      <fo:table-cell padding="2pt" border="1pt solid black">
        <fo:block>$699.95</fo:block>
      </fo:table-cell>
    </fo:table-row>
    ...
  </fo:table-body>
</fo:table>
```



For the complete source code for this code example see "Tutorial/Table.fo" located under XML Documents Samples/Tutorial folder.

Name	Quantity	Price
Cohiba red dot Corona Especial Cigars	25	\$226.95
Fuente Fuente Opus X Perfeccion #4 Cigar cedar wrapped	single	\$28.95
	20	\$324.95

Name	Quantity	Price
Montecristo Double Corona Cigars	25	\$157.95

Figure 13

Things to notice:

- ❶ The `fo:table` element is defined. This table has the `border-collapse` attribute set to "collapse", which will cause cell borders to merge. Columns can have either a fixed width (column ❷) or a percentual width (column ❸).
- ❹ We define table's header and body. If a page break will occurs, the headers and the footers are displayed on the next page as well.
- ❺ Each `fo:table-cell` can span multiple rows and/or columns.

The content of the cell is aligned vertically according to `display-align` property.

Please note, that by default a cell will not clip its content. To clip the cell's content set `overflow` attribute to hidden.

The next sample illustrates these attributes:

```
<fo:table border-collapse="collapse" font-size="14pt" font-family="Arial">
  <fo:table-column column-width="50%" />
  <fo:table-column column-width="50%" />
  <fo:table-body>
    <fo:table-row>
      <fo:table-cell border="1pt solid black" display-align="center"
        height="2.4cm" overflow="hidden" ❶>
        <fo:block font-size="48pt" color="red">Clipped Cell</fo:block>
      </fo:table-cell>
      <fo:table-cell border="1pt solid black" display-align="center">
        <fo:block>Normal table cell.</fo:block>
      </fo:table-cell>
    </fo:table-row>
  </fo:table-body>
</fo:table>
```



For the complete source code for this code example see "Tutorial/Table Cell Clip.fo" located under XML Documents Samples/Tutorial folder.

The rendering result is displayed in the next figure.

Clipped Cell	Normal table cell.
-----------------	--------------------

Figure 14

This example also shows how to create fixed height rows by using `height` attribute (cell ❶).

Lists

XSL-FO lists are created using `fo:list-block` element. A list can contain one or more items (`fo:list-item`). Each item has a label (`fo:list-item-label`) usually used to display a bullet or a number, and a body (`fo:list-item-body`).

```

<fo:flow color="rgb(0,0,128)" flow-name="xsl-region-body" font-size="20pt">
  <fo:block>To do list:</fo:block>
  <fo:list-block> ❶
    <fo:list-item> ❷
      <fo:list-item-label end-indent="label-end()"> ❸
        <fo:block>
          1)
        </fo:block>
      </fo:list-item-label>
      <fo:list-item-body start-indent="body-start()">❹
        <fo:block>
          Very very important stuff
        </fo:block>
      </fo:list-item-body>
    </fo:list-item>
    <fo:list-item>
      <fo:list-item-label end-indent="label-end()">
        <fo:block>
          2)
        </fo:block>
      </fo:list-item-label>
      <fo:list-item-body start-indent="body-start()">
        <fo:block>
          Very important stuff
        </fo:block>
      </fo:list-item-body>
    </fo:list-item>
    ..
  </fo:list-block>
</fo:flow>

```



For the complete source code for this code example see "Tutorial/List.fo" located under XML Documents Samples/Tutorial folder.

The rendering result is displayed in the next figure.

To do list:
 1) *Very very important stuff*
 2) *Very important stuff*
 3) *Other important items*
 4) *Don't forget to eat*
 5) *Sleep would be good*

Figure 15

The important points in this document are:

- ❶ The list is created using `fo:list-block`.
- ❷ A list can have one or multiple items.
- ❸ ❹ Each item has a label, usually used to display a bullet or number, and a body.

Numbered Lists

XSL-FO does not provide an element to create numbered lists like HTML does; you have to generate the numbers using XSL techniques.

Considering the following source XML document:

```
<products>
  <product>Fuji FinePix F700</product>
  <product>Nikon CoolPix 5700</product>
  <product>Cannon Powershot A310</product>
</products>
```

We can create an XSL template that uses *xsl:number* element to generate numbers for each *fo:list-item-label*:

```
<?xml version="1.0" encoding="utf-8" ?>
<?xsl-test-case type="text/xml" href=".\\Numbered List.xml"?>
<xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
  <xsl:output method="xml" encoding="utf-8" indent="yes"/>
  <xsl:template match="/">
    <fo:root xmlns:fo="http://www.w3.org/1999/XSL/Format">
      <fo:layout-master-set>
        <fo:simple-page-master master-name="all">
          <fo:region-body region-name="xsl-region-body" margin="1in"/>
        </fo:simple-page-master>
      </fo:layout-master-set>
      <fo:page-sequence master-reference="all">
        <fo:flow flow-name="xsl-region-body">
          <fo:list-block>
            <xsl:for-each select="products/product">
              <fo:list-item>
                <fo:list-item-label end-indent="label-end()">
                  <fo:block>
                    <xsl:number/>
                  </fo:block>
                </fo:list-item-label>
                <fo:list-item-body start-indent="body-start()">
                  <fo:block>
                    <xsl:value-of select="." />
                  </fo:block>
                </fo:list-item-body>
              </fo:list-item>
            </xsl:for-each>
          </fo:list-block>
        </fo:flow>
      </fo:page-sequence>
    </fo:root>
  </xsl:template>
</xsl:stylesheet>
```



For the complete source code for this code example see "Tutorial/Numbered List.xml" and "Tutorial/Numbered List.xsl" located under XML Documents Samples/Tutorial folder.

Pagination

Every page has the following regions:

- *fo:region-body* which holds the main page content, that is, the content of *fo:flow*
- *fo:region-before*, used to display headers
- *fo:region-after*, used to display footers
- *fo:region-start* and *fo:region-end*, used to display side regions

Of all regions, *fo:region-body* can have multiple columns:

```
<fo:root xmlns:fo="http://www.w3.org/1999/XSL/Format">
  <fo:layout-master-set>
    <fo:simple-page-master master-name="all-pages" page-width="5in"
      page-height="5in">
      <fo:region-body region-name="Content" margin="0.7in" padding="6pt"
        column-gap="0.25in" column-count="2" />❶
      <fo:region-before region-name="Header" extent="0.7in" padding="6pt"
        display-align="after" background-color="red"/>❷
      <fo:region-after region-name="Footer" extent="0.7in" padding="6pt"
        display-align="before" background-color="blue"
        precedence="true"/>❸
      <fo:region-start region-name="LeftSide" extent="0.7in" padding="6pt"
        background-color="green" display-align="after"
        reference-orientation="90"/>❹
      <fo:region-end region-name="RightSide" extent="0.7in" padding="6pt"
        background-color="yellow" display-align="after"
        reference-orientation="-90"/>
    </fo:simple-page-master>
    <fo:page-sequence-master master-name="default-sequence">
      <fo:repeatable-page-master-reference master-reference="all-pages" />
    </fo:page-sequence-master>
  </fo:layout-master-set>
  <fo:page-sequence master-reference="default-sequence">
    <fo:static-content flow-name="Header">
      <fo:block>...</fo:block>
    </fo:static-content>
    <fo:static-content flow-name="Footer">
      <fo:block>...</fo:block>
    </fo:static-content>
    <fo:static-content flow-name="LeftSide">
      <fo:block>...</fo:block>
    </fo:static-content>
    <fo:static-content flow-name="RightSide">
      <fo:block>...</fo:block>
    </fo:static-content>
    <fo:flow flow-name="Content">
      <fo:block>
        The body region's content flows in two columns.
        The body region's content flows in two columns.
        ...
      </fo:block>
      <fo:block span="all" border="1pt solid red">❺
        This block has "span" attribute set to all, which will make
        it span all the columns in the page. Note that span attribute can
        be set only for those blocks with a fo:flow as the direct parent.
      </fo:block>
    </fo:flow>
  </fo:page-sequence>
</fo:root>
```



For the complete source code for this code example see "Tutorial/Regions.fo" located under XML Documents Samples/Tutorial folder.

The rendering result is displayed in the next figure.

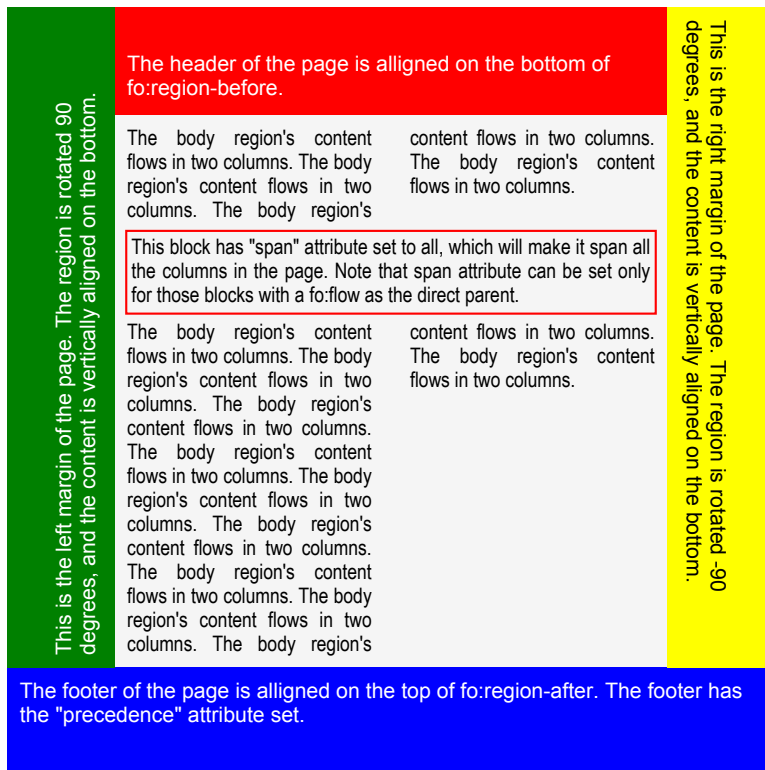


Figure 16

Every sequence of pages generated by the XSL-FO engine can have one or more page layouts associated with it:

- a) The simplest scenario: only one page layout for the whole document. All documents described so far belong to this category.
- c) Different page layouts for the first and subsequent pages, for the case when you want a cover page formatted differently than the rest of the pages.
- b) Different page layouts for even and odd pages, as it happens with most printed books, where the inside margin of a page is slightly larger than the outside margin, to allow binding.

The important points in this document are the following:

- ❶ We set the number of columns for the main flow to 2.
- ❷ A special area ("xsl-footnote-separator" is a reserved name) is created to hold the separator between the footnotes and the document body. This region is optional.
- ❸ The footnote object is declared, inline with the text. The first child element, *fo:inline* ❹ is used to format the number/symbol of the footnote.
- ❺ Footnote's body is defined.

Markers

You have probably noticed how in a book, the current chapter name is displayed on the header. To implement this feature, you need to understand XSL-FO marker elements.

First you "mark" (delimitate) pieces of your content as being retrievable for the purpose of displaying them in headers or footers. For this you use *fo:marker*. The marker is usually associated with a *fo:block*, therefore, the information from the marker can be displayed in all that pages where the *fo:block* generates areas.

Then in the headers or footer you tell the engine to display a marker using *fo:retrieve-marker*. There can be of course multiple markers in a certain page, so you have to use an unique name for each marker, as well as the retrieval rule: first marker with the given name present in the page, or the first that starts in the page, or the last one, etc.

The next example shows a two chapter document, with the chapter title being displayed in the header.

```
<fo:page-sequence master-reference="all-pages" font="italic 10pt Verdana">
  <fo:static-content flow-name="xsl-region-before">
    <fo:block>
      The first title present on this page:
      <fo:retrieve-marker retrieve-class-name="title" ❶
        retrieve-position="first-starting-within-page" ❷
        retrieve-boundary="page"/> ❸
    </fo:block>
  </fo:static-content>
  <fo:flow flow-name="xsl-region-body" font="10pt Verdana">
    <fo:block>
      <fo:block font="18pt 'Arial Narrow'">
        <fo:marker marker-class-name="title" ❹
          Title of Chapter 1
        </fo:marker>
        Chapter 1
      </fo:block>
      Text ...
    </fo:block>
    <fo:block break-before="page">
      <fo:block font="18pt 'Arial Narrow'">
        <fo:marker marker-class-name="title" ❹
          Title of Chapter 2
        </fo:marker>
        Chapter 2
      </fo:block>
      Text ...
    </fo:block>
  </fo:flow>
</fo:page-sequence>
```



For the complete source code for this code example see "Tutorial/Markers.fo" located under XML Documents Samples/Tutorial folder.

The rendering result is displayed in the next figure.

```
The first title starting on this page: Title of Chapter 1


---


Chapter 1
Text text text text text text text text text text text text text text text text text
text text text text text text text text text text text text text text text text text
text text text text text text text text text text text text text text text text text
text text text text text text text text text text text text text text text text text

The first title starting on this page: Title of Chapter 2


---


Chapter 2
Text text text, text
```

Figure 18

The important points in this document are:

- ④ In the body of the page we declare a marker for each *fo:block* in the flow. The second paragraph will go on the second page because it has the *break-before* attribute set.
- ① In the header region, that applies for every generated page, we retrieve the markers using *fo:retrieve-marker*. The marker "type" is specified in *retrieve-class-name* attribute.
- ② ③ Limit the scope of the marker retrieval to be in the same page as *retrieve-marker* element. First occurrence will be displayed.

Bookmarks

XF Rendering Server 2005 implements the bookmarks as defined by the latest [W3C Working Draft](#) for XSL-FO 1.1.

The *fo:bookmark-tree* formatting object is used to hold list of access points within the document such as a table of contents, a list of figures or tables, etc. Each access point is called a bookmark. The *fo:bookmark* object is used to identify an access point, and to specify where that access point is within the current document or another external document. A given bookmark may be further subdivided into a sequence of (sub-)bookmarks to as many levels as the authors desire:

```
<fo:root xmlns:fo="http://www.w3.org/1999/XSL/Format">
  <fo:layout-master-set>
    <fo:simple-page-master master-name="LetterPage" margin="1in">
      <fo:region-body region-name="PageBody"/>
    </fo:simple-page-master>
  </fo:layout-master-set>
  <fo:bookmark-tree>
    <fo:bookmark internal-destination="toc">
      <fo:bookmark-title>Bookmarks Example</fo:bookmark-title>
      <fo:bookmark internal-destination="chapter1">
        <fo:bookmark-title>Hello World</fo:bookmark-title>
      </fo:bookmark>
      <fo:bookmark internal-destination="chapter2">
        <fo:bookmark-title>Paragraphs</fo:bookmark-title>
      </fo:bookmark>
    </fo:bookmark>
  </fo:bookmark-tree>
  <fo:page-sequence master-reference="LetterPage" font="10pt Arial">
    <fo:flow flow-name="PageBody" font-family="Arial Narrow" font-size="10pt">
      <fo:block id="toc">Table of contents</fo:block>
      ...
    </fo:flow>
  </fo:page-sequence>
</fo:root>
```



For the complete source code for this code example see "Tutorial/Bookmarks.fo" located under XML Documents Samples/Tutorial folder.

The bookmarks displayed in Acrobat Reader can be used to navigate the PDF file.

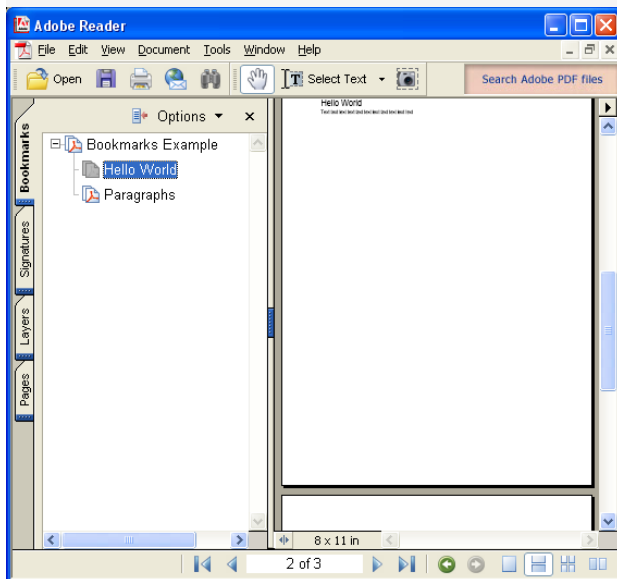


Figure 19

Miscellaneous Inline Elements

There are several inline elements left to describe:

Page Numbers

fo:page-number is used to insert the current page number

fo:page-number-citation is used to retrieve the page number of a give element. This element is also useful in inserting the number of pages in a document, as shown below:

```
<fo:root xmlns:fo="http://www.w3.org/1999/XSL/Format">
  <fo:layout-master-set>
    <fo:simple-page-master master-name="default" page-height="5cm"
      page-width="10cm" margin="5mm">
      <fo:region-body/>
      <fo:region-after region-name="footer" extent="0.5in"/>
    </fo:simple-page-master>
  </fo:layout-master-set>
  <fo:page-sequence master-reference="default" font-family="10pt Verdana">
    <fo:static-content flow-name="footer">
      <fo:block text-align="right" border-top="1pt dashed silver">
        Page
        <fo:page-number/>
        of
        <fo:page-number-citation ref-id="theEnd"/>
      </fo:block>
    </fo:static-content>
    <fo:flow flow-name="xsl-region-body">
      <fo:block>
        The text content of the first page.
      </fo:block>
      <fo:block break-before="page">
        The text content of the second page.
      </fo:block>
      <fo:block id="theEnd"/>
    </fo:flow>
  </fo:page-sequence>
</fo:root>
```



For the complete source code for this code example see "Tutorial/Page Count.fo" located under XML Documents Samples/Tutorial folder.

The rendering result is displayed in the next figure.

The text content of the first page.

Page 1 of 2

The text content of the second page.

Page 2 of 2

Figure 20

To start the numbering from a different page number use *initial-page-number* attribute of *fo:page-sequence*.

Hyperlinks

fo:basic-link can be used to display hyperlinks in a document, either to an external document, or as a cross reference within the current document.

As opposed to HTML, this element, does not underline the text or sets the text the color to blue; it just simply marks the area as being active.

You must use the standard properties like *color* and *text-decoration* to simulate the aspect of HTML hyperlinks.

```
<fo:block>
  Hyperlink to an external resource:
  <fo:basic-link color="blue" text-decoration="underline"
    external-destination="url(http://www.ecrion.com)">
    Ecrion Home
  </fo:basic-link>
</fo:block>
```

Leaders

fo:leader is a more complicated version of HTML's rule element. In the next example we will display a dotted leader in a table of contents:

```
<fo:flow flow-name="xsl-region-body" font-family="Arial Narrow" font-size="12pt">
  <fo:block font-size="18pt">
    Table of Contents
  </fo:block>
  <fo:block text-align-last="justify">
    <fo:basic-link color="blue" internal-destination="chapter1">
      Hello World
    </fo:basic-link>
    <fo:inline keep-together.within-line="always">
      <fo:leader leader-pattern="dots"/>
      <fo:page-number-citation ref-id="chapter1" />
    </fo:inline>
  </fo:block>
  <fo:block text-align-last="justify">
    <fo:basic-link color="blue" internal-destination="chapter2">
      Paragraphs
    </fo:basic-link>
    <fo:inline keep-together.within-line="always">
      <fo:leader leader-pattern="dots" />
      <fo:page-number-citation ref-id="chapter2" />
    </fo:inline>
  </fo:block>
  <fo:block id="chapter1" break-before="page" font-size="18pt">
```

```
    Hello World
  </fo:block>
</fo:flow>
```



For the complete source code for this code example see "Tutorial/Leader.fo" located under XML Documents Samples/Tutorial folder.

The rendering result is displayed in the next figure.

Table Of Contents	
Hello World	2
Paragraphs	3

Hello World

Text text text text text text text text text text

Paragraphs

Text text text text text text text text text text

Figure 21

Extensions

Index Entries

There are two main steps in creating indexes:

- Mark words, phrases or whole blocks.
- Insert page indexes.

Marking

Use *xf:key* attribute for any element that can have an *id*. While an *id* should always be unique, *xf:key* values may not be unique. All occurrences of a specific key will participate in generating the final index.

```
<fo:root xmlns:fo="http://www.w3.org/1999/XSL/Format"
  xmlns:xf="http://www.ecrion.com/xf/1.0">
  ...
  <fo:block xf:key="keywords.lion.range">
    <fo:inline xf:key="keywords.lion">Lions</fo:inline>, along with the other
      big cats such as tigers are in the genus Panthera.
    ...
  </fo:block>
  <fo:block>
    Some <fo:inline xf:key="keywords.lion">lions</fo:inline> are nomadic.
  </fo:block>
</fo:root>
```

In the example above, there are two distinct key values: *keywords.lion.range* and *keywords.lion*. Please note that the string value of key attribute can be anything, but for clarity purposes, we use a dotted notation in the examples presented in this chapter.

Page Indexes

To insert a list of pages corresponding to a index key, use *xf:page-index*.

```
<xf:page-index xmlns:xf="http://www.ecrion.com/xf/1.0"/
  ref-key="string"
  list-separator="string"
  range-separator="string"
```

list-separator represents the separator between non consecutive page numbers; the default value is ", ".

range-separator represents the separator between the first and last pages in a range; the default value is "-".

```
<fo:block>
  lions <xf:page-index ref-key="keywords.lion"/>
</fo:block>
<fo:block start-indent="0.2in">
  range <xf:page-index ref-key="keywords.lion.range"/>
</fo:block>
```



For the complete source code for this code example see "Tutorial/KeywordIndex.fo" located under XML Documents Samples/Tutorial folder.

In normal usage conditions, you would probably generate the index in a XSL transformation.

For example, considering the following XML document:

```
<doc title="African Lion">
  <section title="Classification & Range">
    <keyword>Lions</keyword>, along with the other big cats such as
    <keyword>tigers</keyword>
    ...
  </section>
</doc>
```

To automatically generate a list of keywords and their respective page index, you can use the following XSL template:

```
<xsl:stylesheet version="1.0"
  xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
  xmlns:fo="http://www.w3.org/1999/XSL/Format"
  xmlns:xf="http://www.ecrion.com/xf/1.0"
  xmlns:ms="urn:schemas-microsoft-com:xslt"
  xmlns:tt="samples-and-documentation"
  >
  <ms:script implements-prefix="tt" language="JScript">
    function toLower(str)
    {
      return str.toLowerCase();
    }
  </ms:script>
  <xsl:key name="kkey" match="//keyword" use="tt:toLower(string(text()))"/>
  <xsl:template name="GenerateIndex">
    <xsl:for-each select="//keyword">
      <xsl:sort select="." />
      <xsl:if test="generate-id(.) = generate-id(key('kkey',
tt:toLower(string(.)))[1])">
        <fo:block>
          <fo:inline text-transform="lowercase"><xsl:value-of select="."
/></fo:inline>
          <xf:page-index>
            <xsl:attribute name="ref-key">
              <xsl:value-of select="tt:toLower(string(.))"/>
            </xsl:attribute>
          </xf:page-index>
        </fo:block>
      </xsl:if>
    </xsl:for-each>
  </xsl:template>
</xsl:stylesheet>
```



For the complete source code for this code example see "Tutorial/KeywordIndex.xml" and "Tutorial/KeywordIndex.xsl" located under XML Documents Samples/Tutorial folder.

Encryption

PDF documents can be encrypted, and permission sets can be applied at rendering time using *xf:security* XSL-FO extension.

```
<xf:security xmlns:xf="http://www.ecrion.com/xf/1.0"/
  owner-password="password" user-password="password"
  encryption-strength="128 | 40"
  allow-printing="true | false"
  allow-modify-contents="true | false"
  allow-copy="true | false"
  allow-modify-annotations="true | false"
  allow-fill-in="true | false"
  allow-screen-readers="true | false"
  allow-assembly="true | false"
  allow-degraded-printing="true | false" >
```

owner-password and user-password

There are two passwords that can be specified for a document: an owner password and a user password.

- Opening the document with the correct owner password (assuming it is not the same as the user password) allows full (owner) access to the document. This unlimited access includes the ability to change the document's passwords and access permissions.
- Opening the document with the correct user password (or opening a document that does not have a user password) allows additional operations to be performed according to the user access permissions specified in the document's encryption dictionary. If user-password and owner-passwords are not specified, the document will be encrypted, with user-level access, and a random owner-password is generated; the user will not be prompted for a password, but nobody can have full access to the document. If only user-password is specified, a random owner-password is generated, the document will be encrypted and the user will be prompted for a password. Again, nobody can have full access to the document. If both passwords are specified, the document will be encrypted and the user will be prompted for password; depending on the password entered, the user can have full access, or only restricted access to the document.

encryption-strength

Specifies the encryption strength: 128 or 40 bits. 128 bits is the default.

allow-printing

Print the document (possibly not at the highest quality level, depending on whether allow-degraded-printing is also set).

allow-modify-contents

Modify the contents of the document by operations other than those controlled by allow-modify-annotations, allow-fill-in and allow-assembly.

allow-copy

Copy or otherwise extract text and graphics from the document by operations other than that controlled by allow-screen-readers.

allow-modify-annotations

Add or modify text annotations, fill in interactive form fields, and, if allow-modify-contents is also set, create or modify interactive form fields (including signature fields).

allow-fill-in

Fill in existing interactive form fields (including signature fields), even if allow-modify-annotations is not set.

allow-screen-readers

Extract text and graphics (in support of accessibility to disabled users or for other purposes).

allow-assembly

Assemble the document (insert, rotate, or delete pages and create bookmarks or thumbnail images), even if allow-modify-contents is not set.

allow-degraded-printing

When this is set (and allow-printing is set also), printing is limited to a low level representation of the appearance, possibly of degraded quality.

Metadata

There are two ways to include metadata in the PDF files generated by **XF Rendering Server 2005**:

a) Using *xf:info* for generic document information including author, name, subject and keywords data:

```
<fo:root xmlns:fo="http://www.w3.org/1999/XSL/Format">
  <xf:info xmlns:xf="http://www.ecrion.com/xf/1.0">
    <xf:title>About Metadata</xf:title>
    <xf:author>Joe Doe</xf:author>
    <xf:subject>Example Metata Document</xf:subject>
    <xf:keywords>PDF XML XMP</xf:keywords>
  </xf:info>
  ...
</fo:root>
```

b) Using *xf:meta* for metadata described in RDF/XML (Resource Description Framework) format.

xf:meta can contain one or many *rdf:RDF* nodes. For each node, the engine will create an **XMP** (Extensible Metadata Platform) packet in the generated PDF file.

```
<fo:root xmlns:fo="http://www.w3.org/1999/XSL/Format">
  <xf:meta xmlns:xf="http://www.ecrion.com/xf/1.0">
    <rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#">
      <rdf:Description about="" xmlns:my="http://www.mydomain.org/myschema/">
        <my:field>value</my:field>
        <my:my-collection>
          <rdf:Bag>
            <rdf:li>red</rdf:li>
            <rdf:li>green</rdf:li>
            <rdf:li>blue</rdf:li>
          </rdf:Bag>
        </my:my-collection>
      </rdf:Description>
    </rdf:RDF>
  </xf:meta>
</fo:root>
```

```

    </xf:meta>
    ...
</fo:root>

```

In this example, we have defined a object containing a member (called "field") and a collection containing three items.

Adobe PDF promotes **XMP** as a common standard that every application that works with PDF embedded metadata must understand. XMP supports a subset of RDF/XML. In addition XMP standardizes the definition, creation and processing of metadata by providing:

- A storage model. The format in which data is serialized in PDF (and in conformance with the XMP standard) is handled by XF Rendering Server 2005. This includes generating the envelope of your XML data, as well as encoding it properly.
- A set of predefined schemas. Adobe schemas provide property definitions that are relevant for a wide range of applications (including Adobe's editing and publishing products). One of the most interesting features is customization of FileInfo dialog in Adobe applications that support XMP.

Digital Signatures

XF Rendering Server 2005 offers support for generating digital signatures in PDF output. There are two types of signatures:

- **invisible signatures** - documents are signed and their authenticity can be verified, but they don't have an associated graphic element.
- **visible signatures** - signatures are associated with a graphic element; usually this element displays a scanned hand signature or a office stamp.

A digitally signed document can not be altered without invalidating the signature.

To generate an invisible signature:

```

<fo:root xmlns:fo="http://www.w3.org/1999/XSL/Format"
xmlns:xf="http://www.ecrion.com/xf/1.0">
  <xf:signature
    name="Dr. Joe Doe" location="Rockville, Maryland" reason="Prescription"
    certificate-serial-number="58 e9 4c 55 00 00 00 00 0c"
    certificate-issuer="Ecrion CA"/>
  ...
</fo:root>

```

To generate a visible signature:

```

<fo:root xmlns:fo="http://www.w3.org/1999/XSL/Format"
xmlns:xf="http://www.ecrion.com/xf/1.0">
  <xf:signature
    id="Sig1"
    name="Dr. Joe Doe" location="Rockville, Maryland" reason="Prescription"
    certificate-serial-number="58 e9 4c 55 00 00 00 00 0c"
    certificate-issuer="Ecrion CA"/>
  ...
  <fo:block ref-signature="Sig1">Signature</fo:block>
</fo:root>

```

Notes:

- "reason" attribute is optional
- "certificate-serial-number" and "certificate-issuer" must identify a certificate installed in XF Management Console (see below).
- You can use any element to reference a signature, for example a fo:inline-object or a fo:external-graphic

To generate a self signed certificate:

- Open XF Management Console
- Right click "Server Certificates" and click "Install Certificate"
- Click Self-signed certificate; click next

- Fill in all fields and click Finish
- The certificate should be displayed in the list of certificates.
- On Windows XP you can also click Properties to view the certificate

To generate a certificate request to be submitted to a Certification Authority (Thawte, Verisign)

- Open XF Management Console
- Right click "Server Certificates" and click "Install Certificate"
- Click "Certificate issued by a CA"; click Next
- Click "Prepare certificate request"; click Next
- Fill in all fields; click Next
- Enter a file name and click finish

To install a certificate issued by a CA:

- Open XF Management Console
- Right click "Server Certificates" and click "Install Certificate"
- Click "Certificate issued by a CA"; click Next
- Click "Process pending request"; click Next
- Enter a file name; click Finish
- The certificate should be displayed in the list of certificates.
- On Windows XP you can also click Properties to view the certificate

We recommend that you test using self signed certificates. If you also have a Windows Server computer you can setup your own CA and issue certificates to be used by XF.

You will need Acrobat 6.0 and higher to validate the signatures.

You will need Acrobat 7.0 or higher to validate signatures issued by a CA because Acrobat 6.0 will display an error message when a certificate chain is embedded in the signature.

Barcodes

XF Rendering Server 2005 offers support for drawing UPC-A, UPC-E, EAN-13, EAN-8 barcodes.

```
<xf:barcode xmlns:xf="http://www.ecrion.com/xf/1.0"/
  value="upc code" type="AUTO|UPC-A|UPC-E|EAN-13|EAN-8"
  bar-unit="length"
  fo:content-width="length"
  fo:content-height="length"
  fo:content-scaling="non-uniform|uniform"
  fo:font="font"
  fo:padding="padding"
  fo:border="border"
  fo:color="color">
```

value

- The code value specified with or without the check digit (the last digit). If the check digit is specified, then **type** attribute must be specified as well.
- Non-digit characters in the value are ignored.

type

- UPC-A
- UPC-E
- EAN-13
- EAN-8

If type attribute is not specified, XF will try to detect the code type.

bar-unit

- The length of one unit of the barcode. Barcode lines can be between 1 and 4 such units.

Normal formatting attributes (font, color) still apply, but they must be prefixed by the **fo:** namespace.

```
<fo:block>
  <xf:barcode value="075-67 816 4125" type="UPC-A" bar-unit="1px"
    fo:font="7.5pt Arial"
    fo:color="blue"
    fo:scaling="non-uniform" fo:content-height="50pt"
    fo:border="1pt solid purple" fo:padding="10pt"/>
</fo:block>
```



For the complete source code for this code example see "Advanced/Barcodes.fo" located under XML Documents Samples folder.

















The rendering result is displayed in the next figure.



Figure 22

Appendix A - Colors

A color value may either be a hexadecimal number (prefixed by a hash mark) or one of the following sixteen color names. The color names are case-insensitive.

	Black = "#000000"		Green = "#008000"
	Silver = "#C0C0C0"		Lime = "#00FF00"
	Gray = "#808080"		Olive = "#808000"
	White = "#FFFFFF"		Yellow = "#FFFF00"
	Maroon = "#800000"		Navy = "#000080"
	Red = "#FF0000"		Blue = "#0000FF"
	Purple = "#800080"		Teal = "#008080"
	Fuchsia = "#FF00FF"		Aqua = "#00FFFF"