Форматирующие объекты XSL

XML Bible (второе издание), Глава 18: Форматирующие объекты XSL

Данная серия документов подготовлена на основе материалов сайта Школы Консорциума W3C. Этот сайт является экспериментальным сервером, на котором содержание документов хранится в формате XML. Пользователям сайта эти документы доступны в виде HTML (преобразование на стороне клиента с помощью таблицы стилей XSLT) и в виде PDF (преобразование тех же документов в XSL-FO, а затем в формат PDF).

Содержание и вступление

Форматирующие объекты XSL (XSL-FO, Formatting Objects) - вторая половина расширяемого языка таблиц стилей XSL. XSL-FO - это XML-приложение, описывающее внешний вид стра-

6: http://xml.nsu.ru/extra/fo_6.xml
7: http://xml.nsu.ru/extra/fo_7.xml
8: http://xml.nsu.ru/extra/fo_8.xml
9: http://xml.nsu.ru/extra/fo_9.xml
10: http://xml.nsu.ru/extra/fo_10.xml
11: http://xml.nsu.ru/extra/fo_11.xml
12: http://xml.nsu.ru/extra/fo_12.xml

ниц, представляемых читателю. Для преобразования XML-документа, который использует некоторый семантический словарь, в новый XML-документ, который использует словарь XSL-FO, применяется таблица стилей, написанная на языке трансформаций XSL. Можно надеяться, что когда-нибудь веб-браузеры смогут прямо отображать данные, оформленные с помощью форматирующих объектов XSL, а пока нужен еще один шаг, на котором выходной документ дополнительно преобразуется в какой-то другой формат, например, в Adobe PDF.

Форматирующие объекты и их свойства

XSL-FO предоставляет более совершенную визуальную модель страницы, нежели HTML+CSS. В отличие от HTML+CSS, XSL-FO поддерживает такие типы форматирования, как направление письма справа налево и снизу вверх, сноски, заметки на полях документа, перекрестные ссылки на номера страниц и другие. В отличие от каскадных таблиц стилей CSS, которые преимущественно предназначены для веб-страниц, XSL-FO предназначен для более широкого круга задач. Можно, например, написать таблицу стилей XSL, которая с помощью форматирующих объектов сгенерирует верстку целой книги. Другая таблица стилей сможет сгенерировать из того же XML-документа веб-сайт.

Несколько замечаний относительно форматирующих объектов XSL

Язык XSL продолжает развиваться. Он уже радикально менялся и почти наверняка изменится в будущем. Эта глава основывается на предварительной рекомендации по XSL, датированной 21 ноября 2000 года. К тому моменту, когда вы будете читать эту книгу, эта рекомендация будет заменена официальной рекомендацией и некоторые детали синтаксиса XSL-FO могут при этом измениться. Если вы обнаружите, что некоторые примеры не работают, нужно будет сравнить их с официальной спецификацией по XSL-FO.

Что еще затрудняет дело, не существует программного обеспечения, которое полностью реализовывало бы данную предварительную рекомендацию по XSL. Фактически, есть только несколько отдельных программ, которые преобразуют документы XSL-FO в файлы PDF. Пока не существует браузеров, которые могли бы прямо отображать документы, написанные с помощью форматирующих объектов XSL. Конечно, в конце концов, по мере того, как данный стандарт будет приходить к своей устоявшейся форме, ситуация исправится, и большее количество разработчиков реализуют поддержку форматирующих объектов XSL.

Существует ровно 56 форматирующих объектов XSL. Они находятся в пространстве имен http://www.w3.org/1999/XSL/Format. В 99 процентах случаев выбирается префикс fo. Далее в этой главе я буду для данного пространства имен использовать префикс fo.

Из этих 56 элементов большая часть обозначает различные виды прямоугольных зон. Остальные в основном являются контейнерами для прямоугольных зон и пробелов. Вот эти форматирующие объекты в алфавитном порядке:

- fo:basic-link
- fo:bidi-override
- fo:block
- fo:block-container
- fo:character
- fo:color-profile
- fo:conditional-page-master-reference
- fo:declarations

- fo:external-graphic
- fo:float
- fo:flow
- fo:footnote
- fo:footnote-body
- fo:initial-property-set
- fo:inline
- fo:inline-container
- fo:instream-foreign-object
- fo:layout-master-set
- fo:leader
- fo:list-block
- fo:list-item
- fo:list-item-body
- fo:list-item-label
- fo:marker
- fo:multi-case
- fo:multi-properties
- fo:multi-property-set
- fo:multi-switch
- fo:multi-toggle
- fo:page-number
- fo:page-number-citation
- fo:page-sequence
- fo:page-sequence-master
- fo:region-after
- fo:region-before
- fo:region-body
- fo:region-end
- fo:region-start
- fo:repeatable-page-master-alternatives
- fo:repeatable-page-master-reference
- fo:retrieve-marker
- fo:root
- fo:simple-page-master
- fo:single-page-master-reference
- fo:static-content
- fo:table
- fo:table-and-caption
- fo:table-body
- fo:table-caption
- fo:table-cell

- fo:table-column
- fo:table-footer
- fo:table-header
- fo:table-row
- fo:title
- fo:wrapper

Модель форматирования XSL основана на прямоугольных областях, которые называются зонами (areas). Зоны могут содержать текст, пустое пространство, изображения и другие форматирующие объекты. Как и в CSS, зона имеет границы, отбивку (padding) на каждой стороне. В отличие от CSS, отступы зон (margins) заменяются объектами XSL space-before и space-after. Форматирующая машина XSL считывает форматирующие объекты и определяет, где на странице должна быть размещена каждая зона. Многие форматирующие объекты создают по одной зоне (по крайней мере, при обычных условиях), но из-за разбивок страниц, переносов и других деталей, которые нужно учитывать при размещении потенциально бесконечного количества текста в ограниченном пространстве, некоторые форматирующие объекты иногда создают несколько зон сразу.

Главным образом, форматирующие объекты различаются по тому, что они представляют. Например, форматирующий объект fo:list-item-label - это область, которая содержит значок элемента списка, число или другой символ, который размещается перед пунктом списка. Форматирующий объект fo:list-item-body - это область, которая содержит текст пункта списка, уже без значка. А форматирующий объект fo:list-item - это область, которая содержит форматирующие объекты fo:list-item-label и fo:list-item-body.

При обработке документ с форматирующими объектами разбивается на страницы. Окно веб-браузера будет трактоваться как одна очень большая страница. И напротив, формат для печати должен содержать много отдельных страниц. Каждая страница состоит из нескольких зон. Существует четыре основных вида зон:

- области (regions)
- блочные зоны (block areas)
- строковые зоны (line areas)
- внутри-строчные зоны (inline areas)

Они образуют подобие иерархии. Области содержат блочные зоны. Блочные зоны содержат другие блочные зоны, строковые зоны и текстовое содержимое. Строковые зоны содержат внутри-строчные зоны. Внутри-строчные зоны содержат другие внутри-строчные зоны и текстовое содержимое. Если говорить точнее:

- Область является в XSL-FO контейнером верхнего уровня. Страницы этой книги состоят из трех областей: заголовка, основной части и футера. Области создаются следующими форматирующими объектами: fo:region-body, fo:region-before, fo:region-after, fo:region-start и fo:region-end.
- Блочная зона соответствует блоковым элементам, например, параграфу или элементу списка. Хотя блочная зона может содержать в себе другие блочные зоны, перед началом и после окончания каждой блочной зоны должен быть перенос строки. Блочная зона не позиционируется точно по координатам, а размещается в содержащей ее зоне последовательно с другими блоками. Если перед данным блоком появляется или исчезает другой блок, положение блочной зоны соответственно сдвигается. Блочная зона может содержать парсируемые символьные данные, внутри-строчные зоны, строковые зоны и другие блочные зоны, которые последовательно располагаются внутри данной блочной зоны. В число форматирующих объектов, которые создают блочные зоны, входят fo:block, fo:table-and-caption и fo:list-block.
- Строковая зона соответствует строке текста внутри блока. Например, каждая строка в данном пункте списка является строковой зоной. Строковые зоны могут содержать внутри-

строчные зоны и внутри-строчные пробелы. Не существует форматирующих объектов, которые бы создавали чисто строковые зоны. Форматирующие машины вычисляют границы строковых зон, когда распределяют переносы строк внутри блочных зон.

- Внутри-строчные зоны - это части строки, например, отдельная буква, сноска или математическое выражение. Внутри-строчные зоны могут содержать другие внутри-строчные зоны или простой текст. В число форматирующих объектов, которые создают внутри-строчные зоны, входят: fo:character, fo:external-graphic, fo:inline, fo:instream-foreign-object, fo:leader и fo:page-number.

Форматирующие свойства

В совокупности, форматирующие объекты в XSL-FO-документе определяют порядок, в котором размещается на странице ее содержимое. Но имеются еще и форматирующие свойства, которые задают такие детали форматирования как размер, положение, шрифт, цвет, и многие другие. Форматирующие свойства задаются в виде атрибутов отдельных форматирующих объектов.

Многие из этих свойств должны быть вам знакомы из CSS. В настоящее время ведется работа, направленная на то, чтобы для одних и тех же вещей в CSS и XSL-FO использовались бы одни и те же названия. Например, CSS-свойство font-family означает то же самое, что и XSL-свойство font-family; и хотя синтаксис присваивания этим свойствам значений отличается, сам смысл значений аналогичен. Чтобы указать, что содержимое элемента fo:block должно быть отформатировано с применением шрифтов типа Times, можно использовать следующее CSS-правило:

```
fo:block {font-family: 'New York', 'Times New Roman', serif}
```

В XSL-FO для этого нужно включить атрибут font-family в начальный тэг элемента fo:block:

```
<fo:block font-family="'New York', 'Times New Roman', serif">
```

Хотя внешне выглядит по-разному, название стиля (font-family) и его значение ('New York', , serif) одинаковы. Свойство CSS font-family задается в виде списка названий шрифтов, разделенных запятыми в порядке от основного варианта к запасным. Свойство XSL-FO font-family задается в виде списка названий шрифтов, разделенных запятыми в порядке от основного варианта к запасным. И в CSS и в XSL-FO названия шрифтов, которые содержат пробелы, заключаются в кавычки. И CSS и XSL-FO понимают ключевое слово serif, которое указывает на произвольный шрифт с засечками.

Тем не менее, форматирующие объекты XSL поддерживают многие свойства, у которых в CSS нет эквивалента, например, destination-placement-offset, block-progression-dimension, character и hyphenation-keep - их нужно знать, чтобы в полной мере использовать преимущества XSL. Вот список стандартных свойств XSL-FO:

- absolute-position
- active-state
- alignment-adjust
- alignment-baseline
- auto-restore
- azimuth
- background
- background-attachment
- background-color

- background-image
- background-position
- background-position-horizontal
- background-position-vertical
- background-repeat
- baseline-shift
- blank-or-not-blank
- block-progression-dimension
- border
- border-after-color
- border-after-precedence
- border-after-style
- border-after-width
- border-before-color
- border-before-precedence
- border-before-style
- border-before-width
- border-bottom
- border-bottom-color
- border-bottom-style
- border-bottom-width
- border-collapse
- border-color
- border-end-color
- border-end-precedence
- border-end-style
- border-end-width
- border-left
- border-left-color
- border-left-style
- border-left-width
- border-right
- border-right-color
- border-right-style
- border-right-width
- border-separation
- border-spacing
- border-start-color
- border-start-precedence
- border-start-style
- border-start-width
- border-style

- border-top
- border-top-color
- border-top-style
- border-top-width
- border-width
- bottom
- break-after
- break-before
- caption-side
- case-name
- case-title
- character
- clear
- clip
- color
- color-profile-name
- column-count
- column-gap
- column-number
- column-width
- content-height
- content-type
- content-width
- country
- cue
- cue-after
- cue-before
- destination-placement-offset
- direction
- display-align
- dominant-baseline
- elevation
- empty-cells
- end-indent
- ends-row
- extent
- external-destination
- float
- flow-name
- font
- font-family
- font-selection-strategy

- font-size
- font-size-adjust
- font-stretch
- font-style
- font-variant
- font-weight
- force-page-count
- format
- glyph-orientation-horizontal
- glyph-orientation-vertical
- grouping-separator
- grouping-size
- height
- hyphenate
- hyphenation-character
- hyphenation-keep
- hyphenation-ladder-count
- hyphenation-push-character-count
- hyphenation-remain-character-count
- id
- indicate-destination
- initial-page-number
- inline-progression-dimension
- internal-destination
- keep-together
- keep-with-next
- keep-with-previous
- language
- last-line-end-indent
- leader-alignment
- leader-length
- leader-pattern
- leader-pattern-width
- left
- letter-spacing
- letter-value
- linefeed-treatment
- line-height
- line-height-shift-adjustment
- line-stacking-strategy
- margin
- margin-bottom

- margin-left
- margin-right
- margin-top
- marker-class-name
- master-name
- max-height
- maximum-repeats
- max-width
- media-usage
- min-height
- min-width
- number-columns-repeated
- number-columns-spanned
- number-rows-spanned
- odd-or-even
- orphans
- overflow
- padding
- padding-after
- padding-before
- padding-bottom
- padding-end
- padding-left
- padding-right
- padding-start
- padding-top
- page-break-after
- page-break-before
- page-break-inside
- page-height
- page-position
- page-width
- pause
- pause-after
- pause-before
- pitch
- pitch-range
- play-during
- position
- precedence
- provisional-distance-between-starts
- provisional-label-separation

- reference-orientation
- ref-id
- region-name
- relative-align
- relative-position
- rendering-intent
- retrieve-boundary
- retrieve-class-name
- retrieve-position
- richness
- right
- role
- rule-style
- rule-thickness
- scaling
- scaling-method
- score-spaces
- script
- show-destination
- size
- source-document
- space-after
- space-before
- space-end
- space-start
- space-treatment
- span
- speak
- speak-header
- speak-numeral
- speak-punctuation
- speech-rate
- src
- start-indent
- starting-state
- starts-row
- stress
- suppress-at-line-break
- switch-to
- table-layout
- table-omit-footer-at-break
- table-omit-header-at-break

- target-presentation-context
- target-processing-context
- target-stylesheet
- text-align
- text-align-last
- text-altitude
- text-decoration
- text-depth
- text-indent
- text-shadow
- text-transform
- top
- treat-as-word-space
- unicode-bidi
- vertical-align
- visibility
- voice-family
- volume
- white-space
- white-space-collapse
- widows
- width
- word-spacing
- wrap-option
- writing-mode
- xml:lang
- z-index

Преобразование в форматирующие объекты

XSL-FO - это полный XML-словарь, описывающий размещение текста на странице. Документ XSL-FO представляет собой обычный правильный XML-документ, созданный на основе этого словаря. Это означает, что у него должна быть XML-декларация, корневой элемент, дочерние элементы и так далее. Этот документ должен соответствовать всем условиям правильности XML-документов, иначе форматирующая машина его не примет. По договоренности, файлы, содержащие форматирующие объекты XSL, имеют трехбуквенное расширение .fob или двухбуквенное .fo. Но у них может быть и расширение .xml, поскольку они также являются и правильными XML-файлами.

На листинге 18-1 приводится пример простого документа, размеченного с помощью форматирующих объектов XSL. Корнем документа является элемент fo:root. Он содержит элементы fo:layout-master-set и fo:page-sequence. Элемент fo:layout-master-set содержит дочерние элементы fo:simple-page-master. Каждый элемент fo:simple-page-master описывает вид страницы, на которой располагается содержимое. В данном случае есть только одна простая страница, но в более сложных документах могут иметься различные мастер-страницы для ле-

вых и правых страниц, первой и внутренней страницы и так далее - и в каждой может быть свой набор отступов, нумерация страниц и другие свойства.

Содержание размещается на копиях мастер-страницы с помощью элемента fo:page-sequence. У этого элемента есть атрибут master-reference, указывающий имя используемой мастер-страницы. Дочерний элемент fo:flow хранит реальный материал, размещаемый на страницах. Этот материал представлен в виде двух дочерних элементов fo:block, у каждого из них определены свойства font-size со значением в 20 точек, font-family со значением serif и высота строки line-height в 30 точек.

Листинг 18-1: Простой документ XSL-FO

```
<?xml version="1.0"?>
<fo:root xmlns:fo="http://www.w3.org/1999/XSL/Format">
  <fo:layout-master-set>
    <fo:simple-page-master master-name="only">
      <fo:region-body/>
    </fo:simple-page-master>
  </fo:layout-master-set>
  <fo:page-sequence master-reference="only">
    <fo:flow flow-name="xsl-region-body">
      <fo:block font-size="20pt" font-family="serif"
                line-height="30pt">
        Hydrogen
      </fo:block>
      <fo:block font-size="20pt" font-family="serif"
                line-height="30pt" >
        Helium
      </fo:block>
    </fo:flow>
  </fo:page-sequence>
</fo:root>
```

Хотя подобный документ можно написать вручную, при таком подходе теряются все преимущества отделения в XML форматирования от содержания, Правильней преобразовывать в XSL-FO исходный XML-документ с помощью таблицы стилей XSLT. На листинге 18-2 приводится таблица стилей XSLT, которая преобразует документ 17-1[13] из предыдущей главы (описание двух первых химических элементов периодической таблицы в формате XML) в документ 18-1:

Листинг 18-2:
Преобразование из исходного словаря в документ форматирующих объектов XSL

```
<?xml version="1.0"?>
<xsl:stylesheet version="1.0"
  xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
  xmlns:fo="http://www.w3.org/1999/XSL/Format">
  <xsl:output 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="only">
          <fo:region-body/>
        </fo:simple-page-master>
      </fo:layout-master-set>
      <fo:page-sequence master-name="only">
        <fo:flow flow-name="xsl-region-body">
```

13: http://xml.nsu.ru/extra/samples/17-1.xml

```
          <xsl:apply-templates select="//ATOM"/>
        </fo:flow>
      </fo:page-sequence>
    </fo:root>
  </xsl:template>
  <xsl:template match="ATOM">
    <fo:block font-size="20pt" font-family="serif"
              line-height="30pt">
      <xsl:value-of select="NAME"/>
    </fo:block>
  </xsl:template>
</xsl:stylesheet>
```

Применение FOP

К моменту написания этой книги не существовало браузеров, которые бы могли прямо отображать XML-документы, преобразованные в форматирующие объекты XSL. Однако существует несколько приложений, которые могут конвертировать документ XSL-FO в другой просматриваемый формат, например, в PDF или TeX. В этой главе мы будем использовать приложение FOP, разработанное инициативой XML Apache project. FOP - это управляемое из командной строки Java-приложение, которое преобразует документы FO в файлы Adobe Acrobat PDF. В момент написания книги последней версией FOP являлась 0.18.1, которая не полностью поддерживает подмножество форматирующих объектов и свойств по предварительной рекомендации по XSL. Вы можете скачать последнюю версию FOP с xml.apache.org[14] .

Примечание

Довольно вероятно, что когда вы будете это читать, появятся новые версии FOP, которые будут лучше поддерживать XSL-FO. Так что действительно попробуйте скачать последнюю версию.

FOP - это написанная на Java программа, которая должна работать на любой платформе, на которой имеется совместимая с Java 1.1 виртуальная машина. Чтобы инсталлировать ее, нужно просто добавить в CLASSPATH включенные в дистрибутив FOP архивы fop.jar, xerces.jar и w3c.jar. Если вы используете Java 1.2 или более позднюю версию, вы можете просто поместить эти архивы в директорию jre/lib/ext.

Если вы используете Windows и установили не JRE, а JDK, обязательно поместите архивы fop.jar, xerces.jar и w3c.jar в обе директории jre/lib/ext. Одна из них находится там, где вы установили JDK, например, C:\jdk\jre\lib\ext. Другая может находиться здесь: C:\Program Files\JavaSoft\jre\1.3\lib\ext.

Кроме того, убедитесь, что вы используете файл w3c.jar из дистрибутива FOP, а не принадлежащий каким-либо другим проектам Apache, например, Batik. Различные его версии не совместимы.

Примечание переводчика

На момент перевода последним релизом FOP является 0.20.3. Она действительно реализует большую часть языка XSL-FO, хотя по-прежнему не полностью. С момента написания книги стандарт действительно претерпел некоторые изменения. Поскольку в оригинале примеры этой главы не соответствуют нынешнему стандарту по FO, примеры будут немного переработаны. Версия FOP 0.17.0, которую применял автор, не будет работать с обновленными примерами, так что лучше скачать именно 0.20.3. Устанавливается она еще проще: просто распакуйте содержание дистрибутива в какую-нибудь удобную директорию, например, в C:\FOP\

14: http://xml.apache.org/fop/

В документации заявлено, что FOP 0.20.3 работает с любой совместимой с Java 2 виртуальной машиной. Но опыты показали, что лучше использовать, как минимум JRE 1.3.

Класс org.apache.fop.apps.CommandLine содержит метод этой программы main(). Он запускается из командной строки с аргументами, задающими исходный и выходной файлы, например:

```
C:\> java org.apache.fop.apps.CommandLine 18-1.fo 18-1.pdf
```

Результат будет выглядеть следующим образом:

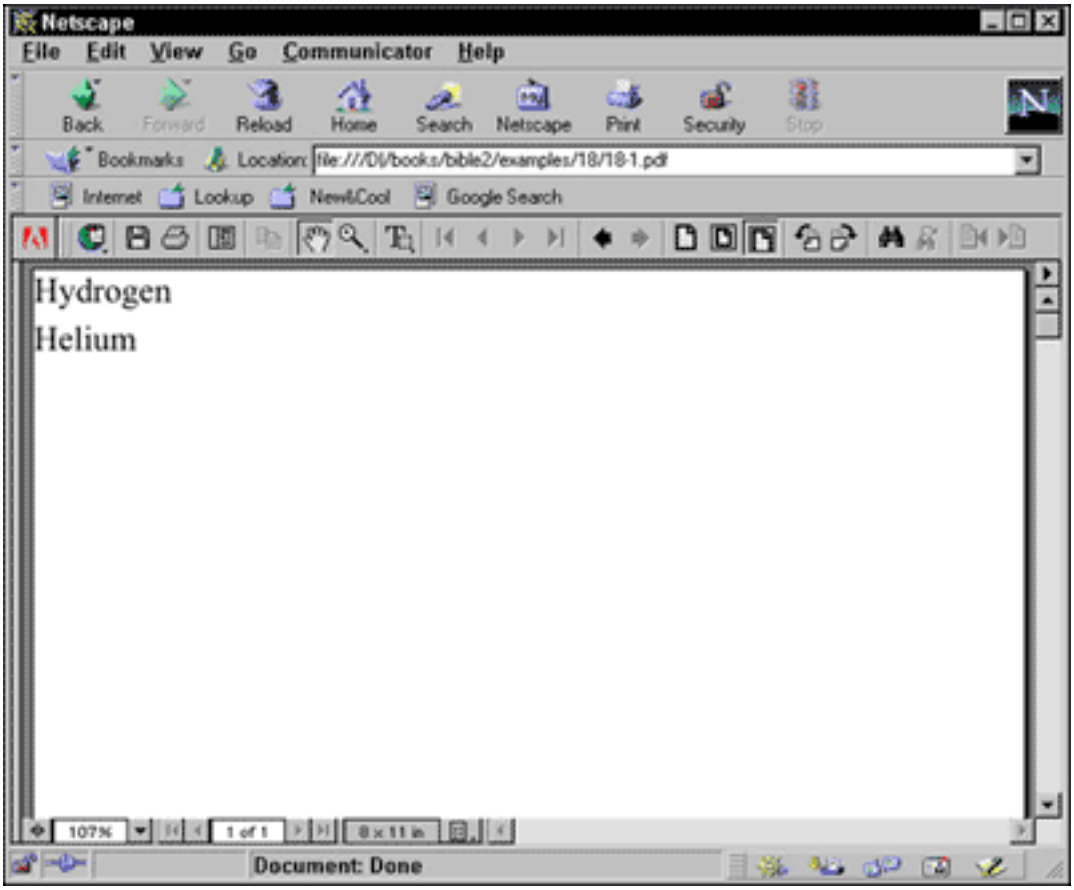
```
java org.apache.fop.apps.CommandLine 18-1.fo 18-1.pdf
FOP 0.17.0 DEV
using SAX parser org.apache.xerces.parsers.SAXParser
using renderer org.apache.fop.render.pdf.PDFRenderer
using element mapping org.apache.fop.fo.StandardElementMapping
using element mapping org.apache.fop.svg.SVGElementMapping
using element mapping
org.apache.fop.extensions.ExtensionElementMapping
using property list mapping
org.apache.fop.fo.StandardPropertyListMapping
using property list mapping
org.apache.fop.svg.SVGPropertyListMapping
using property list mapping
org.apache.fop.extensions.ExtensionPropertyListMapping
building formatting object tree
setting up fonts
formatting FOs into areas
 [1]
rendering areas to PDF
writing out PDF
```

Примечание переводчика

В версии FOP 0.20.3 имеется файл fop.bat, который делает всю "черную работу" - модифицирует CLASSPATH и запускает главный метод программы. Так что при работе с этой версией нужно в командной строке только ввести: E:\FOP>fop 18-1.fo 18-1.pdf. Тут файл 18-1.fo находится в той же директории, что и fop.bat, у нас это E:\FOP\. Результат 18-1.pdf также разместится в этой директории.

В данном случае 18-1.fo - это исходный XML-файл, использующий словарь форматирующих объектов. 18-1.pdf - это результирующий PDF-файл, который может быть просмотрен и распечатан с помощью Adobe Acrobat или другой программы, работающей с файлами PDF.

Хотя PDF-файлы создаются в ASCII-текстовом формате, это не книга посвящена PostScript, поэтому нам незачем углубляться в детали выходного PDF-документа. Если вам интересно, откройте PDF-файл в любом текстовом редакторе. Взглянем лучше на результат преобразования в PDF обычным способом:



PDF - это не единственный и даже не главный конечный формат XML-документов, преобразованных в форматирующие объекты XSL. Вполне можно рассчитывать, что не в очень отдаленном будущем веб-браузеры смогут прямо отображать форматирующие объекты XSL. А пока пост-преобразование в PDF - это единственный доступный способ отображения документов XSL-FO и в этой главе мы будем пользоваться им.

Макет страницы

Корневой элемент документа форматирующих объектов - элемент fo:root. Этот элемент содержит один элемент fo:layout-master-set и один или несколько элементов fo:page-sequence. В элементах fo:page-sequence размещается основное содержимое документа; то есть, располагающиеся на странице тексты и изображения. Элемент fo:layout-master-set содержит шаблоны создаваемых страниц. Когда форматирующая машина считывает XSL-FO-документ, она создает страницу на основе первого шаблона в элементе fo:layout-master-set. Затем она наполняет его содержимым из fo:page-sequence. Когда первая страница наполнена, форматер создает на основе шаблона вторую страницу и наполняет ее. Этот процесс продолжается до тех пор, пока не будет исчерпано все содержимое.

Корневой элемент fo:root

Элемент fo:root обычно имеет атрибут xmlns:fo со значением http://www.w3.org/1999/XSL/Format. Он может также иметь атрибут id (хотя обычно его нет). Элемент fo:root нужен только для объявления пространства имен и служит корневым элементом документа. Он не имеет прямого влияния на макет страницы или форматирование.

Простые мастер-страницы

Шаблоны страницы называются мастер-страницами. Мастер-страницы XSL-FO по назначению похожи на мастер-страницы в QuarkXPress или на мастера слайдов в PowerPoint. Каждый из них задает общий макет страницы, включая ее отступы, размеры заголовков, футеры, основную область страницы и так далее. Каждая реальная страница выводимого документа, которая была создана на основе одной из мастер-страниц, наследует у своей мастер-страницы определенные свойства, например, отступы, нумерацию страниц и общий макет. В XSL-FO 1.0 определяется только один вид мастеров страниц, а именно fo:simple-page-master, который задает прямоугольную страницу. Элемент fo:layout-master-set содержит один или несколько элементов fo:simple-page-master, каждый элемент задает отдельную мастер-страницу.

Замечание

В будущем в XSL-FO появятся и другие виды мастер-страниц, возможно и для непрямоугольных страниц.

Каждая мастер-страница представлена элементом fo:simple-page-master. Этот элемент задает макет страницы, включая размер пред-области (before region), основной области (body region), после-области (after region), конечной области (end region) и начальной области (start region). На рисунке приведен типичный макет расположения этих областей. На этой схеме можно заметить, что основная область перекрывает другие четыре области (но не отступы страницы), то есть, основная область - это все, включая начальную, конечную, пред- и после-области.

Замечание

В обычном английском тексте конечная область находится на правом крае страницы, а начальная область - на левом. В арабском языке все наоборот, потому что в нем строки пишутся справа налево. Почти во всех современных языках пред-область находится в верхней части страницы, а после-область - в нижней, но для языков, в которых письмо идет снизу вверх, расположение этих областей прямо противоположно.

Свойства простой мастер-страницы

Элемент fo:simple-page-master имеет три атрибута:

- master-name: имя, по которому последовательность страниц будет вызывать эту мастер-страницу
- page-height: высота страницы
- page-width: ширина страницы

Если атрибуты page-height и page-width отсутствуют, форматер выбирает некоторое разумное значение размера страницы по умолчанию, в зависимости от используемого типа носителя (например, 8.5” X 11”).

Среди других часто используемых в мастер-страницах атрибутов можно назвать:

- Атрибуты margin-bottom, margin-left, margin-right и margin-top, или обобщенный атрибут margin
- Атрибут writing-mode, который определяет направление, в котором идет текст на странице,

например, left-to-right (слева направо), right-to-left (справа налево) или top-to-bottom (сверху вниз)

- Атрибут reference-orientation, который задает поворот содержимого страницы с шагом в 90 градусов

Например, следующий элемент fo:layout-master-set содержит один элемент fo:simple-page-master с именем US-Letter. Этот элемент задает страницу размером 8.5 на 11 дюймов с полудюймовыми оступами с каждого края. Страница имеет единственную основную область, в которой размещается все содержимое страницы.

```
<fo:layout-master-set>
  <fo:simple-page-master  master-name="US-Letter"
     page-height="11in"   page-width="8.5in"
     margin-top="0.5in"   margin-bottom="0.5in"
     margin-left="0.5in"  margin-right="0.5in">
    <fo:region-body/>
  </fo:simple-page-master>
</fo:layout-master-set>
```

Области

Дизайнер устанавливает размер основной (центральной) области, заголовка, футера, конечной области и начальной области, а также расстояния между ними - все это делается с помощью дочерних элементов элемента fo:simple-page-master - каждый задает свою область:

- fo:region-before
- fo:region-after
- fo:region-body
- fo:region-start
- fo:region-end

Элементы fo:region-before и fo:region-after имеют атрибут extent, который задает высоту этих областей. А их ширина определяется шириной страницы. Элементы fo:region-start и fo:region-end также имеют атрибут extent, который задает их ширину. Их высота определяется расстоянием от низа начальной области до верха конечной. (Здесь идет речь об обычном западном письме. Ситуация выглядит иначе в китайском и некоторых других языках, где письмо ведется не слева направо и сверху вниз, а как-то иначе.)

Элемент fo:region-body не имеет атрибута extent. Считается, что размер основной области определяется размером страницы минус размеры отступов страницы. Таким образом, основная область охватывает все остальные области страницы. Если поместить текст в основную область и в другие четыре области, в некоторых местах текст будет перекрываться. Чтобы избежать этого, нужно установить левый отступ основной области не меньшим или большим, чем ширина начальной области, верхний отступ основной области - не меньше или больше высоты пред-области и так далее.

Каждая из пяти областей простой мастер-страницы может наполняться содержимым с помощью элементов fo:flow или fo:static-content по мере обработки документа. Однако, эти элементы не хранят содержимое, они просто задают границы областей, в которых форматтер будет размещать содержимое. Они являются макетами частей содержимого страницы, а не самим содержимым.

Например, следующий элемент fo:simple-page-master создает страницы с одно-дюймовыми пред- и после-областями. Высота основной области будет определяться расстоянием от низа

пред-области до верха после-области. Ее ширина будет определяться расстоянием от левого до правого края страницы, поскольку в данном случае нет начальной или конечной области.

```
<fo:simple-page-master master-name="table_page">
  <fo:region-before extent="1.0in"/>
  <fo:region-body margin-top="1.0in" margin-bottom="1.0in"/>
  <fo:region-after extent="1.0in"/>
</fo:simple-page-master>
```

В следующем примере элемент fo:layout-master-set задает размеры всех внешних областей в 1 дюйм, кроме того, страница со всех сторон имеет полудюймовые отступы:

```
<fo:layout-master-set>
  <fo:simple-page-master    master-name="only"
       page-width="8.5in"   page-height="11in"
       margin-top="0.5in"   margin-bottom="0.5in"
       margin-left="0.5in"  margin-right="0.5in">
    <fo:region-start  extent="1.0in"/>
    <fo:region-before extent="1.0in"/>
    <fo:region-body   margin="1.0in"/>
    <fo:region-end    extent="1.0in"/>
    <fo:region-after  extent="1.0in"/>
  </fo:simple-page-master>
</fo:layout-master-set>
```

Основные области страниц, созданных на основе этого мастера, будут иметь 5.5 дюймов в ширину и 8 дюймов в высоту. Эти размеры вычисляются вычитанием суммы отступов основной области и отступов страницы из размера страницы.

Последовательности страниц

Кроме элемента fo:layout-master-set, каждый документ форматирующих объектов содержит один или несколько элементов fo:page-sequence. Каждая страница в этой последовательности (page-sequence) имеет связанную с нею мастер-страницу, которая и определяет внешний вид страницы. Какая при этом действует мастер-страница - определяется атрибутом master-reference элемента fo:page-sequence. Он должен указывать на имя одной из мастер-страниц в элементе fo:layout-master-set. В документе на листинге 18-1 действует мастер fo:simple-master-page с именем only. Обычно используется не более одной мастер-страницы - когда их задается несколько, мастер-страницы должны быть сгруппированы в элементе fo:page-sequence-master. Например, вы можете использовать одну мастер-страницу для первой страницы каждой главы, вторую мастер-страницу - для всех внутренних страниц с правой стороны разворота, а третью - для внутренних страниц с левой стороны разворота. Или может использоваться одна простая мастер-страница для представления оглавления документа, а другая - для основного текста. В этом случае вы используете одну последовательность страниц для оглавления и для основного текста.

Каждая последовательность страниц содержит три элемента, идущие в следующем порядке:

- Необязательный элемент fo:title, он несет внутри-строчное содержимое, его можно использовать в качестве титула документа. Например, его можно поместить в титульной строке окна браузера, как это делает элемент TITLE в HTML
- Может содержать (или не содержать) несколько необязательных fo:static-content элементов, которые несут текст, в неизменном виде размещаемый на каждой странице
- Один элемент fo:flow, который содержит основные данные, перетекающие с одной страницы на другую

Главное различие между элементами fo:flow ("течение, поток") и fo:static-content ("статичное содержание") заключается в том, что текст из первого элемента размещается на страницах однократно, в то время как содержимое второго элемента повторяется на каждой странице. Например, слова, которые вы сейчас читаете - это текущее содержание, которое появляется только на этой странице, в то время как название главы в верхней части страницы является статичным содержимым и повторяется на всех страницах этой главы.

Элемент fo:flow содержит, соблюдая порядок, элементы, которые должны быть размещены на странице. Страницы наполняются элементами из этого потока - когда одна страница заполняется элементами из потока, создается новая страница следующей мастер-страницей из мастера последовательности страниц и наполняется оставшимися элементами потока. При использовании простой мастер-страницы для каждой страницы будет повторно использоваться одна и та же мастер-страница - пока содержимое потока не исчерпается.

Элемент fo:static-content содержит информацию, которая должна располагаться на каждой странице. Например, с помощью него в верхнюю часть каждой страницы можно помещать название книги. Статичное содержимое может изменяться в зависимости от мастер-страницы. Например, название главы может размещаться на левых страницах разворотов, а название книги - на правых. Элемент fo:static-content может применяться также для размещения, например, номеров страниц, которые должны вычисляться для каждой страницы. Другими словами, статичным будет являться не сам текст, а вычисление, которое выдает текстовый номер.

Потоки

Объект fo:flow хранит реальное содержимое, которое будет размещаться на страницах в соответствии с мастер-страницами. Это содержимое образуется последовательностью элементов fo:block, fo:block-container, fo:table-and-caption, fo:table и fo:list-block. В этом разделе мы основное внимание уделим основным элементам fo:block, которые являются приближенными эквивалентами HTML-элементов DIV. В остальных разделах мы подробнее познакомимся им с остальными блоковыми элементами, которые могут формировать поток.

Пример: вот поток, который содержит имена некоторых атомов, каждое имя заключено в свой собственный блок:

```
<fo:flow flow-name="xsl-region-body">
  <fo:block>Actinium</fo:block>
  <fo:block>Aluminum</fo:block>
  <fo:block>Americium</fo:block>
</fo:flow>
```

Атрибут flow-name элемента fo:flow имеет здесь значение xsl-region-body, которое указывает, в какой из пяти областей страницы будет размещаться поток. Допустимые значения этого атрибута таковы:

- xsl-region-body
- xsl-region-before
- xsl-region-after
- xsl-region-start
- xsl-region-end

Например, элемент flow, чье содержимое должно пойти в верхнюю часть страницы, должен иметь атрибут flow-name со значением xsl-region-before. Поток для основной части должен иметь атрибут flow-name со значением xsl-region-body. В одной и той же последовательности

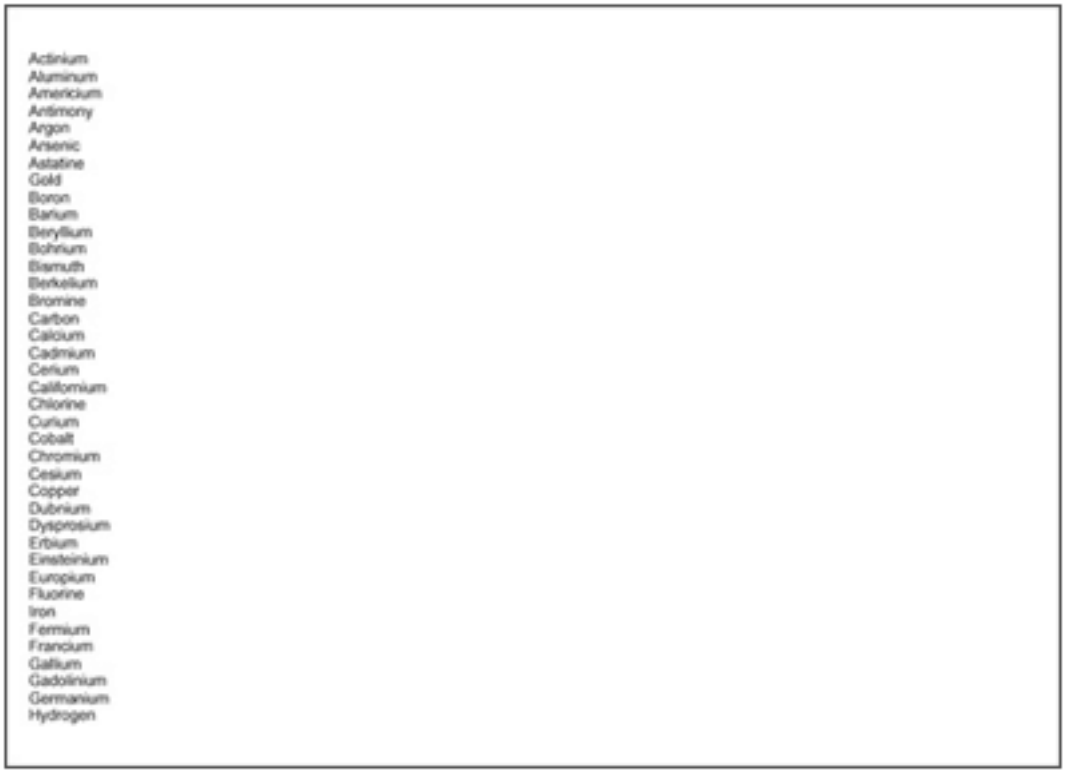
страниц не может быть двух потоков с одним и тем же именем. Таким образом, каждый элемент fo:page-sequence может содержать до пяти дочерних элементов fo:flow - по одному для каждой области страницы.

Теперь мы можем составить таблицу стилей, которая выведет полную периодическую таблицу химических элементов. На листинге 18-3 приводится таблица стилей XSLT, которая преобразует периодическую таблицу в форматирующие объекты XSL. Поток охватывает все химические элементы и помещает каждый из них в своем блоке. Простой мастер страниц с именем only задает мастер-страницу формата A4 в альбомном расположении, у которой с каждой стороны имеются полудюймовые оступы.

Листинг 18-3:
Базовая таблица стилей для периодической таблицы химических элементов

```
<?xml version="1.0"?>
<xsl:stylesheet version="1.0"
  xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
  xmlns:fo="http://www.w3.org/1999/XSL/Format">
  <xsl:template match="/">
    <fo:root xmlns:fo="http://www.w3.org/1999/XSL/Format">
      <fo:layout-master-set>
        <fo:simple-page-master master-name="A4"
           page-width="297mm"  page-height="210mm"
           margin-top="0.5in"  margin-bottom="0.5in"
           margin-left="0.5in" margin-right="0.5in">
          <fo:region-body/>
        </fo:simple-page-master>
      </fo:layout-master-set>
      <fo:page-sequence master-reference="A4">
        <fo:flow flow-name="xsl-region-body">
          <xsl:apply-templates select="//ATOM"/>
        </fo:flow>
      </fo:page-sequence>
    </fo:root>
  </xsl:template>
  <xsl:template match="ATOM">
    <fo:block><xsl:value-of select="NAME"/></fo:block>
  </xsl:template>
</xsl:stylesheet>
```

На скриншоте 18-3 - результирующий документ, полученный после преобразования таблицы химических элементов в форматирующие объекты, а затем в PDF-файл:



Статичное содержание

В то время, как каждый кусочек содержания элемента fo:flow появляется на страницах лишь однажды, каждый кусочек содержания элемента fo:static-content появляется на каждой странице. Например, если бы эта книга была собрана в виде документа XSL-FO, тогда и заголовок на верху каждой страницы и футер внизу каждой страницы могли бы создаваться элементами fo:static-content. Вы не обязаны использовать элементы fo:static-content, но если вы их все же применяете, они должны располагаться перед элементами fo:flow в описании последовательности страниц.

Элементы fo:static-content имеют те же самые атрибуты, что и элементы fo:flow, однако, поскольку fo:static-content не может при необходимости разбить свое содержимое на несколько страниц, в нем обычно содержится меньше данных, чем в элементах fo:flow. Например, документ, получающийся в результате действия таблицы стилей на листинге 18-4, использует элемент fo:static-content для размещения надписи "The Periodic Table" в верхней части таблицы:

Листинг 18-4:
Применение элемента fo:static-content для создания заголовка

```
<?xml version="1.0"?>
<xsl:stylesheet version="1.0"
  xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
  xmlns:fo="http://www.w3.org/1999/XSL/Format">
  <xsl:template match="/">
    <fo:root xmlns:fo="http://www.w3.org/1999/XSL/Format">
      <fo:layout-master-set>
        <fo:simple-page-master master-name="A4"
```

```
            page-width="297mm"   page-height="210mm"
            margin-top="0.5in"  margin-bottom="0.5in"
            margin-left="0.5in" margin-right="0.5in">
          <fo:region-before extent="1.0in"/>
          <fo:region-body margin-top="1.0in"/>
        </fo:simple-page-master>
      </fo:layout-master-set>
      <fo:page-sequence master-reference="A4">
        <fo:static-content flow-name="xsl-region-before">
          <fo:block>The Periodic Table</fo:block>
        </fo:static-content>
        <fo:flow flow-name="xsl-region-body">
          <xsl:apply-templates select="//ATOM"/>
        </fo:flow>
      </fo:page-sequence>
    </fo:root>
  </xsl:template>
  <xsl:template match="ATOM">
    <fo:block><xsl:value-of select="NAME"/></fo:block>
  </xsl:template>
</xsl:stylesheet>
```

На скриншоте 18-4 - последняя страница PDF-файла, созданного при помощи таблицы стилей 18-4. Текст "The Periodic Table" размещен на всех страницах документа:

Нумерация страниц

Элемент fo:page-sequence может иметь восемь необязательных атрибутов, определяющих нумерацию страниц в последовательности. Вот они:

- initial-page-number
- force-page-count
- format
- letter-value
- country
- language
- grouping-separator
- grouping-size

Атрибут initial-page-number задает номер первой страницы в последовательности. Чаще всего значение этого атрибута равно 1, но это может быть любое число, если предыдущие страницы создаются другим элементом fo:page-sequence или даже другим документом. Кроме того, значением атрибута может быть одно из трех ключевых слов:

- auto: 1, если предыдущий элемент fo:page-sequence не увеличит это значение. 1 - дефолтное значение.
- auto-odd: то же самое, что и auto, но добавляет 1, если данное значение является четным числом, то есть, начало на нечетной странице.
- auto-even: то же самое, что и auto, но добавляет 1, если данное значение является нечетным числом, то есть, начало на четной странице.

Атрибут force-page-count используется для указания, что документ должен иметь четное или нечетное количество страниц или должен заканчиваться на четной или нечетной странице. В таком указании иногда возникает необходимость при подготовке печатных изданий. Атрибут force-page-count может иметь одно из шести фиксированных значений:

- auto: делает последнюю страницу нечетной, если атрибут initial-page-number следующего элемента fo:page-sequence делает первую страницу следующей последовательности страниц четной. Делает последнюю страницу четной, если атрибут initial-page-number следующего элемента fo:page-sequence делает первую страницу следующей последовательности страниц нечетной. Если следующий элемент fo:page-sequence отсутствует или он не определяет атрибут initial-page-number, тогда номер последней страницы может быть любой.
- even: требует, чтобы общее число страниц в последовательности было четным, если нужно, вставляет для этого дополнительную пустую страницу.
- odd: требует, чтобы общее число страниц в последовательности было нечетным, если нужно, вставляет для этого дополнительную пустую страницу.
- end-on-even: требует, чтобы последний номер страницы был четным, если нужно, добавляет пустую страницу.
- end-on-odd: требует, чтобы последний номер страницы был нечетным, если нужно, добавляет пустую страницу.
- no-force: не требует ни четного ни нечетного количества страниц.

Атрибут country в качестве значения должен иметь код страны по стандарту RFC 1766. Атрибут language в качестве значения должен иметь код языка по стандарту RFC 1766[19]. Например, для указания на Великобританию следует использовать код en и код us для указания на США.

Перекрестная ссылка

По сути, здесь используются те же значения, что и допустимые значения атрибута xml:lang, который обсуждался в главе 11, за исключением того, что код страны и код языка в данном случае идут в два разных атрибута, а не в один атрибут.

Оставшиеся четыре атрибута имеют точно такой же синтаксис и значение, что и соответствующие атрибуты XSLT-элемента xsl:number, поэтому мы не будем повторяться.

Перекрестная ссылка

Элемент xsl:number и атрибуты format, letter-value, grouping-separator и grouping-size описывались в разделе "Преобразование чисел в строки" главы 17.

Форматирующий объект fo:page-number является пустым внутри-строчным элементом, который вставляет номер текущей страницы. Форматер должен определить, какой именно. Этот элемент может иметь несколько форматирующих атрибутов, общих для внутри-строчных эле-

19: http://www.ietf.org/rfc/rfc1766.txt

ментов, например, font-family и text-decoration. Например, таблица стилей на листинге 18-5 использует объекты fo:static-content и fo:page-number для выведения в нижней части каждой страницы ее номера:

Листинг 18-5:
Использование fo:page-number для размещения в футере номера страницы

```
<?xml version="1.0"?>
<xsl:stylesheet version="1.0"
  xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
  xmlns:fo="http://www.w3.org/1999/XSL/Format">
  <xsl:template match="/">
    <fo:root xmlns:fo="http://www.w3.org/1999/XSL/Format">
      <fo:layout-master-set>
        <fo:simple-page-master master-name="A4"
           page-width="297mm"  page-height="210mm"
           margin-top="0.5in"  margin-bottom="0.5in"
           margin-left="0.5in" margin-right="0.5in">
          <fo:region-before extent="1.0in"/>
          <fo:region-body margin-top="1.0in"
                          margin-bottom="1.0in"/>
          <fo:region-after  extent="1.0in"/>
        </fo:simple-page-master>
      </fo:layout-master-set>
      <fo:page-sequence master-reference="A4"
        initial-page-number="1" language="en" country="us">
        <fo:static-content flow-name="xsl-region-before">
          <fo:block>The Periodic Table</fo:block>
        </fo:static-content>
        <fo:static-content flow-name="xsl-region-after">
          <fo:block>p. <fo:page-number/></fo:block>
        </fo:static-content>
        <fo:flow flow-name="xsl-region-body">
          <xsl:apply-templates select="//ATOM"/>
        </fo:flow>
      </fo:page-sequence>
    </fo:root>
  </xsl:template>
  <xsl:template match="ATOM">
    <fo:block><xsl:value-of select="NAME"/></fo:block>
  </xsl:template>
</xsl:stylesheet>
```

На скриншоте 18-5 - вторая страница PDF-файла, сгенерированного при помощи таблицы стилей 18-5. Номер страницы размещается внизу этой и всех остальных страниц документа:


The Periodic Table

Erbium
Einsteinium
Europium
Fluorine
Iron
Fermium
Francium
Gallium
Gadolinium
Germanium
Hydrogen
Helium
Hafnium
Mercury
Holmium
Hassium
Iodine
Indium
Iridium
Potassium
Krypton
Lanthanum
Lithium
Lawrencium
Lutetium
Mendelevium
Magnesium
Manganese
p. 2

Мастера последовательностей страниц fo:page-sequence-master

Каждая создаваемая форматером страница связана с определенной мастер-страницей, расположенной в элементе fo:layout-master-set, которая задает внешний вид страницы. Какая именно мастер-страница используется - определяет атрибут master-reference элемента fo:page-sequence. В документах на листингах с 18-3 по 18-5 эту роль исполнял единственный элемент fo:simple-master-page с именем A4, но нередко используется и несколько мастер-страниц. Например, можно использовать одну мастер-страницу для первой страницы каждой главы, другую мастер-страницу для внутренних левых страниц разворота и третью - для внутренних правых страниц разворота. В этом случае мастер-страницы должны быть сгруппированы внутри элемента fo:page-sequence-master.

Элемент fo:page-sequence-master является дочерним элементом элемента fo:layout-master-set, в котором с помощью одного или нескольких следующих дочерних элементов перечисляется порядок, в котором будут инициализироваться конкретные мастер-страницы:

- fo:single-page-master-reference
- fo:repeatable-page-master-reference
- fo:repeatable-page-master-alternatives

У каждого из этих элементов есть атрибут master-reference, который определяет, когда и какая мастер-страница используется.

fo:single-page-master-reference

Простейший вариант - fo:single-page-master-reference. Атрибут master-reference этого элемента идентифицирует используемую мастер-страницу. Например, следующий элемент fo:layout-master-set содержит элемент fo:page-sequence-master с именем contents, который дает указание, что весь текст должен размещаться в одном экземпляре мастер-страницы с именем A4:

```
<fo:layout-master-set>
  <fo:simple-page-master master-name="A4"
      page-width="297mm"  page-height="210mm"
      margin-top="0.5in"  margin-bottom="0.5in"
      margin-left="0.5in" margin-right="0.5in">
    <fo:region-body/>
  </fo:simple-page-master>
  <fo:page-sequence-master master-name="contents">
    <fo:single-page-master-reference master-reference="A4"/>
  </fo:page-sequence-master>
</fo:layout-master-set>
```

Этот мастер последовательности страниц позволяет создать лишь одну страницу. Технически должна возникнуть ошибка, если содержание документа не влезает на одну страницу. Но на практике большая часть форматеров просто повторно использует последнюю мастер-страницу до тех пор, пока не будет размещено все содержание документа.

Рассмотри следующий мастер последовательности страниц:

```
<fo:page-sequence-master master-name="contents">
  <fo:single-page-master-reference master-reference="A4"/>
  <fo:single-page-master-reference master-reference="A4"/>
</fo:page-sequence-master>
```

Он предусматривает создание двух страниц, каждая создается на основе мастер-страницы A4. Если переполняется первая страница, создается вторая. Если переполнится и вторая, форматер может выдать ошибку, а может и просто создать третью страницу.

Та же самая техника может применяться и для использования различных мастер-страниц. Например, в следующем определении последовательности первая страница создается на основе мастер-страницы front, а вторая - на основе мастер-страницы back:

```
<fo:page-sequence-master master-name="contents">
  <fo:single-page-master-reference master-reference="front"/>
  <fo:single-page-master-reference master-reference="back"/>
</fo:page-sequence-master>
```

Первая созданная форматером страница будет основываться на мастер-странице front, а вторая на мастер-странице back. Если вторая страница переполнится, форматер может выдать ошибку или создать еще одну страницу на основе последнего использовавшегося мастера back.

fo:repeatable-page-master-reference

Разумеется, обычно заранее трудно предугадать, сколько страниц в итоге получится. Элемент fo:repeatable-page-master-reference позволяет определить, что сколько бы страниц не понадобилось для размещения всего содержимого документа, все страницы будут создаваться на основе одной мастер-страницы. Атрибутmaster-reference определяет, какая именно мастер-страница будет задействована. Например, следующий мастер последовательности страниц будет повторно использовать мастер страницу с именем A4 до тех пор, пока не ис-

черпается все содержимое документа:

```
<fo:page-sequence-master master-name="contents">
  <fo:repeatable-page-master-reference master-reference="A4"/>
</fo:page-sequence-master>
```

В качестве альтернативы можно установить атрибут maximum-repeats элемента fo:repeatable-page-master-reference и ограничить число создаваемых страниц. Например, следующий элемент fo:page-sequence-master генерирует не более 10 страниц на каждый документ:

```
<fo:page-sequence-master master-name="contents">
  <fo:repeatable-page-master-reference master-reference="A4"
                                       maximum-repeats="10"/>
</fo:page-sequence-master>
```

Это в числе прочего позволяет вам, например, применять один мастер для первых двух страниц, другой мастер - для следующих трех, а последний мастер - для очередных 10 страниц.

fo:repeatable-page-master-alternatives

Элемент fo:repeatable-page-master-alternatives задает различные мастер-страницы для первой страницы, четных и нечетных страниц, пустых страниц, последних четных и последних нечетных страниц. Он предназначен для создания печатных книг, в которых первые и последние страницы, а также четные и нечетные страницы традиционно имеют различные отступы, заголовки и футеры.

Поскольку элемент fo:repeatable-page-master-alternatives должен ссылаться более, чем на одну мастер-страницу, он не может использовать атрибут master-reference также, как элементы fo:single-page-master-reference и fo:repeatable-page-master-reference. Вместо этого используется его дочерние элементы fo:conditional-page-master-reference. У каждого из них имеется атрибут master-reference, идентифицирующий мастер-страницу, которая активизируется при определенных условиях. Сами условия определяются следующими тремя атрибутами:

- page-position: Этот атрибут может иметь значения first, last, rest или any, указывающий соответственно, что мастер-страница должна применяться для первой страницы документа, последней страницы, всех страниц кроме первой и для любой страницы.
- odd-or-even: Этот атрибут может иметь значения odd, even или any, указывающий соответственно, что мастер-страница должна применяться только для нечетных страниц, только для четных страниц или для всех страниц.
- blank-or-not-blank: Этот атрибут может иметь значения blank, not-blank или any, указывающий соответственно, что мастер-страница должна применяться только к пустым страницам, только к непустым страницам или ко всем страницам.

Например, следующий мастер последовательности страниц указывает, что первая страница должна быть основана на мастер-странице, имеющей имя letter_first, а все последующие страницы должны использовать мастер-страницу letter:

```
<fo:page-sequence-master master-name="contents">
  <fo:repeatable-page-master-alternatives>
    <fo:conditional-page-master-reference
      page-position="first" master-reference="letter_first"/>
    <fo:conditional-page-master-reference
      page-position="rest"  master-reference="letter"/>
  </fo:repeatable-page-master-alternatives>
```

```
</fo:page-sequence-master master-reference="contents">
```

Если содержимое документа не вмещается на первой странице, содержимое перетечет на вторую страницу. Если закончится место на второй странице, будет создана третья страница - будет создано столько страниц, сколько необходимо для размещения всего содержимого документа.

Содержимое документа

Содержимое XSL-FO-документа (то, что в нем не является разметкой) преимущественно является текстом. Содержимое, которое не является XML, например, GIF и JPEG, можно включать в документ почти также, как это делается в HTML с помощью элемента IMG. Другие формы XML-содержимого, например, MathML или SVG, можно включать прямо в XSL-FO-документ. Все содержимое размещается в нескольких видах элементов, включая:

- Блоковые форматирующие объекты
- Внутри-строчные форматирующие объекты
- Табличные форматирующие объекты
- Вне-строчные форматирующие объекты

Все виды этих элементов являются потомками либо элемента fo:flow либо элемента fo:static-content. Они никогда не размещаются прямо в элементах fo:page-sequence или fo:simple-page-master.

Блоковые форматирующие объекты

Блоковые форматирующие объекты представляются как прямоугольные зоны, разделенные переносами строки и, возможно, дополнительными символами форматирования в содержании, которое предшествует или следует за блоковым форматирующим объектом. Блоки могут содержать в себе другие блоки, в этом случае внутренний блок также отделяется от внешнего переносом строки и, возможно, дополнительными символами форматирования. Блоковыми форматирующими объектами являются:

- fo:block
- fo:block-container
- fo:table-and-caption
- fo:table
- fo:list-block

Элемент fo:block является XSL-FO-эквивалентом свойства display: block в CSS или элемента DIV в HTML. Блоки могут входить в элемент fo:flow, в другие элементы fo:block или в элементы fo:static-content. Каждый элемент fo:block может содержать другие элементы fo:block, другие блоковые элементы, например, fo:table и fo:list-block, а также внутри-строчные элементы, такие, как fo:inline и fo:page-number. Блоковые элементы могут также содержать простой текст, например:

```
<fo:block>The Periodic Table, Page <fo:page-number/></fo:block>
```

Блоковые элементы обычно имеют атрибуты, описывающие свойства зоны и текстового форматирования. Свойства текстового форматирования по умолчанию наследуются всеми дочер-

ними элементами блока.

Замечание

В версии FOP 0.18.1 отсутствует поддержка элементов fo:block-container и fo:table-and-caption.

Внутри-строчные форматирующие объекты

Внутри-строчные форматирующие объекты также представляются как прямоугольные зоны, которые могут содержать текст или другие внутри-строчные элементы. Однако, внутри-строчные зоны чаще всего составляются в строки, идущие слева направо. Когда строка заполняется, под ней начинается новая строка. Точный порядок, в котором размещаются внутри-строчные элементы, зависит от режима письма. Например, при работе с ивритом или арабским письмом, внутри-строчные элементы идут справа налево. К внутри-строчным форматирующим объектам относятся:

- fo:bidi-override
- fo:character
- fo:external-graphic
- fo:initial-property-set
- fo:instream-foreign-object
- fo:inline
- fo:inline-container
- fo:leader
- fo:page-number
- fo:page-number-citation

Замечание

Версия FOP 0.18.1 не поддерживает элементы fo:bidi-override, fo:initial-property-set или fo:inline-container.

Табличные форматирующие объекты

Табличные форматирующие объекты являются XSL-FO-эквивалентами табличных свойств в CSS2. Однако, в XSL-FO работа с таблицами ведется более естественным образом, нежели в CSS. В основном, отдельная таблица является блоковым объектом, в то время как части таблицы не являются ни внутри-строчными ни блоковыми элементами. Тем не менее, полная таблица может быть превращена во внутри-строчный объект с помощью размещения ее в элементе fo:inline-container.

Существует девять табличных форматирующих объектов XSL:

- fo:table-and-caption
- fo:table
- fo:table-caption
- fo:table-column

- fo:table-header
- fo:table-footer
- fo:table-body
- fo:table-row
- fo:table-cell

Корневым элементом таблицы является либо элемент fo:table либо элемент fo:table-and-caption, который содержит элементы fo:table и fo:caption. Элемент fo:table содержит элементы fo:table-header, fo:table-body и fo:table-footer. Тело таблицы (элемент fo:table-body) содержит элементы fo:table-row, которые разбиваются элементами fo:table-cell.

Замечание

FOP 0.18.1 реализует ограниченную поддержку табличных форматирующих объектов и совсем не поддерживает элементы fo:table-and-caption и fo:table-caption.

Вне-строчные форматирующие объекты

Существует три "вне-строчных" форматирующих объекта:

- fo:float
- fo:footnote
- fo:footnote-body

Вне-строчные форматирующие объекты "наплывают" на место имеющихся на странице внутри-строчных и блоковых объектов. На странице они не всегда располагаются между теми же элементами, что и в исходном XML-дереве форматирующих объектов.

Замечание

FOP 0.18.1 не поддерживает fo:float.

Пунктиры и линейки

Линейка - это блок, содержащий горизонтальную линию, которая вставляется в текст так же, как линия под названием на первой странице этой главы. В HTML элемент HR создает линейку. Пунктир - это линия в середине строки, идущая от правой стороны выровненного по левому краю текста до левой стороны какого-то выровненного по правому краю текста на той же строке. Обычно пунктир рисуется отдельными точками, хотя могут применяться и другие символы. Пунктиры чаще всего можно увидеть в меню и в таблицах содержания. Если вы откроете таблицу содержания в начале этой книги, вы увидите пунктир между названиями глав и разделов и номерами страниц.

В XSL-FO и линейки и пунктиры создаются элементом fo:leader. Это внутри-строчный элемент, который и представляет пунктир, хотя его легко можно применить в качестве линейки, если поместить его внутрь элемента fo:block.

Внешний вид линеек описывает шесть атрибутов:

- leader-alignment: этот атрибут может иметь значение reference-area или page, что указы-

вает: начало линейки должно быть выравнено с началом названного объекта. Также можно использовать значения none и inherit.

- leader-length: длина линейки, например, 12pc или 5in.
- leader-pattern: этот атрибут может иметь значения space, rule, dots, use-content или inherit. Значение use-content указывает, что символы, из которых составляется пунктир, должен быть считан из содержимого элемента fo:leader.
- leader-pattern-width: это свойство указывает длину, например, 2mm (или можно использовать значение use-font-metrics, которое указывает, что символы, из которых образована линейка, должны размещаться на естественном расстоянии друг от друга). Это свойство не задает длину всей линейки (это свойство указывается атрибутом leader-length), оно задает длину каждого повторяющегося кусочка линейки, когда она образована какими-то символами. При необходимости для достижения заданной длины между повторяющимися элементами добавляется пустое пространство.
- rule-style: это свойство может иметь те же значения. что и CSS-свойство border-style; то есть, none, dotted, dashed, solid, double, groove, ridge или inherit.
- rule-thickness: это свойство указывает толщину (ширину) линии, по умолчанию она равна 1pt.

Кроме того, к линейкам и пунктирам применимы некоторые другие общие свойства. Например, можно использовать свойство font-family для изменения шрифта, которым рисуется пунктир, или свойство color для изменения цвета линейки или пунктира. Например, в следующем примере рисуется зеленая горизонтальная линия длиной в 7.5 дюймов и толщиной в два пункта:

```
<fo:block>
  <fo:leader leader-length="7.5in" leader-pattern="rule"
             rule-thickness="2pt" color="green"/>
</fo:block>
```

В таблице стилей на листинге 18-6 используется элемент fo:leader для размещения линейки в верхней части футера каждой страницы.

Листинг 18-6:
Применение fo:leader для разделения футера и основного содержания страницы горизонтальной линией

```
<?xml version="1.0"?>
<xsl:stylesheet version="1.0"
  xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
  xmlns:fo="http://www.w3.org/1999/XSL/Format">
  <xsl:template match="/">
    <fo:root xmlns:fo="http://www.w3.org/1999/XSL/Format">
      <fo:layout-master-set>
        <fo:simple-page-master master-name="A4"
           page-width="297mm"  page-height="210mm"
           margin-top="0.5in"  margin-bottom="0.5in"
           margin-left="0.5in" margin-right="0.5in">
          <fo:region-before extent="1.0in"/>
          <fo:region-body margin-top="1.0in"
                          margin-bottom="1.0in"/>
          <fo:region-after  extent="1.0in"/>
        </fo:simple-page-master>
      </fo:layout-master-set>
      <fo:page-sequence master-reference="A4"
        initial-page-number="1" language="en" country="us">
        <fo:static-content flow-name="xsl-region-before">
          <fo:block>The Periodic Table</fo:block>
```

```
        </fo:static-content>
        <fo:static-content flow-name="xsl-region-after">
          <fo:block><fo:leader leader-pattern="rule"
                               leader-length="18cm" />
          </fo:block>
          <fo:block>p. <fo:page-number/></fo:block>
        </fo:static-content>
        <fo:flow flow-name="xsl-region-body">
          <xsl:apply-templates select="//ATOM"/>
        </fo:flow>
      </fo:page-sequence>
    </fo:root>
  </xsl:template>
  <xsl:template match="ATOM">
    <fo:block><xsl:value-of select="NAME"/></fo:block>
  </xsl:template>
</xsl:stylesheet>
```

Взгляните на результат - в нижней части каждой страницы появились линейки:

На скриншоте 18-6 - третья страница PDF-файла, сгенерированного при помощи таблицы стилей 18-6:

Графика

В XSL-FO имеется два способа встраивания изображений в отображаемый документ. Элемент fo:external-graphic вставляет не-XML-графику, например, JPEG. Элемент fo:instream-foreign-object вставляет XML-документ, который не является документом XSL-FO,

например, изображение SVG или выражение MathML.

fo:external-graphic

Элемент fo:external-graphic является эквивалентом HTML-элемента IMG. То есть, он загружает с указанного URL изображение, не обладающее форматом XML. Элемент fo:external-graphic - всегда пустой и не содержит дочерних элементов. Атрибут src содержит URI, указывающий на файл встраиваемого изображения. Например, рассмотрим обычный HTML-элемент IMG:

```
<IMG SRC="cup.gif">
```

Эквивалент fo:external-graphic выглядит следующим образом:

```
<fo:external-graphic src="cup.gif"/>
```

Конечно, если нужно, можно использовать и абсолютный URL:

```
<fo:external-graphic src="http://www.ibiblio.org/xml/cup.gif"/>
```

Аналогично ситуации с веб-браузерами и HTML, нет гарантий, что каждая форматирующая машина будет распознавать и поддерживать все наивозможные графические форматы. В настоящее время FOP поддерживает изображения GIF и JPEG. В будущем будут добавлены и другие форматы.

fo:external-graphic - внутри-строчный элемент. Вы можете сделать изображение блоком, для этого нужно заключить этот элемент в элемент fo:block, вот так:

```
<fo:block><fo:external-graphic src="cup.gif"/></fo:block>
```

На листинге 18-7 приводится таблица стилей, которая загружает изображение с адреса http://www.ibiblio.org/xml/images/atom.jpg и размещает его в заголовке каждой страницы. В данном случае URI изображения твердо закодировано в таблице стилей. Но в общем случае адрес изображения может считываться из исходного документа.

Листинг 18-7:
Таблица стилей XSL, указывающая на внешний графический файл

```
<?xml version="1.0"?>
<xsl:stylesheet version="1.0"
  xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
  xmlns:fo="http://www.w3.org/1999/XSL/Format">
  <xsl:template match="/">
    <fo:root xmlns:fo="http://www.w3.org/1999/XSL/Format">
      <fo:layout-master-set>
        <fo:simple-page-master master-name="A4"
           page-width="297mm"  page-height="210mm"
           margin-top="0.5in"  margin-bottom="0.5in"
           margin-left="0.5in" margin-right="0.5in">
          <fo:region-before extent="1.0in"/>
          <fo:region-body margin-top="1.0in"
                          margin-bottom="1.0in"/>
          <fo:region-after  extent="1.0in"/>
        </fo:simple-page-master>
      </fo:layout-master-set>
      <fo:page-sequence master-reference="A4"
        initial-page-number="1" language="en" country="us">
        <fo:static-content flow-name="xsl-region-before">
```

```
          <fo:block>
           <fo:block>
            The Periodic Table
           </fo:block>
            <fo:external-graphic
             src="fosamples/header.jpg"/>
          </fo:block>
        </fo:static-content>
        <fo:static-content flow-name="xsl-region-after">
          <fo:block>
            <fo:leader leader-pattern="rule"
                       leader-length="18cm"/>
          </fo:block>
          <fo:block>p. <fo:page-number/></fo:block>
        </fo:static-content>
        <fo:flow flow-name="xsl-region-body">
          <xsl:apply-templates select="//ATOM"/>
        </fo:flow>
      </fo:page-sequence>
    </fo:root>
  </xsl:template>
  <xsl:template match="ATOM">
    <fo:block><xsl:value-of select="NAME"/></fo:block>
  </xsl:template>
</xsl:stylesheet>
```

На скриншоте 18-7 - первая страница PDF-файла, сгенерированного с помощью таблицы стилей 18-7. Изображение размещено в верхней части каждой страницы документа.

fo:instream-foreign-object

Элемент fo:instream-foreign-object вставляет в документ графический элемент, описанный в виде XML и который встраивается непосредственно в XSL-FO-документ. Например, элемент fo:instream-foreign-object может содержать изображение SVG. Форматер выводит это изображение в результирующий документ. На листинге 18-8 приведена таблица стилей, которая вырабатывает XSL-FO-документ, в котором в заголовке каждой страницы размещается розовый треугольник (пример SVG из второй главы этой книги):

Листинг 18-8:
Таблица стилей XSL, которая выводит изображение SVG

```
<?xml version="1.0"?>
<xsl:stylesheet version="1.0"
  xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
  xmlns:fo="http://www.w3.org/1999/XSL/Format">
  <xsl:template match="/">
    <fo:root xmlns:fo="http://www.w3.org/1999/XSL/Format">
      <fo:layout-master-set>
        <fo:simple-page-master master-name="A4"
           page-width="297mm"  page-height="210mm"
           margin-top="0.5in"  margin-bottom="0.5in"
           margin-left="0.5in" margin-right="0.5in">
          <fo:region-before extent="1.0in"/>
          <fo:region-body   margin-top="1.0in"/>
        </fo:simple-page-master>
      </fo:layout-master-set>
      <fo:page-sequence master-reference="A4"
        initial-page-number="1" language="en" country="us">
        <fo:static-content flow-name="xsl-region-before">
          <fo:block> The Periodic Table
           <fo:block>
            <fo:instream-foreign-object>
              <svg xmlns="http://www.w3.org/2000/svg"
                 width="1.5cm" height="1cm">
       <polygon style="fill:#FFCCCC" points="0,31 18,0 36,31"/>
              </svg>
            </fo:instream-foreign-object>
          </fo:block>
                              </fo:block>
        </fo:static-content>
        <fo:flow flow-name="xsl-region-body">
          <xsl:apply-templates select="//ATOM"/>
        </fo:flow>
      </fo:page-sequence>
    </fo:root>
  </xsl:template>
  <xsl:template match="ATOM">
    <fo:block><xsl:value-of select="NAME"/></fo:block>
  </xsl:template>
</xsl:stylesheet>
```

На скриншоте 18-8 показана первая страница PDF-файла, сгенерированного при помощи таблицы стилей 18-8. Треугольник размещается в верхней части каждой страницы доку-

мента.

Не все форматеры поддерживают все наивозможные графические форматы XML. Например, FOP вообще не поддерживает MathML и в нем реализовано лишь подмножество SVG. Тем не менее, это полезная техника, особенно в тех случаях, когда нужно, чтобы XSLT генерировала графику прямо в процессе обработки документа. Например, можно написать таблицу стилей XSLT, которая генерирует красиво оформленные годовые отчеты, включая все графические элементы и диаграммы, преобразуя некоторые части исходного документа в XSL-FO, а другие - в SVG.

Графические свойства

Элементы fo:external-graphic и fo:instream-foreign-object обладают несколькими свойствами, предназначенными для масштабирования, позиционирования, обрезки и выравнивания изображения на странице.

Атрибут content-type

Атрибут content-type определяет тип графики. Вы можете задать его как медиа-тип MIME, например, image/jpg или image/svg-xml, вставляя перед записью типа префикс content-type:. Например, чтобы определить, что элемент fo:external-graphic относится к изображению GIF, следует записать:

```
<fo:external-graphic content-type="content-type:image/gif"
                     src="cup.gif" />
```

Значение этого атрибута может также пониматься как префикс при использовании формы namespace-prefix: префикс . Например, чтобы определить, что элемент fo:instream-foreign-object заключает в себе изображение SVG, следует записать:

```
<fo:instream-foreign-object
   xmlns:svg="http://www.w3.org/2000/svg"
   content-type="namespace-prefix:svg">
```

Префикс пространства имен не обязательно декларируется в элементе fo:instream-foreign-object. Можно поместить эту декларацию в одном из предков этого элемента.

Размер

Атрибуты height и width задают горизонтальный и вертикальный размер прямоугольной зоны страницы, в которой размещается изображение. Каждый из этих атрибутов в качестве своего значения может иметь ключевое слово auto - таким образом дается указание использовать собственный размер изображения.

Атрибуты content-height и content-width задают вертикальный и горизонтальный размер собственно изображения. Если один из этих размеров или они оба не совпадают с атрибутами height, width, изображение соответствующим образом масштабируется.

Масштабирование

Атрибут scaling может иметь значение uniform или non-uniform. Первое значение указывает, что при масштабировании изображения следует сохранить его пропорции - это значение используется по умолчанию. Значение non-uniform позволяет масштабировать длину и высоту изображения независимо, так что изображение искажается.

Кроме того, можно выбрать алгоритм, по которому происходит масштабирование, с помощью атрибута scaling-method. Он может иметь значение auto, integer-pixels или resample-any-method. Значение integer-pixels обеспечивает целые значения отношения размеров исходного и результирующего изображения, например, 2:1 или 3:1, но не 1.5:1. В большинстве случаев, отмасштабированные таким образом изображения меньше по объему, чем при значении атрибута resample-any-method, поскольку не требуют сглаживания цветовых переходов. Значение auto разрешает форматеру действовать по своему усмотрению.

Также здесь можно применять и несколько общих для внутри-строчных элементов свойств. Сюда относятся общие звуковые свойства, свойства фоновых изображений и границ, отступов и отбивок. Поскольку графика не должна разбиваться на несколько страниц, здесь нельзя применять обычные свойства переносов, но поддерживаются свойства keep-with-next и keep-with-previous.

Ссылки

Элемент fo:basic-link предназначен для создания ссылок в стиле гиперссылок HTML. Это внутри-строчный форматирующий объект, который можно кликнуть, чтобы перейти к дру-

гому документу или к другому месту текущего документа. Он не нужен в печатных документах, но пригодится, когда веб-браузеры будут прямо поддерживать XSL-FO. Поведение ссылки контролируется восемью атрибутами:

- external-destination
- internal-destination
- indicate-destination
- show-destination
- destination-placement-offset
- target-presentation-context
- target-processing-context
- target-stylesheet

Ссылка на удаленный документ-цель определяется URI в значении атрибута external-destination. Если пользователь активирует ссылку, браузер должен будет заменить текущий документ на документ с соответствующим URI. В большинстве графических интерфейсов пользователь активирует ссылку, кликая по ней мышью. Пример использования ссылки:

```
<fo:block> Be sure to visit the
   <fo:basic-link
     external-destination="http://www.ibiblio.org/xml/">
      Cafe con Leche Web site!
   </fo:basic-link>
</fo:block>
```

Кроме того, можно ссылаться на другой узел текущего документа, используя атрибут internal-destination. Значение этого атрибута - не URI, а идентификатор ID элемента, на который указывает ссылка. Тут можно использовать функцию XSLT generate-id(), которая генерирует и идентификатор элемента-цели и ссылку на этот элемент внутри одного выходного XSL-FO-документа. Не следует использовать в одной ссылке одновременно указание на внешнюю и внутреннюю цель.

Три атрибута цели влияют на внешний вид и поведение ссылки. Атрибут indicate-destination имеет булевское значение (true или false; по умолчанию false), которое определяет, должен ли объект, на который указывает ссылка, при загрузке как-то выделяться на фоне остальных частей того же самого документа. Например, если пойти по ссылке на один из элементов ATOM в таблице из ста химических элементов, атом, на который указывала ссылка, может быть выделен полужирным шрифтом. Детали зависят от конкретной системы.

Атрибут show-destination имеет два возможных значения: replace (по умолчанию) и new. При значении атрибута replace при активации ссылки документ-цель заменяет в окне исходный документ. При значении new, при активации ссылки браузер открывает новое окно, в котором отображается документ-цель.

Когда браузер следует по HTML-ссылке в середину документа, заданный в ссылке объект-цель оказывается на самом верху окна браузера. Атрибут destination-placement-offset определяет, насколько должен браузер прокрутить данный объект в окне браузера вниз. Атрибут задается как длина прокрутки, например, 3in или 156px.

Три свойства цели описывают особенности отображения документа на другом конце ссылки. Атрибут target-presentation-context содержит URI, который обычно показывает некоторое подмножество внешнего целевого документа, который должен быть показан пользователю. Например, здесь можно использовать XPointer, который укажет, что хотя и должна быть загружена целая книга, будет показана только ее седьмая глава.

Атрибут target-processing-context содержит URI, который служит в качестве базового URI, в

том случае, если внешний документ-цель содержит относительный URI. В ином случае он будет пониматься относительно текущего документа.

И наконец, атрибут target-stylesheet содержит URI, который указывает на таблицу стилей, которая должна быть использована для отображения документа-цели. Эта таблица имеет приоритет над любой таблицей стилей, заданной в оригинальном документе-цели - будь то в процессуальной инструкции xml-stylesheet, в HTML-элементе LINK или в HTTP-заголовке.

Кроме того для ссылок могут использоваться обычные свойства-атрибуты, описывающие отступы и отбивки, фоновые изображения и границы, звуковые характеристики.

Списки

Форматирующий объект fo:list-block создает список как блоковый элемент. (внутри-строчных списков не существует.) Список может быть оформлен буллетами, цифрами, отступами или другим форматированием. Каждый элемент fo:list-block содержит либо серии элементов fo:list-item либо пары fo:list-item-label - fo:list-item-body. (нельзя задействовать сразу две альтернативы.) Элемент fo:list-item должен содержать элементы fo:list-item-label и fo:list-item-body. Элемент fo:list-item-label содержит буллет, число или другую отметку для пункта списка как блокового элемента. Элемент fo:list-item-body содержит блоковые элементы, несущие настоящее содержимое пункта списка. Говоря вообще, fo:list-block содержит элементы fo:list-item. Каждый элемент fo:list-item содержит элементы fo:list-item-label и fo:list-item-body. Но элементы fo:list-item могут быть опущены, например:

```
<fo:list-block>
  <fo:list-item>
     <fo:list-item-label><fo:block>*</fo:block>
     </fo:list-item-label>
    <fo:list-item-body>
      <fo:block>Actinium</fo:block>
    </fo:list-item-body>
  </fo:list-item>
  <fo:list-item>
     <fo:list-item-label><fo:block>*</fo:block>
     </fo:list-item-label>
    <fo:list-item-body>
      <fo:block>Aluminum</fo:block>
    </fo:list-item-body>
  </fo:list-item>
</fo:list-block>
```

Или с опущенными тэгами fo:list-item:

```
<fo:list-block>
   <fo:list-item-label>
     <fo:block>*</fo:block>
   </fo:list-item-label>
   <fo:list-item-body>
     <fo:block>Actinium</fo:block>
   </fo:list-item-body>
   <fo:list-item-label>
     <fo:block>*</fo:block>
   </fo:list-item-label>
   <fo:list-item-body>
```

```
      <fo:block>Aluminum</fo:block>
    </fo:list-item-body>
  </fo:list-block>
```

Элемент fo:list-block имеет два специальных атрибута, контролирующих форматирование списка:

- provisional-label-separation: расстояние между отметкой пункта списка и телом пункта списка, задается в виде триплета максимум;минимум;оптимум, например, 2mm;0.5mm;1mm.
- provisional-distance-between-starts: расстояние между начальным краем отметки пункта списка и начальным краем тела пункта списка.

Элемент fo:list-block также имеет обычные атрибуты, отвечающие за отступы и отбивки, фоновые изображения и границы, переносы и разбивки строк, звуковые характеристики. Элемент fo:list-item имеет стандартные блоковые атрибуты, описывающие фоновые изображения, положение, звуковые характеристики, границы, отступы и отбивки, разбивку строк и страниц. Элементы fo:list-item-label и fo:list-item-body могут иметь только атрибуты, отвечающие за свойства, обеспечивающие доступность документа для людей с нарушениями органов восприятия: id и keep-together. Все остальное форматирование этих элементов определяется либо их родительскими элементами (fo:list-item и fo:list-item-block) или дочерними элементами, которые они содержат.

Таблица стилей на листинге 18-9 выводит периодическую таблицу химических элементов в виде списка, в котором атомные номера выступают отметками пунктов списка, а имена элементов - телами пунктов списка.

Листинг 18-9:
Таблица стилей XSL, которая форматирует таблицу химических элементов в виде списка

```
<?xml version="1.0"?>
<xsl:stylesheet version="1.0"
  xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
  xmlns:fo="http://www.w3.org/1999/XSL/Format">
  <xsl:template match="/">
    <fo:root xmlns:fo="http://www.w3.org/1999/XSL/Format">
      <fo:layout-master-set>
        <fo:simple-page-master master-name="A4"
           page-width="297mm"  page-height="210mm"
           margin-top="0.5in"  margin-bottom="0.5in"
           margin-left="0.5in" margin-right="0.5in">
          <fo:region-body/>
        </fo:simple-page-master>
      </fo:layout-master-set>
      <fo:page-sequence master-reference="A4">
        <fo:flow flow-name="xsl-region-body">
          <fo:list-block>
            <xsl:apply-templates select="//ATOM">
              <xsl:sort data-type="number"
                        select="ATOMIC_NUMBER"/>
            </xsl:apply-templates>
          </fo:list-block>
        </fo:flow>
      </fo:page-sequence>
    </fo:root>
  </xsl:template>
  <xsl:template match="ATOM">
```

```
      <fo:list-item>
        <fo:list-item-label>
         <fo:block>
           <xsl:value-of select="ATOMIC_NUMBER"/>
        </fo:block>
       </fo:list-item-label>
        <fo:list-item-body>
         <fo:block start-indent="body-start()">
          <xsl:value-of select="NAME"/>
        </fo:block>
        </fo:list-item-body>
      </fo:list-item>
    </xsl:template>
  </xsl:stylesheet>
```

На скриншоте 18-9 - вторая страница PDF-файла, сгенерированного с помощью таблицы стилей 18-9.

Скриншот 18-9: Периодическая таблица, отформатированная как список

В HTML пункт списка выводится по умолчанию с определенным отступом. Но, как вы могли заметить, в XSL-FO такой отступ по умолчанию отсутствует. Если отступ нужен, можно использовать атрибуты start-indent и end-indent, разместив их в элементах fo:list-item-label и fo:list-item-body. Каждый из этих атрибутов задает длину. Но поскольку тело пункта списка обычно начинается на той же самой строке, как и отметка пункта списка, его начальный отступ часто задается специальной функцией XSL-FO body-start(). Она возвращает комбинированную длину start-indent и provisional-distance-between-starts. Пример:

```
<xsl:template match="ATOM">
  <fo:list-item>
    <fo:list-item-label start-indent="1.0cm"
```

```
                        end-indent="1.0cm">
        <fo:block>
          <xsl:value-of select="ATOMIC_NUMBER"/>
         </fo:block>
      </fo:list-item-label>
      <fo:list-item-body start-indent="body-start()">
        <fo:block>
          <xsl:value-of select="NAME"/>
        </fo:block>
      </fo:list-item-body>
    </fo:list-item>
  </xsl:template>
```

Таблицы

Фундаментальным элементом таблиц в XSL является fo:table-and-caption. Это объект блокового уровня, который содержит элементы fo:table и fo:caption. Если вашей таблице не нужно описание (caption), можно использовать только элемент fo:table. Модель таблицы в XSL-FO довольно близка к модели таблицы HTML. В таблице 18-1 показано соответствие между таблицами HTML 4.0 и таблицами в форматирующих объектах XSL:

Таблица 18-1: Таблицы HTML и таблицы форматирующих объектов XSL

HTML-элемент	XSL-FO-элемент
TABLE	fo:table-and-caption
эквивалента нет	fo:table
CAPTION	fo:table-caption
COL	fo:table-column
COLGROUP	no equivalent
THEAD	fo:table-header
TBODY	fo:table-body
TFOOT	fo:table-footer
TD	fo:table-cell
TR	fo:table-row

Каждый элемент fo:table-and-caption может содержать не обязательный элемент fo:table-caption и один элемент fo:table. Элемент fo:table-caption может содержать любые блоковые элементы, которые вы захотите разместить в описании таблицы. По умолчанию, описания размещаются перед таблицей, но вы может разместить ее в любом месте, используя атрибут caption-side элемента table-and-caption. Возможно одно из восьми значений:

- before
- after
- start
- end

- top
- bottom
- left
- right

Например, в следующей таблице описание размещается под таблицей:

```
<fo:table-and-caption caption-side="bottom">
  <fo:table-caption>
    <fo:block font-weight="bold"
              font-family="Helvetica, Arial, sans"
              font-size="12pt">
    Таблица 18-1: Таблицы HTML и таблицы форматирующих объектов XSL
    </fo:block>
  </fo:table-caption>
  <fo:table>
    <!-- table contents go here -->
  </fo:table>
</fo:table-and-caption>
```

Элемент fo:table содержит элементы fo:table-column и необязательные элементы fo:table-header и fo:table-footer, а также один или несколько элементов fo:table-body. Элемент fo:table-body делится на элементы fo:table-row. Каждый элемент fo:table-row разбит на элементы fo:table-cell. Элементы fo:table-header и fo:table-footer могут составляться либо из элементов fo:table-cell либо fo:table-row. Пример: описание простой таблицы, в которую включены три ряда из приведенной выше таблицы 18-1:

```
<fo:table>
  <fo:table-header>
    <fo:table-cell>
      <fo:block font-family="Helvetica, Arial, sans"
                font-size="11pt" font-weight="bold">
        HTML Element
      </fo:block>
    </fo:table-cell>
    <fo:table-cell>
      <fo:block font-family="Helvetica, Arial, sans"
                font-size="11pt" font-weight="bold">
        XSL FO Element
      </fo:block>
    </fo:table-cell>
  </fo:table-header>
  <fo:table-body>
    <fo:table-row>
      <fo:table-cell>
        <fo:block font-family="Courier, monospace">
          TABLE
        </fo:block>
      </fo:table-cell>
      <fo:table-cell>
        <fo:block font-family="Courier, monospace">
          fo:table-and-caption
        </fo:block>
      </fo:table-cell>
    </fo:table-row>
    <fo:table-row>
```

```
        <fo:table-cell>
          <fo:block>no equivalent</fo:block>
        </fo:table-cell>
        <fo:table-cell>
          <fo:block font-family="Courier, monospace">
            fo:table
          </fo:block>
        </fo:table-cell>
      </fo:table-row>
    </fo:table-body>
  </fo:table>
```

Можно указать, что ячейки таблицы должны перекрывать несколько рядов или колонок, установив атрибуты number-columns-spanned и/или number-rows-spanned в соответствующее целое числовое значение, соответствующее количеству перекрываемых данной ячейкой рядов или колонок. Необязательный атрибут column-number может изменить номер колонки, с которой начинается перекрытие. По умолчанию, это текущая колонка.

Границы вокруг различных частей таблицы можно рисовать, используя обычные атрибуты свойств границ. Атрибут empty-cells может принимать значение show или hide; show - если границы должны рисоваться и вокруг ячеек без содержимого, hide - если нет. По умолчанию используется значение show.

Когда длинная таблица разбивается на несколько страниц, иногда необходимо, чтобы заголовок и футер таблицы повторялись на каждой странице. Это поведение можно настроить, используя атрибуты table-omit-header-at-break и table-omit-footer-at-break элемента fo:table. Значение этих атрибутов false указывает, что заголовки и футеры должны повторяться на всех страницах. Значение true указывает, что они повторяться не должны. По умолчанию используется значение false.

Необязательный элемент fo:table-column - это пустой элемент, который задает свойства всех ячеек определенной колонки. Ячейки, к которым он относится, определяются его атрибутом column-number или по положению самого элемента fo:table-column. Элемент fo:table-column в действительности не содержит ячеек. Этот элемент может применить определенные свойства к нескольким последовательным колонкам используя атрибут number-columns-spanned, значение которого указывает на количество перекрываемых колонок (его значение должно быть больше единицы). Чаще всего с помощью элемента fo:table-column задается свойство column-width (ширина колонки), но можно также задавать и границы, отбивки ячеек и свойства фона (они будут обсуждаться ниже - эти свойства похожи на аналогичные свойства в CSS).

Замечание

FOP 0.18.1 реализует ограниченную поддержку работы с таблицами. В частности, не поддерживаются элементы fo:table-caption и fo:table-and-caption. Кроме того, FOP требует, чтобы вы конкретно задавали ширину колонок с помощью элемента fo:table-column. Нельзя оставлять процессору выбрать подходящую ширину, как это делают браузеры.

Рассмотрим пример. Таблица стилей на листинге 18-10 генерирует документ, в котором свойства химических элементов форматируются в таблицу.

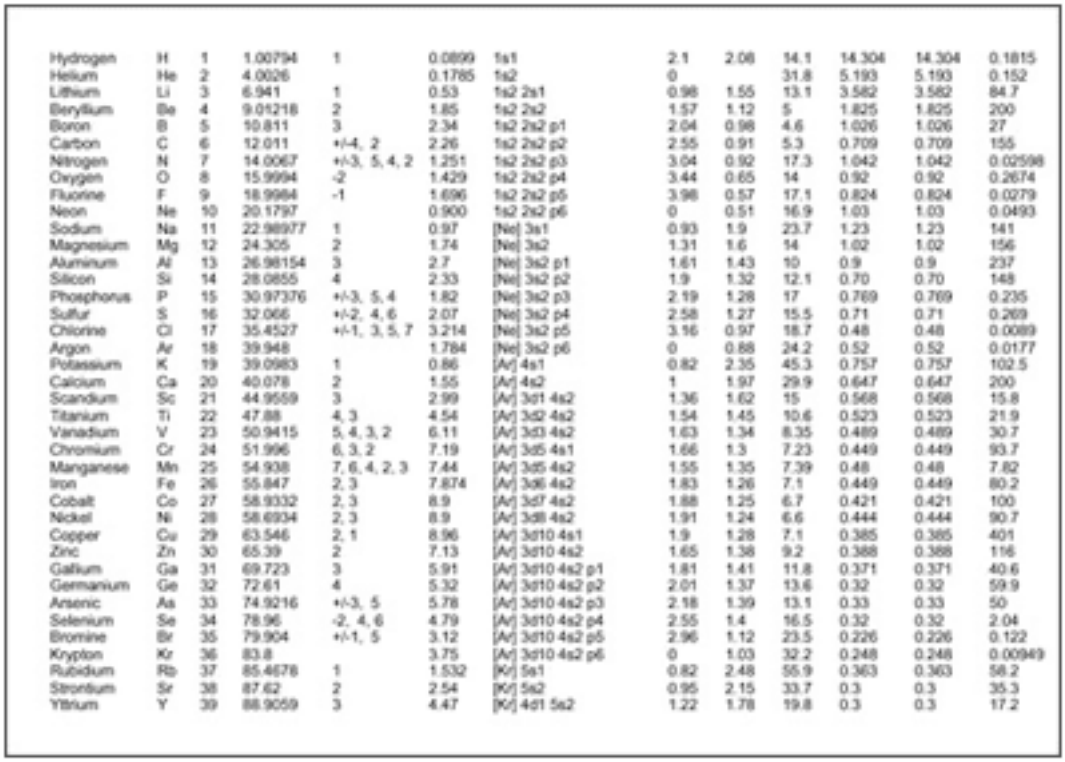
Листинг 18-10:
Таблица стилей XSL, которая форматирует данные о химических элементах в таблицу

```
<?xml version="1.0"?>
<xsl:stylesheet version="1.0"
```

```
xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
xmlns:fo="http://www.w3.org/1999/XSL/Format">
<xsl:template match="/">
  <fo:root xmlns:fo="http://www.w3.org/1999/XSL/Format">
    <fo:layout-master-set>
      <fo:simple-page-master master-name="A4"
         page-width="297mm"  page-height="210mm"
         margin-top="0.5in"  margin-bottom="0.5in"
         margin-left="0.5in" margin-right="0.5in">
        <fo:region-body/>
      </fo:simple-page-master>
    </fo:layout-master-set>
    <fo:page-sequence master-reference="A4">
      <fo:flow flow-name="xsl-region-body">
        <fo:table>
          <fo:table-column column-width="30mm"/>
          <fo:table-column column-width="12mm"/>
          <fo:table-column column-width="12mm"/>
          <fo:table-column column-width="25mm"/>
          <fo:table-column column-width="27mm"/>
          <fo:table-column column-width="18mm"/>
          <fo:table-column column-width="49mm"/>
          <fo:table-column column-width="16mm"/>
          <fo:table-column column-width="16mm"/>
          <fo:table-column column-width="16mm"/>
          <fo:table-column column-width="21mm"/>
          <fo:table-column column-width="21mm"/>
          <fo:table-column column-width="21mm"/>
          <fo:table-body>
            <xsl:apply-templates select="//ATOM">
              <xsl:sort data-type="number"
                select="ATOMIC_NUMBER"/>
            </xsl:apply-templates>
          </fo:table-body>
        </fo:table>
      </fo:flow>
    </fo:page-sequence>
  </fo:root>
</xsl:template>
<xsl:template match="ATOM">
  <fo:table-row>
    <fo:table-cell>
      <fo:block><xsl:value-of select="NAME"/></fo:block>
    </fo:table-cell>
    <fo:table-cell>
      <fo:block><xsl:value-of select="SYMBOL"/></fo:block>
    </fo:table-cell>
    <fo:table-cell>
      <fo:block>
        <xsl:value-of select="ATOMIC_NUMBER"/>
      </fo:block>
    </fo:table-cell>
    <fo:table-cell>
      <fo:block>
        <xsl:value-of select="ATOMIC_WEIGHT"/>
```

```
        </fo:block>
      </fo:table-cell>
      <fo:table-cell>
        <fo:block>
          <xsl:value-of select="OXIDATION_STATES"/>
        </fo:block>
      </fo:table-cell>
      <fo:table-cell>
        <fo:block><xsl:value-of select="DENSITY"/></fo:block>
      </fo:table-cell>
      <fo:table-cell>
        <fo:block>
          <xsl:value-of select="ELECTRON_CONFIGURATION"/>
        </fo:block>
      </fo:table-cell>
      <fo:table-cell>
        <fo:block>
          <xsl:value-of select="ELECTRONEGATIVITY"/>
        </fo:block>
      </fo:table-cell>
      <fo:table-cell>
        <fo:block>
          <xsl:value-of select="ATOMIC_RADIUS"/>
        </fo:block>
      </fo:table-cell>
      <fo:table-cell>
        <fo:block>
          <xsl:value-of select="ATOMIC_VOLUME"/>
        </fo:block>
      </fo:table-cell>
      <fo:table-cell>
        <fo:block>
          <xsl:value-of select="SPECIFIC_HEAT_CAPACITY"/>
        </fo:block>
      </fo:table-cell>
      <fo:table-cell>
        <fo:block>
          <xsl:value-of select="SPECIFIC_HEAT_CAPACITY"/>
        </fo:block>
      </fo:table-cell>
      <fo:table-cell>
        <fo:block>
          <xsl:value-of select="THERMAL_CONDUCTIVITY"/>
        </fo:block>
      </fo:table-cell>
    </fo:table-row>
  </xsl:template>
</xsl:stylesheet>
```

На скриншоте 18-10 - первая страница PDF-файла, сгенерированного с помощью таблицы стилей 18-10.



Hydrogen	H	1	1.00794	1	0.0899	1s1	2.1	2.08	14.1	14.304	14.304	0.1815
Helium	He	2	4.0026		0.1785	1s2	0		31.8	5.193	5.193	0.152
Lithium	Li	3	6.941	1	0.53	1s2 2s1	0.98	1.55	13.1	3.582	3.582	84.7
Beryllium	Be	4	9.01218	2	1.85	1s2 2s2	1.57	1.12	5	1.825	1.825	200
Boron	B	5	10.811	3	2.34	1s2 2s2 p1	2.04	0.98	4.6	1.026	1.026	27
Carbon	C	6	12.011	+/-4, 2	2.26	1s2 2s2 p2	2.55	0.91	5.3	0.709	0.709	155
Nitrogen	N	7	14.0067	+/-3, 5, 4, 2	1.251	1s2 2s2 p3	3.04	0.92	17.3	1.042	1.042	0.02598
Oxygen	O	8	15.9994	-2	1.429	1s2 2s2 p4	3.44	0.65	14	0.92	0.92	0.2674
Fluorine	F	9	18.9984	-1	1.696	1s2 2s2 p5	3.98	0.57	17.1	0.824	0.824	0.0279
Neon	Ne	10	20.1797		0.900	1s2 2s2 p6	0	0.51	16.9	1.03	1.03	0.0493
Sodium	Na	11	22.98977	1	0.97	[Ne] 3s1	0.93	1.9	23.7	1.23	1.23	141
Magnesium	Mg	12	24.305	2	1.74	[Ne] 3s2	1.31	1.6	14	1.02	1.02	156
Aluminum	Al	13	26.98154	3	2.7	[Ne] 3s2 p1	1.61	1.43	10	0.9	0.9	237
Silicon	Si	14	28.0855	4	2.33	[Ne] 3s2 p2	1.9	1.32	12.1	0.70	0.70	148
Phosphorus	P	15	30.97376	+/-3, 5, 4	1.82	[Ne] 3s2 p3	2.19	1.28	17	0.769	0.769	0.235
Sulfur	S	16	32.066	+/-2, 4, 6	2.07	[Ne] 3s2 p4	2.58	1.27	15.5	0.71	0.71	0.269
Chlorine	Cl	17	35.4527	+/-1, 3, 5, 7	3.214	[Ne] 3s2 p5	3.16	0.97	18.7	0.48	0.48	0.0089
Argon	Ar	18	39.948		1.784	[Ne] 3s2 p6	0	0.88	24.2	0.52	0.52	0.0177
Potassium	K	19	39.0983	1	0.86	[Ar] 4s1	0.82	2.35	45.3	0.757	0.757	102.5
Calcium	Ca	20	40.078	2	1.55	[Ar] 4s2	1	1.97	29.9	0.647	0.647	200
Scandium	Sc	21	44.9559	3	2.99	[Ar] 3d1 4s2	1.36	1.62	15	0.568	0.568	15.8
Titanium	Ti	22	47.88	4, 3	4.54	[Ar] 3d2 4s2	1.54	1.45	10.6	0.523	0.523	21.9
Vanadium	V	23	50.9415	5, 4, 3, 2	6.11	[Ar] 3d3 4s2	1.63	1.34	8.35	0.489	0.489	30.7
Chromium	Cr	24	51.996	6, 3, 2	7.19	[Ar] 3d5 4s1	1.66	1.3	7.23	0.449	0.449	93.7
Manganese	Mn	25	54.938	7, 6, 4, 2, 3	7.44	[Ar] 3d5 4s2	1.55	1.35	7.39	0.48	0.48	7.82
Iron	Fe	26	55.847	2, 3	7.874	[Ar] 3d6 4s2	1.83	1.26	7.1	0.449	0.449	80.2
Cobalt	Co	27	58.9332	2, 3	8.9	[Ar] 3d7 4s2	1.88	1.25	6.7	0.421	0.421	100
Nickel	Ni	28	58.6934	2, 3	8.9	[Ar] 3d8 4s2	1.91	1.24	6.6	0.444	0.444	90.7
Copper	Cu	29	63.546	2, 1	8.96	[Ar] 3d10 4s1	1.9	1.28	7.1	0.385	0.385	401
Zinc	Zn	30	65.39	2	7.13	[Ar] 3d10 4s2	1.65	1.38	9.2	0.388	0.388	116
Gallium	Ga	31	69.723	3	5.91	[Ar] 3d10 4s2 p1	1.81	1.41	11.8	0.371	0.371	40.6
Germanium	Ge	32	72.61	4	5.32	[Ar] 3d10 4s2 p2	2.01	1.37	13.6	0.32	0.32	59.9
Arsenic	As	33	74.9216	+/-3, 5	5.78	[Ar] 3d10 4s2 p3	2.18	1.39	13.1	0.33	0.33	50
Selenium	Se	34	78.96	-2, 4, 6	4.79	[Ar] 3d10 4s2 p4	2.55	1.4	16.5	0.32	0.32	2.04
Bromine	Br	35	79.904	+/-1, 5	3.12	[Ar] 3d10 4s2 p5	2.96	1.12	23.5	0.226	0.226	0.122
Krypton	Kr	36	83.8		3.75	[Ar] 3d10 4s2 p6	0	1.03	32.2	0.248	0.248	0.00949
Rubidium	Rb	37	85.4678	1	1.532	[Kr] 5s1	0.82	2.48	55.9	0.363	0.363	58.2
Strontium	Sr	38	87.62	2	2.54	[Kr] 5s2	0.95	2.15	33.7	0.3	0.3	35.3
Yttrium	Y	39	88.9059	3	4.47	[Kr] 4d1 5s2	1.22	1.78	19.8	0.3	0.3	17.2

Скриншот 18-10: Периодическая таблица, отформатированная как таблица

Внутри-строчные элементы

Элемент fo:inline не имеет особого влияния на общий макет страницы. Это просто элемент, на который можно "навесить" такие форматирующие атрибуты, как font-style или color и эти свойства применятся к внутри-строчному содержимому этого элемента. Форматирующий объект fo:inline - это контейнер, который группирует внутри-строчные объекты. Он не может содержать блоковые элементы. Например, можно применять элементы fo:inline для определения стиля различных частей футера, например:

```
<fo:static-content flow-name="xsl-region-after">
  <fo:block font-weight="bold" font-size="10pt"
            font-family="Arial, Helvetica, sans">
     <fo:inline font-style="italic" text-align="start">
       The XML Bible
     </fo:inline>
     <fo:inline text-align="centered">
       Page <fo:page-number/>
     </fo:inline>
     <fo:inline text-align="right">
       Глава 18: Форматирующие объекты XSL
     </fo:inline>
  </fo:block>
</fo:static-content>
```

Сноски

Элемент fo:footnote создает сноску. Автор помещает элемент fo:footnote в поток текста именно в том месте, в котором размещается ссылка на сноску, например, где размещается 1 или *. Элемент fo:footnote содержит и текст ссылки и блоковый элемент fo:footnote-body, содержащий, собственно, текст сноски. Но при этом только ссылка на сноску вставляется прямо в строку. Форматтер помещает текст сноски в после-регионе страницы (обычно, в футере).

Например, следующая сноска использует в качестве маркера звездочку и указывает на "JavaBeans, автора Elliotte Rusty Harold (IDG Books, Foster City, 1998), p. 147". Для нужного форматирования маркера и текста используются стандартные свойства font-size и vertical-align.

```
<fo:footnote>
  <fo:inline font-size="smaller" vertical-align="super">*
  </fo:inline>
  <fo:footnote-body font-size="smaller">
    <fo:inline font-size="smaller" vertical-align="super">
       *
    </fo:inline>
    <fo:inline font-style="italic">JavaBeans</fo:inline>,
      Elliotte Rusty Harold
      (IDG Books, Foster City, 1998), p. 147
  </fo:footnote-body>
</fo:footnote>
```

На заметку

XSL-FO не предоставляет каких-то средств автоматической нумерации и цитирования сносок, но это можно реализовать правильным использованием элемента xsl:number в таблице стилей преобразования. XSL-преобразования позволяют легко реализовать также и примечания.

Врезки

Элемент fo:float генерирует плавающую врезку, привязанную к верху области, в которой она расположена. Врезки fo:float обычно применяются для размещения графики, диаграмм, таблиц и другого вне-строчного содержимого, которое необходимо разместить где-то на странице, но при этом точное расположение не имеет значения. Например, в следующем примере элемент fo:block содержит плавающую врезку с графическим элементом и пояснительным текстом:

```
<fo:block>
  Хотя PDF-файлы являются ASCII-текстом,
  это книга не о PostScript, поэтому нет смысла
  заниматься здесь обсуждением точного синтаксиса
  данной команды. Если вам интересно, откройте,
  PDF-файл в любом текстовом редакторе.
  На скриншоте 18-1
```

```
    <fo:float float="before">
      <fo:external-graphic src="4760-7fg1801.jpg"
                           height="485px" width="623px" />
      <fo:block font-family="Helvetica, sans">
        <fo:inline font-weight="bold">
          Скриншот 18-1:
        </fo:inline>
        PDF-файл, открытый в Netscape Navigator
      </fo:block>
    </fo:float>
    показан результирующий файл, открытый в
    Netscape Navigator при помощи плагина Acrobat.
  </fo:block>
```

Форматер старается разместить врезку с изображением примерно там же, где размещается содержание, окружающее элемент fo:float. Но он не всегда может найти место на той же самой странице. Поэтому он может переместить объект на следующую страницу. С учетом этих ограничений форматер может разместить объект в любом месте страницы.

Значение атрибута float показывает, на какой стороне страницы размещается элемент fo:float. Он может принимать значения before, start, end, left, right, none и inherit.

Атрибут clear может быть установлен для элемента около объекта-врезки, чтобы показать, что он должен обтекать врезку или размещаться под нею. Атрибут может иметь значение start (начальный край объекта не должен быть привязан к объекту-врезке), end (задний край объекта не должен быть привязан к объекту-врезке), left (левый край объекта не должен быть привязан к объекту-врезке), right (правый край объекта не должен быть привязан к объекту-врезке), both (ни левый ни правый край объекта не должен быть привязан к объекту-врезке), none или inherit.

Замечание

FOP 0.18.1 не поддерживает форматирующий объект fo:float.

Форматирующие свойства

Сами по себе форматирующие объекты содержат мало информации о том, как должно форматироваться содержание документа. Они просто размещают содержимое в абстрактных прямоугольных зонах, которые, в свою очередь, размещаются в определенных частях страницы. Стилевые характеристики содержимого этих зон определяют атрибуты различных форматирующих объектов.

Как уже упоминалось, существует более 200 различных форматирующих свойств. Не все из них универсально применимы ко всем элементам. Например, нет смысла в придании свойства font-styleэлементу fo:external-graphic. Тем не менее, основная масса свойств может применяться к нескольким видам форматирующих объектов. (Те, которые применяются только для строго определенных видов объектов, например, src и provisional-label-separation, обсуждались в разделах, посвященных соответствующим форматирующим объектам.) Когда свойство является общим для многих форматирующих объектов, во всех случаях действует одинаковый синтаксис и один смысл. Например, для форматирования элемента fo:title в полужирный Times размером 14 пунктов используется тот же самый код, что и для форматирования в такой кегль элемента fo:block.

Многие свойства XSL-FO похожи на свойства CSS. Значения свойства CSS font-family те же самые, что и возможные значения атрибута XSL-FO font-family. Если вы не читали о CSS в главах с 14 по 16, вы уже более половины из них узнаете, изучив свойства XSL-FO.

Свойство id

Свойство id может применяться к любому элементу. Это XML-атрибут типа ID. Так что значением этого атрибута должно являться XML-имя, которое уникально в данной таблице стилей и внутри результирующего XSL-FO документа. Последнее требование несколько нетривиально, поскольку вполне возможно, чтобы одно правило-шаблон таблицы стилей сгенерировало несколько сот элементов выходного документа. И здесь может быть полезна XSLT-функция generate-id().

Свойство language

Свойство language задает язык содержания либо элементов fo:block, либо элементов fo:character. Обычно значением этого свойства является код языка по стандарту ISO 639, например, en (английский) или la (латинский). Значением также может быть ключевое слово none или use-document. Последнее означает применение языка исходного документа, заданного атрибутом xml:lang. Например, взгляните на первый стих Галльских войн Цезаря:

```
<fo:block id="verse1.1.1" language="la">
  Gallia est omnis divisa in partes tres,
  quarum unam incolunt Belgae, aliam Aquitani,
  tertiam qui ipsorum lingua Celtae, nostra Galli appellantur
</fo:block>
```

Хотя свойство language не имеет прямого действия на форматирование, оно может повлиять на него косвенно, если форматер выбирает различные алгоритмы построения страницы в зависимости от языка. Например, форматер должен использовать различные режимы письма для арабского и английского языка. Различные режимы по-разному размещают начальную и конечную области, а также используют различные направления наполнения внутри-строчных и блоковых элементов.

Свойства абзаца

Свойства абзаца - это стили, на которые нужно смотреть как на свойства, которые применяются к целому блоку текста в обычных текстовых процессорах, хотя возможно, тут более подходит название текстовые свойства блокового уровня. Например, абзацный отступ (indent) - это свойство абзаца, поскольку его можно задать для абзаца, но не для отдельного слова.

Свойства разрывов страниц

Свойства разрывов страниц определяют, где страница может разрываться, а где - нет. Существует семь не очень связанных между собой свойств разрывов страниц:

- keep-with-next
- keep-with-previous
- keep-together
- break-before
- break-after

Свойство keep-with-next определяет меру усилий, которые затратит форматер для того, чтобы сохранить данный форматирующий объект на той же самой странице, что и следующий форматирующий объект. Свойство keep-with-previous определяет меру усилий, которые затратит форматер для того, чтобы оставить данный форматирующий объект на той же странице, что и предыдущий. Свойство keep-together определяет меру усилий, которые затратит форматер для того, чтобы сохранить содержание данного форматирующего объекта на одной странице. Эти правила, тем не менее, не будут являться абсолютными, поскольку не всегда форматирующие объекты могут уместиться на одной странице. Значением каждого из этих свойств является целое число, определяющее степень усилий, которые затратит форматер для выполнения условия (чем больше число, тем больше усилий) или значением могут являться ключевые слова always или auto. Значение always задает максимальное усилие; auto разрешает разрывам появляться там, где нужно.

В отличие от этих свойств, свойства break-before и break-after управляют видами разрыва страниц. Что именно разрывается - определяется значением свойства. Допустимо одно из пяти значений:

- column: разрывается текущая колонка и происходит перенос в следующую.
- page: разрывается текущая страница и происходит перенос на следующую.
- even-page: разрывается текущая страница и происходит перенос на следующую четную страницу. Если текущая страница сама является четной, вставляется пустая страница.
- odd-page: разрывается текущая страница и происходит перенос на следующую нечетную страницу. Если текущая страница сама является нечетной, вставляется пустая страница.
- auto: форматеру разрешается самому решать, где делать разрыв. Значение по умолчанию.

Например, в следующем правиле-шаблоне задано, что каждый химический элемент, даже самого маленького текстового объема, будет выводиться на отдельной странице:

```
<xsl:template match="ATOM">
  <fo:block break-before="page" break-after="page">
    <xsl:apply-templates/>
  </fo:block>
</xsl:template>
```

И наконец, свойство inhibit-line-breaks может принимать булевские значение: true указывает, что не допускаются ни разбивки строк, ни разбивки страниц.

В XSL-FO имеется также три дополнительных свойства разрывов страниц: page-break-after, page-break-before и page-break-inside. Они не являются абсолютно необходимыми, поскольку их эффект может быть достигнут нужной комбинацией вышеописанных свойств, задающих и предотвращающих разрывы. Например, чтобы задать разрыв страницы после элемента, нужно установить значение атрибута break-before в page, а значение атрибута keep-with-previous - в auto.

Свойства переносов

Свойства переносов определяют, где разрешены переносы и как они должны использо-

ваться. Эти свойства применяются только к коротким или "мягким" дефисам, которые, например, используются для разбивки длинных слов в конце строки. Они не действуют на твердые дефисы, как, например, в слове mother-in-law, хотя наличие твердых дефисов может влиять на появление мягких. Имеется шесть свойств переносов, вот они:

- hyphenate: автоматические переносы разрешаются, только если значение этого атрибута - true.
- hyphenation-character: определяется символ Unicode, который используется для обозначения переносов слов, например, – в английском языке.
- hyphenation-keep: значением атрибута может быть одно из четырех ключевых слов (column, none, page, inherit), которые определяют, допускаются ли переносы и где именно. По умолчанию переносы не разрешаются.
- hyphenation-ladder-count: значением является неотрицательное целое число, которое определяет максимальное число перенесенных строк, которые могут идти подряд.
- hyphenation-push-character-count: значением является неотрицательное целое число, которое определяет минимальное количество знаков, которое должно идти за автоматически вставленным символом переноса. (короткие слоги плохо выглядят изолированно.)
- hyphenation-remain-character-count: значением является неотрицательное целое число, которое определяет минимальное количество символов, которое должно предшествовать автоматическому переносу.

Пример:

```
<fo:block hyphenate="true"
          hyphenation-character="-"
          hyphenation-keep="none"
          hyphenation-ladder-count="2"
          hyphenation-push-character-count="4"
          hyphenation-remain-character-count="4" >
  какое-то содержимое...
</fo:block>
```

XSL-FO не конкретизирует алгоритм разбивки слов для определения места, в котором может появляться мягкий перенос. Даже если эти свойства разрешают переносы, полностью от форматера зависит, как переносить каждое конкретное слово. Разумеется, форматеры могут вообще не пытаться переносить слова.

Свойства абзацных отступов

Свойства абзацных отступов определяют, на какое расстояние сдвигаются строки абзаца от обреза текста. Таких свойств существует четыре:

- start-indent
- end-indent
- text-indent
- last-line-end-indent

Свойство start-indent смещает все строки от начального края (левого края в английском языке). Свойство end-indent смещает все строки от конечного края (правого края в английском языке). Свойство text-indent смещает только первую строку от начального края. Свойство last-line-end-indent смещает только последнюю строку от начального края. Значениями является величина смещения. Например, стандартный абзац с полудюймовым смещением первой строки может форматироваться следующим образом:

```
<fo:block text-indent="0.5in">
  Первая строка абзаца смещена
</fo:block>
```

Врезка с дюймовым смещением всех строк с обеих сторон форматируется так:

```
<fo:block start-indent="1.0in" end-indent="1.0in">
  Текст смещен на один дюйм с обеих сторон.
</fo:block>
```

Поскольку атрибут text-indent добавляется к атрибуту start-indent для того, чтобы получить общее смещение первой строки, использование положительного значения в атрибуте start-indent и отрицательного значения в атрибуте text-indent создает висячие смещения. Например, в следующем абзаце все строки за исключением первой смещены на один дюйм. Первая строка смещается только на полдюйма:

```
<fo:block text-indent="-0.5in" start-indent="1.0in">
  В этом абзаце используется висячее смещение.
</fo:block>
```

Свойства символов

Символьные свойства описывают качества отдельных символов. Они применяются к элементам, которые содержат символы, например, к элементам fo:block и fo:list-item-body. Эти свойства описывают цвет, шрифт, стиль и другие характеристики символов.

Цветовые свойства

Свойство color задает цвет символа с помощью того же синтаксиса, который используется и в CSS-свойстве color. Например, в этом элементе fo:inline текст “Lions and tigers and bears, oh my!” окрашивается в розовый цвет:

```
<fo:inline color="#FFCCCC">
  Lions and tigers and bears, oh my!
</fo:inline>
```

Цвета задаются таким же самым способом, что и в CSS; то есть, в виде троек шестнадцатеричных чисел или одним из 16 именованных цветов aqua, black, blue, fuchsia, gray, green, lime, maroon, navy, olive, purple, red, silver, teal, white и yellow.

Свойства шрифта

Любой форматирующий объект, содержащий текст, может обладать большим диапазоном свойств, описывающих шрифт текста. Большая часть из них знакома нам по CSS, включая:

- font-family: Список названий гарнитур в порядке приоритета
- font-size: Кегль
- font-size-adjust: Желательное отношение между x-высотой и размером шрифта. Задается как натуральное число или ключевым словом none
- font-stretch: “толщина” шрифта, задается одним из ключевых слов condensed, expanded, extra-condensed, extra-expanded, narrower, normal, semi-condensed, semi-expanded, ultra-

condensed, ultra-expanded или wider

- font-style: стиль шрифта, задается одним из ключевых слов italic, normal, oblique, reverse-normal или reverse-oblique
- font-variant: либо normal, либо small-caps
- font-weight: толщина линий, которыми рисуется шрифт, задается в виде одного из ключевых слов 100, 200, 300, 400, 500, 600, 700, 800, 900, bold, bolder, lighter или normal

Свойства текста

Текстовые свойства применяют стили к тексту, который более или менее независим от выбранного шрифта. В их число входят:

- text-transform
- text-shadow
- text-decoration
- score-spaces

Свойство text-transform определяет, как капитализируется текст, это свойство идентично одноименному CSS-свойству. Возможны четыре значения:

- none: регистр текста не изменяется (значение по умолчанию)
- capitalize: каждая первая буква каждого слова делается заглавной, а все остальные буквы - прописными
- uppercase: все символы делаются заглавными
- lowercase: все символы делаются прописными

Это свойство несколько зависит от языка. (например, в китайском языке и иврите заглавные и прописные буквы не различаются) Форматеры вполне могут игнорировать это свойство при работе с не романскими текстами.

Свойство text-shadow создает у текста тень. Она похожа на фоновый цвет, но разница в том, что тень привязана к самому тексту, а не к прямоугольной зоне, в которой он размещается. Значением этого свойства может быть ключевое слово none или цвет, указанный именем или шестнадцатеричным числом. Пример:

```
<fo:inline text-shadow="FFFF66">
  Это предложение желтое.
</fo:inline>
```

Свойство text-decoration аналогично CSS-свойству text-decoration. Как и последнее, оно может иметь одно из пяти возможных значений:

- none: текст не оформляется, это значение по умолчанию
- underline: текст подчеркивается
- overline: над текстом проводится линия
- line-through: текст перечеркивается линией
- blink: известная выдумка Netscape - мерцающий текст

Кроме этих пяти значений, которые известны из CSS, в XSL-FO добавлено четыре значения, которые выключают оформление текста, унаследованное от родительского элемента:

- no-underline
- no-overline
- no-line-through

- no-blink

Свойство score-space определяет, действует ли свойство text-decoration на пробелы в тексте. Например, если значение свойства score-spaces равно true, подчеркнутое предложение будет выглядеть так:

если значение свойства score-spaces равно true, подчеркнутое предложение будет выглядеть так:

Свойства предложения

Свойства предложений применяются к группам символов, то есть, эти свойства имеют смысл только для нескольких букв сразу, например, расстояние между буквами слова.

Свойства разрядки букв

Кернинг текста - это относительная мера расстояния между двумя буквами. Кернинг не является абсолютным числом. Большая часть форматеров определяют расстояние между буквами по необходимости, особенно в случае выравнивания по обоим краям текста. Более того, в высококачественных шрифтах между различными символами используется различное пустое расстояние. Тем не менее, в целом можно уменьшить или увеличить разрядку текста.

Свойство letter-spacing добавляет дополнительное пространство между любой парой глифов - кроме того, что определяется кернингом. Свойство определяется в виде длины дополнительного пространства, например:

```
<fo:block letter-spacing="2px">
  Это очень разряженный текст
</fo:block>
```

Длина может быть отрицательной для уплотнения текста. Однако, форматеры обычно ограничивают количество пространства, которое можно добавить или убрать между буквами.

Свойства разрядки слов

Свойство word-spacing определяет количество пространства между словами. То есть, это свойство напоминает разрядку между буквами. Значение этого свойства - длина добавляемого или отнимаемого между словами пространства, например:

```
<fo:block word-spacing="0.3cm">
  Это очень разряженный текст
</fo:block>
```

Свойства разрядки строк

Форматирующие машины XSL-FO разделяют блоковые зоны на строковые. В XSL-FO невозможно прямо создать строковую зону. Однако, с помощью следующих пяти свойств можно влиять на то, как вертикально разносятся строки:

- line-height: минимальная высота строки

- line-height-shift-adjustment: имеет значение consider-shifts если верхние и нижние индексы должны увеличивать высоту строки; или значение disregard-shifts если не должны
- line-stacking-strategy: может иметь значение line-height (по модели CSS, это значение по умолчанию); font-height (делает высоту строки такой же, как высота шрифта плюс text-altitude и text-depth); или max-height (расстояние между максимальной высотой верхнего края литеры и максимальной глубиной нижнего края)
- text-depth: длина, задающая дополнительное вертикальное пространство, которое добавляется после каждой строки; значением также может быть ключевое слово use-font-metrics (используется по умолчанию), которое указывает, что свойство зависит от шрифта
- text-altitude: длина, задающая минимальное вертикальное пространство, добавляемое перед каждой строкой; значением также может быть ключевое слово use-font-metrics (используется по умолчанию), которое указывает, что свойство зависит от шрифта

Высота строки также сильно зависит от размера шрифта, в котором выводится строка. Обычно чем больше шрифт, тем выше строка. Например, следующий абзац из Mary Wollstonecraft (A Vindication of the Rights of Woman) выведен с удвоенным расстоянием между строками:

```
<fo:block font-size="12pt" line-height="24pt">
  In the present state of society it appears necessary to go
  back to first principles in search of the most simple truths,
  and to dispute with some prevailing prejudice every inch of
  ground. To clear my way, I must be allowed to ask some plain
  questions, and the answers will probably appear as
  unequivocal as the axioms on which reasoning is built;
  though, when entangled with various motives of action, they
  are formally contradicted, either by the words or conduct
  of men.
</fo:block>
```

Свойства выравнивания текста

Свойства text-align и text-align-last определяют, как горизонтально выравнивается содержание внутри прямоугольной зоны. Допустимы восемь значений:

- start: выравнивание по левому краю в языках, в которых письмо ведется слева направо
- center: центрирование
- end: выравнивание по правому краю в языках, в которых письмо ведется слева направо
- justify: выравнивание по обоим краям, добавляется необходимое пустое пространство между словами и буквами
- left: выравнивание по левому краю, независимо от направления письма в данном языке
- right: выравнивание по правому краю, независимо от направления письма в данном языке
- inside: выравнивание по внутренним краям страниц, то есть, выравнивание по правому краю левой страницы разворота и выравнивание по левому краю правой страницы разворота
- outside: выравнивание по внешним краям страниц, то есть, выравнивание по правому краю правой страницы разворота и выравнивание по левому краю левой страницы разворота

Свойство text-align-last позволяет задавать особое значение свойства выравнивания для последней строки в блоке. Это особенно важно в случае выравнивания текста по обоим краям, когда последняя строка блока часто содержит слишком мало слов для эффективного выравнивания по обоим краям. Для этого свойства допустимы те же самые значения, что и для свойства text-align, плюс значение relative. Это значение задает относительное выравнива-

ние последней строки - ее выравнивание будет такое же, как и у всего блока, за исключением случая, когда блок выровнен по обоим краям - в этом случае строка выравнивается по начальному краю блока.

Свойства пробелов и символов форматирования

Свойство space-treatment определяет, что должна делать форматирующая машина с пробелами и символами форматирования, которые остаются в документе после того, как оригинальный документ был преобразован в форматирующие объекты. Оно может иметь значение preserve (по умолчанию) или ignore. Если установлено последнее значение, ведущие и замыкающие пробелы и символы форматирования будут удаляться.

Свойство white-space-collapse может иметь значение true (по умолчанию) или false. В первом случае несколько подряд идущих пробелов заменяются единственным. Во втором случае они оставляются в неизменном виде.

Свойство wrap-option определяет, что происходит со слишком длинным для одной строки текстом. Это свойство может иметь значение wrap (по умолчанию) или no-wrap. В первом случае форматеру разрешается вставить в текст перенос строки, если иначе он не вмещается в строку.

Свойства зоны

Свойства зоны применяются к прямоугольным зонам страницы, боксам - это могут быть блоковые или внутри-строчные объекты. Каждый из боксов имеет:

- Фон
- Отступы
- Границы
- Отбивки
- Размер

Свойства фона

Свойства фона идентичны свойствам фона в CSS. Этих свойств пять:

- Свойство background-color определяет цвет фона. Его значением может быть цвет, например, red или #FFCCCC, или ключевое слово transparent.
- Свойство background-image задает URI изображения, которое используется в качестве фонового рисунка. Его значением может быть также ключевое слово none.
- Свойство background-attachment определяет, привязывается ли фоновый рисунок к окну или к документу. Его значением является одно из двух ключевых слов: fixed или scroll.
- Свойство background-position определяет, в какой части бокса размещается фоновое изображение. Возможными значениями являются center, left, right, bottom, middle, top или определенные координаты.
- Свойство background-repeat определяет, повторяется ли фоновый рисунок, если он меньше, чем бокс, и как именно он повторяется. Возможные значения - repeat, no-repeat, repeat-x и repeat-y.

Следующее описание блока использует свойства background-image, background-position, background-repeat и background-color:

```
<fo:block background-image="/bg/paper.gif"
          background-position="0,0"
          background-repeat="repeat"
          background-color="white">
  Two strings walk into a bar...
</fo:block>
```

Замечание

Единственное свойство фона, которое поддерживается FOP 0.18.1 - background-color. Поддержка других свойств, вероятно, будет добавлена в следующих релизах.

Свойства границ

Свойства границ описывают внешний вид границ вокруг бокса. Они аналогичны соответствующим свойствам CSS. Однако, кроме свойств border- XXX -bottom, border- XXX -top, border- XXX -left и border- XXX -right, в XSL-FO имеются также свойства border- XXX -before, border- XXX -after, border- XXX -start и border- XXX -end. В общей сложности получается 31 свойство границ. Это:

- Свойства цвета: border-color, border-before-color, border-after-color, border-start-color, border-end-color, border-top-color, border-bottom-color, border-left-color и border-right-color. По умолчанию используется черный цвет границ.
- Ширина: border-width, border-before-width, border-after-width, border-start-width, border-end-width, border-top-width, border-bottom-width, border-left-width и border-right-width. По умолчанию используется значение ширины medium.
- Стиль: border-style, border-before-style, border-after-style, border-start-style, border-end-style, border-top-style, border-bottom-style, border-left-style, border-right-style. По умолчанию используется значение none.
- Сокращенные свойства: border, border-top, border-bottom, border-left, border-right, border-color, border-style, border-width.

Например, для следующего блока заданы границы синего цвета толщиной в два пикселя:

```
<fo:block border-before-color="blue" border-before-width="2px"
          border-after-color="blue"  border-after-width="2px"
          border-start-color="blue"  border-start-width="2px"
          border-end-color="blue"    border-end-width="2px">
  You have been selected for special high intensity training.
</fo:block>
```

Свойства отбивок

Свойства отбивок определяют количество пространства между границей бокса и его содержимым. Граница бокса, если она видна, разделяет области отбивки и отступа бокса. Свойства отбивок похожи на аналогичные свойства CSS. Однако, кроме свойств padding-bottom, padding-top, padding-left и padding-right, в XSL-FO также имеются свойства padding-before, padding-after, padding-start и padding-end. Таким образом, всего имеется восемь свойств отбивок, значением каждого является размер отбивки. Вот эти свойства:

- padding-after
- padding-before
- padding-bottom
- padding-end
- padding-left
- padding-start
- padding-right
- padding-top

Например, следующий блок имеет отбивку в полсантиметра с каждой стороны:

```
<fo:block padding-before="0.5cm" padding-after="0.5cm"
          padding-start="0.5cm"  padding-end="0.5cm">
  Did you hear the one about the dyslexic agnostic?
</fo:block>
```

Свойства отступов для блоков

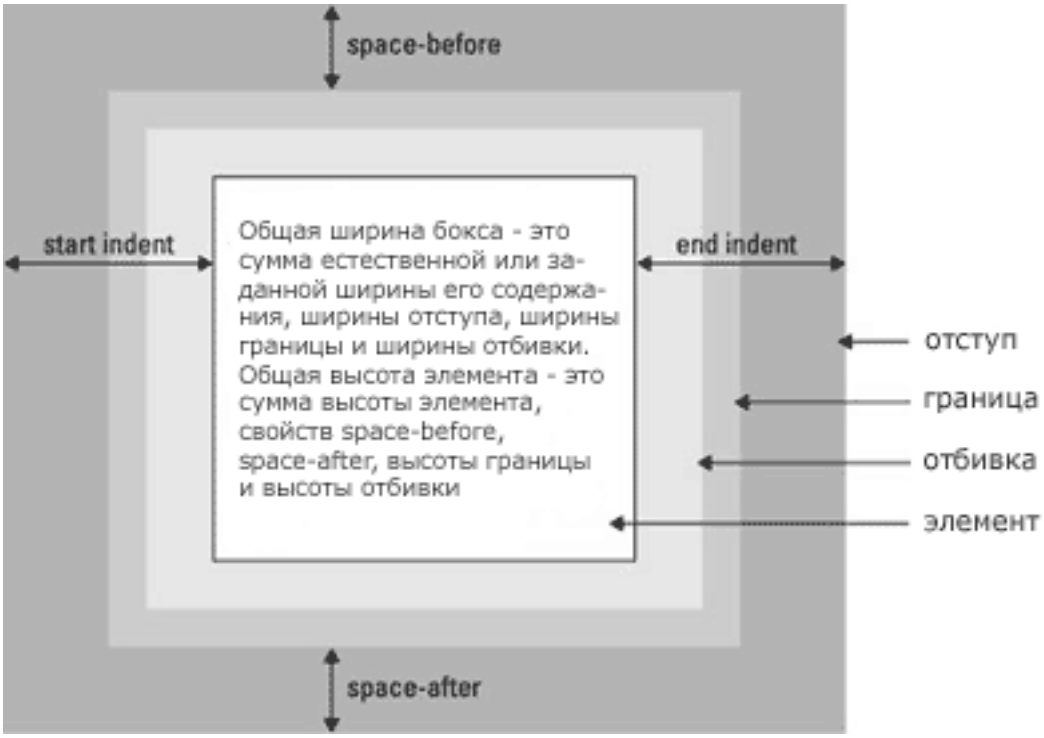
Существует пять свойств отступов, значением их является величина отступа. Вот эти пять свойств:

- margin-top
- margin-bottom
- margin-left
- margin-right
- margin

Однако, эти свойства предусмотрены только для совместимости с CSS. В общем случае рекомендуется использовать другие четыре свойства, поскольку они лучше согласуются с моделью XSL-FO:

- space-before
- space-after
- start-indent
- end-indent

Свойства space-before и space-after эквивалентны свойствам margin-top и margin-bottom соответственно. Свойство start-indent эквивалентно сумме значений свойств padding-left, border-left-width и margin-left. Свойство end-indent эквивалентно сумме значений свойств padding-right, border-right-width и margin-right. Следующий рисунок прояснит ситуацию.



Например, следующий блок имеет отступы в полсантиметра на своей начальной и конечной стороне:

```
<fo:block start-indent="0.5cm" end-indent="0.5cm">
  Two strings walk into a bar...
</fo:block>
```

Однако, в отличие от отступов, свойства space before и space after задаются как характеристики пространства, содержащие несколько значений. В частности, они содержат предпочтительное значение, минимальное значение и максимальное значение, а также условие и определенный приоритет. Это оставляет форматеру некоторую свободу в создании макета страницы. Он может взять любое значение из промежутка от минимального до максимального и выберет такое, которое наилучшим образом согласуется со всей остальной страницей.

Каждая из характеристик пространства представляет собой длину. Условие задается одним из двух ключевых слов discard или retain. Оно определяет, что должно произойти с лишним пространством в конце строки. По умолчанию оно устраняется (discard). Приоритет задается либо целым числом, либо ключевым словом force. Приоритет определяет, что будет происходить, если характеристика space-end одной внутри-строчной зоны конфликтует с характеристикой space-start следующей внутри-строчной зоны. Зона, имеющая больший приоритет получает превосходство. Приоритет по умолчанию равен нулю. Все пять значений этих свойств разделяются точками с запятой.

Рассмотрим, например, следующий элемент fo:block:

```
<fo:block space-before="0in;0.5in;0.166in;discard;force">
 It goes to 11.
</fo:block>
```

Здесь указывается, что в идеале форматер должен добавить перед этим элементом пространство в одну шестую дюйма. Однако, он может и совсем не добавить пространства или может его добавить полдюйма, если понадобится. Поскольку приоритет задан ключевым словом force, это определение получит превосходство над любыми характеристиками пространства,

которые будут конфликтовать с ним. И наконец, если в конце строки останется лишнее пустое пространство, оно будет удалено.

Свойства отступа для строковых боксов

Два свойства отступов применяются только к внутри-строчным элементам:

- space-end
- space-start

Их значением являются характеристики пространства, которые задают диапазон дополнительного пространства, которое будет добавлено перед и после элемента. Реальные пространства могут быть меньше или больше. Поскольку пространства не являются частью собственно боксов, пространство end space одного бокса может быть одновременно пространством start space другого бокса.

Свойства размера

Шесть свойств определяют высоту и ширину контентной зоны бокса:

- height
- width
- max-height
- max-width
- min-height
- min-width

Эти свойства не задают общей ширины или высоты бокса, в которую входят границы, отступы и отбивки. Они задают только ширину и высоту зоны основного содержания бокса. Значением свойств height и width кроме единиц длины может быть и ключевое слово auto, которое позволяет определить ширину и высоту бокса исходя из его содержимого. Но в любом случае эта ширина и высота не может быть больше, чем заданы в свойствах max-height и max-width или меньше, чем заданы в свойствах min-height и min-width. Пример:

```
<fo:block height="2in" width="2in">
  Two strings walk into a bar...
</fo:block>
```

Свойства переполнения

Свойство overflow определяет, что происходит, если содержимое не вмещается в бокс заданного размера. Размер бокса может быть задан и явным образом с использованием свойств размера, так и неявным образом на основе размера страниц и других ограничений. Существует четыре возможности, каждой соответствует ключевое слово:

- auto: При возникновении переполнения используются полосы прокрутки; не используйте это значение, если полос прокрутки нет. Если полосы прокрутки невозможно реализовать (например, на печатной странице), для потокового содержимого создается еще одна страница, а для статического содержимого выдается сообщение об ошибке. Это значение используется по умолчанию.

- hidden: Скрывать любое содержимое, которое не вмещается в бокс.
- scroll: Создать в боксе полосы прокрутки, так чтобы пользователь мог увидеть дополнительное содержимое.
- visible: Показывается полное содержимое бокса; при необходимости ограничения на размер бокса преодолеваются.
- error-if-overflow: Форматер должен остановиться и выдать сообщение об ошибке в случае, если содержание не вмещается в заданный блок.
- paginate: Если объект переполняет страницу, для его размещения создается новая страница.

Свойство clip определяет форму области видимости, если свойство overflow не имеет значения visible. По умолчанию областью видимости является сам бокс. Однако, ее можно изменить, задав конкретный прямоугольник, например, так:

```
clip=rect(верхнее_смещение правое_смещение
                нижнее_смещение левое_смещение)
```

Здесь верхнее_смещение , правое_смещение , нижнее_смещение и левое_смещение - заданная величина смещений области видимости от соответственно верхней, правой, нижней и левой стороны бокса. Это позволяет сделать область видимости больше или меньше, чем сам по себе бокс.

Свойства ориентации

Свойство reference-orientation позволяет указывать, что содержимое бокса должно быть повернуто относительно своего обычного положения. Допустимы только углы с шагом в 90 градусов, отчет ведется против часовой стрелки: 0, 90, 180 и 270. Также можно использовать значения -90, -180 и -270. Например, вот так задается поворот на 90 градусов:

```
<fo:block reference-orientation="90">
  Текст идет снизу вверх
</fo:block>
```

Свойства режима письма

Режим письма задет направление текста в боксе. Он имеет большое влияние на порядок размещения форматирующих объектов в боксе. Обычно говорящие на английском и на других западных языках используют режим письма слева направо и сверху вниз, вот так:

```
A B C D E F G
H I J K L M N
O P Q R S T U
V W X Y Z
```

Однако в иврите и в арабском мире более естественным порядком является справа налево и сверху вниз:

```
G F E D C B A
N M L K J I H
U T S R Q P O
Z Y X W V
```

На Тайване принят порядок сверху-вниз и слева-направо:

```
A E I M Q U Y
B F J N R V Z
C G K O S W
D H L P T X
```

В XSL-FO режим письма не только влияет на текст. Он также влияет на то, в каком направлении упорядочиваются объекты потока, на то, как производятся переносы и на многое другое. Вы уже заметили, что многие свойства используют слова start (начальный), end (конечный), before (до-) и after (после-), а не обычные left (левый), right (правый), top (верхний) и bottom (нижний). Определение стилей в терминах start, end, before и after, а не left, right, top и bottom, позволяет разработать более гибкие и легко локализуемые таблицы стилей.

Свойство writing-mode задает режим письма для данной зоны. Это свойство может иметь одно из тринадцати возможных значений:

- bt-lr: снизу-вверх, слева-направо
- bt-rl: снизу-вверх, справа-налево
- lr-alternating-rl-bt: строки слева-направо чередуются строками справа-налево, снизу-вверх
- lr-alternating-rl-tb: строки слева-направо чередуются строками справа-налево, сверху-вниз
- lr-bt: слева-направо, снизу-вверх
- lr-inverting-rl-tb: Слева-направо, затем происходит сдвиг вверх на следующую строку и далее текст идет справа-налево (то есть, текст ползет по странице, образуя перевернутую букву S)
- lr-inverting-rl-tb: Слева-направо, затем происходит сдвиг вниз на следующую строку и далее текст идет справа-налево (то есть, текст ползет по странице, образуя перевернутую букву S)
- lr-tb: Слева-направо, сверху-вниз
- rl-bt: Справа-налево, снизу-вверх
- rl-tb: Справа-налево, сверху-вниз
- tb-lr: Сверху-вниз, слева-направо
- tb-rl: Сверху-вниз, справа-налево
- tb-rl-in-rl-pairs: Текст пишется группами по два символа, в паре письмо идет справа-налево; затем эти пары выкладываются сверху-вниз и образуют строку; строки выкладываются справа-налево

Висячие строки

Наборщики называют сиротой (orphan) единственную строку нового абзаца в самом низу страницы и вдовой - (widow) единственную строку абзаца наверху страницы. Хорошие наборщики перемещают эти сторки на следующую страницу или на предыдущую, чтобы не допустить возникновения висячих строк. Вы можете задать число строк, которые рассматривается как строки-сироты, установив его значением свойства orphans. Число строк, которые будут рассматриваться как строки-вдовы можно установить с помощью свойства widows. Например, если вы хотите, чтобы каждый конкретный абзац в конце страницы содержал, по меньшей мере, три строки, нужно установить значением свойства orphans число 3:

```
<fo:simple-page-master master-name="even"
    orphans="3" page-height="11in" page-width="8.5in"
/>
```

Аудиальные свойства

XSL-FO поддерживает весь набор аудиальных свойств CSS2, включая:

- azimuth
- cue
- cue-after
- cue-before
- elevation
- pause
- pause-after
- pause-before
- pitch
- pitch-range
- play-during
- richness
- speak
- speak-header
- speak-numeral
- speak-punctuation
- speech-rate
- stress
- voice-family
- volume

Естественно, эти свойства мало пригодны для большей части поддерживаемых в настоящее время средств вывода XSL-FO. Если XSL-FO будет поддерживаться браузерами напрямую, они могут получить большее значение.

Перекрестная ссылка

Аудиальные стилевые свойства описываются в последнем разделе главы 16. В XSL-FO они имеют то же самое значение и синтаксис, что и в CSS2.

Заключение

В этой главе мы познакомились с форматирующими объектами XSL. В частности, мы узнали, что:

- XSL-процессор следует инструкциям таблицы стилей XSLT и преобразует исходный XML-документ в новый XML-документ, размеченный с помощью словаря форматирующих объектов XSL.
- Большая часть форматирующих объектов XSL генерируют одну или несколько прямоугольных зон. Страницы образованы областями, области содержат блоковые зоны. Блоковые зоны содержат другие блоковые зоны и внутри-строчные зоны. Внутри-строчные зоны содержат другие внутри-строчные зоны и символьные зоны.

- Корневым элементом документа XSL-FO является элемент fo:root. Он содержит элементы fo:layout-master-set и элементы fo:page-sequence.
- Каждый элемент fo:layout-master-set содержит один или несколько элементов fo:simple-page-master, каждый из них задает макет определенного вида страниц с помощью разделения страницы на пять областей (до-область, после-область, начальную область, конечную область и основную область) и задания свойств каждой области. Кроме того, он может содержать один или несколько элементов fo:page-sequence-master.
- Каждый элемент fo:page-sequence может содержать один элемент fo:title, один элемент fo:static-content, один элемент fo:flow, а также атрибут master-reference. Содержимое элемента fo:flow копируется в экземпляры страниц, созданные с помощью данной мастер-страницы в порядке, заданном элементом fo:page-sequence-master, который идентифицируется по его атрибуту master-name. Содержимое элементов fo:static-content копируется на каждую создаваемую страницу.
- Элемент fo:external-graphic загружает изображение с заданного URL и отображает его как внутри-строчный элемент.
- Элемент fo:instream-foreign-object отображает изображение, закодированное в формате XML, например, в виде SVG или MathML, встроенного в документ XSL-FO.
- Элемент fo:basic-link создает гипертекстовую ссылку на определенный URL.
- Список представляется как элемент блокового уровня, который создается с помощью элемента fo:list-block. Он содержит элементы блокового уровня fo:list-item. Каждый элемент fo:list-item содержит элементы fo:list-item-label и fo:list-item-body, а они, в свою очередь содержат другие элементы блокового уровня.
- Элемент fo:page-number вставляет текущий номер страницы.
- Элемент fo:inline - это контейнер, предназначенный для привязывания некоторых свойств тексту и всей зоне, которую он охватывает.
- Элемент fo:footnote вставляет расположенную вне строки сноску и внутрь строки - ссылку на эту сноску.
- Элемент fo:float вставляет расположенный вне строки элемент блокового уровня, например, картинку или таблицу. Его свойство float определяет край страницы, к которому "всплывет" эта врезка, а свойство clear определяет, где и как разрешается другим элементам обтекать врезку.
- Существует более 200 отдельный форматирующих свойств XSL, многие из которых идентичны аналогичным одноименным свойствам в CSS. Они привязываются к форматирующим объектам в виде атрибутов.
- Свойства разрывов страниц описывают, где разрешается, а где - не разрешается делать разрыв страницы. Они включают в себя keep-with-next, keep-with-previous, keep-together, break-before, break-after, widows и orphans.
- Свойства переносов описывают, где и как вставляются мягкие дефисы. Они включают в себя hyphenate, hyphenation-character, hyphenation-keep, hyphenation-ladder-count, hyphenation-push-character-count и hyphenation-remain-character-count.
- Свойства абзацного отступа определяют, насколько далеко смещается строка относительно края текста. Этих свойств четыре: start-indent, end-indent, text-indent и last-line-end-indent.
- Свойства символов описывают атрибуты отдельных символов. Они включают в себя: color, font-family, font-size, font-size-adjust, font-stretch, font-style, font-variant, font-weight, text-transform, text-shadow, text-decoration и score-space.
- Свойства предложения описывают форматирование, которое имеет смысл только для группы букв или слов. Эти свойства включают в себя letter-spacing, word-spacing, line-height, line-height-shift-adjustment, line-stacking-strategy, text-depth, text-altitude, text-align, text-align-last, space-treatment, white-space-collapse и wrap-option.
- Свойства зоны описывают атрибуты боксов, генерируемых различными форматирующими объектами, и включают в себя свойства фона, отбивки и отступов.
- XSL-FO поддерживает полное множество аудиальных стилей, определенных в CSS2 для го-

лосового чтения документов.

В следующей главе речь пойдет о XLinks. XLinks предоставляет более мощный синтаксис описания ссылок, чем предусмотрено в стандартном HTML (элемент A) или в XSL (элемент fo:basic-link).