Book Title

Book Title

FirstName LastName

Book Title
by FirstName LastName

Printing History:

Table of Contents

Preface

asciidoc version of boilerplate preface, via passthrough
(from https://prod.oreilly.com/external/tools/docbook/prod/trunk/samples/ch00.xml@297910)
Skip to the first chapter for asciidoc markup demo; below is just "passthrough" DocBook markup...

Conventions Used in This Book

The following typographical conventions are used in this book:

Italic
> Indicates new terms, URLs, email addresses, filenames, and file extensions.

`Constant width`
> Used for program listings, as well as within paragraphs to refer to program elements such as variable or function names, databases, data types, environment variables, statements, and keywords.
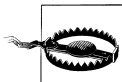
`Constant width bold`
> Shows commands or other text that should be typed literally by the user.

`Constant width italic`
> Shows text that should be replaced with user-supplied values or by values determined by context.

> This icon signifies a tip, suggestion, or general note.

> This icon indicates a warning or caution.

Using Code Examples

This book is here to help you get your job done. In general, you may use the code in this book in your programs and documentation. You do not need to contact us for permission unless you're reproducing a significant portion of the code. For example, writing a program that uses several chunks of code from this book does not require permission. Selling or distributing a CD-ROM of examples from O'Reilly books does require permission. Answering a question by citing this book and quoting example code does not require permission. Incorporating a significant amount of example code from this book into your product's documentation does require permission.

We appreciate, but do not require, attribution. An attribution usually includes the title, author, publisher, and ISBN. For example: "*Book Title* by Some Author (O'Reilly). Copyright 2011 Some Copyright Holder, 978-0-596-xxxx-x."

If you feel your use of code examples falls outside fair use or the permission given above, feel free to contact us at *permissions@oreilly.com*.

Safari® Books Online

Safari Books Online is an on-demand digital library that lets you easily search over 7,500 technology and creative reference books and videos to find the answers you need quickly.

With a subscription, you can read any page and watch any video from our library online. Read books on your cell phone and mobile devices. Access new titles before they are available for print, and get exclusive access to manuscripts in development and post feedback for the authors. Copy and paste code samples, organize your favorites, download chapters, bookmark key sections, create notes, print out pages, and benefit from tons of other time-saving features.

O'Reilly Media has uploaded this book to the Safari Books Online service. To have full digital access to this book and others on similar topics from O'Reilly and other publishers, sign up for free at *http://my.safaribooksonline.com*.

How to Contact Us

Please address comments and questions concerning this book to the publisher:

> O'Reilly Media, Inc.
> 1005 Gravenstein Highway North
> Sebastopol, CA 95472
> 800-998-9938 (in the United States or Canada)
> 707-829-0515 (international or local)
> 707-829-0104 (fax)

We have a web page for this book, where we list errata, examples, and any additional information. You can access this page at:

http://www.oreilly.com/catalog/<catalog page>

Don't forget to update the <url> attribute, too.

To comment or ask technical questions about this book, send email to:

bookquestions@oreilly.com

For more information about our books, courses, conferences, and news, see our website at *http://www.oreilly.com*.

Find us on Facebook: *http://facebook.com/oreilly*

Follow us on Twitter: *http://twitter.com/oreillymedia*

Watch us on YouTube: *http://www.youtube.com/oreillymedia*

Markup Demo

This is a chapter.

This file illustrates some of the most common elements used in O'Reilly books.

> Section titles can be in either of two formats: "underlined" or delimited.
>
> Note that the levels described in the AsciiDoc User Guide (*http://www .methods.co.nz/asciidoc/userguide.html*) can be confusing: in asciidoc, the document (book) is considered level 0; generally a chapter will be at asciidoc level 1 (unless you're dividing the book into parts), and sections within chapters start at asciidoc level 3 (which is equivalent to DocBook `<sect1>`).

Top-Level Section Title (Underlined Style)

Narrative text here.

Second-Level Section Title (Underlined Style)

More text here.

Third-Level Section Title (Underlined Style)

And some more here.

Top-Level Section (Delimited Style)

This one designates asciidoc level 2 (DocBook `<sect1>`).

Second-Level Section (Delimited Style)

This one designates asciidoc level 3 (DocBook `<sect2>`). Also note that you can have the delimiter just on the left as in this one, or on both sides as in "Top-Level Section (Delimited Style)" on page 1.

Third-Level Section (Delimited Style)

This one designates asciidoc level 4 (DocBook `<sect3>`).

Cross-References

To generate a cross-reference, use this syntax:

```
<<ID>>
```

where `ID` is the anchor or BlockID of the target, which you place in double square-brackets above that block.

The O'Reilly build system will transform this into a standard `<xref>` for you: the rendered text will adjust automatically if you later move the target or reword its title, and it will work as a hyperlink in online versions. Any time you refer to another component of your book, please be sure to use xref markup, not hardcoded text.

Table 1-1 shows the standard text generated from xrefs in PDF builds.

Table 1-1. Standard Cross-Reference Formats

Target	Generated Cross-Reference Text
chapter	Chapter 17
table	Table 4-1
figure	Figure 2-3
example	Example 3-5
sidebar	"Fooing the Bar" on page 23
section	"Inline Macros" on page 14

Here are some live examples (hover over the text in the PDF to locate the hyperlink):

- See "Block Elements" on page 5 for details.
- The results is shown in Figure 1-1.
- Flip ahead to Chapter 2 for a preview.

generated from this source:

```
* See <<BLOCKS>> for details.
* The results is shown in <<FIG1>>.
* Flip ahead to <<chapid_2>> for a preview.
```
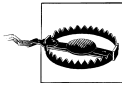
Please do not use asciidoc's optional `xreflabel` and `caption` features on anchors and xrefs, as these interfere with our standard generated xref formats (and will likely be stripped on intake). Your production editor has some flexibility in using alternate xref formats later in production where appropriate. If you have questions about xref formats or want customizations, please contact *toolsreq@oreilly.com*

Inline Elements

Here's some some *italic* and some `monospaced` (aka "constant-width" or "CW") text.

Backticks can also be used for literal (CW) text, for example: `ls -al`

Using inlines in asciidoc can be tricky:

- Delimiters may not be interpreted as intended if they don't abut whitespace on both sides; the fix for this is to double them up, as explained under "Constrained and Unconstrained Quotes" (*http://www.methods.co.nz/asciidoc/userguide.html #X52*) in the AsciiDoc User Guide. For example, compare how these render in the PDF: +*foo*+ *bar* vs. *foo bar*

- Attempting to nest inlines within each other may result in invalid DocBook (because the content model of one may not include the other). For example, **foo** requires the asterisk (*) pair outside and the plus-symbol (+) pair inside. If you try the other way around, the build will fail and you'll find a message like this in *pdf/.dcpsgen.buildlog*:

  ```
  Element emphasis is not declared in literal list of possible children
  ```

- For more precise semantics, you can always use "passthroughs" to embed DocBook markup as is, e.g. *foo*, **foo**, and ***foo*** are fairly common in O'Reilly books, rendered as CWI, CWB, and CWBI, respectively (constant-width with italic, bold, and bold +italic).

  > While it's easy to apply **bold** and it's one of our standard conventions in code (to denote `user input`), please note that it's rarely used for body text in O'Reilly books (for general emphasis, our style is to use *italics*). Please discuss with your editor if you have questions about whether or when boldface is appropriate.

For generic emphasis, you can also surround a string with single, straight quotation marks (apostrophes): i.e., *this* is equivalent to *that*. As an enhancement for finer-tuned semantic distinctions downstream, the ORM config supports the following variants, via `@role` attributes:

- to indicate a filename or path, use */path/to/file.ext* or */path/to/file.ext*
- for a book title, use *This Book Needs No Title* or *This Book Needs No Title*

To get curly quotes you can use asciidoc's mechanism for 'singles' and "doubles" (or you can always use Unicode like so: "quote me"). Note that it's also fine to leave straight quotes in running text; part of our intake process curls quotes in body, while leaving them unchanged in literal and code contexts.

Subscripts and superscripts work like so: H_2O and $2^5 = 32$ (but if you're doing math, you'd probably want to italicize the variables, like so: $x^2 + y^2 = z^2$). For more on math see !!ORMTODO!!.

Standard O'Reilly font conventions are as follows:

DocBook passthrough, for comparison:

```
emphasis (italic)
literal (constant width)
userinput (constant-width bold)
replaceable (constant-width italic)
userinput+replaceable (constant-width bold+italic)
```

These are usually explained in a `<variablelist>` (asciidoc's "labeled list") in the preface, but note that expressing inlines in the "item label" (`<term>`) as opposed to the text description (`<listitem>`) can be tricky, as shown in the first item below. If you have questions about how to do these, please ask...

ORMTODO: Fix/explain first below (override for <term role=plain>)

Plain
> Body text.

Italic
> *Ital* indicates new terms, URLs, email addresses, filenames, and file extensions.

`Constant width`
> `CW` is used for program listings, as well as within paragraphs to refer to program elements such as variable or function names, databases, data types, environment variables, statements, and keywords.

`Constant width bold`
> **`CWB`** shows commands or other text that should be typed literally by the user.

`Constant width italic`
> *`CWI`* shows text that should be replaced with user-supplied values or by values determined by context.

> Please do not use asciidoc's mechanisms for forcing linebreaks, page breaks, or "ruler" lines, as these don't mesh with our production process. (Your production editor will handle linebreaks and pagebreaking at appropriate stages of the production process.)

If using footnotes, please place the macro[*] directly after the text, with no space between (otherwise you'll introduce extra space [†] before the in-text mark); footnotes at the end of a sentence belong after the period.[‡]

Block Elements

Sidebars

Here's a `<sidebar>`:

Titled vs. Untitled Blocks

O'Reilly house style generally uses titles only on formal blocks (figures, tables, examples, and sidebars in particular). Although asciidoc supports optional titles on many other blocks, these generally are not appropriate for O'Reilly books.

Conversely, *omitting* a title from a sidebar is generally not conformant to O'Reilly style. If you have questions about whether content belongs in an sidebar vs. admonition vs. quote or other element, please consult with your editor.

Code ("Listing" Blocks and Examples)

Here's a code block:

```
Hello world!

Standard line length for code is 85 in an "Animal" book:

0        10        20        30        40        50        60        70        80
12345678901234567890123456789012345678901234567890123456789012345678901234567890012345
```

Anything longer than the above will spill into the margin and have to be broken before the book goes to print. It's most efficient for you to keep an eye on code lengths and adjust linebreaks yourself before submitting the book for production.

Note that the maximum length within a formal example is slightly longer (as shown in Example 1-1); within other elements such as notes and sidebars, it will be slightly smaller. (If your book is a "small Animal" (6x9) or a different series, its width constraints will be different; more extensive examples can be found at *https://prod.oreilly.com/external/tools/docbook/prod/trunk/samples/code_lengths/* but please check with your editor or *toolsreq@oreilly.com* if you are unsure which applies to your book.)

---

[*] This is a standard footnote.

[†] This one has extraneous space before the in-text mark (although the footnote itself is fine).

[‡] If your footnote text includes [square brackets], you can escape them with a passthrough macro.

Contrast the code block above with Example 1-1, which is a *formal* code example (titled and cross-referenced).

Example 1-1. An Example

```
Hello world!
```

```
Note that the line length is a bit longer here (90 in an Animal):
```

```
0         10        20        30        40        50        60        70        80         9
123456789012345678901234567890123456789012345678901234567890123456789012345678901234567890
```

Inline Formatting within Code

In asciidoc, there is no built-in mechanism for inline formatting within code. If you want to use inline formatting—in particular, for standard O'Reilly font conventions such as **<userinput> (CW+bold)** and *<replaceable> (CW+italic)* or if you want to include line annotations—you can do so by using a passthrough block, like so:

```
hostname $ date
Sun Apr  1 12:34:56 GMT 1984
```

or this:

```
from __future__ import with_statement # This isn't required in Python 2.6
                            Above is a comment in the code, while this is an "annotation" for the book
with open("hello.txt") as f:
    for line in f:          (note regular italic here vs. constant-width in "hello.txt" on line above)
        print line
```

Notes:

- code blocks in asciidoc convert to <screen> by default; when using a passthrough block, you can choose whether to make it <screen> or <programlisting> (they'll render the same in the PDF).
- all whitespace in <programlisting>s and <screen>s is significant, including leading or trailing blank lines, as shown below

To avoid extra vertical space in layout when using passthrough blocks for code, do them like this:

```
class C:
    def method(self):
        pass
```

not like this:

```

class C:
    def method(self):
        pass
```

External Code Files

To include an external code file that is text-only (no markup), use the `include::` macro inside of a delimited code block, as shown here:

```
package com.marakana;

import android.app.Activity;
import android.os.Bundle;

public class HelloWorld extends Activity {
        /** Called when the activity is first created. */
        @Override
        public void onCreate(Bundle savedInstanceState) {
                super.onCreate(savedInstanceState);
                setContentView(R.layout.main);
        }
}
```

To include an external file that contains inline markup (e.g., for callouts, line annotations, or inline font formatting), skip the code block delimiters around the `include::` macro, and instead place passthrough delimiters within the included file, like so:

```
int GetCustLastOrder(DatabaseHandle db, string customer_id, Order* order) {
  string customer_name;
  LOOKUP_OR_RETURN("names", customer_id, &customer_name);   ❶
  order->set_customer_name(customer_name);

  string last_order_id;   ❷
  LOOKUP_OR_RETURN("recent_orders", customer_id, &last_order_id); lineannotation...

  string order_data;   ❸
  LOOKUP_OR_RETURN("orders", last_order_id, &order_data);
  order->set_order_data(order_data);

  return SUCCESS;
}

#define LOOKUP_OR_RETURN(table_id, row_id, data_ptr) { \
  Table table; \
  int err = OpenTable(db, table_id, &table);   \
  if (err != SUCCESS) return err;   \
  err = RetrieveRow(db, table, row_id, data_ptr); \
  if (err != SUCCESS) return err; \
}
```

❶ When using this approach to include code from an external file, you'll have to escape special XML characters such as `&` (`&`), `>` (`>`), and `<` (`>`).

❷ Inline markup here: `<replaceable>`

❸ Inline markup here: `<userinput>`

Code Callouts

In DocBook, code callouts are used to mark specific lines of code with icons keyed to explanatory text outside the code block. These icon pairs function as bidirectional links in electronic PDF and downstream formats (i.e., you can click on the icon in the code to jump to the explanation, and vice versa).

The built-in asciidoc mechanism (shown in Example 1-2) is somewhat more limited; for one thing, icons are hyperlinked from text to code, but not vice versa. However, you can always use a passthrough block for full functionality, as shown in Example 1-3.

If you have a need to refer to the same bit of explanatory text from more than one line of code, see "Yet More on Code Callouts" on page 13 (in Chapter 2).

Example 1-2. Built-in Callout Mechanism (Unidirectional Callouts)

```
10/17/97    9:04        <DIR>      bin
10/16/97   14:11        <DIR>      DOS              ❶
10/16/97   14:40        <DIR>      Program Files
10/16/97   14:46        <DIR>      TEMP
10/17/97    9:04        <DIR>      tmp
10/16/97   14:37        <DIR>      WINNT
10/16/97   14:25              119  AUTOEXEC.BAT     ❷
 2/13/94    6:21           54,619  COMMAND.COM
10/16/97   14:25              115  CONFIG.SYS
11/16/97   17:17       61,865,984  pagefile.sys
 2/13/94    6:21            9,349  WINA20.386       ❸
```
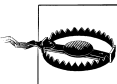
❶ This directory holds MS-DOS.

❷ System startup code for DOS.

❸ Some sort of Windows 3.1 hack.

Example 1-3. Bidirectional Callouts, via Passthrough Markup

```
class C: ❶
    def method(self):
        pass
```

❶ The benefits of bidirectional callouts are more evident on longer listings...

> If using passthroughs for callouts, please use the paired markup as shown above (not `<areaspec>`, which our toolchain does not support). For more on DocBook callout markup, see "Annotating Your Code in DocBook" in our DocBook authoring docs (*https://prod.oreilly.com/external/tools/docbook/docs/authoring/html/docbookguidelines.html#id2557661*).

Admonitions (Notes and Warnings)

Here are some admonitions:
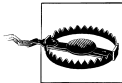
O'Reilly Animal books traditionally make no distinction between the DocBook `<note>`, `<tip>`, and `<important>` elements.

Titled Admonition

We do support optional titles in admonitions (in most series).

O'Reilly Animal books traditionally make no distinction between the DocBook `<warning>` and `<caution>` elements.

O'Reilly Animal books traditionally make no distinction between the DocBook `<warning>` and `<caution>` elements.

Figures and Other Images

Somewhere nearby is Figure 1-1 (a figure, titled and cross-referenced). Unlike other block elements, figures "float" by default (after the book enters production, your production editor will adjust their placement as needed). While asciidoc permits you to specify various image-related attributes for scaling/alignment/etc.,§ generally these sorts of adjustments are not necessary for production (and some of them are only for its HTML output, not DocBook). As part of the production process, your images will be processed by a professional illustrator to ensure their size, resolution, color, and fonts are appropriate for the printed page as well as online channels.

However, if you find that any of your images overflow the PDF page as is, then you can you can force-scale them by setting a `width` attribute as shown in the markup for Figure 1-2; note that this is strictly for your own convenience while reviewing the PDFs; it's not necessary for production. (When the manuscript is submitted for production, our "intake" process will strip out any tweaks you've made to image-related attributes, instead using standardized markup along with reprocessed versions of your images prepared by illustration staff.)

For general guidelines on preparing illustrations—along with how and when to get in touch with our illustration staff—see *https://prod.oreilly.com/external/illustrations/*. Thanks for your attention to these details.

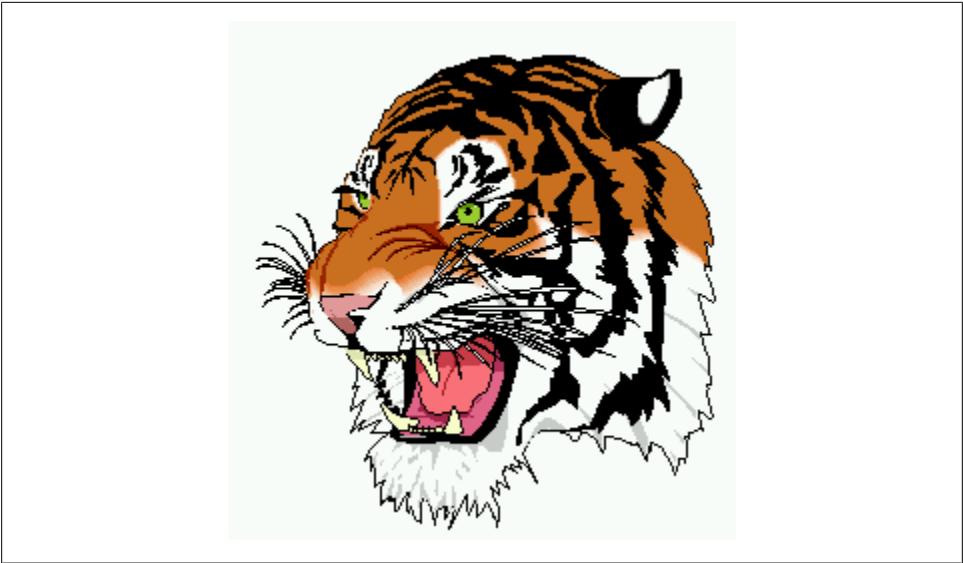§ *http://www.methods.co.nz/asciidoc/userguide.html#X9*

Figure 1-1. A Figure

When Does an Image Not Belong in a Figure?

O'Reilly books generally use informal figures (standalone images without captions) sparingly; please consult first with your editor.

Inline images (such as icons) are appropriate on occasion, but may cause problems downstream, in addition to requiring additional processing by illustration staff (for sizing relative to text) and manual placement during production (to ensure that they align appropriately). If you plan to use inline images, please discuss the rationale early on with your editor and the illustration staff, and call our attention to them when the book is submitted to production so we can handle them properly.

Here's some dummy around an inline image to demonstrate why these can be problematic. Even if were sized appropriately relative to the text height in the printed book and PDF, that might not work so well for other downstream channels.

Lists

Labeled (aka Variable or Term-Definition) Lists

Term 1
    Definition/description

Term 2
    Something else

Figure 1-2. Another Figure
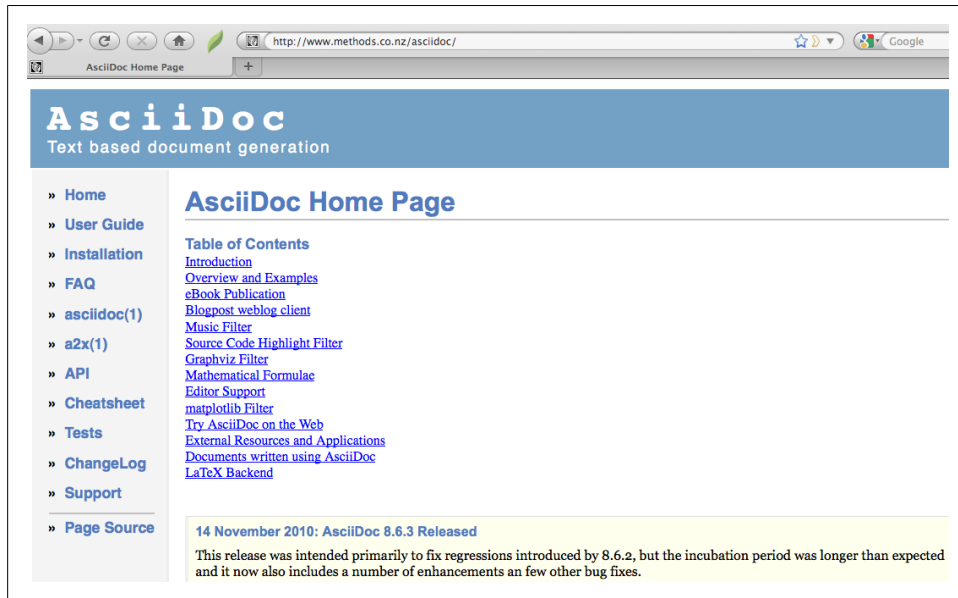
Bulleted (aka Itemized) Lists

- lions
- tigers
  - sabre-toothed
  - teapotted
- bears

Ordered (aka Numbered) Lists

1. Preparation
2. Assembly
   a. Measure
   b. Combine
   c. Bake
3. Applause

Tables

O'Reilly table styles vary slightly between series; those styles are supplied by our stylesheets. Please don't use asciidoc markup to change the gridlines and so forth, as such

alterations may be stripped out once the book moves into production. Similarly, don't bother fine-tuning things like column widths, as that will be handled by your production editor for the print version (and may not carry over for downstream channels). If your material warrants something other than the default style as shown in Table 1-2, please consult with your editor and be sure to communicate what you intend when the book is submitted for production.

Table 1-2. A Table

P	Q	P^Q
T	T	T
T	F	F
F	T	F
F	F	F

Other Block Elements

Use a passthrough block like this for notes to production staff
NB: asciidoc comments do _not_ get passed to DocBook output by
default. If want to add a note for production staff, use a
passthrough block to embed it in a <remark> element instead.
By default, this will be preserved in the generated DocBook
(book.dcpsgen.xml) but will not render in your
PDF; if you would like <remark>s to be visible in your drafts PDFs
for convenience, send a request to toolsreq@oreilly.com and we'll
toggle a switch that will render them in red (as shown in the
pdf/demo_with_commentary.pdf version of this doc).
Here's a `<quote>` attributed to Benjamin Disraeli (by Wilfred Meynell, according to Frank Muir):

> Many thanks; I shall lose no time in reading it.

Finally, keep in mind that there may be situations where it's hard to get asciidoc to format something the way you want. Sometimes there's a trick to get around it; sometimes it's better to use a passthrough block to embed a bit of DocBook; and sometimes there may be a different formatting approach that will mesh better with our production systems. If you need to do something not illustrated in this chapter, please ask *toolsreq@oreilly.com*...

Extras/Experiments

This is another chapter, with some more esoteric/experimental things...

Go back to the first chapter for general asciidoc markup demo; below are some more esoteric/experimental things...

Yet More on Code Callouts

If you want to refer to the same explanation from more than one line of code, please *do not* use the built-in mechanism (Example 2-1), which does not conform to house style. Instead, use either Example 2-2 or Example 2-3.

Example 2-1. Built-in Mechanism (Vertical Layout Bad)

```
10/17/97   9:04        <DIR>    bin
10/16/97  14:11        <DIR>    DOS              ❶
10/16/97  14:40        <DIR>    Program Files
10/16/97  14:46        <DIR>    TEMP
10/17/97   9:04        <DIR>    tmp
10/16/97  14:37        <DIR>    WINNT
10/16/97  14:25            119  AUTOEXEC.BAT     ❷
 2/13/94   6:21         54,619  COMMAND.COM      ❸
10/16/97  14:25            115  CONFIG.SYS       ❹
11/16/97  17:17     61,865,984  pagefile.sys
 2/13/94   6:21          9,349  WINA20.386       ❺
```

❶ This directory holds MS-DOS.

❷ System startup code for DOS. **[Too much whitespace; confusing to readers]**

❸

❹

❺ Some sort of Windows 3.1 hack.

Example 2-2 uses the `<coref>` element to refer to the same `<callout>` from multiple code lines.

Example 2-2. Alternate Approach 1: Repeat Icon in Code

```
10/17/97   9:04       <DIR>    bin
10/16/97  14:11       <DIR>    DOS        ❶
10/16/97  14:40       <DIR>    Program Files
10/16/97  14:46       <DIR>    TEMP
10/17/97   9:04       <DIR>    tmp
10/16/97  14:37       <DIR>    WINNT
10/16/97  14:25             119  AUTOEXEC.BAT   ❷
 2/13/94   6:21          54,619  COMMAND.COM    ❷
10/16/97  14:25             115  CONFIG.SYS     ❷
11/16/97  17:17      61,865,984  pagefile.sys
 2/13/94   6:21           9,349  WINA20.386     ❸
```

❶ This directory holds MS-DOS.

❷ System startup code for DOS.

❸ Some sort of Windows 3.1 hack.

Example 2-3 uses a different mechanism for several code lines to point to the same
`<callout>`. In this case, each one gets a uniquely numbered icon. This is done by placing
multiple values in a single `<callout arearefs=…>` while duplicating the `@linkends` value
in the corresponding `<co>` elements in code. Note also the use of `<?dbfo…>` markup below
the `<calloutlist>` opener; this adjusts the alignment for side-by-side icons.

Example 2-3. Alternate Approach 2: Unique Icons; Align Side-by-Side in Explanation

```
10/17/97   9:04       <DIR>    bin
10/16/97  14:11       <DIR>    DOS        ❶
10/16/97  14:40       <DIR>    Program Files
10/16/97  14:46       <DIR>    TEMP
10/17/97   9:04       <DIR>    tmp
10/16/97  14:37       <DIR>    WINNT
10/16/97  14:25             119  AUTOEXEC.BAT   ❷
 2/13/94   6:21          54,619  COMMAND.COM    ❸
10/16/97  14:25             115  CONFIG.SYS
11/16/97  17:17      61,865,984  pagefile.sys
 2/13/94   6:21           9,349  WINA20.386     ❹
```

❶   This directory holds MS-DOS.

❷❸ System startup code for DOS.

❹   Some sort of Windows 3.1 hack.

Experiments

Literal Blocks (asciidoc terminology)

Following the general asciidoc cheatsheet/samples; many of these don't behave under our toolchain (or relative to house style), so please ask before using anything here...

ORMTODO: "literal" becomes <literallayout class="monospaced">

```
Consul *necessitatibus* per id,
consetetur, eu pro everti postulant
homero verear ea mea, qui.
```

ORMTODO: "verse" becomes <literallayout>, with inline markup

Consul **necessitatibus** per id,
consetetur, eu pro everti postulant
homero verear ea mea, qui.

ORMTODO: indented block becomes <literallayout class="monospaced">

```
Consul *necessitatibus* per id,
consetetur, eu pro everti postulant
homero verear ea mea, qui.
```

ORMTODO: dotted-delimited becomes <literallayout class="monospaced">

```
Consul *necessitatibus* per id,
consetetur, eu pro everti postulant
homero verear ea mea, qui.
```

ORMTODO: "verse" plus underscore-delimited becomes <blockquote><literallayout>

To see a world in a grain of sand,
And a heaven in a wild flower,
Hold infinity in the palm of your hand,
And eternity in an hour.

— William Blake *from Auguries of Innocence*