Table of Contents

1. Introduction 1
2. Setting up your editing environment 2
    2.1. Preliminaries 2
    2.2. Textmate 3
3. Ubuntu Quick Start 4
4. Writing in AsciiDoc 5
    4.1. Editing files 5
    4.2. Generating a table of contents (TOC) 8
    4.3. Generating a PDF on your local machine 9
    4.4. Save your document as an html file 9
5. Publishing using the ORM workflow 9
    5.1. Checkout your subversion repo 10
    5.2. Move your content into the SVN repo 10
    5.3. Add the files to the SVN repo 11
    5.4. Commit your files with a special trigger in the commit message 11
    5.5. Update the repo to download the new PDF and log files 11
    5.6. Review changes and fix them in your working directory 11
6. Useful links 11
7. Using git 12
    7.1. Creating a Repository 12
    7.2. Adding Files 12
    7.3. More commit options 12
    7.4. Branches 12
    7.5. Merging 13
    7.6. Comparing versions of files 13
    7.7. Incorporating changes from contributors 13

1. Introduction

This document describes the workflow for Maker Press, an experimental system for building high quality, commercial documentation for projects for MAKE. Here’s the basic idea:

- Editors sign an author to document a project (or multiple projects)
- The author develops the content in AsciiDoc, a text document format for writing documentation, articles, books, ebooks, slideshows, web pages, man pages and blogs.
- The author, editor, and any interested contributor (more on this in a second) develop and maintain the content in git and github.
- When the content is ready, the author or editor moves it into the O’Reilly publishing workflow so that it can be pushed out through out digital channels

The goal of the project is to create a lightweight way for people to:

- Create and distribute content through O’Reilly

- Invite participation from the community
- Keep material up to date more quickly
- Publish material on more niche topics where a book may not make sense

Contributing to Maker Press

Although this document was developed with authors and editors in mind, anyone can contribute to a Maker Press project by:

- Setting up a github account
- Forking the project into your own repository
- Pulling down the repository into your own git repository
- Making some changes / edits
- Pushing your changes back up into your repo
- Sending the a pull request to let the author know you've made some changes

The author or editor will review your suggestions and incorporate them as he or she sees fit.

This document walks you through the details of the authoring process. It's mostly focused on tools and workflows — see the Section 6, "Useful links" for more information about formatting content in AsciiDoc, writing for O'Reilly, using git, and various other related topics.

2. Setting up your editing environment

This doc describes how to set up an environment on a Mac to write for Maker Press. (Or, I should say, it's my running notes for it.) Depending on your system, you might have a bit of work to do to get things up and going. So, take this document with a grain of salt.

We'll need to do a few things to get going:

- Install a package manager such as MacPorts or Homebrew to facilitate installing and managing additional software package
- Install git, a distributed version-control system, and asciidoc, which is a package for formatting text
- Install TextMate, as well as a few bundles

The following sections explain this in more detail.

2.1. Preliminaries

You will need the XCode development tools and X11 (which comes on some Macs). Next, it is highly recommended that you install a package manager to automate further software installation. Whereas many Linux distributions have a default package manager (such as apt or yum), the Mac does not, but there are at least two reasonable package managers you can install. The first is MacPorts [http://www.macports.org/install.php], and the second is Homebrew [http://mxcl.github.com/homebrew/]; read about each online to decide which better suits your tastes. (Don’t use Fink.) For the moment, we will stick with MacPorts.

http://tedwise.com/2010/08/28/homebrew-vs-macports/

One annoyance I had on MacPorts is that it lists the OS versions by name, rather than number, which I found annoying. You can find your OS version by clicking the Apple at the top left corner and then selecting "About this Mac" and then find the corresponding name using the list below (from Wikipedia):

```
* Version 10.0: "Cheetah"
* Version 10.1: "Puma"
* Version 10.2: "Jaguar"
* Version 10.3: "Panther"
* Version 10.4: "Tiger"
* Version 10.5: "Leopard"
* Version 10.6: "Snow Leopard"
* Version 10.7: "Lion"
```

Then make sure your PATH is correct in ~/.profile. So here are the two lines you’ll want (updating for your username):

```
PATH=$PATH:/opt/local/bin:/opt/local/sbin:/Applications:/Applications/Utilities:/usr/loca
export PATH
```

Git

If you don’t already have git, you can use MacPorts to install it. Just drop into the terminal and type:

```
$ sudo port install git
```

It should fire up and install with no problems.

asciidoc, a2x (8.6.4)

Asciidoc refers to two different things: a wiki-like **markup language** you can write in, and the various **tools** that convert that markup into various other formats. This section describes how to set up the tools. You can probably already see the punchline coming, but here’s how you set up the asciidoc tools on your system:

```
$sudo port install asciidoc
```

The downside of this is that this step takes a **very** long time. So, be prepared to run it and then go out and get some coffee. Or, two coffees.

2.2. Textmate

You can edit your documents in whatever editor you like — vi, emacs, or whatever. I’ve been using TextMate [http://macromates.com/], an editor for the Mac that is popular in the developer community.

It costs about $60 U.S., but it’s got some addictive features that make it worth the price. Plus, you can try it our for free for 30 days.

On of the coolest features of TextMate is that it offers a *bundles* — collections of macros, commands, snippets, drag commands, templates, preferences, and language grammars — that make development much quicker. There are two main bundles I’ve been using: asciidoc and git.

Install the asciidoc bundle

The AsciiDoc bundle [https://github.com/zuckschwerdt/asciidoc.tmbundle] makes it much easier to work with AsciiDoc in textmate by offering things like automatic previews, source highlighting, and so forth. Here’s what you do:

```
mkdir -p /Library/Application\ Support/TextMate/Bundles
cd ~/"Library/Application Support/TextMate/Bundles/"
git clone git://github.com/zuckschwerdt/asciidoc.tmbundle.git "AsciiDoc.tmbundle"
osascript -e 'tell app "TextMate" to reload bundles'
```

Once the bundle is installed, you’re asciidoc markup will have be al the color coded goodness you’ve come to expect. One note: you have to give the files a ".asc" extension for the color coding to happen.

Install the git bundle

The git bundle [http://gitorious.org/git-tmbundle] allows you to save stuff in git right from within TextMate. Not sure what this buys me yet, but here are the commands:

```
mkdir -p /Library/Application\ Support/TextMate/Bundles
cd !$
git clone http://git.gitorious.org/git-tmbundle/mainline.git Git.tmbundle
osascript -e 'tell app "TextMate" to reload bundles'
```

You’ll need to add a path variable to tell TextMate where git is installed. First, locate git:

```
$which git
/usr/local/git/bin/git
```

Then go to "Preferences → Advanced" and click the "Shell Variables" tab. Then add a variable named "TM_GIT" and put in the path to git you found on the last step.

3. Ubuntu Quick Start

To get started as quickly as possible, find an Ubuntu installation and run the following commands:

```
sudo apt-get install git-core asciidoc fop
git clone https://github.com/odewahn/writingInAsciiDoc.git
cd /path/to/writingInAsciiDoc
./compile.sh
evince book.pdf
```

These commands will install the git version control system along with tools for working with AsciiDoc and generating PDF output, clone the master git repository for this document from GitHub, compile a fresh copy of this document, and view it. Try editing one of the .asc files, recompiling, and viewing your changes!

4. Writing in AsciiDoc

As described earlier, AsciiDoc [http://www.methods.co.nz/asciidoc/index.html] is a text document format for writing notes, documentation, articles, books, ebooks, slideshows, web pages, man pages and blogs. The main advantage of AsciiDoc is that it plays well with DocBook, O’Reilly’s native content format, but is something that you can actually write in.

This Asciidoc cheat sheet [http://powerman.name/doc/asciidoc] covers most of what you’ll need to know. As you’ll see, AsciiDoc is most similar to wiki markup — if you can write a wikipedia article, then you’re pretty much 99% of the way there.

Formatting O’Reilly books in AscciDoc

You can get the official guide for writing a full O’Reilly book in AsciiDoc from our Subversion repository:

```
svn checkout --username "odewahn@oreilly.com"
 "https://prod.oreilly.com/internal/books/sandbox_odewahn_RT79726/current/tools/cookbo
```

I’ve also placed relevant files into the "samples" directory in this repository.

This document (still in R&D, BTW!) tells you how to do pretty much anything you’d like. I anticipate that Maker Press will only use a subset of the markup features, but this guide is there if you need it.

4.1. Editing files

The best way to get started with AsciiDoc and Maker Press is to pull down the repository for this document (how’s that for recursion!) and look out how the files are structured. Here’s how you can get the repo:

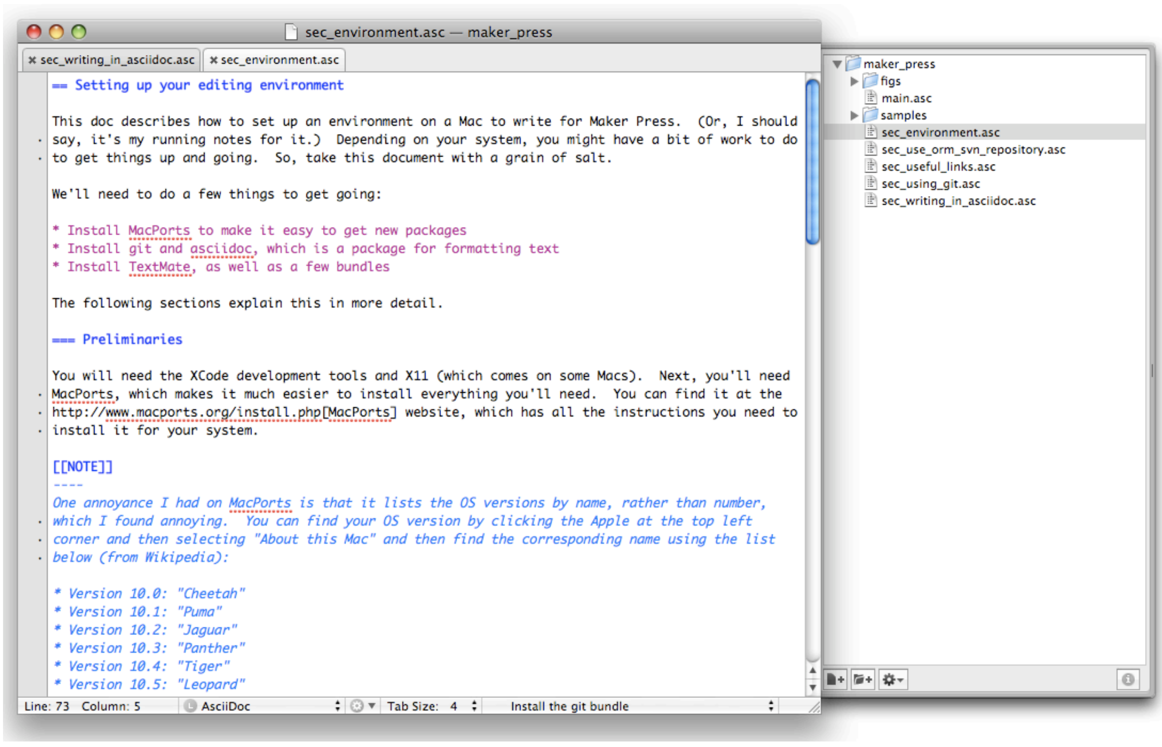
```
Need to have a link to the git repository for this doc
```

Once you have the files, you can open them in TextMate using the command:

```
mate .
```

This will open the editor and display the *project drawer*, which is a navigation tree that you can use to move around between files. Use the project drawer to open the file called *sec_environments.asc*, as shown in Figure 1, “Use TextMate’s bundle editor to add in the path”.

Figure 1. Use TextMate’s bundle editor to add in the path

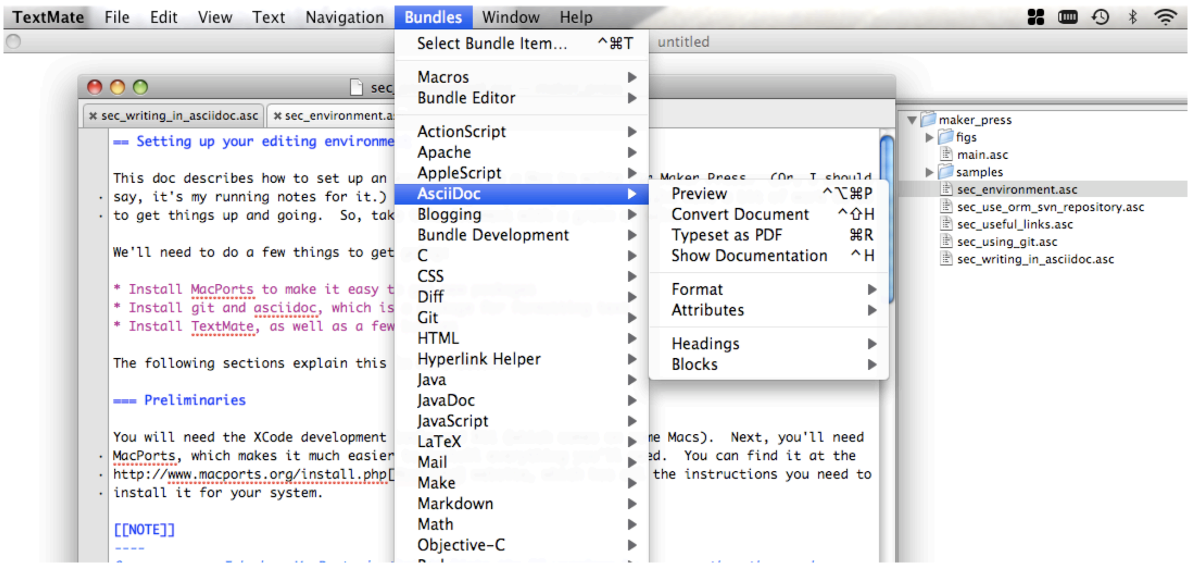


```
== Setting up your editing environment

This doc describes how to set up an environment on a Mac to write for Maker Press.  (Or, I should
say, it's my running notes for it.)  Depending on your system, you might have a bit of work to do
to get things up and going.  So, take this document with a grain of salt.

We'll need to do a few things to get going:

* Install MacPorts to make it easy to get new packages
* Install git and asciidoc, which is a package for formatting text
* Install TextMate, as well as a few bundles

The following sections explain this in more detail.

=== Preliminaries

You will need the XCode development tools and X11 (which comes on some Macs).  Next, you'll need
MacPorts, which makes it much easier to install everything you'll need.  You can find it at the
http://www.macports.org/install.php[MacPorts] website, which has all the instructions you need to
install it for your system.

[[NOTE]]
----
One annoyance I had on MacPorts is that it lists the OS versions by name, rather than number,
which I found annoying.  You can find your OS version by clicking the Apple at the top left
corner and then selecting "About this Mac" and then find the corresponding name using the list
below (from Wikipedia):

* Version 10.0: "Cheetah"
* Version 10.1: "Puma"
* Version 10.2: "Jaguar"
* Version 10.3: "Panther"
* Version 10.4: "Tiger"
* Version 10.5: "Leopard"
```

If you’ve done any wiki markup, this should look pretty familiar. Also, note how the various AsciiDoc elements are all nicely color coded — this is thanks to the AsciiDoc bundle we installed earlier.

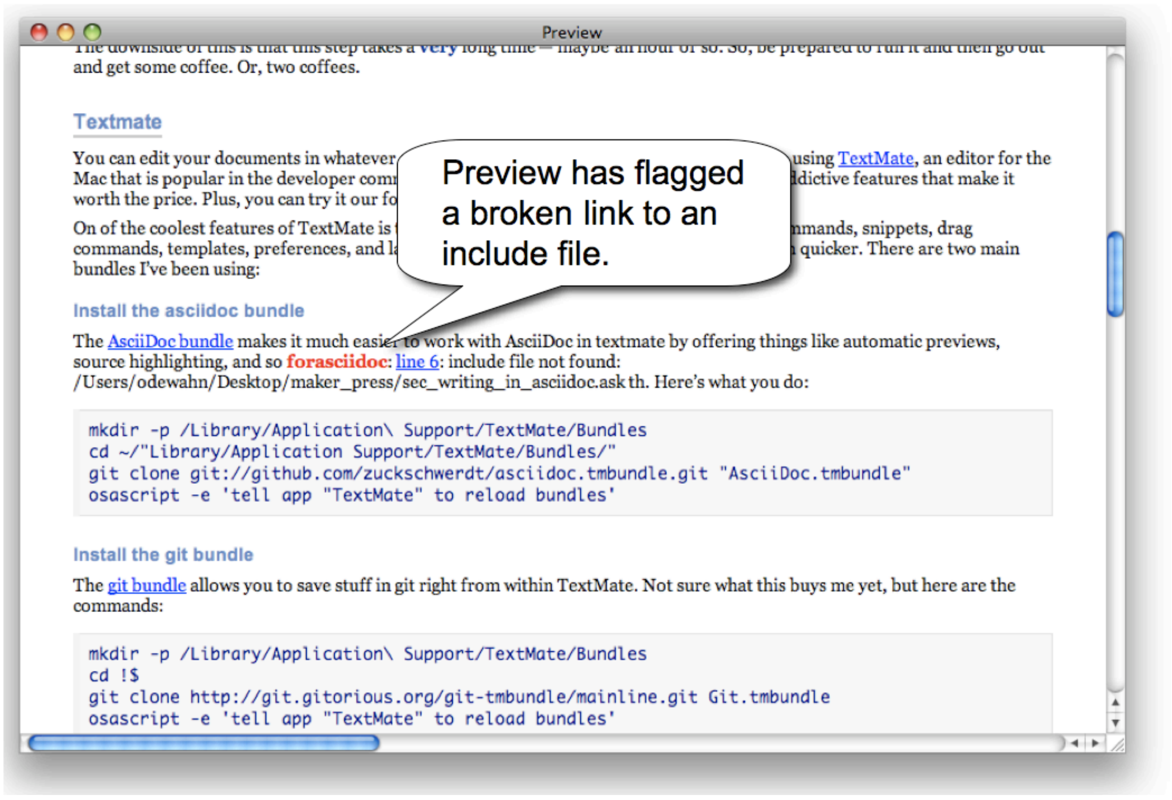
Muck around in there for a while and make a few changes (remember — this is all in your local git repository, so you can’t hurt anything). When you’re ready, you can use the "Bundles → AsciiDoc → Preview" to see how your document will look when it is rendered in HTML, as shown in Figure 2, “Generating a preview”.

Figure 2. Generating a preview



The preview will convert the AsciiDoc into HTML as best it can. If preview identifies any errors, it will flag in red. For example, Figure 3, “A broken link to an include file shows up as a red hyperlink” shows how preview has found a broken link to an include file.

Figure 3. A broken link to an include file shows up as a red hyperlink



The downside of this is that this step takes a very long time — maybe an hour or so. So, be prepared to run it and then go out and get some coffee. Or, two coffees.

Textmate

You can edit your documents in whatever … using TextMate, an editor for the Mac that is popular in the developer com… …ddictive features that make it worth the price. Plus, you can try it our fo…

On of the coolest features of TextMate is … …mmands, snippets, drag commands, templates, preferences, and l… …quicker. There are two main bundles I've been using:

Install the asciidoc bundle

The AsciiDoc bundle makes it much easier to work with AsciiDoc in textmate by offering things like automatic previews, source highlighting, and so forasciidoc: line 6: include file not found: /Users/odewahn/Desktop/maker_press/sec_writing_in_asciidoc.ask th. Here's what you do:

```
mkdir -p /Library/Application\ Support/TextMate/Bundles
cd ~/"Library/Application Support/TextMate/Bundles/"
git clone git://github.com/zuckschwerdt/asciidoc.tmbundle.git "AsciiDoc.tmbundle"
osascript -e 'tell app "TextMate" to reload bundles'
```

Install the git bundle

The git bundle allows you to save stuff in git right from within TextMate. Not sure what this buys me yet, but here are the commands:

```
mkdir -p /Library/Application\ Support/TextMate/Bundles
cd !$
git clone http://git.gitorious.org/git-tmbundle/mainline.git Git.tmbundle
osascript -e 'tell app "TextMate" to reload bundles'
```

Most times, though, you'll have to scan the document and make sure it looks right, just like you would a wiki page. The advantage of using Preview to clean up your errors is that it will save you time when you try to render the document in the O'Reilly toolchain.

If you have trouble with Preview, try this first

On my system, there was a weird problem when I tried to run the bundle commands — it couldn't locate the asciidoc command. There's probably some environment variable or the other that I need to set, but here's what I did to fix it. First, I located where MacPorts installed asciidoc, which you can do by dropping into a terminal and typing this:

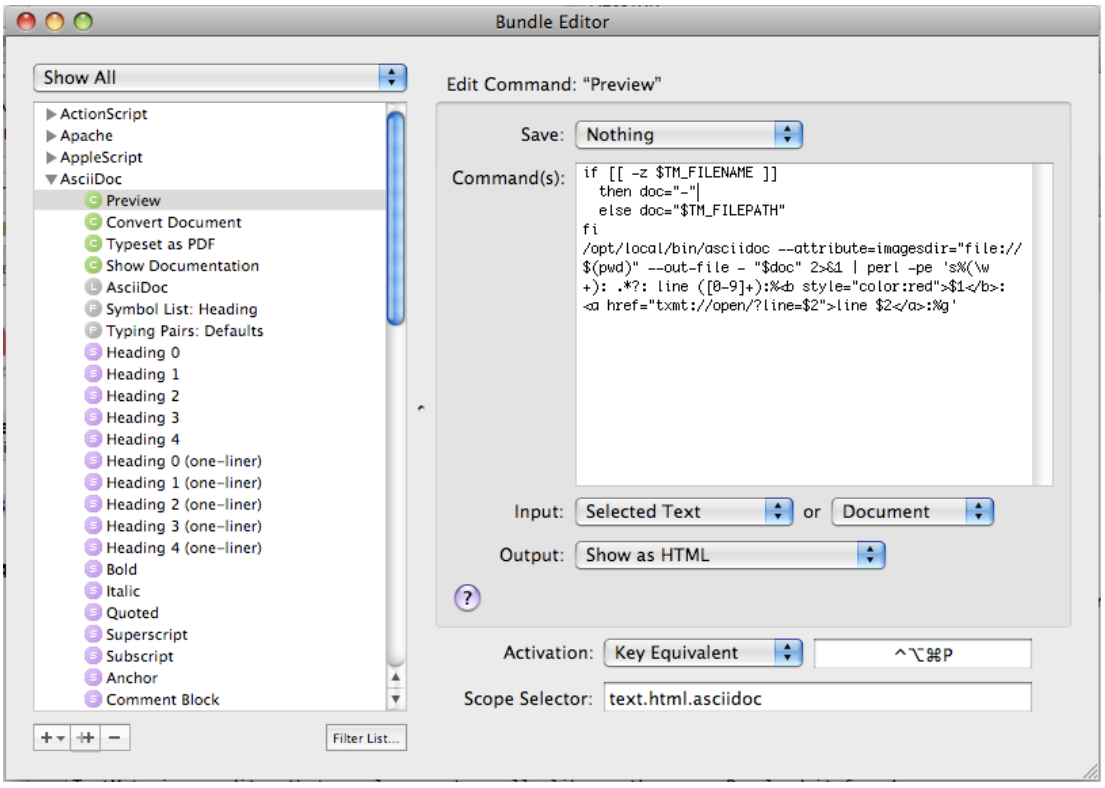
```
$ which asciidoc

/opt/local/bin/asciidoc
```

Then go into the bundle editor (Bundles → Bundle Editor → Show Bundle Editor) and select the asciidoc bundle. You can then go into the commands (the have "c" in fromt of them) and add this path in front of everyplace where asciidoc appears. For example, here's the revised version of the "Preview" command:

```
if [[ -z $TM_FILENAME ]]
  then doc="-"
  else doc="$TM_FILEPATH"
fi
/opt/local/bin/asciidoc --attribute=imagesdir="file://$(pwd)" --out-file - "$doc" 2>&1
```

It should look something like Figure 4, "Use TextMate's bundle editor to add in the path" in Textmate.

Figure 4. Use TextMate’s bundle editor to add in the path



Edit Command: “Preview”

Save: Nothing

Command(s):

```
if [[ -z $TM_FILENAME ]]
  then doc="-"
  else doc="$TM_FILEPATH"
fi
/opt/local/bin/asciidoc --attribute=imagesdir="file://
$(pwd)" --out-file - "$doc" 2>&1 | perl -pe 's%(\w
+): .*?: line ([0-9]+):%<b style="color:red">$1</b>:
<a href="txmt://open/?line=$2">line $2</a>:%g'
```

Input: Selected Text or Document

Output: Show as HTML

Activation: Key Equivalent ^⌥⌘P

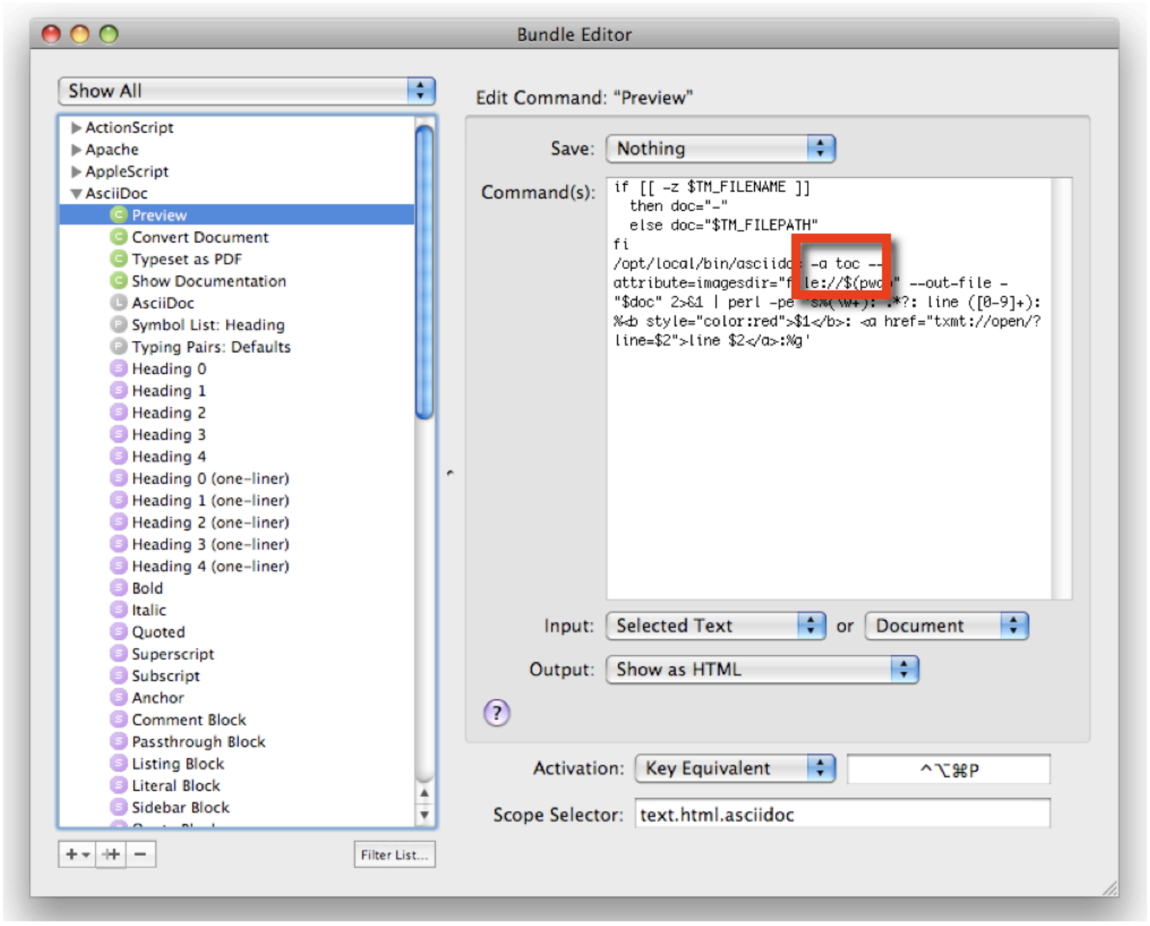
Scope Selector: text.html.asciidoc

4.2. Generating a table of contents (TOC)

You should review your TOC often to make sure that the document is structured correctly. It’s really easy to generate this automatically — all you have to do is use the bundle editor to add the option "-a toc" to the asciidoc command.

Open up the bundle editor (Bundles → Bundle Editor), select the AsciiDoc bundle, and then click the "Preview" command. Then, add the put "-a toc" after the AsciiDoc command, as shown in Figure 5, “Automatically adding a TOC to Preview”

Figure 5. Automatically adding a TOC to Preview



Edit Command: “Preview”

Save: Nothing

Command(s):

```
if [[ -z $TM_FILENAME ]]
  then doc="-"
  else doc="$TM_FILEPATH"
fi
/opt/local/bin/asciidoc -a toc --
attribute=imagesdir="file://$(pwd)" --out-file -
"$doc" 2>&1 | perl -pe 's%(\w+): .*?: line ([0-9]+):
%<b style="color:red">$1</b>: <a href="txmt://open/?
line=$2">line $2</a>:%g'
```

Input: Selected Text or Document

Output: Show as HTML

Activation: Key Equivalent ^⌥⌘P

Scope Selector: text.html.asciidoc

4.3. Generating a PDF on your local machine

If you just want to generate something that you can use for copyediting or sharing with your friends, the following command will generate a PDF using the default stylesheets that come with AsciiDoc:

```
a2x -v --fop  --no-xmllint book.asc
```

This **--no-xmllint** option turns off the XML validator.

4.4. Save your document as an html file

AsciiDoc has options to create stand-alone documents containing embedded images, stylesheets and scripts. The following AsciiDoc command creates a single file containing embedded images, CSS stylesheets, and JavaScript (for table of contents and footnotes):

```
$ asciidoc -a data-uri -a icons -a toc -a max-width=55em article.txt
```

5. Publishing using the ORM workflow

O’Reilly’s "official" publishing workflow is done is developed in DocBook and managed in Subversion (SVN), another content management system. Your editor will provide you with a SVN repository for your content once you get a contract. To "publish" your work through O'Reilly you'll need to get your content into this system.

It's not so bad, actually. The basic workflow is:

- Checkout your subversion repo
- Move your content into the SVN repo
- Commit your files with a special trigger in the commit message
- Update the repo to download the new PDF and log files
- Review changes and fix them in your working directory

The repo is set up to use a two-stage SVN commit hook, which first tries to convert asciidoc to DocBook (using ORM-customized config files) and then builds a PDF from that (assuming it's succeeded ingenerating valid DocBook) in standard Animal style. Once you're ready, you can also push the content out into our various e-channels.

The following sections cover these steps in detail.

5.1. Checkout your subversion repo

You'll get an official SVN repository when you sign the contract. Your editor will help you with the gory details like getting a password, and so forth.

Checking out the files looks something like this:

```
svn checkout --username "odewahn@oreilly.com" "https://prod.oreilly.com/internal/books/sa
```

5.2. Move your content into the SVN repo

Using "git svn"

You might reasonably ask, "Hey, why do I have to copy this stuff? Can't I just use git svn" by doing something like this (see Chapter 8 from *Pro Git*):

```
git svn clone --username "odewahn@oreilly.com" "https://prod.oreilly.com/internal/book
```

Good idea, but I don't think this is going to work for us using our standard workflow. The main sticking point seems to be that git wants to make a complete history of all the changes that have happened on the repository, so it goes through several hundred thousand (literally!) different revisions. This takes a **long** time to complete — it has to go through almost 300K revisions. Also, it seems from reading around that SVN can have touble keeping up with the more complex histories of git projects.

But, I wonder if there is a way to speed it up? Maybe we could create a separate repository for just these projects?

5.3. Add the files to the SVN repo

5.4. Commit your files with a special trigger in the commit message

You can generate a PDF of your book by including the string "orm:asciidoc" in the commit message when you commit the repository. For example:

```
$ svn commit -m'Made some really important changes to Chapter 3; orm:asciidoc'
```

This will kick off the process to convert the asciidoc files into DocBook.

5.5. Update the repo to download the new PDF and log files

To retrieve the results, run the command "svn up" (update) copy about 5–10 minutes after committing your files. If all goes well, the process will generate the following files:

- pdf/book.dcpsgen.xml.pdf (the PDF, assuming the process succeeded)
- book.dcpsgen.xml (generated DocBook)
- .dcps.a2xlog (log from asciidoc#DocBook conversion)
- pdf/.dcpsgen.buildlog (log from PDF build of generated DocBook)

5.6. Review changes and fix them in your working directory

If you don’t see a fresh PDF when you *svn up* after a few minutes, take a look at the logfiles above for errors. **Be sure to fix any problems in your working git repository, not the temporary SVN repository**. Then, give it another shot.

6. Useful links

These are various linke that I’ve found that help explain AsciiDoc, O’Reilly processes, and other stuff:

- Matt Neuburg article on how to write a book using asciidoc [http://www.apeth.net/matt/iosbooktoolchain.html]. Nice article that explains how to do lots of good stuff.
- Asciidoc cheat sheet [http://powerman.name/doc/asciidoc]. Pretty much what it says — this shows you the markup in a no-nonsense kind of way.
- O’Reilly Production Guidelines [https://prod.oreilly.com/external/tools/docbook/docs/authoring/docbookguidelines_web.pdf]. Describes how to use the main O’Reilly subversion repository.

- Looks like a pretty good intor to the basic features of git [http://ryanflorence.com/git-for-beginners/]

7. Using git

These are just some random notes about git — I need to add more in this section later as we start to develop some ideas about how to best use it. Most of this is cribbed from Scott Chacon’s git in one hour [http://www.youtube.com/watch?v=OFkgSjRnay4] webcast.

7.1. Creating a Repository

There are two ways to create a repository:

- clone it from an exitsing repo
- run "git commit -m *initial commit*"
- use "git status" to get a status of working changes
- use "git log" to get a log file of all commits
- "git show" shows what the last commit was

7.2. Adding Files

Next, you need to add files using "git add <filename>"

7.3. More commit options

Then you use "git commit -m *some message*" to commit the changes You can also used "git add <filename>" to stage your commits, which means that you can commit in different groups. this is helpful if you want to create a clear audit trail of what you were doing.

- To just commit everything, you can use "git commit -a"
- To commit stuff in steps, use "git add <filename>"

7.4. Branches

git encourages you to do lots of little stuff inside branches, not just big stuff. For example, you might create a new branch to resolve a ticket, fix it in the branch, and then merge it back in later. This is called a "topic branch".

The command to see a branch is called "git branch".

You can make a new branch using "git branch test" to create a new branch. To work in the branch, you do "git checkout test".

You can make any changes you want in the directory, and all changes happen in just that branch. So, you can add files, make new changes, or whatever, and they’ll happen in only that one place. Changes are saved at each commit.

If you want to move back to another branch, all you have to do it check it back out and the files will look just as they did at the last commit point.

You can view all the activity in the branches using "gitk". This will pop up a little browser that you can use to go through all the changes on the branch you’re on. You can use "gitk --all" to see the changes across all branches.

7.5. Merging

I need to review this section, but merging basically collapses two branches into a single branch. You can then remove the topic branches using "git branch -d branch name" (double check this syntax)

You do something to add a remote server

You can then use "git fetch" to pull off some remote copy of the repository, but moves it into your repo as a separate branch. This way you can inspect what someone else is doing, but not mess up your own work, which is really pretty cool. You then merge that brach back in as you see fit.

7.6. Comparing versions of files

Use "git diff" to see the differences between the current version of the repo and the new verion. You can use "git diff | mate" to see the diff visualized in textmate, which is pretty cool.

7.7. Incorporating changes from contributors