# logo_zw.jpg

# INC hybrid workflow how-to

***Test panel, 12 September 2014***

**Schedule**

13.30 - 13.45: Welcome, introduction to the workflow and the test panel by Michael and Miriam

13.45 - 15.00: Start testing the different steps / roles

15.00 - 15.30: Feedback and discussion. Answer questionnaire on google doc.

15.30 - End

The hybrid workflow described and visualized below is developed by the Institute of Network Cultures within the Digital Publishing Toolkit. The research into this workflow was aimed at making the transition from a print-centered publication process towards a digital and print (hybrid) publication process. Instead of ‘adding’ the digital publication at the end of an existing workflow, based on the printed book as end result, as is done often by publishers, the workflow should be adjusted and made efficient and practical towards hybrid publishing in an earlier stage. The description below starts at a point which in reality is not the beginning of the publication trajectory, namely when an author hands in the definitive manuscript - so after the editing and rewriting process has been rounded off.

NB: An important step preceding the publication trajectory lies in the formulation of the in-house style guide, where authors and editors can find the requirements for the manuscript. This style guide must be adapted according to the hybrid workflow as an absolute start. When interested, you can receive the INC style guide (send an email to Miriam).

NB: Should the author already be working in Markdown format, this will change the workflow. However, in our experience manuscripts are mainly produced in Microsoft Word and delivered in .doc or .docx.

### Before you start:

Copy the folder ‘INC test panel’ from the USB stick to your Documents folder. In the folder there is a subfolder called ‘test docs’, one for each role - this contains the documents we’re working with; and a subfolder called software - this has the required software tools.

1. Open the subfolder ‘software’ and install Pandoc by double clicking on the file. Pandoc is working in the so-called command line mode and not in a user interface environment. Hence you don’t see an icon.

Windows: To start pandoc type cmd in the RUN (also called ‘search programs and files’ in the start panel which can be found under the MS window icon down in the toolbar), this will enable you to start the command mode. You get a white/black window saying C:\user\yourusername>. There you type pandoc (enter) and the same line reappears, waiting for pandoc input (see further below).

Mac: To use pandoc open the Terminal from your Applications folder, or through the search bar in the top right of your screen. Pandoc will be used to convert files in the steps below.

2. Install a Markdown-editor. For Mac, use Mou.zip; for Windows, use markdownpad2-setup.exe.

3. Install a html editor. For Mac, use Sublime Text 2.0.2.dmg, for Windows use Sublime Text 2.0.2 Setup.exe.

4. Designer: Go to the test docs folder and copy HTMLtoXML\_InDesignFlavour-master to your documents folder. We’ll call this the repository folder.

# Who: Editor

**What:** You receive a digital document containing the definitive manuscript from the author. Open the test docs-folder. The file to start with (the one an author has supposedly sent to you) is Astrid\_Mager\_def.doc.

STEP 1: Open the file in Word.

STEP 2: Convert the document to docx-format by saving it as Astrid\_Mager\_def.docx.

STEP 3: Apply header styling in the Word-document. Use header 1 for title and author, header 2 for section title, header 3 for level below, et cetera.

STEP 4: Save.

STEP 5: Convert the docx to Markdown using pandoc:

* Pandoc is a command-line tool. There is no graphic user interface. So, to use it, you’ll need to open a terminal window (see above).
* Create a subfolder in your Documents folder called ‘pandoc-test’. For Windows, put this under C:\users\yourname\documents (in many cases this is the default directory).
* Put the docx-document you want to convert in this folder called pandoc-test. In this case: Astrid\_Mager\_def.docx.
* Go to the terminal and type cd Documents. This means the terminal will ‘change directory’ to the Documents folder.
* Now type cd pandoc-test. The terminal will change directory to the folder within the Documents folder called pandoc-test. Now you can work with the documents in there.
* Type ls (Mac) or dir (Windows) to get a list of files in the current folder. The Astrid\_Mager\_def.docx should be listed.
* To convert the file from docx to markdown, type the following into the terminal: pandoc Astrid\_Mager\_def.docx -f docx -t markdown -s -o Astrid\_Mager\_def.md
* The filename Astrid\_Mager\_def.docx tells pandoc which file to convert, -f docx -t markdown, so from docx to markdown. The -s option says to create a ‘standalone’ file, with a header and footer, not just a fragment. And the -o Astrid\_Mager\_def.md says to put the output in a file named Astrid\_Mager\_def.md. (Note: in Mac you can copy-paste the command, in Windows you can’t copy-paste.)
* Check that the file was created by typing ls or dir again. You should see Astrid\_Mager\_def.md.

STEP 6: Now open this in your Markdown-editor. Open the Markdown editor that you installed (Mou or Markdownpad), navigate to the pandoc-test directory and open the file we just created using pandoc, Astrid\_Mager\_def.md. The file will now be opened in the markdown editor. You see two panels. The left panel is the markdown format which we use to work in, the right panel is the rendering of the coded panel into a user-readable-form. (You can also try to type ‘open Astrid\_Mager\_def.md’ in the terminal to open the file.)

STEP 7: Check the Markdown file after conversion: Are the headers still marked, is there no funny formatting in the text, are the blockquotes and italics preserved, for example in the references?

STEP 8: Now open the file Template\_INC\_essay.md which is in the folder with documents from the USB stick. In this template you can see how formatting should be applied in a correct way and which metadata to add.

STEP 9: Copy the text in Astrid\_Mager\_def.md as formatted in Markdown into the document template and save as Astrid\_Mager\_final.md.

STEP 10: Style the text according to the template: especially pay attention to title, author name, headings, references, notes, the table and the image.

Note: the best way is to open the template file in a separate window from the document.

Then copy the relevant article text in the coded template window, for example at the bottom (so not in the rendered text in the 2nd panel).

STEP 11: Add metadata information. You can find some of the metadata in the Proposal.rtf sent in by the author.

STEP 12: Save the final Markdown file into an archive folder. This is the document which can be send to the print designer and epub developer for them to work with.

STEP 13: Check format of the images: is the quality good enough for print?

STEP 14: Scale images to smaller size for e-book publication.

STEP 15: Store the images and send to designer and developer.

EXTRA STEP: Corrections

\* You’ve received the PDF design of the text! Now there are some corrections. Where do you make them? You can find the prints of the PDF in the (real life) folder you received.

# 

# Who: Print book designer

**What:** Receives the final manuscript from the editor in markdown format; works with [InDesign, install it if necessary](http://www.adobe.com/it/products/indesign.html).

STEP 1: Convert the Markdown file to html using Pandoc:

* Pandoc is a command-line tool. There is no graphic user interface. So, to use it, you’ll need to open a terminal window (see above).
* Create a subfolder in your Documents folder called ‘pandoc-test’. For Windows, put this under C:\users\yourname\documents (in many cases this is the default directory).
* Put the markdown file you want to convert in this folder called pandoc-test. In this case: Astrid\_Mager\_final.md.
* Go to the terminal and type cd Documents. This means the terminal will ‘change directory’ to the Documents folder.
* Now type cd pandoc-test. The terminal will change directory to the folder within the Documents folder called pandoc-test. Now you can work with the documents in there.
* Type ls (Mac) or dir (Windows) to get a list of files in the current folder. The Astrid\_Mager\_final.md should be listed.
* To convert the file from markdown to html, type the following into the terminal:

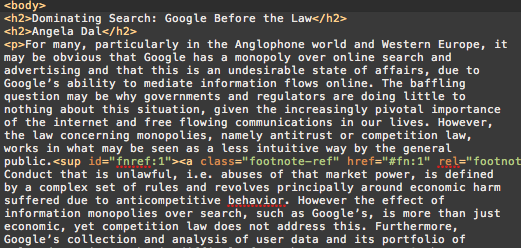
pandoc Astrid\_Mager\_final.md -f markdown -t html -s -o Astrid\_Mager\_final.html

* The filename Astrid\_Mager\_final.md tells pandoc which file to convert, -f markdown -t html, so from markdown to to html. The -s option says to create a ‘standalone’ file, with a header and footer, not just a fragment. And the -o Astrid\_Mager\_final.html says to put the output in a file named Astrid\_Mager\_final.html. (Note: in Mac you can copy-paste the command, in Windows you can’t copy-paste.)
* Check that the file was created by typing ls or dir again. You should see Astrid\_Mager\_final.html.

STEP 2. Convert the html to xml: this step is in development. We’ve written here in the manual how to do it, but for the testing panel today we want to proceed directly to the next step. Please continue on to STEP 3. Install footnote plugin in InDesign.

**a. Manual way:**

* Open the html document in Sublime, the html editor you have installed.
* Keep only the body: In order to be able to import the file into InDesign, it is necessary to delete the headers (eg.*<!DOCTYPE html>*) and the head tag. Also the upper html tag should be deleted. Finally you will have only body of your document, like in the following image. Don’t forget to remove the head tag at the bottom of the file.



* Remove redundant line breaks: unlike the way browsers render html, InDesign will preserve every line break that is in the document. Only the ones after headers or paragraphs are needed, so the other ones need to be substituted with a space. With python it is possible to do this as follows:
  + Open a new file in Sublime.
  + Copy-paste the script below in the file and save as ‘clean.py’ in the test docs folder where the html file is stored.

#!/usr/bin/env python  
# usage: clean.py myfile.xml   
  
import re, sys  
  
source = str(sys.argv[1])  
  
f = open(source, 'r+')  
doc = f.read()  
  
cleandoc = re.sub('(?<!\>)\n',' ', doc)  
print(cleandoc)

* + Open terminal and type: cd Documents and hit enter. Now type cd test\ docs\ -\ Designer/ (this is the folder where mager-mod.html is stored, you can also drag and drop).
  + Type in terminal: python clean.py mager-mod.html > Astrid\_Mager\_def.html

(The script will look for line breaks that are not preceded by a *>*, and will substitute them with a white space.)

* Change the extension of your html file to .xml.

**b. Automatic way (if it doesn’t work, use *mager-mod.xml)***

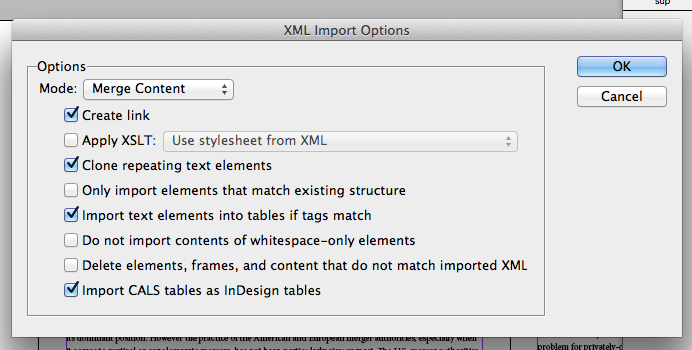
* Go to the terminal.
* Install [pip](https://pypi.python.org/pypi/pip) by typing: sudo easy\_install pip - the terminal will ask for your computer password.
* Install [Beautiful Soup](http://www.crummy.com/software/BeautifulSoup/) by typing: sudo pip install beautifulsoup4
* Go to the repository folder in terminal. In terminal: cd Documents and hit enter. Now type cd fromHTMLtoXML\_InDesignFlavour-master.
* Next In terminal: python main.py sources/mager-mod.html > Astrid\_Mager\_def.xml.
* Astrid\_Mager\_def.xml is created in the repository folder.

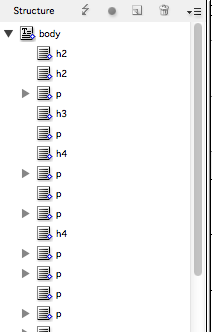
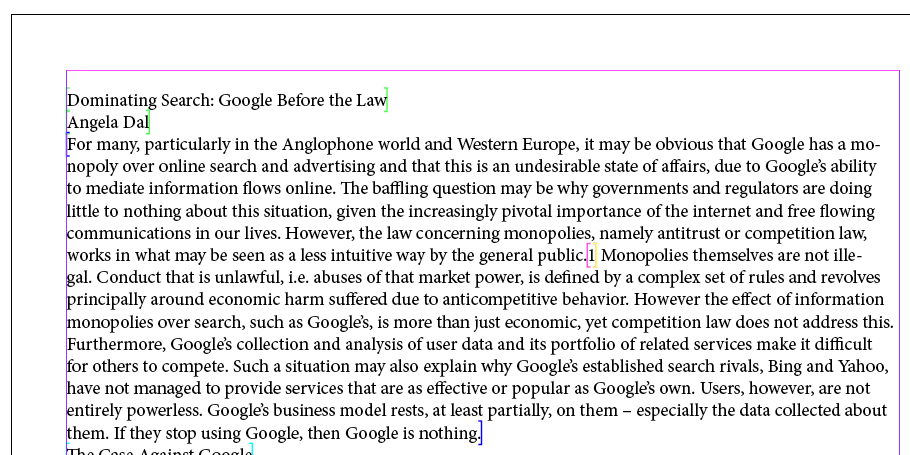
STEP 3. Install footnote plugin in InDesign:

* Copy the ReFoot\_mod.js script into InDesign.
* Open InDesign
* Go to *Window - Utilities - Scripts*
* Go to the scripts window and right click User and click Reveal in finder.
* Open another finder screen and go to the repository folder and copy-paste the ReFoot\_mod.js script in the Scripts Panel folder you’ve just opened via InDesign.
* ReFoot\_mod.js should now appear in the scripts window in InDesign under User.
* Restart InDesign.

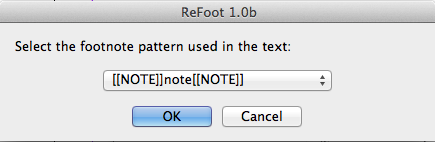
STEP 4. Import mager-mod.xml from your test docs folder into InDesign:

* Open InDesign and open a new file. Set it to A5 and facing pages.
* go to *File > Import XML* and choose the file [mager-mod.xml].



* By checking the *Create link* box you will be able to modify the content of the xml file and it will be automatically updated into InDesign. Check the Create link box.
* A window with the structure of your document will appear on the left. Click open the html icon on the left; inside there is an icon called body. By drag-and-dropping the body icon into a page, the content will appear.
* Run the ReFoot\_mod.js script.

Double click ReFoot\_mod.js in your scripts window in InDesign: *Window - Utilities - Scripts*. (You see that the endnotes are now turned into footnotes that are linked to the place in the text.) Select the footnote pattern in the text: [[Note]] note [[Note]]

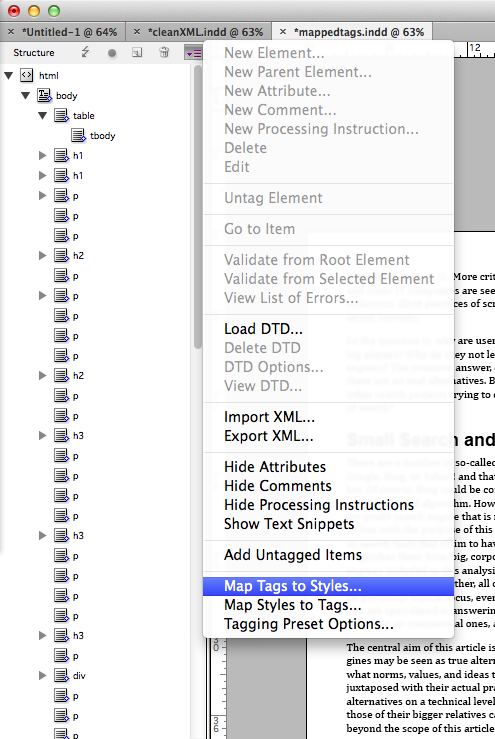


STEP 5. Remove metadata and other ‘extra information’ thatis above the title of the article.

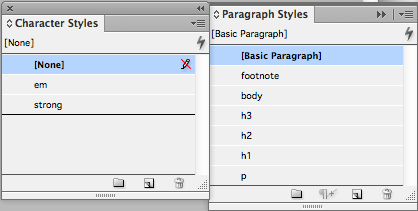
STEP 6. Check where the image should by searching for “fig”. Add the image manually (check resolution and CMYK) (We hope to develop a plugin that will add the images automatically)

STEP 7. Create paragraph and character styles based on the html tags.

* To check the html tags used in the document go to the structure window and open the drop down-menu and select [map tags to styles]. This gives you an overview of the tags.



* Create the paragraph styles and character styles for the tags shown in the image below. It make sense to keep the name of the style consistent with the relative HTML tag (eg. *p* to be used with *<p>* tag).



**footnote:** cambria 8 pt / 10 pt

**body:** cambria 10 pt / 12 pt

**h3:** arial 12pt / 14pt bold

**h2:** arial 16pt / 18pt bold

**h1:** arial 20pt / 18 pt bold

**p:** cambria 10 pt / 12 pt.

Indents and spacing:

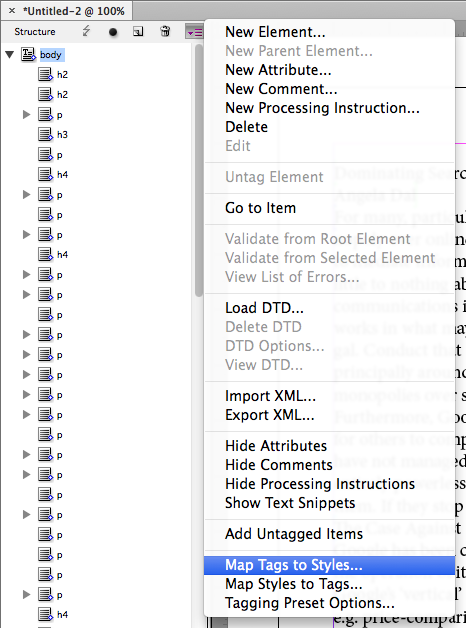
space before 0p6

space after 0p3

**em:** italic

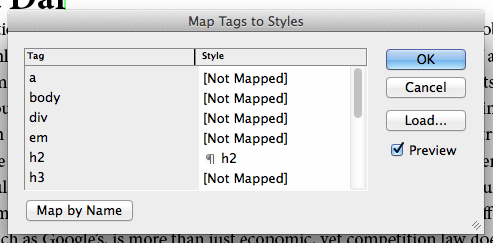
**strong:** bold

STEP 8. Map Tags to Styles



Connect the HTML tags to the relative paragraph and character styles. To do so click on *Map Tags to Styles* as illustrated in the image.

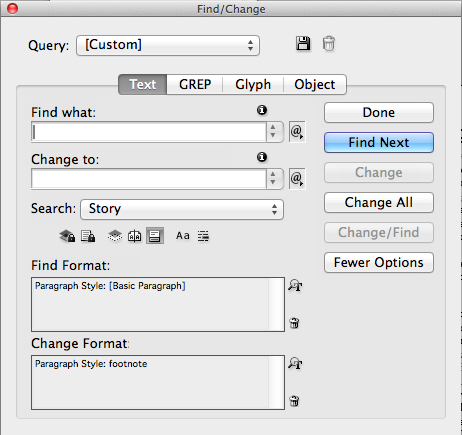
By clicking on *Map by Name* tags and styles will be automatically associated.



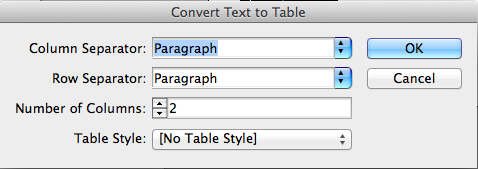
STEP 9. Add paragraph style to footnotes. (as the footnotes aren’t defined in the html they do not appear as a tag. You have to add a paragraph style to them manually)

NOTE: Before following this step make sure all other tags have their styles assigned to them. otherwise they will be styled similar to the footnotes.

* Go to Find/Change under [edit] (or press apple F) and select [Basic Paragraph] under Find Format and the paragraph style [footnote] under change format. Press: change all



STEP 10: In the scripts window (window-utilities-scripts) you can see the tag table. Double click the second table tag. You now jump to this part in the text. This is text that should be formatted as a table. Select it and go to Table and select Text to Table. Choose the following settings.

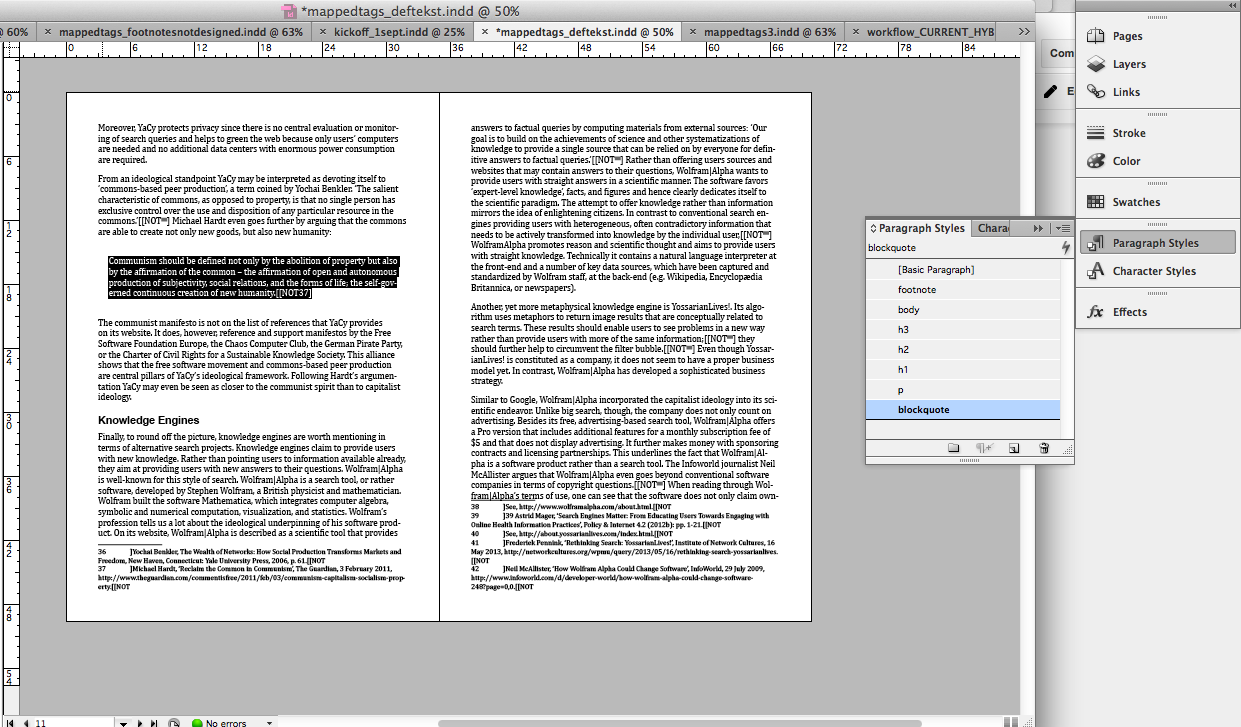


Delete the redundant white space in the table and format the first line as h3.

STEP 11: Make a paragraph style for blockquotes:

blockquote: based on p. indents and spacing: add left indent 1p0

In the scripts window (*Window-Utilities-Scripts*) you can see the tag blockquote. Double click it so you jump to it in the text. Format the block of text by choosing the blockquote paragraph style manually. Do this for all the blockquote tags.



* export to low-res PDF. this is the file you send to the editor for corrections.

EXTRA STEP: Corrections

\* You’ve received some corrections from the editor! Where do you make them? You can find the prints with corrections in the (real life) folder you received.

# **Who:** Developer. Create epub (& more) from markdown files using pandoc & make

STEP 1:

Receive markdown files from an editor (by mail or by pulling them from a git-managed repository)

git pull

STEP 2: Now we create a makefile. In a makefile you can create a “rule” to convert any markdown into HTML. The general format of a make rule is:

*# this is a comment*

*target : prerequisites*

*recipe*

So for example::

*sandwich : bread filling*

*take two slices of bread*

*put some filling on one, close with the other*

Means that to make a sandwich you need two things: bread and filling, and if you have those ingredients, you can actually make it by following the “recipe.”

As described in the steps for designers, pandoc can be used to convert markdown sources into HTML. The command looked like this:

pandoc Astrid\_Mager\_final.md -f markdown -t html -s -o Astrid\_Mager\_final.html

You could wrap this in a makefile rule by saying:

Astrid\_Mager\_final.html : Astrid\_Mager\_final.md

pandoc Astrid\_Mager\_final.md -f markdown -t html -s -o Astrid\_Mager\_final.html

IMPORTANT: The “recipe” lines in a makefile must be indented with tab characters (and not spaces). Otherwise it will give obscure error messages.. Save this text as a file named “makefile” in the same directory as the markdown source, and type the make command:

make

You should see:

pandoc Astrid\_Mager\_final.md -f markdown -t html -s -o Astrid\_Mager\_final.html

When no target is specified, make follows the first rule it finds.

If you type the make command again, you should see:

make: `Astrid\_Mager\_final.html' is up to date.

Make uses timestamps on files to decide whether or not rules need to be applied. If you change the markdown file, the make command will run pandoc again to update the HTML. Of course, it’s more useful to make a rule that can turn any markdown file into HTML. The % sign works like a wildcard to create a pattern that matches any filename:

%.html : %.md

pandoc $< -f markdown -t html -s -o $@

In this case the special symbols $< means the input, and $@ the output. This kind of rule is known as an implicit rule. If you replace the previous rule with the above and run make, it will complain:

make: \*\*\* No targets. Stop.

Implicit rules don’t automatically create a “target” so you need to be more specific:

make Astrid\_Mager\_final.html

or you could also say:

make Something\_Else.html

STEP 3:

Use a makefile to automatically convert multiple markdowns into separate HTML files (to publish to the web for instance)

SOURCES = $(wildcard \*.md)

HTML = $(SOURCES:%.md=%.html)

all: $(HTML)

%.html : %.md

pandoc $< -f markdown -t html -s -o $@

Here the wildcard command assigns all md files to a variable named SOURCES. The HTML variable is then made replacing the md with html to refer to all (possible) HTML files. Finally the “all” rule is used to require all the HTML files to be generated. In this case, the rule has no recipe lines on its own, but rather is just used to trigger the implicit rule for each of the markdown sources.

STEP 4:

Use a makefile to compile multiple markdown files into an epub.

Pandoc can directly convert multiple markdown files into single epub:

pandoc -o reader.epub \

Introduction.md \

Kylie-Jarrett.md \

Andrea\_Miconi.md \

Vito\_Campanelli.md

In makefile form:

SOURCES = $(wildcard \*.md)

HTML = $(SOURCES:%.md=%.html)

all: $(HTML)

reader.epub : $(SOURCES)

pandoc -o reader.epub $(SOURCES)

%.html : %.md

pandoc $< -f markdown -t html -s -o $@

You can now type:

make reader.epub

A really useful hack to debug makefile variables, is to use the following rule:

print-%:

@echo '$\*=$($\*)'

With this you can say:

make print-SOURCES

STEP 5: Add metadata to individual articles and use them in your epub.

Pandoc supports metadata in documents. These are called YAML metadata blocks and they look like this:

---

Title: Is Small Really Beautiful? Big Search and Its Alternatives

Author: Astrid Mager

---

Google is big in many ways. The company offers a myriad of services and

products ranging from basic keyword search to futuristic glass...

YAML is “yet another” simple markup language for entering structured data that, like markdown, is relatively human readable. An advantage of using YAML is that you can include lists or other kinds of nested structure within your text file. When pandoc processes this file, it first reads the metadata then makes this data available as variables to an output template. Each output format has a default template. You can also provide a custom template to make use of custom variables in your metadata (or if the structure of your metadata is different from what the default pandoc templates expect).

The default HTML template can be displayed with the command:

pandoc --print-default-template html

and looks (in part) like this:

$for(author-meta)$

<meta name="author" content="$author-meta$" />

$endfor$

$if(date-meta)$

<meta name="date" content="$date-meta$" />

$endif$

<title>$if(title-prefix)$$title-prefix$ - $endif$$pagetitle$</title>

$for(css)$

<link rel="stylesheet" href="$css$" $if(html5)$$else$type="text/css" $endif$/>

$endfor$

Create a new file with the contents, and save it as “template.md”:

## $Title$ {.title}

<div class="separator">¬</div>

$for(Author)$

<div class="author">$Author$</div>

$endfor$

$body$

$for(include-after)$

$include-after$

$endfor$

then:

pandoc --to markdown --template template.md Astrid\_Mager.md

See also: <http://johnmacfarlane.net/pandoc/README.html#extension-yaml_metadata_block>

STEP 6:

Create a reader with sections, a table of contents, and metadata.

The final makefile rule used for the Society of the Query looks like this:

sourcemd = $(wildcard source/\*.md)

buildmd = $(patsubst source/%,build/%,$(sourcemd))

all : SotQreader.epub

SotQreader.epub : $(buildmd)

pandoc \

--self-contained \

--table-of-contents \

--toc-depth=2 \

--epub-chapter-level=2 \

--epub-stylesheet=epub.css \

--epub-cover-image=images/cover.png \

--epub-metadata=metadata.xml \

-o SotQreader.epub \

sections/section01.md \

build/Introduction.md \

sections/section02.md \

build/Kylie-Jarrett.md \

build/Andrea\_Miconi.md \

build/Vito\_Campanelli.md \

sections/section03.md \

build/Dirk\_Lewandowski.md \

build/Astrid\_Mager.md \

build/Ippolita\_Pre\_Afterword.md \

build/Angela\_Daly.md \

sections/section04.md \

build/Richard\_Graham.md \

build/Tantner.md \

sections/section05.md \

build/Min\_Jiang.md \

build/JobinGlassey.md \

build/Amanda\_Scardamaglia.md \

build/Martin-Reiche\_Ulrich-Gehmann.md \

sections/section06.md \

build/Jacob\_Ormen.md \

build/Martin\_Feuz.md \

build/David\_Crusoe.md \

build/Simon\_Knight.md \

sections/section07.md \

build/Hogan\_Luka.md \

build/Mahnke\_Uprichard.md \

build/Jones\_Amir.md \

sections/section08.md \

build/SotQ\_Conferences.md \

build/Author\_Bios.md \

build/colophon.md

build/%.md: source/%.md

if [ ! -d build ]; then mkdir build; fi

pandoc \

--to html \

--template article.head.template.md \

$< > $@

pandoc \

--to markdown \

--template article.body.template.md \

--id-prefix=$<- \

--base-header-level=2 \

$< >> $@

clean:

rm -f $(buildmd)

STEP 7:

Create other rules for other kinds of outputs (like an ebook trailer)

References:

Git and Make: not just for code

<https://www.socallinuxexpo.org/scale12x/presentations/git-and-make-not-just-code>

EXTRA STEP: Corrections

\* You’ve received some corrections from the editor! Where do you make them? You can find the prints with corrections in the (real life) folder you received.