Porting Tinderbox HTML Help to PDF

Background

Despote some previous success creating a PDF from the same TBX file as creates the app's HTML Help, it was noticed that the 'internal' links in the document didn't work. Indeed, although a single HTML page of all the HTML content was exporting, intra-doc Tinderbox (TB) links exported pointing to where the individual constituent notes' pages would reside. Although the latter do actually export on a ful HTML export of the source TBX document, the 'PDF' export is from a single note that uses an envelope/letter recursing template cascade to make a single document which can be exported via the container's HTML view.

TB v5 doesn't allow the user control of link mark-up from $Text links. But with a few assumptions, discussed at the end of this document, it is possible to work around the limitation.

The source document for this workflow isn't publicly available but the process described should be clear enough to clone to other TBXs. In this workflow, all page content for the HTML is contained within a root-level container in the TBX called 'elements'. Incidentally the latter exports a folder/file of the same name. No aliases are used in the content area (see assumptions). In-page navigation links like multiple 'go to start' links can be coded entirely within templates so there is no problem to design and implement them as necessary; such steps are not discussed here.

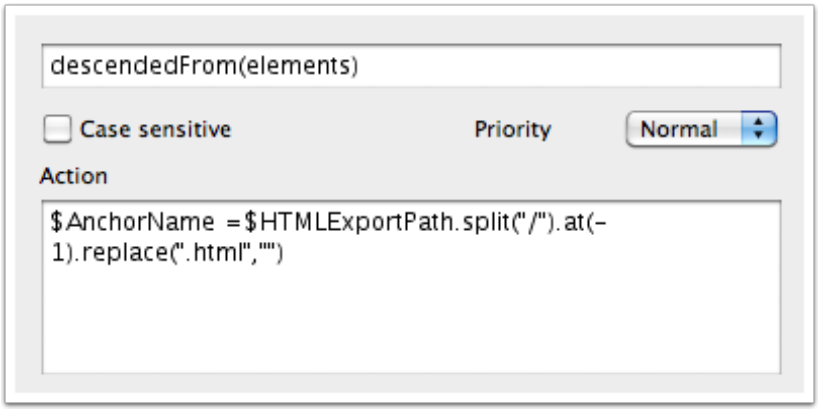
This article addresses the problem of intra-TBX text links exported from $Text via the ^text^ export code. The task was two-fold:

- Make UIDs for anchors targets
- Post-process links in HTML to use those targets

This meant the anchor needed to be information that might be part of the link source HTML. Currently the only safe - although fragile (see assumptions) method is to use the filename. In case that might start with a number, I also use an 'x' prefix.

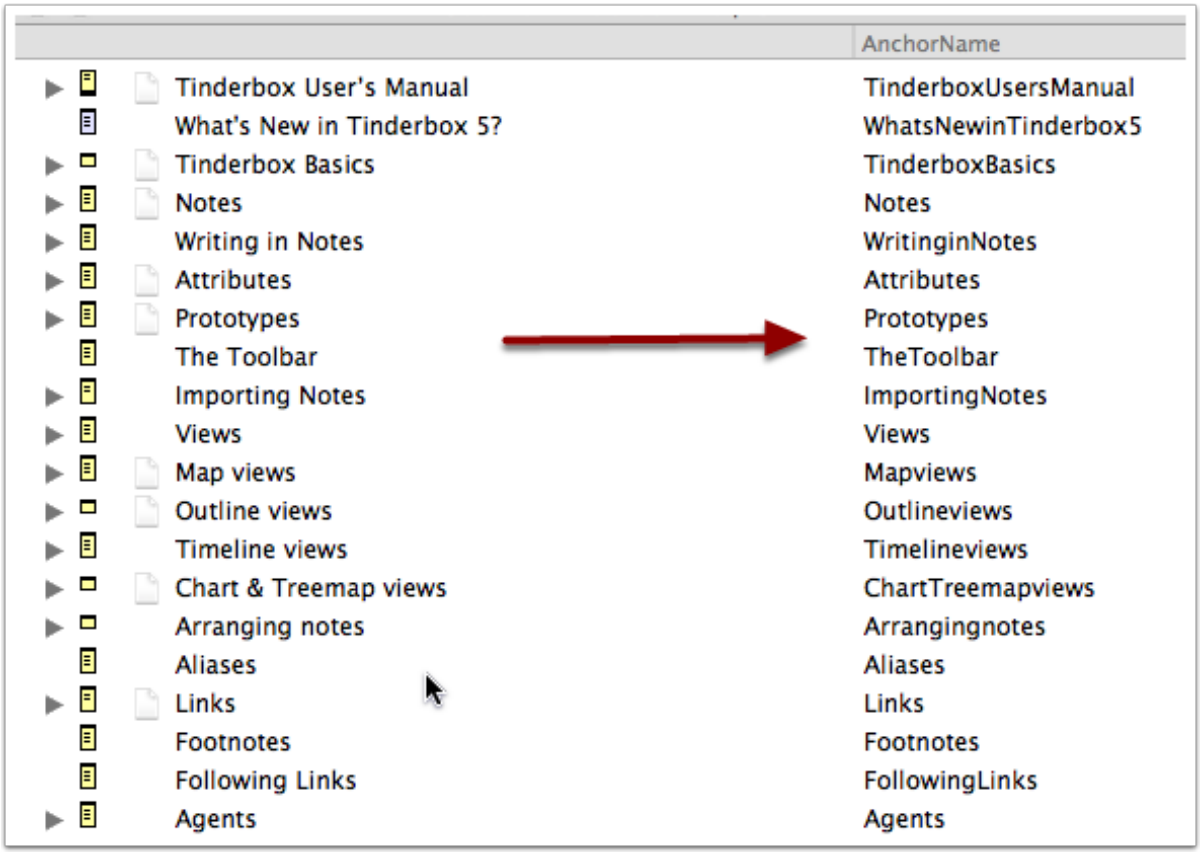
The first step was to make a web-safe string attribute $AnchorName from each exporting note's HTML export name and use agent to populate the new attribute.

Populating $AnchorName - agent detail



This shows the agent. It's left running in case note titles change.

The result of the agent

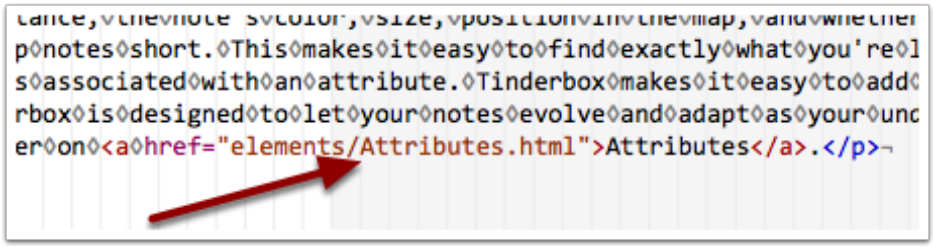


Hopefully self-explanatory!

Considering the problem

The next few steps use an example of the exported HTML to indicate the tasks to be done.

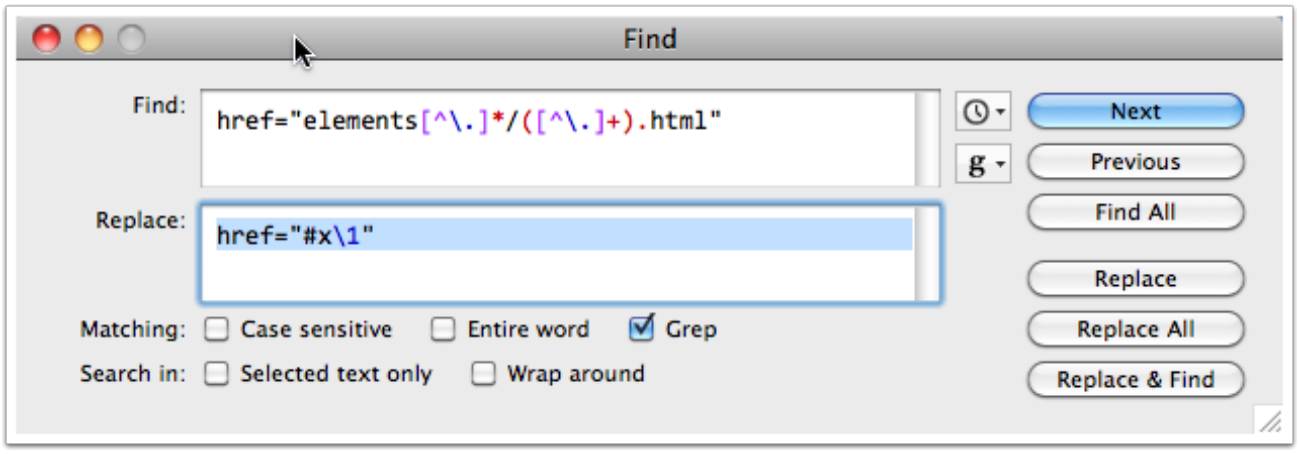
Fixing the exported links in the exported HTML page



The next task was then to find the 'internal' URLs in the exported single-page HTML document (as will be used to make the PDF).

It was important to find only those URLs in HTML tag 'href' attributes and of these only work on those which didn't point to external page (e.g. links to other web resources). Once found, the task was to strip the path and file extension of the front & back of the filename whilst adding back a '#x' prefix to the filename

Post-processing in BBEdit



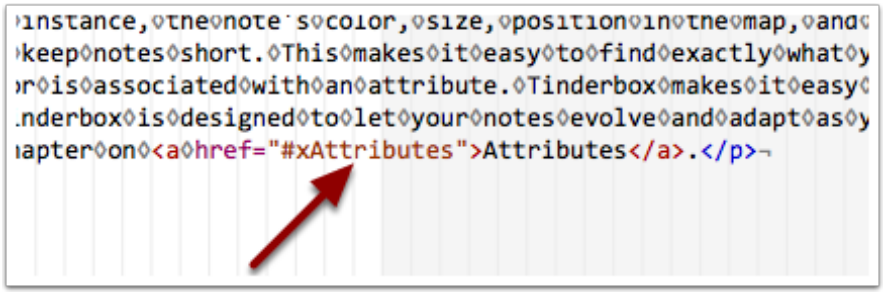
For speed, the HTML single-page export was created and then 'cleaned' in BBEdit, using the settings above. TextWrangler or other regular-expression capable text tool would also work. Note the regex syntax here is for BBEdit (also TextWrangler) and may require modification for use in other editing apps. The Find code:

```
href="elements[^\.]*/([^\.]+).html"
```

The Replace code:

```
href="#x\1"
```

Links in the HTML after running the fix.

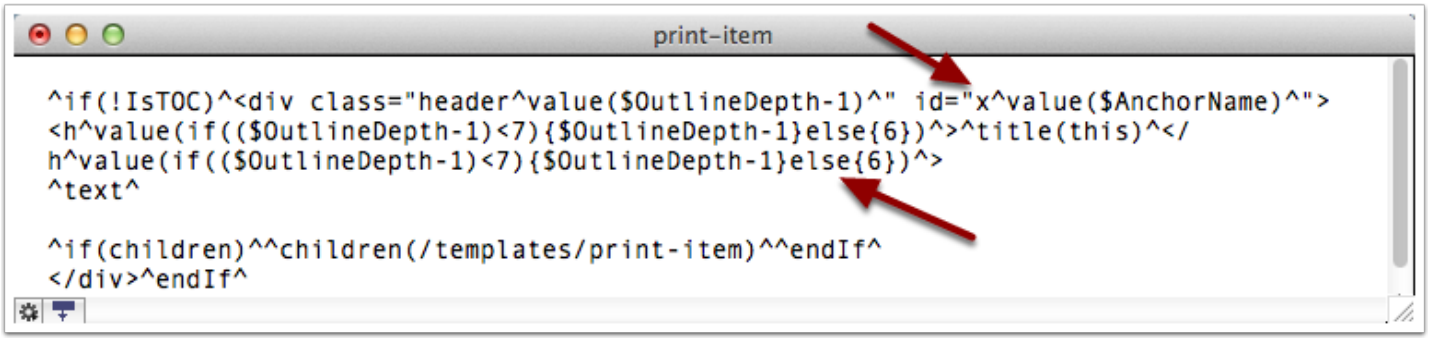


We now have a valid in-page jump-link. Compare the code with that 2 steps above. The URL 'elements/Attributes.html' is now 'xAttributes'. Of course the 'Attributes' value is assumed to be a unique page export name (see the assumptions at the end of the article).

Having validated the basic transform post-export and validated the HTML works, the next step is to achieve this during HTML export from TB.

The easiest part is setting up export of the <a> anchors that are the targets of in-page links.

Setting up the link target anchors



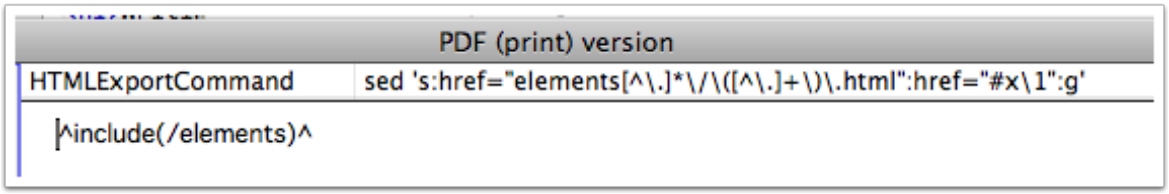
Next, a simple tweak to the export template(s), one of which is shown here; in the source example there were several templates needing this same edit. Here the '#' symbol omitted from the calling URL. Just the 'x' prefix and $AnchorName value are used as the target anchor tag's 'id' attribute value.

The conditional for the heading tag is so heading stop at <h6>, event if there are more deeply nested source notes. The HTML standard doesn't define an <h7> and so on and whilst web browsers don't mind such tags, they seemed to crash the PDF generation stage at the end.

What the output targets look like

Note the well-formed HTML anchor that results. It needs no further processing. The 'x' prefix deals with a situation of a note with a name like '123'. A valide HTML 'id' or anchor must begin with a letter and not a number or other character. Thus the 'x' prefix ensures a HTML id value that will validate, i.e. 'x123' instead of 123'.

Fixing the calling in-page links via $HTMLExportCommand - 1

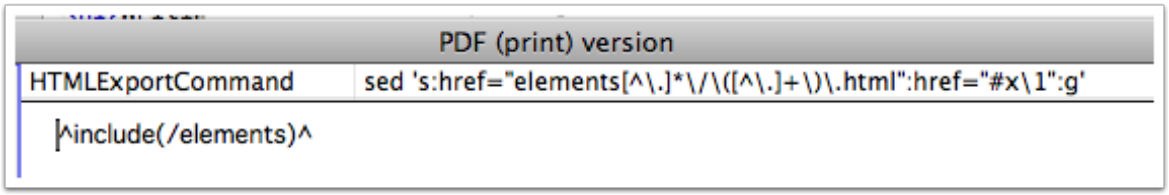


A slightly harder task is the transform of the in-page HTML links, using the method modelled above in BBEdit. The next step was to turn the BBEdit grep-find into a valid Mac Terminal command line. A bit of tinkering with *sed* syntax (Mac differs from normal Unix here) gives this result:

```
sed -E 's:href=\"elements[^\.]*\/([^\.]+)\.html\":href=\"#x\1\":g'
```

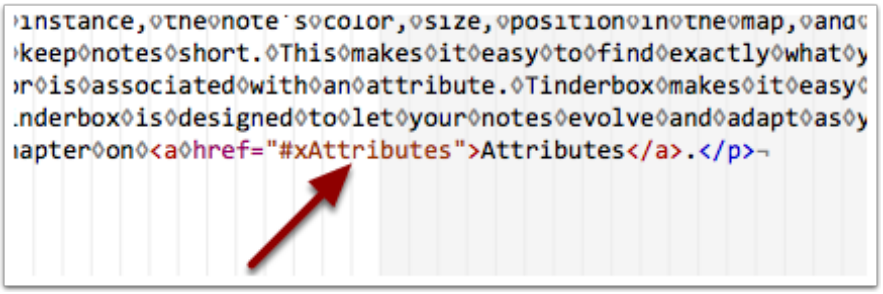
Note another assumption here - the command line assumes all 'normal' export lies within, or is descended from, an export folder 'elements'. For other projects this value would need tweaking.

Fixing the calling in-page links via $HTMLExportCommand - 2



After testing that above successfully in Terminal it was then added to a 'code note' (one using the 'code' prototype). The code note then set its $Text as the single-page export container's $HTMLExportCommand.

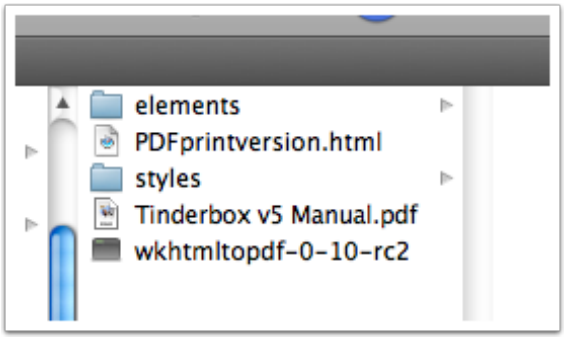
Check the exported HTML



Re-export the simple page. Open in a web-browser and check the in-page links work and check the source code looks correct.

Assuming you've styled your HTML, you are now ready to export to PDF.

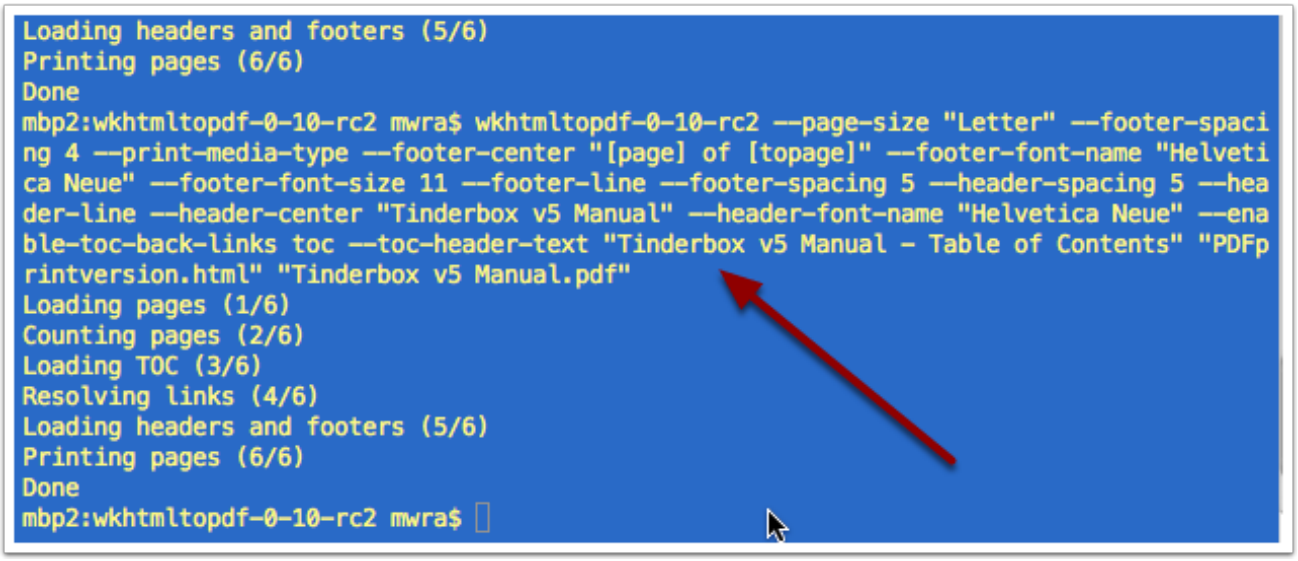
Preparing to use wkhtmltopdf



So, HTML is exported and checked. The HTML, CSS and images are all copied - maintaining relative layout - to the same folder as the wkhtmltopdf tool.

Here, version 0-10-rc2 was used as the latest v11 "wkhtmltopdf.app" was buggy and as-yet unfixed. The tool can be found at: http://code.google.com/p/wkhtmltopdf/.

Run wkhtmltopdf app via Terminal

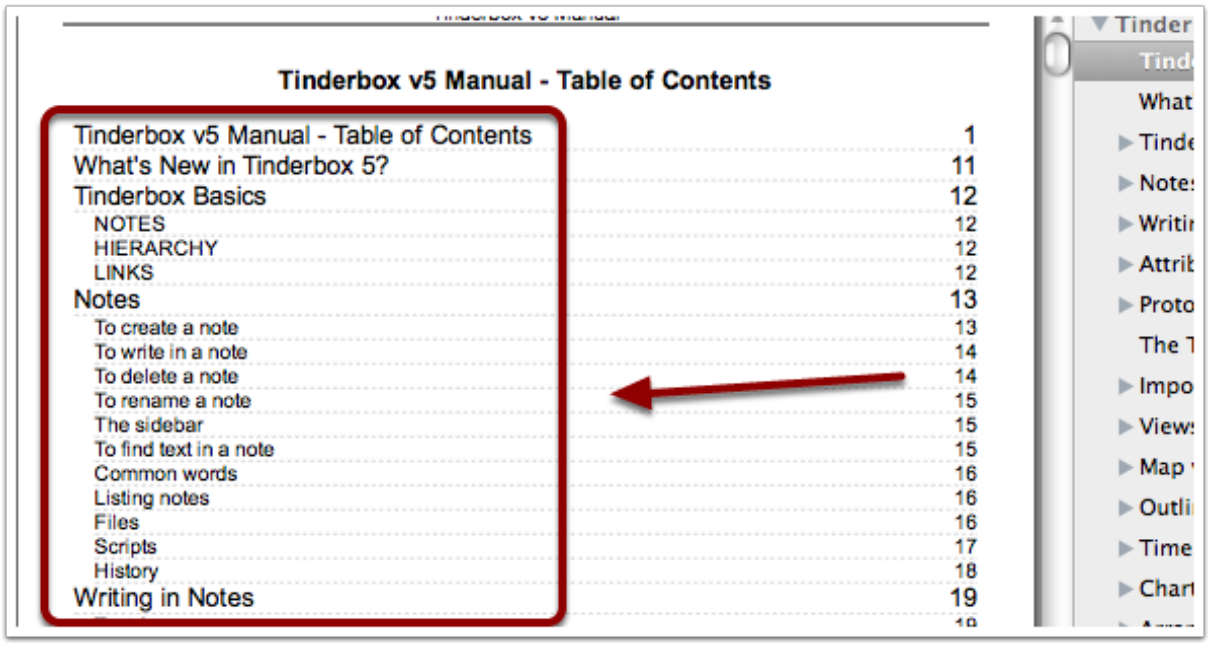


Making the PDF is a matter of changing the Terminal's working folder to the folder holding the above files and running this command:

```
wkhtmltopdf-0-10-rc2 --page-size "Letter" --footer-spacing 4 --print-media-type --
footer-center "[page] of [topage]" --footer-font-name "Helvetica Neue" --footer-font-
size 11 --footer-line --footer-spacing 5 --header-spacing 5 --header-line --header-
center "Tinderbox v5 Manual" --header-font-name "Helvetica Neue" --enable-toc-back-
links toc --toc-header-text "Tinderbox v5 Manual - Table of Contents" "PDFprintversion.
html" "Tinderbox v5 Manual.pdf"
```

It is left to the reader to experiment further with their own settings, but the above should give reasonable output.

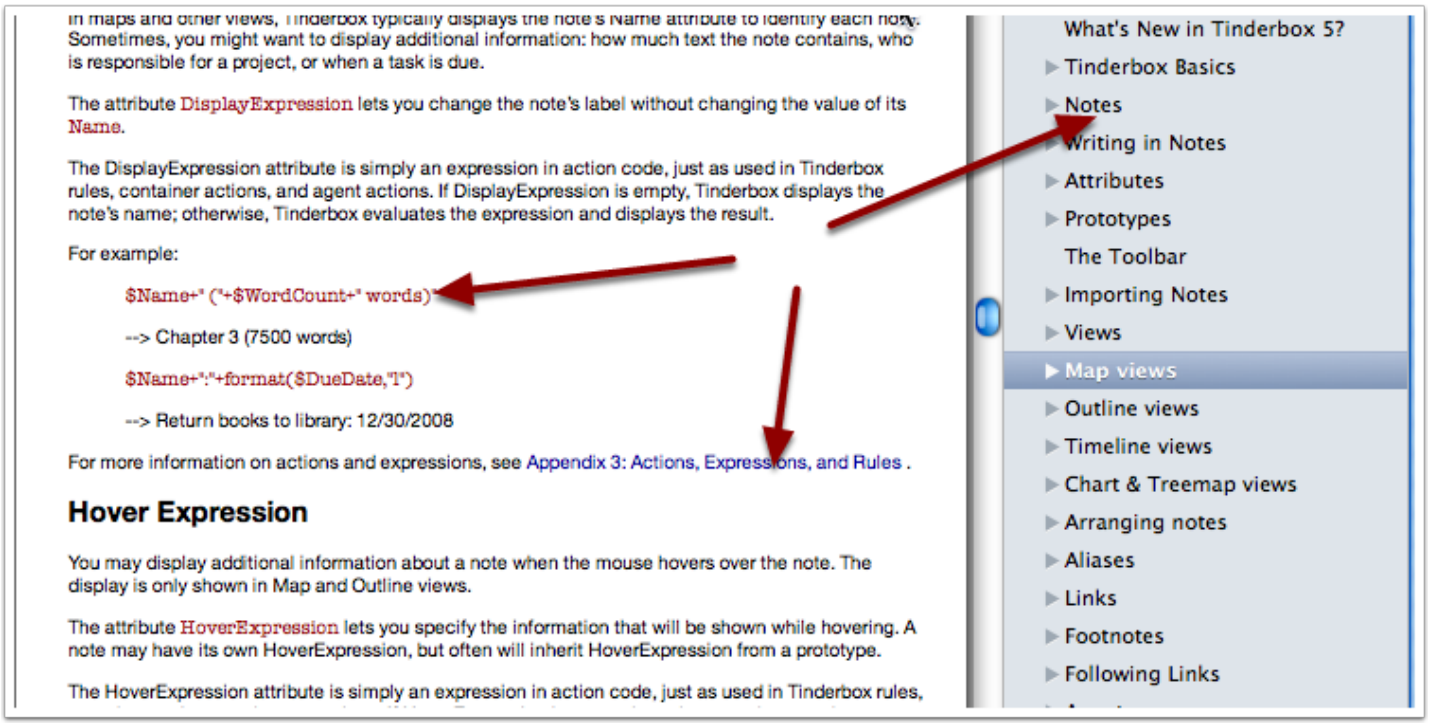
PDF - the TOC



The PDF is ready to use, in the working directory. wkhtmltopdf creates a TOC. The TOC entries are working in-doc links, using the text labels (as opposed to the page numbers - as some might intuit).

Note though that TOC links aren't coloured. The app uses defaults for the TOC styling unless a separate user-supplied style-sheet is provided but there's next to no documentation on the latter!

PDF - content



As shown above, the PDF also gives an outline in the sidebar. TB '<code>' text is in red and intra-doc PDF links are blue.

Job done!

Assumptions

This process makes some assumptions, a few of which are fragile:

- Export filenames are unique. With auto-generated names this can only realistically be held true - without further inspection - if the same note names don't occur at different outline levels.
- Valid HTML anchors cannot start with a digit, so $ID alone cannot be used.
- No aliases are used as these will share their original's filename (unless in the same container).

Because $ID can change for aliases, a first attempt was to use a sequential number attribute for the anchor values. But until/unless TB allows the user to define the way a text link is exporting, it isn't possible to seed the number into the source end of the link, though it's trivial for the anchor. Put another way, the links have to use whatever is in $HTMLExportFileName.

Note that the current solution above would likely break down due to naming collisions if more internal links were used along with re-use of note names (latter not a problem within TB) and/or use of aliases.

Still, for a workaround of link export limitations of TB text and the next-to-none documentation of wkhtmltopdf the result is not so bad!