Test 011: A fenced div environment in the new format used by bookdown

Emma Cliffe, Skills Centre: MASH, University of Bath

August 2020

1 Blah

We should be able to intersect a fenced div WHEN:

- It corresponds to an internal and static list of Bookdown theorem types OR
- It has been declared by a clavertondown user in the _bookdown.yml file as a newtheorem type

NO other fenced divs should be disturbed.

Further, this should continue to function as expected in ALL clavertondown formats and not just HTML and PDF as in Bookdown.

2 What needs to happen

Fenced div:

For a right triangle, if c denotes the length of the hypotenuse and a and b denote the

lengths of the other two sides, we have

$$a^2 + b^2 = c^2$$

should result in exactly the same output as

Theorem 2.1 (Pythagorean theorem)**.**
For a right triangle, if c denotes the length of the hypotenuse and a and b denote the lengths of the other two sides, we have

$$a^2 + b^2 = c^2$$

but other fenced divs INCLUDING those which use user defined clavertondown environments SHOULD NOT convert. We are not opening this up as a format due to the fundamental differences.

Hence the below will NOT convert

For a right triangle, if c denotes the length of the hypotenuse and a and b denote the lengths of the other two sides, we have

$$a^2 + b^2 = c^2$$

```
1+2
```

```
## [1] 3
```

but should have been encoded as

Nugget 2.2.
For a right triangle, if c denotes the length of the hypotenuse and a and b denote the lengths of the other two sides, we have

$$a^2 + b^2 = c^2$$

but that would not be backwards compatibility at this point. That would be new functionality.

The following should not convert: :::: {.theorem #pyth1 name="Pythagorean theorem"} For a right triangle, if c denotes the length of the hypotenuse and a and b denote the lengths of the other two sides, we have

$$a^2 + b^2 = c^2$$

::::

There is an added problem. It is feasible, but I haven't tested it, that an engine based code environment can be embedded in a fenced div environment in the new bookdown approach. For backwards compatibility we need to honour this if needed. It isn't possible in the engine based method to put an engine environment inside another so this functionality cannot be added to standard clavertondown environments - as they are intersepted by knitr rather than Pandoc. It is therefore feasible that this new syntax may become the favoured syntax within clavertondown. So, we have to be really careful here.

We also need to keep all the things we have added working to - which DO NOT work in Bookdown in the new fenced div approach.

3 How to achieve this?

Fenced divs in Pandoc are HTML transformable but not LaTeX/PDF transformable. But, in Pandoc these specific fenced divs ARE transformable to LaTeX (but, I assume, not others). This means that we should look at the latex.R code in the new Bookdown as they must be intercepting at this point.

This does lead to the obvious question: Why only LaTeX/PDF, why not also make it work in EPub and Word? Is there something fundamental I have missed that means these can't be intercepted in this case?

- EPub is basically HTML, maybe it works more or less but they haven't tested it/fixed some small things. Try to not intercept it.
- Word is a different matter.

4 Things to note immediately

Out the box we have the text and in the right style but we have lost everything else. The latter is unsurprising but the former is odd.

Actually, this is true only in HTML so this is less odd.

5 This is a custom Lua filter

Okay, so what happens is there is a new function called common_format_config. This does a few things but the one we care about right now is that it passes ALL control of what happens in LaTeX AND HTML to a custom Lua script for Pandoc.

This is not great. It replicates the same hard-encoding that we see in Bookdown but now we are outside of a situation where we can realistically do anything about it. They have definitely gone further down the hard-encoding route - and further away from being able to do any of the things that our user base wants e.g. with respect to numbering. There is nothing which isn't a terrible hackfest if we go in the Lua direction. It is here that the begins turn up for the LaTeX and there is hard-encoded assumptions about the LaTeX you can get out.

So, we aren't going to do that. That means that we need to intercept and change what is happening BEFORE we get to Pandoc. What are the options?

We need to define a preprocessor. . . There are examples of this in word and ebook formats already. Basically, as a ‘simple’ first fix we are just going to rewrite the markdown. So, we should be able to intercept at this point?

Yes, we can. It rewrites from Rmd to md. BUT, we can’t just replace : with ‘ etc. This is ‘too late’ already. Instead we have to . . . do what the engine does? Is there really no way to rewrite and then re-trigger the engines?

6 Testing all the hard encoded bookdown envs

Theorem

Lemma

Corollary

Proposition

Conjecture

Definition

Example

Exercise

Hypothesis

Proof

Solution

Remark

And this shouldn’t work:

Thing that shouldn’t convert