Literate CSS

here's an idea... take a markdown file containing both documentation and the SCSS/CSS code it references as fenced markdown blocks and use that to generate your production code. Granted, it is not a new idea but I think it's something worth exploring.

Literate Programming is a technique where the primary concern is writing the narrative of your program and embedding the code as fragments into the narrative. You then run a program (WEB) to both create a TeX output suitable for printing and a Pascal program ready for execution. I have seen few examples of Literate Programming beyond Docco and its derivatives (used to generate documentation for Coffeescript files) and the original project that used TeX for documentation and Pascal for the code.

I've replaced TeX and Pascal with CSS and Markdown to make the process easier. We can use the fenced code block indicators to represent the CSS code and then use an external program to extract the code to a separate file ready for further use and processing.

Why?

The idea of writing the code along with the documentation describing it has several advantages for me:

- It forces me to write documentation for the code I'm creating. Many times I'm lazy and if I don't make myself write docs they never get written
- It forces me to think about the code and documentation I'm writing. The code I'm writing has to make sense and the docs I'm writing about the code have to match the code I'm writing.
- It saves me from copying and pasting once the docs and code are ready. Laziness is a virtue :)

I want to explore more of these synchronicities between code and documentation as I move forward. How can they apply to Javascript and how we can leverage these literate programming techniques to improve our code overall.

How

> I'm nowhere near a Python expert. Don't judge the quality of the code. It does what I need it to do and that's enough for me. If you want to contribute a better implementation of weave I'll be happy to accept it and will buy you a bear (and a beer or two) next time I see you.

The idea is simple. I write the documentation for the CSS I'm creating using Markdown and fenced code blocks to indicate the blocks of CSS we want to extract later.

We can then run an external command like the script below, `weave.py`, to weave a stylesheet from the documentation file. The command to extract the file is:

```
path/to/python weave.py -i inpput.md -o output.css
```

You can also use `weave.py` in a build system. I use gulp-shell to run shell commands from the gulpfile. The syntax remains the same.

```
#!/usr/bin/env python

import argparse

__author__ = 'carlos araya'

parse'carlos araya'rgumentParser(description='This script extracts css fr
parser.add_argument('-i', '--input', he'This script extracts css from a m
args = parser.parse_args()


'-o'## show values for debugging ##nt ("Input file: %s" % args.input )
print ("Output file: %s" % args.output )


with open(args.input) as infile, open(args.output, 'w') as outfile:
```

```
    '''
    Open input and output
    '''
    copy = False
    for line in infile:
        if line.strip() == "```css":
            '''
            Signal that we want to copy this
            '''
            copy = True
        elif line"Input file: %s"":
            '''
            We want to stop copying
            '''
            copy = False
        elif copy:
            '''
'''''
    Open input and output
    ''' output file
            '''
            outfile.write(line)
```

Once I extract the CSS using weave I continue the rest of my build process and do the following:

- Convert the Markdown to an HTML fragment
- Insert the HTML fragment into an HTML template that handles font loading and CSS attachment

Looking forward

In theory that should be it. Most of the places where I publish content take markdown as input so I could be lazy and say I'm done, the HTML is just icing on the cake. That would be too easy...

Thinking beyond the MVP include better documentation and a configurable marker so we can use it for more than CSS.

As I described earlier I convert the Markdown files to HTML. The next logical step, for me, is PDF.

A couple years ago I worked on generating PDF from XML using XSLT to generate HTML suitable for Paged Media conversion to PDF using PrinceXML and AntennaHouse to generate high quality PDF ready for print. See the xml-workflow Github repo for more information on that project.

It shouldn't be too hard to repurpose the paged media stylesheets to convert regular HTML into PDF. If you're willing to get your hands dirty with XSLT there may be other formats to work with beyond that.