Pandoc Notes

Gus Dunn

Contents

1 Code syntax highlighting

1.1 Basic example from johnmacfarlane.net

Source: johnmacfarlane.net/pandoc/demo/example18f.html

Here's what a delimited code block looks like:

```
-- | Inefficient quicksort in haskell.
qsort :: (Enum a) => [a] -> [a]
qsort []     = []
qsort (x:xs) = qsort (filter (< x) xs) ++ [x] ++
               qsort (filter (>= x) xs)
```

Here's some python, with numbered lines (specify {.python .numberLines}):

```
class FSM(object):

"""This is a Finite State Machine (FSM).
"""

```

```
def __init__(self, initial_state, memory=None):

   """This creates the FSM. You set the initial state here. The "memory"
   attribute is any object that you want to pass along to the action
   functions. It is not used by the FSM. For parsing you would typically
   pass a list to be used as a stack. """

   # Map (input_symbol, current_state) --> (action, next_state).
   self.state_transitions = {}
   # Map (current_state) --> (action, next_state).
   self.state_transitions_any = {}
   self.default_transition = None
   ...
```

1.2 Highlighting languages/styles available

- based on excerpts from `Highlighting.hs` on MacFarlane's github repo: `jgm/pandoc-highlight`

1.2.1 Languages

```
...

langsList :: [(String, String)]
langsList =    [("ada","Ada")
               ,("java","Java")
               ,("prolog","Prolog")
               ,("python","Python")
               ,("gnuassembler","Assembler")
               ,("commonlisp","Lisp")
               ,("r","R")
               ,("awk","Awk")
               ,("bash","bash")
               ,("makefile","make")
               ,("c","C")
               ,("matlab","Matlab")
               ,("ruby","Ruby")
               ,("cpp","C++")
               ,("ocaml","Caml")
```

```
                ,("modula2","Modula-2")
                ,("sql","SQL")
                ,("eiffel","Eiffel")
                ,("tcl","tcl")
                ,("erlang","erlang")
                ,("verilog","Verilog")
                ,("fortran","Fortran")
                ,("vhdl","VHDL")
                ,("pascal","Pascal")
                ,("perl","Perl")
                ,("xml","XML")
                ,("haskell","Haskell")
                ,("php","PHP")
                ,("xslt","XSLT")
                ,("html","HTML")
                ]

...
```

1.2.2 Styles

Looks like only these?

- pygments
- espresso
- zenburn
- tango
- kate
- monochrom
- haddock

```
module Text.Pandoc.Highlighting ( languages
                                , languagesByExtension
                                , highlight
                                , formatLaTeXInline
                                , formatLaTeXBlock
                                , styleToLaTeX
                                , formatHtmlInline
                                , formatHtmlBlock
                                , styleToCss
```

```
                                  , pygments
                                  , espresso
                                  , zenburn
                                  , tango
                                  , kate
                                  , monochrome
                                  , haddock
                                  , Style
                                  , fromListingsLanguage
                                  , toListingsLanguage
                                  ) where
```

1.2.3 `highlighting-kate`

Actually it looks like any highlighting style supported by Kate (*as long as you have it installed properly?*) should be supported.

From benjeffrey.com:

> **The highlighting-kate Package**
>
> Unless you dig into Pandoc's Haddock documentation, you won't find much information on the internet telling you what the generated markup classes (.kw, .co, .ot, etc.) mean, or how you can customize them.
>
> It turns out that Pandoc relies on another one of John Macfarlane's creations in order to mark up code syntax, the highlighting-kate package,
>
> > *a syntax highlighting library with support for nearly one hundred languages. The syntax parsers are automatically generated from Kate syntax descriptions (http://kate-editor.org/), so any syntax supported by Kate can be added.*
> >
> > -highlighting-kate package description
>
> So it turns out that the syntax tokens are based on definitions provided by the KDE text editor, Kate.

1.2.4 `Pygments` styles for highlighting

For this you will need to write a filter that sends every code block to `pygments` and makes sure the result ends up in the document.

1.3 Invoking the `highlighting-kate` styles

You add `--highlight-style <style-name>` to the command line or in your `Makefile`.