Evolution of the Document Model

Sébastien Boisgérault, MINES ParisTech, under CC BY-NC-SA 4.0

February 7, 2018

Contents

Introduction

`Text.Pandoc.Definition` is the module of the pandoc-types Haskell package that contains the definitions for the Pandoc data structure used by pandoc to represent structured documents.

Actually, there is not a single model of document, but a collection of models since the original design has evolved over time. Unless you deliberately use a fixed version of pandoc – and therefore a unique document model – you may at some point need to manage several models of documents.

To ease this work, we describe here the evolution of this model, from the creation of pandoc-types to the present day.

Evolutions

Version 1.7

The original model, included below for the sake of completeness: 66 lines only, that describe the structure of any pandoc document.

```
data Pandoc = Pandoc Meta [Block]
data Meta
  = Meta {docTitle :: [Inline],
          docAuthors :: [[Inline]],
          docDate :: [Inline]}
data Alignment
  = AlignLeft | AlignRight | AlignCenter | AlignDefault
type ListAttributes = (Int, ListNumberStyle, ListNumberDelim)
data ListNumberStyle
  = DefaultStyle
  | Example
  | Decimal
  | LowerRoman
  | UpperRoman
  | LowerAlpha
  | UpperAlpha
data ListNumberDelim = DefaultDelim | Period | OneParen | TwoParens
type Attr = (String, [String], [(String, String)])
type TableCell = [Block]
data Block
  = Plain [Inline]
  | Para [Inline]
  | CodeBlock Attr String
  | RawHtml String
  | BlockQuote [Block]
  | OrderedList ListAttributes [[Block]]
  | BulletList [[Block]]
  | DefinitionList [([Inline], [[Block]])]
  | Header Int [Inline]
  | HorizontalRule
  | Table [Inline] [Alignment] [Double] [TableCell] [[TableCell]]
  | Null
```

```
data QuoteType = SingleQuote | DoubleQuote
type Target = (String, String)
data MathType = DisplayMath | InlineMath
data Inline
  = Str String
  | Emph [Inline]
  | Strong [Inline]
  | Strikeout [Inline]
  | Superscript [Inline]
  | Subscript [Inline]
  | SmallCaps [Inline]
  | Quoted QuoteType [Inline]
  | Cite [Citation] [Inline]
  | Code String
  | Space
  | EmDash
  | EnDash
  | Apostrophe
  | Ellipses
  | LineBreak
  | Math MathType String
  | TeX String
  | HtmlInline String
  | Link [Inline] Target
  | Image [Inline] Target
  | Note [Block]
data Citation
  = Citation {citationId :: String,
              citationPrefix :: [Inline],
              citationSuffix :: [Inline],
              citationMode :: CitationMode,
              citationNoteNum :: Int,
              citationHash :: Int}
data CitationMode = AuthorInText | SuppressAuthor | NormalCitation
```

Version 1.8

Inline code gets attributes

The code block element of the 1.7 document model has an attributes (`Attr`) field. For example, a code block described by

```
~~~~ {#mycode .haskell .numberLines startFrom="100"}
qsort []     = []
qsort (x:xs) = qsort (filter (< x) xs) ++ [x] ++
```

```
               qsort (filter (>= x) xs)
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
```

has the attributes field

```
[["mycode",["haskell","numberLines"],[["startFrom","100"]]]
```

This triple should be interpreted as id, classes and key-value pairs, something very similar to the model of HTML elements.

In the 1.7 model, however, inline code lack attributes; in the 1.8 version, they get this field too:

```
data Inline
  = ...
  | Code Attr String
  | ...
```

For example, the markdown text

```
`<$>`{.haskell}
```

attaches the class `haskell` to the inline code `<$>`. A `nullAttr` constant is also introduced to denote the default value of the attributes field

```
nullAttr :: Attr
```

Raw elements

In the 1.7 model, HTML fragments could be inserted in documents as `RawHtml` blocks or `HtmlInline` inlines; TeX fragments could only be inline, with the `TeX` type.

In the 1.8 model, these constructs are generalized and merged: native (HTML or TeX) elements are now described either as raw blocks or raw inlines:

```
type Format = String
data Block
  = ...
  | RawBlock Format String
  | ...
data Inline
  = ...
  | RawInline Format String
  | ...
```

Concretely, the format is either “tex” or “html” but this is a not a hard constraint of the document model.

Version 1.9(.0.1)

(there is no version 1.9 released, only 1.9.0.1, 1.9.0.2 and 1.9.1)

New representation for em-dash, en-dash, apostrophe and ellipses

`EmDash`, `EnDash`, `Apostrophe` and `Ellipses` disappear from the `Inline` type. You can still use the corresponding constructs in you markdown text, but they will be described as regular string (`Str`) instances, which makes sense: since the unicode standard knows about them, it is pointless to have an extra construct.

Version 1.12

A new metadata model

In the initial document model, the metadata of a document was made of three fixed attributes, specified in a title block: the document title, author list and date.

```
data Meta
  = Meta {docTitle :: [Inline],
          docAuthors :: [[Inline]],
          docDate :: [Inline]}
```

A typical markdown document would start with lines prefixed by percentages to specify these attributes, such as

```
 % Document Title
 % Author 1; Author 2; Author 3
 % Date
```

Starting with 1.12, pandoc support arbitrary YAML metadata blocks

```
newtype Meta = Meta {unMeta :: Map String MetaValue}
data MetaValue
  = MetaMap (Map String MetaValue)
  | MetaList [MetaValue]
  | MetaBool Bool
  | MetaString String
  | MetaInlines [Inline]
  | MetaBlocks [Block]
```

Helper functions are created to help with the new structure, for example to get the title, authors and date.

```
nullMeta :: Meta
isNullMeta :: Meta -> Bool
```

```
lookupMeta :: String -> Meta -> Maybe MetaValue
docTitle :: Meta -> [Inline]
docAuthors :: Meta -> [[Inline]]
docDate :: Meta -> [Inline]
```

Introduction of span and div

The span and div constructs from HTML are incorporated to the document model, as inline and block elements respectively:

```
data Inline
  = ...
  | Span Attr [Inline]
  | ...

data Block
  = ...
  | Div Attr [Block]
  | ...
```

Their markdown syntax is the classic HTML one.

Headers get attributes

Instead of

```
data Block
  = ...
  | Header Int [Inline]
  | ...
```

we now get

```
data Block
  = ...
  | Header Int Attr [Inline]
  | ...
```

`Format` becomes a `newtype`

A technical change to improve the type safety of the data model: `Format` was a simple synonym for `String`

```
type Format = String
```

It becomes a distinct type whose content is still a string:

```
newtype Format = Format String
```

Version 1.16

Soft breaks

When pandoc reads markdown files, only explicit line breaks – specified with either `\` or two spaces at the end of the line – are registered – as `LineBreak` elements – and thus may be preserved in output documents.

With soft breaks, introduced in the 1.16 document model, the input to output behavior remains the same by default, but the weaker variants of line breaks, only specified by a new line are now registered; to make them apear in your output document, you can use the option `--wrap=preserve`.

```
data Inline
  = ...
  | SoftBreak
  | ...
```

Links and images get attributes

The new model is:

```
data Inline
  = ...
  | Link Attr [Inline] Target
  | Image Attr [Inline] Target
  | ...
```

Version 1.17

Line blocks

A structure that preserves line breaks and leading spaces (refer to the user manual for details):

```
data Block
  = ...
  | LineBlock [[Inline]]
  | ...
```

Version 1.19

No change with respect to 1.17.

Version 2.0

No change with respect to 1.19.

Version 2.1

No change with respect to 2.0.