

# Understanding pandoc lua filters

by *Abhishek Ulayil*

**Abstract** Pandoc supports intermediate modification of the Abstract Syntax Tree (AST) between the parsing and writing phase using filters. This supplement paper highlights the use cases of such filters written in the Lua language on LaTeX, markdown and native AST.

## 1 Introduction

To fully understand pandoc filters, one must first grasp the document conversion process. Pandoc will read/parse an article and then store it internally as an intermediate form, known as an abstract syntax tree (AST). Then, A document writer(a specialized type-setter) will read the AST and produce the contents in the chosen article format. Converting markdown to HTML, for example, will entail these pandoc operations. Figure 1 shows the conversion workflow within pandoc.

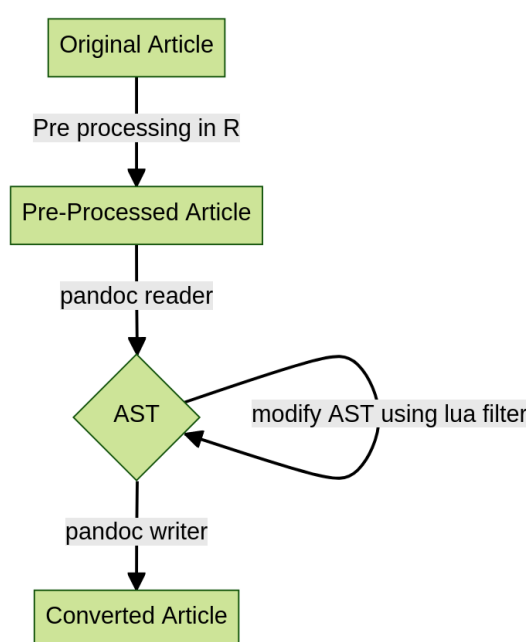


Figure 1: Conversion workflow.

When specific elements require adjustment, the Lua filters enter the picture. Assume you're converting markdown to HTML and want the page to have automated numbering for figures or tables. If you use markdown, there is no system for automatically numbering figures/tables; to retain 1:1 conversion, the HTML output will also lack numbering.

At this point, you might wish there was a way to add such a feature. But including every customization as an option in pandoc or the writer is not viable. To address this, the idea of filters emerged, with which you could modify the AST before the writer could read it to obtain the desired result (MacFarlane and other pandoc authors, 2023).

Pandoc filters can technically be written in any language (called JSON filters) <sup>1</sup>.

## 2 Need of Lua filters

Continuing the example above, let's create a dummy article, where we need to add numbering. One way could be to keep a counter of images, and add a prefix "Figure X : " to each figure caption. This will serve the purpose of numbering the images in the end result.

```
![R logo](Rlogo-5.png){width="10%"}

```

<sup>1</sup>More on this [here](#).

```
![penguins](penguins.png){width="15%"}
```

```
pandoc example.md --from markdown --to html5 --output example.html
```

**Figure 2:** pandoc command without lua filter or extensions.

Now if we convert the above markdown file to HTML5 using the pandoc command in Figure 2, we get Figure 3.



**Figure 3:** Vanilla HTML5 output.

As we can see, there is no figure numbering done automatically, which is generally the expected result. If we want to include numbering, we would need to write a Lua filter. This Lua filter will modify the AST and make the changes we desire.

We call a Lua filter in the pandoc command show in Figure 2 using the `--lua-filter name_of_filter.lua` option in pandoc.

In the next section we write a Lua filter to manipulate the figures in Figure 4.

### 3 Writing pandoc Lua filters

Everything in Lua revolves around tables; for example, the pandoc AST or the document is one giant table with sub-tables.

In pandoc, there are two sorts of elements: 'Blocks' and 'Inlines'. 'Blocks' are complicated constructions constructed from simpler pieces ('Inlines'). A 'Para', for example, is a 'Block' made up of several 'Str' 'Inlines'.

The first step in writing a filter is to select a target type, that is, which 'Block' or 'Inline' this filter will target. Following the selection, we name the function after the target type and each element in the argument as 'el'. For instance, in the case of our filter, it would be function `Figure(el)`. Pandoc is smart enough to match the names of filter functions to the AST elements.

Now within the filter function we can assume that `el` will contain the table object of a Figure element from the document. We can use many functions over it or 'Inlines' contained in it. One such function to walk over all the elements inside the block is `walk_block`. Walk functions are a great way to check or count the presence of certain elements in a block. We use `walk_block` to check if there are any 'Image' inline elements within the 'Figure' block. This is because after pandoc 3 (Krewinkel, Lucero, 2023) 'Figure' blocks can now contain elements other than 'Images' as well.

If there is an Image then we append "Figure X : " to the caption of the Figure element. and return the modified element.

```

figures = 0
is_fig = 0

function Figure(el)
  local label = ""
  pandoc.walk_block(el,{ Image = function(el)
    is_fig = 1
  end})
  if is_fig == 1 then
    figures = figures + 1
    label = "Figure " .. tostring(figures) .. ":"
  end
  local caption = el.caption
  if not caption then
    caption = {pandoc.Str(label)}
  else
    caption = {pandoc.Str(label),pandoc.Space()}
  end
  el.caption.long[1].content = caption .. el.caption.long[1].content
  is_fig = 0
  return el
end

```

**Figure 4:** A Lua filter to add figure numbering.

## 4 Interpreting changes using Lua filters

The filter in Figure 4 when included in the pandoc command will generate Figure 6 using the command in Figure 5.

```

pandoc example.md --from markdown --to html5 --output filtered-example.html
--lua-filter image_numbering_filter.lua

```

**Figure 5:** pandoc command with Lua filter.

## 5 Usage of pandoc lua filters in the **texor** package

Pandoc Lua filters are used in the **texor** package to modify the AST for various markups. Table 1 summarizes the use of each filter.

### Bibliography

A. Krewinkel and A. Lucero, pandoc 3.0 Release notes, *pandoc* 2023 URL <https://pandoc.org/releases.html> [p2]

John MacFarlane and other pandoc authors, Pandoc Lua filters, *pandoc documentation* 2023 URL <https://pandoc.org/lua-filters.html> [p1]

*Abhishek Ulayil*  
*Student, Institute of Actuaries of India*  
*Mumbai, India*  
*ORCID: 0009-0000-6935-8690*



Figure 1: R logo



Figure 2: penguins

Figure 6: HTML5 output with desired filtering.

File Name	Description
abs_filter.lua	Filters out unicode 182 character.
bib_filter.lua	Clears out the bibliography from the article itself as it will be added to the metadata.
R_code.lua	Adds a class component 'r' to CodeBlocks for code highlighting.
image_filter.lua	Fixes extensions for image paths without an extension or with .pdf extension.
table_caption.lua	Adds table numbering in the captions similar to the ones found in R Markdown and <b>kable</b> -based tables.
image_caption.lua	Adds Figure/Algorithm numbering as well as clears out residuals of tikz/algorithm images.
conversion_compat_check.lua	This filter keeps a count of all Inline and Block elements and writes it to a yaml file.
equation_filter.lua	Adds <b>bookdown</b> style equation numbering to LaTeX equation/math environments.
bookdown_ref.lua	Corrects numbering for various elements.
issue_checker.lua	Searches and notifies any occurrence of unrecognized math commands.
find_pdf_files.lua	This filter creates a list for all image PDF files included in the article.
widetable_patcher.lua	This filter sets the representation for widetables.

**Table 1:** A summary of all Lua filters used in the **texor** package (available in the `/inst` folder of the package).