Panflute API Documentation

1. General functions (that replace the ones in pandocfilters)

toJSONFilter(action, prepare=None, finalize=None, **kwargs)

- **action(element, doc, . . .)**: A function that works on each element. See walk to see how the iteration is done
- **prepare(doc)**: A function run after the document is processed, before the filters are applied.
- **finalize(doc)**: A function run after the filters are applied are processed, before the document is converted back to JSON and returned to Pandoc.
- **kwargs**: extra keywords that are passed to action()

toJSONFilters(actions, prepare=None, finalize=None, input__stream=None, output__stream=None, **kwargs)

Like `toJSONFilter` but takes a list of actions instead of a single action.

Notes:

- The second action is only applied after the first is done with the entire document, and so on.

doc = load()

Load the document from stdin.

dump(doc)

Dump the document as JSON back into stdout, returning control to Pandoc.

walk(element, action, doc)

Walk through element and all other elements within it, applying action to them.

- element: the root element of the tree that will be walked. It can also be a list or the doc element
- **action(element, doc, . . .)**: A function that works on each element.
- **doc**: The entire document

Notes:

- The walking is done in a depth-first pattern, going as deep as possible into the items of each element, and only then going to the next element.

stringify(element)

2. Additional functions (useful for complex filters)

convert_markdown(text, format='json')

Use Pandoc to parse a string as markdown, and return the corresponding list of elements, or (if a format other than 'json' is used) return the desired format.

Based on pyandoc by Kenneth Reitz.

yaml_filter(element, doc, tag, function)

This filter will call `function(options=options, data=data, element=element, doc=doc)` for all code blocks that have the `tag` class. The content is then parsed by YAML and loaded into the `options` dictionary. Any content below the YAML ending delimiters (`...` or `---`) is passed unprocessed as `data`.

shell(args, wait=True, msg=None)

Call a shell program. Still experimental.

debug(*args, **kwargs) {#debug}

Use this as a replacement for print(), because print doesn't work while using a filter from pandoc.

replace_keyword(doc, keyword, replacement)

- **doc**: The entire document
- **keyword**: A word (without spaces) that will be replaced
- **replacement**: The element that will replace the Str() that contains the word. If the element is a Block, we will replace the Para() that contains the string; if the para contains more than just the string, the replacement will fail.

3. Special Elements

Doc(metadata, items, format)

Properties:

- **.metadata**: has a nested dictionary with metadata items (possibly with other lists or dictionaries within)
- **.items**: list with the top–level elements
- **.format**: contains the output format

Methods:

- **get_metadata(request, default)**: Use this to retrieve nested keywords. For instance, `doc.get_metadata('setup.spacing', 1.0)` is an alternative to `doc.metadata.get('setup', {}).get('spacing', 1.0)`

Notes:

- `doc` allows for optional attributes, which can be used as global variables across calls

Element

Super class of all standard elements

Block

Super class of all standard block elements

Inline

Super class of all standard inline elements

4. Standard Elements

Null() -> Block

Nothing

Space() -> Inline

Inter-word space

HorizontalRule() -> Block

Horizontal rule

SoftBreak() -> Inline

Soft line break

LineBreak() -> Inline

Hard line break

Plain(*args -> Inline) -> Block

Plain text, not a paragraph

Para(*args -> Inline) -> Block

Paragraph

BlockQuote(*args -> Block) -> Block

Block quote (list of blocks)

Emph(*args -> Inline) -> Inline

Emphasized text (list of inlines)

Strong() -> Inline

Strongly emphasized text (list of inlines)

Strikeout(*args -> Inline) -> Inline

Strikeout text (list of inlines)

Superscript(*args -> Inline) -> Inline

Superscripted text (list of inlines)

Subscript(*args -> Inline) -> Inline

Subscripted text (list of inlines)

SmallCaps(*args -> Inline) -> Inline

Small caps text (list of inlines)

Note(*args -> Block) -> Inline

Footnote or endnote

Header(*args -> Inline, level=1, identifier=", classes=[], attributes={}) -> Block

Header - level (integer) and text (inlines)

Div(*args -> Block, identifier=", classes=[], attributes={}) -> Block

Generic block container with attributes

Span(*args -> Inline, identifier=", classes=[], attributes={}) -> Inline

Generic inline container with attributes

Quoted(*args -> Inline, quote_type='DoubleQuote') -> Inline

Quoted text (list of inlines)

quote_type:

- SingleQuote
- DoubleQuote

Cite(*args -> Inline, citations) -> Inline

Cite (list of inlines)

Citation(citationHash, citationId, citationMode, citationNoteNum, citationPrefix, citationSuffix) -> Element

Link(*args -> Inline, url, title, identifier=”, classes=[], attributes={}) -> Inline

Hyperlink: alt text (list of inlines), target

url Target (string) title
Alt. title (?)

Image(*args -> Inline, url, title, identifier=”, classes=[], attributes={}) -> Inline

Image: alt text (list of inlines), target

url Target (string) title
Alt. title (?)

Str(text) -> Inline

Text (string)

CodeBlock(text, identifier=”, classes=[], attributes={}) -> Block

Code block (literal) with attributes

RawBlock(text, format) -> Block

Raw block

format:

- html
- tex
- latex

Code(text, identifier=”, classes=[], attributes={}) -> Inline

Inline code (literal)

Math(text, format) -> Inline

TeX math (literal)

format:

- DisplayMath
- InlineMath

RawInline(text, format) -> Inline

Raw inline

format:

- html
- tex
- latex

BulletList(*args -> [Block]) -> Block

Bullet list (list of items, each a list of blocks)

```
Block BulletList(items=[[Block]])"""
```

OrderedList(*args -> [Block], start=1, style='Decimal', delimiter='Period') -> Block

Ordered list (attributes and a list of items, each a list of blocks)

Number styles:

- DefaultStyle
- Example
- Decimal
- LowerRoman
- UpperRoman
- LowerAlpha
- UpperAlpha

Number delimiters:

- DefaultDelim
- Period
- OneParen
- TwoParens

DefinitionList(*args -> [([Inline] , [Block])) -> Block

Definition list Each list item is a pair consisting of a term (a list of inlines) and one or more definitions (each a list of blocks)

Table(*args -> [[Block]], header=None -> [Block], caption=None -> Inline, alignment=None -> [Alignment], width=None -> [Double]) -> Block

Table, with caption, column alignments (required), relative column widths (0 = default), column headers (each a list of blocks), and rows (each a list of lists of blocks)

Alignment:

- AlignLeft
- AlignRight
- AlignCenter
- AlignDefault