Org mode (La)TEX macros for HTML and LATEX export

Brian C. Wells

August 25, 2016

Contents

1 Preliminary Macros **2**
1.1 when-fmt 2
1.2 preamble 2

2 Definition Macros **3**
2.1 def 3
2.2 newcommand 3
2.3 renewcommand 4
2.4 newenvironment 4
2.5 renewenvironment 4

3 Installation **4**

4 Usage Examples **4**
4.1 **TODO** def 4
4.2 newcommand (without parameters) 4
4.3 newcommand (with a parameter) 5
4.4 **TODO** renewcommand 5
4.5 **TODO** newenvironment 5
4.6 **TODO** renewenvironment 5

This is an Org mode Literate Program, which is tangled (by pressing `C-c C-v t`, in the default keymap with the `.org` file loaded) to an Org mode setup file named `define.setup`. The program uses Org mode's builtin macros, which (apparently) have an undocumented feature of executing any Emacs Lisp code they contain (tested in Emacs 24.5.1, Org mode 8.2.9), to define a generic macro `when-fmt` for including code in only specified export

formats (and any formats derived from them). It then uses this to define a `preamble` macro for inserting preamble code in HTML and LaTeX formats, and finally several macros for each definition command supported by both MathJax and (La)TeX.

This document should be exported to HTML and LaTeX to check that the proper code is generated. A PDF file should also be available, but looks a bit bad because Org mode macros *must* be written on a single line, and some of these macros overfill the line (even in a fairly small font).

The first line makes sure that the file can be edited in org-mode despite the file being named with an extension of `.setup`.

```
# -*- mode: org -*-
#+MACRO: when-fmt (eval (when (org-export-derived-backend-p org-export-current-backend '$1) "$2"))
#+MACRO: preamble {{{when-fmt(html,\\($1\\))}}}{{{when-fmt(latex,\n#+LATEX_HEADER: $1\n)}}}
#+MACRO: def {{{preamble(\\def$1{$2})}}}
#+MACRO: newcommand {{{preamble(\\newcommand{$1}$3{$2})}}}
#+MACRO: renewcommand {{{preamble(\\renewcommand{$1}$3{$2})}}}
#+MACRO: newenvironment {{{preamble(\\newenvironment{$1}$4{$2}{$3})}}}
#+MACRO: renewenvironment {{{preamble(\\renewenvironment{$1}$4{$2}{$3})}}}
```

1 Preliminary Macros

1.1 when-fmt

This is inspired by the `if-latex-else` macro under the 'Advanced' heading here: `https://github.com/fniessen/org-macros`. Apparently, Org mode will evalute Emacs Lisp code in macros, although I have not yet found any documentation that explains *why* this works.

The `when` form is like `if`, except it only returns a string when the condition is true, returning `nil` instead when it is false. We use `when` because we want to perform an action for LaTeX and HTML formats, and we do not want to assume that the user wants the same behavior for all non-LaTeX or non-HTML formats.

```
#+MACRO: when-fmt (eval (when (org-export-derived-backend-p org-export-current-backend '$1) "$2"))
```

Since the parameter `$2` is inside quotes, it will be necessary to double any (literal) backslashes, despite the fact that Org mode macros do not normally require this (except between parameters).

1.2 preamble

Using the 1.1 macro, we wrap HTML output in `\(...\)` so that the MathJax library will recognize that it should process them. So long as we only use

this to define LaTeX macros, MathJax will not generate any spurious output. In LaTeX output, we use the `#+LATEX_HEADER:` Org mode syntax to ensure that it is put in the proper LaTeX preamble.

```
#+MACRO: preamble {{{when-fmt(html,\\($1\\))}}}{{{when-fmt(latex,\n#+LATEX_HEADER: $1\n)}}}
```

Note: the `\n` before `#+LATEX_HEADER:` and after the parameter `$1` are an *attempt* to ensure that the command starts and ends with a new line, so that Org mode commands are recognized correctly. This is not bulletproof: if you put a newline *inside* the command, that may still foul things up.

2 Definition Macros

Using the 1.2 macro, we specify macros that use the TeX and LaTeX macros to define macros. Once again, due to the use of an Emacs Lisp string, it is necessary to double any literal backslashes provided to these macros, as explained for the 1.1 macro.

2.1 def

The plain TeX command, which allows you to (re)define anything that (MathJax's implementation of) TeX can handle, including LaTeX. This flexibility comes at the price of no warnings when you attempt to define something new and it redefines something essential. The first argument is the control sequence to define, which may begin with a backslash (escaped: `\\`), but must not contain curly brackets (`{...}`); the second argument is what it should expand to, automatically enclosed in curly brackets.

```
#+MACRO: def {{{preamble(\\def$1{$2})}}}
```

2.2 newcommand

The standard LaTeX command for defining a macro, which *must not* already exist. The first argument is the command name, which is automatically enclosed in brackets. The second argument is what it should expand to, which is also automatically enclosed in brackets. The third argument, if any, is inserted *as is* between the first and second argument; this lets you give a parameter count, or a parameter count *and* a default value for the first parameter, each in square brackets (`[...]`).

```
#+MACRO: newcommand {{{preamble(\\newcommand{$1}$3{$2})}}}
```

2.3 renewcommand

The LaTeX command for redefining an existing macro, which *must* already exist. The arguments are the same as for the 2.2 macro.

```
#+MACRO: renewcommand {{{preamble(\\renewcommand{$1}$3{$2})}}}
```

2.4 newenvironment

The LaTeX command for defining a new environment, which *must not* already exist. The first, second, and third arguments are enclosed in curly brackets in that order. The fourth argument, if any, is inserted (as is) between the first and second arguments, like the third argument for the 2.2 macro.

```
#+MACRO: newenvironment {{{preamble(\\newenvironment{$1}$4{$2}{$3})}}}
```

2.5 renewenvironment

The LaTeX command for redefining an existing environment, which *must* already exist. The arguments are the same as for the 2.4 macro.

```
#+MACRO: renewenvironment {{{preamble(\\renewenvironment{$1}$4{$2}{$3})}}}
```

3 Installation

To use this setup file, you only need to "tangle" this document from within Emacs (press `C-c C-v t` in the default keymap), drop it into the same directory as your Org mode document(s), and load it with

```
#+SETUPFILE: define.setup
```

near the beginning of each file.

4 Usage Examples

4.1 TODO def

4.2 newcommand (without parameters)

Say you are writing a book on Intermediate Algebra, and introducing the vertex form of quadratic equations. And you find yourself referring to the equation

$$y = 2(x-1)^2 + 1$$

a lot, so you would like to abbreviate it. Easy! Anywhere (almost) *before* your first usage — we do it right before this text, to help keep it consistent with the example text and explanation here — just type

```
{{{newcommand(\\myvqeqn,y = 2(x-1)^2+1)}}}
```

and use it like so:

```
The equation $\myvqeqn$ shows ...
```

It will look like "The equation $y = 2(x-1)^2+1$ shows ...", in both HTML and LaTeX, the latter thanks to the use of `#+LATEX_HEADER:` in the 1.2 macro.

4.3 newcommand (with a parameter)

Say you are writing a book on Linear Algebra. You will certainly be writing a lot about the names of certain matrices, and you will probably want to abbreviate the names of (at least) the most common matrices you talk about. For two reasons, you will want to use a parameter for the matrix name:

1. you may be writing about several matrices, and it would be a waste of effort to write a separate command for each one;
2. you want to ensure that your typographical treatment of the various matrices is consistent, even if you change your mind later.

So, you might define a command like this:

```
{{{newcommand(\\mat,\\mathbf{#1},[1])}}}
```

and use it like this:

```
The matrices $\mat{A}$ and $\mat{B}$ represent ...
```

Now, the result looks like "The matrices $\mathbf{A}$ and $\mathbf{B}$ represent ...", but if you want to change how it looks later, there is only one definition that needs to be changed.

4.4 TODO renewcommand

4.5 TODO newenvironment

4.6 TODO renewenvironment