lisp-babel

Derek Feichtinger

December 29, 2015

Contents

1 Version information

```
(princ (concat (format "Emacs version: %s\n" (emacs-version))
               (format "org version: %s\n" (org-version))))
```

```
Emacs version: GNU Emacs 24.5.1 (x86_64-unknown-linux-gnu, GTK+ Version 3.10.8)
 of 2015-05-04 on dflt1w
org version: 8.2.10
```

2 using a table as input

We first create a table from a lisp **list of lists**. Each inner list will form a row in the resulting table. I already insert a header row with the names of three columns. A separator line can be obtained by putting the `hline` symbol into the resulting list.

```
(cons '(col1 col2 col3)
      (cons 'hline
            (loop for i from 5 to 15 collect `(,i ,(* i 5) ""))))
```

col1	col2	col3
5	25	
6	30	
7	35	
8	40	
9	45	
10	50	
11	55	
12	60	
13	65	
14	70	
15	75	

We now can fill the third column by passing the table into the next source block. We force babel to treat the first row as table header by using the **:colnames yes** header argument. This also causes the result table to contain the headers (as long as the new table has the same number of columns as the original table)

Here I also demonstrate the use of the **-n** option that will export the code with line numbers.

```
(let (result)
  (dolist (row tbl result)
    (setf (nth 2 row) (* 2 (nth 1 row)))
    (setq result (cons row result)))
  (reverse result))
```

col1	col2	col3
5	25	50
6	30	60
7	35	70
8	40	80
9	45	90
10	50	100
11	55	110
12	60	120
13	65	130
14	70	140
15	75	150

3 calling source blocks as a function

3.1 simple call syntax

We first define a function in a named code block called `mydouble`. The variable x will be passed in by defining a header argument `:var x`

```
(* 2 x)
```

Now we can call this babel function by using the code block's name `mydouble` from any place in the document. For example:

```
10
```

Another example where we pass in two variables x and y.

```
(/ x y)
```

Note that **you can/must pass additional header arguments** to the call. The ones added at the end influence the final result (e.g. putting it into a drawer), while the ones added in [] are evaluated in the context of the original definition (e.g whether to capture the output).

```
4
```

Another alternative calling syntax

```
5
```

Note that I **can have another piece of code implicitly called** by using its name as an input variable in a normal code block. So, I could directly fill the third column of our initial example table without ever having to print out that table table. We can just pass into the next function a variable `tbl` and the name of the initial code block `make-table1`.

```
(let (result)
  (dolist (row tbl result)
    (setf (nth 2 row) (* 2 (nth 1 row)))
    (setq result (cons row result)))
  (reverse result))
```

4 Inline src calls

This is the result of an inline src call in lisp: 15

and this is another: 15

```
15
```

5 Defining buffer wide variables for src blocks

One can use a verbatim block like this. I define a named block `myvar` and I pass it into the variable s of the following code block.

```
world
```

```
(concat "hello " s)
```

```
hello world
```

6 Using a :post function for table formatting

Often I produce multiple tables from a source block (e.g. printing several pandas data frames). These tables do not get aligned in the org document after the execution of the code block (even though they will get aligned upon exporting the document)

```
(princ
 (concat
  "#+CAPTION: Test1\n"
  "|A|B|C|\n"
  "|---\n"
  "|1|20|300|\n"
  "|200|30|4|\n"
  "\n#+CAPTION: Test2\n"
  "|A|B|C|\n"
  "|---\n"
  "|1|20|300|\n"
  "|200|30|4|\n"
  ))
```

Table 1: Test1

A	B	C
1	20	300
200	30	4

Table 2: Test2

A	B	C
1	20	300
200	30	4

The following function can be used in a :post argument for getting all tables in the output aligned, as shown further below

```
(with-temp-buffer
  (erase-buffer)
  (insert text)
  (beginning-of-buffer)
  (org-mode)
  (while
      (search-forward-regexp org-table-any-line-regexp nil t)
    (org-table-align)
    (goto-char (org-table-end)))
  (buffer-string))

(princ
 (concat
  "#+CAPTION: Test1\n"
  "|A|B|C|\n"
"|---\n"
"|1|20|300|\n"
"|200|30|4|\n"
"\n#+CAPTION: Test2\n"
"|A|B|C|\n"
"|---\n"
"|1|20|300|\n"
"|200|30|4|\n"
))
```

Table 3: Test1

A	B	C
1	20	300
200	30	4

Table 4: Test2

A	B	C
1	20	300
200	30	4

7 Problems, Questions

- ☐ How can I produce an initial table by code that already has a nicely separated (dashes) column name row? **:colnames yes** only produces such a table heading if a table of the same dimension was read in by the **:var** directive

Emacs 24.5.1 (Org mode 8.3.2)