chickenize

Arno Trautmann
arno.trautmann@gmx.de

July 4, 2011

Abstract

This is the documentation of the package `chickenize`. It allows you to substitute or change the contents of a LuaLaTeX document.[1] You have e. g. the possibility to substitute every word of a document with the word "chicken", translate it into chi speak, make it totally colorfull or use upper/lowercase all randomly. Of course this package is *not* meant for any serious document, but only for fun and – because we can!

If you have any suggestions or comments, just drop me a mail, I'll be happy to get any response!

Contents

1 Usage

This package should be useable some time …

[1] The code is based on pure LuaTeX features, so don't try to use it with any other TeX flavour.

2 Working Principle

We make use of LuaTEXs callbacks, especially the `pre_linebreak_filter` and the `post_linebreak_filter`. Hooking a function into these, we can chanke the input (into "chicken") or add/transform the input (putting color in, changing lower/uppercase).

2.1 Package Options

There surely will be some options etc.

3 Implementation

This is the README file that should contain some important information. So far I can only tell you to run the file chickenize.dtx to produce the three files chickenize.pdf (documentation) chickenize.sty (LaTeX user interface) chickenize.lua (Lua package code)

You need an up-to-date TeX Live (2011, if possible) to use this package.

For any comments or suggestions, contact me: arno dot trautmann at gmx dot de

Hope you have fun with this!

4 Preparation

Loading of packages and defition of constants. Will change somewhat when migrating to expl3 (?)

```
\RequirePackage{
  expl3,
  luatexbase,
  xkeyval,
  xparse
}
%% So far, no keys are needed.
\ExplSyntaxOn
\keys_define:nn {chick} {
  columns.tl_gset:N = \chick_cards_colums,
  columns.default:n = 2,
  printonly.code:n = \tl_set:Nn\chick_print_only{#1}\bool_set_true:N\chick_print_only_true,
  sectionsoncards.bool_set:N = \chick_sectionsoncards_true,
  german.tl_set:N = \chick_language,
}
\NewDocumentCommand\chicksetup{m}{
  \keys_set:nn{chick}{#1}
}
\directlua{dofile("chickenize.lua")}
```

5 Definition of Macros

```
%
```

6 Lua Module

This file contains all the necessary functions.

```
local HLIST = node.id("hlist")
local RULE = node.id("rule")
local GLUE = node.id("glue")
local WHAT = node.id("whatsit")
local COL = node.subtype("pdf_colorstack")
local GLYPH = node.id("glyph")

local color_push = node.new(WHAT,COL)
local color_pop = node.new(WHAT,COL)
color_push.stack = 0
color_pop.stack = 0
color_push.cmd = 1
color_pop.cmd = 2

uppercasecolor = function (head)
  for line in node.traverse_id(HLIST,head) do
    for upper in node.traverse_id(GLYPH,line.list) do
      if (((upper.char > 64) and (upper.char < 91)) or
          ((upper.char > 57424) and (upper.char < 57451)))  then  -- for small caps! nice
        color_push.data = math.random()..math.random()..math.random().." rg"
        line.head = node.insert_before(line.list,upper,node.copy(color_push))
        node.insert_after(line.list,upper,node.copy(color_pop))
      end
    end
  end
  return head
end

randomuclc = function(head)
  for i in node.traverse_id(37,head) do
    if math.random() < 0.5 then
      i.char = tex.uccode[i.char]
    else
      i.char = tex.lccode[i.char]
      i.yoffset = "15 pt"
end
  end
  return head
end

function chickenize(head)
  for i in node.traverse_id(37,head) do  --find start of a word
```

```
63    while ((i.next.id == 37) or (i.next.id == 11) or (i.next.id == 7) or (i.next.id == 0)) do
64      i.next = i.next.next
65    end
66
67    chicken = {}
68    chicken[0] = node.new(37,1)
69    for i = 1,8 do
70      chicken[i] = node.new(37,1)
71      chicken[i].font = font.current()
72    end
73    node.insert_before(head,i,chicken[1])
74
75    chicken[8].char = 67
76    chicken[7].char = 104
77    chicken[6].char = 105
78    chicken[5]  = node.new(7,3)
79    hyphennode = node.new(37,1)
80    hyphennode.char = 67
81    hyphennode.font = font.current()
82    chicken[5].pre = hyphennode
83    chicken[5].post = hyphennode
84    chicken[4].char = 99
85    chicken[3].char = 107
86    chicken[2].char = 101
87    chicken[1].char = 110
88
89    for k = 1,7 do
90      node.insert_before(head,chicken[k],chicken[k+1])
91    end
92    chicken[1].next = i.next
93  end
94
95  return head
96 end
```

7 Known Bugs

There are surely some bugs …

???

8 To Dos

Some things that should be implemented but aren't so far or are very poor at the moment:

?