

The impnattyto package

Raphaël Pinson
raphink@gmail.com

1.5 from 2019/03/04

1 Introduction

When it comes to French typography, the *Lexique des règles typographiques en usage de l’Imprimerie Nationale* is a definite reference.

While the majority of the recommendations of this book has been implemented in the frenchb module for babel, other recommendations still deserve to be automatized in order to be implemented in L^AT_EX.

Such is the original goal of this package, initiated by a question on the tex.stackexchange.com¹ website, and which implements several of the rules listed in this booklet so as to make them more easily applicable to texts edited with L^AT_EX.

As this package grew, functionalities were added, including some that were not directly related to the booklet, but improved the typographic quality of documents.

2 Usage

In order to use the impnattyto package, use the following line:

```
\usepackage[<options>]{impnattyto}
```

The package options are described in the following sections.

2.1 Hyphenation

hyphenation Besides the general hyphenation rules, the booklet indicates that we should “prevent hyphenation of words on more than two consecutive lines.”

In order to simplify the code, the suggested implementation strongly discourages hyphenation at the end of pages, as well as hyphenation on two consecutive lines.

To active this functionality, use the hyphenation option:

```
\usepackage[hyphenation]{impnattyto}
```

¹<http://tex.stackexchange.com/questions/20493/french-typography-recommendations>

2.2 Paragraph formatting

`parindent` The booklet advises to indent paragraphs by 1em. This `\parindent` setting can be achieved by using the `parindent` option:

```
\usepackage[parindent]{imnatty}
```

`lastparline` Moreover, it is indicated in the “Hyphenation” section that “the last line of a paragraph must contain a word or the end of a word of a width at least equal to the double of the indent of the next paragraph.” Since implementing this solution exactly is quite tricky, the `lastparline` option ensures that the last line of a paragraph is at least as long as the double value of `\parindent`.²

When Lua \TeX is used, the solution provided by Patrick Gundlach³ is used. With other rendering engines, it is the native solution provided by Enrico Gregorio⁴ that serves as an implementation:

```
\usepackage[lastparline]{imnatty}
```

When the `draft` option is activated and Lua \TeX is used, the inserted ties are colored in teal. The color can be tuned with the `lastparlinecolor` option.

`nosingleletter` It is also recommended to avoid hyphenation points that would isolate a single letter. The solution proposed by Patrick Gundlach⁵ allows to fix this by using Lua \TeX . To activate this functionality, you can use the `nosingleletter` option:

```
\usepackage[nosingleletter]{imnatty}
```

When this option is activated, only Lua \TeX (with the `lualatex` command) can render the document.

When the `draft` option is activated, the inserted ties are colored in brown. The color can be tuned by setting the `nosinglelettercolor` option.

`homeoarchy` When two consecutive lines begin (homeoarchy) or end (homoioteleuton) with the same word or series of letters, it can confuse the reader, so this has to be avoided.

Fixing this problem automatically is very complex and generally not a good idea.⁶ For this reason, the `homeoarchy` option in this package only detects and highlights them. Fixing them will usually be a matter of introducing ties in the paragraph:

```
\usepackage[homeoarchy]{imnatty}
```

When this option is activated, only Lua \TeX (with the `lualatex` command) can render the document.

This option is only effective if the `draft` option is activated.

The inserted ties are colored with two colors:

²<http://tex.stackexchange.com/questions/28357/ensure-minimal-length-of-last-line>

³<http://tex.stackexchange.com/questions/28357/ensure-minimal-length-of-last-line/28361#28361>

⁴<http://tex.stackexchange.com/questions/28357/ensure-minimal-length-of-last-line/28358#28358>

⁵<http://tex.stackexchange.com/questions/27780/one-letter-word-at-the-end-of-line>

⁶<http://tex.stackexchange.com/questions/27588/repetition-of-a-word-on-two-lines>

- Entire words are colored in **red** and this color can be set with the `homearchywordcolor` option;
- Partial words are colored in **orange** and this color can be set by means of the `homearchycharcolor` option;

A glyph sequence is considered problematic when:

- The number of entire matching words is greater than 1. This parameter can be tuned with the `homearchymaxwords` option;
- The number of matching characters is greater than 3. This parameter can be tuned with the `homearchymaxchars` option;

`rivers` A river is a vertical alignment of spaces in a paragraph. The `rivers` option allows to color rivers so as to identify them. This option does not fix the detected rivers:

```
\usepackage[rivers]{imnatty}
```

When this option is activated, only LuaTeX (with the `lualatex` command) can render the document.

This option is only effective if the `draft` option is activated.

The inserted ties are colored in **lime**. This color can be tuned by means of the `riverscolor` option.

2.3 Chapter numbering

`frenchchapters` When it comes to chapter numbering, the booklet indicates: “In a title, chapter numbers are typeset in roman capital numbers, except for the ordinal ‘premier’ written in letters in spite of the current fashion to write it in the cardinal form Chapter I.”

The `frenchchapters` option of the package implements this recommendation:

```
\usepackage[frenchchapters]{imnatty}
```

Should you wish to use the ordinal form ‘premier’ without using roman numbers for chapter numbering, you can redefine the `frenchchapter` macro, for example:

```
\let\frenchchapter\arabic % use arabic numbers
\let\frenchchapter\babylonian % use babylonian numbers
```

2.4 Widows and Orphans

It is recommended not to leave widows and orphans in a document. For this reason, we recommend you use the `nowidow` package:

```
\usepackage[all]{nowidow}
```

See the package documentation for more options.

2.5 Draft mode

The `impnatty` package features a draft mode allowing to visualize the penalties (ties) inserted by the `nosingletter` and `lastparline` options, as well as the information added by the `homeoarchy` and `rivers` options. In draft mode, places where ties were inserted are indicated by colored squares.

To activate the draft mode, use the `draft` option, for example:

```
\usepackage[draft,lastparline]{impnatty}
```

This document is generated with the `draft` option on in order to demonstrate its effects.

3 Implementation

```
1 \ProvidesPackage{impnatty}
2 \RequirePackage{ifluatex}
3 \RequirePackage{kvoptions}
4 \SetupKeyvalOptions{
5   family=impnatty,
6   prefix=int,
7 }
8 \DeclareBoolOption{draft}
9 \DeclareBoolOption{frenchchapters}
10 \DeclareBoolOption{hyphenation}
11 \DeclareBoolOption{nosingletter}
12 \DeclareBoolOption{parindent}
13 \DeclareBoolOption{lastparline}
14 \DeclareBoolOption{homeoarchy}
15 \DeclareBoolOption{rivers}
16 \DeclareStringOption[red]{homeoarchywordcolor}
17 \DeclareStringOption[orange]{homeoarchycharcolor}
18 \DeclareStringOption[brown]{nosinglettercolor}
19 \DeclareStringOption[teal]{lastparlinecolor}
20 \DeclareStringOption[lime]{riverscolor}
21 \DeclareStringOption[1]{homeoarchymaxwords}
22 \DeclareStringOption[3]{homeoarchymaxchars}
23 \ProcessKeyvalOptions*
24 \RequirePackage{xcolor}
25 \def\usecolor#1{\csname\string\color@#1\endcsname\space}
26 \ifinthyphenation
27   \brokenpenalty=10000
28   \doublehyphendemerits=100000000
29 \fi
30 \ifintfrenchchapters
31   \let\frenchchapter\Roman
32   \renewcommand{\thechapter}{%
```

No page finishes with an hyphenated word

Discourage hyphenation on two lines in a row

Number chapters

No single letter

```
33 \ifnum\value{chapter}=1
34   premier%
35 \else
36   \frenchchapter{chapter}%
37 \fi
38 }
39 \fi

40 \ifintnosingleletter
41   \ifluatex
42     \RequirePackage{luatexbase,luacode}
43     \begin{luacode}
44       local glyph_id = node.id "glyph"
45       local glue_id = node.id "glue"
46       local hlist_id = node.id "hlist"
47
48       local prevent_single_letter = function (head)
49         while head do
50           if head.id == glyph_id then
51             if unicode.utf8.match(unicode.utf8.char(head.char),"%a") then -- glyph
52               if head.prev.id == glue_id and head.next.id == glue_id then -- only i
53
54                 local p = node.new("penalty")
55                 p.penalty = 10000
56
57                 \ifintdraft
58                   local w = node.new("whatsit","pdf_literal")
59                   w.data = "q \usecolor{\intnosinglelettercolor} 0 0 m 0 5 1 2 5 1 2 0 1 b Q"
60
61                   node.insert_after(head,head,w)
62                   node.insert_after(head,w,p)
63                 \else
64                   node.insert_after(head,head,p)
65                 \fi
66               end
67             end
68           end
69           head = head.next
70         end
71         return true
72       end
73
74       luatexbase.add_to_callback("pre_linebreak_filter",prevent_single_letter,"~")
75     \end{luacode}
76 \else
77   \PackageError{The nosingleletter option only works with LuaTeX}
78 \fi
79 \fi

80 \ifintparindent
81 \setlength{\parindent}{1em}
```

Paragraph indentation

Last line of paragraph

```
82 \fi
83 \ifintlastparline
84   \ifluatex
85     \RequirePackage{luatexbase,luacode}
86     \begin{luacode}
87       local glyph_id = node.id "glyph"
88       local glue_id = node.id "glue"
89       local hlist_id = node.id "hlist"
90
91       last_line_twice_parindent = function (head)
92         while head do
93           local _w,_h,_d = node.dimensions(head)
94           if head.id == glue_id and head.subtype ~= 15 and (_w < 2 * tex.parindent) then
95
96             -- we are at a glue and have less than 2*\parindent to go
97             local p = node.new("penalty")
98             p.penalty = 10000
99
100            \ifintdraft
101              local w = node.new("whatsit","pdf_literal")
102              w.data = "q \usecolor{\intlastparlinecolor} 0 0 m 0 5 1 2 5 1 2 0 1 b Q"
103
104              node.insert_after(head,head.prev,w)
105              node.insert_after(head,w,p)
106            \else
107              node.insert_after(head,head.prev,p)
108            \fi
109          end
110
111          head = head.next
112        end
113        return true
114      end
115
116      luatexbase.add_to_callback("pre_linebreak_filter",last_line_twice_parindent,"lastparline")
117    \end{luacode}
118  \else
119    \setlength{\parfillskip}{0pt plus\dimexpr\textwidth-2\parindent}
120  \fi
121 \fi
```

Detect homeoarchies

```
122 \ifinthomeoarchy
123 \ifintdraft
124   \ifluatex
125     \RequirePackage{luatexbase,luacode}
126     \begin{luacode}
127       local glyph_id = node.id "glyph"
128       local glue_id = node.id "glue"
129       local hlist_id = node.id "hlist"
130
```

```

131 compare_lines = function (line1,line2)
132     local head1 = line1.head
133     local head2 = line2.head
134
135     local char_count = 0
136     local word_count = 0
137
138     while head1 and head2 do
139         if (head1.id == glyph_id and head2.id == glyph_id
140             and head1.char == head2.char)           -- identical glyph
141             or (head1.id == glue_id and head2.id == glue_id) then -- glue
142
143                 if head1.id == glyph_id then -- glyph
144                     char_count = char_count + 1
145                 elseif char_count > 0 and head1.id == glue_id then -- glue
146                     word_count = word_count + 1
147                 end
148                 head1 = head1.next
149                 head2 = head2.next
150             elseif (head1.id == 0 or head2.id == 0) then -- end of line
151                 break
152             elseif (head1.id ~= glyph_id and head1.id ~= glue_id) then -- some other kind of nod
153                 head1 = head1.next
154             elseif (head2.id ~= glyph_id and head2.id ~= glue_id) then -- some other kind of nod
155                 head2 = head2.next
156             else -- no match, no special node
157                 break
158             end
159         end
160         -- analyze last non-matching node, check for punctuation
161         if ((head1 and head1.id == glyph_id and head1.char > 49)
162             or (head2 and head2.id == glyph_id and head2.char > 49)) then
163             -- not a word
164         elseif char_count > 0 then
165             word_count = word_count + 1
166         end
167         return char_count,word_count,head1,head2
168     end
169
170 compare_lines_reverse = function (line1,line2)
171     local head1 = node.tail(line1.head)
172     local head2 = node.tail(line2.head)
173
174     local char_count = 0
175     local word_count = 0
176
177     while head1 and head2 do
178         if (head1.id == glyph_id and head2.id == glyph_id
179             and head1.char == head2.char)           -- identical glyph
180             or (head1.id == glue_id and head2.id == glue_id) then -- glue

```

```

181
182     if head1.id == glyph_id then -- glyph
183         char_count = char_count + 1
184     elseif char_count > 0 and head1.id == glue_id then -- glue
185         word_count = word_count + 1
186     end
187     head1 = head1.prev
188     head2 = head2.prev
189     elseif (head1.id == 0 or head2.id == 0) then -- start of line
190         break
191     elseif (head1.id ~= glyph_id and head1.id ~= glue_id) then -- some other kind of node
192         head1 = head1.prev
193     elseif (head2.id ~= glyph_id and head2.id ~= glue_id) then -- some other kind of node
194         head2 = head2.prev
195     elseif (head1.id == glyph_id and head1.char < 48) then -- punctuation
196         head1 = head1.prev
197     elseif (head2.id == glyph_id and head2.char < 48) then -- punctuation
198         head2 = head2.prev
199     else -- no match, no special node
200         break
201     end
202 end
203 -- analyze last non-matching node, check for punctuation
204 if ((head1 and head1.id == glyph_id and head1.char > 49)
205     or (head2 and head2.id == glyph_id and head2.char > 49)) then
206     -- not a word
207 elseif char_count > 0 then
208     word_count = word_count + 1
209 end
210 return char_count,word_count,head1,head2
211 end
212
213 highlight = function (line,nend,color)
214     local n = node.new("whatsit","pdf_literal")
215
216     -- get dimensions
217     local w,h,d = node.dimensions(line.head,nend)
218     local w_pts = w/65536 -- scaled points to points
219
220     -- set data
221     n.data = "q " .. color .. " 0 0 m 0 5 l " .. w_pts .. " 5 l " .. w_pts .. " 0 l b Q"
222
223     -- insert node
224     n.next = line.head
225     line.head = n
226     node.slide(line.head)
227 end
228
229 highlight_reverse = function (nstart,line,color)
230     local n = node.new("whatsit","pdf_literal")

```



```

231
232
233     -- get dimensions
234     local w,h,d = node.dimensions(nstart,node.tail(line.head))
235     local w_pts = w/65536 -- scaled points to points
236
237     -- set data
238     n.data = "q " .. color .. " 0 0 m 0 5 l " .. w_pts .. " 5 l " .. w_pts .. " 0 l b Q"
239
240     -- insert node
241     node.insert_after(line.head,nstart,n)
242 end
243
244 homeoarchy = function (head)
245     local cur_line = head
246     local prev_line -- initiate prev_line
247
248     local max_char = tonumber(\inhomeoarchymaxchars)
249     local max_word = tonumber(\inhomeoarchymaxwords)
250
251     while head do
252         if head.id == hlist_id then -- new line
253             prev_line = cur_line
254             cur_line = head
255             if prev_line.id == hlist_id then
256                 -- homeoarchy
257                 char_count,word_count,prev_head,cur_head = compare_lines(prev_line,cur_line)
258                 if char_count >= max_char or word_count >= max_word then
259                     local color
260                     if word_count >= max_word then
261                         color = "q \usecolor{\inhomeoarchywordcolor}"
262                     else
263                         color = "q \usecolor{\inhomeoarchycharcolor}"
264                     end
265
266                     -- highlight both lines
267                     highlight(prev_line,prev_head,color)
268                     highlight(cur_line,cur_head,color)
269                 end
270             end
271         end
272         head = head.next
273     end
274     return true
275 end
276
277 luatexbase.add_to_callback("post_linebreak_filter",homeoarchy,"homeoarchy")
278
279 homoioteleuton = function (head)
280     local cur_line = head

```

```

281     local prev_line -- initiate prev_line
282
283     local max_char = tonumber(\inhomeoarchymaxchars)
284     local max_word = tonumber(\inhomeoarchymaxwords)
285
286     local linecounter = 0
287
288     while head do
289         if head.id == hlist_id then -- new line
290             linecounter = linecounter + 1
291             if linecounter > 1 then
292                 prev_line = cur_line
293                 cur_line = head
294                 if prev_line.id == hlist_id then
295                     -- homoioteleuton
296                     char_count,word_count,prev_head,cur_head = compare_lines_reverse(prev_line,cur_line)
297                     if char_count >= max_char or word_count >= max_word then
298                         local color
299                         if word_count >= max_word then
300                             color = "q \usecolor{\inhomeoarchywordcolor}"
301                         else
302                             color = "q \usecolor{\inhomeoarchycharcolor}"
303                         end
304
305                         -- highlight both lines
306                         highlight_reverse(prev_head,prev_line,color)
307                         highlight_reverse(cur_head,cur_line,color)
308                     end
309                 end
310             end
311         end
312         head = head.next
313     end
314
315     return true
316 end
317
318     luatexbase.add_to_callback("post_linebreak_filter",homoioteleuton,"homoioteleuton")
319 \end{luacode}
320 \else
321     \PackageError{The homeoarchy option only works with LuaTeX}
322 \fi
323 \fi
324 \fi
325 \ifintrivers
326 \ifintdraft
327 \ifluatex
328     \RequirePackage{luatexbase,luacode}
329     \begin{luacode}

```

Detect rivers

```

330 local glyph_id = node.id "glyph"
331 local glue_id = node.id "glue"
332 local hlist_id = node.id "hlist"
333
334 river_analyze_line = function(line,dim1,dim2,precision)
335     local head = line.head
336
337     while head do
338         if head.id == glue_id then -- glue node
339             local w1,h1,d1 = node.dimensions(line.glue_set,line.glue_sign,line.glue_order,line.head)
340             local w2,h2,d2 = node.dimensions(line.glue_set,line.glue_sign,line.glue_order,line.head)
341             --print("dim1: "..dim1..""; dim2: "..dim2..""; w1: "..w1..""; w2: "..w2")
342             if w1 > dim2 + precision then -- out of range
343                 return false,head
344             elseif w1 < (dim2 + precision) and w2 > (dim1 - precision) then -- found
345                 return true,head
346             end
347         end
348         head = head.next
349     end
350
351     return false,head
352 end
353
354 rivers = function (head)
355     local prev_prev_line
356     local prev_line
357     local cur_line = head
358     local cur_node
359     local char_count
360
361     local linecounter = 0
362
363     while head do
364         if head.id == hlist_id then -- new line
365             linecounter = linecounter + 1
366             prev_prev_line = prev_line
367             prev_line = cur_line
368             cur_line = head
369             if linecounter > 2 then
370                 cur_node = cur_line.head
371                 char_count = 0
372
373                 while cur_node do
374                     if cur_node.id == glyph_id then -- glyph
375                         char_count = char_count + 1
376                     elseif cur_node.id == glue_id and char_count > 0 and cur_node.next then -- glue
377                         -- prev_line
378                         local w1,h1,d1 = node.dimensions(head.glue_set,head.glue_sign,head.glue_order)
379                         local w2,h2,d2 = node.dimensions(head.glue_set,head.glue_sign,head.glue_order)

```

```

380         -- if we allow up to 45° diagonal rivers, then there can be up to + or - line
381         local w_p,h_p,d_p = node.dimensions(prev_line.head,cur_line.head) -- calculat
382         found_p,head_p = river_analyze_line(prev_line,w1,w2,h_p)
383
384         if found_p then
385             -- prev_prev_line
386             local w1,h1,d1 = node.dimensions(prev_line.glue_set,prev_line.glue_sign,pr
387             local w2,h2,d2 = node.dimensions(prev_line.glue_set,prev_line.glue_sign,pr
388             -- if we allow up to 45° diagonal rivers, then there can be up to + or - 1
389             local w_p,h_p,d_p = node.dimensions(prev_prev_line.head,prev_line.head) --
390             found_pp,head_pp = river_analyze_line(prev_prev_line,w1,w2,h_p)
391
392             if found_pp then
393                 local n_pp = node.new("whatsit","pdf_literal")
394                 n_pp.data = "q \usecolor{\intriverscolor} 0 0 m 0 5 1 5 5 1 5 0 1 b Q"
395                 node.insert_after(prev_prev_line,head_pp.prev,n_pp)
396
397                 local n_p = node.new("whatsit","pdf_literal")
398                 n_p.data = "q \usecolor{\intriverscolor} 0 0 m 0 5 1 5 5 1 5 0 1 b Q"
399                 node.insert_after(prev_line,head_p.prev,n_p)
400
401                 local n_c = node.new("whatsit","pdf_literal")
402                 n_c.data = "q \usecolor{\intriverscolor} 0 0 m 0 5 1 5 5 1 5 0 1 b Q"
403                 node.insert_after(cur_line,cur_node.prev,n_c)
404             end
405         end
406     end
407     cur_node = cur_node.next
408 end
409 end
410 end
411 head = head.next
412 end
413
414 return true
415
416 end
417
418
419 luatexbase.add_to_callback("post_linebreak_filter",rivers,"rivers")
420     \end{luacode}
421 \else
422     \PackageError{The homearchy option only works with LuaTeX}
423 \fi
424 \fi
425 \fi

```

Change History

0.1	General: First version	1	0.9	General: River detection returns false by default	1
0.2	General: Add nosingleletter option	1	1.0	General: Improve documentation, simplify internal variables	1
0.3	General: Add parindent and lastparline options	1	1.1	General: Fix French documentation	1
0.4	General: Add draft mode	1	1.2	General: Fix French documentation	1
0.5	General: Add homearchy detection	1	1.3	General: Fix French documentation	1
0.6	General: Words contain at least one character	1	1.4	General: Fix release date	1
0.7	General: Add homoioteleuton detection	1	1.5	General: Fix support for <code>TeXLive 2016</code> (new <code>luatex</code> compatibility). Thanks to Michal Hoftich	1
0.8	General: Add river detection	1			

Index

Numbers written in *italic* refer to the page where the corresponding entry is described; numbers underlined refer to the code line of the definition; numbers in roman refer to the code lines where the entry is used.

B	E	<code>\ifintfrenchchapters</code>	30
<code>\begin</code>	<code>\else</code>	<code>\ifinthehomearchy</code>	122
43, 86, 126, 329	35, 63, 76, 106, 118, 320, 421	<code>\ifinthyphenation</code>	26
<code>\brokenpenalty</code>	<code>\end</code>	<code>\ifintlastparline</code>	83
27	75, 117, 319, 420	<code>\ifintnosingleletter</code>	40
C	<code>\endcsname</code>	<code>\ifintparindent</code>	80
<code>\color@</code>	25	<code>\ifintrivers</code>	325
25	F	<code>\ifluatex</code>	41, 84, 124, 327
<code>\csname</code>	<code>\fi</code> 29, 37, 39, 65, 78, 79, 82, 108, 120, 121, 322–324, 423–425	<code>\ifnum</code>	33
D	<code>\frenchchapter</code>	<code>\inthehomearchycharcolor</code>	263, 302
<code>\DeclareBoolOption</code>	<code>\frenchchapters</code>	<code>\inthehomearchymaxchars</code>	248, 283
8–15	3	<code>\inthehomearchymaxwords</code>	249, 284
<code>\DeclareStringOption</code>	H	<code>\inthehomearchywordcolor</code>	261, 300
16–22	<code>\homearchy</code>	<code>\intlastparlinecolor</code> 102	
<code>\def</code>	<code>\hyphenation</code>		
25	1		
<code>\dimexpr</code>	I		
119	<code>\ifintdraft</code>		
<code>\doublehyphendemerits</code>		
28	57, 100, 123, 326		

<code>\intnosinglelettercolor</code>	<code>\parindent</code> . 2, 81, 96, 119	<code>\space</code> 25
..... 59	<code>\ProcessKeyvalOptions</code>	<code>\string</code> 25
<code>\intriverscolor</code> 23	
..... 394, 398, 402	<code>\ProvidesPackage</code> 1	T
		<code>\textwidth</code> 119
L	R	<code>\thechapter</code> 32
<code>\lastparline</code> 2	<code>\renewcommand</code> 32	
<code>\let</code> 31	<code>\RequirePackage</code> .. 2,	U
	3, 24, 42, 85, 125, 328	<code>\usecolor</code> 25,
N	<code>\rivers</code> 3	59, 102, 261, 263,
<code>\nosingleletter</code> 2	<code>\Roman</code> 31	300, 302, 394, 398, 402
P	S	V
<code>\PackageError</code> 77, 321, 422	<code>\setlength</code> 81, 119	<code>\value</code> 33
<code>\parfillskip</code> 119	<code>\SetupKeyvalOptions</code> . 4	