

# Le paquet impnattypo

Raphaël Pinson  
raphink@gmail.com

1.1 en date du 2013/06/17

## 1 Introduction

En matière de typographie française, le *Lexique des règles typographiques en usage de l'Imprimerie Nationale* est une référence incontournable.

Si la majorité des recommandations de cet ouvrage est implémentée dans le module frenchb pour babel, certaines autres recommandations méritent encore d'être automatisées pour être implémentées en L<sup>A</sup>T<sub>E</sub>X.

C'est le but original de ce paquet, initié par une question sur le site tex.stackexchange.com<sup>1</sup>, et qui implémente plusieurs règles édictées dans ce lexique afin de les rendre plus facilement applicables aux textes édités avec L<sup>A</sup>T<sub>E</sub>X.

Au fur et à mesure que ce paquet a grandi, des fonctionnalités sont venues s'ajouter, dont certaines ne sont pas directement liées au lexique, mais améliorent la qualité typographique des documents.

## 2 Utilisation

Pour utiliser le paquet impnattypo, entrez la ligne :

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

Les options du paquet sont décrites dans les sections suivantes.

### 2.1 Césures

hyphenation En dehors des règles générales de coupure des mots, le lexique indique qu'il faut « [éviter] les coupures de mots sur plus de trois lignes consécutives. »

Afin de simplifier le code, l'implémentation proposée décourage fortement les césures en fin de page, ainsi que les césures sur deux lignes consécutives.

Pour activer cette fonctionnalité, utilisez l'option `hyphenation` :

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

---

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

## 2.2 Formatage des paragraphes

`parindent` Le lexique conseille une indentation des paragraphes de 1em. Ce réglage de `\parindent` peut être obtenu par l'utilisation de l'option `parindent`:

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

`lastparline` De plus, il est indiqué dans la section « Coupure des mots » que « la dernière ligne d'un alinéa doit comporter un mot ou une fin de mot de longueur au moins égale au double du renforcement de l'alinéa suivant. » À défaut d'implémenter exactement cette solution, l'option `lastparline` s'assure que la dernière ligne d'un alinéa est au moins aussi longue que le double de la valeur de `\parindent`.<sup>2</sup>

Lorsque LuaTeX est utilisé, la solution de Patrick Gundlach<sup>3</sup> est utilisée. Avec les autres moteurs de rendu, c'est la solution native de Enrico Gregorio<sup>4</sup> qui fait office d'implémentation:

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

Lorsque l'option `draft` est activée et que LuaTeX est utilisé, les espaces insécables insérés sont colorés en teal. La couleur utilisée peut être ajustée par l'option `lastparlinecolor`.

`nosingleletter` Il est également recommandé d'éviter les coupures isolant une lettre. La solution proposée par Patrick Gundlach<sup>5</sup> permet de remédier à cela en utilisant LuaTeX. Pour activer cette fonctionnalité, il faut utiliser l'option `nosingleletter`:

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

Lorsque cette option est activée, seul LuaTeX (via la commande `lualatex`) pourra effectuer le rendu du document.

Lorsque l'option `draft` est activée, les espaces insécables insérés sont colorés en brown. La couleur utilisée peut être ajustée par l'option `nosinglelettercolor`.

`homeoarchy` Lorsque deux lignes consécutives commencent ou finissent par le même mot ou la même série de lettres, cela peut induire le lecteur en erreur et cela est donc à éviter.

La correction automatique de ce phénomène est très complexe et en général non souhaitable.<sup>6</sup> C'est pourquoi l'option `homeoarchy` de ce paquet se contente de les détecter et de les afficher. Leur correction consistera en général en l'introduction d'une espace insécable dans le paragraphe:

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

---

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>

Lorsque cette option est activée, seul LuaTeX (via la commande `lualatex`) pourra effectuer le rendu du document.

Cette option n'est effective que si l'option `draft` est activée.

Les espaces insécables insérées sont colorées de deux couleurs :

- Les mots entiers sont colorés en `red` et cette couleur peut être ajustée par l'option `homeoarchywordcolor` ;
- Les mots partiels sont colorés en `orange` et cette couleur peut être ajustée par l'option `homeoarchycharcolor` ;

Une séquence de glyphes est considérée comme problématique si :

- Le nombre de mots entiers matchant est supérieur à `1`. Ce paramètre peut être ajusté par l'option `homeoarchymaxwords` ;
- Le nombre de caractères matchant est supérieur à `3`. Ce paramètre peut être ajusté par l'option `homeoarchymaxchars` ;

`rivers` Une lézarde est un alignement vertical d'espaces dans un paragraphe. L'option `rivers` permet de colorer les lézardes afin de les identifier. Cette option ne corrige pas les lézardes détectées :

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

Lorsque cette option est activée, seul LuaTeX (via la commande `lualatex`) pourra effectuer le rendu du document.

Cette option n'est effective que si l'option `draft` est activée.

Les espaces insécables insérées sont colorées en `lime`. Cette couleur peut être ajustée par l'option `riverscolor`.

## 2.3 Numérotation des chapitres

`frenchchapters` Concernant la numérotation des chapitres, le lexique indique : « Dans un titre, on compose en chiffres romains grandes capitales les numéros de chapitres, à l'exception de l'ordinal « premier » en toutes lettres malgré la tendance actuelle qui tend à lui substituer la forme cardinale Chapitre I. »

L'option `frenchchapters` du paquet implémente cette recommandation :

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

Si vous souhaitez bénéficier de la forme ordinale « premier » sans pour autant utiliser une numérotation des chapitres en chiffres romains, il est possible de redéfinir la macro `frenchchapter`, par exemple :

```
\let\frenchchapter\arabic % numérotation en chiffres arabes
\let\frenchchapter\babylonian % numérotation en chiffres babyloniens
```

## 2.4 Lignes orphelines et veuves

Il est fortement recommandé de ne pas laisser de lignes orphelines dans un document. Pour cela, nous vous conseillons d'utiliser le paquet `nowidow`:

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

Voir la documentation de ce paquet pour plus d'options.

## 2.5 Mode brouillon

`draft` Le paquet `impnatty` dispose d'un mode brouillon permettant de visualiser les pénalités (espaces insécables) ajoutés par les options `nosingleletter` et `lastparline`, ainsi que les informations ajoutées par les options `homeoarchy` et `rivers`. En mode brouillon, les emplacements des espaces insécables insérés sont marqués par des rectangles de couleur.

Pour activer le mode brouillon, utilisez l'option `draft`, par exemple:

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

Cette document est générée avec l'option `draft` afin d'en montrer les effets.

## 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{nosingleletter}
12 \DeclareBoolOption{parindent}
13 \DeclareBoolOption{lastparline}
14 \DeclareBoolOption{homeoarchy}
15 \DeclareBoolOption{rivers}
16 \DeclareStringOption[red]{homeoarchywordcolor}
17 \DeclareStringOption[orange]{homeoarchycharcolor}
18 \DeclareStringOption[brown]{nosinglelettercolor}
19 \DeclareStringOption[teal]{lastparlinecolor}
20 \DeclareStringOption[lime]{riverscolor}
21 \DeclareStringOption[1]{homeoarchymaxwords}
22 \DeclareStringOption[3]{homeoarchymaxchars}
23 \ProcessKeyvalOptions*
```

No page finishes with an hyphenated word  
 Discourage hyphenation on two lines in a row  
 Number chapters

No single letter

```

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}{%
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
45       local prevent_single_letter = function (head)
46         while head do
47           if head.id == 37 then
48             if unicode.utf8.match(unicode.utf8.char(head.char),"%a") then
49               if head.prev.id == 10 and head.next.id == 10 then
50
51                 local p = node.new("penalty")
52                 p.penalty = 10000
53
54                 \ifintdraft
55                   local w = node.new("whatsit","pdf_literal")
56                   w.data = "q \usecolor{\intnosinglelettercolor} 0 0 m 0 5 1 2 5 1 2 0 1 b Q"
57
58                   node.insert_after(head,head,w)
59                   node.insert_after(head,w,p)
60                 \else
61                   node.insert_after(head,head,p)
62                 \fi
63               end
64             end
65           end
66           head = head.next
67         end
68         return true
69       end
70
71       luatexbase.add_to_callback("pre_linebreak_filter",prevent_single_letter,"~")

```

Paragraph indentation

Last line of paragraph

Detect homeoarchies

```
72     \end{luacode}
73     \else
74         \PackageError{The nosingleletter option only works with LuaTeX}
75     \fi
76 \fi

77 \ifintparindent
78 \setlength{\parindent}{1em}
79 \fi

80 \ifintlastparline
81     \ifluatex
82         \RequirePackage{luatexbase,luacode}
83         \begin{luacode}
84             last_line_twice_parindent = function (head)
85                 while head do
86                     local _w,_h,_d = node.dimensions(head)
87                     if head.id == 10 and head.subtype ~= 15 and (_w < 2 * tex.parindent) then
88
89                         -- we are at a glue and have less than 2*\parindent to go
90                         local p = node.new("penalty")
91                         p.penalty = 10000
92
93                         \ifintdraft
94                             local w = node.new("whatsit","pdf_literal")
95                             w.data = "q \usecolor{\intlastparlinecolor} 0 0 m 0 5 1 2 5 1 2 0 1 b Q"
96
97                             node.insert_after(head,head.prev,w)
98                             node.insert_after(head,w,p)
99                         \else
100                            node.insert_after(head,head.prev,p)
101                        \fi
102                    end
103
104                    head = head.next
105                end
106                return true
107            end
108
109            luatexbase.add_to_callback("pre_linebreak_filter",last_line_twice_parindent,"lastparline"
110            \end{luacode}
111        \else
112            \setlength{\parfillskip}{0pt plus\dimexpr\textwidth-2\parindent}
113        \fi
114 \fi

115 \ifinthomeoarchy
116 \ifintdraft
117     \ifluatex
118         \RequirePackage{luatexbase,luacode}
119         \begin{luacode}
120             compare_lines = function (line1,line2)
```

```

121     local head1 = line1.head
122     local head2 = line2.head
123
124     local char_count = 0
125     local word_count = 0
126
127     while head1 and head2 do
128         if (head1.id == 37 and head2.id == 37
129             and head1.char == head2.char)           -- identical glyph
130             or (head1.id == 10 and head2.id == 10) then -- glue
131
132                 if head1.id == 37 then -- glyph
133                     char_count = char_count + 1
134                 elseif char_count > 0 and head1.id == 10 then -- glue
135                     word_count = word_count + 1
136                 end
137                 head1 = head1.next
138                 head2 = head2.next
139             elseif (head1.id == 0 or head2.id == 0) then -- end of line
140                 break
141             elseif (head1.id ~= 37 and head1.id ~= 10) then -- some other kind of node
142                 head1 = head1.next
143             elseif (head2.id ~= 37 and head2.id ~= 10) then -- some other kind of node
144                 head2 = head2.next
145             else -- no match, no special node
146                 break
147             end
148         end
149         -- analyze last non-matching node, check for punctuation
150         if ((head1 and head1.id == 37 and head1.char > 49)
151             or (head2 and head2.id == 37 and head2.char > 49)) then
152             -- not a word
153         elseif char_count > 0 then
154             word_count = word_count + 1
155         end
156         return char_count,word_count,head1,head2
157     end
158
159     compare_lines_reverse = function (line1,line2)
160         local head1 = node.tail(line1.head)
161         local head2 = node.tail(line2.head)
162
163         local char_count = 0
164         local word_count = 0
165
166         while head1 and head2 do
167             if (head1.id == 37 and head2.id == 37
168                 and head1.char == head2.char)           -- identical glyph
169                 or (head1.id == 10 and head2.id == 10) then -- glue

```

```

171         if head1.id == 37 then -- glyph
172             char_count = char_count + 1
173         elseif char_count > 0 and head1.id == 10 then -- glue
174             word_count = word_count + 1
175         end
176         head1 = head1.prev
177         head2 = head2.prev
178     elseif (head1.id == 0 or head2.id == 0) then -- start of line
179         break
180     elseif (head1.id ~= 37 and head1.id ~= 10) then -- some other kind of node
181         head1 = head1.prev
182     elseif (head2.id ~= 37 and head2.id ~= 10) then -- some other kind of node
183         head2 = head2.prev
184     elseif (head1.id == 37 and head1.char < 48) then -- punctuation
185         head1 = head1.prev
186     elseif (head2.id == 37 and head2.char < 48) then -- punctuation
187         head2 = head2.prev
188     else -- no match, no special node
189         break
190     end
191 end
192 -- analyze last non-matching node, check for punctuation
193 if ((head1 and head1.id == 37 and head1.char > 49)
194     or (head2 and head2.id == 37 and head2.char > 49)) then
195     -- not a word
196 elseif char_count > 0 then
197     word_count = word_count + 1
198 end
199 return char_count,word_count,head1,head2
200 end
201
202 highlight = function (line,nend,color)
203     local n = node.new("whatsit","pdf_literal")
204
205     -- get dimensions
206     local w,h,d = node.dimensions(line.head,nend)
207     local w_pts = w/65536 -- scaled points to points
208
209     -- set data
210     n.data = "q " .. color .. " 0 0 m 0 5 l " .. w_pts .. " 5 l " .. w_pts .. " 0 l b Q"
211
212     -- insert node
213     n.next = line.head
214     line.head = n
215     node.slide(line.head)
216 end
217
218 highlight_reverse = function (nstart,line,color)
219     local n = node.new("whatsit","pdf_literal")
220

```



```

221
222     -- get dimensions
223     local w,h,d = node.dimensions(nstart,node.tail(line.head))
224     local w_pts = w/65536 -- scaled points to points
225
226     -- set data
227     n.data = "q " .. color .. " 0 0 m 0 5 l " .. w_pts .. " 5 l " .. w_pts .. " 0 l b Q"
228
229     -- insert node
230     node.insert_after(line.head,nstart,n)
231 end
232
233 homeoarchy = function (head)
234     local cur_line = head
235     local prev_line -- initiate prev_line
236
237     local max_char = tonumber(\inthomeoarchymaxchars)
238     local max_word = tonumber(\inthomeoarchymaxwords)
239
240     while head do
241         if head.id == 0 then -- new line
242             prev_line = cur_line
243             cur_line = head
244             if prev_line.id == 0 then
245                 -- homeoarchy
246                 char_count,word_count,prev_head,cur_head = compare_lines(prev_line,cur_line)
247                 if char_count >= max_char or word_count >= max_word then
248                     local color
249                     if word_count >= max_word then
250                         color = "q \usecolor{\inthomeoarchywordcolor}"
251                     else
252                         color = "q \usecolor{\inthomeoarchycharcolor}"
253                     end
254
255                     -- highlight both lines
256                     highlight(prev_line,prev_head,color)
257                     highlight(cur_line,cur_head,color)
258                 end
259             end
260         end
261         head = head.next
262     end
263     return true
264 end
265
266 luatexbase.add_to_callback("post_linebreak_filter",homeoarchy,"homeoarchy")
267
268 homoioteleuton = function (head)
269     local cur_line = head
270     local prev_line -- initiate prev_line

```

```

271
272     local max_char = tonumber(\inhomeoarchymaxchars)
273     local max_word = tonumber(\inhomeoarchymaxwords)
274
275     local linecounter = 0
276
277     while head do
278         if head.id == 0 then -- new line
279             linecounter = linecounter + 1
280             if linecounter > 1 then
281                 prev_line = cur_line
282                 cur_line = head
283                 if prev_line.id == 0 then
284                     -- homoioteleuton
285                     char_count,word_count,prev_head,cur_head = compare_lines_reverse(prev_line,cur_line)
286                     if char_count >= max_char or word_count >= max_word then
287                         local color
288                         if word_count >= max_word then
289                             color = "q \usecolor{\inhomeoarchywordcolor}"
290                         else
291                             color = "q \usecolor{\inhomeoarchycharcolor}"
292                         end
293
294                         -- highlight both lines
295                         highlight_reverse(prev_head,prev_line,color)
296                         highlight_reverse(cur_head,cur_line,color)
297                     end
298                 end
299             end
300         end
301         head = head.next
302     end
303
304     return true
305 end
306
307     luatexbase.add_to_callback("post_linebreak_filter",homoioteleuton,"homoioteleuton")
308     \end{luacode}
309 \else
310     \PackageError{The homeoarchy option only works with LuaTeX}
311 \fi
312 \fi
313 \fi
314 \ifintrivers
315 \ifintdraft
316     \ifluatex
317         \RequirePackage{luatexbase,luacode}
318         \begin{luacode}
319 river_analyze_line = function(line,dim1,dim2,precision)

```

Detect rivers

```

320 local head = line.head
321
322 while head do
323   if head.id == 10 then -- glue node
324     local w1,h1,d1 = node.dimensions(line.glue_set,line.glue_sign,line.glue_order,line.head)
325     local w2,h2,d2 = node.dimensions(line.glue_set,line.glue_sign,line.glue_order,line.head)
326     --print("dim1: "..dim1..""; dim2: "..dim2..""; w1: "..w1..""; w2: "..w2")
327     if w1 > dim2 + precision then -- out of range
328       return false,head
329     elseif w1 < (dim2 + precision) and w2 > (dim1 - precision) then -- found
330       return true,head
331     end
332   end
333   head = head.next
334 end
335
336 return false,head
337 end
338
339 rivers = function (head)
340   local prev_prev_line
341   local prev_line
342   local cur_line = head
343   local cur_node
344   local char_count
345
346   local linecounter = 0
347
348   while head do
349     if head.id == 0 then -- new line
350       linecounter = linecounter + 1
351       prev_prev_line = prev_line
352       prev_line = cur_line
353       cur_line = head
354       if linecounter > 2 then
355         cur_node = cur_line.head
356         char_count = 0
357
358         while cur_node do
359           if cur_node.id == 37 then -- glyph
360             char_count = char_count + 1
361           elseif cur_node.id == 10 and char_count > 0 and cur_node.next then -- glue node
362             -- prev_line
363             local w1,h1,d1 = node.dimensions(head.glue_set,head.glue_sign,head.glue_order)
364             local w2,h2,d2 = node.dimensions(head.glue_set,head.glue_sign,head.glue_order)
365             -- if we allow up to 45° diagonal rivers, then there can be up to + or - line
366             local w_p,h_p,d_p = node.dimensions(prev_line.head,cur_line.head) -- calculate
367             found_p,head_p = river_analyze_line(prev_line,w1,w2,h_p)
368
369             if found_p then

```

```

370         -- prev_prev_line
371         local w1,h1,d1 = node.dimensions(prev_line.glue_set,prev_line.glue_sign,pr
372         local w2,h2,d2 = node.dimensions(prev_line.glue_set,prev_line.glue_sign,pr
373         -- if we allow up to 45° diagonal rivers, then there can be up to + or - 1
374         local w_p,h_p,d_p = node.dimensions(prev_prev_line.head,prev_line.head) --
375         found_pp,head_pp = river_analyze_line(prev_prev_line,w1,w2,h_p)
376
377         if found_pp then
378             local n_pp = node.new("whatsit","pdf_literal")
379             n_pp.data = "q \usecolor{\intriverscolor} 0 0 m 0 5 1 5 5 1 5 0 1 b Q"
380             node.insert_after(prev_prev_line,head_pp.prev,n_pp)
381
382             local n_p = node.new("whatsit","pdf_literal")
383             n_p.data = "q \usecolor{\intriverscolor} 0 0 m 0 5 1 5 5 1 5 0 1 b Q"
384             node.insert_after(prev_line,head_p.prev,n_p)
385
386             local n_c = node.new("whatsit","pdf_literal")
387             n_c.data = "q \usecolor{\intriverscolor} 0 0 m 0 5 1 5 5 1 5 0 1 b Q"
388             node.insert_after(cur_line,cur_node.prev,n_c)
389         end
390     end
391 end
392     cur_node = cur_node.next
393 end
394 end
395     end
396     head = head.next
397 end
398
399 return true
400
401 end
402
403
404 luatexbase.add_to_callback("post_linebreak_filter",rivers,"rivers")
405     \end{luacode}
406     \else
407         \PackageError{The homearchy option only works with LuaTeX}
408     \fi
409 \fi
410 \fi

```

## Change History

0.1	General : First version	0.3	General : Add parindent and lastparline options
0.2	General : Add nosingleletter option		

0.4	General : Add draft mode . . . . .	1	0.8	General : Add river detection . . . . .	1
0.5	General : Add homearchy detection . . . . .	1	0.9	General : River detection returns false by default . . . . .	1
0.6	General : Words contain at least one character . . . . .	1	1.0	General : Improve documentation, simplify internal variables . . . . .	1
0.7	General : Add homoioteleuton 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>B</b>	<code>\begin</code> . . . . . 43, 83, 119, <b>318</b>	<code>\hyphenation</code> . . . . . <b>11</b>	<b>P</b>	<code>\PackageError</code> 74, 310, <b>407</b>
	<code>\brokenpenalty</code> . . . . . <b>27</b>	<b>I</b>		<code>\parfillskip</code> . . . . . <b>112</b>
<b>C</b>		<code>\ifintdraft</code> 54, 93, 116, <b>315</b>		<code>\parindent</code> . 2, 78, 89, <b>112</b>
	<code>\color@</code> . . . . . <b>25</b>	<code>\ifintfrenchchapters</code> <b>30</b>		<code>\ProcessKeyvalOptions</code> . . . . . <b>23</b>
	<code>\csname</code> . . . . . <b>25</b>	<code>\ifinthehomearchy</code> . . . . . <b>115</b>		<code>\ProvidesPackage</code> . . . . . <b>1</b>
<b>D</b>		<code>\ifinthyphenation</code> . . . . . <b>26</b>	<b>R</b>	
	<code>\DeclareBoolOption</code> 8--15	<code>\ifintlastparline</code> . . . . . <b>80</b>		<code>\renewcommand</code> . . . . . <b>32</b>
	<code>\DeclareStringOption</code> . . . . . 16--22	<code>\ifintnosingleletter</code> <b>40</b>		<code>\RequirePackage</code> . . 2, 3, 24, 42, 82, 118, <b>317</b>
	<code>\def</code> . . . . . <b>25</b>	<code>\ifintparindent</code> . . . . . <b>77</b>		<code>\rivers</code> . . . . . <b>3</b>
	<code>\dimexpr</code> . . . . . <b>112</b>	<code>\ifintrivers</code> . . . . . <b>314</b>		<code>\Roman</code> . . . . . <b>31</b>
	<code>\doublehyphendemerits</code> . . . . . <b>28</b>	<code>\ifluatex</code> . 41, 81, 117, <b>316</b>	<b>S</b>	
	<code>\draft</code> . . . . . <b>4</b>	<code>\ifnum</code> . . . . . <b>33</b>		<code>\setlength</code> . . . . . 78, <b>112</b>
<b>E</b>		<code>\inthehomearchycharcolor</code> . . . . . 252, <b>291</b>		<code>\SetupKeyvalOptions</code> . . <b>4</b>
	<code>\else</code> . . . . . 35, 60, 73, 99, 111, 309, <b>406</b>	<code>\inthehomearchymaxchars</code> . . . . . 237, <b>272</b>		<code>\space</code> . . . . . <b>25</b>
	<code>\end</code> . . . . . 72, 110, 308, <b>405</b>	<code>\inthehomearchymaxwords</code> . . . . . 238, <b>273</b>		<code>\string</code> . . . . . <b>25</b>
	<code>\endcsname</code> . . . . . <b>25</b>	<code>\inthehomearchywordcolor</code> . . . . . 250, <b>289</b>	<b>T</b>	
<b>F</b>		<code>\intlastparlinecolor</code> <b>95</b>		<code>\textwidth</code> . . . . . <b>112</b>
	<code>\fi</code> <b>29</b> , 37, 39, 62, 75, 76, 79, 101, <b>113</b> , 114, 311--313, <b>408--410</b>	<code>\intnosinglelettercolor</code> . . . . . <b>56</b>		<code>\thechapter</code> . . . . . <b>32</b>
	<code>\frenchchapter</code> . . . . . <b>31</b> , <b>36</b>	<code>\intriverscolor</code> . . . . . 379, 383, <b>387</b>	<b>U</b>	
	<code>\frenchchapters</code> . . . . . <b>3</b>	<b>L</b>		<code>\usecolor</code> . . . . . 25, 56, 95, 250, <b>252</b> , 289, 291, 379, <b>383</b> , <b>387</b>
<b>H</b>		<code>\lastparline</code> . . . . . <b>2</b>	<b>V</b>	
	<code>\homearchy</code> . . . . . <b>2</b>	<code>\let</code> . . . . . <b>31</b>		<code>\value</code> . . . . . <b>33</b>
		<b>N</b>		
		<code>\nosingleletter</code> . . . . . <b>2</b>		