

The impnattypo package

Raphaël Pinson
raphink@finchnest.ch

1.8 from 2026/06/27

1 Introduction

When it comes to French typography, the *Lexique des règles typographiques en usage à 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 impnattypo package, use the following line:

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

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]{impnattypo}
```

2.2 Paragraph formatting

parindent The booklet advises to indent paragraphs by 1em. This `\parindent` setting can be

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

achieved by using the `parindent` option:

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

`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 LuaTeX 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. The option `lastparlineminlength` allows to set, for the last line of a paragraph, a minimal length greater (but not smaller) than its default (`2\parindent`):

```
\usepackage[lastparline, lastparlineminlength=3em]{impnattyto}
```

With LuaTeX, the value given to the option `lastparlineminlength` is stored in the length `\lastparlineminlength`, which can be modified later, within the document. Hyphenation is also suppressed near the end of the paragraph to prevent hyphenation points from interfering with the minimal length constraint.

When the draft option is activated and LuaTeX 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 LuaTeX. To activate this functionality, you can use the `nosingleletter` option:

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

When this option is activated, only LuaTeX (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]{impnattyto}
```

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 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 `homeoarchywordcolor` option;
- Partial words are colored in **orange** and this color can be set by means of the `homeoarchycharcolor` 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 `homeoarchymaxwords` option;
- The number of matching characters is greater than 3. This parameter can be tuned with the `homeoarchymaxchars` 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]{imnpnatytpo}
```

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 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]{imnpnatytpo}
```

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 `nosinglet` 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{nosinglet}
12 \DeclareBoolOption{parindent}
13 \DeclareBoolOption{lastparline}
14 \DeclareBoolOption{homeoarchy}
15 \DeclareBoolOption{rivers}
16 \DeclareStringOption[red]{homeoarchywordcolor}
17 \DeclareStringOption[orange]{homeoarchycharcolor}
18 \DeclareStringOption[brown]{nosingletcolor}
19 \DeclareStringOption[teal]{lastparlinecolor}
20 \DeclareStringOption[lime]{riverscolor}
21 \DeclareStringOption[1]{homeoarchymaxwords}
22 \DeclareStringOption[3]{homeoarchymaxchars}
23 \DeclareStringOption[Opt]{lastparlineminlength}
24 \ProcessKeyvalOptions*
25 \RequirePackage{xcolor}
26 \def\usecolor#1{\csname\string\color@#1\endcsname\space}
27 \ifinthyphenation
28   \brokenpenalty=10000
29   \doublehyphendemerits=1000000000
30 \fi
31 \ifintfrenchchapters
32   \let\frenchchapter\Roman
33   \renewcommand{\thechapter}{%
34     \ifnum\value{chapter}=1
35       premier%
36     \else
```

No page finishes with an hyphenated word

Discourage hyphenation on two lines in a row

Number chapters

No single letter

```
37     \frenchchapter{chapter}%
38     \fi
39   }
40 \fi

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

Paragraph indentation

```
81 \ifintparindent
82 \setlength{\parindent}{1em}
83 \fi
```

Last line of paragraph

```
84 \ifintlastparline
85   \ifluatex
86     \newlength{\lastparlineminlength}
87     \setlength{\lastparlineminlength}{\intlastparlineminlength}
88     \RequirePackage{luatexbase,luacode}
89     \begin{luacode}
```

```

90     local glyph_id = node.id "glyph"
91     local glue_id = node.id "glue"
92     local hlist_id = node.id "hlist"
93     local disc_id = node.id "disc"
94
95     last_line_minimal_length = function (head)
96         while head do
97             local _w,_h,_d = node.dimensions(head)
98             if
99                 _w < 2 * tex.parindent or _w < tex.skip['lastparlineminlength'].width
100             then
101                 -- we have less than 2*\parindent or \lastparlineminlength to go
102
103                 if head.id == disc_id then
104                     -- we are at a discretionary: avoid hyphenations
105                     head.penalty=10000
106                 end
107
108                 if head.id == glue_id and head.subtype ~= 15 then
109                     -- we are at a glue: avoid line break
110                     local p = node.new("penalty")
111                     p.penalty = 10000
112
113                     \ifintdraft
114                         local w = node.new("whatsit","pdf_literal")
115                         w.data = "q \usecolor{\intlastparlinecolor} 0 0 m 0 5 1 2 5 1 2 0 1 b Q"
116
117                         node.insert_after(head,head.prev,w)
118                         node.insert_after(head,w,p)
119                     \else
120                         node.insert_after(head,head.prev,p)
121                     \fi
122                 end
123             end
124
125             head = head.next
126         end
127         return true
128     end
129
130     luatexbase.add_to_callback("pre_linebreak_filter",last_line_minimal_length,"lastparline
131     \end{luacode}
132
133     \else
134         \setlength{\@tempskipa}{\intlastparlineminlength}
135         \ifdim\@tempskipa < 2\parindent
136             \setlength{\@tempskipa}{2\parindent}
137         \fi
138         \setlength{\parfillskip}{0pt plus\dimexpr\textwidth-\@tempskipa}
139     \fi
140 \fi
141 \ifinthehomeoarchy
142 \ifintdraft
143 \ifluatex

```

Detect homeoarchies

```

144 \RequirePackage{luatexbase,luacode}
145 \begin{luacode}
146 local glyph_id = node.id "glyph"
147 local glue_id = node.id "glue"
148 local hlist_id = node.id "hlist"
149
150 compare_lines = function (line1,line2)
151     local head1 = line1.head
152     local head2 = line2.head
153
154     local char_count = 0
155     local word_count = 0
156
157     while head1 and head2 do
158         if (head1.id == glyph_id and head2.id == glyph_id
159             and head1.char == head2.char) -- identical glyph
160             or (head1.id == glue_id and head2.id == glue_id) then -- glue
161
162             if head1.id == glyph_id then -- glyph
163                 char_count = char_count + 1
164             elseif char_count > 0 and head1.id == glue_id then -- glue
165                 word_count = word_count + 1
166             end
167             head1 = head1.next
168             head2 = head2.next
169         elseif (head1.id == 0 or head2.id == 0) then -- end of line
170             break
171         elseif (head1.id ~= glyph_id and head1.id ~= glue_id) then -- some other kind of n
172             head1 = head1.next
173         elseif (head2.id ~= glyph_id and head2.id ~= glue_id) then -- some other kind of n
174             head2 = head2.next
175         else -- no match, no special node
176             break
177         end
178     end
179     -- analyze last non-matching node, check for punctuation
180     if ((head1 and head1.id == glyph_id and head1.char > 49)
181         or (head2 and head2.id == glyph_id and head2.char > 49)) then
182         -- not a word
183     elseif char_count > 0 then
184         word_count = word_count + 1
185     end
186     return char_count,word_count,head1,head2
187 end
188
189 compare_lines_reverse = function (line1,line2)
190     local head1 = node.tail(line1.head)
191     local head2 = node.tail(line2.head)
192
193     local char_count = 0
194     local word_count = 0
195
196     while head1 and head2 do
197         if (head1.id == glyph_id and head2.id == glyph_id

```

```

198         and head1.char == head2.char)           -- identical glyph
199         or (head1.id == glue_id and head2.id == glue_id) then -- glue
200
201         if head1.id == glyph_id then -- glyph
202             char_count = char_count + 1
203         elseif char_count > 0 and head1.id == glue_id then -- glue
204             word_count = word_count + 1
205         end
206         head1 = head1.prev
207         head2 = head2.prev
208     elseif (head1.id == 0 or head2.id == 0) then -- start of line
209         break
210     elseif (head1.id ~= glyph_id and head1.id ~= glue_id) then -- some other kind of node
211         head1 = head1.prev
212     elseif (head2.id ~= glyph_id and head2.id ~= glue_id) then -- some other kind of node
213         head2 = head2.prev
214     elseif (head1.id == glyph_id and head1.char < 48) then -- punctuation
215         head1 = head1.prev
216     elseif (head2.id == glyph_id and head2.char < 48) then -- punctuation
217         head2 = head2.prev
218     else -- no match, no special node
219         break
220     end
221 end
222 -- analyze last non-matching node, check for punctuation
223 if ((head1 and head1.id == glyph_id and head1.char > 49)
224     or (head2 and head2.id == glyph_id and head2.char > 49)) then
225     -- not a word
226 elseif char_count > 0 then
227     word_count = word_count + 1
228 end
229 return char_count, word_count, head1, head2
230 end
231
232 highlight = function (line, nend, color)
233     local n = node.new("whatsit", "pdf_literal")
234
235     -- get dimensions
236     local w, h, d = node.dimensions(line.head, nend)
237     local w_pts = w/65536 -- scaled points to points
238
239     -- set data
240     n.data = "q " .. color .. " 0 0 m 0 5 1 " .. w_pts .. " 5 1 " .. w_pts .. " 0 1 b Q"
241
242     -- insert node
243     n.next = line.head
244     line.head = n
245     node.slide(line.head)
246 end
247
248 highlight_reverse = function (nstart, line, color)
249     local n = node.new("whatsit", "pdf_literal")
250
251

```



```

252     -- get dimensions
253     local w,h,d = node.dimensions(nstart,node.tail(line.head))
254     local w_pts = w/65536 -- scaled points to points
255
256     -- set data
257     n.data = "q " .. color .. " 0 0 m 0 5 1 " .. w_pts .. " 5 1 " .. w_pts .. " 0 1 b Q"
258
259     -- insert node
260     node.insert_after(line.head,nstart,n)
261 end
262
263 homeoarchy = function (head)
264     local cur_line = head
265     local prev_line -- initiate prev_line
266
267     local max_char = tonumber(\inhomeoarchymaxchars)
268     local max_word = tonumber(\inhomeoarchymaxwords)
269
270     while head do
271         if head.id == hlist_id then -- new line
272             prev_line = cur_line
273             cur_line = head
274             if prev_line.id == hlist_id then
275                 -- homeoarchy
276                 char_count,word_count,prev_head,cur_head = compare_lines(prev_line,cur_line)
277                 if char_count >= max_char or word_count >= max_word then
278                     local color
279                     if word_count >= max_word then
280                         color = "q \usecolor{\inhomeoarchywordcolor}"
281                     else
282                         color = "q \usecolor{\inhomeoarchycharcolor}"
283                     end
284
285                     -- highlight both lines
286                     highlight(prev_line,prev_head,color)
287                     highlight(cur_line,cur_head,color)
288                 end
289             end
290             end
291             head = head.next
292         end
293         return true
294     end
295
296     luatexbase.add_to_callback("post_linebreak_filter",homeoarchy,"homeoarchy")
297
298     homoioteleuton = function (head)
299         local cur_line = head
300         local prev_line -- initiate prev_line
301
302         local max_char = tonumber(\inhomeoarchymaxchars)
303         local max_word = tonumber(\inhomeoarchymaxwords)
304
305         local linecounter = 0

```

```

306
307     while head do
308         if head.id == hlist_id then -- new line
309             linecounter = linecounter + 1
310             if linecounter > 1 then
311                 prev_line = cur_line
312                 cur_line = head
313                 if prev_line.id == hlist_id then
314                     -- homoioteleuton
315                     char_count,word_count,prev_head,cur_head = compare_lines_reverse(prev_line,
316                     if char_count >= max_char or word_count >= max_word then
317                         local color
318                         if word_count >= max_word then
319                             color = "q \usecolor{\inthomeoarchywordcolor}"
320                         else
321                             color = "q \usecolor{\inthomeoarchycharcolor}"
322                         end
323
324                         -- highlight both lines
325                         highlight_reverse(prev_head,prev_line,color)
326                         highlight_reverse(cur_head,cur_line,color)
327                     end
328                 end
329             end
330         end
331         head = head.next
332     end
333
334     return true
335 end
336
337     luatexbase.add_to_callback("post_linebreak_filter",homoioteleuton,"homoioteleuton")
338 \end{luacode}
339 \else
340     \PackageError{The homeoarchy option only works with LuaTeX}
341 \fi
342 \fi
343 \fi
344 \ifintrivers
345 \ifintdraft
346     \ifluatex
347         \RequirePackage{luatexbase,luacode}
348         \begin{luacode}
349 local glyph_id = node.id "glyph"
350 local glue_id = node.id "glue"
351 local hlist_id = node.id "hlist"
352
353 river_analyze_line = function(line,dim1,dim2,precision)
354     local head = line.head
355
356     while head do
357         if head.id == glue_id then -- glue node
358             local w1,h1,d1 = node.dimensions(line.glue_set,line.glue_sign,line.glue_order,line.h
359             local w2,h2,d2 = node.dimensions(line.glue_set,line.glue_sign,line.glue_order,line.h

```

Detect rivers

```

360         --print("dim1:"..dim1..""; dim2:"..dim2..""; w1:"..w1..""; w2:"..w2")
361         if w1 > dim2 + precision then -- out of range
362             return false,head
363         elseif w1 < (dim2 + precision) and w2 > (dim1 - precision) then -- found
364             return true,head
365         end
366     end
367     head = head.next
368 end
369
370 return false,head
371 end
372
373 rivers = function (head)
374     local prev_prev_line
375     local prev_line
376     local cur_line = head
377     local cur_node
378     local char_count
379
380     local linecounter = 0
381
382     while head do
383         if head.id == hlist_id then -- new line
384             linecounter = linecounter + 1
385             prev_prev_line = prev_line
386             prev_line = cur_line
387             cur_line = head
388             if linecounter > 2 then
389                 cur_node = cur_line.head
390                 char_count = 0
391
392                 while cur_node do
393                     if cur_node.id == glyph_id then -- glyph
394                         char_count = char_count + 1
395                     elseif cur_node.id == glue_id and char_count > 0 and cur_node.next then -- glue
396                         -- prev_line
397                         local w1,h1,d1 = node.dimensions(head.glue_set,head.glue_sign,head.glue_order)
398                         local w2,h2,d2 = node.dimensions(head.glue_set,head.glue_sign,head.glue_order)
399                         -- if we allow up to 45° diagonal rivers, then there can be up to + or - line
400                         local w_p,h_p,d_p = node.dimensions(prev_line.head,cur_line.head) -- calculate
401                         found_p,head_p = river_analyze_line(prev_line,w1,w2,h_p)
402
403                     if found_p then
404                         -- prev_prev_line
405                         local w1,h1,d1 = node.dimensions(prev_line.glue_set,prev_line.glue_sign,prev_line.glue_order)
406                         local w2,h2,d2 = node.dimensions(prev_line.glue_set,prev_line.glue_sign,prev_line.glue_order)
407                         -- if we allow up to 45° diagonal rivers, then there can be up to + or - line
408                         local w_p,h_p,d_p = node.dimensions(prev_prev_line.head,prev_line.head) -- calculate
409                         found_pp,head_pp = river_analyze_line(prev_prev_line,w1,w2,h_p)
410
411                     if found_pp then
412                         local n_pp = node.new("whatsit","pdf_literal")
413                         n_pp.data = "q \usecolor{\intriverscolor} 0 0 m 0 5 1 5 5 1 5 0 1 b Q"

```

```

414         node.insert_after(prev_prev_line,head_pp.prev,n_pp)
415
416         local n_p = node.new("whatsit","pdf_literal")
417         n_p.data = "q \usecolor{\intriverscolor} 0 0 m 0 5 1 5 5 1 5 0 1 b Q"
418         node.insert_after(prev_line,head_p.prev,n_p)
419
420         local n_c = node.new("whatsit","pdf_literal")
421         n_c.data = "q \usecolor{\intriverscolor} 0 0 m 0 5 1 5 5 1 5 0 1 b Q"
422         node.insert_after(cur_line,cur_node.prev,n_c)
423     end
424 end
425 end
426     cur_node = cur_node.next
427 end
428 end
429     end
430     head = head.next
431 end
432
433 return true
434
435 end
436
437
438 luatexbase.add_to_callback("post_linebreak_filter",rivers,"rivers")
439     \end{luacode}
440 \else
441     \PackageError{The homeoarchy option only works with LuaTeX}
442 \fi
443 \fi
444 \fi

```

Change History

0.1	General: First version	0.8	General: Add river detection
0.2	General: Add nosingleletter option	0.9	General: River detection returns false by default
0.3	General: Add parindent and lastparline options	1.0	General: Improve documentation, simplify internal variables
0.4	General: Add draft mode	1.1	General: Fix French documentation
0.5	General: Add homeoarchy detection	1.2	General: Fix French documentation
0.6	General: Words contain at least one character	1.3	General: Fix French documentation
0.7	General: Add homoioteleuton detection	1.4	General: Fix release date

1.5	General: Fix support for <code>TeXLive 2016</code> (new <code>luatex compatibility</code>) Thanks to Michal Hoftich 1	1.7	General: Avoids hyphenations when lastparline is active (lua only) 1
1.6	General: Add <code>lastparlineminlength</code> option 1	1.8	General: Update maintainer email, copyright year, README license statement, drop <code>tds.zip</code> 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.

Symbols	<code>\hyphenation</code> 1	N
<code>\@tempskipa</code>		<code>\newlength</code> 86
. . . 134, 135, 136, 138		<code>\nosingleletter</code> 2
B	I	P
<code>\begin</code> 44, 89, 145, 348	<code>\ifdim</code> 135	<code>\PackageError</code> 78, 340, 441
<code>\brokenpenalty</code> 28	<code>\ifintdraft</code>	<code>\parfillskip</code> 138
	. . . 58, 113, 142, 345	<code>\parindent</code>
C	<code>\ifintfrenchchapters</code> 31	. . . 1, 82, 101, 135, 136
<code>\color@</code> 26	<code>\ifinhomeoarchy</code> . . . 141	<code>\ProcessKeyvalOptions</code>
<code>\csname</code> 26	<code>\ifinthyphenation</code> . . . 27 24
	<code>\ifintlastparline</code> . . . 84	<code>\ProvidesPackage</code> 1
D	<code>\ifintnosingleletter</code> 41	
<code>\DeclareBoolOption</code> 8,	<code>\ifintparindent</code> 81	
9, 10, 11, 12, 13, 14, 15	<code>\ifintrivers</code> 344	
<code>\DeclareStringOption</code>	<code>\ifluatex</code> . . 42, 85, 143, 346	R
. 16,	<code>\ifnum</code> 34	<code>\renewcommand</code> 33
17, 18, 19, 20, 21, 22, 23	<code>\inhomeoarchycharcolor</code>	<code>\RequirePackage</code> . . 2,
<code>\def</code> 26 282, 321	3, 25, 43, 88, 144, 347
<code>\dimexpr</code> 138	<code>\inhomeoarchymaxchars</code>	<code>\rivers</code> 3
<code>\doublehyphendemerits</code> 267, 302	<code>\Roman</code> 32
. 29	<code>\inhomeoarchymaxwords</code>	
 268, 303	S
E	<code>\inhomeoarchywordcolor</code>	<code>\setlength</code>
<code>\else</code> 36, 64, 280, 319	. . . 82, 87, 134, 136, 138
77, 119, 133, 339, 440	<code>\intlastparlinecolor</code> 115	<code>\SetupKeyvalOptions</code> . . 4
<code>\end</code> 76, 132, 338, 439	<code>\intlastparlineminlength</code>	<code>\space</code> 26
<code>\endcsname</code> 26 87, 134	<code>\string</code> 26
F	<code>\intnosinglelettercolor</code>	
<code>\fi</code> 30, 38, 40, 60	T
66, 79, 80, 83, 121,	<code>\intriverscolor</code>	<code>\textwidth</code> 138
137, 139, 140, 341, 413, 417, 421	<code>\thechapter</code> 33
342, 343, 442, 443, 444		
<code>\frenchchapter</code> . . . 32, 37	L	U
<code>\frenchchapters</code> 3	<code>\lastparline</code> 2	<code>\usecolor</code> 26,
	<code>\lastparlineminlength</code>	60, 115, 280, 282,
 86, 87, 101	319, 321, 413, 417, 421
H	<code>\let</code> 32	V
<code>\homeoarchy</code> 2		<code>\value</code> 34