

The impnattypo package

Raphaël Pinson
raphink@gmail.com

1.6 from 2026/05/27

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]{impnatty}
```

`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. 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]{impnatty}
```

With Lua_{TeX}, the value given to the option `lastparlineminlength` is stored in the length `\lastparlineminlength`, which can be modified later, within the document.

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]{impnatty}
```

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]{impnatty}
```

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 `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]{impnatty}
```

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]{impnatty}
```

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

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

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

Paragraph indentation

Last line of paragraph

```

90     local glyph_id = node.id "glyph"
91     local glue_id = node.id "glue"
92     local hlist_id = node.id "hlist"
93
94     last_line_minimal_length = function (head)
95         while head do
96             local _w,_h,_d = node.dimensions(head)
97             if head.id == glue_id and head.subtype ~= 15
98                 and (_w < 2 * tex.parindent or _w < tex.skip['lastparlineminlength'].width)
99             then
100
101                 -- we are at a glue and have less then 2*\parindent of \lastparlineminlength to
102                 local p = node.new("penalty")
103                 p.penalty = 10000
104
105                 \ifintdraft
106                     local w = node.new("whatsit","pdf_literal")
107                     w.data = "q \usecolor{\intlastparlinecolor} 0 0 m 0 5 1 2 5 1 2 0 1 b Q"
108
109                     node.insert_after(head,head.prev,w)
110                     node.insert_after(head,w,p)
111                 \else
112                     node.insert_after(head,head.prev,p)
113                 \fi
114             end
115
116             head = head.next
117         end
118         return true
119     end
120
121     luatexbase.add_to_callback("pre_linebreak_filter",last_line_minimal_length,"lastparline
122     \end{luacode}
123 \else
124     \setlength{\@tempskipa}{\intlastparlineminlength}
125     \ifdim\@tempskipa < 2\parindent
126         \setlength{\@tempskipa}{2\parindent}
127     \fi
128     \setlength{\parfillskip}{0pt plus\dimexpr\textwidth-\@tempskipa}
129 \fi
130 \fi
131 \ifinhomeoarchy
132 \ifintdraft
133     \ifluatex
134         \RequirePackage{luatexbase,luacode}
135         \begin{luacode}
136             local glyph_id = node.id "glyph"
137             local glue_id = node.id "glue"
138             local hlist_id = node.id "hlist"
139
140             compare_lines = function (line1,line2)
141                 local head1 = line1.head
142                 local head2 = line2.head
143

```

Detect homeoarchies

```

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

```

```

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

```



```

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

```

```

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

```

Detect rivers

```

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

```

```

414         end
415     end
416     cur_node = cur_node.next
417 end
418 end
419 end
420 head = head.next
421 end
422
423 return true
424
425 end
426
427
428 luatexbase.add_to_callback("post_linebreak_filter",rivers,"rivers")
429 \end{luacode}
430 \else
431 \PackageError{The homeoarchy option only works with LuaTeX}
432 \fi
433 \fi
434 \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 homeoarchy 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 TexLive 2016 (new luatex compatibility). Thanks to Michal Hoftich 1
0.8	General: Add river detection 1	1.6	General: Add lastparlineminlength option 1

Index

Numbers written in *italics* 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	\hyphenation I	N
\@tempskipa		\newlength 86
. . . 124, 125, 126, 128	I	\nosingleletter 12
B	\ifdim 125	P
\begin 44, 89, 135, 338	\ifintdraft	\PackageError 78, 330, 431
\brokenpenalty 28	. . . 58, 105, 132, 335	\parfillskip 128
C	\ifintfrenchchapters 31	\parindent
\color 26	\ifintheoarchy 131	. . . 1, 82, 101, 125, 126
\csname 26	\ifinthyphenation 27	\ProcessKeyvalOptions
D	\ifintlastparline 84 24
\DeclareBoolOption 8,	\ifintnosingleletter 41	\ProvidesPackage 1
9, 10, 11, 12, 13, 14, 15	\ifintparindent 81	R
\DeclareStringOption	\ifintrivers 334	\renewcommand 33
. 16,	\ifluatex 42, 85, 133, 336	\RequirePackage 2,
17, 18, 19, 20, 21, 22, 23	\ifnum 34	3, 25, 43, 88, 134, 337
\def 26	\intheoarchycharcolor	\rivers 3
\dimexpr 128 272, 311	\Roman 32
\doublehyphenemerits	\intheoarchymaxchars	
. 29 257, 292	S
E	\intheoarchymaxwords	\setlength
\else 36, 64, 258, 293 82, 87, 124, 126, 128
77, 111, 123, 329, 430	\intheoarchywordcolor	\SetupKeyvalOptions 4
\end 76, 122, 328, 429 270, 309	\space 26
\endcsname 26	\intlastparlinecolor 107	\string 26
F	\intlastparlineminlength	
\fi 30, 38, 40, 87, 124	T
66, 79, 80, 83, 113,	\intnosinglelettercolor	\textwidth 128
127, 129, 130, 331, 60	\thechapter 33
332, 333, 432, 433, 434	\intriverscolor	
\frenchchapter 32, 37 403, 407, 411	U
\frenchchapters 3	I	\usecolor 26,
H	\lastparline 2	60, 107, 270, 272,
\homeoarchy 2	\lastparlineminlength	309, 311, 403, 407, 411
 86, 87, 101	V
	\let 32	\value 34