The impnattypo 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 à l’Imprimerie Nationale* is a definite reference.

While the majority of the recommendations of this book has been implemented in the french command for babel, other recommendations still deserve to be automatized in order to be implemented in \(\LaTeX\).

Such is the original goal of this package, initiated by a question on the tex.stackexchange.com\(^1\) website, and which implements several of the rules listed in this booklet so as to make them more easily applicable to texts edited with \(\LaTeX\).

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:

\begin{verbatim}
\usepackage{option}{impnattypo}
\end{verbatim}

The package options are described in the following sections.

2.1 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 activate this functionality, use the hyphenation option:

\begin{verbatim}
\usepackage[hyphenation]{impnattypo}
\end{verbatim}

\(^1\)http://tex.stackexchange.com/questions/20493/french-typography-recommendations
2.2 Paragraph formatting

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

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

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.\footnote{http://tex.stackexchange.com/questions/28357/ensure-minimal-length-of-last-line}

When Lua\TeX{} is used, the solution provided by Patrick Gundlach\footnote{http://tex.stackexchange.com/questions/28357/ensure-minimal-length-of-last-line/28361#28361} is used. With other rendering engines, it is the native solution provided by Enrico Gregorio\footnote{http://tex.stackexchange.com/questions/28357/ensure-minimal-length-of-last-line/28358#28358} that serves as an implementation:

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

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

It is also recommended to avoid hyphenation points that would isolate a single letter. The solution proposed by Patrick Gundlach\footnote{http://tex.stackexchange.com/questions/27780/one-letter-word-at-the-end-of-line} allows to fix this by using Lua\TeX{}. To activate this functionality, you can use the nosingleletter option:

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

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 \texttt{brown}. The color can be tuned by setting the nosinglelettercolor option.

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.\footnote{http://tex.stackexchange.com/questions/27588/repetition-of-a-word-on-two-lines} For this reason, the homeoarchy option in this package only detects and highlights them. Fixing them usually be a matter of introducing ties in the paragraph:

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

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:

\begin{itemize}
  \item \texttt{brown}
  \item \texttt{teal}
\end{itemize}
• 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]{impnattypo}

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

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]{impnattypo}

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 impnattypo package features a draft mode allowing to visualize the penalties (ties) inserted by the nosingleletter 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]{impnattypo}
```

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

3 Implementation

```
\ProvidesPackage{impnattypo}
\RequirePackage{ifluatex}
\RequirePackage{kvoptions}
\SetupKeyvalOptions{
  family=impnattypo,
  prefix=int,
}
\DeclareBoolOption{draft}
\DeclareBoolOption{frenchchapters}
\DeclareBoolOption{hyphenation}
\DeclareBoolOption{nosingleletter}
\DeclareBoolOption{parindent}
\DeclareBoolOption{lastparline}
\DeclareBoolOption{homeoarchy}
\DeclareBoolOption{rivers}
\DeclareStringOption[red]{homeoarchywordcolor}
\DeclareStringOption[orange]{homeoarchycharcolor}
\DeclareStringOption[brown]{nosinglelettercolor}
\DeclareStringOption[teal]{lastparlinecolor}
\DeclareStringOption[lime]{riverscolor}
\DeclareStringOption[1]{homeoarchymaxwords}
\DeclareStringOption[3]{homeoarchymaxchars}
\ProcessKeyvalOptions*
\RequirePackage{xcolor}
\def\usecolor#1{\csname\string\color@#1\endcsname\space}
\ifinthyphenation
  \brokenpenalty=10000
\fi
\ifintfrenchchapters
  \let\frenchchapter\Roman
\renewcommand{\thechapter}{%
No single letter

\ifintnosingleletter
  \ifluatex
    \RequirePackage{luatexbase,luacode}
    \begin{luacode}
    local glyph_id = node.id "glyph"
    local glue_id = node.id "glue"
    local hlist_id = node.id "hlist"

    local prevent_single_letter = function (head)
      while head do
        if head.id == glyph_id then -- glyph
          if unicode.utf8.match(unicode.utf8.char(head.char),"%a") then -- some kind of letter
            if head.prev.id == glue_id and head.next.id == glue_id then -- only if we are at a one letter word
              local p = node.new("penalty")
              p.penalty = 10000
              \ifintdraft
                local w = node.new("whatsit","pdf_literal")
                w.data = "q \usecolor{\intnosinglelettercolor} 0 0 m 0 5 l 2 5 l 2 0 l b Q"
                node.insert_after(head,head,w)
                node.insert_after(head,w,p)
              \else
                node.insert_after(head,head,p)
              \end
            end
          end
        end
        head = head.next
      end
      return true
    end
    \end{luacode}
  \else
    \PackageError{The nosingleletter option only works with LuaTeX}
  \fi
\fi

Paragraph indentation

\ifintparindent
  \setlength{\parindent}{1em}
\fi
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 then 2*\parindent to go
97 local p = node.new("penalty")
98 p.penalty = 10000
99
100 \ifintdraft
101 local w = node.new("whatst","pdf literal")
102 w.data = "q \usecolor{intlastparlinecolor} 0 0 m 0 5 1 2 5 1 2 0 1 b Q"
103 node.insert_after(head,head.prev,w)
104 node.insert_after(head,w,p)
105 \else
106 node.insert_after(head,head.prev,p)
107 \fi
108 \fi
109
110 head = head.next
111 end
112 end
113 return true
114 end
115 luatexbase.add_to_callback("pre_linebreak_filter",last_line_twice_parindent,"lastparline")
116 \end{luacode}
117 \else
118 \setlength{\parfillskip}{\Op plus\dimexpr\textwidth-2\parindent}
119 \fi
120 \fi

Detect homearchies

122 \ifinthomearchy
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
compare_lines = function (line1, line2)
local head1 = line1.head
local head2 = line2.head
local char_count = 0
local word_count = 0
while head1 and head2 do
  if (head1.id == glyph_id and head2.id == glyph_id
      and head1.char == head2.char) -- identical glyph
    or (head1.id == glue_id and head2.id == glue_id) then -- glue
    if head1.id == glyph_id then -- glyph
      char_count = char_count + 1
    elseif char_count > 0 and head1.id == glue_id then -- glue
      word_count = word_count + 1
    end
    head1 = head1.next
    head2 = head2.next
  elseif (head1.id == 0 or head2.id == 0) then -- end of line
    break
  elseif (head1.id ~= glyph_id and head1.id ~= glue_id) then -- some other kind of node
    head1 = head1.next
  elseif (head2.id ~= glyph_id and head2.id ~= glue_id) then -- some other kind of node
    head2 = head2.next
  else -- no match, no special node
    break
  end
end
-- analyze last non-matching node, check for punctuation
if ((head1 and head1.id == glyph_id and head1.char > 49)
    or (head2 and head2.id == glyph_id and head2.char > 49)) then
  -- not a word
else if char_count > 0 then
  word_count = word_count + 1
end
return char_count, word_count, head1, head2
end

compare_lines_reverse = function (line1, line2)
local head1 = node.tail(line1.head)
local head2 = node.tail(line2.head)
local char_count = 0
local word_count = 0
while head1 and head2 do
  if (head1.id == glyph_id and head2.id == glyph_id
      and head1.char == head2.char) -- identical glyph
    or (head1.id == glue_id and head2.id == glue_id) then -- glue
    if head1.id == glyph_id then -- glyph
      char_count = char_count + 1
    elseif char_count > 0 and head1.id == glue_id then -- glue
      word_count = word_count + 1
    end
    head1 = head1.next
    head2 = head2.next
  elseif (head1.id == 0 or head2.id == 0) then -- end of line
    break
  elseif (head1.id ~= glyph_id and head1.id ~= glue_id) then -- some other kind of node
    head1 = head1.next
  elseif (head2.id ~= glyph_id and head2.id ~= glue_id) then -- some other kind of node
    head2 = head2.next
  else -- no match, no special node
    break
  end
end
-- analyze last non-matching node, check for punctuation
if ((head1 and head1.id == glyph_id and head1.char > 49)
    or (head2 and head2.id == glyph_id and head2.char > 49)) then
  -- not a word
else if char_count > 0 then
  word_count = word_count + 1
end
return char_count, word_count, head1, head2
end
if head1.id == glyph_id then -- glyph
    char_count = char_count + 1
elseif char_count > 0 and head1.id == glue_id then -- glue
    word_count = word_count + 1
end
head1 = head1.prev
head2 = head2.prev
elseif (head1.id == 0 or head2.id == 0) then -- start of line
    break
elseif (head1.id ~= glyph_id and head1.id ~= glue_id) then -- some other kind of node
    head1 = head1.prev
elseif (head2.id ~= glyph_id and head2.id ~= glue_id) then -- some other kind of node
    head2 = head2.prev
elseif (head1.id == glyph_id and head1.char < 48) then -- punctuation
    head1 = head1.prev
elseif (head2.id == glyph_id and head2.char < 48) then -- punctuation
    head2 = head2.prev
else -- no match, no special node
    break
end
-- analyze last non-matching node, check for punctuation
if ((head1 and head1.id == glyph_id and head1.char > 49)
    or (head2 and head2.id == glyph_id and head2.char > 49)) then
    -- not a word
    elseif char_count > 0 then
        word_count = word_count + 1
    end
return char_count,word_count,head1,head2
end
highlight = function (line,nend,color)
    local n = node.new("whatsit","pdf_literal")
    -- get dimensions
    local w,h,d = node.dimensions(line.head,nend)
    local w_pts = w/65536 -- scaled points to points
    -- set data
    n.data = "q " .. color .. " 0 0 m 0 5 l " .. w_pts .. " 5 l " .. w_pts .. " 0 1 b Q"
    -- insert node
    n.next = line.head
    line.head = n
    node.slide(line.head)
end
highlight_reverse = function (nstart,line,color)
    local n = node.new("whatsit","pdf_literal")
local w,h,d = node.dimensions(nstart,node.tail(line.head))
local w_pts = w/65536 -- scaled points to points

-- set data
n.data = "q " .. color .. " 0 0 m 0 5 l " .. w_pts .. " 5 l " .. w_pts .. " 0 l b Q"

-- insert node
node.insert_after(line.head,nstart,n)
end

homeoarchy = function (head)
local cur_line = head
local prev_line -- initiate prev_line

local max_char = tonumber(\inthomeoarchymaxchars)
local max_word = tonumber(\inthomeoarchymaxwords)

while head do
if head.id == hlist_id then -- new line
  prev_line = cur_line
  cur_line = head
  if prev_line.id == hlist_id then
    -- homeoarchy
    char_count,word_count,prev_head,cur_head = compare_lines(prev_line,cur_line)
    if char_count >= max_char or word_count >= max_word then
      local color
      if word_count >= max_word then
        color = "q \usecolor{\inthomeoarchywordcolor}"
      else
        color = "q \usecolor{\inthomeoarchycharcolor}"
      end
    end
    -- highlight both lines
    highlight(prev_line,prev_head,color)
    highlight(cur_line,cur_head,color)
  end
  end
  head = head.next
end

luatexbase.add_to_callback("post_linebreak_filter",homeoarchy,"homeoarchy")

homoioioteleuton = function (head)
local cur_line = head

local prev_line -- initiate prev_line
local max_char = tonumber(\inthomeoarchymaxchars)
local max_word = tonumber(\inthomeoarchymaxwords)

local linecounter = 0

while head do
  if head.id == hlist_id then -- new line
    linecounter = linecounter + 1
    if linecounter > 1 then
      prev_line = cur_line
      cur_line = head
      if prev_line.id == hlist_id then
        -- homoioteleuton
        char_count,word_count,prev_head,cur_head = compare_lines_reverse(prev_line,cur_line)
        if char_count >= max_char or word_count >= max_word then
          local color
          if word_count >= max_word then
            color = "q \usecolor{\inthomeoarchywordcolor}"
          else
            color = "q \usecolor{\inthomeoarchycharcolor}"
          end
          -- highlight both lines
          highlight_reverse(prev_head,prev_line,color)
          highlight_reverse(cur_head,cur_line,color)
        end
      end
    end
  end
  head = head.next
end

return true

\luatexbase.add_to_callback("post_linebreak_filter",homoioteleuton,"homoioteleuton")
\end{luacode}
\else
\PackageError{The homeoarchy option only works with LuaTeX}
\fi
\fi

Detect rivers
\ifintrivers
\ifintdraft
\ifluatex
 \RequirePackage{luatexbase,luacode}
 \begin{luacode}

10
local glyph_id = node.id "glyph"
local glue_id = node.id "glue"
local hlist_id = node.id "hlist"

river_analyze_line = function(line,dim1,dim2,precision)
    local head = line.head
    while head do
        if head.id == glue_id then -- glue node
            local w1,h1,d1 = node.dimensions(line.glue_set,line.glue_sign,line.glue_order,line.head,head.prev)
            local w2,h2,d2 = node.dimensions(line.glue_set,line.glue_sign,line.glue_order,line.head,head)
            if w1 > dim2 + precision then -- out of range
                return false,head
            elseif w1 < (dim2 + precision) and w2 > (dim1 - precision) then -- found
                return true,head
            end
        end
        head = head.next
    end
    return false,head
end

rivers = function (head)
local prev_prev_line
local prev_line
local cur_line = head
local cur_node
local char_count
local linecounter = 0
while head do
    if head.id == hlist_id then -- new line
        linecounter = linecounter + 1
        prev_prev_line = prev_line
        prev_line = cur_line
        cur_line = head
        if linecounter > 2 then
            cur_node = cur_line.head
            char_count = 0
            while cur_node do
                if cur_node.id == glyph_id then -- glyph
                    char_count = char_count + 1
                elseif cur_node.id == glue_id and char_count > 0 and cur_node.next then -- glue
                    -- prev_line
                    local w1,h1,d1 = node.dimensions(head.glue_set,head.glue_sign,head.glue_order
                    local w2,h2,d2 = node.dimensions(head.glue_set,head.glue_sign,head.glue_order
                end
            end
        end
    end
end
if we allow up to 45° diagonal rivers, then there can be up to ± line height between spaces

local w_p,h_p,d_p = node.dimensions(prev_line.head,cur_line.head) -- calculate line height

local p,head_p = river_analyze_line(prev_line,w1,w2,h_p)

if found_p then
  -- prev_prev_line
  local w1,h1,d1 = node.dimensions(prev_line.glue_set,prev_line.glue_sign,prev_line.glue_order,prev_line.head,head_p.prev)
  local w2,h2,d2 = node.dimensions(prev_line.glue_set,prev_line.glue_sign,prev_line.glue_order,prev_line.head,head_p)
  -- if we allow up to 45° diagonal rivers, then there can be up to ± line height between spaces
  local w_p,h_p,d_p = node.dimensions(prev_prev_line.head,prev_line.head) -- calculate line height
  local p,head_pp = river_analyze_line(prev_prev_line,w1,w2,h_p)

  if found_pp then
    local n_pp = node.new("whatsit","pdf_literal")
    n_pp.data = "q \usecolor{\intriverscolor} 0 0 m 0 5 l 5 5 l 5 0 l b Q"
    node.insert_after(prev_prev_line,head_pp.prev,n_pp)

    local n_p = node.new("whatsit","pdf_literal")
    n_p.data = "q \usecolor{\intriverscolor} 0 0 m 0 5 l 5 5 l 5 0 l b Q"
    node.insert_after(prev_line,head_p.prev,n_p)

    local n_c = node.new("whatsit","pdf_literal")
    n_c.data = "q \usecolor{\intriverscolor} 0 0 m 0 5 l 5 5 l 5 0 l b Q"
    node.insert_after(cur_line,cur_node.prev,n_c)
  end

  cur_node = cur_node.next
end

end

head = head.next

return true

luatexbase.add_to_callback("post_linebreak_filter",rivers,"rivers")

\end{luacode}

\else
\PackageError{The homeoarchy option only works with LuaTeX}
Change History

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

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.

\begin{code}
\begin{itemize}
\item \verb|begin| 43, 86, 126
\item \verb|brokenpenalty| 27
\item \verb|color| 25
\item \verb|csname| 25
\item \verb|DeclareBoolOption| 8–15
\item \verb|DeclareStringOption| 16–22
\item \verb|def| 25
\item \verb|dimexpr| 19
\item \verb|doublehyphenemerges| 28
\item \verb|else| 35, 63, 76, 106, 118, 320
\item \verb|end| 75, 117, 319
\item \verb|endcsname| 25
\item \verb|fi| 29, 37, 39, 65, 78, 79, 82, 108, 120, 121, 322–324
\item \verb|fi| 423–425
\item \verb|frenchchap\ldots| 31
\item \verb|frenchchapters| 3
\item \verb|ifintdraft\ldots| 57, 100, 123
\item \verb|ifintfrenchchapters| 30
\item \verb|ifinthomeoarchy\ldots| 122
\item \verb|ifinthomeoarchymaxchars| 248
\item \verb|ifinthomeoarchymaxwords| 249
\item \verb|ifinthyphenation| 36
\item \verb|ifintnosingleletter| 100
\item \verb|ifintparindent| 80
\item \verb|ifintrivers| 325
\item \verb|ifluatex| 41, 84, 124
\item \verb|ifnum| 13
\item \verb|inthomeoarchycharcolor| 263
\item \verb|inthomeoarchymaxchars| 283
\item \verb|inthomeoarchymaxwords| 284
\item \verb|inthomeoarchywordcolor| 261
\end{itemize}
\end{code}