

# The `luacolor` package

Heiko Oberdiek\*

2019/07/25 v1.12

## Abstract

Package `luacolor` implements color support based on Lua $\TeX$ 's node attributes.

## Contents

<b>1</b>	<b>Documentation</b>	<b>2</b>
1.1	Introduction	2
1.2	Usage	2
1.3	Limitations	3
<b>2</b>	<b>Implementation</b>	<b>3</b>
2.1	Catcodes and identification	3
2.2	Check for Lua $\TeX$	4
2.3	Check for disabled colors	4
2.4	Load module and check version	4
2.5	Find driver	5
2.6	Attribute setting	5
2.7	Whatsit insertion	6
2.8	<code>\pdfxform</code> support	6
2.9	Lua module	7
2.9.1	Driver detection	7
2.9.2	Color strings	8
2.9.3	Attribute register	8
2.9.4	Whatsit insertion	9
<b>3</b>	<b>Test</b>	<b>11</b>
3.1	Catcode checks for loading	11
3.2	Driver detection	13
<b>4</b>	<b>Installation</b>	<b>13</b>
4.1	Download	13
4.2	Bundle installation	13
4.3	Package installation	14
4.4	Refresh file name databases	14
4.5	Some details for the interested	14

---

\*Please report any issues at <https://github.com/ho-tex/oberdiek/issues>

<b>5 History</b>	<b>14</b>
[2007/12/12 v1.0]	14
[2009/04/10 v1.1]	15
[2010/03/09 v1.2]	15
[2010/12/13 v1.3]	15
[2011/03/29 v1.4]	15
[2011/04/22 v1.5]	15
[2011/04/23 v1.6]	15
[2011/10/22 v1.7]	15
[2011/11/01 v1.8]	15
[2016/05/13 v1.9]	16
[2016/05/16 v1.10]	16
[2018/11/22 v1.11]	16
[2019/07/25 v1.12]	16

# 1 Documentation

## 1.1 Introduction

This package uses a LuaTeX's attribute register to to annotate nodes with color information. If a color is set, then the attribute register is set to this color and all nodes created in its scope (current group) are annotated with this attribute. Now the color property behaves much the same way as the font property.

## 1.2 Usage

Package `color` is loaded automatically by this package `luacolor`. If you need a special driver option or you prefer package `xcolor`, then load it before package `luacolor`, for example:

```
\usepackage[dvipdfmx]{xcolor}
```

The package `luacolor` is loaded without options:

```
\usepackage{luacolor}
```

It is able to detect PDF mode and DVI drivers are differentiated by its color specials. Therefore the package do need driver options.

Then it redefines the color setting commands to set attributes instead of what-sits for color.

At last the attribute annotations of the nodes in the output box must be analyzed to insert the necessary color whatsits. Currently LuaTeX lacks an appropriate callback function. Therefore package `atbegshi` is used to get control before a box is shipped out.

`\luacolorProcessBox {⟨box⟩}`

Macro `\luacolorProcessBox` processes the box `⟨box⟩` in the previously described manner. It is automatically called for pages, but not for XForm objects. Before passing a box to `\pdfxform`, call `\luacolorProcessBox` first.

## 1.3 Limitations

**Ligatures with different colored components:** Package `luacolor` sees the ligature after the paragraph building and page breaking, when a page is to be shipped out. Therefore it cannot break ligatures, because the components might occupy different space. Therefore it is the responsibility of the ligature forming process to deal with different colored glyphs that form a ligature. The user can avoid the problem entirely by explicitly breaking the ligature at the places where the color changes.

...

## 2 Implementation

```
1 (*package)
```

### 2.1 Catcodes and identification

```
2 \begingroup\catcode61\catcode48\catcode32=10\relax%
3 \catcode13=5 % ^~M
4 \endlinechar=13 %
5 \catcode123=1 % {
6 \catcode125=2 % }
7 \catcode64=11 % @
8 \def\x{\endgroup
9 \expandafter\edef\csname LuaCol@AtEnd\endcsname{%
10 \endlinechar=\the\endlinechar\relax
11 \catcode13=\the\catcode13\relax
12 \catcode32=\the\catcode32\relax
13 \catcode35=\the\catcode35\relax
14 \catcode61=\the\catcode61\relax
15 \catcode64=\the\catcode64\relax
16 \catcode123=\the\catcode123\relax
17 \catcode125=\the\catcode125\relax
18 }%
19 }%
20 \x\catcode61\catcode48\catcode32=10\relax%
21 \catcode13=5 % ^~M
22 \endlinechar=13 %
23 \catcode35=6 % #
24 \catcode64=11 % @
25 \catcode123=1 % {
26 \catcode125=2 % }
27 \def\TMP@EnsureCode#1#2{%
28 \edef\LuaCol@AtEnd{%
29 \LuaCol@AtEnd
30 \catcode#1=\the\catcode#1\relax
31 }%
32 \catcode#1=#2\relax
33 }
34 \TMP@EnsureCode{34}{12}% "
35 \TMP@EnsureCode{39}{12}% '
36 \TMP@EnsureCode{40}{12}% (
37 \TMP@EnsureCode{41}{12}% )
38 \TMP@EnsureCode{42}{12}% *
39 \TMP@EnsureCode{43}{12}% +
40 \TMP@EnsureCode{44}{12}% ,
41 \TMP@EnsureCode{45}{12}% -
```

```

42 \TMP@EnsureCode{46}{12}% .
43 \TMP@EnsureCode{47}{12}% /
44 \TMP@EnsureCode{58}{12}% :
45 \TMP@EnsureCode{60}{12}% <
46 \TMP@EnsureCode{62}{12}% >
47 \TMP@EnsureCode{91}{12}% [
48 \TMP@EnsureCode{93}{12}% ]
49 \TMP@EnsureCode{95}{12}% _ (other!)
50 \TMP@EnsureCode{96}{12}% `
51 \edef\LuaCol@AtEnd{\LuaCol@AtEnd\noexpand\endinput}

Package identification.
52 \NeedsTeXFormat{LaTeX2e}
53 \ProvidesPackage{luacolor}%
54 [2019/07/25 v1.12 Color support via LuaTeX's attributes (HO)]

```

## 2.2 Check for LuaTeX

Without LuaTeX there is no point in using this package.

```

55 \RequirePackage{infwarerr}[2010/04/08]%
56 \RequirePackage{ifluatex}[2010/03/01]%
57 \RequirePackage{ifpdf}[2011/01/30]%
58 \RequirePackage{ltxcmds}[2011/04/18]%
59 \RequirePackage{color}

```

require ltuatex rather than luatex package support for LuaTeX allocations.

```

60 \ifluatex
61   \ifx\newattribute\@undefined
62     \RequirePackage{ltluatex}%
63   \fi
64 \else
65   \@PackageError{luacolor}{%
66     This package may only be run using LuaTeX%
67   }\@ehc
68   \expandafter\LuaCol@AtEnd
69 \fi%

```

\LuaCol@directlua

```

70 \let\LuaCol@directlua\directlua

```

## 2.3 Check for disabled colors

```

71 \ifcolors@
72 \else
73   \@PackageWarningNoLine{luacolor}{%
74     Colors are disabled by option `monochrome'%
75   }%
76   \def\set@color{}%
77   \def\reset@color{}%
78   \def\set@page@color{}%
79   \def\define@color#1#2{}%
80   \expandafter\LuaCol@AtEnd
81 \fi%

```

## 2.4 Load module and check version

```

82 \LuaCol@directlua{%
83   require("luacolor")%
84 }
85 \begingroup

```

```

86 \edef\x{\LuaCol@directlua{tex.write("2019/07/25 v1.12")}}%
87 \edef\y{%
88   \LuaCol@directlua{%
89     if oberdiek.luacolor.getversion then %
90       oberdiek.luacolor.getversion()%
91     end%
92   }%
93 }%
94 \ifx\x\y
95 \else
96   \@PackageError{luacolor}{%
97     Wrong version of lua module.\MessageBreak
98     Package version: \x\MessageBreak
99     Lua module: \y
100   }\@ehc
101 \fi
102 \endgroup

```

## 2.5 Find driver

```

103 \ifpdf
104 \else
105   \begingroup
106     \def\current@color{%
107       \def\reset@color{%
108         \setbox\z@=\hbox{%
109           \begingroup
110             \set@color
111           \endgroup
112         }%
113       \edef\reserved@a{%
114         \LuaCol@directlua{%
115           oberdiek.luacolor.dvidetect()%
116         }%
117       }%
118       \ifx\reserved@a@empty
119         \@PackageError{luacolor}{%
120           DVI driver detection failed because of\MessageBreak
121           unrecognized color \string\special
122         }\@ehc
123       \endgroup
124       \expandafter\expandafter\expandafter\LuaCol@AtEnd
125     \else
126       \@PackageInfoNoLine{luacolor}{%
127         Type of color \string\special: \reserved@a
128       }%
129     \fi%
130   \endgroup
131 \fi

```

## 2.6 Attribute setting

\LuaCol@Attribute

```

132 \ltx@ifundefined{newluatexattribute}{%
133   \newattribute\LuaCol@Attribute
134 }{%
135   \newluatexattribute\LuaCol@Attribute
136 }
137 \ltx@ifundefined{setluatexattribute}{%

```

```

138 \let\LuaCol@setattribute\setattribute
139 }{%
140 \let\LuaCol@setattribute\setluatexattribute
141 }
142 \LuaCol@directlua{%
143 oberdiek.luacolor.setattribute(\number\allocationnumber)%
144 }

\set@color
145 \protected\def\set@color{%
146 \LuaCol@setattribute\LuaCol@Attribute{%
147 \LuaCol@directlua{%
148 oberdiek.luacolor.get("\luaescapestring{\current@color}")%
149 }%
150 }%
151 }

\reset@color
152 \def\reset@color{}

```

## 2.7 Whatsit insertion

```

\luacolorProcessBox
153 \def\luacolorProcessBox#1{%
154 \LuaCol@directlua{%
155 oberdiek.luacolor.process(\number#1)%
156 }%
157 }

158 \RequirePackage{atbegshi}[2011/01/30]
159 \AtBeginShipout{%
160 \luacolorProcessBox\AtBeginShipoutBox
161 }

Set default color.
162 \set@color

```

## 2.8 \pdfxform support

```

163 \ifpdf
164 \ifx\pdfxform\@undefined
165 \let\pdfxform\saveboxresource
166 \fi
167 \ltx@ifundefined{pdfxform}{%
168 \directlua{%
169 tex.enableprimitives('',{%
170 'pdfxform','pdflastxform','pdfrefxform'%
171 })%
172 }%
173 }{}%
174 \ltx@ifundefined{protected}{%
175 \directlua{tex.enableprimitives('',{'protected'})}%
176 }{}%
177 \ltx@ifundefined{pdfxform}{%
178 \@PackageWarning{luacolor}{\string\pdfxform\space not found}%
179 }{}%
180 \let\LuaCol@org@pdfxform\pdfxform
181 \begingroup\expandafter\expandafter\expandafter\endgroup

```

```

182 \expandafter\ifx\csname protected\endcsname\relax
183 \PackageWarning{luacolor}{\string\protected\space not found}%
184 \else
185 \expandafter\protected
186 \fi
187 \def\pdfxform{%
188 \begingroup
189 \afterassignment\LuaCol@pdfxform
190 \count@=%
191 }%
192 \def\LuaCol@pdfxform{%
193 \luacolorProcessBox\count@
194 \LuaCol@org@pdfxform\count@
195 \endgroup
196 }%
197 }%
198 \fi
199 \LuaCol@AtEnd%
200 \endpackage

```

## 2.9 Lua module

```

201 \lua

```

Box zero contains a `\hbox` with the color `\special`. That is analyzed to get the prefix for the color setting `\special`.

```

202 oberdiek = oberdiek or {}
203 local luacolor = oberdiek.luacolor or {}
204 oberdiek.luacolor = luacolor

```

```

getversion()

```

```

205 function luacolor.getversion()
206 tex.write("2019/07/25 v1.12")
207 end

```

### 2.9.1 Driver detection

```

208 local ifpdf = tonumber(tex.outputmode or tex.pdfoutput) > 0
209 local prefix
210 local prefixes = {
211 dvips = "color ",
212 dvipdfm = "pdf:sc ",
213 truetex = "textcolor:",
214 pctexps = "ps::",
215 }
216 local patterns = {
217 ["^color "] = "dvips",
218 ["^pdf: *begincolor "] = "dvipdfm",
219 ["^pdf: *bcolor "] = "dvipdfm",
220 ["^pdf: *bc "] = "dvipdfm",
221 ["^pdf: *setcolor "] = "dvipdfm",
222 ["^pdf: *scolor "] = "dvipdfm",
223 ["^pdf: *sc "] = "dvipdfm",
224 ["^textcolor:"] = "truetex",
225 ["^ps::"] = "pctexps",
226 }

```

```

info()

```

```

227 local function info(msg, term)
228 local target = "log"

```

```

229 if term then
230     target = "term and log"
231 end
232 texio.write_nl(target, "Package luacolor info: " .. msg .. ".")
233 texio.write_nl(target, "")
234 end

```

dvidetect()

```

235 function luacolor.dvidetect()
236     local v = tex.box[0]
237     assert(v.id == node.id("hlist"))
238     for v in node.traverse_id(node.id("whatsit"), v.head) do
239         if v and v.subtype == node.subtype("special") then
240             local data = v.data
241             for pattern, driver in pairs(patterns) do
242                 if string.find(data, pattern) then
243                     prefix = prefixes[driver]
244                     tex.write(driver)
245                     return
246                 end
247             end
248             info("\\special{" .. data .. "}", true)
249             return
250         end
251     end
252     info("Missing \\special", true)
253 end

```

## 2.9.2 Color strings

```

254 local map = {
255     n = 0,
256 }

```

get()

```

257 function luacolor.get(color)
258     tex.write("" .. luacolor.getvalue(color))
259 end

```

getvalue()

```

260 function luacolor.getvalue(color)
261     local n = map[color]
262     if not n then
263         n = map.n + 1
264         map.n = n
265         map[n] = color
266         map[color] = n
267     end
268     return n
269 end

```

## 2.9.3 Attribute register

setattribute()

```

270 local attribute
271 function luacolor.setattribute(attr)
272     attribute = attr
273 end

```



getattribute()

```
274 function luacolor.getattribute()
275   return attribute
276 end
```

#### 2.9.4 Whatsit insertion

```
277 local LIST = 1
278 local LIST_LEADERS = 2
279 local LIST_DISC = 3
280 local COLOR = 4
281 local RULE = node.id("rule")
282 local node_types = {
283   [node.id("hlist")] = LIST,
284   [node.id("vlist")] = LIST,
285   [node.id("rule")] = COLOR,
286   [node.id("glyph")] = COLOR,
287   [node.id("disc")] = LIST_DISC,
288   [node.id("whatsit")] = {
289     [node.subtype("special")] = COLOR,
290     [node.subtype("pdf_literal")] = COLOR,
291     [node.subtype("pdf_save")] = COLOR,
292     [node.subtype("pdf_restore")] = COLOR, -- probably not needed
293 -- TODO (DPC)    [node.subtype("pdf_refximage")] = COLOR,
294   },
295   [node.id("glue")] =
296     function(n)
297       if n.subtype >= 100 then -- leaders
298         if n.leader.id == RULE then
299           return COLOR
300         else
301           return LIST_LEADERS
302         end
303       end
304     end,
305 }
```

get\_type()

```
306 local function get_type(n)
307   local ret = node_types[n.id]
308   if type(ret) == 'table' then
309     ret = ret[n.subtype]
310   end
311   if type(ret) == 'function' then
312     ret = ret(n)
313   end
314   return ret
315 end

316 local mode = 2 -- luatex.pdfliteral.direct
317 local WHATSIT = node.id("whatsit")
318 local SPECIAL = node.subtype("special")
319 local PDFLITERAL = node.subtype("pdf_literal")
320 local DRY_FALSE = false
321 local DRY_TRUE = true
```

traverse()

```
322 local function traverse(list, color, dry)
```

```

323 if not list then
324     return color
325 end
326 local head
327 if get_type(list) == LIST then
328     head = list.head
329 elseif get_type(list) == LIST_DISC then
330     head = list.replace
331 else
332     texio.write_nl("!!! Error: Wrong list type: " .. node.type(list.id))
333     return color
334 end
335 (debug)texio.write_nl("traverse: " .. node.type(list.id))
336 for n in node.traverse(head) do
337 (debug)texio.write_nl(" node: " .. node.type(n.id))
338     local t = get_type(n)
339 (debug)texio.write_nl("TYPE " .. tostring(t) .. " " .. tostring(node.type(node.getid(n))) .. " " .. to
340     if t == LIST or t == LIST_DISC then
341         color = traverse(n, color, dry)
342     elseif t == LIST_LEADERS then
343         local color_after = traverse(n.leader, color, DRY_TRUE)
344         if color == color_after then
345             traverse(n.leader, color, DRY_FALSE or dry)
346         else
347             traverse(n.leader, '', DRY_FALSE or dry)
348 % The color status is unknown here, because the leader box
349 % will or will not be set.
350             color = ''
351         end
352     elseif t == COLOR then
353         local v = node.has_attribute(n, attribute)
354         if v then
355             local newColor = map[v]
356             if newColor ~= color then
357                 color = newColor
358                 if dry == DRY_FALSE then
359                     local newNode
360                     if ifpdf then
361                         newNode = node.new(WHATSIT, PDFLITERAL)
362                         newNode.mode = mode
363                         newNode.data = color
364                     else
365                         newNode = node.new(WHATSIT, SPECIAL)
366                         newNode.data = prefix .. color
367                     end
368                     head = node.insert_before(head, n, newNode)
369                 end
370             end
371         end
372     end
373 end
374 if get_type(list) == LIST then
375     list.head = head
376 else
377     list.replace = head
378 end
379 return color
380 end

```

process()

```
381 function luacolor.process(box)
382   local color = ""
383   local list = tex.getbox(box)
384   traverse(list, color, DRY_FALSE)
385 end
386  $\langle$ /lua $\rangle$ 
```

### 3 Test

```
387  $\langle$ *test1 $\rangle$ 
388 \documentclass{article}
389 \usepackage{color}
390  $\langle$ /test1 $\rangle$ 
```

#### 3.1 Catcode checks for loading

```
391  $\langle$ *test1 $\rangle$ 
392 \catcode`\{=1 %
393 \catcode`\}=2 %
394 \catcode`\#=6 %
395 \catcode`\@=11 %
396 \expandafter\ifx\csname count@\endcsname\relax
397   \countdef\count@=255 %
398 \fi
399 \expandafter\ifx\csname @gobble\endcsname\relax
400   \long\def\@gobble#1{}%
401 \fi
402 \expandafter\ifx\csname @firstofone\endcsname\relax
403   \long\def\@firstofone#1{#1}%
404 \fi
405 \expandafter\ifx\csname loop\endcsname\relax
406   \expandafter\@firstofone
407 \else
408   \expandafter\@gobble
409 \fi
410 {%
411   \def\loop#1\repeat{%
412     \def\body{#1}%
413     \iterate
414   }%
415   \def\iterate{%
416     \body
417     \let\next\iterate
418   \else
419     \let\next\relax
420   \fi
421   \next
422 }%
423 \let\repeat=\fi
424 }%
425 \def\RestoreCatcodes{}
426 \count@=0 %
427 \loop
428   \edef\RestoreCatcodes{%
429     \RestoreCatcodes
430     \catcode\the\count@=\the\catcode\count@\relax
```

```

431 }%
432 \ifnum\count@<255 %
433   \advance\count@ 1 %
434 \repeat
435
436 \def\RangeCatcodeInvalid#1#2{%
437   \count@=#1\relax
438   \loop
439     \catcode\count@=15 %
440   \ifnum\count@<#2\relax
441     \advance\count@ 1 %
442   \repeat
443 }
444 \def\RangeCatcodeCheck#1#2#3{%
445   \count@=#1\relax
446   \loop
447     \ifnum#3=\catcode\count@
448   \else
449     \errmessage{%
450       Character \the\count@\space
451       with wrong catcode \the\catcode\count@\space
452       instead of \number#3%
453     }%
454   \fi
455   \ifnum\count@<#2\relax
456     \advance\count@ 1 %
457   \repeat
458 }
459 \def\space{ }
460 \expandafter\ifx\csname LoadCommand\endcsname\relax
461 \def\LoadCommand{\input luacolor.sty\relax}%
462 \fi
463 \def\Test{%
464   \RangeCatcodeInvalid{0}{47}%
465   \RangeCatcodeInvalid{58}{64}%
466   \RangeCatcodeInvalid{91}{96}%
467   \RangeCatcodeInvalid{123}{255}%
468   \catcode`\@=12 %
469   \catcode`\=0 %
470   \catcode`\%=14 %
471   \LoadCommand
472   \RangeCatcodeCheck{0}{36}{15}%
473   \RangeCatcodeCheck{37}{37}{14}%
474   \RangeCatcodeCheck{38}{47}{15}%
475   \RangeCatcodeCheck{48}{57}{12}%
476   \RangeCatcodeCheck{58}{63}{15}%
477   \RangeCatcodeCheck{64}{64}{12}%
478   \RangeCatcodeCheck{65}{90}{11}%
479   \RangeCatcodeCheck{91}{91}{15}%
480   \RangeCatcodeCheck{92}{92}{0}%
481   \RangeCatcodeCheck{93}{96}{15}%
482   \RangeCatcodeCheck{97}{122}{11}%
483   \RangeCatcodeCheck{123}{255}{15}%
484   \RestoreCatcodes
485 }
486 \Test
487 \csname @@end\endcsname
488 \end

```

```
489 </test1>
```

## 3.2 Driver detection

```
490 <*test2>
491 \NeedsTeXFormat{LaTeX2e}
492 \ifcsname driver\endcsname
493   \expandafter\PassOptionsToPackage\expandafter{\driver}{color}%
494   \pdfoutput=0 %
495 \fi
496 \documentclass{minimal}
497 \usepackage{luacolor}[2018/11/01]
498 \csname @@end\endcsname
499 \end
500 </test2>
501 <*test3>
502 \NeedsTeXFormat{LaTeX2e}
503 \documentclass{minimal}
504 \usepackage{luacolor}[2018/11/01]
505 \usepackage{qstest}
506 \IncludeTests{*}
507 \LogTests{log}{*}{*}
508 \makeatletter
509 \@@@end
510 </test3>
```

# 4 Installation

## 4.1 Download

**Package.** This package is available on CTAN<sup>1</sup>:

[CTAN:macros/latex/contrib/oberdiek/luacolor.dtx](#) The source file.

[CTAN:macros/latex/contrib/oberdiek/luacolor.pdf](#) Documentation.

**Bundle.** All the packages of the bundle ‘oberdiek’ are also available in a TDS compliant ZIP archive. There the packages are already unpacked and the documentation files are generated. The files and directories obey the TDS standard.

[CTAN:install/macros/latex/contrib/oberdiek.tds.zip](#)

*TDS* refers to the standard “A Directory Structure for T<sub>E</sub>X Files” ([CTAN:tds/tds.pdf](#)). Directories with `texmf` in their name are usually organized this way.

## 4.2 Bundle installation

**Unpacking.** Unpack the `oberdiek.tds.zip` in the TDS tree (also known as `texmf` tree) of your choice. Example (linux):

```
unzip oberdiek.tds.zip -d ~/texmf
```

**Script installation.** Check the directory `TDS:scripts/oberdiek/` for scripts that need further installation steps. Package `attachfile2` comes with the Perl script `pdfatfi.pl` that should be installed in such a way that it can be called as `pdfatfi`. Example (linux):

```
chmod +x scripts/oberdiek/pdfatfi.pl
cp scripts/oberdiek/pdfatfi.pl /usr/local/bin/
```

---

<sup>1</sup>[CTAN:pkg/luacolor](#)

### 4.3 Package installation

**Unpacking.** The `.dtx` file is a self-extracting `docstrip` archive. The files are extracted by running the `.dtx` through plain  $\mathrm{T}_{\mathrm{E}}\mathrm{X}$ :

```
tex luacolor.dtx
```

**TDS.** Now the different files must be moved into the different directories in your installation TDS tree (also known as `texmf` tree):

```
luacolor.sty → tex/latex/oberdiek/luacolor.sty
luacolor.lua → scripts/oberdiek/luacolor.lua
luacolor.pdf → doc/latex/oberdiek/luacolor.pdf
luacolor.dtx → source/latex/oberdiek/luacolor.dtx
```

If you have a `docstrip.cfg` that configures and enables `docstrip`'s TDS installing feature, then some files can already be in the right place, see the documentation of `docstrip`.

### 4.4 Refresh file name databases

If your  $\mathrm{T}_{\mathrm{E}}\mathrm{X}$  distribution (`te $\mathrm{T}_{\mathrm{E}}\mathrm{X}$` , `mik $\mathrm{T}_{\mathrm{E}}\mathrm{X}$` , ...) relies on file name databases, you must refresh these. For example, `te $\mathrm{T}_{\mathrm{E}}\mathrm{X}$`  users run `texhash` or `mktextlsr`.

### 4.5 Some details for the interested

**Unpacking with  $\mathrm{L}^{\mathrm{A}}\mathrm{T}_{\mathrm{E}}\mathrm{X}$ .** The `.dtx` chooses its action depending on the format:

**plain  $\mathrm{T}_{\mathrm{E}}\mathrm{X}$ :** Run `docstrip` and extract the files.

**$\mathrm{L}^{\mathrm{A}}\mathrm{T}_{\mathrm{E}}\mathrm{X}$ :** Generate the documentation.

If you insist on using  $\mathrm{L}^{\mathrm{A}}\mathrm{T}_{\mathrm{E}}\mathrm{X}$  for `docstrip` (really, `docstrip` does not need  $\mathrm{L}^{\mathrm{A}}\mathrm{T}_{\mathrm{E}}\mathrm{X}$ ), then inform the autodetect routine about your intention:

```
latex \let\install=y\input{luacolor.dtx}
```

Do not forget to quote the argument according to the demands of your shell.

**Generating the documentation.** You can use both the `.dtx` or the `.drv` to generate the documentation. The process can be configured by the configuration file `ltxdoc.cfg`. For instance, put this line into this file, if you want to have A4 as paper format:

```
\PassOptionsToClass{a4paper}{article}
```

An example follows how to generate the documentation with `pdf $\mathrm{L}^{\mathrm{A}}\mathrm{T}_{\mathrm{E}}\mathrm{X}$` :

```
pdflatex luacolor.dtx
makeindex -s gind.ist luacolor.idx
pdflatex luacolor.dtx
makeindex -s gind.ist luacolor.idx
pdflatex luacolor.dtx
```

## 5 History

[2007/12/12 v1.0]

- First public version.

**[2009/04/10 v1.1]**

- Fixes for changed syntax of `\directlua` in LuaTeX 0.36.

**[2010/03/09 v1.2]**

- Adaptation for package `luatex` 2010/03/09 v0.4.

**[2010/12/13 v1.3]**

- Support for `\pdfxform` added.
- Loaded package `luatexbase-attr` recognized.
- Update for LuaTeX: ‘list’ fields renamed to ‘head’ in v0.65.0.

**[2011/03/29 v1.4]**

- Avoid whatsit insertion if option `monochrome` is used (thanks Manuel Pégourié-Gonnard).

**[2011/04/22 v1.5]**

- Bug fix by Manuel Pégourié-Gonnard: A typo prevented the detection of whatsits and applying color changes for `\pdfliteral` and `\special` nodes that might contain typesetting material.
- Bug fix by Manuel Pégourié-Gonnard: Now colors are also applied to leader boxes.
- Unnecessary color settings are removed for leaders boxes, if after the leader box the color has not changed. The costs are a little runtime, leader boxes are processed twice.
- Additional whatsits that are colored: `pdf_refximage`.
- Workaround for bug with `node.insert_before` removed for the version after LuaTeX 0.65, because bug was fixed in 0.27. (Thanks Manuel Pégourié-Gonnard.)

**[2011/04/23 v1.6]**

- Bug fix for nested leader boxes.
- Bug fix for leader boxes that change color, but are not set because of missing place.
- Version check for Lua module added.

**[2011/10/22 v1.7]**

- Lua functions `getattribute` and `getvalue` added to tell other external Lua functions the attribute register number for coloring.

**[2011/11/01 v1.8]**

- Use of `node.subtype` instead of magic numbers.

[2016/05/13 v1.9]

- More use of `node.subtype` instead of magic numbers.
- luatex 85 updates

[2016/05/16 v1.10]

- Documentation updates.

[2018/11/22 v1.11]

- handle issue 43.
- removed pre-0.65 stuff

[2019/07/25 v1.12]

- removed uses of module function, see PR70

## 6 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; plain numbers refer to the code lines where the entry is used.

Symbols		26, 30, 32, 392, 393, 394, 395,	
<code>\#</code> .....	394	430, 439, 447, 451, 468, 469, 470	
<code>\%</code> .....	470	<code>\count@</code> .... 190, 193, 194, 397, 426,	
<code>\@</code> .....	395, 468	430, 432, 433, 437, 439, 440,	
<code>\@@end</code> .....	509	441, 445, 447, 450, 451, 455, 456	
<code>\@PackageError</code> .....	65, 96, 119	<code>\countdef</code> .....	
<code>\@PackageInfoNoLine</code> .....	126	397	
<code>\@PackageWarning</code> .....	178, 183	<code>\csname</code> .....	
<code>\@PackageWarningNoLine</code> .....	73	9, 182,	
<code>\@ehc</code> .....	67, 100, 122	396, 399, 402, 405, 460, 487, 498	
<code>\@empty</code> .....	118	<code>\current@color</code> .....	
<code>\@firstofone</code> .....	403, 406	106, 148	
<code>\@gobble</code> .....	400, 408	D	
<code>\@undefined</code> .....	61, 164	<code>\define@color</code> .....	
<code>\</code> .....	248, 252, 469	79	
<code>\{</code> .....	392	<code>\directlua</code> .....	
<code>\}</code> .....	393	70, 168, 175	
		<code>\documentclass</code> .....	
		388, 496, 503	
		<code>\driver</code> .....	
		493	
		<code>\dvidetect()</code> .....	
		<u>235</u>	
		E	
		<code>\end</code> .....	
		488, 499	
		<code>\endcsname</code> .....	
		9, 182, 396,	
		399, 402, 405, 460, 487, 492, 498	
		<code>\endinput</code> .....	
		51	
		<code>\endlinechar</code> .....	
		4, 10, 22	
		<code>\errmessage</code> .....	
		449	
		G	
		<code>\get()</code> .....	
		<u>257</u>	
		<code>\get_type()</code> .....	
		<u>306</u>	
		<code>\getattribute()</code> .....	
		<u>274</u>	
		<code>\getvalue()</code> .....	
		<u>260</u>	
		<code>\getversion()</code> .....	
		<u>205</u>	
A			
<code>\advance</code> .....	433, 441, 456		
<code>\afterassignment</code> .....	189		
<code>\allocationnumber</code> .....	143		
<code>\AtBeginShipout</code> .....	159		
<code>\AtBeginShipoutBox</code> .....	160		
B			
<code>\body</code> .....	412, 416		
C			
<code>\catcode</code> .....	2, 3, 5, 6, 7, 11, 12, 13, 14,		
	15, 16, 17, 20, 21, 23, 24, 25,		



<b>H</b>		<b>\protected</b> ..... 145, 183, 185	
<b>\hbox</b> .....	108	<b>\ProvidesPackage</b> .....	53
<b>I</b>		<b>R</b>	
<b>\ifcolors@</b> .....	71	<b>\RangeCatcodeCheck</b> .....	
<b>\ifcsname</b> .....	492	. 444, 472, 473, 474, 475, 476,	
<b>\ifluatex</b> .....	60	477, 478, 479, 480, 481, 482, 483	
<b>\ifnum</b> .....	432, 440, 447, 455	<b>\RangeCatcodeInvalid</b> .....	
<b>\ifpdf</b> .....	103, 163	..... 436, 464, 465, 466, 467	
<b>\ifx</b> .....	61, 94, 118,	<b>\repeat</b> .....	411, 423, 434, 442, 457
164, 182, 396, 399, 402, 405, 460		<b>\RequirePackage</b> .....	
<b>\IncludeTests</b> .....	506	..... 55, 56, 57, 58, 59, 62, 158	
<b>\info()</b> .....	227	<b>\reserved@a</b> .....	113, 118, 127
<b>\input</b> .....	461	<b>\reset@color</b> .....	77, 107, 152
<b>\iterate</b> .....	413, 415, 417	<b>\RestoreCatcodes</b> ..	425, 428, 429, 484
<b>L</b>		<b>S</b>	
<b>\LoadCommand</b> .....	461, 471	<b>\saveboxresource</b> .....	165
<b>\LogTests</b> .....	507	<b>\set@color</b> .....	76, 110, 145, 162
<b>\loop</b> .....	411, 427, 438, 446	<b>\set@page@color</b> .....	78
<b>\ltx@ifundefined</b> 132, 137, 167, 174, 177		<b>\setattribute</b> .....	138
<b>\LuaCol@AtEnd</b> 28, 29, 51, 68, 80, 124, 199		<b>\setattribute()</b> .....	270
<b>\LuaCol@Attribute</b> .....	132, 146	<b>\setbox</b> .....	108
<b>\LuaCol@directlua</b> .....		<b>\setluatexattribute</b> .....	140
. 70, 82, 86, 88, 114, 142, 147, 154		<b>\space</b> .....	178, 183, 450, 451, 459
<b>\LuaCol@org@pdfxform</b> .....	180, 194	<b>\special</b> .....	121, 127
<b>\LuaCol@pdfxform</b> .....	189, 192	<b>T</b>	
<b>\LuaCol@setattribute</b> ..	138, 140, 146	<b>\Test</b> .....	463, 486
<b>\luacolorProcessBox</b> ..	2, 153, 160, 193	<b>\the</b> .....	10, 11, 12,
<b>\luaescapestring</b> .....	148	13, 14, 15, 16, 17, 30, 430, 450, 451	
<b>M</b>		<b>\TMP@EnsureCode</b> .....	27,
<b>\makeatletter</b> .....	508	34, 35, 36, 37, 38, 39, 40, 41,	
<b>\MessageBreak</b> .....	97, 98, 120	42, 43, 44, 45, 46, 47, 48, 49, 50	
<b>N</b>		<b>\traverse()</b> .....	322
<b>\NeedsTeXFormat</b> .....	52, 491, 502	<b>U</b>	
<b>\newattribute</b> .....	61, 133	<b>\usepackage</b> .....	389, 497, 504, 505
<b>\newluatexattribute</b> .....	135	<b>X</b>	
<b>\next</b> .....	417, 419, 421	<b>\x</b> .....	8, 20, 86, 94, 98
<b>\number</b> .....	143, 155, 452	<b>Y</b>	
<b>P</b>		<b>\y</b> .....	87, 94, 99
<b>\PassOptionsToPackage</b> .....	493	<b>Z</b>	
<b>\pdfoutput</b> .....	494	<b>\z@</b> .....	108
<b>\pdfxform</b> .....	164, 165, 178, 180, 187		
<b>\process()</b> .....	381		