

The luacolor package

Heiko Oberdiek*

2019/11/29 v1.13

Abstract

Package luacolor implements color support based on LuaTeX's node attributes.

Contents

1	Documentation	2
1.1	Introduction	2
1.2	Usage	2
1.3	Limitations	3
2	Implementation	3
2.1	Catcodes and identification	3
2.2	Check for LuaTeX	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	\pdfxform 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
3	Test	11
3.1	Catcode checks for loading	11
3.2	Driver detection	13
4	Installation	13
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/luacolor/issues>

5 History	14
[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
[2019/11/29 v1.13]	16
6 Index	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 [2019/11/29 v1.13 Color support via LuaTeX's attributes (H0)]

```

2.2 Check for LuaTeX

Without LuaTeX there is no point in using this package.

```

55 \RequirePackage{infwarerr}[2010/04/08]%
56 \RequirePackage{iftex}[2019/11/07]%
57 \RequirePackage{ltxcmds}[2011/04/18]%
58 \RequirePackage{color}

```

require ltuatex rather than luatex package support for LuaTeX allocations.

```

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

```

\LuaCol@directlua

```

69 \let\LuaCol@directlua\directlua

```

2.3 Check for disabled colors

```

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

```

2.4 Load module and check version

```

81 \LuaCol@directlua{%
82   require("luacolor")%
83 }

84 \begingroup
85   \edef\x{\LuaCol@directlua{tex.write("2019/11/29 v1.13")}}%

```

```

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

```

2.5 Find driver

```

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

```

2.6 Attribute setting

\LuaCol@Attribute

```

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

```

```

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

\set@color

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

\reset@color

151 \def\reset@color{}

```

2.7 Whatsit insertion

```

\luacolorProcessBox

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

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

    Set default color.

161 \set@color

```

2.8 \pdfxform support

```

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

```

```

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

198 \LuaCol@AtEnd%
199 \</package>

```

2.9 Lua module

```

200 <lua>

```

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

```

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

```

```

getversion()

```

```

204 function luacolor.getversion()
205   tex.write("2019/11/29 v1.13")
206 end

```

2.9.1 Driver detection

```

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

```

```

info()

```

```

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

```

```

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

dvidetect()

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

```

2.9.2 Color strings

```

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

get()

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

getvalue()

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

```

2.9.3 Attribute register

```

setattribute()

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

```

getattribute()

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

2.9.4 Whatsit insertion

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

get_type()

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

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

traverse()

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

```

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

```

process()

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

3 Test

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

3.1 Catcode checks for loading

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

```

430 }%
431 \ifnum\count@<255 %
432   \advance\count@ 1 %
433 \repeat
434
435 \def\RangeCatcodeInvalid#1#2{%
436   \count@=#1\relax
437   \loop
438     \catcode\count@=15 %
439   \ifnum\count@<#2\relax
440     \advance\count@ 1 %
441   \repeat
442 }
443 \def\RangeCatcodeCheck#1#2#3{%
444   \count@=#1\relax
445   \loop
446     \ifnum#3=\catcode\count@
447     \else
448       \errmessage{%
449         Character \the\count@\space
450         with wrong catcode \the\catcode\count@\space
451         instead of \number#3%
452       }%
453     \fi
454   \ifnum\count@<#2\relax
455     \advance\count@ 1 %
456   \repeat
457 }
458 \def\space{ }
459 \expandafter\ifx\csname LoadCommand\endcsname\relax
460 \def\LoadCommand{\input luacolor.sty\relax}%
461 \fi
462 \def\Test{%
463   \RangeCatcodeInvalid{0}{47}%
464   \RangeCatcodeInvalid{58}{64}%
465   \RangeCatcodeInvalid{91}{96}%
466   \RangeCatcodeInvalid{123}{255}%
467   \catcode'\@=12 %
468   \catcode'\=0 %
469   \catcode'\%=14 %
470   \LoadCommand
471   \RangeCatcodeCheck{0}{36}{15}%
472   \RangeCatcodeCheck{37}{37}{14}%
473   \RangeCatcodeCheck{38}{47}{15}%
474   \RangeCatcodeCheck{48}{57}{12}%
475   \RangeCatcodeCheck{58}{63}{15}%
476   \RangeCatcodeCheck{64}{64}{12}%
477   \RangeCatcodeCheck{65}{90}{11}%
478   \RangeCatcodeCheck{91}{91}{15}%
479   \RangeCatcodeCheck{92}{92}{0}%
480   \RangeCatcodeCheck{93}{96}{15}%
481   \RangeCatcodeCheck{97}{122}{11}%
482   \RangeCatcodeCheck{123}{255}{15}%
483   \RestoreCatcodes
484 }
485 \Test
486 \csname @@end\endcsname
487 \end

```

488 \langle /test1 \rangle

3.2 Driver detection

```
489  $\langle$ *test2 $\rangle$ 
490 \NeedsTeXFormat{LaTeX2e}
491 \ifcsname driver\endcsname
492   \expandafter\PassOptionsToPackage\expandafter{\driver}{color}%
493   \pdfoutput=0 %
494 \fi
495 \documentclass{minimal}
496 \usepackage{luacolor}[2018/11/01]
497 \csname @@end\endcsname
498 \end
499  $\langle$ /test2 $\rangle$ 

500  $\langle$ *test3 $\rangle$ 
501 \NeedsTeXFormat{LaTeX2e}
502 \documentclass{minimal}
503 \usepackage{luacolor}[2018/11/01]
504 \usepackage{qstest}
505 \IncludeTests{*}
506 \LogTests{log}{*}{*}
507 \makeatletter
508 @@end
509  $\langle$ /test3 $\rangle$ 
```

4 Installation

4.1 Download

Package. This package is available on CTAN¹:

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

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

Bundle. All the packages of the bundle ‘luacolor’ 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/luacolor.tds.zip](#)

TDS refers to the standard “A Directory Structure for T_EX Files” ([CTAN:pkg/tds](#)). Directories with `texmf` in their name are usually organized this way.

4.2 Bundle installation

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

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

Script installation. Check the directory `TDS:scripts/luacolor/` for scripts that need further installation steps.

¹[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 `TEX`:

```
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/luacolor/luacolor.sty
luacolor.lua → scripts/luacolor/luacolor.lua
luacolor.pdf → doc/latex/luacolor/luacolor.pdf
luacolor.dtx → source/latex/luacolor/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 `TEX` distribution (`TEX Live`, `mikTEX`, ...) relies on file name databases, you must refresh these. For example, `TEX Live` users run `texhash` or `mktexlsr`.

4.5 Some details for the interested

Unpacking with L^AT_EX. The `.dtx` chooses its action depending on the format:

plain T_EX: Run `docstrip` and extract the files.

L^AT_EX: Generate the documentation.

If you insist on using L^AT_EX for `docstrip` (really, `docstrip` does not need L^AT_EX), 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 pdfL^AT_EX:

```
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.

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

- Documentation updates.

- handle issue 43.
- removed pre-0.65 stuff

- removed uses of module function, see PR70

- Documentation updates.
- Use iftex directly.

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.

<code>\endcsname</code>	9, 181, 395, 398, 401, 404, 459, 486, 491, 497	<code>\number</code>	142, 154, 451
<code>\endinput</code>	51	P	
<code>\endlinechar</code>	4, 10, 22	<code>\PassOptionsToPackage</code>	492
<code>\errmessage</code>	448	<code>\pdfoutput</code>	493
G		<code>\pdfxform</code>	163, 164, 177, 179, 186
<code>\get()</code>	256	<code>\process()</code>	380
<code>\get_type()</code>	305	<code>\protected</code>	144, 182, 184
<code>\getattribute()</code>	273	<code>\ProvidesPackage</code>	53
<code>\getvalue()</code>	259	R	
<code>\getversion()</code>	204	<code>\RangeCatcodeCheck</code>	
H		443, 471, 472, 473, 474, 475, 476, 477, 478, 479, 480, 481, 482
<code>\hbox</code>	107	<code>\RangeCatcodeInvalid</code>	
I		435, 463, 464, 465, 466
<code>\ifcolors@</code>	70	<code>\repeat</code>	410, 422, 433, 441, 456
<code>\ifcsname</code>	491	<code>\RequirePackage</code>	55, 56, 57, 58, 61, 157
<code>\ifluatex</code>	59	<code>\reserved@a</code>	112, 117, 126
<code>\ifnum</code>	431, 439, 446, 454	<code>\reset@color</code>	76, 106, 151
<code>\ifpdf</code>	102, 162	<code>\RestoreCatcodes</code>	424, 427, 428, 483
<code>\ifx</code>	60, 93, 117, 163, 181, 395, 398, 401, 404, 459	S	
<code>\IncludeTests</code>	505	<code>\saveboxresource</code>	164
<code>\info()</code>	226	<code>\set@color</code>	75, 109, 144, 161
<code>\input</code>	460	<code>\set@page@color</code>	77
<code>\iterate</code>	412, 414, 416	<code>\setattribute</code>	137
L		<code>\setattribute()</code>	269
<code>\LoadCommand</code>	460, 470	<code>\setbox</code>	107
<code>\LogTests</code>	506	<code>\setluatexattribute</code>	139
<code>\loop</code>	410, 426, 437, 445	<code>\space</code>	177, 182, 449, 450, 458
<code>\ltx@ifundefined</code>	131, 136, 166, 173, 176	<code>\special</code>	120, 126
<code>\LuaCol@AtEnd</code>	28, 29, 51, 67, 79, 123, 198	T	
<code>\LuaCol@Attribute</code>	131, 145	<code>\Test</code>	462, 485
<code>\LuaCol@directlua</code>		<code>\the</code>	10, 11, 12, 13, 14, 15, 16, 17, 30, 429, 449, 450
.	69, 81, 85, 87, 113, 141, 146, 153	<code>\TMP@EnsureCode</code>	27, 34, 35, 36, 37, 38, 39, 40, 41, 42, 43, 44, 45, 46, 47, 48, 49, 50
<code>\LuaCol@org@pdfxform</code>	179, 193	<code>\traverse()</code>	321
<code>\LuaCol@pdfxform</code>	188, 191	U	
<code>\LuaCol@setattribute</code>	137, 139, 145	<code>\usepackage</code>	388, 496, 503, 504
<code>\luacolorProcessBox</code>	2, 152, 159, 192	X	
<code>\luaescapestring</code>	147	<code>\x</code>	8, 20, 85, 93, 97
M		Y	
<code>\makeatletter</code>	507	<code>\y</code>	86, 93, 98
<code>\MessageBreak</code>	96, 97, 119	Z	
N		<code>\z@</code>	107
<code>\NeedsTeXFormat</code>	52, 490, 501		
<code>\newattribute</code>	60, 132		
<code>\newluatexattribute</code>	134		
<code>\next</code>	416, 418, 420		