

The luacolor package

Heiko Oberdiek*

2026-08-26 v1.19

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	2
2	Implementation	3
2.1	Catcodes and identification	3
2.2	Check for LuaTeX	3
2.3	Check for disabled colors	4
2.4	Load module and check version	4
2.5	Find driver	4
2.6	Attribute setting	5
2.7	l3color support	5
2.8	Whatsit insertion	6
2.9	<code>\pdfxform/\saveboxresource</code> support	6
2.10	Lua module	8
2.10.1	Driver detection	8
2.10.2	Color strings	9
2.10.3	Attribute register	9
2.10.4	Whatsit insertion	9
3	Installation	12
3.1	Download	12
3.2	Package installation	12
3.3	Refresh file name databases	12
3.4	Some details for the interested	12
4	History	13
	[2007/12/12 v1.0]	13
	[2009/04/10 v1.1]	13
	[2010/03/09 v1.2]	13
	[2010/12/13 v1.3]	13
	[2011/03/29 v1.4]	13
	[2011/04/22 v1.5]	13
	[2011/04/23 v1.6]	13
	[2011/10/22 v1.7]	13
	[2011/11/01 v1.8]	14
	[2016/05/13 v1.9]	14

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

[2016/05/16 v1.10]	14
[2018/11/22 v1.11]	14
[2019/07/25 v1.12]	14
[2019/11/29 v1.13]	14
[2020-02-22 v1.14]	14
[2020-02-24 v1.15]	14
[2020-04-04 v1.16]	14
[2021-02-17 v1.17]	14
[2023-08-18 v1.18]	14
[2026-08-26 v1.19]	15

5 Index 15

1 Documentation

1.1 Introduction

This package uses a LuaTeX's attribute register 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. (This assumes that the callback function `pre_shipout_filter` exists).

`\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 [2026-08-26 v1.19 Color support via LuaTeX's attributes (H0)]
```

2.2 Check for LuaTeX

Without LuaTeX there is no point in using this package.

```

55 \RequirePackage{color}
56 \ifx\directlua\@undefined
57   \PackageError{luacolor}{%
58     This package may only be run using LuaTeX%
59   }\@ehc
60   \expandafter\LuaCol@AtEnd
61 \fi%

```

2.3 Check for disabled colors

```

62 \ifcolors@
63 \else
64   \PackageWarningNoLine{luacolor}{%
65     Colors are disabled by option 'monochrome'%
66   }%
67   \def\set@color{}%
68   \def\reset@color{}%
69   \def\set@page@color{}%
70   \def\define@color#1#2{}%
71   \expandafter\LuaCol@AtEnd
72 \fi%

```

2.4 Load module and check version

```

73 \directlua{%
74   require("luacolor")%
75 }
76 \begingroup
77   \edef\x{\directlua{tex.write("2026-08-26 v1.19")}}%
78   \edef\y{%
79     \directlua{%
80       if oberdiek.luacolor.getversion then %
81         oberdiek.luacolor.getversion()%
82       end%
83     }%
84   }%
85   \ifx\x\y
86   \else
87     \PackageError{luacolor}{%
88       Wrong version of lua module.\MessageBreak
89       Package version: \x\MessageBreak
90       Lua module: \y
91     }\@ehc
92   \fi
93 \endgroup

```

2.5 Find driver

```

94 \ifnum\outputmode=\@ne
95 \else
96   \begingroup
97     \def\current@color{}%
98     \def\reset@color{}%
99     \setbox\z@=\hbox{%
100       \begingroup
101         \set@color
102       \endgroup
103     }%
104     \edef\reserved@a{%
105       \directlua{%
106         oberdiek.luacolor.dvidetect()%
107       }%
108     }%

```

```

109 \ifx\reserved@a\@empty
110 \PackageError{luacolor}{%
111     DVI driver detection failed because of\MessageBreak
112     unrecognized color \string\special
113 }{\@ehc
114 \endgroup
115 \expandafter\expandafter\expandafter\LuaCol@AtEnd
116 \else
117 \PackageInfo{luacolor}{%
118     Type of color \string\special: \reserved@a
119     \@gobble}%
120 \fi%
121 \endgroup
122 \fi

```

2.6 Attribute setting

\LuaCol@Attribute

```

123 \newattribute\LuaCol@Attribute
124 \let\LuaCol@setattribute\setattribute
125 \directlua{%
126     oberdiek.luacolor.setattribute(\number\allocationnumber)%
127 }

```

\set@color change 2023-08-18: added \reset@color so that \mathcolor can gobble it, issue 7.

```

128 \protected\def\set@color{%
129     \LuaCol@setattribute\LuaCol@Attribute{%
130         \directlua{%
131             oberdiek.luacolor.get("\luaescapestring{\current@color}")%
132         }%
133     }%
134 \aftergroup\reset@color
135 }

```

\reset@color

```

136 \def\reset@color{}

```

2.7 l3color support

Added 2026-08-26. We redefine the color backend commands. This is temporary code and assumes that a current l3backend is in use. We ensure that the backend is loaded first.

```

\__color_backend_select:w
\__color_backend_fill:n 137 \ExplSyntaxOn
\__color_backend_stroke:n 138 \AddToHook{package/luacolor/after}[pdfmanagement-firstaid]{}
139 \RemoveFromHook{package/luacolor/after}[pdfmanagement-firstaid]
140 \sys_ensure_backend:
141 \cs_set_protected:Npn \__color_backend_select:w #1 \c_space_tl #2 \q_stop
142 {
143     \tl_set:Nn \l__color_backend_fill_tl {#1}
144     \tl_set:Nn \l__color_backend_stroke_tl {#2}
145     \LuaCol@setattribute\LuaCol@Attribute{%
146         \directlua{%
147             oberdiek.luacolor.get("\luaescapestring{#1~#2}")%
148         }}
149 }
150 \cs_set_protected:Npn \__color_backend_fill:n #1
151 {
152     \tl_set:Nn \l__color_backend_fill_tl {#1}
153     \LuaCol@setattribute\LuaCol@Attribute{%

```

```

154     \directlua{%
155         oberdiek.luacolor.get("\luaescapestring{#1~\l__color_backend_stroke_tl}")%
156     }}
157 }
158 \cs_set_protected:Npn \__color_backend_stroke:n #1
159 {
160     \tl_set:Nn \l__color_backend_stroke_tl {#1}
161     \LuaCol@setattribute\LuaCol@Attribute{%
162         \directlua{%
163             oberdiek.luacolor.get("\luaescapestring{#1~\l__color_backend_fill_tl}")%
164         }}
165 }

```

```

\__color_backend_reset:
\__color_backend_fill_reset: 166 \cs_set_protected:Npn \__color_backend_reset: { }
\__color_backend_stroke_reset: 167 \cs_set_eq:NN \__color_backend_fill_reset: \__color_backend_reset:
168 \cs_set_eq:NN \__color_backend_stroke_reset: \__color_backend_reset:
169 \ExplSyntaxOff

```

2.8 Whatsit insertion

```

\luacolorProcessBox
170 \def\luacolorProcessBox#1{%
171     \directlua{%
172         oberdiek.luacolor.process(\number#1)%
173     }%
174 }

Set default color.
175 \set@color

```

2.9 \pdfxform/\saveboxresource support

```

176 \ifnum\outputmode=\@ne
177     \let\LuaCol@org@pdfxform\saveboxresource

```

First we need some helpers to allow expandable code to parse keyword style arguments:

```

178     \def\LuaCol@iii@i@ii#1#2#3#{#1}{#2}}
179     \def\LuaCol@ii@i#1#2#{#2#1}}
180     \def\LuaCol@if@keyword#1#2#3{%
181         \expanded{\unexpanded{\LuaCol@iii@i@ii{#2}{#3}}\expandafter}%
182         \directlua{%
183             token.put_next(token.create(token.scan_keyword(token.scan_string()))
184             and '@firstoftwo'
185             or '@secondoftwo'))
186         }{#1}%
187     }

```

The following macro scans a integer and expands to a token equivalent to a chardef whose value corresponds to the scanned integer. This allows the integer to be passed around as a undelimited argument.

```

188     \def\LuaCol@scan@number{%
189         \directlua{
190             token.put_next(token.new(token.scan_int(), token.command_id'char_given'))
191         }%
192     }

```

$\TeX$ primitives like `\saveboxresource` read braced arguments in a special way. Especially they expand everything until they find a left brace. To simulate this, we use Lua to expand everything else:

```

193     \def\LuaCol@scan@tobraces{%
194         \directlua{

```

```

195     local relax, space = token.command_id'relax', token.command_id'spacer'
196     local t
197     repeat
198         t = token.scan_token()
199     until not (t.command == relax or t.command == space)
200     token.put_next(t)
201 }%
202 }
203 \def\LuaCol@scan@boxresource@i#1#2{%
204     \LuaCol@if@keyword{attr}{%
205         \expanded{\unexpanded{\LuaCol@scan@boxresource@ii{#1#2attr}}}%
206         \expandafter\expandafter\expandafter}%
207     \LuaCol@scan@tobraces
208 }{%
209     \LuaCol@scan@boxresource@ii{#1#2}%
210 }%
211 }
212 \def\LuaCol@scan@boxresource@ii#1#2{\LuaCol@scan@boxresource@ii{#1{#2}}}
213 \def\LuaCol@scan@boxresource@iii#1{%
214     \LuaCol@if@keyword{resources}{%
215         \expanded{\unexpanded{\LuaCol@scan@boxresource@iii{#1resources}}}%
216         \expandafter\expandafter\expandafter}%
217     \LuaCol@scan@tobraces
218 }{%
219     \LuaCol@scan@boxresource@iii{#1}%
220 }%
221 }
222 \def\LuaCol@scan@boxresource@iii#1#2{\LuaCol@scan@boxresource@iii{#1{#2}}}
223 \def\LuaCol@scan@boxresource@iv#1{%
224     \LuaCol@if@keyword{margin}{%
225         \expanded{\unexpanded{\LuaCol@scan@boxresource@iv{#1margin }}}%
226         \expandafter\expandafter\expandafter}%
227     \LuaCol@scan@number
228 }{%
229     \LuaCol@scan@boxresource@iv{#1}{}%
230 }%
231 }
232 \def\LuaCol@scan@boxresource@iv#1#2{%
233     \expanded{\unexpanded{\LuaCol@scan@boxresource@v{#1#2}}}%
234     \expandafter\expandafter\expandafter}%
235     \LuaCol@scan@number
236 }
237 \def\LuaCol@scan@boxresource@v#1#2{%
238     \luacolorProcessBox{#2}%
239     \LuaCol@org@pdfxform#1#2%
240 }
241

```

This could be written in Lua, but at least upto LuaTeX 1.11, feeding back too many tokens from Lua to TeX triggers a segmentation fault. This is written in Lua so the integer setting is expandable and does not interfere with a preceding `\immediate`.

```

242 \protected\def\saveboxresource{%
243     \LuaCol@if@keyword{type}{%
244         \expandafter
245         \expanded{\unexpanded{\LuaCol@scan@boxresource@i{type }}}%
246         \expandafter\expandafter\expandafter}%
247     \LuaCol@scan@number
248 }{%
249     \LuaCol@scan@boxresource@i{}{}%
250 }%
251 }

```

Legacy alias.

```

252 \let\pdfxform\saveboxresource
253 \fi
254 \LuaCol@AtEnd%
255 \end{package}

```

2.10 Lua module

```

256 \lua

```

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

```

257 oberdiek = oberdiek or {}
258 local luacolor = oberdiek.luacolor or {}
259 oberdiek.luacolor = luacolor

```

```

getversion()

```

```

260 function luacolor.getversion()
261   tex.write("2026-08-26 v1.19")
262 end

```

2.10.1 Driver detection

```

263 local ifpdf = tonumber(tex.outputmode or tex.pdfoutput) > 0
264 local prefix
265 local prefixes = {
266   dvips = "color ",
267   dvipdfm = "pdf:sc ",
268   truetex = "textcolor:",
269   pTeXps = "ps::",
270 }
271 local patterns = {
272   ["^color "] = "dvips",
273   ["^pdf: *begincolor "] = "dvipdfm",
274   ["^pdf: *bcolor "] = "dvipdfm",
275   ["^pdf: *bc "] = "dvipdfm",
276   ["^pdf: *setcolor "] = "dvipdfm",
277   ["^pdf: *scolor "] = "dvipdfm",
278   ["^pdf: *sc "] = "dvipdfm",
279   ["^textcolor:"] = "truetex",
280   ["^ps::"] = "pTeXps",
281 }

```

```

info()

```

```

282 local function info(msg, term)
283   local target = "log"
284   if term then
285     target = "term and log"
286   end
287   texio.write_nl(target, "Package luacolor info: " .. msg .. ".")
288   texio.write_nl(target, "")
289 end

```

```

dvidetect()

```

```

290 function luacolor.dvidetect()
291   local v = tex.box[0]
292   assert(v.id == node.id("hlist"))
293   for v in node.traverse_id(node.id("whatsit"), v.head) do
294     if v and v.subtype == node.subtype("special") then
295       local data = v.data
296       for pattern, driver in pairs(patterns) do
297         if string.find(data, pattern) then
298           prefix = prefixes[driver]
299           tex.write(driver)
300           return

```

```

301         end
302     end
303     info("\\special{" .. data .. "}", true)
304     return
305 end
306 end
307 info("Missing \\special", true)
308 end

```

2.10.2 Color strings

```

309 local map = {
310     n = 0,
311 }

get()

312 function luacolor.get(color)
313     tex.write("" .. luacolor.getvalue(color))
314 end

getvalue()

315 function luacolor.getvalue(color)
316     local n = map[color]
317     if not n then
318         n = map.n + 1
319         map.n = n
320         map[n] = color
321         map[color] = n
322     end
323     return n
324 end

```

2.10.3 Attribute register

```

setattribute()

325 local attribute
326 function luacolor.setattribute(attr)
327     attribute = attr
328 end

getattribute()

329 function luacolor.getattribute()
330     return attribute
331 end

```

2.10.4 Whatsit insertion

```

332 local LIST = 1
333 local LIST_LEADERS = 2
334 local LIST_DISC = 3
335 local COLOR = 4
336 local NOCOLOR = 5
337 local RULE = node.id("rule")
338 local node_types = {
339     [node.id("hlist")] = LIST,
340     [node.id("vlist")] = LIST,
341     [node.id("rule")] = COLOR,
342     [node.id("glyph")] = COLOR,
343     [node.id("disc")] = LIST_DISC,
344     [node.id("whatsit")] = {
345         [node.subtype("pdf_colorstack")] =
346             function(n)

```

```

347         return n.stack == 0 and NOCOLOR or nil
348     end,
349     [node.subtype("special")] = COLOR,
350     [node.subtype("pdf_literal")] = COLOR,
351     [node.subtype("pdf_save")] = COLOR,
352     [node.subtype("pdf_restore")] = COLOR, -- probably not needed
353 -- TODO (DPC)     [node.subtype("pdf_refximage")] = COLOR,
354 },
355 [node.id("glue")] =
356     function(n)
357         if n.subtype >= 100 then -- leaders
358             if n.leader.id == RULE then
359                 return COLOR
360             else
361                 return LIST_LEADERS
362             end
363         end
364     end,
365 }

get_type()

366 local function get_type(n)
367     local ret = node_types[n.id]
368     if type(ret) == 'table' then
369         ret = ret[n.subtype]
370     end
371     if type(ret) == 'function' then
372         ret = ret(n)
373     end
374     return ret
375 end

376 local mode = 2 -- luatex.pdf_literal.direct
377 local WHATSIT = node.id("whatsit")
378 local SPECIAL = node.subtype("special")
379 local PDFLITERAL = node.subtype("pdf_literal")
380 local DRY_FALSE = false
381 local DRY_TRUE = true

traverse()

382 local function traverse(list, color, dry)
383     if not list then
384         return color
385     end
386     local head
387     if get_type(list) == LIST then
388         head = list.head
389     elseif get_type(list) == LIST_DISC then
390         head = list.replace
391     else
392         texio.write_nl("!!! Error: Wrong list type: " .. node.type(list.id))
393         return color
394     end
395     <debug>texio.write_nl("traverse: " .. node.type(list.id))
396     for n in node.traverse(head) do
397         <debug>texio.write_nl(" node: " .. node.type(n.id))
398         local t = get_type(n)
399         <debug>texio.write_nl("TYPE " .. tostring(t) .. " " .. tostring(node.type(node.getid(n))) .. " " .. t)
400         if t == LIST or t == LIST_DISC then
401             color = traverse(n, color, dry)
402         elseif t == LIST_LEADERS then
403             local color_after = traverse(n.leader, color, DRY_TRUE)
404             if color == color_after then

```

```

405         traverse(n.leader, color, DRY_FALSE or dry)
406     else
407         traverse(n.leader, '', DRY_FALSE or dry)

```

The color status is unknown here, because the leader box will or will not be set.

```

408         color = ''
409     end
410     elseif t == COLOR then
411         local v = node.has_attribute(n, attribute)
412         if v then
413             local newColor = map[v]
414             if newColor ~= color then
415                 color = newColor
416                 if dry == DRY_FALSE then
417                     local newNode
418                     if ifpdf then
419                         newNode = node.new(WHATSIT, PDFLITERAL)
420                         newNode.mode = mode
421                         newNode.data = color
422                     else
423                         newNode = node.new(WHATSIT, SPECIAL)
424                         newNode.data = prefix .. color
425                     end
426                     head = node.insert_before(head, n, newNode)
427                 end
428             end
429         end
430     elseif t == NOCOLOR then
431         color = ''
432     end
433 end
434 if get_type(list) == LIST then
435     list.head = head
436 else
437     list.replace = head
438 end
439 return color
440 end

```

process()

```

441 function luacolor.process(box)
442     local color = ""
443     local list = tex.getbox(box)
444     traverse(list, color, DRY_FALSE)
445 end
446
447 if luatexbase.callbacktypes.pre_shipout_filter then
448     luatexbase.add_to_callback("pre_shipout_filter", function(list)
449         traverse(list, "", DRY_FALSE)
450         return true
451     end, "luacolor.process")
452 end

```

For recent versions of luaotfload, we can register a callback to control how coloring glyph is handled for the color feature.

```

453 if luaotfload.set_colorhandler then
454     local set_attribute = node.direct.set_attribute
455     luaotfload.set_colorhandler(function(head, n, color)
456         set_attribute(n, attribute, luacolor.getvalue(color))
457         return head, n
458     end)
459 end
460 </lua>

```

3 Installation

3.1 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.

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

3.2 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`.

3.3 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 `mktextlsr`.

3.4 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
```

¹[CTAN:pkg/luacolor](#)

4 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

[2019/11/29 v1.13]

- Documentation updates.
- Use `iftex` directly.

[2020-02-22 v1.14]

- Drop use of `iftex ltxcmds` and `infwarerr`.
- Assume `lualatex` preloaded into format (true since 2015).
- Patch `\saveboxresource` rather than `\pdfxform` (keep old name as alias).
- Grab the number via Lua so that a `\immediate` prefix still works with `\saveboxresource/\pdfxform`.
- Added handler for the color feature of `luaotfload`

[2020-02-24 v1.15]

- Grab all possible arguments for `\saveboxresource/\pdfxform`

[2020-04-04 v1.16]

- Reset color after `pdf_colorstack` whatsits.

[2021-02-17 v1.17]

- Use $\text{\LaTeX 2}_{\epsilon}$'s new `pre_shipout_filter` callback if it's available to allow coloring background and foreground layer material

[2023-08-18 v1.18]

- added `\reset@color` to `\set@color` for `\mathcolor`, issue 7.

[2026-08-26 v1.19]

- Added support for l3color, see
<https://github.com/latex3/tagging-project/issues/1550>

5 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	H
<code>\@ehc</code> 59, 91, 113	<code>\hbox</code> 99
<code>\@empty</code> 109	
<code>\@gobble</code> 119	I
<code>\@ne</code> 94, 176	<code>\ifcolors@</code> 62
<code>\@undefined</code> 56	<code>\ifnum</code> 94, 176
<code>\@</code> 303, 307	<code>\ifx</code> 56, 85, 109
<code>_</code> 141, 150, 158, 166, 167, 168	<code>\info()</code> 282
<code>_color_backend_fill:n</code> 137	
<code>_color_backend_fill_reset:</code> ... 166	L
<code>_color_backend_reset:</code> 166	<code>\l</code> 143, 144, 152, 155, 160, 163
<code>_color_backend_select:w</code> 137	<code>\LuaCol@AtEnd</code> 28, 29, 51, 60, 71, 115, 254
<code>_color_backend_stroke:n</code> 137	<code>\LuaCol@Attribute</code> 123, 129, 145, 153, 161
<code>_color_backend_stroke_reset:</code> . 166	<code>\LuaCol@if@keyword</code> 180, 204, 214, 224, 243
A	<code>\LuaCol@ii@i</code> 179
<code>\AddToHook</code> 138	<code>\LuaCol@iii@i@ii</code> 178, 181
<code>\aftergroup</code> 134	<code>\LuaCol@org@pdfxform</code> 177, 239
<code>\allocationnumber</code> 126	<code>\LuaCol@scan@boxresource@i</code> 203, 245, 249
C	<code>\LuaCol@scan@boxresource@iI</code> 205, 212
<code>\c</code> 141	<code>\LuaCol@scan@boxresource@ii</code> 209, 212, 213
<code>\catcode</code> 2, 3, 5, 6, 7, 11, 12, 13, 14, 15, 16, 17, 20, 21, 23, 24, 25, 26, 30, 32	<code>\LuaCol@scan@boxresource@iiI</code> 215, 222
<code>\cs</code> 141, 150, 158, 166, 167, 168	<code>\LuaCol@scan@boxresource@iii</code> 219, 222, 223
<code>\csname</code> 9	<code>\LuaCol@scan@boxresource@iv</code> 225, 229, 232
<code>\current@color</code> 97, 131	<code>\LuaCol@scan@boxresource@v</code> . 233, 237
D	<code>\LuaCol@scan@number</code> 188, 227, 235, 247
<code>\define@color</code> 70	<code>\LuaCol@scan@tobrace</code> .. 193, 207, 217
<code>\directlua</code> 56, 73, 77, 79, 105, 125, 130, 146, 154, 162, 171, 182, 189, 194	<code>\LuaCol@setattribute</code> 124, 129, 145, 153, 161
<code>\dvidetect()</code> 290	<code>\luacolorProcessBox</code> 2, 170, 238
E	<code>\luaescapestring</code> .. 131, 147, 155, 163
<code>\endcsname</code> 9	M
<code>\endinput</code> 51	<code>\MessageBreak</code> 88, 89, 111
<code>\endlinechar</code> 4, 10, 22	N
<code>\expanded</code> . 181, 205, 215, 225, 233, 245	<code>\NeedsTeXFormat</code> 52
<code>\ExplSyntaxOff</code> 169	<code>\newattribute</code> 123
<code>\ExplSyntaxOn</code> 137	<code>\number</code> 126, 172
G	
<code>\get()</code> 312	O
<code>\get_type()</code> 366	<code>\outputmode</code> 94, 176
<code>\getattribute()</code> 329	P
<code>\getvalue()</code> 315	<code>\PackageError</code> 57, 87, 110
<code>\getversion()</code> 260	<code>\PackageInfo</code> 117

<code>\PackageWarningNoLine</code>	64	<code>\special</code>	112, 118
<code>\pdfxform</code>	252	<code>\sys</code>	140
<code>\process()</code>	441		
<code>\protected</code>	128, 242		
<code>\ProvidesPackage</code>	53		
		T	
		<code>\the</code> . . .	10, 11, 12, 13, 14, 15, 16, 17, 30
		<code>\tl</code>	143, 144, 152, 160
		<code>\TMP@EnsureCode</code>	27,
			34, 35, 36, 37, 38, 39, 40, 41,
			42, 43, 44, 45, 46, 47, 48, 49, 50
		<code>\traverse()</code>	382
		U	
		<code>\unexpanded</code>	181, 205, 215, 225, 233, 245
		X	
		<code>\x</code>	8, 20, 77, 85, 89
		Y	
		<code>\y</code>	78, 85, 90
		Z	
		<code>\z@</code>	99
Q			
<code>\q</code>	141		
R			
<code>\RemoveFromHook</code>	139		
<code>\RequirePackage</code>	55		
<code>\reserved@a</code>	104, 109, 118		
<code>\reset@color</code>	68, 98, 134, 136		
S			
<code>\saveboxresource</code>	177, 242, 252		
<code>\set@color</code>	67, 101, 128, 175		
<code>\set@page@color</code>	69		
<code>\setattribute</code>	124		
<code>\setattribute()</code>	325		
<code>\setbox</code>	99		