The luacolor package

Heiko Oberdiek*
<heiko.oberdiek at googlemail.com>

2016/05/16 v1.10

Abstract

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

Contents

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

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 whatsits 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 respondibility 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

```
⟨*package⟩
```

2.1 Catcodes and identification

```
\begingroup\catcode61\catcode48\catcode32=10\relax%
  \catcode13=5 % ^^M
  \endlinechar=13 %
  \catcode123=1 % {
  \catcode125=2 % }
  \catcode64=11 % @
  \def\x{\endgroup
    \expandafter\edef\csname LuaCol@AtEnd\endcsname{%
      \endlinechar=\the\endlinechar\relax
      \catcode13=\the\catcode13\relax
      \catcode32=\the\catcode32\relax
      \catcode35=\the\catcode35\relax
      \catcode61=\the\catcode61\relax
      \catcode64=\the\catcode64\relax
      \catcode123=\the\catcode123\relax
      \catcode125=\the\catcode125\relax
    }%
  }%
\x\catcode61\catcode48\catcode32=10\relax%
\catcode13=5 % ^^M
\endlinechar=13 %
\catcode35=6 % #
\catcode64=11 % @
\catcode123=1 % {
\catcode125=2 % }
\def\TMP@EnsureCode#1#2{%
  \edef\LuaCol@AtEnd{%
    \LuaCol@AtEnd
    \catcode#1=\the\catcode#1\relax
  }%
  \catcode#1=#2\relax
}
\TMP@EnsureCode{34}{12}% "
\TMP@EnsureCode{39}{12}% '
\TMP@EnsureCode{40}{12}% (
\TMP@EnsureCode{41}{12}% )
\TMP@EnsureCode{42}{12}% *
\TMP@EnsureCode{43}{12}% +
\TMP@EnsureCode{44}{12}% ,
\TMP@EnsureCode{45}{12}% -
\TMP@EnsureCode{46}{12}% .
\TMP@EnsureCode{47}{12}% /
\TMP@EnsureCode{58}{12}% :
\TMP@EnsureCode{60}{12}% <
\TMP@EnsureCode{62}{12}% >
\TMP@EnsureCode{91}{12}% [
\TMP@EnsureCode{93}{12}% ]
\TMP@EnsureCode{95}{12}% _ (other!)
\TMP@EnsureCode{96}{12}% `
\edef\LuaCol@AtEnd{\LuaCol@AtEnd\noexpand\endinput}
```

Package identification.

```
\NeedsTeXFormat{LaTeX2e}
\ProvidesPackage{luacolor}%
  [2016/05/16 v1.10 Color support via LuaTeX's attributes (HO)]
```

2.2 Check for LuaTeX

Without LuaTeX there is no point in using this package.

```
\RequirePackage{infwarerr}[2010/04/08]%
\RequirePackage{ifluatex}[2010/03/01]%
\RequirePackage{ifpdf}[2011/01/30]%
\RequirePackage{ltxcmds}[2011/04/18]%
\RequirePackage{color}
\ifluatex
  \ifx\newattribute\@undefined
  \ltx@ifpackageloaded{luatexbase-attr}{%
  }{%
    \RequirePackage{luatex}[2010/03/09]%
  }%
 \fi
\else
  \@PackageError{luacolor}{%
    This package may only be run using LuaTeX%
  }\@ehc
  \expandafter\LuaCol@AtEnd
\fi%
```

\LuaCol@directlua

```
\ifnum\luatexversion<36 %
  \def\LuaCol@directlua{\directlua0 }%
\else
  \let\LuaCol@directlua\directlua
\fi
```

2.3 Check for disabled colors

```
\ifcolors@
\else
  \@PackageWarningNoLine{luacolor}{%
    Colors are disabled by option `monochrome'%
  }%
  \def\set@color{}%
  \def\reset@color{}%
  \def\set@page@color{}%
  \def\define@color#1#2{}%
  \expandafter\LuaCol@AtEnd
\fi%
```

2.4 Load module and check version

```
\LuaCol@directlua{%
  require("oberdiek.luacolor\ifnum\luatexversion<65 -pre065\fi")%
}
\begingroup
  \edef\x{\LuaCol@directlua{tex.write("2016/05/16 v1.10")}}%
  \edef\y{%
    \LuaCol@directlua{%
      if oberdiek.luacolor.getversion then %
        oberdiek.luacolor.getversion()%
      end%
    }%
  }%
  \ifx\x\y
  \else
    \@PackageError{luacolor}{%
      Wrong version of lua module.\MessageBreak
      Package version: \x\MessageBreak
```

```
      Lua module: \y
    }\@ehc
  \fi
\endgroup
```

2.5 Find driver

```
\ifpdf
\else
  \begingroup
    \def\current@color{}%
    \def\reset@color{}%
    \setbox\z@=\hbox{%
      \begingroup
        \set@color
      \endgroup
    }%
    \edef\reserved@a{%
      \LuaCol@directlua{%
        oberdiek.luacolor.dvidetect()%
      }%
    }%
    \ifx\reserved@a\@empty
      \@PackageError{luacolor}{%
        DVI driver detection failed because of\MessageBreak
        unrecognized color \string\special
      }\@ehc
      \endgroup
      \expandafter\expandafter\expandafter\LuaCol@AtEnd
    \else
      \@PackageInfoNoLine{luacolor}{%
        Type of color \string\special: \reserved@a
      }%
    \fi%
  \endgroup
\fi
```

2.6 Attribute setting

\LuaCol@Attribute

```
\ltx@IfUndefined{newluatexattribute}{%
  \newattribute\LuaCol@Attribute
}{%
  \newluatexattribute\LuaCol@Attribute
}
\ltx@IfUndefined{setluatexattribute}{%
  \let\LuaCol@setattribute\setattribute
}{%
  \let\LuaCol@setattribute\setluatexattribute
}
\LuaCol@directlua{%
  oberdiek.luacolor.setattribute(\number\allocationnumber)%
}
```

\set@color

```
\protected\def\set@color{%
  \LuaCol@setattribute\LuaCol@Attribute{%
    \LuaCol@directlua{%
      oberdiek.luacolor.get("\luaescapestring{\current@color}")%
    }%
  }%
}
```

\reset@color

```
\def\reset@color{}
```

2.7 Whatsit insertion

\luacolorProcessBox

```
\def\luacolorProcessBox#1{%
  \LuaCol@directlua{%
    oberdiek.luacolor.process(\number#1)%
  }%
}
\RequirePackage{atbegshi}[2011/01/30]
\AtBeginShipout{%
  \luacolorProcessBox\AtBeginShipoutBox
}
```

Set default color.

```
\set@color
```

2.8 \pdfxform support

```
\ifpdf
  \ifx\pdfxform\@undefined
    \let\pdfxform\saveboxresource
  \fi
  \ltx@IfUndefined{pdfxform}{%
    \ifnum\luatexversion>36 %
      \directlua{%
        tex.enableprimitives('',{%
          'pdfxform','pdflastxform','pdfrefxform'%
        })%
      }%
    \fi
  }{}%
  \ltx@IfUndefined{protected}{%
    \ifnum\luatexversion>36 %
      \directlua{tex.enableprimitives('',{'protected'})}%
    \fi
  }{}%
  \ltx@IfUndefined{pdfxform}{%
    \@PackageWarning{luacolor}{\string\pdfxform\space not found}%
  }{%
    \let\LuaCol@org@pdfxform\pdfxform
    \begingroup\expandafter\expandafter\expandafter\endgroup
    \expandafter\ifx\csname protected\endcsname\relax
      \@PackageWarning{luacolor}{\string\protected\space not found}%
    \else
      \expandafter\protected
    \fi
    \def\pdfxform{%
      \begingroup
      \afterassignment\LuaCol@pdfxform
      \count@=%
    }%
    \def\LuaCol@pdfxform{%
      \luacolorProcessBox\count@
      \LuaCol@org@pdfxform\count@
      \endgroup
    }%
  }%
\fi
\LuaCol@AtEnd%
```

```
⟨/package⟩
```

2.9 Lua module

```
⟨*lua⟩
```

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

```
module("oberdiek.luacolor", package.seeall)
```

getversion()

```
function getversion()
  tex.write("2016/05/16 v1.10")
end
```

2.9.1 Driver detection

```
local ifpdf
if tonumber(tex.outputmode or tex.pdfoutput) > 0 then
  ifpdf = true
else
  ifpdf = false
end
local prefix
local prefixes = {
  dvips   = "color ",
  dvipdfm = "pdf:sc ",
  truetex = "textcolor:",
  pctexps = "ps::",
}
local patterns = {
  ["^color "]           = "dvips",
  ["^pdf: *begincolor "] = "dvipdfm",
  ["^pdf: *bcolor "]     = "dvipdfm",
  ["^pdf: *bc "]         = "dvipdfm",
  ["^pdf: *setcolor "]   = "dvipdfm",
  ["^pdf: *scolor "]     = "dvipdfm",
  ["^pdf: *sc "]         = "dvipdfm",
  ["^textcolor:"]        = "truetex",
  ["^ps::"]             = "pctexps",
}
```

info()

```
local function info(msg, term)
  local target = "log"
  if term then
    target = "term and log"
  end
  texio.write_nl(target, "Package luacolor info: " .. msg .. ".")
  texio.write_nl(target, "")
end
```

dvidetect()

```
function dvidetect()
  local v = tex.box[0]
  assert(v.id == node.id("hlist"))
⟨!pre065⟩  for v in node.traverse_id(node.id("whatsit"), v.head) do
⟨pre065⟩  for v in node.traverse_id(node.id("whatsit"), v.list) do
    if v and v.subtype == node.subtype("special") then
      local data = v.data
      for pattern, driver in pairs(patterns) do
        if string.find(data, pattern) then
          prefix = prefixes[driver]
          tex.write(driver)
```

```
260           return
261         end
262       end
263       info("\\special{" .. data .. "}", true)
264       return
265     end
266   end
267   info("Missing \\special", true)
268 end
```

2.9.2 Color strings

```
269 local map = {
270   n = 0,
271 }
```

get()

```
272 function get(color)
273   tex.write("" .. getvalue(color))
274 end
```

getvalue()

```
275 function getvalue(color)
276   local n = map[color]
277   if not n then
278     n = map.n + 1
279     map.n = n
280     map[n] = color
281     map[color] = n
282   end
283   return n
284 end
```

2.9.3 Attribute register

setattribute()

```
285 local attribute
286 function setattribute(attr)
287   attribute = attr
288 end
```

getattribute()

```
289 function getattribute()
290   return attribute
291 end
```

2.9.4 Whatsit insertion

```
292 local LIST = 1
293 local LIST_LEADERS = 2
294 local COLOR = 3
295 local RULE = node.id("rule")
296 local node_types = {
297   [node.id("hlist")] = LIST,
298   [node.id("vlist")] = LIST,
299   [node.id("rule")]  = COLOR,
300   [node.id("glyph")] = COLOR,
301   [node.id("disc")]  = COLOR,
302   [node.id("whatsit")] = {
303     [node.subtype("special")] = COLOR,
304     [node.subtype("pdf_literal")] = COLOR,
305 -- TODO (DPC)     [node.subtype("pdf_refximage")] = COLOR,
```

```
306   },
307   [node.id("glue")] =
308     function(n)
309       if n.subtype >= 100 then -- leaders
310         if n.leader.id == RULE then
311           return COLOR
312         else
313           return LIST_LEADERS
314         end
315       end
316     end,
317 }
```

get_type()

```
318 local function get_type(n)
319   local ret = node_types[n.id]
320   if type(ret) == 'table' then
321     ret = ret[n.subtype]
322   end
323   if type(ret) == 'function' then
324     ret = ret(n)
325   end
326   return ret
327 end

328 local mode = 2 -- luatex.pdfliteral.direct
329 local WHATSIT = node.id("whatsit")
330 local SPECIAL = node.subtype("special")
331 local PDFLITERAL = node.subtype("pdf_literal")
332 local DRY_FALSE = false
333 local DRY_TRUE = true
```

traverse()

```
334 local function traverse(list, color, dry)
335   if not list then
336     return color
337   end
338   if get_type(list) ~= LIST then
339     texio.write_nl("!!! Error: Wrong list type: " .. node.type(list.id))
340     return color
341   end
342 ⟨debug⟩texio.write_nl("traverse: " .. node.type(list.id))
343 ⟨!pre065⟩  local head = list.head
344 ⟨pre065⟩  local head = list.list
345   for n in node.traverse(head) do
346 ⟨debug⟩texio.write_nl("  node: " .. node.type(n.id))
347     local t = get_type(n)
348     if t == LIST then
349       color = traverse(n, color, dry)
350     elseif t == LIST_LEADERS then
351       local color_after = traverse(n.leader, color, DRY_TRUE)
352       if color == color_after then
353         traverse(n.leader, color, DRY_FALSE or dry)
354       else
355         traverse(n.leader, '', DRY_FALSE or dry)
356 % The color status is unknown here, because the leader box
357 % will or will not be set.
358         color = ''
359       end
360     elseif t == COLOR then
361       local v = node.has_attribute(n, attribute)
362       if v then
363         local newColor = map[v]
```

```
        if newColor ~= color then
          color = newColor
          if dry == DRY_FALSE then
            local newNode
            if ifpdf then
              newNode = node.new(WHATSIT, PDFLITERAL)
              newNode.mode = mode
              newNode.data = color
            else
              newNode = node.new(WHATSIT, SPECIAL)
              newNode.data = prefix .. color
            end
⟨*!pre065⟩
            head = node.insert_before(head, n, newNode)
⟨/!pre065⟩
⟨*pre065⟩
            if head == n then
              newNode.next = head
              local old_prev = head.prev
              head.prev = newNode
              head = newNode
              head.prev = old_prev
            else
              head = node.insert_before(head, n, newNode)
            end
⟨/pre065⟩
          end
        end
      end
    end
  end
⟨!pre065⟩  list.head = head
⟨pre065⟩  list.list = head
  return color
end
```

process()

```
function process(box)
  local color = ""
  local list = tex.getbox(box)
  traverse(list, color, DRY_FALSE)
end

⟨/lua⟩
```

3 Test

```
⟨*test1⟩
\documentclass{article}
\usepackage{color}
⟨/test1⟩
```

3.1 Catcode checks for loading

```
⟨*test1⟩
\catcode`\{=1 %
\catcode`\}=2 %
\catcode`\#=6 %
\catcode`\@=11 %
\expandafter\ifx\csname count@\endcsname\relax
  \countdef\count@=255 %
\fi
```

```
\expandafter\ifx\csname @gobble\endcsname\relax
  \long\def\@gobble#1{}%
\fi
\expandafter\ifx\csname @firstofone\endcsname\relax
  \long\def\@firstofone#1{#1}%
\fi
\expandafter\ifx\csname loop\endcsname\relax
  \expandafter\@firstofone
\else
  \expandafter\@gobble
\fi
{%
  \def\loop#1\repeat{%
    \def\body{#1}%
    \iterate
  }%
  \def\iterate{%
    \body
      \let\next\iterate
    \else
      \let\next\relax
    \fi
    \next
  }%
  \let\repeat=\fi
}%
\def\RestoreCatcodes{}
\count@=0 %
\loop
  \edef\RestoreCatcodes{%
    \RestoreCatcodes
    \catcode\the\count@=\the\catcode\count@\relax
  }%
\ifnum\count@<255 %
  \advance\count@ 1 %
\repeat

\def\RangeCatcodeInvalid#1#2{%
  \count@=#1\relax
  \loop
    \catcode\count@=15 %
  \ifnum\count@<#2\relax
    \advance\count@ 1 %
  \repeat
}
\def\RangeCatcodeCheck#1#2#3{%
  \count@=#1\relax
  \loop
    \ifnum#3=\catcode\count@
    \else
      \errmessage{%
        Character \the\count@\space
        with wrong catcode \the\catcode\count@\space
        instead of \number#3%
      }%
    \fi
  \ifnum\count@<#2\relax
    \advance\count@ 1 %
  \repeat
}
\def\space{ }
\expandafter\ifx\csname LoadCommand\endcsname\relax
```

```
  \def\LoadCommand{\input luacolor.sty\relax}%
\fi
\def\Test{%
  \RangeCatcodeInvalid{0}{47}%
  \RangeCatcodeInvalid{58}{64}%
  \RangeCatcodeInvalid{91}{96}%
  \RangeCatcodeInvalid{123}{255}%
  \catcode`\@=12 %
  \catcode`\\=0 %
  \catcode`\%=14 %
  \LoadCommand
  \RangeCatcodeCheck{0}{36}{15}%
  \RangeCatcodeCheck{37}{37}{14}%
  \RangeCatcodeCheck{38}{47}{15}%
  \RangeCatcodeCheck{48}{57}{12}%
  \RangeCatcodeCheck{58}{63}{15}%
  \RangeCatcodeCheck{64}{64}{12}%
  \RangeCatcodeCheck{65}{90}{11}%
  \RangeCatcodeCheck{91}{91}{15}%
  \RangeCatcodeCheck{92}{92}{0}%
  \RangeCatcodeCheck{93}{96}{15}%
  \RangeCatcodeCheck{97}{122}{11}%
  \RangeCatcodeCheck{123}{255}{15}%
  \RestoreCatcodes
}
\Test
\csname @@end\endcsname
\end
```

⟨/test1⟩

3.2 Driver detection

```
⟨*test2⟩
\NeedsTeXFormat{LaTeX2e}
\ifcsname driver\endcsname
  \expandafter\PassOptionsToPackage\expandafter{\driver}{color}%
  \pdfoutput=0 %
\fi
\documentclass{minimal}
\usepackage{luacolor}[2016/05/16]
\csname @@end\endcsname
\end
⟨/test2⟩
⟨*test3⟩
\NeedsTeXFormat{LaTeX2e}
\documentclass{minimal}
\usepackage{luacolor}[2016/05/16]
\usepackage{qstest}
\IncludeTests{*}
\LogTests{log}{*}{*}
\makeatletter
\@@end
⟨/test3⟩
```

4 Installation

4.1 Download

Package. This package is available on CTAN[1]:

CTAN:macros/latex/contrib/oberdiek/luacolor.dtx The source file.

[1] http://ctan.org/pkg/luacolor

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 TEX 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/
```

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/oberdiek/luacolor.sty
oberdiek.luacolor.lua	→ scripts/oberdiek/oberdiek.luacolor.lua
luacolor.lua	→ scripts/oberdiek/luacolor.lua
oberdiek.luacolor-pre065.lua	→ scripts/oberdiek/oberdiek.luacolor-pre065.lua
luacolor-pre065.lua	→ scripts/oberdiek/luacolor-pre065.lua
luacolor.pdf	→ doc/latex/oberdiek/luacolor.pdf
test/luacolor-test1.tex	→ doc/latex/oberdiek/test/luacolor-test1.tex
test/luacolor-test2.tex	→ doc/latex/oberdiek/test/luacolor-test2.tex
test/luacolor-test3.tex	→ doc/latex/oberdiek/test/luacolor-test3.tex
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 TEX distribution (teTEX, mikTEX, ...) relies on file name databases, you must refresh these. For example, teTEX users run texhash or mktexlsr.

4.5 Some details for the interested

Unpacking with LaTeX. The .dtx chooses its action depending on the format:

plain TeX: Run docstrip and extract the files.

LaTeX: Generate the documentation.

If you insist on using LaTeX for docstrip (really, docstrip does not need LaTeX), 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 pdfLaTeX:

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

5 Catalogue

The following XML file can be used as source for the TeX Catalogue. The elements caption and description are imported from the original XML file from the Catalogue. The name of the XML file in the Catalogue is luacolor.xml.

```
⟨*catalogue⟩
<?xml version='1.0' encoding='us-ascii'?>
<!DOCTYPE entry SYSTEM 'catalogue.dtd'>
<entry datestamp='$Date$' modifier='$Author$' id='luacolor'>
  <name>luacolor</name>
  <caption>Color support based on LuaTeX's node attributes.</caption>
  <authorref id='auth:oberdiek'/>
  <copyright owner='Heiko Oberdiek' year='2007,2009-2011'/>
  <license type='lppl1.3'/>
  <version number='1.10'/>
  <description>
     This package implements color support based on LuaTeX's node
     attributes.
   <p/>
   The package is part of the <xref refid='oberdiek'>oberdiek</xref> bundle.
  </description>
  <documentation details='Package documentation'
     href='ctan:/macros/latex/contrib/oberdiek/luacolor.pdf'/>
  <ctan file='true' path='/macros/latex/contrib/oberdiek/luacolor.dtx'/>
  <miktex location='oberdiek'/>
  <texlive location='oberdiek'/>
  <install path='/macros/latex/contrib/oberdiek/oberdiek.tds.zip'/>
</entry>
⟨/catalogue⟩
```

6 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.

7 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.

M

N

P

R

S

T

U

X

Y

Z