Abstract

Package luacolor implements color support based on Lua\TeX{}'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 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 \texttt{\textbackslash 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 ................................ 8

3 Test 10
3.1 Catcode checks for loading ............................... 10
3.2 Driver detection ..................................... 12

4 Installation 12
4.1 Download ............................................. 12
4.2 Bundle installation .................................... 13
4.3 Package installation ................................... 13
4.4 Refresh file name databases ............................. 13
4.5 Some details for the interested ......................... 14

5 Catalogue 14

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

1.1 Introduction

This package uses a Lua\TeX{}'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 \texttt{color} is loaded automatically by this package \texttt{luacolor}. If you need a special driver option or you prefer package \texttt{xcolor}, then load it before package \texttt{luacolor}, for example:

\begin{verbatim}
\usepackage[dvipdfmx]{xcolor}
\end{verbatim}

The package \texttt{luacolor} is loaded without options:

\begin{verbatim}
\usepackage{luacolor}
\end{verbatim}

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 whatsis for color.

At last the attribute annotations of the nodes in the output box must be analyzed to insert the necessary color whatsis. Currently Lua\TeX{} lacks an appropriate callback function. Therefore package \texttt{atbegshi} is used to get control before a box is shipped out.

\begin{verbatim}
\luacolorProcessBox{⟨box⟩}
\end{verbatim}

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

1.3 Limitations

Ligatures with different colored components: Package \texttt{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

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% 
\endlinechar=\the\endlinechar\relax 
\catcode13=\the\catcode13\relax 
\catcode35=\the\catcode35\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\LuaCol@AtEnd\noexpand\endinput}
\PackageIdentification.
\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{ltexcmds}[2011/04/18]%
\RequirePackage{color}

\ifluatex
\ifx\newattribute\@undefined
\ltx@ifpackageloaded{luatexbase-attr}{%}
\else
\fi
\else
\PackageError{luacolor}{This package may only be run using \LaTeXX}{%}
\fi

2.3 Check for disabled colors

\ifcolors@
\else
\PackageWarningNoLine{luacolor}{Colors are disabled by option `monochrome'}%
\fi
\def\set@color{}
\def\reset@color{}
\def\set@page@color{}
\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.\MessageBreakPackage version: \x\MessageBreak}{%}
\fi
\endgroup
\edef\x{\LuaCol@directlua{tex.write("2016/05/16 v1.10")}}%
2.5 Find driver

2.6 Attribute setting
2.7 What'sit insertion

\reset@color{}

\def\luacolorProcessBox#1{%
  \LuaCol@directlua{\oberdiek.luacolor.process(\number#1)}%
%
}\AtBeginShipout{\luacolorProcessBox\AtBeginShipoutBox}

Set default color.
\set@color

2.8 \pdfxform support

\ifpdf
  \ifx\pdfxform@undefined
    \let\pdfxform\saveboxresource
  \fi
  \ifx\pdfxform\undefined
    \let\pdfxform\saveboxresource
  \fi
  \ifnum\luatexversion>36
    \directlua{tex.enableprimitives('',{\pdfxform,\pdflastxform,\pdfrefxform})}
  \fi
  \fi
  \ifnum\luatexversion>36
    \directlua{tex.enableprimitives('',{"protected"})}%
  \fi
  \fi
  \if\pdfxform@undefined
    \PackageWarning{luacolor}{\string\pdfxform\space not found}%
  \else
  \fi
  \begin{group}
  \afterassignment\LuaCol@pdfxform
  \count@=%
  \def\LuaCol@pdfxform{%
    \luacolorProcessBox\count@
    \LuaCol@org@pdfxform\count@
  }
  \end{group}

  \fi

  \if\pdfxform@undefined
    \PackageWarning{luacolor}{\string\pdfxform\space not found}%
  \else
  \fi

  \def\pdfxform{%
    \begingroup
    \afterassignment\LuaCol@pdfxform
    \count@=\%
  }

  \def\LuaCol@pdfxform{%
    \luacolorProcessBox\count@
    \LuaCol@org@pdfxform\count@
  }

  \end{group}

  \fi

\end{group}
2.9 Lua module

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)
          end
        end
      end
    end
  end
end
local map = {
  n = 0,
}

function get(color)
  return tex.write("" .. getvalue(color))
end

function getvalue(color)
  local n = map[color]
  if not n then
    n = map.n + 1
    map.n = n
    map[n] = color
    map[color] = n
  end
  return n
end

local attribute
function setattribute(attr)
  attribute = attr
end

function getattribute()
  return attribute
end

local LIST = 1
local LIST_LEADERS = 2
local COLOR = 3
local RULE = node.id("rule")
local node_types = {
  [node.id("hlist")] = LIST,
  [node.id("vlist")] = LIST,
  [node.id("rule")] = COLOR,
  [node.id("glyph")] = COLOR,
  [node.id("disc")] = COLOR,
  -- TODO (DPC) [node.subtype("special")]] = COLOR,
  [node.subtype("pdf_literal")]] = COLOR,
  [node.subtype("pdf_refximage")]] = COLOR,
function(n)
  if n.subtype >= 100 then -- leaders
    if n.leader.id == RULE then
      return COLOR
    else
      return LIST_LEADERS
    end
  end
end,

local function get_type(n)
  local ret = node_types[n.id]
  if type(ret) == 'table' then
    ret = ret[n.subtype]
  end
  if type(ret) == 'function' then
    ret = ret(n)
  end
  return ret
end

local mode = 2 -- luatex.pdfliteral.direct
local WHATSIT = node.id("whatsit")
local SPECIAL = node.subtype("special")
local PDFLITERAL = node.subtype("pdf_literal")
local DRY_FALSE = false
local DRY_TRUE = true

local function traverse(list, color, dry)
  if not list then
    return color
  end
  if get_type(list) ~= LIST then
    texio.write_nl("!!! Error: Wrong list type: " .. node.type(list.id))
    return color
  end
  ⟨debug⟩
  texio.write_nl("traverse: " .. node.type(list.id))
  ⟨!pre065⟩
  local head = list.head
  ⟨pre065⟩
  local head = list.list
  for n in node.traverse(head) do
    ⟨debug⟩
    texio.write_nl(" node: " .. node.type(n.id))
    local t = get_type(n)
    if t == LIST then
      color = traverse(n, color, dry)
    elseif t == LIST_LEADERS then
      local color_after = traverse(n.leader, color, DRY_TRUE)
      if color == color_after then
        traverse(n.leader, color, DRY_FALSE or dry)
      else
        traverse(n.leader, '', DRY_FALSE or dry)
        % The color status is unknown here, because the leader box
        % will or will not be set.
        color = ''
      end
    elseif t == COLOR then
      local v = node.has_attribute(n, attribute)
      if v then
        local newColor = map[v]
      end
    end
  end
end

get_type()
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
    head = node.insert_before(head, n, newNode)
  end
end
end
end
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

3 Test

Test

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

3.1 Catcode checks for loading

\catcode`\{=1 \%
\catcode`\}=2 \%
\catcode`\#=6 \%
\catcode`\@=11 \%
\expandafter\ifx\csname count\endcsname\relax\countdef\count@=255 \%
\fi
\expandafter\ifx\csname count\endcsname\relax\countdef\count@=255 \%
\fi
3.2 Driver detection

4 Installation

4.1 Download

Package. This package is available on CTAN\(^1\):

\texttt{CTAN:macros/latex/contrib/oberdiek/luacolor.dtx} The source file.

\(^1\)http://ctan.org/pkg/luacolor
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 (\TeX{}, m\TeX{}, ...) relies on file name databases, you must refresh these. For example, \TeX{} users run texhash or mktexlar.
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:

\texttt{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 \texttt{lxdoc.cfg}. For instance, put this line into this file, if you want to have A4 as
paper format:

\texttt{\PassOptionsToClass{a4paper}{article}}

An example follows how to generate the documentation with pdf\LaTeX:

\texttt{pdflatex luacolor.dtx}
\texttt{makeindex -s gind.ist luacolor.idx}
\texttt{pdflatex luacolor.dtx}
\texttt{makeindex -s gind.ist luacolor.idx}
\texttt{pdflatex luacolor.dtx}

5 Catalogue

The following XML file can be used as source for the \TeX Catalogue. The ele-
ments \texttt{caption} and \texttt{description} are imported from the original XML file from the
Catalogue. The name of the XML file in the Catalogue is \texttt{luacolor.xml}.

\begin{verbatim}
<catalogue>
  <content asymmetric="true" 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>
  </content>
</catalogue>
\end{verbatim}
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 `\pdfform` 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.
Use of \texttt{node.subtype} instead of magic numbers.

More use of \texttt{node.subtype} instead of magic numbers.

\texttt{luatex 85 updates}

\texttt{Documentation updates.}

\section{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.

\begin{verbatim}
\texttt{Symbols} \hfill \texttt{D} \\
\# ........................................ 412 \hfill \texttt{define@color} ...................... 86  \\
\% ........................................ 488 \hfill \texttt{directlua} ...................... 74, 76, 176, 185  \\
\@ ........................................ 413, 486 \hfill \texttt{documentclass} ............... 406, 514, 521  \\
\@end .................................. 527 \hfill \texttt{driver} ......................... 511  \\
\@PackageError ................................ 68, 103, 126 \hfill \texttt{dvidetect()} .................... 249  \\
\@PackageInfoNoLine ......................... 133 \hfill \texttt{E} \\
\@PackageWarning .......................... 189, 194 \hfill \texttt{end} ......................... 506, 517  \\
\@PackageWarningNoLine ..................... 80 \hfill \texttt{endcsname} ....................... 9, 193, 414, 417, 420, 423, 478, 505, 510, 516  \\
\@ehc ................................... 70, 107, 129 \hfill \texttt{endinput} ....................... 51  \\
\@empty ................................ 125 \hfill \texttt{endlinechar} .................... 4, 10, 22  \\
\@firstofone ................................ 421, 424 \hfill \texttt{errmessage} ..................... 467  \\
\@gobble ................................ 418, 426 \hfill \texttt{G} \\
\@undefined ............................... 61, 171 \hfill \texttt{get()} ......................... 272  \\
\\ ...................................... 263, 267, 487 \hfill \texttt{get@type()} ...................... 318  \\
\{ ........................................ 410 \hfill \texttt{getattribute()} .................. 289  \\
\} ........................................ 411 \hfill \texttt{getvalue()} ...................... 275  \\

\texttt{A} \hfill \texttt{D} \\
\texttt{advance} .................. 451, 459, 474 \hfill \texttt{getversion()} .................. 214  \\
\texttt{afterassignment} ............... 200 \hfill \texttt{G} \\
\texttt{allocationnumber} ............... 150 \hfill \texttt{hbox} ......................... 115  \\
\texttt{AtBeginShipout} ................. 166 \hfill \texttt{H} \\
\texttt{AtBeginShipoutBox} ............... 167 \hfill \texttt{I} \\

\texttt{B} \hfill \texttt{L} \\
\texttt{body} ......................... 430, 434 \hfill \texttt{ifcolors@} ....................... 78  \\

\texttt{C} \hfill \texttt{M} \\
\texttt{catcode} .................. 2, 3, 5, 6, 7, 11, 12, 13, 14, 15, 16, 17, 20, 21, 23, 24, 25, 26, 30, 32, 410, 411, 412, 413, 448, 457, 465, 469, 486, 487, 488 \hfill \texttt{ifpdf} ..................... 510  \\
\texttt{count@} .................. 201, 204, 205, 415, 444, 448, 450, 451, 455, 457, 458, 459, 463, 465, 468, 469, 473, 474 \hfill \texttt{iflua} ..................... 60  \\
\texttt{countdef} ..................... 415 \hfill \texttt{infrun} ..................... 73, 90, 175, 184, 450, 458, 465, 473  \\
\texttt{csname} ....................... 9, 193, 414, 417, 420, 423, 478, 505, 516 \hfill \texttt{includeTests} ................ 524  \\
\texttt{current@color} ............. 113, 155 \hfill \texttt{info()} ...................... 241  \\

\texttt{D} \hfill \texttt{L} \\
\texttt{define@color} ...................... 86 \hfill \texttt{LoadCommand} .................. 479, 489  \\
\texttt{directlua} ...................... 74, 76, 176, 185 \hfill \texttt{LogTests} .................... 525  \\
\texttt{documentclass} ............... 406, 514, 521 \hfill \texttt{loop} ..................... 429, 445, 456, 464  \\
\texttt{driver} ......................... 511 \hfill \texttt{M} \\
\texttt{dvidetect()} .................... 249 \hfill \texttt{N} \\
\texttt{end} ......................... 506, 517 \hfill \texttt{O} \\
\texttt{endcsname} ....................... 9, 193, 414, 417, 420, 423, 478, 505, 510, 516 \hfill \texttt{P} \\
\texttt{endinput} ....................... 51 \hfill \texttt{Q} \\
\texttt{endlinechar} .................... 4, 10, 22 \hfill \texttt{R} \\
\texttt{errmessage} ..................... 467 \hfill \texttt{S} \\
\texttt{get()} ......................... 272 \hfill \texttt{symbols} .................... 410  \\
\texttt{get@type()} ...................... 318 \hfill \texttt{T} \\
\texttt{getattribute()} .................. 289 \hfill \texttt{U} \\
\texttt{getvalue()} ...................... 275 \hfill \texttt{V} \\
\texttt{getversion()} .................. 214 \hfill \texttt{W} \\
\texttt{hbox} ......................... 115 \hfill \texttt{X} \\
\texttt{I} 78  \\
\texttt{ifcolors@} ....................... 78 \hfill \texttt{Y} \\
\end{verbatim}

16