

The pdfrender package

Heiko Oberdiek*

2019/12/29 v1.6

Abstract

The PDF format has some graphics parameter like line width or text rendering mode. This package provides an interface for setting these parameters.

Contents

1	Documentation	2
1.1	Usage	2
1.2	Macros	2
1.3	Parameters	2
1.3.1	Details	3
1.4	Color stack	4
2	Implementation	4
2.1	Look for pdfTeX, its mode and features	6
2.2	Enable color support of L ^A T _E X	8
2.3	Hook into \normalcolor	8
2.4	Declare and setup parameters	13
2.5	Fill and stroke color support	15
3	Installation	19
3.1	Download	19
3.2	Bundle installation	19
3.3	Package installation	19
3.4	Refresh file name databases	20
3.5	Some details for the interested	20
4	Acknowledgement	20
5	References	20
6	History	21
	[2010/01/26 v1.0]	21
	[2010/01/27 v1.1]	21
	[2010/01/28 v1.2]	21
	[2016/05/14 v1.3]	21
	[2016/05/17 v1.4]	21
	[2018/11/01 v1.5]	21
	[2019/12/29 v1.6]	21

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

1 Documentation

This package `pdfrender` defines an interface for PDF specific parameters that affects the rendering of graphics or text. The interface and its implementation uses the same technique as package `color` for color settings. Therefore this package is loaded to enable L^AT_EX's color interface.

At different places L^AT_EX uses `\normalcolor` to avoid that header, footer or floats are print in the current color of the main text. `\setgroup@color` is used to start a save box with the color that is set at box saving time. Package `pdfrender` extends these macros to add its own hooks of its parameters. Therefore L^AT_EX3 should generalize L^AT_EX 2_ε's color interface.

1.1 Usage

In L^AT_EX the package is loaded as normal package. Options are not defined for this package.

```
\usepackage{pdfrender}
```

This package can also be used in plain T_EX and even iniT_EX:

```
input pdfrender.sty
```

1.2 Macros

`\pdfrender {⟨key value list⟩}`

The first parameter *⟨key value list⟩* contains a list of parameter settings. The key entry is the parameter name. The macro works like `\color` (without optional argument) for color setting.

`\textpdfrender {⟨key value list⟩} {⟨text⟩}`

In the same way as `\pdfrender` the first argument specifies the parameters that should be set. This parameter setting affects *⟨text⟩* only. Basically it works the same way as `\textcolor` (without optional argument).

1.3 Parameters

The following table shows an overview for the supported parameters and values:

Parameter	Value	Alias
TextRenderingMode	0	Fill
	1	Stroke
	2	FillStroke
	3	Invisible
	4	FillClip
	5	StrokeClip
	6	FillStrokeClip
	7	Clip
LineWidth	<i>positive number, unit is bp</i>	<i>T_{EX} dimen</i>
LineCapStyle	0	Butt
	1	Round
	2	ProjectingSquare
LineJoinStyle	0	Miter
	1	Round
	2	Bevel
MiterLimit	<i>positive number</i>	
Flatness	<i>number between 0 and 100</i>	
LineDashPattern	<i>numbers in square brackets, followed by number, units are bp</i>	
RenderingIntent	AbsoluteColorimetric	
	RelativeColorimetric	
	Saturation	
	Perceptual	
FillColor		<i>color specification</i>
StrokeColor		<i>color specification</i>

1.3.1 Details

The description and specification of these parameters are available in the PDF specification [1]. Therefore they are not repeated here.

Value: The values in the second column lists or describe the values that are specified by the PDF specification.

Alias: Instead of magic numbers the package also defines some aliases that can be given as value. Example: `LineCapStyle=Round` has the same effect as `LineCapStyle=1`.

Number: The term *number* means an integer or real number. The real number is given as plain decimal number without exponent. The decimal separator is a period. At least one digit must be present.

LineWidth: As alias a T_{EX} dimen specification can be given. This includes explicit specifications with number and unit, e.g. `LineWidth=0.5pt`. Also L^AT_{EX} length registers may be used. If ε -T_{EX}'s `\dimexpr` is available, then it is automatically added. However package `calc` is not supported.

FillColor, StrokeColor: Package `color` or `xcolor` must be loaded before these options can be used (since version 1.2). L^AT_EX's color support sets both colors at the same time to the same value. However parameter `TextRenderingMode` offers the value `FillStroke` that makes only sense, if the two color types can be set separately. If one of the options `FillColor` or `StrokeColor` is specified, then also the color is set. For compatibility with the L^AT_EX color packages (`color` or `xcolor`), always both colors must be set. Thus if one of them is not specified, it is taken from the current color.

Both options `FillColor` and `StrokeColor` expect a L^AT_EX color specification as value. Also the optional color model argument is supported. Example:

```
FillColor=yellow,
StrokeColor=[cmyk]{1,.5,0,0}
```

1.4 Color stack

If the pdfT_EX version provides color stacks, then each parameter is assigned a page based color stack. The assignment of a stack takes place, when its parameter is set the first time. This avoids the use of color stacks that are not needed.

2 Implementation

```
1 \*package)
```

Reload check, especially if the package is not used with L^AT_EX.

```
2 \begingroup\catcode61\catcode48\catcode32=10\relax%
3 \catcode13=5 % ^~M
4 \endlinechar=13 %
5 \catcode35=6 % #
6 \catcode39=12 % '
7 \catcode44=12 % ,
8 \catcode45=12 % -
9 \catcode46=12 % .
10 \catcode58=12 % :
11 \catcode64=11 % @
12 \catcode123=1 % {
13 \catcode125=2 % }
14 \expandafter\let\expandafter\x\csname ver@pdfrender.sty\endcsname
15 \ifx\x\relax % plain-TeX, first loading
16 \else
17 \def\empty{}%
18 \ifx\x\empty % LaTeX, first loading,
19 % variable is initialized, but \ProvidesPackage not yet seen
20 \else
21 \expandafter\ifx\csname PackageInfo\endcsname\relax
22 \def\x#1#2{%
23 \immediate\write-1{Package #1 Info: #2.}%
24 }%
25 \else
26 \def\x#1#2{\PackageInfo{#1}{#2, stopped}}%
27 \fi
28 \x{pdfrender}{The package is already loaded}%
29 \aftergroup\endinput
30 \fi
31 \fi
```

```

32 \endgroup%
Package identification:
33 \begingroup\catcode61\catcode48\catcode32=10\relax%
34 \catcode13=5 % ^~M
35 \endlinechar=13 %
36 \catcode35=6 % #
37 \catcode39=12 % '
38 \catcode40=12 % (
39 \catcode41=12 % )
40 \catcode44=12 % ,
41 \catcode45=12 % -
42 \catcode46=12 % .
43 \catcode47=12 % /
44 \catcode58=12 % :
45 \catcode64=11 % @
46 \catcode91=12 % [
47 \catcode93=12 % ]
48 \catcode123=1 % {
49 \catcode125=2 % }
50 \expandafter\ifx\csname ProvidesPackage\endcsname\relax
51   \def\x#1#2#3[#4]{\endgroup
52     \immediate\write-1{Package: #3 #4}%
53     \xdef#1{#4}%
54   }%
55 \else
56   \def\x#1#2[#3]{\endgroup
57     #2[#{#3}]%
58     \ifx#1\undefined
59       \xdef#1{#3}%
60     \fi
61     \ifx#1\relax
62       \xdef#1{#3}%
63     \fi
64   }%
65 \fi
66 \expandafter\x\csname ver@pdfrender.sty\endcsname
67 \ProvidesPackage{pdfrender}%
68 [2019/12/29 v1.6 Access to some PDF graphics parameters (HO)]%
69 \begingroup\catcode61\catcode48\catcode32=10\relax%
70 \catcode13=5 % ^~M
71 \endlinechar=13 %
72 \catcode123=1 % {
73 \catcode125=2 % }
74 \catcode64=11 % @
75 \def\x{\endgroup
76   \expandafter\edef\csname PdfRender@AtEnd\endcsname{%
77     \endlinechar=\the\endlinechar\relax
78     \catcode13=\the\catcode13\relax
79     \catcode32=\the\catcode32\relax
80     \catcode35=\the\catcode35\relax
81     \catcode61=\the\catcode61\relax
82     \catcode64=\the\catcode64\relax
83     \catcode123=\the\catcode123\relax
84     \catcode125=\the\catcode125\relax
85   }%
86 }%
87 \x\catcode61\catcode48\catcode32=10\relax%
88 \catcode13=5 % ^~M

```

```

89 \endlinechar=13 %
90 \catcode35=6 % #
91 \catcode64=11 % @
92 \catcode123=1 % {
93 \catcode125=2 % }
94 \def\TMP@EnsureCode#1#2{%
95   \edef\PdfRender@AtEnd{%
96     \PdfRender@AtEnd
97     \catcode#1=\the\catcode#1\relax
98   }%
99   \catcode#1=#2\relax
100 }
101 \TMP@EnsureCode{10}{12}% ^^J
102 \TMP@EnsureCode{36}{3}% $
103 \TMP@EnsureCode{39}{12}% '
104 \TMP@EnsureCode{40}{12}% (
105 \TMP@EnsureCode{41}{12}% )
106 \TMP@EnsureCode{42}{12}% *
107 \TMP@EnsureCode{43}{12}% +
108 \TMP@EnsureCode{44}{12}% ,
109 \TMP@EnsureCode{45}{12}% -
110 \TMP@EnsureCode{46}{12}% .
111 \TMP@EnsureCode{47}{12}% /
112 \TMP@EnsureCode{58}{12}% :
113 \TMP@EnsureCode{59}{12}% ;
114 \TMP@EnsureCode{60}{12}% <
115 \TMP@EnsureCode{62}{12}% >
116 \TMP@EnsureCode{63}{12}% ?
117 \TMP@EnsureCode{91}{12}% [
118 \TMP@EnsureCode{93}{12}% ]
119 \TMP@EnsureCode{94}{7}% ^ (superscript)
120 \TMP@EnsureCode{96}{12}% '
121 \TMP@EnsureCode{124}{12}% |

122 \def\PdfRender@AtEndHook{}
123 \expandafter\def\expandafter\PdfRender@AtEnd\expandafter{%
124   \expandafter\PdfRender@AtEndHook
125   \PdfRender@AtEnd
126   \endinput
127 }

```

2.1 Look for pdfTeX, its mode and features

\PdfRender@newif

```

128 \def\PdfRender@newif#1{%
129   \expandafter\edef\csname PdfRender@#1true\endcsname{%
130     \let
131     \expandafter\noexpand\csname ifPdfRender@#1\endcsname
132     \noexpand\iftrue
133   }%
134   \expandafter\edef\csname PdfRender@#1false\endcsname{%
135     \let
136     \expandafter\noexpand\csname ifPdfRender@#1\endcsname
137     \noexpand\iffalse
138   }%
139   \csname PdfRender@#1false\endcsname
140 }

```

\ifPdfRender@Stack

```

141 \PdfRender@newif{Stack}

\ifPdfRender@Match
142 \PdfRender@newif{Match}

\PdfRender@RequirePackage
143 \begingroup\expandafter\expandafter\expandafter\endgroup
144 \expandafter\ifx\csname RequirePackage\endcsname\relax
145 \def\PdfRender@RequirePackage#1[#2]{%
146 \expandafter\def\expandafter\PdfRender@AtEndHook\expandafter{%
147 \PdfRender@AtEndHook
148 \ltx@ifpackagelater{#1}{#2}{}{%
149 \@PackageWarningNoLine{pdfrender}{%
150 You have requested version\MessageBreak
151 '#2' of package '#1',\MessageBreak
152 but only version\MessageBreak
153 '\csname ver@#1.\ltx@pkgextension\endcsname'\MessageBreak
154 is available%
155 }%
156 }%
157 }%
158 \input #1.sty\relax
159 }%
160 \else
161 \let\PdfRender@RequirePackage\RequirePackage
162 \fi

```

Luatex compatibility

```

163 \ifx\pdfextension\@undefined\else
164 \def\pdfcolorstackinit {\pdffeedback colorstackinit}
165 \protected\def\pdfcolorstack {\pdfextension colorstack}
166 \protected\def\pdfliteral {\pdfextension literal}
167 \fi

168 \PdfRender@RequirePackage{iftex}[2019/11/07]
169 \PdfRender@RequirePackage{infwarerr}[2007/09/09]
170 \PdfRender@RequirePackage{ltxcmds}[2010/01/28]

171 \ifpdf
172 \ltx@ifundefined{pdfcolorstackinit}{%
173 \@PackageWarning{pdfrender}{%
174 Missing \string\pdfcolorstackinit
175 }%
176 }{%
177 \PdfRender@Stacktrue
178 }%
179 \ltx@ifundefined{pdfmatch}{%
180 \@PackageInfoNoLine{pdfrender}{%
181 \string\pdfmatch\ltx@space not found. %
182 Therefore the values\MessageBreak
183 of some parameters are not validated%
184 }%
185 }{%
186 \PdfRender@Matchtrue
187 }%
188 \else
189 \@PackageWarning{pdfrender}{%
190 Missing pdfTeX in PDF mode%
191 }%

```

```

192 \ltx@ifundefined{newcommand}{%

\pdfrender
193 \def\pdfrender#1{%

\textpdfrender
194 \long\def\textpdfrender#1#2{#2}%

195 }{%

\pdfrender
196 \newcommand*{\pdfrender}[1]{}%

\textpdfrender
197 \newcommand{\textpdfrender}[2]{#2}%

198 }%
199 \expandafter\PdfRender@AtEnd
200 \fi%

```

2.2 Enable color support of L^AT_EX

```

201 \ltx@ifpackageloaded{color}{}%
202 \def\color@setgroup{\begingroup\set@color}%
203 \let\color@begingroup\begingroup
204 \def\color@endgroup{\endgraf\endgroup}%
205 \def\color@hbox{\hbox\bgroup\color@begingroup}%
206 \def\color@vbox{\vbox\bgroup\color@begingroup}%
207 \def\color@endbox{\color@endgroup\egroup}%
208 \ltx@ifundefined{bgroup}{%
209 \let\bgroup=\let\egroup=%
210 }{}%
211 \ltx@ifundefined{endgraf}{%
212 \let\endgraf=\par
213 }{}%
214 }

```

2.3 Hook into \normalcolor

The problem is that packages `color` and `xcolor` each overwrite `\normalcolor`. For example, after the package loading order `color`, `pdfrender` and `xcolor` the patched version of `\normalcolor` is overwritten by package `xcolor`. Also using `\AtBeginDocument` for patching is not enough. If package `hyperref` is loaded later, it might load package `color` using `\AtBeginDocument`.

```

\PdfRender@NormalColorHook
215 \def\PdfRender@NormalColorHook{}

\PdfRender@ColorSetGroupHook
216 \def\PdfRender@ColorSetGroupHook{}

\PdfRender@TestBox
217 \def\PdfRender@TestBox#1{%
218 \setbox0=\color@hbox#1\color@endbox
219 }

```


\PdfRender@PatchNormalColor

```
220 \def\PdfRender@PatchNormalColor{%
221   \ltx@ifundefined{normalcolor}{%
222     \gdef\normalcolor{\PdfRender@NormalColorHook}%
223   }{%
224     \begingroup
225       \def\PdfRender@NormalColorHook{\let\PdfRender@temp=Y}%
226       \PdfRender@TestBox{%
227         \let\set@color\relax
228         \normalcolor
229         \ifx\PdfRender@temp Y%
230           \else
231             \ltx@GlobalAppendToMacro\normalcolor{%
232               \PdfRender@NormalColorHook
233             }%
234           \fi
235         }%
236       \endgroup
237     }%
238     \ifx\nodocument\relax
239       \global\let\PdfRender@PatchNormalColor\relax
240     \fi
241   }%
```

\PdfRender@PatchColorSetGroup

```
242 \def\PdfRender@PatchColorSetGroup{%
243   \begingroup
244     \def\PdfRender@ColorSetGroupHook{\let\PdfRender@temp=Y}%
245     \PdfRender@TestBox{%
246       \let\set@color\relax
247       \color@setgroup\color@endgroup
248       \ifx\PdfRender@temp Y%
249         \else
250           \ltx@GlobalAppendToMacro\color@setgroup{%
251             \PdfRender@ColorSetGroupHook
252           }%
253         \fi
254       }%
255     \endgroup
256     \ifx\nodocument\relax
257       \global\let\PdfRender@PatchColorSetGroup\relax
258     \fi
259   }%
```

\PdfRender@PatchColor

```
260 \def\PdfRender@PatchColor{%
261   \PdfRender@PatchNormalColor
262   \PdfRender@PatchColorSetGroup
263 }

264 \PdfRender@PatchColor
265 \ltx@ifundefined{AtBeginDocument}{}{%
266   \AtBeginDocument{\PdfRender@PatchColor}%
267 }

\AfterPackage is provided by package scrfile.
268 \ltx@ifundefined{AfterPackage}{%
269 }%
```

```

270 \AfterPackage{color}{\PdfRender@PatchColor}%
271 \AfterPackage{xcolor}{\PdfRender@PatchColor}%
272 \AfterPackage{etoolbox}{%
273 \AfterEndPreamble{\PdfRender@PatchColor}%
274 }%
275 }%
\AfterEndPreamble is provided by package etoolbox.
276 \ltx@ifundefined{AfterEndPreamble}{%
277 }{%
278 \AfterEndPreamble{\PdfRender@PatchColor}%
279 }%

280 \PdfRender@RequirePackage{kvsetkeys}[2010/01/28]

\PdfRender@texorpdfstring

281 \def\PdfRender@texorpdfstring{%
282 \ltx@ifundefined{texorpdfstring}\ltx@firstoftwo\texorpdfstring
283 }

\pdfrender

284 \ltx@ifundefined{DeclareRobustCommand}%
285 \ltx@firstoftwo\ltx@secondoftwo
286 {%
287 \def\pdfrender#1%
288 }{%
289 \newcommand{\pdfrender}{}%
290 \DeclareRobustCommand*\pdfrender[1]%
291 }%
292 {%
293 \PdfRender@texorpdfstring{%
294 \PdfRender@PatchNormalColor
295 \global\let\PdfRender@FillColor\ltx@empty
296 \global\let\PdfRender@StrokeColor\ltx@empty
297 \kvsetkeys{PDFRENDER}{#1}%
298 \PdfRender@SetColor
299 }{%
300 }

\textpdfrender

301 \ltx@ifundefined{DeclareRobustCommand}%
302 \ltx@firstoftwo\ltx@secondoftwo
303 {%
304 \long\def\textpdfrender#1#2%
305 }{%
306 \newcommand{\textpdfrender}{}%
307 \DeclareRobustCommand{\textpdfrender}[2]%
308 }%
309 {%
310 \PdfRender@texorpdfstring{%
311 \begingroup
312 \pdfrender{#1}%
313 #2%
314 \endgroup
315 }{#2}%
316 }

\ifPdfRender@Values

317 \PdfRender@newif{Values}

```

\PdfRender@NewClassValues

```
318 \def\PdfRender@NewClassValues#1#2#3#4{%
319   \PdfRender@Valuestrue
320   \PdfRender@NewClass{#1}{#2}{#3}{#4}{}%
321 }
```

\PdfRender@NewClass

```
322 \def\PdfRender@NewClass#1#2#3#4#5{%
323   \PdfRender@newif{Active#1}%
324   \expandafter\def\csname PdfRender@Default#1\endcsname{#2}%
325   \expandafter\let\csname PdfRender@Current#1\expandafter\endcsname
326     \csname PdfRender@Default#1\endcsname
327   \ifPdfRender@Stack
328     \expandafter\edef\csname PdfRender@Init#1\endcsname{%
329       \global\chardef
330       \expandafter\noexpand\csname PdfRender@Stack#1\endcsname=%
331       \noexpand\pdfcolorstackinit page direct{%
332         \noexpand#3%
333         \expandafter\noexpand\csname PdfRender@Default#1\endcsname
334       }\relax
335       \noexpand\@PackageInfo{pdfrender}{%
336         New color stack ‘#1’ = \noexpand\number
337         \expandafter\noexpand\csname PdfRender@Stack#1\endcsname
338       }%
339       \gdef\expandafter\noexpand\csname PdfRender@Init#1\endcsname{%
340       }%
341       \expandafter\edef\csname PdfRender@Set#1\endcsname{%
342         \expandafter\noexpand\csname PdfRender@Init#1\endcsname
343         \noexpand\pdfcolorstack
344         \expandafter\noexpand\csname PdfRender@Stack#1\endcsname
345         push{%
346           #3{\expandafter\noexpand\csname PdfRender@Current#1\endcsname}%
347         }%
348         \noexpand\aftergroup
349         \expandafter\noexpand\csname PdfRender@Reset#1\endcsname
350       }%
351       \expandafter\edef\csname PdfRender@Reset#1\endcsname{%
352         \expandafter\noexpand\csname PdfRender@Init#1\endcsname
353         \noexpand\pdfcolorstack
354         \expandafter\noexpand\csname PdfRender@Stack#1\endcsname
355         pop\relax
356       }%
357     \else
358       \expandafter\edef\csname PdfRender@Set#1\endcsname{%
359         \noexpand\pdfliteral direct{%
360           #3{\expandafter\noexpand\csname PdfRender@Current#1\endcsname}%
361         }%
362         \noexpand\aftergroup
363         \expandafter\noexpand\csname PdfRender@Reset#1\endcsname
364       }%
365       \expandafter\edef\csname PdfRender@Reset#1\endcsname{%
366         \noexpand\pdfliteral direct{%
367           #3{\expandafter\noexpand\csname PdfRender@Current#1\endcsname}%
368         }%
369       }%
370     \fi
371   \expandafter\edef\csname PdfRender@Normal#1\endcsname{%
372     \let
```

```

373 \expandafter\noexpand\csname PdfRender@Current#1\endcsname
374 \expandafter\noexpand\csname PdfRender@Default#1\endcsname
375 \noexpand\PdfRender@Set{#1}%
376 }%
377 \expandafter\ltx@GlobalAppendToMacro\expandafter\PdfRender@NormalColorHook
378 \expandafter{%
379 \csname PdfRender@Normal#1\endcsname
380 }%
381 \ltx@GlobalAppendToMacro\PdfRender@ColorSetGroupHook{%
382 \PdfRender@Set{#1}%
383 }%
384 \ifPdfRender@Values
385 \kv@parse@normalized{#4}{%
386 \expandafter\let\csname PdfRender@#1\kv@key\endcsname\kv@key
387 \ifx\kv@value\relax
388 \else
389 \expandafter\let\csname PdfRender@#1\kv@value\endcsname\kv@key
390 \fi
391 \ltx@gobbletwo
392 }%
393 \PdfRender@define@key{PDFRENDER}{#1}{%
394 \global\csname PdfRender@Active#1true\endcsname
395 \def\PdfRender@Current{##1}%
396 \PdfRender@SetValidateValues{#1}%
397 }%
398 \PdfRender@Valuesfalse
399 \else
400 \PdfRender@define@key{PDFRENDER}{#1}{%
401 \global\csname PdfRender@Active#1true\endcsname
402 \expandafter\def\csname PdfRender@Current#1\endcsname{##1}%
403 \ltx@ifundefined{PdfRender@PostProcess#1}{%
404 }{%
405 \csname PdfRender@PostProcess#1\endcsname
406 }%
407 \PdfRender@SetValidate{#1}{#4}{#5}%
408 }%
409 \fi
410 }%

```

\PdfRender@define@key

```

411 \ltx@ifundefined{define@key}{%
412 \def\PdfRender@define@key#1#2{%
413 \expandafter\def\csname KV@#1@#2\endcsname##1%
414 }%
415 }{%
416 \let\PdfRender@define@key\define@key
417 }

```

\PdfRender@Set

```

418 \def\PdfRender@Set#1{%
419 \csname ifPdfRender@Active#1\endcsname
420 \csname PdfRender@Set#1\expandafter\endcsname
421 \fi
422 }

```

\PdfRender@Reset

```

423 \def\PdfRender@Reset#1{%
424 \csname ifPdfRender@Active#1\endcsname

```

```

425 \csname PdfRender@Reset#1\expandafter\endcsname
426 \fi
427 }

```

\PdfRender@ErrorInvalidValue

```

428 \def\PdfRender@ErrorInvalidValue#1{%
429 \PackageError{pdfrender}{%
430 Ignoring parameter setting for ‘#1’\MessageBreak
431 because of invalid value %
432 ‘\csname PdfRender@Current#1\endcsname’%
433 }\@ehc
434 \expandafter\let\csname PdfRender@Current#1\endcsname\ltx@empty
435 }%

```

\PdfRender@SetValidate

```

436 \ifPdfRender@Match
437 \def\PdfRender@SetValidate#1#2#3{%
438 \ifnum\pdfmatch{~(#2)$}{\csname PdfRender@Current#1\endcsname}=1 %
439 \csname PdfRender@Set#1\expandafter\endcsname
440 \else
441 \PdfRender@ErrorInvalidValue{#1}%
442 \fi
443 }%
444 \else
445 \def\PdfRender@SetValidate#1#2#3{%
446 \expandafter\let\expandafter\PdfRender@Current
447 \csname PdfRender@Current#1\endcsname
448 #3%
449 \ifx\PdfRender@Current\@empty
450 \PdfRender@ErrorInvalidValue{#1}%
451 \else
452 \csname PdfRender@Set#1\expandafter\endcsname
453 \fi
454 }%
455 \fi

```

\PdfRender@SetValidateValues

```

456 \def\PdfRender@SetValidateValues#1{%
457 \ltx@ifundefined{PdfRender@#1@\PdfRender@Current}{%
458 \expandafter\let\csname PdfRender@Current#1\endcsname
459 \PdfRender@Current
460 \PdfRender@ErrorInvalidValue{#1}%
461 }{%
462 \expandafter\edef\csname PdfRender@Current#1\endcsname{%
463 \csname PdfRender@#1@\PdfRender@Current\endcsname
464 }%
465 \csname PdfRender@Set#1\endcsname
466 }%
467 }

```

\PdfRender@OpValue

```

468 \def\PdfRender@OpValue#1#2{#2\ltx@space#1}%

```

\PdfRender@OpName

```

469 \def\PdfRender@OpName#1#2{/#2\ltx@space#1}%

```

2.4 Declare and setup parameters

```

470 \PdfRender@NewClassValues{TextRenderingMode}%
471         {0}%
472         {\PdfRender@OpValue{Tr}}{%
473     0=Fill,%
474     1=Stroke,%
475     2=FillStroke,%
476     3=Invisible,%
477     4=FillClip,%
478     5=StrokeClip,%
479     6=FillStrokeClip,%
480     7=Clip,%
481 }%
482 \PdfRender@NewClass{LineWidth}{1}{\PdfRender@OpValue{w}}{%
483     [0-9]+\string\.[0-9]*|\string\.[0-9]+%
484 }{%
485 \ltx@ifundefined{dimexpr}{%
486     \def\PdfRender@dimexpr{%
487 }{%
488     \let\PdfRender@dimexpr\dimexpr
489 }
490 \def\PdfRender@PostProcessLineWidth{%
491     \begingroup
492     \afterassignment\PdfRender@@PostProcessLineWidth
493     \dimen0=\PdfRender@dimexpr\PdfRender@CurrentLineWidth bp %
494     \PdfRender@let\PdfRender@relax\PdfRender@relax
495 }
496 \let\PdfRender@let\let
497 \let\PdfRender@relax\relax
498 \def\PdfRender@@PostProcessLineWidth#1\PdfRender@let{%
499     \ifx\#1\%
500         \endgroup
501     \else
502         \dimen0=.996264\dimen0 % 72/72.27
503         \edef\x{\endgroup
504             \def\noexpand\PdfRender@CurrentLineWidth{%
505                 \strip@pt\dimen0%
506             }%
507         }%
508         \expandafter\x
509     \fi
510 }
511 \PdfRender@NewClassValues{LineCapStyle}{0}{\PdfRender@OpValue{J}}{%
512     0=Butt,%
513     1=Round,%
514     2=ProjectingSquare,%
515 }%
516 \PdfRender@NewClassValues{LineJoinStyle}{0}{\PdfRender@OpValue{j}}{%
517     0=Miter,%
518     1=Round,%
519     2=Bevel,%
520 }%
521 \PdfRender@NewClass{MiterLimit}{10}{\PdfRender@OpValue{M}}{%
522     [0-9]*[1-9][0-9]*\string\.[0-9]*|
523     [0-9]*\string\.[0-9]*[1-9][0-9]*%
524 }{%
525 \PdfRender@NewClass{Flatness}{0}{\PdfRender@OpValue{i}}{%
526     100(\string\0*)?|[0-9][0-9](\string\.[0-9]*)?|\string\.[0-9]+%

```

```

527 }{}%
528 \PdfRender@NewClass{LineDashPattern}{[]0}{\PdfRender@OpValue{d}}{%
529   \string\[%
530   ( ?([0-9]+\string\.[0-9]*|\string\.[0-9]+) ?)*%
531   \string\] ?%
532   ([0-9]+\string\.[0-9]*|\string\.[0-9]+)%
533 }{}%
534 \PdfRender@NewClassValues{RenderingIntent}%
535   {RelativeColorimetric}%
536   {\PdfRender@OpName{ri}}{%
537   AbsoluteColorimetric,%
538   RelativeColorimetric,%
539   Saturation,%
540   Perceptual,%
541 }%

```

2.5 Fill and stroke color support

```

542 \PdfRender@define@key{PDFRENDER}{FillColor}{%
543   \begin{group}
544     \def\PdfRender@Color{#1}%
545     \ifx\PdfRender@Color\ltx@empty
546       \global\let\PdfRender@FillColor\ltx@empty
547     \else
548       \PdfRender@ColorAvailable{%
549         \PdfRender@TestBox{%
550           \expandafter\PdfRender@TryColor\PdfRender@Color\ltx@empty
551           \PdfRender@GetFillColor
552           \ifx\PdfRender@FillColor\ltx@empty
553             \@PackageWarning{pdfrender}{%
554               Cannot extract fill color\MessageBreak
555               from value ‘#1’%
556             }%
557           \fi
558         }%
559       }%
560     \fi
561   \end{group}
562 }
563 \PdfRender@define@key{PDFRENDER}{StrokeColor}{%
564   \begin{group}
565     \def\PdfRender@Color{#1}%
566     \ifx\PdfRender@Color\ltx@empty
567       \global\let\PdfRender@StrokeColor\ltx@empty
568     \else
569       \PdfRender@ColorAvailable{%
570         \PdfRender@TestBox{%
571           \expandafter\PdfRender@TryColor\PdfRender@Color\ltx@empty
572           \PdfRender@GetStrokeColor
573           \ifx\PdfRender@StrokeColor\ltx@empty
574             \@PackageWarning{pdfrender}{%
575               Cannot extract stroke color\MessageBreak
576               from value ‘#1’%
577             }%
578           \fi
579         }%
580       }%
581     \fi
582   \end{group}

```

```

583 }

\PdfRender@ColorAvailable

584 \def\PdfRender@ColorAvailable{%
585   \@ifundefined{set@color}{%
586     \@PackageError{pdfrender}{%
587       Ignoring color options, because neither\MessageBreak
588       package 'color' nor package 'xcolor' is loaded%
589     }\@ehc
590     \global\let\PdfRender@ColorAvailable\ltx@gobble
591   }{%
592     \global\let\PdfRender@ColorAvailable\ltx@firstofone
593   }%
594   \PdfRender@ColorAvailable
595 }

\PdfRender@TryColor

596 \def\PdfRender@TryColor{%
597   \@ifnextchar[\color\PdfRender@@TryColor
598 }

\PdfRender@@TryColor

599 \def\PdfRender@@TryColor#1\ltx@empty{%
600   \expandafter\color\expandafter{\PdfRender@Color}%
601 }

\PdfRender@SetColor

602 \def\PdfRender@SetColor{%
603   \chardef\PdfRender@NeedsCurrentColor=0 %
604   \ifx\PdfRender@FillColor\ltx@empty
605     \ifx\PdfRender@StrokeColor\ltx@empty
606       \else
607         \edef\PdfRender@CurrentColor{%
608           \noexpand\PdfRender@FillColor\ltx@space\PdfRender@StrokeColor
609         }%
610         \chardef\PdfRender@NeedsCurrentColor=1 %
611       \fi
612     \else
613       \ifx\PdfRender@StrokeColor\ltx@empty
614         \edef\PdfRender@CurrentColor{%
615           \PdfRender@FillColor\ltx@space\noexpand\PdfRender@StrokeColor
616         }%
617         \chardef\PdfRender@NeedsCurrentColor=2 %
618       \else
619         \edef\current@color{%
620           \PdfRender@FillColor\ltx@space\PdfRender@StrokeColor
621         }%
622         \set@color
623       \fi
624     \fi
625     \ifnum\PdfRender@NeedsCurrentColor=1 %
626       \PdfRender@GetFillColor
627       \ifx\PdfRender@FillColor\ltx@empty
628         \@PackageWarning{pdfrender}{%
629           Cannot extract current fill color%
630         }%
631       \else
632         \edef\current@color{\PdfRender@CurrentColor}%

```



```

633     \set@color
634   \fi
635 \else
636   \ifnum\PdfRender@NeedsCurrentColor=2 %
637     \PdfRender@GetStrokeColor
638     \ifx\PdfRender@StrokeColor\ltx@empty
639       \@PackageWarning{pdfrender}{%
640         Cannot extract current stroke color%
641       }%
642     \else
643       \edef\current@color{\PdfRender@CurrentColor}%
644       \set@color
645     \fi
646   \fi
647 \fi
648 }

```

\PdfRender@PatternFillColor

```

649 \edef\PdfRender@PatternFillColor{ % space
650   (%)
651   [0-9\string\.] + g|%
652   [0-9\string\.] + [0-9\string\.] + [0-9\string\.] + rg|%
653   [0-9\string\.] + [0-9\string\.] + %
654   [0-9\string\.] + [0-9\string\.] + k%
655   ) % space
656   (.)$%
657 }

```

\PdfRender@PatternStrokeColor

```

658 \edef\PdfRender@PatternStrokeColor{ % space
659   (%)
660   [0-9\string\.] + G|%
661   [0-9\string\.] + [0-9\string\.] + [0-9\string\.] + RG|%
662   [0-9\string\.] + [0-9\string\.] + %
663   [0-9\string\.] + [0-9\string\.] + K%
664   ) % space
665   (.)$%
666 }

```

\PdfRender@MatchPattern

```

667 \def\PdfRender@MatchPattern#1{%
668   \ifnum\pdfmatch{\PdfRender@Pattern}{\PdfRender@String}=1 %
669     \xdef#1{%
670       \expandafter\strip@prefix\pdflastmatch 1%
671     }%
672     \edef\PdfRender@String{%
673       \expandafter\strip@prefix\pdflastmatch 2%
674     }%
675     \ifx\PdfRender@String\ltx@empty
676     \else
677       \expandafter\expandafter\expandafter\PdfRender@MatchPattern
678       \expandafter\expandafter\expandafter#1%
679     \fi
680   \fi
681 }

```

\PdfRender@GetFillColor

```

682 \def\PdfRender@GetFillColor{%

```

```

683 \global\let\PdfRender@FillColor\ltx@empty
684 \begingroup
685 \ifPdfRender@Match
686 \let\PdfRender@Pattern\PdfRender@PatternFillColor
687 \edef\PdfRender@String{\ltx@space\current@color\ltx@space}%
688 \PdfRender@MatchPattern\PdfRender@FillColor
689 \else
690 \edef\current@color{\current@color\ltx@space}%
691 \let\PdfRender@OP\relax
692 \PdfRender@FindOp{g}0%
693 \PdfRender@FindOp{G}1%
694 \PdfRender@FindOp{rg}0%
695 \PdfRender@FindOp{RG}1%
696 \PdfRender@FindOp{k}0%
697 \PdfRender@FindOp{K}1%
698 \PdfRender@FilterOp 0\PdfRender@FillColor
699 \fi
700 \endgroup
701 }

```

\PdfRender@GetStrokeColor

```

702 \def\PdfRender@GetStrokeColor{%
703 \global\let\PdfRender@StrokeColor\ltx@empty
704 \begingroup
705 \ifPdfRender@Match
706 \let\PdfRender@Pattern\PdfRender@PatternStrokeColor
707 \edef\PdfRender@String{\ltx@space\current@color\ltx@space}%
708 \PdfRender@MatchPattern\PdfRender@StrokeColor
709 \else
710 \edef\current@color{\current@color\ltx@space}%
711 \let\PdfRender@OP\relax
712 \PdfRender@FindOp{g}0%
713 \PdfRender@FindOp{G}1%
714 \PdfRender@FindOp{rg}0%
715 \PdfRender@FindOp{RG}1%
716 \PdfRender@FindOp{k}0%
717 \PdfRender@FindOp{K}1%
718 \PdfRender@FilterOp 1\PdfRender@StrokeColor
719 \fi
720 \endgroup
721 }

722 \ifPdfRender@Match
723 \expandafter\PdfRender@AtEnd
724 \fi%

```

\PdfRender@FindOp

```

725 \def\PdfRender@FindOp#1#2{%
726 \def\PdfRender@temp##1 #1 ##2\@nil{%
727 ##1%
728 \ifx\##2\%
729 \expandafter\@gobble
730 \else
731 \PdfRender@OP{#1}#2%
732 \expandafter\@firstofone
733 \fi
734 {%
735 \PdfRender@temp##2\@nil

```

```

736     }%
737 }%
738 \edef\current@color{%
739   \@firstofone{\expandafter\PdfRender@temp\current@color} #1 \@nil
740 }%
741 }

\PdfRender@FilterOp

742 \def\PdfRender@FilterOp#1#2{%
743   \expandafter\PdfRender@@FilterOp\expandafter#1\expandafter#2%
744   \current@color\PdfRender@OP{ }-%
745 }

\PdfRender@@FilterOp

746 \def\PdfRender@@FilterOp#1#2#3\PdfRender@OP#4#5{%
747   \ifx\#4#5\%
748   \else
749     \ifnum#1=#5 %
750       \xdef#2{#3 #4}%
751     \fi
752     \expandafter\PdfRender@@FilterOp\expandafter#1\expandafter#2%
753   \fi
754 }

755 \PdfRender@AtEnd%
756 \endpackage

```

3 Installation

3.1 Download

Package. This package is available on CTAN¹:

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

[CTAN:macros/latex/contrib/oberdiek/pdfrender.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:pkg/tds](#)). Directories with `texmf` in their name are usually organized this way.

3.2 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
```

¹[CTAN:pkg/pdfrender](#)

3.3 Package installation

Unpacking. The `.dtx` file is a self-extracting `docstrip` archive. The files are extracted by running the `.dtx` through plain `TEX`:

```
tex pdfrender.dtx
```

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

```
pdfrender.sty → tex/generic/oberdiek/pdfrender.sty
pdfrender.pdf → doc/latex/oberdiek/pdfrender.pdf
pdfrender.dtx → source/latex/oberdiek/pdfrender.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`.

3.4 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 `mktextlsr`.

3.5 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 `LATEX` for `docstrip` (really, `docstrip` does not need `LATEX`), then inform the autodetect routine about your intention:

```
latex \let\install=y\input{pdfrender.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 pdfrender.dtx
makeindex -s gind.ist pdfrender.idx
pdflatex pdfrender.dtx
makeindex -s gind.ist pdfrender.idx
pdflatex pdfrender.dtx
```

4 Acknowledgement

Friedrich Vosberg asked in the newsgroup `de.comp.text.tex` for the font outline feature [2].

Gaius Pupus proposed the basic method using `\pdfliteral` in this thread [3].

Rolf Niepraschk added color support [4].

5 References

- [1] Adobe Systems Incorporated. *PDF Reference – Adobe Portable Document format – Version 1.7*. 6th ed. 2006. URL: http://www.adobe.com/devnet/acrobat/pdfs/pdf_reference_1-7.pdf.
- [2] Friedrich Vosberg, *Text in Buchstabenumrissen*, de.comp.text.tex, 2010-01-22. URL: <https://groups.google.com/group/de.comp.text.tex/msg/f442310ac8b2d506>.
- [3] Gaius Pupus, *Re: Text in Buchstabenumrissen*, de.comp.text.tex, 2010-01-23. URL: <https://groups.google.com/group/de.comp.text.tex/msg/95d890d77ac47eb1>.
- [4] Rolf Niepraschk, *Re: Text in Buchstabenumrissen*, de.comp.text.tex, 2010-01-24. URL: <https://groups.google.com/group/de.comp.text.tex/msg/4eb61a5879db54db>.

6 History

[2010/01/26 v1.0]

- The first version.

[2010/01/27 v1.1]

- Macros `\pdfrender` and `\textpdfrender` are made robust.
- Color extraction rewritten for the case that `\pdfmatch` is not available. This fixes wrong color assignments in case of nesting.
- Color extraction of case `\pdfmatch` is fixed for the case that the color string contains several fill or several stroke operations.

[2010/01/28 v1.2]

- Dependency from package `color` is removed.
- Compatibility for plain $\text{T}_{\text{E}}\text{X}$ and even $\text{iniT}_{\text{E}}\text{X}$ added.

[2016/05/14 v1.3]

- Use package `luatex85` for compatibility with new Lua $\text{T}_{\text{E}}\text{X}$.

[2016/05/17 v1.4]

- Documentation updates.
- adjust `luatex85` reference so that it works in plain $\text{T}_{\text{E}}\text{X}$.

[2018/11/01 v1.5]

- Remove `luatex85` dependency

[2019/12/29 v1.6]

- `iftex` package.

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.

Symbols	
<code>\.</code>	483, 522, 523, 526, 530, 532, 651, 652, 653, 654, 660, 661, 662, 663
<code>\@PackageError</code>	586
<code>\@PackageInfo</code>	335
<code>\@PackageInfoNoLine</code>	180
<code>\@PackageWarning</code>	173, 189, 553, 574, 628, 639
<code>\@PackageWarningNoLine</code>	149
<code>\@ehc</code>	433, 589
<code>\@empty</code>	449
<code>\@firstofone</code>	732, 739
<code>\@gobble</code>	729
<code>\@ifnextchar</code>	597
<code>\@ifundefined</code>	585
<code>\@nil</code>	726, 735, 739
<code>\@nodocument</code>	238, 256
<code>\@undefined</code>	58, 163
<code>\[</code>	529
<code>\]</code>	499, 728, 747
<code>\]</code>	531
A	
<code>\afterassignment</code>	492
<code>\AfterEndPreamble</code>	273, 278
<code>\aftergroup</code>	29, 348, 362
<code>\AfterPackage</code>	270, 271, 272
<code>\AtBeginDocument</code>	266
C	
<code>\catcode</code>	2, 3, 5, 6, 7, 8, 9, 10, 11, 12, 13, 33, 34, 36, 37, 38, 39, 40, 41, 42, 43, 44, 45, 46, 47, 48, 49, 69, 70, 72, 73, 74, 78, 79, 80, 81, 82, 83, 84, 87, 88, 90, 91, 92, 93, 97, 99
<code>\chardef</code>	329, 603, 610, 617
<code>\color</code>	597, 600
<code>\color@begingroup</code>	203, 205, 206
<code>\color@endbox</code>	207, 218
<code>\color@endgroup</code>	204, 207, 247
<code>\color@hbox</code>	205, 218
<code>\color@setgroup</code>	202, 247, 250
<code>\color@vbox</code>	206
<code>\csname</code>	14, 21, 50, 66, 76, 129, 131, 134, 136, 139, 144, 153, 324, 325, 326, 328, 330, 333, 337, 339, 341, 342, 344, 346, 349, 351, 352, 354, 358, 360, 363, 365, 367, 371, 373, 374, 379, 386, 389, 394, 401, 402, 405, 413, 419,
	420, 424, 425, 432, 434, 438, 439, 447, 452, 458, 462, 463, 465
<code>\current@color</code>	619, 632, 643, 687, 690, 707, 710, 738, 739, 744
D	
<code>\DeclareRobustCommand</code>	290, 307
<code>\define@key</code>	416
<code>\dimen</code>	493, 502, 505
<code>\dimexpr</code>	488
E	
<code>\empty</code>	17, 18
<code>\endscname</code>	14, 21, 50, 66, 76, 129, 131, 134, 136, 139, 144, 153, 324, 325, 326, 328, 330, 333, 337, 339, 341, 342, 344, 346, 349, 351, 352, 354, 358, 360, 363, 365, 367, 371, 373, 374, 379, 386, 389, 394, 401, 402, 405, 413, 419, 420, 424, 425, 432, 434, 438, 439, 447, 452, 458, 462, 463, 465
<code>\endgraf</code>	204, 212
<code>\endinput</code>	29, 126
<code>\endlinechar</code>	4, 35, 71, 77, 89
G	
<code>\gdef</code>	222, 339
H	
<code>\hbox</code>	205
I	
<code>\iffalse</code>	137
<code>\ifnum</code>	438, 625, 636, 668, 749
<code>\ifpdf</code>	171
<code>\ifPdfRender@Match</code>	142, 436, 685, 705, 722
<code>\ifPdfRender@Stack</code>	141, 327
<code>\ifPdfRender@Values</code>	317, 384
<code>\iftrue</code>	132
<code>\ifx</code>	15, 18, 21, 50, 58, 61, 144, 163, 229, 238, 248, 256, 387, 449, 499, 545, 552, 566, 573, 604, 605, 613, 627, 638, 675, 728, 747
<code>\immediate</code>	23, 52
<code>\input</code>	158
K	
<code>\kv@key</code>	386, 389
<code>\kv@parse@normalized</code>	385
<code>\kv@value</code>	387, 389
<code>\kvsetkeys</code>	297

L		
<code>\ltx@empty</code>	295, 296, 434, 545, 546, 550, 552, 566, 567, 571, 573, 599, 604, 605, 613, 627, 638, 675, 683, 703	<code>\PdfRender@CurrentColor</code> 607, 614, 632, 643
<code>\ltx@firstofone</code>	592	<code>\PdfRender@CurrentLineWidth</code> 493, 504
<code>\ltx@firstoftwo</code>	282, 285, 302	<code>\PdfRender@define@key</code> 393, 400, 411, 542, 563
<code>\ltx@GlobalAppendToMacro</code>	231, 250, 377, 381	<code>\PdfRender@dimexpr</code> 486, 488, 493
<code>\ltx@gobble</code>	590	<code>\PdfRender@ErrorInvalidValue</code> 428, 441, 450, 460
<code>\ltx@gobbletwo</code>	391	<code>\PdfRender@FillColor</code> 295, 546, 552, 604, 608, 615, 620, 627, 683, 688, 698
<code>\ltx@ifpackagelater</code>	148	<code>\PdfRender@FilterOp</code> 698, 718, 742
<code>\ltx@ifpackageloaded</code>	201	<code>\PdfRender@FindOp</code> 692, 693, 694, 695, 696, 697, 712, 713, 714, 715, 716, 717, 725
<code>\ltx@ifUndefined</code>	172, 179, 192, 265, 268, 276, 282, 284, 301, 403, 411, 457, 485	<code>\PdfRender@GetFillColor</code> 551, 626, 682
<code>\ltx@ifundefined</code>	208, 211, 221	<code>\PdfRender@GetStrokeColor</code> 572, 637, 702
<code>\ltx@pkgextension</code>	153	<code>\PdfRender@let</code> 494, 496, 498
<code>\ltx@secondoftwo</code>	285, 302	<code>\PdfRender@MatchPattern</code> 667, 688, 708
<code>\ltx@space</code>	181, 468, 469, 608, 615, 620, 687, 690, 707, 710	<code>\PdfRender@Matchtrue</code> 186
M		<code>\PdfRender@NeedsCurrentColor</code> 603, 610, 617, 625, 636
<code>\MessageBreak</code>	150, 151, 152, 153, 182, 430, 554, 575, 587	<code>\PdfRender@NewClass</code> 320, 322, 482, 521, 525, 528
N		<code>\PdfRender@NewClassValues</code> 318, 470, 511, 516, 534
<code>\newcommand</code>	196, 197, 289, 306	<code>\PdfRender@newif</code> 128, 141, 142, 317, 323
<code>\normalcolor</code>	222, 228, 231	<code>\PdfRender@NormalColorHook</code> 215, 222, 225, 232, 377
<code>\number</code>	336	<code>\PdfRender@OP</code> 691, 711, 731, 744, 746
P		<code>\PdfRender@OpName</code> 469, 536
<code>\PackageError</code>	429	<code>\PdfRender@OpValue</code> 472, 482, 511, 516, 521, 525, 528
<code>\PackageInfo</code>	26	<code>\PdfRender@PatchColor</code> 260, 264, 266, 270, 271, 273, 278
<code>\par</code>	212	<code>\PdfRender@PatchColorSetGroup</code> 242, 262
<code>\pdfcolorstack</code>	165, 343, 353	<code>\PdfRender@PatchNormalColor</code> 220, 261, 294
<code>\pdfcolorstackinit</code>	164, 174, 331	<code>\PdfRender@Pattern</code> 668, 686, 706
<code>\pdfextension</code>	163, 165, 166	<code>\PdfRender@PatternFillColor</code> 649, 686
<code>\pdffeedback</code>	164	<code>\PdfRender@PatternStrokeColor</code> 658, 706
<code>\pdflastmatch</code>	670, 673	<code>\PdfRender@PostProcessLineWidth</code> 490
<code>\pdfliteral</code>	166, 359, 366	<code>\PdfRender@relax</code> 494, 497
<code>\pdfmatch</code>	181, 438, 668	<code>\PdfRender@RequirePackage</code> 143, 168, 169, 170, 280
<code>\pdfrender</code>	2, 193, 196, 284, 312	<code>\PdfRender@Reset</code> 423
<code>\PdfRender@@FilterOp</code>	743, 746	<code>\PdfRender@Set</code> 375, 382, 418
<code>\PdfRender@@PostProcessLineWidth</code>	492, 498	<code>\PdfRender@SetColor</code> 298, 602
<code>\PdfRender@@TryColor</code>	597, 599	<code>\PdfRender@SetValidate</code> 407, 436
<code>\PdfRender@AtEnd</code>	95, 96, 123, 125, 199, 723, 755	<code>\PdfRender@SetValidateValues</code> 396, 456
<code>\PdfRender@AtEndHook</code>	122, 124, 146, 147	<code>\PdfRender@Stacktrue</code> 177
<code>\PdfRender@Color</code>	544, 545, 550, 565, 566, 571, 600	<code>\PdfRender@String</code> 668, 672, 675, 687, 707
<code>\PdfRender@ColorAvailable</code>	548, 569, 584	
<code>\PdfRender@ColorSetGroupHook</code>	216, 244, 251, 381	
<code>\PdfRender@Current</code>	395, 446, 449, 457, 459, 463	

\PdfRender@StrokeColor	\strip@prefix	670, 673
..... 296, 567, 573, 605, 608,	\strip@pt	505
613, 615, 620, 638, 703, 708, 718		
\PdfRender@temp	T	
. 225, 229, 244, 248, 726, 735, 739	\texorpdfstring	282
\PdfRender@TestBox	\textpdfrender	2, 194, 197, 301
..... 217, 226, 245, 549, 570	\the ... 77, 78, 79, 80, 81, 82, 83, 84, 97	
\PdfRender@texorpdfstring	\TMP@EnsureCode 94, 101, 102,	
..... 281, 293, 310	103, 104, 105, 106, 107, 108,	
\PdfRender@TryColor ... 550, 571, 596	109, 110, 111, 112, 113, 114,	
\PdfRender@Valuesfalse	115, 116, 117, 118, 119, 120, 121	
398		
\PdfRender@Valustrue	V	
319	\vbox	206
\protected		
165, 166		
\ProvidesPackage		
19, 67		
R	W	
\RequirePackage	\write	23, 52
161		
S	X	
\set@color 202, 227, 246, 622, 633, 644	\x	14, 15, 18, 22,
\setbox	26, 28, 51, 56, 66, 75, 87, 503, 508	
218		