# The l3color package
# Experimental color support

The LaTeX3 Project*

Released 2020-06-18

## 1 Color models

A color *model* is a way to represent sets of colors. Different models are particularly suitable for different output methods, *e.g.* screen or print. Parameter-based models can describe a very large number of unique colors, and have a varying number of *axes* which define a color space. In contrast, various proprietary models are available which define *spot* colors.

Core models are used to pass color information to output; these are "native" to l3color. Core models use real numbers in the range $[0, 1]$ to represent values. The core models supported here are

- `gray` Grayscale color, with a single axis running from 0 (fully black) to 1 (fully white)
- `rgb` Red-green-blue color, with three axes, one for each of the components
- `cmyk` Cyan-magenta-yellow-black color, with four axes, one for each of the components
- `spot` Spot color, with one value, the name of the color (see `https://helpx.adobe.com/indesign/using/spot-process-colors.html` for details of the use of spot colors in print)

There are also interface models: these are convenient for users but have to be manipulated before storing/passing to the backend. Interface models are primarily integer-based: see below for more detail. The supported interface models are

- `Hsb` Hue-saturation-brightness color, with three axes, integer in the range $[0, 360]$ for hue, real values in the range $[0, 1]$ for saturation and brightness
- `HSB` Hue-saturation-brightness color, with three axes, integer in the range $[0, 360]$ for hue, integer values in the range $[0, 255]$ for saturation and brightness
- `HTML` HTML format representation of RGB color given as a single six-digit hexadecimal number

*E-mail: latex-team@latex-project.org

- `RGB` Red-green-blue color, with three axes, one for each of the components, values as integers from 0 to 255

All interface models are internally stored as `rgb`.

Additional models may be created to allow mixing of spot colors with each other or with those from other models. See Section 6 for more detail of spot color support.

When color is selected by model, the ⟨*values*⟩ given are specified as a comma-separated list. The length of the list will therefore be determined by the detail of the model involved.

Color models (and interconversion) are complex, and more details are given in the manual to the LaTeX 2$_\varepsilon$ xcolor package and in the *PostScript Language Reference Manual*, published by Addison–Wesley.

## 2 Color expressions

In addition to allowing specification of color by model and values, l3color also supports color expressions. These are created by combining one or more color names, with the amount of each specified as a percentage. The latter is given between `!` symbols in the expression. Thus for example

```
red!50!green
```

is a mixture of 50 % red and 50 % green. A trailing percentage is interpreted as implicitly followed by `white`, and so

```
red!25
```

specifies 25 % red mixed with 75 % white.

Where the models for the mixed colors are different, the model of the first color is used. Thus

```
red!50!cyan
```

will result in a color specification using the `rgb` model, made up of 50 % red and 50 % of cyan *expressed in* `rgb`. As color model interconversion is not exact.

The one exception to the above is where the first model in an expression is `gray`. In this case, the order of mixing is "swapped" internally, so that for example

```
black!50!red
```

has the same result as

```
red!50!black
```

(the predefined colors `black` and `white` use the `gray` model).

Where more than two colors are mixed in an expression, evaluation takes place in a stepwise fashion. Thus in

```
cyan!50!magenta!10!yellow
```

the sub-expression

```
cyan!50!magenta
```

is first evaluated to give an intermediate color specification, before the second step

```
<intermediate>!10!yellow
```

where `<intermediate>` represents this transitory calculated value.

Within a color expression, `.` may be used to represent the color active for typesetting (the current color). This allows for example

```
.!50
```

to mean a mixture of 50 % of current color with white.

(Color expressions supported here are a subset of those provided by the LaTeX $2_\varepsilon$ xcolor package. At present, only such features as are clearly useful have been added here.)

# 3 Named colors

Color names are stored in a single namespace, which makes them accessible as part of color expressions. Whilst they are not reserved in a technical sense, the names `black`, `white`, `red`, `green`, `blue`, `cyan`, `magenta` and `yellow` have special meaning and should not be redefined. Color names should be made up of letters, numbers and spaces only: other characters are reserved for use in color expressions. In particular, `.` represents the current color at the start of a color expression.

`\color_set:nn` `\color_set:nn {⟨name⟩} {⟨color expression⟩}`

Evaluates the ⟨*color expression*⟩ and stores the resulting color specification as the ⟨*name*⟩.

`\color_set:nnn` `\color_set:nnn {⟨name⟩} {⟨model⟩} {⟨value(s)⟩}`

Stores the color specification equivalent to the ⟨*model*⟩ and ⟨*values*⟩ as the ⟨*name*⟩.

`\color_set_eq:nn` `\color_set_eq:nn {⟨name1⟩} {⟨name2⟩}`

Copies the color specification in ⟨*name2*⟩ to ⟨*name1*⟩. The special name `.` may be used to represent the current color, allowing it to be saved to a name.

`\color_show:n` `\color_show:n {⟨name⟩}`

Displays the color specification stored in the ⟨*name*⟩ on the terminal.

# 4 Selecting colors

`\color_select:n` `\color_select:n {⟨color expression⟩}`

Parses the ⟨*color expression*⟩ and then activates the resulting color specification for typeset material.

`\color_select:nn` `\color_select:nn {⟨model⟩} {⟨value(s)⟩}`

Activates the color specification equivalent to the ⟨*model*⟩ and ⟨*value(s)*⟩ for typeset material.

\l_color_fixed_model_tl

When this is set to a non-empty value, colors will be converted to the specified model when they are selected. Note that included images and similar are not influenced by this setting.

## 5 Core color representation

To allow data to be handled internally, l3color uses a simple representation of color, comprising two ⟨*balanced text*⟩ entries, the first the ⟨*model*⟩ and the second the ⟨*values*⟩ given *separated by spaces*.

This core representation is produced when parsing color expressions.

\color_parse:nN

`\color_parse:nN {`⟨*color expression*⟩`} {`⟨*tl*⟩`}`

Parses the ⟨*color expression*⟩ as described above, and sets the ⟨*tl*⟩ to the equivalent ⟨*core color representation*⟩ (used at the backend level and based on `dvips` color representation).

## 6 Spot colors

# Index

The italic numbers denote the pages where the corresponding entry is described, numbers underlined point to the definition, all others indicate the places where it is used.