

MENUKEYS

Tobias Weh

`mail@tobiw.de`

`http://tobiw.de/en`

`http://github.com/tweh/menukeys`

`http://www.ctan.org/pkg/menukeys`

`☒ macros › latex › contrib › menukeys`

2016/08/08 — v1.5

Abstract

This package is build to format menu sequences, paths and keystrokes.

You're welcome to send me feedback, questions, bug reports and feature requests. If you like to support this package – especially improving or proof-reading the manual – send me an e-mail, please.

Many thanks to Ahmed Musa, who provided the list parsing code at <http://tex.stackexchange.com/a/44989/4918>.

Contents

1	Introduction	3
2	Installation	3
3	Package loading and options	4
4	Usage	4
4.1	Basics	4
4.2	Styles	5
4.2.1	Predefined styles	5
4.2.2	Declaring styles	9
4.2.3	Copying styles	10
4.2.4	Changing styles	10
4.3	Color themes	11
4.3.1	Predefined themes	11
4.3.2	Create a theme	11
4.3.3	Copy a theme	12
4.3.4	Change a theme	12
4.4	Menu macros	12
4.4.1	Predefined menu macros	12
4.4.2	Defining or changing menu macros	12
4.5	Keys	13
5	Known issues and bugs	14
6	Implementation	14
6.1	Required packages	14
6.2	Helper macros	15
6.3	Options	15
6.4	Workarounds	16
6.4.1	<code>hyperref</code> 's <code>colorlinks</code> option	16
6.5	Color themes	16
6.5.1	Internal commands	16
6.5.2	User-level commands	16
6.5.3	Predefined themes	18
6.6	Styles	18
6.6.1	Internal commands	18
6.6.2	User-level commands	19
6.6.3	Copying and changing	21
6.6.4	Predefined styles	22
6.7	Menu macros	28
6.7.1	Internal commands	28
6.7.2	User-level commands	29
6.7.3	Predefined menu macros	29
6.8	Keys	30

7	Change history	37
8	Macro index	37

1 Introduction

The `menukeys` package is mainly designed to parse and print sequences of software menus, folders and files or keystrokes. The most predefined styles use the power of `TikZ`¹ to format the output.

For example if you want to tell the reader of a manual how to set the ruler unit you may type

```
To set the unit of the rulers go to \menu{Extras > Settings > Rulers}
and choose between millimeters, inches and pixels. The shortcut
to view the rulers is \keys{cmd + R}. Pressing these keys again
will hide the rulers.
```

```
The standard path for saving your document is \directory{Macintosh HD/Users/
Your Name/Documents} but you can change it at \menu{Extras > Settings
> Saving} by clicking \menu{Change save path}.
```

and get this:

```
To set the unit of the rulers go to Extras >> Settings >> Rulers and choose between
millimeters, inches and pixels. The shortcut to view the rulers is cmd + R.
Pressing these keys again will hide the rulers.
```

```
The standard path for saving your document is Macintosh HD > Users > Your
Name > Documents but you can change it at Extras >> Settings >> Saving by clicking
Change save path.
```

The package is loaded as usual via

```
\usepackage{menukeys}
```

2 Installation

To install `menukeys` manually run

```
latex menukeys.ins
```

and copy `menukeys.sty` to a path where `LATEX` can find it.

To typeset this manual run

```
pdflatex menukeys.dtx
makeindex -s gglo.ist -o menukeys.gls menukeys.glo
makeindex -s gind.ist -o menukeys.ind menukeys.idx
pdflatex menukeys.dtx
pdflatex menukeys.dtx
```

¹ See <http://www.ctan.org/pkg/pgf>.

3 Package loading and options

Since `menukeys` uses `catoptions`, which does some heavy changes on key-value options, it is recommended to load `menukeys` as the **last package** (even after `hyperref`²)!

These are the possible options:

definenummacros: Most of `menukeys`' macros should not conflict with other packages³ but the predefined menu macros should be short and easy-to-read commands, which means that `\menu{A,B,C}` is preferred against `\printmenusequence{A,B,C}`. For that it's not unlikely that they conflict with other packages. To prevent this you can tell `menukeys` to not define them by calling the option `definenummacros=false`. The default value is `true`.

`definenummacros` (opt.)

If you do so you have to define your own menu macros, see section 4.4 for details.

definekeys: Equal to `definenummacros` for the key macros. The default value is `true`.

`definekeys` (opt.)

mackeys: This option allows you to decide whether the mac keys are shown as text (`mackeys=text`) or symbols (`mackeys=symbols`). The default value is `symbols`.

`mackeys` (opt.)

os: You can specify the OS by saying `os=mac` or `os=win`. This will cause some key macros to be rendered differently. The default value is `mac`.

`os` (opt.)

hyperrefcolorlinks: Use this if you want `hyperref`'s colored links, since you can't use the `hyperref` option `colorlinks` directly (see sec. 5 and 6.4.1).

`hyperrefcolorlinks` (opt.)

4 Usage

4.1 Basics

`menukeys` comes with three “menu macros” that parse and print lists. We have `\menu{<menu sequence>}`, with `>` as default input list separator, `\directory{<path and files>}` with `/` as default separator and `\keys{<keystrokes>}` with `+` as default separator. You've seen examples for all of them in section 1.

`\menu`
`\directory`
`\keys`

These macros have also an optional argument to set the input list separator. E.g. if you want to put in your menus with `,` instead of `>` you can say `\menu[,]{<menu sequence>}`.⁴


The possible input separators are `/`, `=`, `*`, `+`, `,`, `;`, `:`, `-`, `>`, `<` and `bslash` (to use `\` as separator). You can hide a separator from the parser by putting a

² See <http://tex.stackexchange.com/q/237683/4918> and <https://github.com/tweh/menukeys/issues/41>.

³ If you find a conflict send an e-mail.

⁴ If you want to change the input separator globally it's recommended to renew the menu macro as described in section 4.4.

part of the sequence in braces. Spaces around the separator will be ignored, i.e. `\keys{\ctrl+C}` equals `\keys{\ctrl + C}`.

Example `\menu[,]{Extras,Settings,{Units, rulers and origin}}` gives



4.2 Styles

`menukeys` defines several “styles” that determine the output format of a menu macro. There are some predefined styles and others can be created by the user.

4.2.1 Predefined styles


Name: `menu`



This is some more or less blind text, to demonstrate how the sequence looks in text. This  is the result of a style which name is `menu`. And again some blind text without any sense.


Name: `roundedmenu`



This is some more or less blind text, to demonstrate how the sequence looks in text. This  is the result of a style which name is `roundedmenu`. And again some blind text without any sense.

Name: `angularmenu`



This is some more or less blind text, to demonstrate how the sequence looks in text. This  is the result of a style which name is `angularmenu`. And again some blind text without any sense.

Name: **roundedkeys**

Ctrl + Alt + Q

S

This is some more or less blind text, to demonstrate how the sequence looks in text. This Ctrl + Alt + Q is the result of a style which name is **roundedkeys**. And again some blind text without any sense.

The color of + is taken from optional color B.

Name: **shadowedroundedkeys**

Ctrl + Alt + Q

S

This is some more or less blind text, to demonstrate how the sequence looks in text. This Ctrl + Alt + Q is the result of a style which name is **shadowedroundedkeys**. And again some blind text without any sense.

*The color of + is taken from optional color B.
The shadow color is taken from optional color C.*

Name: **angularkeys**

Ctrl + Alt + Q

S

This is some more or less blind text, to demonstrate how the sequence looks in text. This Ctrl + Alt + Q is the result of a style which name is **angularkeys**. And again some blind text without any sense.

The color of + is taken from optional color B.

Name: **shadowedangularkeys**

Ctrl + Alt + Q

S

This is some more or less blind text, to demonstrate how the sequence looks in text. This Ctrl + Alt + Q is the result of a style which name is **shadowedangularkeys**. And again some blind text without any sense.

*The color of + is taken from optional color B.
The shadow color is taken from optional color C.*

Name: `typewriterkeys`

Ⓢ + Ⓚ

Ⓢ

This is some more or less blind text, to demonstrate how the sequence looks in text. This Ⓢ + Ⓚ is the result of a style which name is `typewriterkeys`. And again some blind text without any sense.

The color of + is taken from optional color B.

Name: `paths`

C: ›User›Folder›MyFile.tex

MyFile.tex

This is some more or less blind text, to demonstrate how the sequence looks in text. This C: ›User›Folder›MyFile.tex is the result of a style which name is `paths`. And again some blind text without any sense.

The sep color is taken from optional color C.

Name: `pathswithfolder`

☐ C: ›User›Folder›MyFile.tex

☐ MyFile.tex

This is some more or less blind text, to demonstrate how the sequence looks in text. This ☐ C: ›User›Folder›MyFile.tex is the result of a style which name is `pathswithfolder`. And again some blind text without any sense.

The folder draw color is taken from optional color B.

The folder fill color is taken from optional color A.

The sep color is taken from optional color C.

Name: `pathswithblackfolder`

▣ C: ›User›Folder›MyFile.tex

▣ MyFile.tex

This is some more or less blind text, to demonstrate how the sequence looks in text. This ▣ C: ›User›Folder›MyFile.tex is the result of a style which name is `pathswithblackfolder`. And again some blind text without any sense.

The folder draw color is taken from optional color B.

The folder fill color is taken from optional color C.

The sep color is taken from optional color C.

The following three styles allow paths elements to be hyphenated, but they insert only a line break without a hyphen dash. Note that they only work with T1 and

OT1 encoding (at least I tested only these ones) and that this in some cases doesn't work very well.

Name: hyphenatepaths

C: › Database › User › ALongUserNameHere › ALongerFolderNameAtThisPlace › MyFile.tex

MyFile.tex

This is some more or less blind text, to demonstrate how the sequence looks in text. This C: › Database › User › ALongUserNameHere › ALongerFolderNameAtThisPlace › MyFile.tex is the result of a style which name is hyphenatepaths. And again some blind text without any sense.

The sep color is taken from optional color C.

Name: hyphenatepathswithfolder

☐ C: › Database › User › ALongUserNameHere › ALongerFolderNameAtThisPlace › MyFile.tex

☐ MyFile.tex

This is some more or less blind text, to demonstrate how the sequence looks in text. This ☐ C: › Database › User › ALongUserNameHere › ALongerFolderNameAtThisPlace › MyFile.tex is the result of a style which name is hyphenatepathswithfolder. And again some blind text without any sense.

The folder draw color is taken from optional color B.

The folder fill color is taken from optional color A.

The sep color is taken from optional color C.

Name: hyphenatepathswithblackfolder

▣ C: › Database › User › ALongUserNameHere › ALongerFolderNameAtThisPlace › MyFile.tex

▣ MyFile.tex

This is some more or less blind text, to demonstrate how the sequence looks in text. This ▣ C: › Database › User › ALongUserNameHere › ALongerFolderNameAtThisPlace › MyFile.tex is the result of a style which name is hyphenatepathswithblackfolder. And again some blind text without any sense.

The folder draw color is taken from optional color B.

The folder fill color is taken from optional color C.

The sep color is taken from optional color C.

`\drawtikzfolder` **Hint** The folder is drawn with the command `\drawtikzfolder` which is part of `menukeys` and has two optional arguments to change the color of the lines and the fill color of the front:
`\drawtikzfolder` [*front fill*] [*draw*]

4.2.2 Declaring styles

`\newmenustylesimple` The simplest way to define a new style is to use `\newmenustylesimple`. It has six arguments: `\newmenustylesimple` *** {*name*} [*pre*] {*style*} [*sep*] [*post*] {*theme*}

name is the name of the new style. It must follow the specifications of T_EX control sequences, which means it must contain only letters and no numbers.

pre is the code which is executed before a menu macro.

style is the style for the first list element. It has to be a TikZ-style which is applied to a node, e.g. `draw,blue`.

sep is the code executed between the lists elements, e.g. some space or a symbol.

post is the code which is executed after a menu macro.

theme is a color theme (see section 4.3).

Example Let us consider we want a list that prints a frame around its elements and separates them by a star. We can use

```
\newmenustylesimple{mystyle}{draw}[$\ast$]{mycolors}
```

`\newmenustyle` The more advanced command is `\newmenustyle`. It has nine arguments: `\newmenustyle` *** {*name*} [*pre*] {*first*} [*sep*] {*mid*} {*last*} {*single*} [*post*] {*theme*}

name is the name of the new style. It must follow the specifications of T_EX control sequences, which means it must contain only letters and no numbers.

pre is the code which is executed before a menu macro.

first is the style for the first list element. It has to be a TikZ-style which is applied to a node, e.g. `draw,blue`.

sep is the code executed between the lists elements, e.g. some space or a symbol.

mid is the style for all elements between the first and the last one. It has to be a TikZ style.

last is the style for the last list element. It has to be a TikZ style.

single this style is used if the list contains only one element. It has to be a TikZ style.

post is the code which is executed after a menu macro.

theme is a color theme (see section 4.3).

Example We can extend the previous example and desire that the first and the last element became red, and a single element should have a dashed frame. Furthermore the menu sequence should be preceded and followed by a bullet point:

```
\newmenustyle{mystyle}[$\bullet$]{draw,red}[$\ast$]%
{draw}{draw,red}{draw,dashed}[$\bullet$]
```

If the TikZ node system doesn't fit your needs there are the **starred versions**: Use them and the arguments *(first)*, *(mid)*, *(last)*, *(single)* can be any L^AT_EX code.

`\CurrentMenuElement` To access the current list element use `\CurrentMenuElement`.

Example consider that we want all menu elements simple be fat and not drawn with a TikZ node. The separator should be the star again:

```
\newmenustylesimple*{mystyle}{\textbf{\CurrentMenuElement}}[$\ast$]
```

`\usemenucolor` If you want to make your own style you must take care of using the color theme. To access a color of the currently applied theme while defining a style use `\usemenucolor{<element>}` (See section 4.3 for details about possible elements).

4.2.3 Copying styles

`\copymenustyle` To copy an existing style to a new style use `\copymenustyle {<copy>}{<original>}`.

Example To copy the definition of `mystyle` to `mycopy` use

```
\copymenustyle{mycopy}{mystyle}
```

4.2.4 Changing styles

`\changemenuelement` The simplest change we can imagine is to change a single element or the color theme of an existing style. For the first case there is `\changemenuelement{*}{<name>}{<element>}{<definition>}`, where the starred version works like the one of `\newmenustyle` does.

Example To change the single element of `mystyle` from dashed to solid use the following code. You may save the original style by copying it as described above.

```
\changemenuelement{mystyle}{single}{draw}
```

`\changemenucolortheme` To satisfy the second case use `\changemenucolortheme {<name>}{<color theme>}`.

Example To change the color theme of `mystyle` to `myothercolors` call

```
\changemenucolortheme{mystyle}{myothercolors}
```

`\renewmenustylesimple` The next level is redefining a style. This package provides the following
`\providemenustylesimple` macros the work like their L^AT_EX-paragons and have the same arguments as the
`\renewmenustyle` above described macros: `\renewmenustylesimple`, `\providemenustylesimple`,
`\providemenustyle` `\renewmenustyle` and `\providemenustyle`.

4.3 Color themes

To make the colors of a style become changeable without touching the style itself, `menukeys` uses “color themes”. Every color theme must contain three color definitions that can be used to draw a `node` background, a `node` frame and a text color, and additionally two optional colors used by some themes.

4.3.1 Predefined themes

There are two predefined color themes

Name: `gray`

Background: Border: Text: (A: B: C:)

Name: `blackwhite`

Background: Border: Text: (A: B: C:)

4.3.2 Create a theme

`\newmenucolortheme` To create a new theme use `\newmenucolortheme`. It uses the following arguments:
`\newmenucolortheme{<name>}{<model>}{<bg>}{
}{<txt>}[<a>][][<c>]`

name is the name of the theme and must contain only letters.

model is the `xcolor` color model which is used to define a color, e.g. `named`, `rgb`, `cmyk`, ...

bg is the color definition for the `node` background.

br is the color definition for the `node` border.

txt is the color definition for the `node`'s text.

a is an optional additional color (by default same as `bg`).

b is an optional additional color (by default same as `br`).

c is an optional additional color (by default same as `txt`).

Example To create a theme called `mycolors` we can say

```
\newmenucolortheme{mycolors}{named}{red}{green}{blue}
```

4.3.3 Copy a theme

`\copymenucolortheme` To copy the definitions of one theme to another, use `\copymenucolortheme` $\{\langle copy \rangle\}\{\langle original \rangle\}$.

Example To copy the colors of `mycolors` to `copycolors` type

```
\copymenucolortheme{copycolors}{mycolors}
```

4.3.4 Change a theme

`\changemenucolor` If you want to change the color of a theme's element use `\changemenucolor` $\{\langle name \rangle\}\{\langle element \rangle\}\{\langle model \rangle\}\{\langle color\ definition \rangle\}$, where `name` is the theme's name and `element` is `bg`, `br`, or `txt`.

Example Let's change the text color of `mycolors`:

```
\changemenucolor{mycolors}{txt}{named}{gray}
```

`\renewmenucolortheme` To redefine a complete theme use `\renewmenucolortheme`. It works with the same arguments as `\newmenucolortheme`.

4.4 Menu macros

The “menu macros” take a list separated by a special symbol to print it with a menu style.

4.4.1 Predefined menu macros

See section 4.1.

4.4.2 Defining or changing menu macros

`\newmenumacro` To define a new menu macro call `\newmenumacro` $\{\langle macro \rangle\} [\langle input\ sep \rangle] \{\langle style \rangle\}$.

name is a L^AT_EX control sequence name.

input sep is the default separator used in the input list (see section 4.1 for a list of valid separators).

If you don't give it the package's default (,) is used.

style is a menu style.

This will give you a macro like `\langle macro \rangle [\langle input\ sep \rangle] \{\langle list \rangle\}`

Example Assuming you need a command to format Windows paths, you can define it with

```
\newmenumacro{\winpath}[bslash]{mystyle}
```

and then use it as e.g. `\winpath{C:\System\Deep\Deeper\YourFile.txt}`. Note that `mystyle` must be defined before you call `\newmenumacro`.

`\providemenumacro` There are also the two commands `\providemenumacro` and `\renewmenumacro`
`\renewmenumacro` which take the same arguments as `\newmenumacro` and work like the L^AT_EX macros
`\renewcommand` and `\providecommand`.

Example To change the default input separator of `\menu` you must know the default style (which is `menus`) and then you can say

```
\renewmenumacro{\menu}{,}{menus}
```

4.5 Keys

The `menukeys` package comes with some macros to print special keys in the sequences set with `\keys`. Depending on the given OS (see section 3) some macros behave differently to be able to use a key even if it's undefined via the `os` option macros like `\<key>mac` and `\<key>win` that will always give the right symbol.

The full list of key macros is shown in table 1.

Table 1: Overview of all key macros.

Macro	Mac	Win.	Macro	Mac	Win.
<code>\shift</code>	⇧	⇧	<code>\winmenu</code>		☰
<code>\capslock</code>	⇨	⇩	<code>\backspace</code>	←	←
<code>\tab</code>	→	↔	<code>\del</code>	Del. / ☒	Del.
<code>\esc</code>	esc / ⌘	Esc	<code>\backdel</code>	Del. / ☓	Del.
<code>\oldesc</code>	esc / ⌘	Esc	<code>\arrowkey{^}</code>	↑	↑
<code>\ctrl</code>	ctrl	Ctrl	<code>\arrowkeyup</code>	↑	↑
<code>\Alt</code>	alt / ⌥	Alt	<code>\arrowkey{v}</code>	↓	↓
<code>\AltGr</code>		Alt Gr	<code>\arrowkeydown</code>	↓	↓
<code>\cmd</code>	cmd / ⌘		<code>\arrowkey{>}</code>	→	→
<code>\Space</code>	[empty sp.]	[empty sp.]	<code>\arrowkeyright</code>	→	→
<code>\SPACE</code>	Space	Space	<code>\arrowkey{<}</code>	←	←
<code>\return</code>	↵	↵	<code>\arrowkeyleft</code>	←	←
<code>\enter</code>	↵	Enter			

`\arrowkey` The macro `\arrowkey{<direction>}` is a little special since it takes the direction as a single character `^`, `v` (lower case `v`), `>` or `<`.

`\ctrlname` The texts for `\ctrl`, `\del` and `\SPACE` are saved in `\ctrlname`, `\delname`,
`\delname` `\spacename` respectively. So you can change them with `\renewcommand`.

`\spacename` The rendering of some Mac macros depend on the option `mackeys` The different
`mackeys` (opt.) versions are shown in the table (left: text, right: symbols).

I apologize that there are no commands for the windows key and the apple logo, but that would be a copyright infringement.

5 Known issues and bugs

- If you use the `inputenc` package `menukeys` must be loaded after it. Otherwise some key macros get corrupted.
- `menukeys` must be loaded after `xcolor`, if you load the latter with options. Otherwise you'll get an option clash. Since `menukeys` loads `xcolor` internally you may pass options as global options via `\documentclass`.

Example Set `xcolor` to `cmym` model:

```
\documentclass[cmym]{article}
\usepackage{menukeys}
\begin{document}
  Hello World!
\end{document}
```

- Using `hyperref` with the `colorlinks` options causes an option clash. If you want colored links please load `hyperref` *without* this option and load `menukeys` with `hyperrefcolorlinks`.

If you find something to add to this list please send me an e-mail or report a bug on GitHub (<https://github.com/tweh/menukeys>).

6 Implementation

6.1 Required packages

Load the required packages

```
1 \RequirePackage{xparse}
2 \RequirePackage{xstring}
3 \RequirePackage{etoolbox}
```

Furthermore we need `TikZ` and some of its libraries,

```
4 \RequirePackage{tikz}
5 \usetikzlibrary{calc,shapes.symbols,shadows}
```

the color package `xcolor` and `adjustbox` for the `typewriterkeys` style.

```
6 \RequirePackage{xcolor}
7 \RequirePackage{adjustbox}
```

Load `relsize` to be able to change the font size relative to the surrounding text.

```
8 \RequirePackage{relsize}
```

To define the list parsing commands and allow `\` as a separator we load `catoptions`

```
9 \RequirePackage{catoptions}[2011/12/07]
```

6.2 Helper macros

```
\tw@mk@error Define macros to call \PackageError and warnings
\tw@mk@warning 10 \newcommand*\tw@mk@error}[2] [Please consult the manual for more information.]{%
\tw@mk@warning@noline 11   \PackageError{menukeys}{#2}{#1}%
12 }
13 \newcommand*\tw@mk@warning}[1]{%
14   \PackageWarning{menukeys}{#1}%
15 }
16 \newcommand*\tw@mk@warning@noline}[1]{%
17   \PackageWarningNoLine{menukeys}{#1}%
18 }

\tw@mk@tempa Some commands for temporary use:
\tw@mk@tempb 19 \def\tw@mk@tempa{}
20 \def\tw@mk@tempb{}

\tw@mk@gobble@args Define a command to gobble arguments.
21 \DeclareDocumentCommand{\tw@mk@gobble@args}{m}{%
22   \RenewDocumentCommand{\tw@mk@tempa}{#1}{}%
23   \tw@mk@tempa%
24 }
```

6.3 Options

First we declare and process the package options

```
25 \RequirePackage{kvoptions}
26 \SetupKeyvalOptions{
27   family=tw@mk,
28   prefix=tw@mk@
29 }
30 \DeclareBoolOption[true]{definenumacros}
31 \DeclareBoolOption[true]{definekeys}
32 \DeclareBoolOption[false]{hyperrefcolorlinks}
33 \DeclareStringOption[mac]{os}
34 \DeclareStringOption[symbols]{mackeys}
35 \ProcessKeyvalOptions{tw@mk}\relax
```

Now we have to do some error treatment:

```
36 \IfSubStr{.mac.win.}{.\tw@mk@os.}{}{%
37   \tw@mk@error{Unknown value for option 'os'\MessageBreak
38   Possible values are 'mac' or 'win'.}%
39 }
40 \IfSubStr{.symbols.text.}{.\tw@mk@mackeys.}{}{%
41   \tw@mk@error{Unknown value for option 'mackeys'\MessageBreak
42   Possible values are 'symbols' or 'text'.}%
43 }
```


6.4 Workarounds

Some workarounds to “slove” some incompatibilities:

6.4.1 hyperref’s colorlinks option

Since the `colorlinks` option of `hyperref` loads `color` (with some kind of `\AtBeginDocument`) it results in an option `clas` due to the changes made by `catoptions`. Thus one can’t use `colorlinks`. Here we provide the code to activate colored links without the extra loading of `color`.

```
44 \iftw@mk@hyperrefcolorlinks
45   \Hy@AtBeginDocument{% (hyperref.sty, line 4790)
46     \def\@pdfborder{0 0 0}% (hyperref.sty, line 4806...)
47     \let\@pdfborderstyle\@empty
48 %     \ifHy@typexml% <-----+
49 %     \else% | This part
50 %       \Hy@CatcodeWrapper{% | bust be
51 %         \RequirePackage{color}% | omitted
52 %       }% |
53 %     \fi% <-----+
54     \def\Hy@colorlink#1{%
55       \begingroup
56       \HyColor@UseColor#1%
57     }%
58     \def\Hy@endcolorlink{\endgroup}%
59     \Hy@Info{Link coloring ON}%
60   }
61 \fi
```

6.5 Color themes

6.5.1 Internal commands

`\tw@make@color@theme` First we define an internal command to make a color theme

```
62 \newcommand*\tw@make@color@theme}[8]{%
63   \definecolor{tw@color@theme@#1@bg}{#2}{#3}%
64   \definecolor{tw@color@theme@#1@br}{#2}{#4}%
65   \definecolor{tw@color@theme@#1@txt}{#2}{#5}%
66   \definecolor{tw@color@theme@#1@a}{#2}{#6}%
67   \definecolor{tw@color@theme@#1@b}{#2}{#7}%
68   \definecolor{tw@color@theme@#1@c}{#2}{#8}%
69 }
```

6.5.2 User-level commands

`\newmenucolortheme` After that we define the user-level commands:

```
\renewmenucolortheme 70 \NewDocumentCommand{\newmenucolortheme}{ m m m m m 0{#3} 0{#4} 0{#5} }{%
71   \@ifundefinedcolor{tw@color@theme@#1@bg}{%
72     \tw@make@color@theme{#1}{#2}{#3}{#4}{#5}{#6}{#7}{#8}%
```

```

73 }{%
74 \tw@mk@error{Color theme '#1' already defined!\MessageBreak
75 Use \string\renewmenucolortheme\space instead.}%
76 }
77 }
78 \NewDocumentCommand{\renewmenucolortheme}{ m m m m m O{#3} O{#4} O{#5} }{%
79 \tw@make@color@theme{#1}{#2}{#3}{#4}{#5}{#6}{#7}{#8}%
80 }

```

`\changemenucolor` Lastly we define the changing and copying commands

```

\changemenucolor
\copymenucolortheme
81 \newcommand*{\changemenucolor}[4]{%
82 \IfSubStr{ bg br txt }{ #2 }{%
83 \definecolor{tw@color@theme@#1@#2}{#3}{#4}%
84 }{%
85 \tw@mk@error{No such color element ('#2')!\MessageBreak
86 Possible values are bg, br and txt.}
87 }%
88 }
89 \newcommand*{\copymenucolortheme}[2]{%
90 \@ifundefinedcolor{tw@color@theme@#1@bg}{%
91 \colorlet{tw@color@theme@#1@bg}{tw@color@theme@#2@bg}%
92 \colorlet{tw@color@theme@#1@br}{tw@color@theme@#2@br}%
93 \colorlet{tw@color@theme@#1@txt}{tw@color@theme@#2@txt}%
94 \colorlet{tw@color@theme@#1@a}{tw@color@theme@#2@a}%
95 \colorlet{tw@color@theme@#1@b}{tw@color@theme@#2@b}%
96 \colorlet{tw@color@theme@#1@c}{tw@color@theme@#2@c}%
97 }{%
98 \tw@mk@error{Color theme '#1' already defined!\MessageBreak
99 Use \string\renewmenucolortheme\space instead.}
100 }
101 }

```

`\changemenucolortheme` To be able to change the color theme of a style we must define this:

```

102 \newcommand{\changemenucolortheme}[2]{%
103 \ifcsundef{tw@style@#1@pre}{%
104 \tw@mk@error{Style '#1' undefined!\MessageBreak
105 Maybe you misspelled it?}%
106 }{%
107 \@ifundefinedcolor{tw@color@theme@#2@bg}{%
108 \tw@mk@error{Color theme '#2' is not defined!}%
109 }{%
110 \csdef{tw@style@#1@color@theme}{#2}%
111 }%
112 }%
113 }

```

`\usemenucolor` To use a color of a theme we define `\usemenucolor` as following.

```

114 \newcommand{\usemenucolor}[1]{%
115 tw@color@theme@tw@current@color@theme @#1%
116 }

```

6.5.3 Predefined themes

There are two predefined color themes

```
117 \newmenucolortheme{gray}{gray}{0.95}{0.3}{0}[0.95][0][0]
118 \newmenucolortheme{blacknwhite}{gray}{1}{0}{0}[1][0][0]
```

6.6 Styles

The style generating commands will set some commands that are named like `\tw@style@<name>@<element>`.

```
\tw@default@sep Before we can define the internal declaring macro to use it later in the user level
\tw@default@pre commands, we have to set some defaults for the optional arguments
\tw@default@post
119 \newcommand{\tw@default@sep}{%
120   \hspace{0.2em plus 0.1em minus 0.5em}%
121 }
122 \newcommand{\tw@default@pre}{}
123 \newcommand{\tw@default@post}{}%
```

6.6.1 Internal commands

Now we can define the internal commands.

```
\tw@declare@style@simple Our first step is to define the simple command.
124 \DeclareDocumentCommand{\tw@declare@style@simple}{%
125   s m 0{\tw@default@pre} m 0{\tw@default@sep} 0{\tw@default@post} m
126 }{%
127   \csdef{tw@style@#2@color@theme}{#7}%
128   \csdef{tw@style@#2@pre}{#3}%
129   \csdef{tw@style@#2@sep}{#5}%
130   \csdef{tw@style@#2@post}{#6}%
131   \IfBooleanTF{#1}{%
132     \csdef{tw@style@#2@single}{#4}%
133     \csdef{tw@style@#2@first}{#4}%
134     \csdef{tw@style@#2@mid}{#4}%
135     \csdef{tw@style@#2@last}{#4}%
136   }{%
137     \csdef{tw@style@#2@single}{%
138       \tikz[baseline=(tw@node.base)]{%
139         \node(tw@node)[#4]{\strut\CurrentMenuElement};}%
140     \csdef{tw@style@#2@first}{%
141       \tikz[baseline=(tw@node.base)]{%
142         \node(tw@node)[#4]{\strut\CurrentMenuElement};}%
143     \csdef{tw@style@#2@mid}{%
144       \tikz[baseline=(tw@node.base)]{%
145         \node(tw@node)[#4]{\strut\CurrentMenuElement};}%
146     \csdef{tw@style@#2@last}{%
147       \tikz[baseline=(tw@node.base)]{%
148         \node(tw@node)[#4]{\strut\CurrentMenuElement};}%
149     }
```

```

149   }%
150 }

```

`\tw@declare@style` The next step is to create the extended command. This command must have ten arguments (including the star) so we have to define a helping macro to grab the last two macros.

```

151 \DeclareDocumentCommand{\tw@declare@style@extra@args}{%
152   0{\tw@default@post} m
153 }{%
154   \csdef{tw@style@\tw@current@style @post}{#1}%
155   \csdef{tw@style@\tw@current@style @color@theme}{#2}%
156 }

```

Now we can define `\tw@declare@style`:

```

157 \DeclareDocumentCommand{\tw@declare@style}{%
158   s m 0{\tw@default@pre} m 0{\tw@default@sep} m m m
159 }{%
160   \def\tw@current@style{#2}
161   \csdef{tw@style@#2@pre}{#3}%
162   \csdef{tw@style@#2@sep}{#5}%
163   \IfBooleanTF{#1}{%
164     \csdef{tw@style@#2@single}{#8}%
165     \csdef{tw@style@#2@first}{#4}%
166     \csdef{tw@style@#2@mid}{#6}%
167     \csdef{tw@style@#2@last}{#7}%
168   }{%
169     \csdef{tw@style@#2@single}{%
170       \tikz[baseline=(tw@node.base)]{%
171         \node(tw@node)[#8]{\strut\CurrentMenuElement};}%
172     \csdef{tw@style@#2@first}{%
173       \tikz[baseline=(tw@node.base)]{%
174         \node(tw@node)[#4]{\strut\CurrentMenuElement};}%
175     \csdef{tw@style@#2@mid}{%
176       \tikz[baseline=(tw@node.base)]{%
177         \node(tw@node)[#6]{\strut\CurrentMenuElement};}%
178     \csdef{tw@style@#2@last}{%
179       \tikz[baseline=(tw@node.base)]{%
180         \node(tw@node)[#7]{\strut\CurrentMenuElement};}%
181   }%
182   \tw@declare@style@extra@args%
183 }

```

6.6.2 User-level commands

```

newmenustylesimple It's time to define the user-level commands now:
renewmenustylesimple 184 \NewDocumentCommand{\newmenustylesimple}{s m}{%
providemenustylesimple 185   \ifcsundef{tw@style@#2@pre}{%
newmenustyle 186     \IfBooleanTF{#1}{%
renewmenustyle 187       \tw@declare@style@simple*{#2}%
providemenustyle 188     }{%

```

```

189         \tw@declare@style@simple{#2}%
190     }%
191 }{%
192     \tw@mk@error{Style '#2' already defined!\MessageBreak
193     Use \string\renewmenustylesimple\space instead.}%
194     \tw@mk@gobble@args{o m o o m}%
195 }%
196 }
197 \NewDocumentCommand{\renewmenustylesimple}{s m}{%
198     \IfBooleanTF{#1}{%
199         \tw@declare@style@simple*{#2}%
200     }{%
201         \tw@declare@style@simple{#2}%
202     }%
203 }
204 \NewDocumentCommand{\providemenustylesimple}{s m}{%
205     \ifcsundef{tw@style@#2@pre}{%
206         \IfBooleanTF{#1}{%
207             \tw@declare@style@simple*{#2}%
208         }{%
209             \tw@declare@style@simple{#2}%
210         }%
211     }{%
212         \tw@mk@warning{Trying to provide style '#2' failed,\MessageBreak
213         because it's already defined.\MessageBreak
214         You may use \string\renewmenustylesimple\space instead.}%
215         \tw@mk@gobble@args{o m o o m}%
216     }%
217 }
218
219 \NewDocumentCommand{\newmenustyle}{s m}{%
220     \ifcsundef{tw@style@#2@pre}{%
221         \IfBooleanTF{#1}{%
222             \tw@declare@style*{#2}%
223         }{%
224             \tw@declare@style{#2}%
225         }%
226     }{%
227         \tw@mk@error{Style '#2' already defined!\MessageBreak
228         Use \string\renewmenustyle\space instead.}%
229         \tw@mk@gobble@args{o m o m m m o m}%
230     }%
231 }
232 \NewDocumentCommand{\renewmenustyle}{s m}{%
233     \IfBooleanTF{#1}{%
234         \tw@declare@style*{#2}%
235     }{%
236         \tw@declare@style{#2}%
237     }%
238 }

```

```

239 \NewDocumentCommand{\providemenustyle}{s m}{%
240   \ifcsundef{tw@style@#2@pre}{%
241     \IfBooleanTF{#1}{%
242       \tw@declare@style*{#2}%
243     }{%
244       \tw@declare@style{#2}%
245     }%
246   }{%
247     \tw@mk@warning{Trying to provide style #2 failed,\MessageBreak
248     because it's already defined.\MessageBreak
249     You may use \string\renewmenustyle\space instead.}%
250     \tw@mk@gobble@args{o m o m m m o m}%
251   }%
252 }

```

6.6.3 Copying and changing

`\copymenustyle` The last two steps in this part are to define a command to copy styles

```

253 \newcommand*{\copymenustyle}[2]{%
254   \ifcsundef{tw@style@#1@pre}{%
255     \ifcsundef{tw@style@#2@pre}{%
256       \tw@mk@error{Can't copy not existing style ('#2')!}%
257     }{%
258       \csletcs{tw@style@#1@pre}{tw@style@#2@pre}%
259       \csletcs{tw@style@#1@post}{tw@style@#2@post}%
260       \csletcs{tw@style@#1@sep}{tw@style@#2@sep}%
261       \csletcs{tw@style@#1@single}{tw@style@#2@single}%
262       \csletcs{tw@style@#1@first}{tw@style@#2@first}%
263       \csletcs{tw@style@#1@mid}{tw@style@#2@mid}%
264       \csletcs{tw@style@#1@last}{tw@style@#2@last}%
265       \csletcs{tw@style@#1@color@theme}{tw@style@#2@color@theme}%
266     }%
267   }{%
268     \tw@mk@error{Style '#1' already exists!}%
269   }%
270 }

```

`\changemenuelement` and one to change a single element of a style.

```

271 \NewDocumentCommand{\changemenuelement}{s m m m}{%
272   \ifcsundef{tw@style@#2@pre}{%
273     \tw@mk@error{Style '#2' undefined.}%
274   }{%
275     \IfSubStr{ single first middle last pre post sep }{ #3 }{%
276       \IfBooleanTF{#1}{%
277         \csdef{tw@style@#2@#3}{#4}%
278       }{%
279         \IfSubStr{ pre post sep }{ #3 }{%
280           \csdef{tw@style@#2@#3}{#4}%
281         }%

```

```

282         \csdef{tw@style@#2@#3}{%
283             \tikz[baseline=(tw@node.base)]{%
284                 \node(tw@node)[#4]{\strut\color{\usemenucolor{txt}}\CurrentMenuElement};}%
285             }%
286         }%
287     }{\tw@mk@error{No element '#3'. Possible values are\MessageBreak
288         single, first, middle, last, pre, post or sep.}}%
289 }%
290 }

```

6.6.4 Predefined styles

We define several styles for menu sequences, paths and keystrokes.

`tw@set@tikz@colors` First we define a TikZ-style to apply the color theme to a node easily

```

291 \tikzset{tw@set@tikz@colors/.style={%
292     draw=\usemenucolor{br},
293     fill=\usemenucolor{bg},
294     text=\usemenucolor{txt},
295 }}

```

Now we can define the styles. To keep the most settings of a style together we make additional TikZ-styles instead of setting everything directly to the nodes.

```

296 \tikzset{tw@menus@base/.style={%
297     tw@set@tikz@colors,
298     rounded corners=0.15ex,
299     inner sep=0pt,
300     inner xsep=2pt,
301     text height=1.825ex,
302     text depth=0.7ex,
303     minimum width=1.5em,
304     font=\relsize{-1}\sffamily,
305     signal,
306     signal to=nowhere,
307     signal pointer angle=110,
308 }}
309 \tw@declare@style*{menus}{%
310     \tikz[baseline={{$(tw@node.base)+(0,-0.2ex)$}}]{%
311         \node(tw@node)[tw@menus@base,signal to=east]%
312             {\strut\color{\usemenucolor{txt}}\CurrentMenuElement};}%
313 }[\hspace{-0.2em}\hspace{0em plus 0.1em minus 0.05em}]%
314 {%
315     \tikz[baseline={{$(tw@node.base)+(0,-0.2ex)$}}]{%
316         \node(tw@node)[tw@menus@base,signal from=west,signal to=east]%
317             {\strut\color{\usemenucolor{txt}}\CurrentMenuElement};}%
318 }{%
319     \tikz[baseline={{$(tw@node.base)+(0,-0.2ex)$}}]{%
320         \node(tw@node)[tw@menus@base,signal from=west,%
321             {\strut\color{\usemenucolor{txt}}\CurrentMenuElement};}%
322 }{%

```

```

323 \tikz[baseline={{$(tw@node.base)+(0,-0.2ex)$}}]{%
324 \node(tw@node)[tw@menus@base]{\strut\color{\usemenucolor{txt}}\CurrentMenuElement};}%
325 }{gray}
326
327 \tikzset{tw@roundedmenus@base/.style={%
328 tw@set@tikz@colors,
329 rounded corners=0.3ex,
330 inner sep=0pt,
331 inner xsep=2pt,
332 text height=1.825ex,
333 text depth=0.7ex,
334 minimum width=1.5em,
335 font=\relsize{-1}\sffamily,
336 signal,
337 signal to=nowhere,
338 signal pointer angle=110,
339 }}
340 \tw@declare@style*{roundedmenus}{%
341 \tikz[baseline={{$(tw@node.base)+(0,-0.2ex)$}}]{%
342 \node(tw@node)[tw@roundedmenus@base,signal to=east]%
343 {\strut\color{\usemenucolor{txt}}\CurrentMenuElement};}%
344 }[\hspace{-0.2em}\hspace{0em plus 0.1em minus 0.05em}]%
345 {%
346 \tikz[baseline={{$(tw@node.base)+(0,-0.2ex)$}}]{%
347 \node(tw@node)[tw@roundedmenus@base,signal from=west,signal to=east]%
348 {\strut\color{\usemenucolor{txt}}\CurrentMenuElement};}%
349 }{%
350 \tikz[baseline={{$(tw@node.base)+(0,-0.2ex)$}}]{%
351 \node(tw@node)[tw@roundedmenus@base,signal from=west,]%
352 {\strut\color{\usemenucolor{txt}}\CurrentMenuElement};}%
353 }{%
354 \tikz[baseline={{$(tw@node.base)+(0,-0.2ex)$}}]{%
355 \node(tw@node)[tw@roundedmenus@base]{\strut\color{\usemenucolor{txt}}\CurrentMenuElement};}%
356 }{gray}
357
358 \tikzset{tw@angularmenus@base/.style={%
359 tw@set@tikz@colors,
360 inner sep=0pt,
361 inner xsep=2pt,
362 text height=1.825ex,
363 text depth=0.7ex,
364 minimum width=1.5em,
365 font=\relsize{-1}\sffamily,
366 signal,
367 signal to=nowhere,
368 signal pointer angle=110,
369 }}
370 \tw@declare@style*{angularmenus}{%
371 \tikz[baseline={{$(tw@node.base)+(0,-0.2ex)$}}]{%
372 \node(tw@node)[tw@angularmenus@base,signal to=east]%

```



```

373     {\strut\color{\usemenucolor{txt}}\CurrentMenuElement};}%
374 }[\hspace{-0.2em}\hspace{0em plus 0.1em minus 0.05em}]%
375 {%
376   \tikz[baseline={{$(tw@node.base)+(0,-0.2ex)$}}]{%
377     \node(tw@node)[tw@angularmenus@base,signal from=west,signal to=east]%
378       {\strut\color{\usemenucolor{txt}}\CurrentMenuElement};}%
379 }{%
380   \tikz[baseline={{$(tw@node.base)+(0,-0.2ex)$}}]{%
381     \node(tw@node)[tw@angularmenus@base,signal from=west,]%
382       {\strut\color{\usemenucolor{txt}}\CurrentMenuElement};}%
383 }{%
384   \tikz[baseline={{$(tw@node.base)+(0,-0.2ex)$}}]{%
385     \node(tw@node)[tw@angularmenus@base]{\strut\color{\usemenucolor{txt}}\CurrentMenuElement};}%
386 }{gray}
387
388 \tikzset{tw@roundedkeys@base/.style={%
389   tw@set@tikz@colors,
390   rounded corners=0.3ex,
391   inner sep=0pt,
392   inner xsep=2pt,
393   text height=1.825ex,
394   text depth=0.7ex,
395   minimum width=1.5em,
396   font=\relsize{-1}\sffamily,
397 }}
398 \tw@declare@style@simple*{roundedkeys}{%
399   \tikz[baseline={{$(tw@node.base)+(0,-0.2ex)$}}]{%
400     \node(tw@node)[tw@roundedkeys@base]%
401       {\strut\color{\usemenucolor{txt}}\CurrentMenuElement};}%
402 }[%
403   \hspace{0.1em plus 0.1em minus 0.05em}%
404   \textcolor{\usemenucolor{b}}{\raisebox{0.25ex}{\sffamily\relsize{-2}+}}%
405   \hspace{0.1em plus 0.1em minus 0.05em}%
406 ]{gray}
407
408 \tikzset{tw@shadowedroundedkeys@base/.style={%
409   tw@set@tikz@colors,
410   rounded corners=0.3ex,
411   inner sep=0pt,
412   inner xsep=2pt,
413   text height=1.825ex,
414   text depth=0.7ex,
415   minimum width=1.5em,
416   font=\relsize{-1}\sffamily,
417   general shadow={%
418     shadow xshift=.2ex, shadow yshift=-.15ex,
419     fill=\usemenucolor{c},
420   },
421 }}
422 \tw@declare@style@simple*{shadowedroundedkeys}{%

```

```

423 \tikz[baseline={($ (tw@node.base)+(0,-0.2ex)$)]{%
424 \node(tw@node)[tw@shadowedroundedkeys@base]%
425 {\strut\color{\usemenucolor{txt}}\CurrentMenuElement};%
426 }%
427 }[%
428 \hspace{0.2ex}\hspace{0.1em plus 0.1em minus 0.05em}%
429 \textcolor{\usemenucolor{b}}{\raisebox{0.25ex}{\sffamily\relsize{-2}}}%
430 \hspace{0.1em plus 0.1em minus 0.05em}%
431 ][\hspace{0.2ex}]{gray}
432
433 \tikzset{tw@angularkeys@base/.style={%
434 tw@set@tikz@colors,
435 inner sep=0pt,
436 inner xsep=2pt,
437 text height=1.825ex,
438 text depth=0.7ex,
439 minimum width=1.5em,
440 font=\relsize{-1}\sffamily,
441 }}
442 \tw@declare@style@simple*{angularkeys}{%
443 \tikz[baseline={($ (tw@node.base)+(0,-0.2ex)$)]{%
444 \node(tw@node)[tw@angularkeys@base]%
445 {\strut\color{\usemenucolor{txt}}\CurrentMenuElement};%
446 }[%
447 \hspace{0.1em plus 0.1em minus 0.05em}%
448 \textcolor{\usemenucolor{b}}{\raisebox{0.25ex}{\sffamily\relsize{-2}}}%
449 \hspace{0.1em plus 0.1em minus 0.05em}%
450 ]{gray}
451
452 \tikzset{tw@shadowedangularkeys@base/.style={%
453 tw@set@tikz@colors,
454 inner sep=0pt,
455 inner xsep=2pt,
456 text height=1.825ex,
457 text depth=0.7ex,
458 minimum width=1.5em,
459 font=\relsize{-1}\sffamily,
460 general shadow={%
461 shadow xshift=.2ex, shadow yshift=-.15ex,
462 fill=\usemenucolor{c},
463 },
464 }}
465 \tw@declare@style@simple*{shadowedangularkeys}{%
466 \tikz[baseline={($ (tw@node.base)+(0,-0.2ex)$)]{%
467 \node(tw@node)[tw@shadowedangularkeys@base]%
468 {\strut\color{\usemenucolor{txt}}\CurrentMenuElement};%
469 }[%
470 \hspace{0.2ex}\hspace{0.1em plus 0.1em minus 0.05em}%
471 \textcolor{\usemenucolor{b}}{\raisebox{0.25ex}{\sffamily\relsize{-2}}}%
472 \hspace{0.1em plus 0.1em minus 0.05em}%

```

```

473 ][\hspace{0.2ex}]{gray}
474
475 \tikzset{tw@typewriterkeys@base/.style={%
476   tw@set@tikz@colors,
477   shape=circle,
478   minimum size=2ex,
479   inner sep=0.5pt, outer sep=1pt,
480   font=\ttfamily\relsize{-1},
481 }}
482 \tw@declare@style@simple*{typewriterkeys}{%
483   \def\tw@typewriterkeys@curr@elem{%
484     \maxsizebox*{2ex}{2ex}{\CurrentMenuElement}%
485   }%
486   \begin{tikzpicture}[baseline={($(\tw@node.south)+(0,0.8ex)$)}]%
487     \node(\tw@node)[%
488       tw@typewriterkeys@base, inner sep=1.25pt, line width=0.6pt%
489       ]{\color{\usemenucolor{txt}}\tw@typewriterkeys@curr@elem};
490     \node[tw@typewriterkeys@base]%
491       {\color{\usemenucolor{txt}}\tw@typewriterkeys@curr@elem};
492   \end{tikzpicture}%
493 }[%
494   \hspace{0.2ex}\hspace{0.1em plus 0.1em minus 0.05em}%
495   \textcolor{\usemenucolor{b}}{\raisebox{0.25ex}{\sffamily\relsize{-2}+}}%
496   \hspace{0.1em plus 0.1em minus 0.05em}%
497 ]{blacknwhite}
498
499 \tw@declare@style@simple*{paths}{%
500   {\ttfamily\color{\usemenucolor{txt}}\CurrentMenuElement}%
501 }[%
502   \hspace{0.2em plus 0.1em}%
503   \raisebox{0.08ex}{-%
504     \tikz{\fill[\usemenucolor{c}] (0,0) -- (0.5ex,0.5ex)%
505       -- (0,1ex) -- cycle;}%
506   }%
507   \hspace{0.2em plus 0.1em}%
508 ]{blacknwhite}
509
510 \newcounter{tw@hyphen@char@num}
511 \newif\if@tw@hyphen@tepaths@warnig
512 \@tw@hyphen@tepaths@warnigtrue
513 \tw@declare@style@simple*{hyphenatepaths}{%
514   {\ttfamily
515     \IfStrEq{T1}{\encodingdefault}{-%
516       \setcounter{tw@hyphen@char@num}{23}%
517     }-%
518     \IfStrEq{OT1}{\encodingdefault}{-%
519       \setcounter{tw@hyphen@char@num}{255}%
520     }-%
521     \if@tw@hyphen@tepaths@warnig%
522     \tw@mk@warning{The hyphenatepaths styles will probably only\MessageBreak

```

```

523         work with T1 or OT1 encoding.}%
524         \fi\global\@tw@hyphenatepaths@warnigfalse%
525     }%
526 }%
527 \hyphenchar\font=\value{tw@hyphen@char@num}\relax
528 \color{\usemenucolor{txt}}%
529 \CurrentMenuElement}%
530 }[%
531 \hspace{0.2em plus 0.1em}%
532 \raisebox{0.08ex}{%
533     \tikz{\fill[\usemenucolor{c}] (0,0) -- (0.5ex,0.5ex)%
534         -- (0,1ex) -- cycle;}%
535 }%
536 \hspace{0.2em plus 0.1em}%
537 }{blacknwhite}
538
539 \NewDocumentCommand{\drawtikzfolder}{0{white} 0{black}}{%
540     \begin{tikzpicture}[rounded corners=0.02ex,scale=0.7]
541         \draw [#2] (0,0) -- (1em,0) -- (1em,1.5ex) -- (0.5em,1.5ex) -- %
542             (0.4em,1.7ex) -- (0.1em,1.7ex) -- (0,1.5ex) -- cycle;
543         \draw [#2,fill=#1] (0,0) -- (1em,0) -- (0.85em,1.15ex) -- %
544             ++(-1em,0) -- cycle;
545     \end{tikzpicture}%
546 }
547
548 \copymenustyle{pathswithfolder}{paths}
549 \changemenuelement{pathswithfolder}{pre}{%
550     \drawtikzfolder[\usemenucolor{a}][\usemenucolor{b}]%
551     \hspace{0.2em plus 0.1em}%
552 }
553
554 \copymenustyle{pathswithblackfolder}{paths}
555 \changemenuelement{pathswithblackfolder}{pre}{%
556     \drawtikzfolder[\usemenucolor{c}][\usemenucolor{b}]%
557     \hspace{0.2em plus 0.1em}%
558 }
559
560 \copymenustyle{hyphenatepathswithfolder}{hyphenatepaths}
561 \changemenuelement{hyphenatepathswithfolder}{pre}{%
562     \drawtikzfolder[\usemenucolor{a}][\usemenucolor{b}]%
563     \hspace{0.2em plus 0.1em}%
564 }
565
566 \copymenustyle{hyphenatepathswithblackfolder}{hyphenatepaths}
567 \changemenuelement{hyphenatepathswithblackfolder}{pre}{%
568     \drawtikzfolder[\usemenucolor{c}][\usemenucolor{b}]%
569     \hspace{0.2em plus 0.1em}%
570 }

```

6.7 Menu macros

6.7.1 Internal commands

```
\tw@default@input@sep First we define our default input separator
571 \edef\tw@default@input@sep{,}

\CurrentMenuElement and the \CurrentMenuElement dummy
572 \def\CurrentMenuElement{}

\tw@define@menu@macro Then we set up the internal command to create new menu macros. The list parsing
code was essentially provided by Ahmed Musa at http://tex.stackexchange.com/a/44989/4918. Thank you very much!
573 \begingroup
574 \lccode'\,=1
575 \lowercase{\endgroup
576 \robust@def*\tw@mk@test@input@sep#1{%
577 \xifinsetTF{,\cpttrimspace{#1}},{\,bslash,backslash,directory,location,}%
578 }%
579 }
580 \NewDocumentCommand{\tw@define@menu@macro}{%
581 m O{\tw@default@input@sep} m
582 }{%
583 \ifcsundef{tw@style@#3@sep}{%
584 \tw@mk@error{Can't define menu macro \string#1\space,\MessageBreak
585 because the style '#3' is not available!}
586 }{%
587 \csdef{tw@parse@menu@list@\expandafter\@gobble\string#1}##1{%
588 \iflastindris
589 \ifnum\indrisnr=\@ne
590 \def\CurrentMenuElement{##1}%
591 \@nameuse{tw@style@#3@single}%
592 \else
593 \def\CurrentMenuElement{##1}%
594 \@nameuse{tw@style@#3@sep}\@nameuse{tw@style@#3@last}%
595 \fi
596 \else
597 \ifnum\indrisnr=\@ne
598 \def\CurrentMenuElement{##1}%
599 \@nameuse{tw@style@#3@first}%
600 \else
601 \def\CurrentMenuElement{##1}%
602 \@nameuse{tw@style@#3@sep}\@nameuse{tw@style@#3@mid}%
603 \fi
604 \fi
605 }%
606 \expandafter\newcommand\csname\expandafter\@gobble\string#1\endcsname[2][#2]{%
607 \leavevmode%
608 {\def\tw@current@color@theme{\csname tw@style@#3@color@theme\endcsname}%
609 \@nameuse{tw@style@#3@pre}%
```

```

610     \tw@mk@test@input@sep{##1}{%
611         \edef\tw@menu@list{\detokenize{##2}}\edef\tw@mk@tempa{\@backslashchar}%
612     }{%
613         \edef\tw@menu@list{\unexpanded{##2}}\edef\tw@mk@tempa{\cpttrimspaces{##1}}%
614     }%
615     {\letcs{\tw@mk@tempb}{tw@parse@menu@list@expandafter@gobble\string#1}%
616     \cptexpanded{\indrisloop*[\tw@mk@tempa]}\tw@menu@list\tw@mk@tempb}%
617     \@nameuse{tw@style@#3@post}}%
618 }%
619 }%
620 }
621 \edef\cpt@parserlist{\cpt@parserlist\@backslashchar}

```

6.7.2 User-level commands

```

\newmenumacro Now it's time to build the user-level commands
\renewmenumacro
\providemenumacro
622 \NewDocumentCommand{\newmenumacro}{m O{\tw@default@input@sep} m}{%
623     \ifcsundef{\expandafter@gobble\string#1}{%
624         \tw@define@menu@macro{#1}[#2]{#3}%
625         \expandafter\cptrobustify\csname\expandafter@gobble\string#1\endcsname
626     }{
627         \tw@mk@error{Menu macro '\string#1' already defined!\MessageBreak
628         Use \string\renewmenustyle\space instead.}
629     }%
630 }
631 \NewDocumentCommand{\renewmenumacro}{m O{\tw@default@input@sep} m}{%
632     \cslet{\expandafter@gobble\string#1}{\relax}%
633     \tw@define@menu@macro{#1}[#2]{#3}%
634 }
635 \NewDocumentCommand{\providemenumacro}{m O{\tw@default@input@sep} m}{%
636     \ifcsundef{\expandafter@gobble\string#1}{%
637         \tw@define@menu@macro{#1}[#2]{#3}%
638     }{
639         \tw@mk@warning{Menu macro '\string#1' already defined!\MessageBreak
640         Use \string\renewmenustyle\space to redefine it.}
641     }%
642 }

```

6.7.3 Predefined menu macros

Now we got all tools to predefine some menu macros. To be sure that these commands won't conflict with other packages we introduced the option `definemacros`. Here we have to check it:

```
643 \iftw@mk@definemacros
```

```

\menu And then we define three basic macros.
\directory 644 \newmenumacro{\menu}[>]{menus}
\keys 645 \newmenumacro{\directory}[/]{paths}
646 \newmenumacro{\keys}[+]{roundedkeys}

```

Lastly we close the `definemacros` if statement:

```
647 \fi
```

6.8 Keys

Before we define anything we check if the user allows it:

```
648 \iftw@mk@definekeys
```

Before define the key macros we create some macros that save some typing by condensing the similarities between the key macros.

`\tw@make@key@box` The first of these macros helps us building save boxes to store the `{tikzpicture}`, that will draw the key later. This is necessary because otherwise the picture will inherit the style of the key sequence `node`.

```
649 \NewDocumentCommand{\tw@make@key@box}{m m}{%
650 %   \expandafter\newbox\csname tw@mk@box@#1\endcsname
651 %   \expandafter\sbox\csname tw@mk@box@#1\endcsname{%
652 %     #2%
653 %   }%
654   \csdef{tw@mk@#1}{%
655 %     \expandafter\usebox\csname tw@mk@box@#1\endcsname%
656 %     #2%
657 %   }%
658 }
```

`\tw@make@key@macro` The next macro defines the user level command by accessing a macro like `tw@mk@<key>` or `tw@mk@<key>@<os>`, if the appearance differs between Mac and Windows. To use this macro we assume that the `tw@mk@<key>` commands are defined.

```
659 \NewDocumentCommand{\tw@make@key@macro}{s m}{%
660   \IfBooleanTF{#1}{%
661     \expandafter\providecommand\csname\expandafter@gobble\string#2\endcsname{%
662       \expandonce{\maxsizebox!}{1.8ex}{%
663         \@nameuse{tw@mk@\expandafter@gobble\string#2@tw@mk@os}}%
664     }%
665   }%
666   \expandafter\providecommand\csname\expandafter@gobble\string#2mac\endcsname{%
667     \expandonce{\maxsizebox!}{1.8ex}{%
668       \@nameuse{tw@mk@\expandafter@gobble\string#2@mac}}%
669   }%
670 }%
671   \expandafter\providecommand\csname\expandafter@gobble\string#2win\endcsname{%
672     \expandonce{\maxsizebox!}{1.8ex}{%
673       \@nameuse{tw@mk@\expandafter@gobble\string#2@win}}%
674   }%
675 }%
676 }{%
677   \expandafter\providecommand\csname\expandafter@gobble\string#2\endcsname{%
678     \expandonce{\maxsizebox!}{1.8ex}{%
679       \@nameuse{tw@mk@\expandafter@gobble\string#2}}%
680   }%
681 }
```

```

680     }%
681   }%
682 }%
683 }

```

`\tw@define@mackey` The last helping macro is `\tw@define@mackey`. We use it to execute code depending on the `mackeys` option.

```

684 \newcommand*{\tw@define@mackey}[2]{%
685   \IfStrEq{text}{\tw@mk@mackeys}{#1}{%
686     \IfStrEq{symbols}{\tw@mk@mackeys}{#2}{}%
687   }%
688 }

```

Next thing to do is to set up some *TikZ*-styles.

```

689 \tikzset{
690   menukeys key symbol/.style={
691     rounded corners=0pt,
692     line width=0.1ex,
693     baseline={(0,0)},
694   },
695   menukeys thick/.style={line width=0.25ex},
696 }

```

Now we are prepared to generate the key macros. I will be nearly the same way for all keys. Step one is to build a `\tw@mk@<key>` macro and then we define the user-level command `\<key>`

`\shift`

```

697 \normalsize
698 \tw@make@key@box{shift}{%
699   \begin{tikzpicture}[yshift=-0.1ex,menukeys key symbol]
700     \draw (0.3ex,0) -- (1.1ex,0) -- (1.1ex,1.2ex) -- %
701           (1.5ex,1.2ex) -- (0.7ex,1.9ex) -- (-0.1ex,1.2ex) -- %
702           (0.3ex,1.2ex) -- cycle;
703   \end{tikzpicture}%
704 }
705 \tw@make@key@macro{\shift}

```

It's a little more complicated if the appearance should differ depending on the OS: The first step again is to define `\tw@mk@<key>@mac` and `\tw@mk@<key>@win`. And then use the starred version `\tw@make@key@macro*` which creates `\<key>` that depends on the `os` option, `\<key>mac` and `\<key>win`, that are not affected by `os`.

`\capslock`

```

706 \tw@make@key@box{capslock@mac}{%
707   \begin{tikzpicture}[yshift=-0.1ex,menukeys key symbol]
708     \draw (0.3ex,0.7ex) -- (1.1ex,0.7ex) -- (1.1ex,1.2ex) -- %
709           (1.5ex,1.2ex) -- (0.7ex,1.9ex) -- (-0.1ex,1.2ex) -- %
710           (0.3ex,1.2ex) -- cycle;
711     \draw (0.3ex,0) rectangle (1.1ex,0.4ex);

```



```

712 \end{tikzpicture}%
713 }
714 \tw@make@key@box{capslock@win}{%
715 \begin{tikzpicture}[yscale=-1,yshift=-1.8ex,menukeys key symbol]
716 \draw (0.3ex,0) -- (1.1ex,0) -- (1.1ex,1.2ex) -- %
717 (1.5ex,1.2ex) -- (0.7ex,1.9ex) -- (-0.1ex,1.2ex) -- %
718 (0.3ex,1.2ex) -- cycle;
719 \end{tikzpicture}%
720 }
721 \tw@make@key@macro*\capslock}

```

Here are the other macros:

`\tab`

```

722 \tw@make@key@box{tab@mac}{%
723 \begin{tikzpicture}[yshift=0.6ex,menukeys key symbol]
724 \draw [->] (0,0) -- (1em,0);
725 \draw (1em,-0.35ex) -- (1em,0.35ex);
726 \end{tikzpicture}%
727 }
728 \tw@make@key@box{tab@win}{%
729 \begin{tikzpicture}[yshift=0.1ex,menukeys key symbol]
730 \draw [->] (0.2em,0) -- (1.2em,0);
731 \draw (1.2em,-0.35ex) -- (1.2em,0.35ex);
732 \draw [<-] (0,1ex) -- (1em,1ex);
733 \draw (0,0.65ex) -- (0,1.35ex);
734 \end{tikzpicture}%
735 }
736 \tw@make@key@macro*\tab}

```

`\esc`

`\oldesc`

```

737 \def\tw@mk@esc@win{Esc}
738 \tw@define@mackey{%
739 \def\tw@mk@esc@mac{esc}
740 }{%
741 \tw@make@key@box{esc@mac}{%
742 \begin{tikzpicture}[yshift=-0.1ex,menukeys key symbol]
743 \draw [->] (0.5ex,0.5ex) -- ++(135:1.1ex);
744 \draw (0.5ex,0.5ex) ++(105:0.6ex) arc (105:-195:0.6ex);
745 \end{tikzpicture}%
746 }%
747 }
748 \tw@make@key@macro*\esc}
749 \def\tw@mk@oldesc@win{Esc}
750 \tw@define@mackey{%
751 \def\tw@mk@oldesc@mac{esc}
752 }{%
753 \tw@make@key@box{oldesc@mac}{%
754 \begin{tikzpicture}[yshift=-0.1ex,menukeys key symbol]
755 \draw [->] (0.5ex,0.5ex) -- ++(45:1.1ex);

```

```

756         \draw (0.5ex,0.5ex) ++(15:0.6ex) arc (15:-285:0.6ex);
757     \end{tikzpicture}%
758 }%
759 }
760 \tw@make@key@macro*{\oldesc}

\ctrl
761 \providecommand\ctrlname{Ctrl}
762 \def\tw@mk@ctrl@win{\ctrlname}
763 \def\tw@mk@ctrl@mac{ctrl}
764 \tw@make@key@macro*{\ctrl}

\Alt
\AltGr
765 \def\tw@mk@Alt@win{Alt}
766 \tw@define@mackey{%
767     \def\tw@mk@Alt@mac{alt}%
768 }{%
769     \tw@make@key@box{Alt@mac}{%
770         \begin{tikzpicture}[yshift=-0.1ex,menukeys key symbol]
771             \draw (0,1ex) -- (0.5ex,1ex) -- (1ex,0.3ex) -- (1.8ex,0.3ex);
772             \draw (0.8ex,1ex) -- (1.8ex,1ex);
773         \end{tikzpicture}%
774     }%
775 }
776 \tw@make@key@macro*{\Alt}
777 \providecommand*{\AltGr}{Alt\,Gr}

\cmd
778 \def\tw@mk@cmd@win{%
779     \tw@mk@warning{'\string\cmd' only for Mac!}%
780 }
781 \tw@define@mackey{%
782     \def\tw@mk@cmd@mac{cmd}%
783 }{%
784     \tw@make@key@box{cmd@mac}{%
785         \begin{tikzpicture}[yshift=-0.15ex,menukeys key symbol]
786             \draw (0.5ex,0.7ex) -- (0.5ex,1.25ex) arc (0:270:0.25ex) -- %
787                 (1.25ex,1ex) arc (-90:180:0.25ex) -- (1ex,0.25ex) %
788                 arc (-180:90:0.25ex) -- (0.25ex,0.5ex) arc (90:360:0.25ex) %
789                 -- cycle;
790         \end{tikzpicture}%
791     }%
792 }
793 \tw@make@key@macro*{\cmd}

\Space
\SPACE
794 \providecommand*{\Space}{\expandonce{\rule{3em}{0pt}}}
795 \newcommand{\spacename}{Space}
796 \providecommand*{\SPACE}{\expandonce{\rule{2em}{0pt}\spacename\rule{2em}{0pt}}}

```

```

\return
797 \tw@make@key@box{return@mac}{%
798   \begin{tikzpicture}[yshift=0.25ex,menukeys key symbol]
799     \draw [->, rounded corners=0.2ex] (1.25ex,1ex) -| %
800       (2ex,0) -- (0,0);
801   \end{tikzpicture}%
802 }
803 \tw@make@key@box{return@win}{%
804   \begin{tikzpicture}[menukeys key symbol]
805     \draw [->] (1ex,1.25ex) |- (0,0);
806   \end{tikzpicture}%
807 }
808 \tw@make@key@macro*{\return}

\enter
809 \def\tw@mk@enter@win{Enter}
810 \tw@make@key@box{enter@mac}{%
811   \begin{tikzpicture}[menukeys key symbol]
812     \draw (0,0) -- (0.5ex,0.5ex) -- (1ex,0);
813     \draw (0,0.55ex) -- (1ex,0.55ex);
814   \end{tikzpicture}%
815 }
816 \tw@make@key@macro*{\enter}

\winmenu
817 \def\tw@mk@winmenu@mac{%
818   \tw@mk@warning{'\string\winmenu' only for Windows!}%
819 }
820 \tw@make@key@box{winmenu@win}{%
821   \begin{tikzpicture}[yshift=-0.2ex,menukeys key symbol]
822     \draw (0,0) rectangle (1.5ex,1.8ex);
823     \draw (0.25ex,1.4ex) -- ++(1ex,0);
824     \draw (0.25ex,1ex) -- ++(1ex,0);
825     \draw (0.25ex,0.6ex) -- ++(1ex,0);
826   \end{tikzpicture}%
827 }
828 \tw@make@key@macro*{\winmenu}

\backspace
829 \tw@make@key@box{backspace}{%
830   \begin{tikzpicture}[yshift=0.65ex,menukeys key symbol]
831     \draw [<-,menukeys thick] (0,0) -- (1.35em,0);
832   \end{tikzpicture}%
833 }
834 \tw@make@key@macro*{\backspace}

\del
\backdel 835 \providecommand{\delname}{Del.}
836 \def\tw@mk@del@win{\delname}

```

```

837 \tw@define@mackey{%
838   \def\tw@mk@del@mac{\delname}%
839 }{%
840   \tw@make@key@box{\del@mac}{%
841     \begin{tikzpicture}[yshift=0.2ex,menukeys key symbol]
842       \draw (0,0) -- (1.5ex,0) -- (2ex,0.5ex) --%
843         (1.5ex,1ex) -- (0,1ex) -- cycle;
844       \draw (0.5ex,0.2ex) -- (1.1ex,0.8ex);
845       \draw (0.5ex,0.8ex) -- (1.1ex,0.2ex);
846     \end{tikzpicture}%
847   }%
848 }
849 \tw@make@key@macro*{\del}
850 \def\tw@mk@backdel@win{\delname}
851 \tw@define@mackey{%
852   \def\tw@mk@backdel@mac{\delname}%
853 }{%
854   \tw@make@key@box{\backdel@mac}{%
855     \begin{tikzpicture}[yshift=0.2ex,menukeys key symbol]
856       \draw (2ex,0) -- (0.5ex,0) -- (0,0.5ex) --%
857         (0.5ex,1ex) -- (2ex,1ex) -- cycle;
858       \draw (1ex,0.2ex) -- (1.6ex,0.8ex);
859       \draw (1ex,0.8ex) -- (1.6ex,0.2ex);
860     \end{tikzpicture}%
861   }%
862 }
863 \tw@make@key@macro*{\backdel}

```

\arrowkeyup Lastly we define the arrow macros:

```

\arrowkeydown 864 \tw@make@key@box{\arrowkeyup}{%
\arrowkeyleft 865   \begin{tikzpicture}[yshift=-0.2ex,menukeys key symbol]
\arrowkeyright 866     \draw [->] (0,0) -- (0,0.8em);
867     \end{tikzpicture}%
868   }
869 \tw@make@key@macro{\arrowkeyup}
870
871 \tw@make@key@box{\arrowkeydown}{%
872   \begin{tikzpicture}[yshift=0.7em,menukeys key symbol]
873     \draw [->] (0,0) -- (0,-0.8em);
874   \end{tikzpicture}%
875 }
876 \tw@make@key@macro{\arrowkeydown}
877
878 \tw@make@key@box{\arrowkeyright}{%
879   \begin{tikzpicture}[yshift=0.5ex,menukeys key symbol]
880     \draw [->] (0,0) -- (0.8em,0);
881   \end{tikzpicture}%
882 }
883 \tw@make@key@macro{\arrowkeyright}
884

```

```

885 \tw@make@key@box{arrowkeyleft}{%
886   \begin{tikzpicture}[yshift=0.5ex,menukeys key symbol]
887     \draw [->] (0,0) -- (-0.8em,0);
888   \end{tikzpicture}%
889 }
890 \tw@make@key@macro{\arrowkeyleft}

```

\arrowkey And the \arrowkey macro that get's it's direction as argument.

```

891 \newcommand{\arrowkey}[1]{%
892   \IfStrEq{^}{#1}{\arrowkeyup}{%
893     \IfStrEq{v}{#1}{\arrowkeydown}{%
894       \IfStrEq{<}{#1}{\arrowkeyleft}{%
895         \IfStrEq{>}{#1}{\arrowkeyright}{%
896           \tw@mk@error{Wrong value '#1' for \string\arrowkey\MessageBreak
897             Possible values are '^', 'v', '<' or '>'}%
898         }%
899       }%
900     }%
901   }%
902 }

```

Close the \iftw@mk@definekeys

```
903 \fi
```

7 Change history

v1.0	Tidy up version and date	1
General: Initial version		1
v1.1	General: Added braces to the	
<code>\directory</code> : Renamed <code>\path</code> to	<code>\tikz</code> macro since the parser	
<code>\directory</code> because it crashes	seems to crash with <code>babel</code> 's	
with <code>biblatex</code>	french option otherwise.	1
General: Improved manual	Replaced obsolete <code>\tikzstyle</code>	1
Load <code>xcolor</code> before <code>menukeys</code>		14
v1.1a	v1.2c	
<code>\newmenumacro</code> : Added a line to	<code>\tw@define@menu@macro</code> : Replaced	
make a new macro robust.	<code>\protected@edef</code> by <code>\def</code>	28
<code>\tw@define@menu@macro</code> : Fixed	v1.3	
minor bug, that causes a	General: Added TikZ-styles for the	
warning about robustifying	key symbols.	1
(issue #23), by deleting the	Improved key symbols.	1
line to make the command	v1.4	
robust.	<code>\backdel</code> : Added <code>\backdel</code>	28
v1.2	<code>\oldesc</code> : Fixed direction of	
<code>\tw@define@menu@macro</code> : Added	<code>\escmac</code> ; added <code>\oldesc</code>	32
<code>\leavevmode</code>	General: Extended color theme	
Replaced <code>\edef</code> by	features.	1
<code>\protected@edef</code>	The <code>path...</code> styles now use the	
General: Added <code>\normalsize</code>	text color of the selected color	
before symbol definitions to	theme (fix issue #16).	1
make the <code>ex</code> unit available	v1.5	
Added <code>\SPACE</code> and <code>\spacename</code>	General: New option	
Fixed GitHub issues #9, #10,	<code>hyperrefcolorlinks</code>	16
#11, #13, #17, #24 and #26		

8 Macro index

Numbers written in bold face refer to the page where the corresponding entry is described; italic numbers refer to the code line of the definition; numbers in roman refer to the code lines where the entry is used.

Symbols	
<code>\@pdfborder</code>	<code>angularkeys</code> (style) 6
<code>\@pdfborderstyle</code>	<code>angularmenus</code> (style) 5
<code>\@tw@hyphenatepaths@warnigfalse</code>	<code>\arrowkey</code> 13, 891
<code>\@tw@hyphenatepaths@warnigtrue</code>	<code>\arrowkeydown</code> <i>864, 893</i>
	<code>\arrowkeyleft</code> <i>864, 894</i>
A	<code>\arrowkeyright</code> <i>864, 895</i>
<code>\Alt</code>	<code>\arrowkeyup</code> <i>864, 892</i>
<code>\AltGr</code>	

B		H	
<code>\backdel</code>	835	<code>\Hy@AtBeginDocument</code>	45
<code>\backspace</code>	829	<code>\Hy@CatcodeWrapper</code>	50
<code>blacknwhite</code> (theme)	11	<code>\Hy@colorlink</code>	54
C		<code>\Hy@endcolorlink</code>	58
<code>\capslock</code>	706	<code>\Hy@Info</code>	59
<code>\changemenucolor</code>	12, 81	<code>\HyColor@UseColor</code>	56
<code>\changemenucolortheme</code>	10, 102	<code>hyperrefcolorlinks</code> (option)	4
<code>\changemenuelement</code>		<code>hyphenatepaths</code> (style)	8
.....	10, 271, 549, 555, 561, 567	<code>hyphenatepathswithblackfolder</code> (style)	8
<code>\cmd</code>	778	<code>hyphenatepathswithfolder</code> (style) ..	8
<code>\color</code>	284, 312, 317, 321, 324, 343, 348, 352, 355, 373, 378, 382, 385, 401, 425, 445, 468, 489, 491, 500, 528	I	
Color themes:		<code>\iftw@hyphenatepaths@warnig</code>	511, 521
<code>blacknwhite</code>	11	<code>\ifHy@typexml</code>	48
<code>gray</code>	11	<code>\iftw@mk@definekeys</code>	648
<code>\copymenucolortheme</code>	12, 81	<code>\iftw@mk@definenumacros</code>	643
<code>\copymenustyle</code>		<code>\iftw@mk@hyperrefcolorlinks</code>	44
.....	10, 253, 548, 554, 560, 566	K	
<code>\ctrl</code>	761	<code>\keys</code>	4, 644
<code>\ctrlname</code>	13, 761, 762	M	
<code>\CurrentMenuElement</code> .	10, 139, 142, 145, 148, 171, 174, 177, 180, 284, 312, 317, 321, 324, 343, 348, 352, 355, 373, 378, 382, 385, 401, 425, 445, 468, 484, 500, 529, 572, 590, 593, 598, 601	<code>mackeys</code> (option)	4, 13
D		<code>\menu</code>	4, 644
<code>definekeys</code> (option)	4	<code>menus</code> (style)	5
<code>definenumacros</code> (option)	4	N	
<code>\del</code>	835	<code>\newmenucolortheme</code> ..	11, 70, 117, 118
<code>\delname</code> ...	13, 835, 836, 838, 850, 852	<code>\newmenumacro</code> ..	12, 622, 644, 645, 646
<code>\directory</code>	4, 644	<code>\newmenustyle</code>	9, 184, 219
<code>\drawtikzfolder</code>		<code>\newmenustylesimple</code>	9, 184, 184
.....	9, 539, 550, 556, 562, 568	O	
E		<code>\oldesc</code>	737
<code>\enter</code>	809	Options:	
<code>\esc</code>	737	<code>definekeys</code>	4
F		<code>definenumacros</code>	4
<code>\font</code>	527	<code>hyperrefcolorlinks</code>	4
G		<code>mackeys</code>	4, 13
<code>gray</code> (theme)	11	<code>os</code>	4
		<code>os</code> (option)	4
		P	
		<code>paths</code> (style)	7
		<code>pathswithblackfolder</code> (style)	7
		<code>pathswithfolder</code> (style)	7
		<code>\providenumacro</code>	13, 622
		<code>\providemenustyle</code>	11, 184, 239
		<code>\providemenustylesimple</code>	11, 184, 204

R	
<code>\relsize</code>	304, 335, 365, 396, 404, 416, 429, 440, 448, 459, 471, 480, 495
<code>\renewmenucolortheme</code>	119, 125, 152
<code>\renewmenumacro</code>	119, 125, 158
<code>\renewmenustyle</code>	119, 125, 158
..	11, 184, 228, 232, 249, 628, 640
<code>\renewmenustylesimple</code>	11, 184, 193, 197, 214
<code>\return</code>	797
<code>roundedkeys</code> (style)	6
<code>roundedmenus</code> (style)	5
S	
<code>shadowedangularkeys</code> (style)	6
<code>shadowedroundedkeys</code> (style)	6
<code>\shift</code>	697
<code>\SPACE</code>	794
<code>\Space</code>	794
<code>\spacename</code>	13, 795, 796
Styles:	
<code>angularkeys</code>	6
<code>angularmenus</code>	5
<code>hyphenatepathswithblackfolder</code>	8
<code>hyphenatepathswithfolder</code>	8
<code>hyphenatepaths</code>	8
<code>menus</code>	5
<code>pathswithblackfolder</code>	7
<code>pathswithfolder</code>	7
<code>paths</code>	7
<code>roundedkeys</code>	6
<code>roundedmenus</code>	5
<code>shadowedangularkeys</code>	6
<code>shadowedroundedkeys</code>	6
<code>typewriterkeys</code>	7
T	
<code>\tab</code>	722
<code>\tikzset</code>	291, 296, 327, 358, 388, 408, 433, 452, 475, 689
<code>\tw@current@color@theme</code>	115, 608
<code>\tw@current@style</code>	154, 155, 160
<code>\tw@declare@style</code>	157, 222, 224, 234, 236, 242, 244, 309, 340, 370
<code>\tw@declare@style@simple</code>	124, 187, 189, 199, 201, 207, 209, 398, 422, 442, 465, 482, 499, 513
<code>\tw@declare@style@extra@args</code>	151
<code>\tw@default@input@sep</code>	571, 581, 622, 631, 635
<code>\tw@default@post</code>	119, 125, 152
<code>\tw@default@pre</code>	119, 125, 158
<code>\tw@default@sep</code>	119, 125, 158
<code>\tw@define@mackey</code>	684, 738, 750, 766, 781, 837, 851
<code>\tw@define@menu@macro</code>	573, 624, 633, 637
<code>\tw@make@color@theme</code>	62, 72, 79
<code>\tw@make@key@box</code>	649, 698, 706, 714, 722, 728, 741, 753, 769, 784, 797, 803, 810, 820, 829, 840, 854, 864, 871, 878, 885
<code>\tw@make@key@macro</code>	659, 705, 721, 736, 748, 760, 764, 776, 793, 808, 816, 828, 834, 849, 863, 869, 876, 883, 890
<code>\tw@menu@list</code>	611, 613, 616
<code>\tw@mk@Alt@mac</code>	767
<code>\tw@mk@Alt@win</code>	765
<code>\tw@mk@backdel@mac</code>	852
<code>\tw@mk@backdel@win</code>	850
<code>\tw@mk@cmd@mac</code>	782
<code>\tw@mk@cmd@win</code>	778
<code>\tw@mk@ctrl@mac</code>	763
<code>\tw@mk@ctrl@win</code>	762
<code>\tw@mk@del@mac</code>	838
<code>\tw@mk@del@win</code>	836
<code>\tw@mk@center@win</code>	809
<code>\tw@mk@error</code>	10, 37, 41, 74, 85, 98, 104, 108, 192, 227, 256, 268, 273, 287, 584, 627, 896
<code>\tw@mk@esc@mac</code>	739
<code>\tw@mk@esc@win</code>	737
<code>\tw@mk@gobble@args</code>	21, 194, 215, 229, 250
<code>\tw@mk@mackeys</code>	40, 685, 686
<code>\tw@mk@oldesc@mac</code>	751
<code>\tw@mk@oldesc@win</code>	749
<code>\tw@mk@os</code>	36, 663
<code>\tw@mk@tempa</code>	19, 22, 23, 611, 613, 616
<code>\tw@mk@tempb</code>	19, 615, 616
<code>\tw@mk@test@input@sep</code>	576, 610
<code>\tw@mk@warning</code>	10, 212, 247, 522, 639, 779, 818
<code>\tw@mk@warning@noline</code>	10
<code>\tw@mk@winmenu@mac</code>	817
<code>\tw@set@tikz@colors</code>	291

\tw@typewriterkeys@curr@elem . . .	373, 378, 382, 385, 401, 404,
. 483, 489, 491	419, 425, 429, 445, 448, 462,
typewriterkeys (style) 7	468, 471, 489, 491, 495, 500,
	504, 528, 533, 550, 556, 562, 568
U	
\usemenucolor 10 , <i>114</i> ,	
284, 292, 293, 294, 312, 317,	
321, 324, 343, 348, 352, 355,	
	W
	\winmenu <i>817</i>