

# MENUKEYS

Tobias Weh

`mail@tobiw.de`

`https://tobiw.de/en`

Jonathan P. Spratte

`jspratte@yahoo.de`

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

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

`macros › latex › contrib › menukeys`

2020/12/19 — v1.6.1

## 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 original list parsing code at <https://tex.stackexchange.com/a/44989/4918>.*

# Contents

<b>1</b>	<b>Introduction</b>	<b>4</b>
<b>2</b>	<b>Installation</b>	<b>4</b>
<b>3</b>	<b>Package loading and options</b>	<b>5</b>
<b>4</b>	<b>Usage</b>	<b>5</b>
4.1	Basics . . . . .	5
4.2	Styles . . . . .	6
4.2.1	Predefined styles . . . . .	6
4.2.2	Declaring styles . . . . .	10
4.2.3	Copying styles . . . . .	11
4.2.4	Changing styles . . . . .	11
4.3	Color themes . . . . .	12
4.3.1	Predefined themes . . . . .	12
4.3.2	Create a theme . . . . .	12
4.3.3	Copy a theme . . . . .	13
4.3.4	Change a theme . . . . .	13
4.4	Menu macros . . . . .	13
4.4.1	Predefined menu macros . . . . .	13
4.4.2	Defining or changing menu macros . . . . .	13
4.5	Keys . . . . .	14
<b>5</b>	<b>Known issues and bugs</b>	<b>15</b>
<b>6</b>	<b>Implementation</b>	<b>15</b>
6.1	Required packages . . . . .	15
6.2	Helper macros . . . . .	17
6.3	Options . . . . .	18
6.4	Workarounds . . . . .	18
6.4.1	hyperref's colorlinks option . . . . .	18
6.5	Color themes . . . . .	19
6.5.1	Internal commands . . . . .	19
6.5.2	User-level commands . . . . .	19
6.5.3	Predefined themes . . . . .	20
6.6	Styles . . . . .	20
6.6.1	Internal commands . . . . .	21
6.6.2	User-level commands . . . . .	22
6.6.3	Copying and changing . . . . .	23
6.6.4	Predefined styles . . . . .	24
6.7	Menu macros . . . . .	30
6.7.1	Internal commands . . . . .	30
6.7.2	User-level commands . . . . .	32
6.7.3	Predefined menu macros . . . . .	32
6.8	Keys . . . . .	33

7	Change history	40
8	Macro index	41

# 1 Introduction

The `menukeys` package is mainly designed to parse and print sequences of software menus, folders and files, or keystrokes. Most predefined styles use the power of `TikZ`<sup>1</sup> 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
```

---

<sup>1</sup> See <https://www.ctan.org/pkg/pgf>.

### 3 Package loading and options

Since `menukeys` used to use `catoptions`, which does some heavy changes on key-value options, it **was** recommended to load `menukeys` as the last package (even after `hyperref`<sup>2</sup>). **This is no longer the case!**

These are the possible options:

**definenumacros:** Most of `menukeys`' macros should not conflict with other packages<sup>3</sup> 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 `definenumacros=false`. The default value is `true`.

`definenumacros` (opt.)

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

**definekeys:** Equal to `definenumacros` 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:** *Obsolete* (see sec. 5 and 6.4.1).

## 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>}`.<sup>4</sup>


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

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

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

<sup>3</sup> If you find a conflict send an e-mail.

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

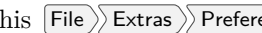
Name: `menu`



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<sub>E</sub>X 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<sub>E</sub>X 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<sup>A</sup>T<sub>E</sub>X 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<sup>A</sup>T<sub>E</sub>X-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>}{<br>}{<txt>}[<a>][<b>][<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<sup>A</sup>T<sub>E</sub>X 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<sup>A</sup>T<sub>E</sub>X 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{&gt;}</code>	→	→
<code>\Space</code>	[empty sp.]	[empty sp.]	<code>\arrowkeyright</code>	→	→
<code>\SPACE</code>	Space	Space	<code>\arrowkey{&lt;}</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` or directly to it via `\PassOptionsToPackage`.

**Example** Set `xcolor` to `cmym` model:

```
\documentclass{article}
\PassOptionsToPackage{cmym}{xcolor}
\usepackage{menukeys}
\begin{document}
  Hello World!
\end{document}
```

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 used to load `catoptions`. Instead we now use some `expl3` functions to replace the macros we required from `catoptions`.

The first few of these functions are more or less direct equivalents. A bit of attention has to be paid for `\tw@mk@xifinsetTF` as it requires the arguments to get swapped.

```

9 \ExplSyntaxOn
10 \cs_new_eq:NN \tw@mk@trimspaces \tl_trim_spaces:n
11 \cs_new_eq:NN \tw@mk@exp@Nnno \exp_args:Nnno
12 \cs_new_eq:NN \tw@mk@string \cs_to_str:N
13 \prg_generate_conditional_variant:Nnn \tl_if_in:nn { xx } { TF }
14 \cs_new:Npn \tw@mk@xifinsetTF #1 #2
15 {
16   \tl_if_in:xxTF {#2} {#1}
17 }

```

The replacement for `\indrisloop` will not set the conditional `\iflastindris`, instead we can check whether the sequence is empty to see whether this is the last element. This test will not use a TeX-like `\iflastindris... \else... \fi` construct but instead two branches.

```

18 \cs_new:Npn \tw@mk@iflastindris
19 {
20   \seq_if_empty:NTF \l__twmk_indris_seq
21 }

```

Replacing `\indrisloop` is a bit more work as it requires us to push some values to a stack (to allow nested usage, this may not be necessary for `menukeys`, but it is part of the original `\indrisloop` so we should play nice here). First we'll need a few variables.

```

22 \seq_new:N \l__twmk_indris_seq
23 \int_new:N \l__twmk_indris_int
24 \tl_new:N \l__twmk_indris_tl
25 \cs_new_eq:NN \tw@mk@indrisnr \l__twmk_indris_int
26 \seq_new:N \l__twmk_indris_seqstack_seq
27 \seq_new:N \l__twmk_indris_intstack_seq

```

Our stack will use another sequence in which the definitions of the parent call will be stored for the sequence and the integer. The other variables put on a stack by `\indrisloop` aren't required. The synopsis of `\tw@mk@indrisloop` will be different to the one provided by `catoptions`. The delimiter will be a mandatory argument (not in brackets), and there is no starred version.

```

28 \cs_new_protected:Npn \__twmk_pushseq:
29 {
30   \seq_push:No \l__twmk_indris_seqstack_seq \l__twmk_indris_seq
31 }
32 \cs_new_protected:Npn \__twmk_pushint:
33 {
34   \seq_push:NV \l__twmk_indris_intstack_seq \l__twmk_indris_int
35 }

```



```

36 \cs_new_protected:Npn \__twmk_popseq:
37   {
38     \seq_if_empty:NTF \l__twmk_indris_seqstack_seq
39     { \seq_clear:N \l__twmk_indris_seq }
40     { \seq_pop:NN \l__twmk_indris_seqstack_seq \l__twmk_indris_seq }
41   }
42 \cs_new_protected:Npn \__twmk_popint:
43   {
44     \seq_if_empty:NTF \l__twmk_indris_intstack_seq
45     { \int_zero:N \l__twmk_indris_int }
46     {
47       \group_begin:
48       \seq_pop:NN \l__twmk_indris_intstack_seq \l__twmk_indris_tl
49       \exp_args:NNNo
50       \group_end:
51       \int_set:Nn \l__twmk_indris_int \l__twmk_indris_tl
52     }
53   }

```

The real loop works by first splitting the input into a sequence according to the delimiter in #1. Then this sequence is stepped through, but instead of using `\seq_map:NN` we'll have to pop the sequence into a local variable so that our test for the last element works. The parameter #2 has to be expanded once as it is handed in as a token storing the real argument in later use.

```

54 \cs_generate_variant:Nn \seq_set_split:Nnn { Noo }
55 \cs_new_protected:Npn \tw@mk@indrisloop #1 #2 #3
56   {
57     \__twmk_pushseq:
58     \__twmk_pushint:
59     \seq_set_split:Noo \l__twmk_indris_seq {#1} {#2}
60     \int_zero:N \l__twmk_indris_int
61     \bool_do_while:nn { \bool_not_p:n { \seq_if_empty_p:N \l__twmk_indris_seq } }
62     {
63       \int_incr:N \l__twmk_indris_int
64       \seq_pop_left:NN \l__twmk_indris_seq \l__twmk_indris_tl
65       \exp_args:No #3 \l__twmk_indris_tl
66     }
67     \__twmk_popseq:
68     \__twmk_popint:
69   }
70 \ExplSyntaxOff

```

## 6.2 Helper macros

```

\tw@mk@error Define macros to call \PackageError and warnings
\tw@mk@warning 71 \newcommand*{\tw@mk@error}[2][Please consult the manual for more information.]{%
\tw@mk@warning@noline 72   \PackageError{menukeys}{#2}{#1}%
73 }
74 \newcommand*{\tw@mk@warning}[1]{%

```

```

75 \PackageWarning{menukeys}{#1}%
76 }
77 \newcommand*\tw@mk@warning@noline}[1]{%
78 \PackageWarningNoLine{menukeys}{#1}%
79 }

```

`\tw@mk@tempa` Some commands for temporary use:

```

\tw@mk@tempb 80 \def\tw@mk@tempa{}
              81 \def\tw@mk@tempb{}

```

`\tw@mk@gobble@args` Define a command to gobble arguments.

```

82 \DeclareDocumentCommand{\tw@mk@gobble@args}{m}{%
83 \RenewDocumentCommand{\tw@mk@tempa}{#1}{}%
84 \tw@mk@tempa%
85 }

```

## 6.3 Options

First we declare and process the package options

```

86 \RequirePackage{kvoptions}
87 \SetupKeyvalOptions{
88 family=tw@mk,
89 prefix=tw@mk@
90 }
91 \DeclareBoolOption[true]{definenummacros}
92 \DeclareBoolOption[true]{definekeys}
93 \DeclareBoolOption[false]{hyperrefcolorlinks}
94 \DeclareStringOption[mac]{os}
95 \DeclareStringOption[symbols]{mackeys}
96 \ProcessKeyvalOptions{tw@mk}\relax

```

Now we have to do some error treatment:

```

97 \IfSubStr{.mac.win.}{.\tw@mk@os.}{}{}%
98 \tw@mk@error{Unknown value for option 'os'\MessageBreak
99 Possible values are 'mac' or 'win'.}%
100 }
101 \IfSubStr{.symbols.text.}{.\tw@mk@mackeys.}{}{}%
102 \tw@mk@error{Unknown value for option 'mackeys'\MessageBreak
103 Possible values are 'symbols' or 'text'.}%
104 }

```

## 6.4 Workarounds

Some workarounds to “solve” some incompatibilities:

### 6.4.1 `hyperref`’s `colorlinks` option

There used to be an issue with using the `colorlinks` option of `hyperref` due to `catoptions` being loaded. Since `catoptions` isn’t required any more, this

workaround isn't necessary. For backwards compatibility the `hyperrefcolorlinks` option is still evaluated and just uses `\hypersetup` or `\PassOptionsToPackage` depending on whether `hyperref` is already loaded.

```

105 \iftw@mk@hyperrefcolorlinks
106   \tw@mk@warning{The option 'hyperrefcolorlinks' is obsolete}
107   \@ifpackageloaded{hyperref}
108     {\hypersetup{colorlinks}}
109     {\PassOptionsToPackage{colorlinks}{hyperref}}
110 \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

```

111 \newcommand*{\tw@make@color@theme}[8]{%
112   \definecolor{tw@color@theme@#1@bg}{#2}{#3}%
113   \definecolor{tw@color@theme@#1@br}{#2}{#4}%
114   \definecolor{tw@color@theme@#1@txt}{#2}{#5}%
115   \definecolor{tw@color@theme@#1@a}{#2}{#6}%
116   \definecolor{tw@color@theme@#1@b}{#2}{#7}%
117   \definecolor{tw@color@theme@#1@c}{#2}{#8}%
118 }

```

### 6.5.2 User-level commands

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

```

\renewmenucolortheme
119 \NewDocumentCommand{\newmenucolortheme}{ m m m m m O{#3} O{#4} O{#5} }{%
120   \@ifundefinedcolor{tw@color@theme@#1@bg}{%
121     \tw@make@color@theme{#1}{#2}{#3}{#4}{#5}{#6}{#7}{#8}%
122   }{%
123     \tw@mk@error{Color theme '#1' already defined!\MessageBreak
124       Use \string\renewmenucolortheme\space instead.}%
125   }
126 }
127 \NewDocumentCommand{\renewmenucolortheme}{ m m m m m O{#3} O{#4} O{#5} }{%
128   \tw@make@color@theme{#1}{#2}{#3}{#4}{#5}{#6}{#7}{#8}%
129 }

```

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

```

\copymenucolortheme
130 \newcommand*{\changemenucolor}[4]{%
131   \IfSubStr{ bg br txt }{ #2 }{%
132     \definecolor{tw@color@theme@#1@#2}{#3}{#4}%
133   }{%
134     \tw@mk@error{No such color element ('#2')!\MessageBreak
135       Possible values are bg, br and txt.}
136   }%
137 }
138 \newcommand*{\copymenucolortheme}[2]{%

```

```

139 \@ifundefinedcolor{tw@color@theme@#1@bg}{%
140   \colorlet{tw@color@theme@#1@bg}{tw@color@theme@#2@bg}%
141   \colorlet{tw@color@theme@#1@br}{tw@color@theme@#2@br}%
142   \colorlet{tw@color@theme@#1@txt}{tw@color@theme@#2@txt}%
143   \colorlet{tw@color@theme@#1@a}{tw@color@theme@#2@a}%
144   \colorlet{tw@color@theme@#1@b}{tw@color@theme@#2@b}%
145   \colorlet{tw@color@theme@#1@c}{tw@color@theme@#2@c}%
146 }{%
147   \tw@mk@error{Color theme '#1' already defined!\MessageBreak
148   Use \string\renewmenucolortheme\space instead.}
149 }
150 }

```

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

```

151 \newcommand{\changemenucolortheme}[2]{%
152   \ifcsundef{tw@style@#1@pre}{%
153     \tw@mk@error{Style '#1' undefined!\MessageBreak
154     Maybe you misspelled it?}%
155   }{%
156     \@ifundefinedcolor{tw@color@theme@#2@bg}{%
157       \tw@mk@error{Color theme '#2' is not defined!}%
158     }{%
159       \csdef{tw@style@#1@color@theme}{#2}%
160     }%
161   }%
162 }

```

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

```

163 \newcommand{\usemenucolor}[1]{%
164   tw@color@theme@\tw@current@color@theme @#1%
165 }

```

### 6.5.3 Predefined themes

There are two predefined color themes

```

166 \newmenucolortheme{gray}{gray}{0.95}{0.3}{0}{0.95}[0][0]
167 \newmenucolortheme{blackwhite}{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 commands, we have to set some defaults for the optional arguments

```

168 \newcommand{\tw@default@sep}{%
169   \hspace{0.2em plus 0.1em minus 0.5em}%
170 }
171 \newcommand{\tw@default@pre}{}
172 \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.
173 \DeclareDocumentCommand{\tw@declare@style@simple}{%
174   s m 0{\tw@default@pre} m 0{\tw@default@sep} 0{\tw@default@post} m
175 }{%
176   \csdef{tw@style@#2@color@theme}{#7}%
177   \csdef{tw@style@#2@pre}{#3}%
178   \csdef{tw@style@#2@sep}{#5}%
179   \csdef{tw@style@#2@post}{#6}%
180   \IfBooleanTF{#1}{%
181     \csdef{tw@style@#2@single}{#4}%
182     \csdef{tw@style@#2@first}{#4}%
183     \csdef{tw@style@#2@mid}{#4}%
184     \csdef{tw@style@#2@last}{#4}%
185   }{%
186     \csdef{tw@style@#2@single}{%
187       \tikz[baseline=(tw@node.base)]{%
188         \node(tw@node)[#4]{\strut\CurrentMenuElement};}%
189     \csdef{tw@style@#2@first}{%
190       \tikz[baseline=(tw@node.base)]{%
191         \node(tw@node)[#4]{\strut\CurrentMenuElement};}%
192     \csdef{tw@style@#2@mid}{%
193       \tikz[baseline=(tw@node.base)]{%
194         \node(tw@node)[#4]{\strut\CurrentMenuElement};}%
195     \csdef{tw@style@#2@last}{%
196       \tikz[baseline=(tw@node.base)]{%
197         \node(tw@node)[#4]{\strut\CurrentMenuElement};}%
198     }%
199 }

```

`\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.

```

\tw@declare@style@extra@args
200 \DeclareDocumentCommand{\tw@declare@style@extra@args}{%
201   0{\tw@default@post} m
202 }{%
203   \csdef{tw@style@\tw@current@style @post}{#1}%
204   \csdef{tw@style@\tw@current@style @color@theme}{#2}%
205 }

```

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

```

206 \DeclareDocumentCommand{\tw@declare@style}{%
207   s m 0{\tw@default@pre} m 0{\tw@default@sep} m m m
208 }{%
209   \def\tw@current@style{#2}
210   \csdef{tw@style@#2@pre}{#3}%
211   \csdef{tw@style@#2@sep}{#5}%

```

```

212 \IfBooleanTF{#1}{%
213   \csdef{tw@style@#2@single}{#8}%
214   \csdef{tw@style@#2@first}{#4}%
215   \csdef{tw@style@#2@mid}{#6}%
216   \csdef{tw@style@#2@last}{#7}%
217 }{%
218   \csdef{tw@style@#2@single}{%
219     \tikz[baseline=(tw@node.base)]{%
220       \node(tw@node)[#8]{\strut\CurrentMenuElement};}}%
221   \csdef{tw@style@#2@first}{%
222     \tikz[baseline=(tw@node.base)]{%
223       \node(tw@node)[#4]{\strut\CurrentMenuElement};}}%
224   \csdef{tw@style@#2@mid}{%
225     \tikz[baseline=(tw@node.base)]{%
226       \node(tw@node)[#6]{\strut\CurrentMenuElement};}}%
227   \csdef{tw@style@#2@last}{%
228     \tikz[baseline=(tw@node.base)]{%
229       \node(tw@node)[#7]{\strut\CurrentMenuElement};}}%
230 }%
231 \tw@declare@style@extra@args%
232 }

```

## 6.6.2 User-level commands

```

newmenustylesimple It's time to define the user-level commands now:
renewmenustylesimple 233 \NewDocumentCommand{\newmenustylesimple}{s m}{%
providemenustylesimple 234   \ifcsundef{tw@style@#2@pre}{%
  newmenustyle 235     \IfBooleanTF{#1}{%
    renewmenustyle 236       \tw@declare@style@simple*{#2}%
    providemenustyle 237       }{%
    238         \tw@declare@style@simple{#2}%
    239       }%
    240     }{%
    241       \tw@mk@error{Style '#2' already defined!\MessageBreak
    242         Use \string\renewmenustylesimple\space instead.}%
    243       \tw@mk@gobble@args{o m o m}%
    244     }%
    245   }
    246 \NewDocumentCommand{\renewmenustylesimple}{s m}{%
    247   \IfBooleanTF{#1}{%
    248     \tw@declare@style@simple*{#2}%
    249   }{%
    250     \tw@declare@style@simple{#2}%
    251   }%
    252 }
    253 \NewDocumentCommand{\providemenustylesimple}{s m}{%
    254   \ifcsundef{tw@style@#2@pre}{%
    255     \IfBooleanTF{#1}{%
    256       \tw@declare@style@simple*{#2}%

```

```

257     }{%
258         \tw@declare@style@simple{#2}%
259     }%
260 }{%
261     \tw@mk@warning{Trying to provide style '#2' failed,\MessageBreak
262     because it's already defined.\MessageBreak
263     You may use \string\renewmenustylesimple\space instead.}%
264     \tw@mk@gobble@args{o m o o m}%
265 }%
266 }
267
268 \NewDocumentCommand{\newmenustyle}{s m}{%
269     \ifcsundef{tw@style@#2@pre}{%
270         \IfBooleanTF{#1}{%
271             \tw@declare@style*{#2}%
272         }{%
273             \tw@declare@style{#2}%
274         }%
275     }{%
276         \tw@mk@error{Style '#2' already defined!\MessageBreak
277         Use \string\renewmenustyle\space instead.}%
278         \tw@mk@gobble@args{o m o m m o m}%
279     }%
280 }
281 \NewDocumentCommand{\renewmenustyle}{s m}{%
282     \IfBooleanTF{#1}{%
283         \tw@declare@style*{#2}%
284     }{%
285         \tw@declare@style{#2}%
286     }%
287 }
288 \NewDocumentCommand{\providemenustyle}{s m}{%
289     \ifcsundef{tw@style@#2@pre}{%
290         \IfBooleanTF{#1}{%
291             \tw@declare@style*{#2}%
292         }{%
293             \tw@declare@style{#2}%
294         }%
295     }{%
296         \tw@mk@warning{Trying to provide style #2 failed,\MessageBreak
297         because it's already defined.\MessageBreak
298         You may use \string\renewmenustyle\space instead.}%
299         \tw@mk@gobble@args{o m o m m o m}%
300     }%
301 }

```

### 6.6.3 Copying and changing

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

```

302 \newcommand*\copymenustyle}[2]{%
303   \ifcsundef{tw@style@#1@pre}{%
304     \ifcsundef{tw@style@#2@pre}{%
305       \tw@mk@error{Can't copy not existing style ('#2')}!}%
306     }{%
307       \csletcs{tw@style@#1@pre}{tw@style@#2@pre}%
308       \csletcs{tw@style@#1@post}{tw@style@#2@post}%
309       \csletcs{tw@style@#1@sep}{tw@style@#2@sep}%
310       \csletcs{tw@style@#1@single}{tw@style@#2@single}%
311       \csletcs{tw@style@#1@first}{tw@style@#2@first}%
312       \csletcs{tw@style@#1@mid}{tw@style@#2@mid}%
313       \csletcs{tw@style@#1@last}{tw@style@#2@last}%
314       \csletcs{tw@style@#1@color@theme}{tw@style@#2@color@theme}
315     }%
316   }{%
317     \tw@mk@error{Style '#1' already exists!}%
318   }%
319 }

```

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

```

320 \NewDocumentCommand{\changemenuelement}{s m m m}{%
321   \ifcsundef{tw@style@#2@pre}{%
322     \tw@mk@error{Style '#2' undefined.}%
323   }{%
324     \IfSubStr{ single first middle last pre post sep }{ #3 }{%
325       \IfBooleanTF{#1}{%
326         \csdef{tw@style@#2@#3}{#4}%
327       }{%
328         \IfSubStr{ pre post sep }{ #3 }{%
329           \csdef{tw@style@#2@#3}{#4}%
330         }{%
331           \csdef{tw@style@#2@#3}{%
332             \tikz[baseline=(tw@node.base)]{%
333               \node(tw@node)[#4]{\strut\color{\usemenucolor{txt}}\CurrentMenuElement};}%
334           }%
335         }%
336       }{\tw@mk@error{No element '#3'. Possible values are\MessageBreak
337         single, first, middle, last, pre, post or sep.}}%
338     }%
339 }

```

#### 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

```

340 \tikzset{tw@set@tikz@colors/.style={%
341   draw=\usemenucolor{br},
342   fill=\usemenucolor{bg},
343   text=\usemenucolor{txt},

```



344 }}

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.

```
345 \tikzset{tw@menus@base/.style={%
346   tw@set@tikz@colors,
347   rounded corners=0.15ex,
348   inner sep=0pt,
349   inner xsep=2pt,
350   text height=1.825ex,
351   text depth=0.7ex,
352   minimum width=1.5em,
353   font=\relsize{-1}\sffamily,
354   signal,
355   signal to=nowhere,
356   signal pointer angle=110,
357 }}
358 \tw@declare@style*{menus}{%
359   \tikz[baseline={{$(tw@node.base)+(0,-0.2ex)$}}]{%
360     \node(tw@node)[tw@menus@base,signal to=east]%
361       {\strut\color{\usemenucolor{txt}}\CurrentMenuElement};}%
362 }[\hspace{-0.2em}\hspace{0em plus 0.1em minus 0.05em}]{%
363 }{%
364   \tikz[baseline={{$(tw@node.base)+(0,-0.2ex)$}}]{%
365     \node(tw@node)[tw@menus@base,signal from=west,signal to=east]%
366       {\strut\color{\usemenucolor{txt}}\CurrentMenuElement};}%
367 }{%
368   \tikz[baseline={{$(tw@node.base)+(0,-0.2ex)$}}]{%
369     \node(tw@node)[tw@menus@base,signal from=west,]%
370       {\strut\color{\usemenucolor{txt}}\CurrentMenuElement};}%
371 }{%
372   \tikz[baseline={{$(tw@node.base)+(0,-0.2ex)$}}]{%
373     \node(tw@node)[tw@menus@base]{\strut\color{\usemenucolor{txt}}\CurrentMenuElement};}%
374 }{gray}
375
376 \tikzset{tw@roundedmenus@base/.style={%
377   tw@set@tikz@colors,
378   rounded corners=0.3ex,
379   inner sep=0pt,
380   inner xsep=2pt,
381   text height=1.825ex,
382   text depth=0.7ex,
383   minimum width=1.5em,
384   font=\relsize{-1}\sffamily,
385   signal,
386   signal to=nowhere,
387   signal pointer angle=110,
388 }}
389 \tw@declare@style*{roundedmenus}{%
390   \tikz[baseline={{$(tw@node.base)+(0,-0.2ex)$}}]{%
```

```

391     \node(tw@node)[tw@roundedmenus@base,signal to=east]%
392     {\strut\color{\usemenucolor{txt}}\CurrentMenuElement};}%
393 }[\hspace{-0.2em}\hspace{0em plus 0.1em minus 0.05em}]%
394 {%
395   \tikz[baseline={{$(tw@node.base)+(0,-0.2ex)$}}]{%
396     \node(tw@node)[tw@roundedmenus@base,signal from=west,signal to=east]%
397     {\strut\color{\usemenucolor{txt}}\CurrentMenuElement};}%
398 }{%
399   \tikz[baseline={{$(tw@node.base)+(0,-0.2ex)$}}]{%
400     \node(tw@node)[tw@roundedmenus@base,signal from=west,]%
401     {\strut\color{\usemenucolor{txt}}\CurrentMenuElement};}%
402 }{%
403   \tikz[baseline={{$(tw@node.base)+(0,-0.2ex)$}}]{%
404     \node(tw@node)[tw@roundedmenus@base]{\strut\color{\usemenucolor{txt}}\CurrentMenuElement.
405 }{gray}
406
407 \tikzset{tw@angularmenus@base/.style={%
408   tw@set@tikz@colors,
409   inner sep=0pt,
410   inner xsep=2pt,
411   text height=1.825ex,
412   text depth=0.7ex,
413   minimum width=1.5em,
414   font=\relsize{-1}\sffamily,
415   signal,
416   signal to=nowhere,
417   signal pointer angle=110,
418 }}
419 \tw@declare@style*{angularmenus}{%
420   \tikz[baseline={{$(tw@node.base)+(0,-0.2ex)$}}]{%
421     \node(tw@node)[tw@angularmenus@base,signal to=east]%
422     {\strut\color{\usemenucolor{txt}}\CurrentMenuElement};}%
423 }[\hspace{-0.2em}\hspace{0em plus 0.1em minus 0.05em}]%
424 {%
425   \tikz[baseline={{$(tw@node.base)+(0,-0.2ex)$}}]{%
426     \node(tw@node)[tw@angularmenus@base,signal from=west,signal to=east]%
427     {\strut\color{\usemenucolor{txt}}\CurrentMenuElement};}%
428 }{%
429   \tikz[baseline={{$(tw@node.base)+(0,-0.2ex)$}}]{%
430     \node(tw@node)[tw@angularmenus@base,signal from=west,]%
431     {\strut\color{\usemenucolor{txt}}\CurrentMenuElement};}%
432 }{%
433   \tikz[baseline={{$(tw@node.base)+(0,-0.2ex)$}}]{%
434     \node(tw@node)[tw@angularmenus@base]{\strut\color{\usemenucolor{txt}}\CurrentMenuElement.
435 }{gray}
436
437 \tikzset{tw@roundedkeys@base/.style={%
438   tw@set@tikz@colors,
439   rounded corners=0.3ex,
440   inner sep=0pt,

```

```

441   inner xsep=2pt,
442   text height=1.825ex,
443   text depth=0.7ex,
444   minimum width=1.5em,
445   font=\relsize{-1}\sffamily,
446 }}
447 \tw@declare@style@simple*{roundedkeys}{%
448   \tikz[baseline={($(\tw@node.base)+(0,-0.2ex)$)}]{%
449     \node(\tw@node)[tw@roundedkeys@base]%
450       {\strut\color{\usemenucolor{txt}}\CurrentMenuElement};%
451 }[%
452   \hspace{0.1em plus 0.1em minus 0.05em}%
453   \textcolor{\usemenucolor{b}}{\raisebox{0.25ex}{\sffamily\relsize{-2}+}}%
454   \hspace{0.1em plus 0.1em minus 0.05em}%
455 ]{gray}
456
457 \tikzset{tw@shadowedroundedkeys@base/.style={%
458   tw@set@tikz@colors,
459   rounded corners=0.3ex,
460   inner sep=0pt,
461   inner xsep=2pt,
462   text height=1.825ex,
463   text depth=0.7ex,
464   minimum width=1.5em,
465   font=\relsize{-1}\sffamily,
466   general shadow={%
467     shadow xshift=.2ex, shadow yshift=-.15ex,
468     fill=\usemenucolor{c},
469   },
470 }}
471 \tw@declare@style@simple*{shadowedroundedkeys}{%
472   \tikz[baseline={($(\tw@node.base)+(0,-0.2ex)$)}]{%
473     \node(\tw@node)[tw@shadowedroundedkeys@base]%
474       {\strut\color{\usemenucolor{txt}}\CurrentMenuElement};%
475   }%
476 }[%
477   \hspace{0.2ex}\hspace{0.1em plus 0.1em minus 0.05em}%
478   \textcolor{\usemenucolor{b}}{\raisebox{0.25ex}{\sffamily\relsize{-2}+}}%
479   \hspace{0.1em plus 0.1em minus 0.05em}%
480 ][\hspace{0.2ex}]{gray}
481
482 \tikzset{tw@angularkeys@base/.style={%
483   tw@set@tikz@colors,
484   inner sep=0pt,
485   inner xsep=2pt,
486   text height=1.825ex,
487   text depth=0.7ex,
488   minimum width=1.5em,
489   font=\relsize{-1}\sffamily,
490 }}

```

```

491 \tw@declare@style@simple*{angularkeys}{%
492   \tikz[baseline={{$(tw@node.base)+(0,-0.2ex)$}}]{%
493     \node(tw@node)[tw@angularkeys@base]%
494       {\strut\color{\usemenucolor{txt}}\CurrentMenuElement};}%
495 }[%
496   \hspace{0.1em plus 0.1em minus 0.05em}%
497   \textcolor{\usemenucolor{b}}{\raisebox{0.25ex}{\sffamily\relsize{-2}}}%
498   \hspace{0.1em plus 0.1em minus 0.05em}%
499 ]{gray}
500
501 \tikzset{tw@shadowedangularkeys@base/.style={%
502   tw@set@tikz@colors,
503   inner sep=0pt,
504   inner xsep=2pt,
505   text height=1.825ex,
506   text depth=0.7ex,
507   minimum width=1.5em,
508   font=\relsize{-1}\sffamily,
509   general shadow={%
510     shadow xshift=.2ex, shadow yshift=-.15ex,
511     fill=\usemenucolor{c},
512   },
513 }}
514 \tw@declare@style@simple*{shadowedangularkeys}{%
515   \tikz[baseline={{$(tw@node.base)+(0,-0.2ex)$}}]{%
516     \node(tw@node)[tw@shadowedangularkeys@base]%
517       {\strut\color{\usemenucolor{txt}}\CurrentMenuElement};}%
518 }[%
519   \hspace{0.2ex}\hspace{0.1em plus 0.1em minus 0.05em}%
520   \textcolor{\usemenucolor{b}}{\raisebox{0.25ex}{\sffamily\relsize{-2}}}%
521   \hspace{0.1em plus 0.1em minus 0.05em}%
522 ][\hspace{0.2ex}]{gray}
523
524 \tikzset{tw@typewriterkeys@base/.style={%
525   tw@set@tikz@colors,
526   shape=circle,
527   minimum size=2ex,
528   inner sep=0.5pt, outer sep=1pt,
529   font=\ttfamily\relsize{-1},
530 }}
531 \tw@declare@style@simple*{typewriterkeys}{%
532   \def\tw@typewriterkeys@curr@elem{%
533     \maxsizebox*{2ex}{2ex}{\CurrentMenuElement}%
534   }%
535   \begin{tikzpicture}[baseline={{$(tw@node.south)+(0,0.8ex)$}}]{%
536     \node(tw@node)[%
537       tw@typewriterkeys@base, inner sep=1.25pt, line width=0.6pt%
538     ]{\color{\usemenucolor{txt}}\tw@typewriterkeys@curr@elem};
539     \node[tw@typewriterkeys@base]%
540       {\color{\usemenucolor{txt}}\tw@typewriterkeys@curr@elem};

```

```

541 \end{tikzpicture}%
542 }[%
543 \hspace{0.2ex}\hspace{0.1em plus 0.1em minus 0.05em}%
544 \textcolor{\usemenucolor{b}}{\raisebox{0.25ex}{\sffamily\relsize{-2}}}%
545 \hspace{0.1em plus 0.1em minus 0.05em}%
546 ]{blacknwhite}
547
548 \tw@declare@style@simple*{paths}{%
549 {\ttfamily\color{\usemenucolor{txt}}\CurrentMenuElement}%
550 }[%
551 \hspace{0.2em plus 0.1em}%
552 \raisebox{0.08ex}{%
553 \tikz{\fill[\usemenucolor{c}] (0,0) -- (0.5ex,0.5ex)%
554 -- (0,1ex) -- cycle;}%
555 }%
556 \hspace{0.2em plus 0.1em}%
557 ]{blacknwhite}
558
559 \newcounter{tw@hyphen@char@num}
560 \newif\if@tw@hyphenatepaths@warnig
561 \@tw@hyphenatepaths@warnigtrue
562 \tw@declare@style@simple*{hyphenatepaths}{%
563 {\ttfamily
564 \IfStrEq{T1}{\encodingdefault}{%
565 \setcounter{tw@hyphen@char@num}{23}%
566 }{%
567 \IfStrEq{OT1}{\encodingdefault}{%
568 \setcounter{tw@hyphen@char@num}{255}%
569 }{%
570 \if@tw@hyphenatepaths@warnig%
571 \tw@mk@warning{The hyphenatepaths styles will probably only\MessageBreak
572 work with T1 or OT1 encoding.}%
573 \fi\global\@tw@hyphenatepaths@warnigfalse%
574 }%
575 }%
576 \hyphenchar\font=\value{tw@hyphen@char@num}\relax
577 \color{\usemenucolor{txt}}}%
578 \CurrentMenuElement}%
579 }[%
580 \hspace{0.2em plus 0.1em}%
581 \raisebox{0.08ex}{%
582 \tikz{\fill[\usemenucolor{c}] (0,0) -- (0.5ex,0.5ex)%
583 -- (0,1ex) -- cycle;}%
584 }%
585 \hspace{0.2em plus 0.1em}%
586 ]{blacknwhite}
587
588 \NewDocumentCommand{\drawtikzfolder}{0{white} 0{black}}{%
589 \begin{tikzpicture}[rounded corners=0.02ex,scale=0.7]
590 \draw [#2] (0,0) -- (1em,0) -- (1em,1.5ex) -- (0.5em,1.5ex) -- %

```

```

591         (0.4em,1.7ex) -- (0.1em,1.7ex) -- (0,1.5ex) -- cycle;
592     \draw [#2,fill=#1] (0,0) -- (1em,0) -- (0.85em,1.15ex) -- %
593         ++(-1em,0) -- cycle;
594 \end{tikzpicture}%
595 }
596
597 \copymenustyle{pathswithfolder}{paths}
598 \changemenuelement{pathswithfolder}{pre}{%
599     \drawtikzfolder[\usemenucolor{a}][\usemenucolor{b}]%
600     \hspace{0.2em plus 0.1em}%
601 }
602
603 \copymenustyle{pathswithblackfolder}{paths}
604 \changemenuelement{pathswithblackfolder}{pre}{%
605     \drawtikzfolder[\usemenucolor{c}][\usemenucolor{b}]%
606     \hspace{0.2em plus 0.1em}%
607 }
608
609 \copymenustyle{hyphenatepathswithfolder}{hyphenatepaths}
610 \changemenuelement{hyphenatepathswithfolder}{pre}{%
611     \drawtikzfolder[\usemenucolor{a}][\usemenucolor{b}]%
612     \hspace{0.2em plus 0.1em}%
613 }
614
615 \copymenustyle{hyphenatepathswithblackfolder}{hyphenatepaths}
616 \changemenuelement{hyphenatepathswithblackfolder}{pre}{%
617     \drawtikzfolder[\usemenucolor{c}][\usemenucolor{b}]%
618     \hspace{0.2em plus 0.1em}%
619 }

```

## 6.7 Menu macros

### 6.7.1 Internal commands

```

\tw@default@input@sep First we define our default input separator
620 \def\tw@default@input@sep{,}

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

\tw@define@menu@macro Then we set up the internal command to create new menu macros. The list parsing
\tw@define@menu@macro@ code was essentially provided by Ahmed Musa at https://tex.stackexchange.com/a/44989/4918. Jonathan P. Spratte made some major changes to make menukeys work without catoptions and reimplemented the parsing code using LATEX3. Thank you both very much!
622 \begingroup
623 \lccode'\,=1
624 \lowercase{\endgroup
625     \@ifdefinable\tw@mk@test@input@sep

```

```

626   {%
627     \protected\def\tw@mk@test@input@sep#1{%
628       \tw@mk@xifinsetTF
629       {,\tw@mk@trimspaces{#1},}{,bslash,backslash,directory,location,}%
630     }%
631   }%
632 }
633 \newcommand\tw@define@menu@macro[3]
634   {%
635     \IfNoValueTF{#3}
636     {\tw@mk@exp@Nnno\tw@define@menu@macro@{#1}{#2}\tw@default@input@sep}
637     {\tw@define@menu@macro@{#1}{#2}{#3}}%
638   }
639 \newcommand\tw@define@menu@macro@[4]
640   {%
641     \ifcsundef{tw@style@#4@sep}
642     {%
643       \tw@mk@error
644       {%
645         Can't define menu macro \string#2\space,\MessageBreak
646         because the style '#4' is not available!%
647       }%
648     }
649     {%
650       \csdef{tw@parse@menu@list@\tw@mk@string#2}##1%
651       {%
652         \def\CurrentMenuElement{##1}%
653         \tw@mk@iflastindris
654         {%
655           \ifnum\tw@mk@indrisnr=\@ne
656             \@nameuse{tw@style@#4@single}%
657           \else
658             \@nameuse{tw@style@#4@sep}%
659             \@nameuse{tw@style@#4@last}%
660           \fi
661         }
662         {%
663           \ifnum\tw@mk@indrisnr=\@ne
664             \@nameuse{tw@style@#4@first}%
665           \else
666             \@nameuse{tw@style@#4@sep}%
667             \@nameuse{tw@style@#4@mid}%
668           \fi
669         }%
670       }%
671       #1#2{+0{#3}+m}%
672     }%
673     \leavevmode
674     \begingroup
675     \def\tw@current@color@theme

```

```

676         {\csname tw@style@#4@color@theme\endcsname}%
677         \@nameuse{tw@style@#4@pre}%
678         \tw@mk@test@input@sep{##1}
679         {%
680         \edef\tw@menu@list{\detokenize{##2}}%
681         \edef\tw@mk@tempa{\@backslashchar}%
682         }
683         {%
684         \edef\tw@menu@list{\unexpanded{##2}}%
685         \edef\tw@mk@tempa{\tw@mk@trimspaces{##1}}%
686         }%
687         \begingroup
688         \letcs{\tw@mk@tempb}{tw@parse@menu@list@\tw@mk@string#2}%
689         \tw@mk@indrisloop\tw@mk@tempa\tw@menu@list\tw@mk@tempb
690         \endgroup
691         \@nameuse{tw@style@#4@post}%
692     \endgroup
693 }%
694 }%
695 }

```

### 6.7.2 User-level commands

```

\newmenumacro Now it's time to build the user-level commands
\renewmenumacro 696 \NewDocumentCommand{\newmenumacro}{m o m}{%
\providemenumacro 697 \ifcsundef{\tw@mk@string#1}{%
698 \tw@define@menu@macro\NewDocumentCommand{#1}{#2}{#3}%
699 }{%
700 \tw@mk@error{Menu macro '\string#1' already defined!\MessageBreak
701 Use \string\renewmenustyle\space instead.}%
702 }%
703 }
704 \NewDocumentCommand{\renewmenumacro}{m o m}{%
705 \tw@define@menu@macro\RenewDocumentCommand{#1}{#2}{#3}%
706 }
707 \NewDocumentCommand{\providemenumacro}{m o m}{%
708 \ifcsundef{\tw@mk@string#1}{%
709 \tw@define@menu@macro\ProvideDocumentCommand{#1}{#2}{#3}%
710 }{%
711 \tw@mk@warning{Menu macro '\string#1' already defined!\MessageBreak
712 Use \string\renewmenustyle\space to redefine it.}%
713 }%
714 }

```

### 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:



```
715 \iftw@mk@definenumacros
```

```
\menu And then we define three basic macros.
\directory 716 \newenumacro{\menu}[>]{menus}
\keys 717 \newenumacro{\directory}[/{]{paths}
718 \newenumacro{\keys}[+]{roundedkeys}
```

Lastly we close the `definmacros` if statement:

```
719 \fi
```

## 6.8 Keys

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

```
720 \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`.

```
721 \NewDocumentCommand{\tw@make@key@box}{m m}{%
722 % \expandafter\newbox\csname tw@mk@box@#1\endcsname
723 % \expandafter\savebox\csname tw@mk@box@#1\endcsname{%
724 % #2%
725 % }%
726 \csdef{tw@mk@#1}{%
727 % \expandafter\usebox\csname tw@mk@box@#1\endcsname%
728 % #2%
729 % }%
730 }
```

`\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.

```
731 \NewDocumentCommand{\tw@make@key@macro}{s m}{%
732 \IfBooleanTF{#1}{%
733 \expandafter\providecommand\csname\tw@mk@string#2\endcsname{%
734 \expandonce{\maxsizebox{!}{1.8ex}}{%
735 \@nameuse{tw@mk@\tw@mk@string#2@\tw@mk@os}}%
736 }%
737 }%
738 \expandafter\providecommand\csname\tw@mk@string#2mac\endcsname{%
739 \expandonce{\maxsizebox{!}{1.8ex}}{%
740 \@nameuse{tw@mk@\tw@mk@string#2mac}}%
741 }%
742 }%
743 \expandafter\providecommand\csname\tw@mk@string#2win\endcsname{%
744 \expandonce{\maxsizebox{!}{1.8ex}}{%
```

```

745         \@nameuse{tw@mk@tw@mk@string#2@win}}%
746     }%
747 }%
748 }{%
749     \expandafter\providecommand\csname\tw@mk@string#2\endcsname{%
750         \expandonce{\maxsizebox{!}{1.8ex}}{%
751             \@nameuse{tw@mk@tw@mk@string#2}}%
752         }%
753     }%
754 }%
755 }

```

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

```

756 \newcommand*{\tw@define@mackey}[2]{%
757     \IfStrEq{text}{\tw@mk@mackeys}{#1}{%
758         \IfStrEq{symbols}{\tw@mk@mackeys}{#2}{%
759             }%
760 }

```

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

```

761 \tikzset{
762     menukeys key symbol/.style={
763         rounded corners=0pt,
764         line width=0.1ex,
765         baseline={(0,0)},
766     },
767     menukeys thick/.style={line width=0.25ex},
768 }

```

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
769 \normalsize
770 \tw@make@key@box{shift}{%
771     \begin{tikzpicture}[yshift=-0.1ex,menukeys key symbol]
772         \draw (0.3ex,0) -- (1.1ex,0) -- (1.1ex,1.2ex) -- %
773             (1.5ex,1.2ex) -- (0.7ex,1.9ex) -- (-0.1ex,1.2ex) -- %
774             (0.3ex,1.2ex) -- cycle;
775     \end{tikzpicture}%
776 }
777 \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

```
778 \tw@make@key@box{capslock@mac}{%
779   \begin{tikzpicture}[yshift=-0.1ex,menukeys key symbol]
780     \draw (0.3ex,0.7ex) -- (1.1ex,0.7ex) -- (1.1ex,1.2ex) -- %
781           (1.5ex,1.2ex) -- (0.7ex,1.9ex) -- (-0.1ex,1.2ex) -- %
782           (0.3ex,1.2ex) -- cycle;
783     \draw (0.3ex,0) rectangle (1.1ex,0.4ex);
784   \end{tikzpicture}%
785 }
786 \tw@make@key@box{capslock@win}{%
787   \begin{tikzpicture}[yscale=-1,yshift=-1.8ex,menukeys key symbol]
788     \draw (0.3ex,0) -- (1.1ex,0) -- (1.1ex,1.2ex) -- %
789           (1.5ex,1.2ex) -- (0.7ex,1.9ex) -- (-0.1ex,1.2ex) -- %
790           (0.3ex,1.2ex) -- cycle;
791   \end{tikzpicture}%
792 }
793 \tw@make@key@macro*{\capslock}
```

Here are the other macros:

\tab

```
794 \tw@make@key@box{tab@mac}{%
795   \begin{tikzpicture}[yshift=0.6ex,menukeys key symbol]
796     \draw [->] (0,0) -- (1em,0);
797     \draw (1em,-0.35ex) -- (1em,0.35ex);
798   \end{tikzpicture}%
799 }
800 \tw@make@key@box{tab@win}{%
801   \begin{tikzpicture}[yshift=0.1ex,menukeys key symbol]
802     \draw [->] (0.2em,0) -- (1.2em,0);
803     \draw (1.2em,-0.35ex) -- (1.2em,0.35ex);
804     \draw [<-] (0,1ex) -- (1em,1ex);
805     \draw (0,0.65ex) -- (0,1.35ex);
806   \end{tikzpicture}%
807 }
808 \tw@make@key@macro*{\tab}
```

\esc

\oldesc

```
809 \def\tw@mk@esc@win{Esc}
810 \tw@define@mackey{%
811   \def\tw@mk@esc@mac{esc}
812 }{%
813   \tw@make@key@box{esc@mac}{%
814     \begin{tikzpicture}[yshift=-0.1ex,menukeys key symbol]
815       \draw [->] (0.5ex,0.5ex) -- ++(135:1.1ex);
816       \draw (0.5ex,0.5ex) ++(105:0.6ex) arc (105:-195:0.6ex);
817     \end{tikzpicture}%
818   }%
819 }
820 \tw@make@key@macro*{\esc}
```

```

821 \def\tw@mk@oldesc@win{Esc}
822 \tw@define@mackey{%
823   \def\tw@mk@oldesc@mac{esc}
824 }{%
825   \tw@make@key@box{oldesc@mac}{%
826     \begin{tikzpicture}[yshift=-0.1ex,menukeys key symbol]
827       \draw [->] (0.5ex,0.5ex) -- ++(45:1.1ex);
828       \draw (0.5ex,0.5ex) ++(15:0.6ex) arc (15:-285:0.6ex);
829     \end{tikzpicture}%
830   }%
831 }
832 \tw@make@key@macro*{\oldesc}

\ctrl
833 \providecommand\ctrlname{Ctrl}
834 \def\tw@mk@ctrl@win{\ctrlname}
835 \def\tw@mk@ctrl@mac{ctrl}
836 \tw@make@key@macro*{\ctrl}

\Alt
\AltGr 837 \def\tw@mk@Alt@win{Alt}
838 \tw@define@mackey{%
839   \def\tw@mk@Alt@mac{alt}%
840 }{%
841   \tw@make@key@box{Alt@mac}{%
842     \begin{tikzpicture}[yshift=-0.1ex,menukeys key symbol]
843       \draw (0,1ex) -- (0.5ex,1ex) -- (1ex,0.3ex) -- (1.8ex,0.3ex);
844       \draw (0.8ex,1ex) -- (1.8ex,1ex);
845     \end{tikzpicture}%
846   }%
847 }
848 \tw@make@key@macro*{\Alt}
849 \providecommand*{\AltGr}{Alt\,Gr}

\cmd
850 \def\tw@mk@cmd@win{%
851   \tw@mk@warning{'\string\cmd' only for Mac!}%
852 }
853 \tw@define@mackey{%
854   \def\tw@mk@cmd@mac{cmd}%
855 }{%
856   \tw@make@key@box{cmd@mac}{%
857     \begin{tikzpicture}[yshift=-0.15ex,menukeys key symbol]
858       \draw (0.5ex,0.7ex) -- (0.5ex,1.25ex) arc (0:270:0.25ex) -- %
859         (1.25ex,1ex) arc (-90:180:0.25ex) -- (1ex,0.25ex) %
860         arc (-180:90:0.25ex) -- (0.25ex,0.5ex) arc (90:360:0.25ex) %
861         -- cycle;
862     \end{tikzpicture}%
863   }%

```

```

864 }
865 \tw@make@key@macro*{\cmd}

\Space
\SPACE 866 \providecommand*{\Space}{\expandonce{\rule{3em}{0pt}}}
867 \newcommand{\spacename}{Space}
868 \providecommand*{\SPACE}{\expandonce{\rule{2em}{0pt}\spacename\rule{2em}{0pt}}}

\return
869 \tw@make@key@box{return@mac}{%
870   \begin{tikzpicture}[yshift=0.25ex,menukeys key symbol]
871     \draw [->, rounded corners=0.2ex] (1.25ex,1ex) -| %
872       (2ex,0) -- (0,0);
873   \end{tikzpicture}%
874 }
875 \tw@make@key@box{return@win}{%
876   \begin{tikzpicture}[menukeys key symbol]
877     \draw [->] (1ex,1.25ex) |- (0,0);
878   \end{tikzpicture}%
879 }
880 \tw@make@key@macro*{\return}

\enter
881 \def\tw@mk@enter@win{Enter}
882 \tw@make@key@box{enter@mac}{%
883   \begin{tikzpicture}[menukeys key symbol]
884     \draw (0,0) -- (0.5ex,0.5ex) -- (1ex,0);
885     \draw (0,0.55ex) -- (1ex,0.55ex);
886   \end{tikzpicture}%
887 }
888 \tw@make@key@macro*{\enter}

\winmenu
889 \def\tw@mk@winmenu@mac{%
890   \tw@mk@warning{'\string\winmenu' only for Windows!}%
891 }
892 \tw@make@key@box{winmenu@win}{%
893   \begin{tikzpicture}[yshift=-0.2ex,menukeys key symbol]
894     \draw (0,0) rectangle (1.5ex,1.8ex);
895     \draw (0.25ex,1.4ex) -- ++(1ex,0);
896     \draw (0.25ex,1ex) -- ++(1ex,0);
897     \draw (0.25ex,0.6ex) -- ++(1ex,0);
898   \end{tikzpicture}%
899 }
900 \tw@make@key@macro*{\winmenu}

\backspace
901 \tw@make@key@box{backspace}{%
902   \begin{tikzpicture}[yshift=0.65ex,menukeys key symbol]

```

```

903     \draw [←,menukeys thick] (0,0) -- (1.35em,0);
904     \end{tikzpicture}%
905 }
906 \tw@make@key@macro{\backspace}

\del
\backdel 907 \providecommand{\delname}{Del.}
908 \def\tw@mk@del@win{\delname}
909 \tw@define@macrokey{%
910     \def\tw@mk@del@mac{\delname}%
911 }{%
912     \tw@make@key@box{del@mac}{%
913         \begin{tikzpicture}[yshift=0.2ex,menukeys key symbol]
914             \draw (0,0) -- (1.5ex,0) -- (2ex,0.5ex) --%
915                 (1.5ex,1ex) -- (0,1ex) -- cycle;
916             \draw (0.5ex,0.2ex) -- (1.1ex,0.8ex);
917             \draw (0.5ex,0.8ex) -- (1.1ex,0.2ex);
918         \end{tikzpicture}%
919     }%
920 }
921 \tw@make@key@macro*{\del}
922 \def\tw@mk@backdel@win{\delname}
923 \tw@define@macrokey{%
924     \def\tw@mk@backdel@mac{\delname}%
925 }{%
926     \tw@make@key@box{backdel@mac}{%
927         \begin{tikzpicture}[yshift=0.2ex,menukeys key symbol]
928             \draw (2ex,0) -- (0.5ex,0) -- (0,0.5ex) --%
929                 (0.5ex,1ex) -- (2ex,1ex) -- cycle;
930             \draw (1ex,0.2ex) -- (1.6ex,0.8ex);
931             \draw (1ex,0.8ex) -- (1.6ex,0.2ex);
932         \end{tikzpicture}%
933     }%
934 }
935 \tw@make@key@macro*{\backdel}

\arrowkeyup Lastly we define the arrow macros:
\arrowkeydown 936 \tw@make@key@box{arrowkeyup}{%
\arrowkeyleft 937     \begin{tikzpicture}[yshift=-0.2ex,menukeys key symbol]
\arrowkeyright 938     \draw [→] (0,0) -- (0,0.8em);
939     \end{tikzpicture}%
940 }
941 \tw@make@key@macro{\arrowkeyup}
942
943 \tw@make@key@box{arrowkeydown}{%
944     \begin{tikzpicture}[yshift=0.7em,menukeys key symbol]
945     \draw [→] (0,0) -- (0,-0.8em);
946     \end{tikzpicture}%
947 }
948 \tw@make@key@macro{\arrowkeydown}

```

```

949
950 \tw@make@key@box{arrowkeyright}{%
951   \begin{tikzpicture}[yshift=0.5ex,menukeys key symbol]
952     \draw [->] (0,0) -- (0.8em,0);
953   \end{tikzpicture}%
954 }
955 \tw@make@key@macro{\arrowkeyright}
956
957 \tw@make@key@box{arrowkeyleft}{%
958   \begin{tikzpicture}[yshift=0.5ex,menukeys key symbol]
959     \draw [->] (0,0) -- (-0.8em,0);
960   \end{tikzpicture}%
961 }
962 \tw@make@key@macro{\arrowkeyleft}

```

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

```

963 \newcommand{\arrowkey}[1]{%
964   \IfStrEq{~}{#1}{\arrowkeyup}{%
965     \IfStrEq{v}{#1}{\arrowkeydown}{%
966       \IfStrEq{<}{#1}{\arrowkeyleft}{%
967         \IfStrEq{>}{#1}{\arrowkeyright}{%
968           \tw@mk@error{Wrong value '#1' for \string\arrowkey\MessageBreak
969             Possible values are '~', 'v', '<' or '>'}%
970         }%
971       }%
972     }%
973   }%
974 }

```

Close the `\iftw@mk@definekeys`

```

975 \fi

```

## 7 Change history

v1.0	Improved key symbols. . . . .	1
General: Initial version . . . . .		1
v1.1		
<code>\directory</code> : Renamed <code>\path</code> to		
<code>\directory</code> because it crashes		
with <code>biblatex</code> . . . . .		33
General: Improved manual . . . . .		1
Load <code>xcolor</code> before <code>menukeys</code> . . . . .		15
v1.1a		
<code>\newmenuacro</code> : Added a line to		
make a new macro robust. . . . .		32
<code>\tw@define@menu@macro@</code> : Fixed		
minor bug, that causes a		
warning about robustifying		
(issue #23), by deleting the		
line to make the command		
robust. . . . .		30
v1.2		
<code>\tw@define@menu@macro@</code> : Added		
<code>\leavevmode</code> . . . . .		30
Replaced <code>\edef</code> by		
<code>\protected@edef</code> . . . . .		30
General: Added <code>\normalsize</code>		
before symbol definitions to		
make the <code>ex</code> unit available . . . . .		1
Added <code>\SPACE</code> and <code>\spacename</code> . . . . .		1
Fixed GitHub issues #9, #10,		
#11, #13, #17, #24 and #26 . . . . .		1
Tidy up version and date . . . . .		1
v1.2a		
General: Added braces to the		
<code>\tikz</code> macro since the parser		
seems to crash with <code>babel</code> 's		
<code>french</code> option otherwise. . . . .		1
Replaced obsolete <code>\tikzstyle</code> . . . . .		1
v1.2c		
<code>\tw@define@menu@macro@</code> :		
Replaced <code>\protected@edef</code> by		
<code>\def</code> . . . . .		30
v1.3		
General: Added TikZ-styles for the		
key symbols. . . . .		1
v1.4		
<code>\backdel</code> : Added <code>\backdel</code> . . . . .		38
<code>\oldesc</code> : Fixed direction of		
<code>\escmac</code> ; added <code>\oldesc</code> . . . . .		35
General: Extended color theme		
features. . . . .		1
The <code>path...</code> styles now use the		
text color of the selected color		
theme (fix issue #16). . . . .		1
v1.5		
General: New option		
<code>hyperrefcolorlinks</code> . . . . .		18
v1.6		
<code>\newmenuacro</code> : use		
<code>\NewDocumentCommand</code> . . . . .		32
<code>\providemenuacro</code> : use		
<code>\ProvideDocumentCommand</code> . . . . .		32
<code>\renewmenuacro</code> : use		
<code>\RenewDocumentCommand</code> . . . . .		32
<code>\tw@define@menu@macro@</code> : Don't		
use <code>\NewDocumentCommand</code> . . . . .		30
General: <code>hyperrefcolorlinks</code>		
obsolete . . . . .		18
Don't load <code>catoptions</code> . . . . .		15
Load order no longer important . . . . .		5
v1.6.1		
<code>\newmenuacro</code> : default handled by		
<code>\tw@define@menu@macro</code> . . . . .		32
<code>\providemenuacro</code> : default		
handled by		
<code>\tw@define@menu@macro</code> . . . . .		32
<code>\renewmenuacro</code> : default handled		
by <code>\tw@define@menu@macro</code> . . . . .		32
<code>\tw@define@menu@macro</code> : Handles		
default input separator. . . . .		30
<code>\tw@define@menu@macro@</code> : No		
x-type expansion on the		
separator to call the loop . . . . .		30
Renamed from		
<code>\tw@define@menu@macro</code> . . . . .		30



## 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>\@ifdefinable</code> .....	625
<code>\@tw@hyphenatepaths@warnigfalse</code> .	573
<code>\@tw@hyphenatepaths@warnigtrue</code> .	561
<code>\_</code> .....	28, 32, 36, 42, 57, 58, 67, 68
<b>A</b>	
<code>\Alt</code> .....	837
<code>\AltGr</code> .....	837
<code>angularkeys</code> (style) .....	7
<code>angularmenus</code> (style) .....	6
<code>\arrowkey</code> .....	14, 963
<code>\arrowkeydown</code> .....	936, 965
<code>\arrowkeyleft</code> .....	936, 966
<code>\arrowkeyright</code> .....	936, 967
<code>\arrowkeyup</code> .....	936, 964
<b>B</b>	
<code>\backdel</code> .....	907
<code>\backspace</code> .....	901
<code>blacknwhite</code> (theme) .....	12
<code>\bool</code> .....	61
<b>C</b>	
<code>\capslock</code> .....	778
<code>\changemenucolor</code> .....	13, 130
<code>\changemenucolortheme</code> .....	11, 151
<code>\changemenuelement</code> .....	11, 320, 598, 604, 610, 616
<code>\cmd</code> .....	850
<code>\color</code> .....	333, 361, 366, 370, 373, 392, 397, 401, 404, 422, 427, 431, 434, 450, 474, 494, 517, 538, 540, 549, 577
Color themes:	
<code>blacknwhite</code> .....	12
<code>gray</code> .....	12
<code>\copymenucolortheme</code> .....	13, 130
<code>\copymenustyle</code> .....	11, 302, 597, 603, 609, 615
<code>\cs</code> .....	10, 11, 12, 14, 18, 25, 28, 32, 36, 42, 54, 55
<code>\ctrl</code> .....	833
<code>\ctrlname</code> .....	14, 833, 834
<code>\CurrentMenuElement</code> .....	11, 188, 191, 194, 197, 220, 223, 226, 229, 333, 361, 366, 370, 373, 392, 397, 401, 404, 422, 427, 431, 434, 450, 474, 494, 517, 533, 549, 578, 621, 652
<b>D</b>	
<code>definekeys</code> (option) .....	5
<code>definenumacros</code> (option) .....	5
<code>\del</code> .....	907
<code>\delname</code> ...	14, 907, 908, 910, 922, 924
<code>\directory</code> .....	5, 716
<code>\drawtikzfolder</code> .....	10, 588, 599, 605, 611, 617
<b>E</b>	
<code>\enter</code> .....	881
<code>\esc</code> .....	809
<code>\exp</code> .....	11, 49, 65
<code>\ExplSyntaxOff</code> .....	70
<code>\ExplSyntaxOn</code> .....	9
<b>F</b>	
<code>\font</code> .....	576
<b>G</b>	
<code>gray</code> (theme) .....	12
<code>\group</code> .....	47, 50
<b>H</b>	
<code>\hypersetup</code> .....	108
<code>hyphenatepaths</code> (style) .....	9
<code>hyphenatepathswithblackfolder</code> (style) .....	9
<code>hyphenatepathswithfolder</code> (style) ..	9
<b>I</b>	
<code>\if@tw@hyphenatepaths@warnig</code>	560, 570
<code>\IfNoValueTF</code> .....	635
<code>\iftw@mk@definekeys</code> .....	720
<code>\iftw@mk@definenumacros</code> .....	715
<code>\iftw@mk@hyperrefcolorlinks</code> ...	105
<code>\int</code> .....	23, 45, 51, 60, 63

<b>K</b>	
<code>\keys</code> .....	5, 716
<b>L</b>	
<code>\l</code> .....	20, 22, 23, 24, 25, 26, 27, 30, 34, 38, 39, 40, 44, 45, 48, 51, 59, 60, 61, 63, 64, 65
<b>M</b>	
<code>mackeys</code> (option) .....	5, 14
<code>\menu</code> .....	5, 716
<code>menus</code> (style) .....	6
<b>N</b>	
<code>\newmenucolortheme</code> .	12, 119, 166, 167
<code>\newmenumacro</code> ..	13, 696, 716, 717, 718
<code>\newmenustyle</code> .....	10, 233, 268
<code>\newmenustylesimple</code> ....	10, 233, 233
<b>O</b>	
<code>\oldesc</code> .....	809
Options:	
<code>definekeys</code> .....	5
<code>definenumacros</code> .....	5
<code>mackeys</code> .....	5, 14
<code>os</code> .....	5
<code>os</code> (option) .....	5
<b>P</b>	
<code>\PassOptionsToPackage</code> .....	109
<code>paths</code> (style) .....	8
<code>pathswithblackfolder</code> (style) .....	8
<code>pathswithfolder</code> (style) .....	8
<code>\prg</code> .....	13
<code>\protected</code> .....	627
<code>\providenumacro</code> .....	14, 696
<code>\providemenustyle</code> .....	12, 233, 288
<code>\providemenustylesimple</code> .....	12, 233, 253
<b>R</b>	
<code>\relsize</code> 353, 384, 414, 445, 453, 465, 478, 489, 497, 508, 520, 529, 544	
<code>\renewmenucolortheme</code> ...	13, 119, 148
<code>\renewnumacro</code> .....	14, 696
<code>\renewmenustyle</code> .....	.. 12, 233, 277, 281, 298, 701, 712
<code>\renewmenustylesimple</code> .....	.. 12, 233, 242, 246, 263
<code>\return</code> .....	869
<code>roundedkeys</code> (style) .....	7
<code>roundedmenus</code> (style) .....	6
<b>S</b>	
<code>\seq</code> .....	20, 22, 26, 27, 30, 34, 38, 39, 40, 44, 48, 54, 59, 61, 64
<code>shadowedangularkeys</code> (style) .....	7
<code>shadowedroundedkeys</code> (style) .....	7
<code>\shift</code> .....	769
<code>\SPACE</code> .....	866
<code>\Space</code> .....	866
<code>\spacename</code> .....	14, 867, 868
Styles:	
<code>angularkeys</code> .....	7
<code>angularmenus</code> .....	6
<code>hyphenatepathswithblackfolder</code> .	9
<code>hyphenatepathswithfolder</code> ....	9
<code>hyphenatepaths</code> .....	9
<code>menus</code> .....	6
<code>pathswithblackfolder</code> .....	8
<code>pathswithfolder</code> .....	8
<code>paths</code> .....	8
<code>roundedkeys</code> .....	7
<code>roundedmenus</code> .....	6
<code>shadowedangularkeys</code> .....	7
<code>shadowedroundedkeys</code> .....	7
<code>typewriterkeys</code> .....	8
<b>T</b>	
<code>\tab</code> .....	794
<code>\tikzset</code> .....	340, 345, 376, 407, 437, 457, 482, 501, 524, 761
<code>\tl</code> .....	10, 13, 16, 24
<code>\tw@current@color@theme</code> ...	164, 675
<code>\tw@current@style</code> ....	203, 204, 209
<code>\tw@declare@style</code> .	206, 271, 273, 283, 285, 291, 293, 358, 389, 419
<code>\tw@declare@style@simple</code> ...	173, 236, 238, 248, 250, 256, 258, 447, 471, 491, 514, 531, 548, 562
<code>\tw@declare@sytle</code> .....	200
<code>\tw@declare@sytle@extra@args</code> ..	200
<code>\tw@default@input@sep</code> .....	620, 636
<code>\tw@default@post</code> .....	168, 174, 201
<code>\tw@default@pre</code> .....	168, 174, 207
<code>\tw@default@sep</code> .....	168, 174, 207
<code>\tw@define@mackey</code> .....	.. 756, 810, 822, 838, 853, 909, 923
<code>\tw@define@menu@macro</code> .....	.. 622, 698, 705, 709
<code>\tw@define@menu@macro@</code> .....	622

<code>\tw@make@color@theme</code>	.. 111, 121, 128	<code>\tw@mk@mackeys</code>	..... 101, 757, 758
<code>\tw@make@key@box</code>	..... 721, 770,	<code>\tw@mk@oldesc@mac</code>	..... 823
	778, 786, 794, 800, 813, 825,	<code>\tw@mk@oldesc@win</code>	..... 821
	841, 856, 869, 875, 882, 892,	<code>\tw@mk@os</code>	..... 97, 735
	901, 912, 926, 936, 943, 950, 957	<code>\tw@mk@string</code>	.....
<code>\tw@make@key@macro</code>	.....	.. 12, 650, 688, 697, 708, 733,	
	. 731, 777, 793, 808, 820, 832,	735, 738, 740, 743, 745, 749, 751	
	836, 848, 865, 880, 888, 900,	<code>\tw@mk@tempa</code>	. 80, 83, 84, 681, 685, 689
	906, 921, 935, 941, 948, 955, 962	<code>\tw@mk@tempb</code>	..... 80, 688, 689
<code>\tw@menu@list</code>	..... 680, 684, 689	<code>\tw@mk@test@input@sep</code>	. 625, 627, 678
<code>\tw@mk@Alt@mac</code>	..... 839	<code>\tw@mk@trimspaces</code>	..... 10, 629, 685
<code>\tw@mk@Alt@win</code>	..... 837	<code>\tw@mk@warning</code>	..... 71,
<code>\tw@mk@backdel@mac</code>	..... 924		106, 261, 296, 571, 711, 851, 890
<code>\tw@mk@backdel@win</code>	..... 922	<code>\tw@mk@warning@noline</code>	..... 71
<code>\tw@mk@cmd@mac</code>	..... 854	<code>\tw@mk@winmenu@mac</code>	..... 889
<code>\tw@mk@cmd@win</code>	..... 850	<code>\tw@mk@xifinsetTF</code>	..... 14, 628
<code>\tw@mk@ctrl@mac</code>	..... 835	<code>\tw@set@tikz@colors</code>	..... 340
<code>\tw@mk@ctrl@win</code>	..... 834	<code>\tw@typewriterkeys@curr@elem</code>	...
<code>\tw@mk@del@mac</code>	..... 910		..... 532, 538, 540
<code>\tw@mk@del@win</code>	..... 908	<code>typewriterkeys (style)</code>	..... 8
<code>\tw@mk@enter@win</code>	..... 881		
<code>\tw@mk@error</code>	.... 71, 98, 102, 123,	<b>U</b>	
	134, 147, 153, 157, 241, 276,	<code>\usemenucolor</code>	..... 11, 163,
	305, 317, 322, 336, 643, 700, 968		333, 341, 342, 343, 361, 366,
<code>\tw@mk@esc@mac</code>	..... 811		370, 373, 392, 397, 401, 404,
<code>\tw@mk@esc@win</code>	..... 809		422, 427, 431, 434, 450, 453,
<code>\tw@mk@exp@Nnno</code>	..... 11, 636		468, 474, 478, 494, 497, 511,
<code>\tw@mk@gobble@args</code>	.....		517, 520, 538, 540, 544, 549,
	..... 82, 243, 264, 278, 299		553, 577, 582, 599, 605, 611, 617
<code>\tw@mk@iflastindris</code>	..... 18, 653	<b>W</b>	
<code>\tw@mk@indrisloop</code>	..... 55, 689	<code>\winmenu</code>	..... 889
<code>\tw@mk@indrisnr</code>	..... 25, 655, 663		