



Tobias Weh

mail@tobiw.de

<http://tobiw.de/en>

<http://github.com/tweh/menukeys>

<http://www.ctan.org/pkg/menukeys>

[macros > latex > contrib > menukeys](#)

2016/04/18 — v1.4

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	Color themes	16
6.4.1	Internal commands	16
6.4.2	User-level commands	16
6.4.3	Predefined themes	17
6.5	Styles	17
6.5.1	Internal commands	17
6.5.2	User-level commands	19
6.5.3	Copying and changing	20
6.5.4	Predefined styles	21
6.6	Menu macros	27
6.6.1	Internal commands	27
6.6.2	User-level commands	28
6.6.3	Predefined menu macros	29
6.7	Keys	29
7	Change history	36

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 L^AT_EX 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:

<code>definemenumacros</code> (opt.)	definemenumacros: Most of <code>menukeys</code> ' macros should not conflict with other packages ³ but the predefined menu macros should be short and easy-to-read commands, which means that <code>\menu{A,B,C}</code> is preferred against <code>\printmenusequence{A,B,C}</code> . For that it's not unlikely that they conflict with other packages. To prevent this you can tell <code>menukeys</code> to not define them by calling the option <code>definemenumacros=false</code> . The default value is <code>true</code> .
	If you do so you have to define your own menu macros, see section 4.4 for details.
<code>definekeys</code> (opt.)	definekeys: Equal to <code>definemenumacros</code> for the key macros. The default value is <code>true</code> .
<code>mackeys</code> (opt.)	mackeys: This option allows you to decide whether the mac keys are shown as text (<code>mackeys=text</code>) or symbols (<code>mackeys=symbols</code>). The default value is <code>symbols</code> .
<code>os</code> (opt.)	os: You can specify the OS by saying <code>os=mac</code> or <code>os=win</code> . This will cause some key macros to be rendered differently. The default value is <code>mac</code> .

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.

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

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

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

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

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

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: `menus`

File > Extras > Preferences

Menu

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

Name: `roundedmenus`

File > Extras > Preferences

Menu

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

Name: `angularmenus`

File > Extras > Preferences

Menu

This is some more or less blind text, to demonstrate how the sequence looks in text. This File > Extras > Preferences is the result of a style which name is `angularmenus`. 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 TeX 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 TeX 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. 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}}{$\ast$}
```

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
\providemenustylesimple
    \renewmenustyle
\providemenustyle
```

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

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

4.3.2 Create a theme

```
\newmenucolortheme{<name>}{<model>}{<bg>}{<br>}{<txt>}[<a>][<b>][<c>]
```

To create a new theme use `\newmenucolortheme`. It uses the following arguments:

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{<copy>}{<original>}`

To copy the definitions of one theme to another, use `\copymenucolortheme{<copy>}{<original>}`.

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{<name>}{<element>}{<model>}{<color definition>}`, 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 marcos” 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{<macro>} [<input sep>]{<style>}`.

`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 wil 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
\renewmenumacro
```

There are also the two commands `\providemenumacro` and `\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			

- | | |
|-----------------------------|---|
| <code>\arrowkey</code> | The macro <code>\arrowkey{⟨direction⟩}</code> is a little special since it takes the direction as a single character ^, v (lower case v), > or <. |
| <code>\ctrlname</code> | The texts for <code>\ctrl</code> , <code>\del</code> and <code>\SPACE</code> are saved in <code>\ctrlname</code> , <code>\delname</code> , <code>\spacename</code> respectively. So you can change them with <code>\renewcommand</code> . |
| <code>\delname</code> | |
| <code>\spacename</code> | |
| <code>mackeys</code> (opt.) | The rendering of some Mac macros depend on the option <code>mackeys</code> . The different versions are shown in the table (left: <code>text</code> , right: <code>symbols</code>). |

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 `cmyk` model:

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

If you find something to add to this list please send me an e-mail.

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]{definemenumacros}
31 \DeclareBoolOption[true]{definekeys}
32 \DeclareStringOption[mac]{os}
33 \DeclareStringOption[symbols]{mackeys}
34 \ProcessKeyvalOptions{tw@mk}\relax
```

Now we have to do some error treatment:

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

6.4 Color themes

6.4.1 Internal commands

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

```
43 \newcommand*\tw@make@color@theme[8]{%
44   \definecolor{tw@color@theme@#1@bg}{#2}{#3}%
45   \definecolor{tw@color@theme@#1@br}{#2}{#4}%
46   \definecolor{tw@color@theme@#1@txt}{#2}{#5}%
47   \definecolor{tw@color@theme@#1@a}{#2}{#6}%
48   \definecolor{tw@color@theme@#1@b}{#2}{#7}%
49   \definecolor{tw@color@theme@#1@c}{#2}{#8}%
50 }
```

6.4.2 User-level commands

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

```
51 \NewDocumentCommand{\newmenucolortheme}{ m m m m m O{#3} O{#4} O{#5} }{%
52   \@ifundefinedcolor{tw@color@theme@#1@bg}{%
53     \tw@make@color@theme{#1}{#2}{#3}{#4}{#5}{#6}{#7}{#8}}%
54   }{%
55     \tw@mk@error{Color theme '#1' already defined!\MessageBreak
56     Use \string\renewmenucolortheme\space instead.}%
57   }
58 }
59 \NewDocumentCommand{\renewmenucolortheme}{ m m m m m O{#3} O{#4} O{#5} }{%
60   \tw@make@color@theme{#1}{#2}{#3}{#4}{#5}{#6}{#7}{#8}}%
61 }
```

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

```
62 \newcommand*\changemenucolor[4]{%
63   \IfSubStr{ bg br txt }{ #2 }{%
64     \definecolor{tw@color@theme@#1@#2}{#3}{#4}%
65   }{%
66     \tw@mk@error{No such color element ('#2')!\MessageBreak
67     Possible values are bg, br and txt.}%
68   }%
69 }
70 \newcommand*\copymenucolortheme[2]{%
71   \@ifundefinedcolor{tw@color@theme@#1@bg}{%
72     \colorlet{tw@color@theme@#1@bg}{tw@color@theme@#2@bg}%
73     \colorlet{tw@color@theme@#1@br}{tw@color@theme@#2@br}%
74     \colorlet{tw@color@theme@#1@txt}{tw@color@theme@#2@txt}%
75     \colorlet{tw@color@theme@#1@a}{tw@color@theme@#2@a}%
76     \colorlet{tw@color@theme@#1@b}{tw@color@theme@#2@b}%
77     \colorlet{tw@color@theme@#1@c}{tw@color@theme@#2@c}%
78   }{%
79     \tw@mk@error{Color theme '#1' already defined!\MessageBreak
80     Use \string\renewmenucolortheme\space instead.}%
81   }
```

```

82 }

\changemenucolortheme To be able to change the color theme of a style we must define this:
83 \newcommand{\changemenucolortheme}[2]{%
84   \ifcsundef{tw@style@#1@pre}{%
85     \tw@mk@error{Style '#1' undefined!}\MessageBreak
86     Maybe you misspelled it?}%
87 }{%
88   \@ifundefinedcolor{tw@color@theme@#2@bg}{%
89     \tw@mk@error{Color theme '#2' is not defined!}%
90   }{%
91     \csdef{tw@style@#1@color@theme}{#2}%
92   }%
93 }%
94 }

\usemenucolor To use a color of a theme we define \usemenucolor as following.
95 \newcommand{\usemenucolor}[1]{%
96   tw@color@theme@\tw@current@color@theme @#1%
97 }

```

6.4.3 Predefined themes

There are two predefined color themes

```

98 \newmenucolortheme{gray}{gray}{0.95}{0.3}{0}[0][0]
99 \newmenucolortheme{blacknwhite}{gray}{1}{0}{0}[1][0][0]

```

6.5 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
100 \newcommand{\tw@default@sep}{%
101   \hspace{0.2em plus 0.1em minus 0.5em}%
102 }
103 \newcommand{\tw@default@pre}{}
104 \newcommand{\tw@default@post}{}

```

6.5.1 Internal commands

Now we can define the internal commands.

```

\tw@declare@style@simple Our first step is to define the simple command.
105 \DeclareDocumentCommand{\tw@declare@style@simple}{%
106   s m O{\tw@default@pre} m O{\tw@default@sep} O{\tw@default@post} m
107 }{%
108   \csdef{tw@style@#2@color@theme}{#7}%

```

```

109  \csdef{tw@style@#2@pre}{#3}%
110  \csdef{tw@style@#2@sep}{#5}%
111  \csdef{tw@style@#2@post}{#6}%
112  \IfBooleanTF{#1}{%
113      \csdef{tw@style@#2@singl}{#4}%
114      \csdef{tw@style@#2@first}{#4}%
115      \csdef{tw@style@#2@mid}{#4}%
116      \csdef{tw@style@#2@last}{#4}%
117  }{%
118      \csdef{tw@style@#2@singl}{%
119          \tikz [baseline=(tw@node.base)]{%
120              \node(tw@node)[#4]{\strut\CurrentMenuElement};}}%
121      \csdef{tw@style@#2@first}{%
122          \tikz [baseline=(tw@node.base)]{%
123              \node(tw@node)[#4]{\strut\CurrentMenuElement};}}%
124      \csdef{tw@style@#2@mid}{%
125          \tikz [baseline=(tw@node.base)]{%
126              \node(tw@node)[#4]{\strut\CurrentMenuElement};}}%
127      \csdef{tw@style@#2@last}{%
128          \tikz [baseline=(tw@node.base)]{%
129              \node(tw@node)[#4]{\strut\CurrentMenuElement};}}%
130  }%
131 }

```

\tw@declare@sytle 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.

```

132 \DeclareDocumentCommand{\tw@declare@sytle@extra@args}{%
133     O{\tw@default@post} m
134 }{%
135     \csdef{tw@style@#1@current@style @post}{#1}%
136     \csdef{tw@style@#1@current@style @color@theme}{#2}%
137 }

```

Now we can define \tw@declare@style:

```

138 \DeclareDocumentCommand{\tw@declare@style}{%
139     s m O{\tw@default@pre} m O{\tw@default@sep} m m m
140 }{%
141     \def\tw@current@style{#2}
142     \csdef{tw@style@#2@pre}{#3}%
143     \csdef{tw@style@#2@sep}{#5}%
144     \IfBooleanTF{#1}{%
145         \csdef{tw@style@#2@singl}{#8}%
146         \csdef{tw@style@#2@first}{#4}%
147         \csdef{tw@style@#2@mid}{#6}%
148         \csdef{tw@style@#2@last}{#7}%
149     }{%
150         \csdef{tw@style@#2@singl}{%
151             \tikz [baseline=(tw@node.base)]{%
152                 \node(tw@node)[#8]{\strut\CurrentMenuElement};}}%

```

```

153   \csdef{tw@style@#2@first}{%
154     \tikz [baseline=(tw@node.base)]{%
155       \node(tw@node)[#4]{\strut\CurrentMenuElement};}}%
156   \csdef{tw@style@#2@mid}{%
157     \tikz [baseline=(tw@node.base)]{%
158       \node(tw@node)[#6]{\strut\CurrentMenuElement};}}%
159   \csdef{tw@style@#2@last}{%
160     \tikz [baseline=(tw@node.base)]{%
161       \node(tw@node)[#7]{\strut\CurrentMenuElement};}}%
162   }%
163   \tw@declare@style@extra@args%
164 }

```

6.5.2 User-level commands

```

newmenustyle simple It's time to define the user-level commands now:
renewmenustyle simple
providemenustyle simple
newmenustyle
renewmenustyle
providemenustyle
165 \NewDocumentCommand{\newmenustyle}{s m}{%
166   \ifcsundef{tw@style@#2@pre}{%
167     \IfBooleanTF{#1}{%
168       \tw@declare@style@simpler*{#2}%
169     }{%
170       \tw@declare@style@simpler{#2}%
171     }%
172   }{%
173     \tw@mk@error{Style '#2' already defined!\MessageBreak
174     Use \string\newmenustyle\space instead.}%
175     \tw@mk@gobble@args{o m o o m}%
176   }%
177 }
178 \NewDocumentCommand{\renewmenustyle}{s m}{%
179   \IfBooleanTF{#1}{%
180     \tw@declare@style@simpler*{#2}%
181   }{%
182     \tw@declare@style@simpler{#2}%
183   }%
184 }
185 \NewDocumentCommand{\providemenustyle}{s m}{%
186   \ifcsundef{tw@style@#2@pre}{%
187     \IfBooleanTF{#1}{%
188       \tw@declare@style@simpler*{#2}%
189     }{%
190       \tw@declare@style@simpler{#2}%
191     }%
192   }{%
193     \tw@mk@warning{Trying to provide style '#2' failed,\MessageBreak
194     because it's already defined.\MessageBreak
195     You may use \string\renewmenustyle\space instead.}%
196     \tw@mk@gobble@args{o m o o m}%
197   }%

```

```

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

```

6.5.3 Copying and changing

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

```

234 \newcommand*{\copymenustyle}[2]{%
235     \ifcsundef{tw@style@#1@pre}{%
236         \ifcsundef{tw@style@#2@pre}{%
237             \tw@mk@error{Can't copy not existing style ('#2')!}%
238         }{%
239             \csletcs{tw@style@#1@pre}{tw@style@#2@pre}%
240             \csletcs{tw@style@#1@post}{tw@style@#2@post}%
241             \csletcs{tw@style@#1@sep}{tw@style@#2@sep}%
242             \csletcs{tw@style@#1@single}{tw@style@#2@single}%

```

```

243      \csletcs{tw@style@#1@first}{tw@style@#2@first}%
244      \csletcs{tw@style@#1@mid}{tw@style@#2@mid}%
245      \csletcs{tw@style@#1@last}{tw@style@#2@last}%
246      \csletcs{tw@style@#1@color@theme}{tw@style@#2@color@theme}%
247      }%
248  }{%
249      \tw@mk@error{Style '#1' already exists!}%
250  }%
251 }

```

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

```

252 \NewDocumentCommand{\changemenuelement}{s m m m}{%
253     \ifcsundef{tw@style@#2@pre}{%
254         \tw@mk@error{Style '#2' undefined.}%
255     }{%
256         \IfSubStr{single first middle last pre post sep }{ #3 }{%
257             \IfBooleanTF{#1}{%
258                 \csdef{tw@style@#2@#3}{#4}%
259             }{%
260                 \IfSubStr{pre post sep }{ #3 }{%
261                     \csdef{tw@style@#2@#3}{#4}%
262                 }{%
263                     \csdef{tw@style@#2@#3}{%
264                         \tikz [baseline=(tw@node.base)]{%
265                             \node(tw@node)[#4]{\strut\color{\usemenucolor{txt}}\CurrentMenuElement};}%
266                     }%
267                 }%
268             }{%
269                 \tw@mk@error{No element '#3'. Possible values are\MessageBreak
270                 single, first, middle, last, pre, post or sep.}%
271             }%
272         }{%
273             \tw@mk@error{Style '#1' already exists!}%
274         }%
275     }%
276 }

```

6.5.4 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

```

272 \tikzset{tw@set@tikz@colors/.style={%
273     draw=\usemenucolor{br},
274     fill=\usemenucolor{bg},
275     text=\usemenucolor{txt},
276 }}

```

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`.

```

277 \tikzset{tw@menus@base/.style={%
278     tw@set@tikz@colors,
279     rounded corners=0.15ex,
280     inner sep=0pt,
281     inner xsep=2pt,

```

```

282     text height=1.825ex,
283     text depth=0.7ex,
284     minimum width=1.5em,
285     font=\relsize{-1}\sffamily,
286     signal,
287     signal to=nowhere,
288     signal pointer angle=110,
289 }
290 \tw@declare@style*{menus}{%
291     \tikz[baseline={($tw@node.base)+(0,-0.2ex)}]{%
292         \node(tw@node)[tw@menus@base,signal to=east]%
293             {\strut\color{\usemenucolor{txt}}\CurrentMenuElement};}%
294 }[\hspace{-0.2em}\hspace{0em plus 0.1em minus 0.05em}]%
295 {%
296     \tikz[baseline={($tw@node.base)+(0,-0.2ex)}]{%
297         \node(tw@node)[tw@menus@base,signal from=west,signal to=east]%
298             {\strut\color{\usemenucolor{txt}}\CurrentMenuElement};}%
299 }{%
300     \tikz[baseline={($tw@node.base)+(0,-0.2ex)}]{%
301         \node(tw@node)[tw@menus@base,signal from=west,]%
302             {\strut\color{\usemenucolor{txt}}\CurrentMenuElement};}%
303 }{%
304     \tikz[baseline={($tw@node.base)+(0,-0.2ex)}]{%
305         \node(tw@node)[tw@menus@base]{\strut\color{\usemenucolor{txt}}\CurrentMenuElement};}%
306 }{gray}
307
308 \tikzset{tw@roundedmenus@base/.style={%
309     tw@set@tikz@colors,
310     rounded corners=0.3ex,
311     inner sep=0pt,
312     inner xsep=2pt,
313     text height=1.825ex,
314     text depth=0.7ex,
315     minimum width=1.5em,
316     font=\relsize{-1}\sffamily,
317     signal,
318     signal to=nowhere,
319     signal pointer angle=110,
320 }}
321 \tw@declare@style*{roundedmenus}{%
322     \tikz[baseline={($tw@node.base)+(0,-0.2ex)}]{%
323         \node(tw@node)[tw@roundedmenus@base,signal to=east]%
324             {\strut\color{\usemenucolor{txt}}\CurrentMenuElement};}%
325 }[\hspace{-0.2em}\hspace{0em plus 0.1em minus 0.05em}]%
326 {%
327     \tikz[baseline={($tw@node.base)+(0,-0.2ex)}]{%
328         \node(tw@node)[tw@roundedmenus@base,signal from=west,signal to=east]%
329             {\strut\color{\usemenucolor{txt}}\CurrentMenuElement};}%
330 }{%
331     \tikz[baseline={($tw@node.base)+(0,-0.2ex)}]{%

```

```

332      \node(tw@node)[tw@roundedmenus@base,signal from=west,]%
333      {\strut\color{\usemenucolor{txt}}\CurrentMenuElement;}%
334 }{%
335   \tikz[baseline={($(tw@node.base)+(0,-0.2ex)$)}]{%
336     \node(tw@node)[tw@roundedmenus@base]{\strut\color{\usemenucolor{txt}}\CurrentMenuElement}%
337   }{gray}
338
339 \tikzset{tw@angularmenus@base/.style={%
340   tw@set@tikz@colors,
341   inner sep=0pt,
342   inner xsep=2pt,
343   text height=1.825ex,
344   text depth=0.7ex,
345   minimum width=1.5em,
346   font=\relsize{-1}\sffamily,
347   signal,
348   signal to=nowhere,
349   signal pointer angle=110,
350 }}%
351 \tw@declare@style*{angularmenus}{%
352   \tikz[baseline={($(tw@node.base)+(0,-0.2ex)$)}]{%
353     \node(tw@node)[tw@angularmenus@base,signal to=east]%
354     {\strut\color{\usemenucolor{txt}}\CurrentMenuElement;}%
355   }[\hspace{-0.2em}\hspace{0em plus 0.1em minus 0.05em}]%
356 }{%
357   \tikz[baseline={($(tw@node.base)+(0,-0.2ex)$)}]{%
358     \node(tw@node)[tw@angularmenus@base,signal from=west,signal to=east]%
359     {\strut\color{\usemenucolor{txt}}\CurrentMenuElement;}%
360 }{%
361   \tikz[baseline={($(tw@node.base)+(0,-0.2ex)$)}]{%
362     \node(tw@node)[tw@angularmenus@base,signal from=west,]%
363     {\strut\color{\usemenucolor{txt}}\CurrentMenuElement;}%
364 }{%
365   \tikz[baseline={($(tw@node.base)+(0,-0.2ex)$)}]{%
366     \node(tw@node)[tw@angularmenus@base]{\strut\color{\usemenucolor{txt}}\CurrentMenuElement}%
367   }{gray}
368
369 \tikzset{tw@roundedkeys@base/.style={%
370   tw@set@tikz@colors,
371   rounded corners=0.3ex,
372   inner sep=0pt,
373   inner xsep=2pt,
374   text height=1.825ex,
375   text depth=0.7ex,
376   minimum width=1.5em,
377   font=\relsize{-1}\sffamily,
378 }}%
379 \tw@declare@style@simple*{roundedkeys}{%
380   \tikz[baseline={($(tw@node.base)+(0,-0.2ex)$)}]{%
381     \node(tw@node)[tw@roundedkeys@base]%

```

```

382           {\strut\color{\usemenucolor{txt}}\CurrentMenuElement;}%
383 }[%
384   \hspace{0.1em plus 0.1em minus 0.05em}%
385   \textcolor{\usemenucolor{b}}{\raisebox{0.25ex}{\sffamily\relsize{-2}+}}%
386   \hspace{0.1em plus 0.1em minus 0.05em}%
387 ]{gray}
388
389 \tikzset{tw@shadowedroundedkeys@base/.style={%
390   tw@set@tikz@colors,
391   rounded corners=0.3ex,
392   inner sep=0pt,
393   inner xsep=2pt,
394   text height=1.825ex,
395   text depth=0.7ex,
396   minimum width=1.5em,
397   font=\relsize{-1}\sffamily,
398   general shadow={%
399     shadow xshift=.2ex, shadow yshift=-.15ex,
400     fill=\usemenucolor{c},
401   },
402 }}%
403 \tw@declare@style@simpler*{shadowedroundedkeys}{%
404   \tikz[baseline={$(tw@node.base)+(0,-0.2ex)$}]{%
405     \node(tw@node)[tw@shadowedroundedkeys@base]%
406       {\strut\color{\usemenucolor{txt}}\CurrentMenuElement};%
407   }%
408 }[%
409   \hspace{0.2ex}\hspace{0.1em plus 0.1em minus 0.05em}%
410   \textcolor{\usemenucolor{b}}{\raisebox{0.25ex}{\sffamily\relsize{-2}+}}%
411   \hspace{0.1em plus 0.1em minus 0.05em}%
412 ][\hspace{0.2ex}]{gray}
413
414 \tikzset{tw@angularkeys@base/.style={%
415   tw@set@tikz@colors,
416   inner sep=0pt,
417   inner xsep=2pt,
418   text height=1.825ex,
419   text depth=0.7ex,
420   minimum width=1.5em,
421   font=\relsize{-1}\sffamily,
422 }}%
423 \tw@declare@style@simpler*{angularkeys}{%
424   \tikz[baseline={$(tw@node.base)+(0,-0.2ex)$}]{%
425     \node(tw@node)[tw@angularkeys@base]%
426       {\strut\color{\usemenucolor{txt}}\CurrentMenuElement};%
427   }%
428   \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 }{gray}

```

```

432 \tikzset{tw@shadowedangularkeys@base/.style={%
433   tw@set@tikz@colors,
434   inner sep=0pt,
435   inner xsep=2pt,
436   text height=1.825ex,
437   text depth=0.7ex,
438   minimum width=1.5em,
439   font=\relsize{-1}\sffamily,
440   general shadow={%
441     shadow xshift=.2ex, shadow yshift=-.15ex,
442     fill=\usemenucolor{c},
443   },
444 }
445 }%
446 \tw@declare@style@simple*{shadowedangularkeys}{%
447   \tikz[baseline={($ (tw@node.base)+(0,-0.2ex)$)}]{%
448     \node(tw@node)[tw@shadowedangularkeys@base]{
449       \strut\color{\usemenucolor{txt}}\CurrentMenuElement};}%
450 }[%]
451   \hspace{0.2ex}\hspace{0.1em plus 0.1em minus 0.05em}%
452   \textcolor{\usemenucolor{b}}{\raisebox{0.25ex}{\sffamily\relsize{-2}+}}}%
453   \hspace{0.1em plus 0.1em minus 0.05em}%
454 ][\hspace{0.2ex}]{gray}
455
456 \tikzset{tw@typewriterkeys@base/.style={%
457   tw@set@tikz@colors,
458   shape=circle,
459   minimum size=2ex,
460   inner sep=0.5pt, outer sep=1pt,
461   font=\ttfamily\relsize{-1},
462 }}
463 \tw@declare@style@simple*{typewriterkeys}{%
464   \def\tw@typewriterkeys@curr@elem{%
465     \maxsizebox*{2ex}{2ex}{\CurrentMenuElement}}%
466   }%
467 \begin{tikzpicture}[baseline={($ (tw@node.south)+(0,0.8ex)$)}]%
468   \node(tw@node)[%  

469     tw@typewriterkeys@base, inner sep=1.25pt, line width=0.6pt%
470   ]{\color{\usemenucolor{txt}}\tw@typewriterkeys@curr@elem};
471   \node[tw@typewriterkeys@base]{
472     \color{\usemenucolor{txt}}\tw@typewriterkeys@curr@elem};
473   \end{tikzpicture}%
474 }[%]
475   \hspace{0.2ex}\hspace{0.1em plus 0.1em minus 0.05em}%
476   \textcolor{\usemenucolor{b}}{\raisebox{0.25ex}{\sffamily\relsize{-2}+}}}%
477   \hspace{0.1em plus 0.1em minus 0.05em}%
478 ]{blacknwhite}
479
480 \tw@declare@style@simple*{paths}{%
481   \ttfamily\color{\usemenucolor{txt}}\CurrentMenuElement}%

```

```

482 }[%]
483   \hspace{0.2em plus 0.1em}%
484   \raisebox{0.08ex}{%
485     \tikz{\fill[\usemenucolor{c}] (0,0) -- (0.5ex,0.5ex)%
486       -- (0,1ex) -- cycle;}%
487   }%
488   \hspace{0.2em plus 0.1em}%
489 ]{blacknwhite}
490
491 \newcounter{tw@hyphen@char@num}
492 \newif\iftw@hyphenatepaths@warnig
493 \tw@hyphenatepaths@warnigtrue
494 \tw@declare@style@simpler*{hyphenatepaths}{%
495   {\ttfamily
496   \IfStrEq{T1}{\encodingdefault}{%
497     \setcounter{tw@hyphen@char@num}{23}%
498   }{%
499     \IfStrEq{OT1}{\encodingdefault}{%
500       \setcounter{tw@hyphen@char@num}{255}%
501     }{%
502       \if@tw@hyphenatepaths@warnig%
503         \tw@mk@warning{The hyphenatepaths styles will probably only\MessageBreak
504           work with T1 or OT1 encoding.}%
505         \fi\global\tw@hyphenatepaths@warnigfalse%
506     }%
507   }%
508   \hyphenchar\font=\value{tw@hyphen@char@num}\relax
509   \color{\usemenucolor{txt}}%
510   \CurrentMenuElement}%
511 }[%]
512   \hspace{0.2em plus 0.1em}%
513   \raisebox{0.08ex}{%
514     \tikz{\fill[\usemenucolor{c}] (0,0) -- (0.5ex,0.5ex)%
515       -- (0,1ex) -- cycle;}%
516   }%
517   \hspace{0.2em plus 0.1em}%
518 ]{blacknwhite}
519
520 \NewDocumentCommand{\drawtikzfolder}{O{white} O{black}}{%
521   \begin{tikzpicture}[rounded corners=0.02ex,scale=0.7]
522     \draw [#2] (0,0) -- (1em,0) -- (1em,1.5ex) -- (0.5em,1.5ex) -- %
523       (0.4em,1.7ex) -- (0.1em,1.7ex) -- (0,1.5ex) -- cycle;
524     \draw [#2,fill=#1] (0,0) -- (1em,0) -- (0.85em,1.15ex) -- %
525       ++(-1em,0) -- cycle;
526   \end{tikzpicture}%
527 }
528
529 \copymenustyle{pathswithfolder}{paths}
530 \changemenuelement{pathswithfolder}{pre}{%
531   \drawtikzfolder[\usemenucolor{a}][\usemenucolor{b}]%

```

```

532   \hspace{0.2em plus 0.1em}%
533 }
534
535 \copymenustyle{pathswithblackfolder}{paths}
536 \changemenuelement{pathswithblackfolder}{pre}{%
537   \drawtikzfolder[\usemenucolor{c}][\usemenucolor{b}]%
538   \hspace{0.2em plus 0.1em}%
539 }
540
541 \copymenustyle{hyphenatepathswithfolder}{hyphenatepaths}
542 \changemenuelement{hyphenatepathswithfolder}{pre}{%
543   \drawtikzfolder[\usemenucolor{a}][\usemenucolor{b}]%
544   \hspace{0.2em plus 0.1em}%
545 }
546
547 \copymenustyle{hyphenatepathswithblackfolder}{hyphenatepaths}
548 \changemenuelement{hyphenatepathswithblackfolder}{pre}{%
549   \drawtikzfolder[\usemenucolor{c}][\usemenucolor{b}]%
550   \hspace{0.2em plus 0.1em}%
551 }

```

6.6 Menu macros

6.6.1 Internal commands

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

\CurrentMenuElement and the \CurrentMenuElement dummy
553 \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!
554 \begingroup
555 \lccode`\.=1
556 \lowercase{\endgroup
557 \robust@def*\tw@mk@test@input@sep#1{%
558 \xifinsetTF{,\cpttrimspaces{#1},}{,bslash,backslash,directory,location,}%
559 }%
560 }
561 \NewDocumentCommand{\tw@define@menu@macro}{%
562 m O{\tw@default@input@sep} m
563 }{%
564 \ifcsundef{tw@style@#3@sep}{%
565 \tw@mk@error{Can't define menu macro \string#1\space,\MessageBreak
566 because the style '#3' is not available!}%
567 }{%
568 \csdef{tw@parse@menu@list@expandafter@gobble\string#1}##1{%

```

569     \iflastinndris
570         \ifnum\indrisnr=\@ne
571             \def\CurrentMenuElement{##1}%
572             \cnameuse{tw@style@#3@singl}%
573         \else
574             \def\CurrentMenuElement{##1}%
575             \cnameuse{tw@style@#3@sep}\cnameuse{tw@style@#3@last}%
576         \fi
577     \else
578         \ifnum\indrisnr=\@ne
579             \def\CurrentMenuElement{##1}%
580             \cnameuse{tw@style@#3@first}%
581         \else
582             \def\CurrentMenuElement{##1}%
583             \cnameuse{tw@style@#3@sep}\cnameuse{tw@style@#3@mid}%
584         \fi
585     \fi
586 }
587 \expandafter\newcommand\csname\expandafter\@gobble\string#1\endcsname[2][#2]{%
588     \leavevmode%
589     {\def\tw@current@color@theme{\csname tw@style@#3@color@theme\endcsname}%
590     \cnameuse{tw@style@#3@pre}%
591     \tw@mk@test@input@sep{##1}{%
592         \edef\tw@menu@list{\detokenize{##2}}\edef\tw@mk@tempa{\@backslashchar}%
593     }{%
594         \edef\tw@menu@list{\unexpanded{##2}}\edef\tw@mk@tempa{\cpttrimspaces{##1}}%
595     }%
596     \letcs{\tw@mk@tempb}{\tw@parse@menu@list@\expandafter\@gobble\string#1}%
597     \cptexpanded{\indrisloop*[\tw@mk@tempa]\tw@menu@list\tw@mk@tempb}%
598     \cnameuse{tw@style@#3@post}%
599     }%
600 }
601 }
602 \edef\cpt@parserlist{\cpt@parserlist\@backslashchar}

```

6.6.2 User-level commands

```

\newmenumacro Now it's time to build the user-level commands
\renewmenumacro 603 \NewDocumentCommand{\newmenumacro}{m 0{\tw@default@input@sep} m}{%
\providemenumacro 604     \ifcsundef{\expandafter\@gobble\string#1}{%
605         \tw@define@menu@macro{#1}[#2]{#3}%
606         \expandafter\cptrobustify\csname\expandafter\@gobble\string#1\endcsname
607     }{%
608         \tw@mk@error{Menu macro '\string#1' already defined! \MessageBreak
609         Use \string\renewmenustyle\space instead.}
610     }%
611 }
612 \NewDocumentCommand{\renewmenumacro}{m 0{\tw@default@input@sep} m}{%
613     \cslet{\expandafter\@gobble\string#1}{\relax}%

```

```

614     \tw@define@menu@macro{#1}{#2}{#3}%
615 }
616 \NewDocumentCommand{\providemenumacro}{m O{\tw@default@input@sep} m}{%
617     \ifcsundef{\expandafter\gobble\string#1}{%
618         \tw@define@menu@macro{#1}{#2}{#3}%
619     }{%
620         \tw@mk@warning{Menu macro '\string#1' already defined!}\MessageBreak
621         Use \string\renewmenustyle\space to redefine it.%
622     }%
623 }

```

6.6.3 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:

```
624 \iftw@mk@definemenumacros
```

```
\menu And then we define three basic macros.
\directory 625 \newmenumacro{\menu}{>}{menus}
\keys 626 \newmenumacro{\directory}{/}{paths}
627 \newmenumacro{\keys}{+}{roundedkeys}
```

Lastly we close the `definemacros` if statement:

```
628 \fi
```

6.7 Keys

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

```
629 \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`.

```

630 \NewDocumentCommand{\tw@make@key@box}{m m}{%
631 %   \expandafter\newbox\csname tw@mk@box@#1\endcsname
632 %   \expandafter\sbox\csname tw@mk@box@#1\endcsname{%
633 %       #2%
634 %   }%
635 %   \csdef{tw@mk@#1}{%
636 %       \expandafter\usebox\csname tw@mk@box@#1\endcsname%
637 %       #2%
638 %   }%
639 }
```

\tw@make@key@macro

```

640 \NewDocumentCommand{\tw@make@key@macro}{s m}{%
641   \IfBooleanTF{#1}{%
642     \expandafter\providecommand\csname\expandafter\gobble\string#2\endcsname{%
643       \expandonce{\maxsizebox{!}{1.8ex}}{%
644         \nameuse{\tw@mk@\expandafter\gobble\string#2@tw@mk@os}}}{%
645       }%
646     }%
647     \expandafter\providecommand\csname\expandafter\gobble\string#2mac\endcsname{%
648       \expandonce{\maxsizebox{!}{1.8ex}}{%
649         \nameuse{\tw@mk@\expandafter\gobble\string#2@mac}}}{%
650       }%
651     }%
652     \expandafter\providecommand\csname\expandafter\gobble\string#2win\endcsname{%
653       \expandonce{\maxsizebox{!}{1.8ex}}{%
654         \nameuse{\tw@mk@\expandafter\gobble\string#2@win}}}{%
655       }%
656     }%
657   }{%
658     \expandafter\providecommand\csname\expandafter\gobble\string#2\endcsname{%
659       \expandonce{\maxsizebox{!}{1.8ex}}{%
660         \nameuse{\tw@mk@\expandafter\gobble\string#2}}}{%
661       }%
662     }%
663   }%
664 }

```

\tw@define@mackey

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

```

665 \newcommand*{\tw@define@mackey}[2]{%
666   \IfStrEq{text}{\tw@mk@mackeys}{#1}{%
667     \IfStrEq{symbols}{\tw@mk@mackeys}{#2}{%
668   }%
669 }

```

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

```

670 \tikzset{
671   menukeys key symbol/.style= {
672     rounded corners=0pt,
673     line width=0.1ex,
674     baseline={(0,0)},
675   },
676   menukeys thick/.style={line width=0.25ex},
677 }

```

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
678 \normalsize
679 \tw@make@key@box{\shift}{%
680   \begin{tikzpicture}[yshift=-0.1ex,menukeys key symbol]
681     \draw (0.3ex,0) -- (1.1ex,0) -- (1.1ex,1.2ex) -- %
682       (1.5ex,1.2ex) -- (0.7ex,1.9ex) -- (-0.1ex,1.2ex) -- %
683       (0.3ex,1.2ex) -- cycle;
684   \end{tikzpicture}%
685 }
686 \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 `\langle key` that depends on the `os` option, `\langle key\rangle mac` and `\langle key\rangle win`, that are not affected by `os`.

```

\capslock
687 \tw@make@key@box{\capslock@mac}{%
688   \begin{tikzpicture}[yshift=-0.1ex,menukeys key symbol]
689     \draw (0.3ex,0.7ex) -- (1.1ex,0.7ex) -- (1.1ex,1.2ex) -- %
690       (1.5ex,1.2ex) -- (0.7ex,1.9ex) -- (-0.1ex,1.2ex) -- %
691       (0.3ex,1.2ex) -- cycle;
692     \draw (0.3ex,0) rectangle (1.1ex,0.4ex);
693   \end{tikzpicture}%
694 }
695 \tw@make@key@box{\capslock@win}{%
696   \begin{tikzpicture}[yscale=-1,yshift=-1.8ex,menukeys key symbol]
697     \draw (0.3ex,0) -- (1.1ex,0) -- (1.1ex,1.2ex) -- %
698       (1.5ex,1.2ex) -- (0.7ex,1.9ex) -- (-0.1ex,1.2ex) -- %
699       (0.3ex,1.2ex) -- cycle;
700   \end{tikzpicture}%
701 }
702 \tw@make@key@macro*{\capslock}

```

Here are the other macros:

```

\tab
703 \tw@make@key@box{\tab@mac}{%
704   \begin{tikzpicture}[yshift=0.6ex,menukeys key symbol]
705     \draw [->] (0,0) -- (1em,0);
706     \draw (1em,-0.35ex) -- (1em,0.35ex);
707   \end{tikzpicture}%
708 }
709 \tw@make@key@box{\tab@win}{%
710   \begin{tikzpicture}[yshift=0.1ex,menukeys key symbol]
711     \draw [->] (0.2em,0) -- (1.2em,0);
712     \draw (1.2em,-0.35ex) -- (1.2em,0.35ex);
713     \draw [<-] (0,1ex) -- (1em,1ex);
714     \draw (0,0.65ex) -- (0,1.35ex);
715   \end{tikzpicture}%

```

```

716 }
717 \tw@make@key@macro*\{\tab}

\esc
\oldesc 718 \def\tw@mk@esc@win{Esc}
719 \tw@define@mackey{%
720   \def\tw@mk@esc@mac{esc}
721 }{%
722   \tw@make@key@box{esc@mac}{%
723     \begin{tikzpicture}[yshift=-0.1ex,menukeys key symbol]
724       \draw [-] (0.5ex,0.5ex) -- +(135:1.1ex);
725       \draw (0.5ex,0.5ex) +(105:0.6ex) arc (105:-195:0.6ex);
726     \end{tikzpicture}%
727   }%
728 }
729 \tw@make@key@macro*\{\esc}
730 \def\tw@mk@oldesc@win{Esc}
731 \tw@define@mackey{%
732   \def\tw@mk@oldesc@mac{esc}
733 }{%
734   \tw@make@key@box{oldesc@mac}{%
735     \begin{tikzpicture}[yshift=-0.1ex,menukeys key symbol]
736       \draw [-] (0.5ex,0.5ex) -- +(45:1.1ex);
737       \draw (0.5ex,0.5ex) +(15:0.6ex) arc (15:-285:0.6ex);
738     \end{tikzpicture}%
739   }%
740 }
741 \tw@make@key@macro*\{\oldesc}

\ctrl
742 \providecommand\ctrlname{Ctrl}
743 \def\tw@mk@ctrl@win{\ctrlname}
744 \def\tw@mk@ctrl@mac{ctrl}
745 \tw@make@key@macro*\{\ctrl}

\Alt
\AltGr 746 \def\tw@mk@Alt@win{Alt}
747 \tw@define@mackey{%
748   \def\tw@mk@Alt@mac{alt}%
749 }{%
750   \tw@make@key@box{Alt@mac}{%
751     \begin{tikzpicture}[yshift=-0.1ex,menukeys key symbol]
752       \draw (0,1ex) -- (0.5ex,1ex) -- (1ex,0.3ex) -- (1.8ex,0.3ex);
753       \draw (0.8ex,1ex) -- (1.8ex,1ex);
754     \end{tikzpicture}%
755   }%
756 }
757 \tw@make@key@macro*\{\Alt}
758 \providecommand*\{\AltGr\}{Alt\,,Gr}

```

```

\cmd
759 \def\tw@mk@cmd@win{%
760   \tw@mk@warning{'\string\cmd' only for Mac!}%
761 }
762 \tw@define@mackey{%
763   \def\tw@mk@cmd@mac{\cmd}%
764 }{%
765   \tw@make@key@box{\cmd@mac}{%
766     \begin{tikzpicture}[yshift=-0.15ex,menukeys key symbol]
767       \draw (0.5ex,0.7ex) -- (0.5ex,1.25ex) arc (0:270:0.25ex) -- %
768         (1.25ex,1ex) arc (-90:180:0.25ex) -- (1ex,0.25ex) %
769         arc (-180:90:0.25ex) -- (0.25ex,0.5ex) arc (90:360:0.25ex) %
770         -- cycle;
771     \end{tikzpicture}%
772   }%
773 }
774 \tw@make@key@macro*\{\cmd\}

\Space
\SPACE 775 \providecommand*{\Space}{\expandonce{\rule{3em}{0pt}}}
776 \newcommand{\spacename}{Space}
777 \providecommand*{\SPACE}{\expandonce{\rule{2em}{0pt}\spacename\rule{2em}{0pt}}}

\return
778 \tw@make@key@box{return@mac}{%
779   \begin{tikzpicture}[yshift=0.25ex,menukeys key symbol]
780     \draw [->, rounded corners=0.2ex] (1.25ex,1ex) -| %
781       (2ex,0) -- (0,0);
782   \end{tikzpicture}%
783 }
784 \tw@make@key@box{return@win}{%
785   \begin{tikzpicture}[menukeys key symbol]
786     \draw [->] (1ex,1.25ex) |- (0,0);
787   \end{tikzpicture}%
788 }
789 \tw@make@key@macro*\{\return\}

\enter
790 \def\tw@mk@enter@win{Enter}
791 \tw@make@key@box{enter@mac}{%
792   \begin{tikzpicture}[menukeys key symbol]
793     \draw (0,0) -- (0.5ex,0.5ex) -- (1ex,0);
794     \draw (0,0.55ex) -- (1ex,0.55ex);
795   \end{tikzpicture}%
796 }
797 \tw@make@key@macro*\{\enter\}

\winmenu
798 \def\tw@mk@winmenu@mac{%

```

```

799      \tw@mk@warning{'\string\winmenu' only for Windows!}%
800 }
801 \tw@make@key@box{winmenu@win}{%
802   \begin{tikzpicture}[yshift=-0.2ex,menukeys key symbol]
803     \draw (0,0) rectangle (1.5ex,1.8ex);
804     \draw (0.25ex,1.4ex) -- ++(1ex,0);
805     \draw (0.25ex,1ex) -- ++(1ex,0);
806     \draw (0.25ex,0.6ex) -- ++(1ex,0);
807   \end{tikzpicture}%
808 }
809 \tw@make@key@macro*\{\winmenu}

\backspace

810 \tw@make@key@box{backspace}{%
811   \begin{tikzpicture}[yshift=0.65ex,menukeys key symbol]
812     \draw [<-,menukeys thick] (0,0) -- (1.35em,0);
813   \end{tikzpicture}%
814 }
815 \tw@make@key@macro{\backspace}

\del

\backdel 816 \providecommand{\delname}{\text{Del.}}
817 \def\tw@mk@del@win{\delname}
818 \tw@define@mackey{%
819   \def\tw@mk@del@mac{\delname}%
820 }{%
821   \tw@make@key@box{\del@mac}{%
822     \begin{tikzpicture}[yshift=0.2ex,menukeys key symbol]
823       \draw (0,0) -- (1.5ex,0) -- (2ex,0.5ex) --%
824         (1.5ex,1ex) -- (0,1ex) -- cycle;
825       \draw (0.5ex,0.2ex) -- (1.1ex,0.8ex);
826       \draw (0.5ex,0.8ex) -- (1.1ex,0.2ex);
827     \end{tikzpicture}%
828   }%
829 }
830 \tw@make@key@macro*\{\del}
831 \def\tw@mk@backdel@win{\delname}
832 \tw@define@mackey{%
833   \def\tw@mk@backdel@mac{\delname}%
834 }{%
835   \tw@make@key@box{\backdel@mac}{%
836     \begin{tikzpicture}[yshift=0.2ex,menukeys key symbol]
837       \draw (2ex,0) -- (0.5ex,0) -- (0,0.5ex) --%
838         (0.5ex,1ex) -- (2ex,1ex) -- cycle;
839       \draw (1ex,0.2ex) -- (1.6ex,0.8ex);
840       \draw (1ex,0.8ex) -- (1.6ex,0.2ex);
841     \end{tikzpicture}%
842   }%
843 }
844 \tw@make@key@macro*\{\backdel}

```

```

\arrowkeyup Lastly we define the arrow macros:
\arrowkeydown
\arrowkeyleft
\arrowkeyright
845 \tw@make@key@box{\arrowkeyup}{%
846   \begin{tikzpicture}[yshift=-0.2ex,menukeys key symbol]
847     \draw [->] (0,0) -- (0,0.8em);
848   \end{tikzpicture}%
849 }
850 \tw@make@key@macro{\arrowkeyup}
851
852 \tw@make@key@box{\arrowkeydown}{%
853   \begin{tikzpicture}[yshift=0.7em,menukeys key symbol]
854     \draw [->] (0,0) -- (0,-0.8em);
855   \end{tikzpicture}%
856 }
857 \tw@make@key@macro{\arrowkeydown}
858
859 \tw@make@key@box{\arrowkeyright}{%
860   \begin{tikzpicture}[yshift=0.5ex,menukeys key symbol]
861     \draw [->] (0,0) -- (0.8em,0);
862   \end{tikzpicture}%
863 }
864 \tw@make@key@macro{\arrowkeyright}
865
866 \tw@make@key@box{\arrowkeyleft}{%
867   \begin{tikzpicture}[yshift=0.5ex,menukeys key symbol]
868     \draw [->] (0,0) -- (-0.8em,0);
869   \end{tikzpicture}%
870 }
871 \tw@make@key@macro{\arrowkeyleft}

\arrowkey And the \arrowkey macro that get's it's direction as argument.
872 \newcommand{\arrowkey}[1]{%
873   \IfStrEq{^}{#1}{\arrowkeyup}{%
874     \IfStrEq{v}{#1}{\arrowkeydown}{%
875       \IfStrEq{<}{#1}{\arrowkeyleft}{%
876         \IfStrEq{>}{#1}{\arrowkeyright}{%
877           \tw@mk@error{Wrong value '#1' for \string\arrowkey\MessageBreak
878             Possible values are '^', 'v', '<' or '>'}%
879         }%
880       }%
881     }%
882   }%
883 }

Close the \iftw@mk@definekeys
884 \fi

```

7 Change history

v1.0		Fixed GitHub issues #9, #10, #11, #13, #17, #24 and #26 1
General: Initial version 1		
v1.1		Tidy up version and date 1
\directory: Renamed \path to \directory because it crashes with biblatax 29		
General: Improved manual 1		
Load xcolor before menukeys. 14		
v1.1a		
\newmenumacro: Added a line to make a new macro robust. 28		
\tw@define@menu@macro: Fixed minor bug, that causes a warning about robustifying (issue #23), by deleting the line to make the command robust. 27		
v1.2		
\tw@define@menu@macro: Added \leavevmode 27		
Replaced \edef by \protected@edef 27		
General: Added \normalsize before symbol definitions to make the ex unit available 1		
Added \SPACE and \spacename 1		
v1.2a		
General: Added braces to the \tikz macro since the parser seems to crash with babel's french option otherwise. 1		
Replaced obsolete \tikzstyle 1		
v1.2c		
\tw@define@menu@macro: Replaced \protected@edef by \def 27		
v1.3		
General: Added TikZ-styles for the key symbols. 1		
Improved key symbols. 1		
v1.4		
\backdel: Added \backdel 34		
\oldesc: Fixed direction of \escmac; added \oldesc 32		
General: Extended color theme features. 1		
The path... styles now use the text color of the selected color theme (fix issue #16). 1		

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.

A	B
\Alt 746	\arrowkeyleft 845, 875
\AltGr 746	\arrowkeyright 845, 876
angularkeys (style) 6	\arrowkeyup 845, 873
angularmenus (style) 5	
\arrowkey 13, 872	\backdel 816
\arrowkeydown 845, 874	\backspace 810
	blacknwhite (theme) 11
C	D
	\capslock 687
	\changemenucolor 12, 62

\changemenucolortheme	10, 83	\iftw@mk@definekeys	629
\changemenuelement	10, 252, 530, 536, 542, 548	\iftw@mk@definemenumacros	624
\cmd	759		
\color	265, 293, 298, 302, 305, 324, 329, 333, 336, 354, 359, 363, 366, 382, 406, 426, 449, 470, 472, 481, 509		
Color themes:			
blacknwhite	11	\keys	4, 625
gray	11		
\copymenucolortheme	12, 62		
\copymenustyle	10, 234, 529, 535, 541, 547	M	
\ctrl	742	mackeys (option)	4, 13
\ctrlname	13, 742, 743	\menu	4, 625
\CurrentMenuElement	10, 120, 123, 126, 129, 152, 155, 158, 161, 265, 293, 298, 302, 305, 324, 329, 333, 336, 354, 359, 363, 366, 382, 406, 426, 449, 465, 481, 510, 553, 571, 574, 579, 582	menus (style)	5
D			
definekeys (option)	4	N	
definemenumacros (option)	4	\newmenucolortheme	11, 51, 98, 99
\del	816	\newmenumacro	12, 603, 625, 626, 627
\delname	13, 816, 817, 819, 831, 833	\newmenustyle	9, 165, 200
\directory	4, 625	\newmenustylesimple	9, 165, 165
\drawtikzfolder	9, 520, 531, 537, 543, 549		
E		O	
\enter	790	\oldesc	718
\esc	718	Options:	
F		definekeys	4
\font	508	definemenumacros	4
G		mackeys	4, 13
gray (theme)	11	os	4
H		os (option)	4
hyphenatepaths (style)	8		
hyphenatepathswithblackfolder		P	
(style)	8	paths (style)	7
hyphenatepathswithfolder	8	pathswithblackfolder (style)	7
I		pathswithfolder (style)	7
\if@tw@hyphenatepaths@warnig	492, 502	\providemenumacro	13, 603
		\providemenustyle	11, 165, 220
		\providemenustylesimple	11, 165, 185
R			
		relsize	285, 316, 346, 377, 385, 397, 410, 421, 429, 440, 452, 461, 476
		\renewmenucolortheme	12, 51, 80
		\renewmenumacro	13, 603
		\renewmenustyle	11, 165, 209, 213, 230, 609, 621
		\renewmenustylesimple	11, 165, 174, 178, 195
		\return	778
		roundedkeys (style)	6
		roundedmenus (style)	5
S			
		shadowedangularkeys (style)	6
		shadowedroundedkeys (style)	6
		\shift	678

\SPACE	775	745, 757, 774, 789, 797, 809,
\Space	775	815, 830, 844, 850, 857, 864, 871
\spacename	13, 776, 777	\tw@menu@list 592, 594, 597
Styles:		\tw@mk@Alt@mac 748
angularkeys	6	\tw@mk@Alt@win 746
angularmenus	5	\tw@mk@backdel@mac 833
hyphenatepathswithblackfolder	8	\tw@mk@backdel@win 831
hyphenatepathswithfolder	8	\tw@mk@cmd@mac 763
hyphenatepaths	8	\tw@mk@cmd@win 759
menus	5	\tw@mk@ctrl@mac 744
pathswithblackfolder	7	\tw@mk@ctrl@win 743
pathswithfolder	7	\tw@mk@del@mac 819
paths	7	\tw@mk@del@win 817
roundedkeys	6	\tw@mk@center@win 790
roundedmenus	5	\tw@mk@error 10, 36,
shadowedangularkeys	6	40, 55, 66, 79, 85, 89, 173, 208,
shadowedroundedkeys	6	237, 249, 254, 268, 565, 608, 877
typewriterkeys	7	\tw@mk@esc@mac 720
		\tw@mk@esc@win 718
		\tw@mk@gobble@args 21, 175, 196, 210, 231
T		\tw@mk@mackey 39, 666, 667
\tab	703	\tw@mk@olddesc@mac 732
\tikzset	272, 277, 308, 339, 369, 389, 414, 433, 456, 670	\tw@mk@olddesc@win 730
\tw@current@color@theme	96, 589	\tw@mk@os 35, 644
\tw@current@style	135, 136, 141	\tw@mk@tempa 19, 22, 23, 592, 594, 597
\tw@declare@style	138, 203, 205, 215, 217, 223, 225, 290, 321, 351	\tw@mk@tempb 19, 596, 597
\tw@declare@style@simple	105, 168, 170, 180, 182, 188, 190, 379, 403, 423, 446, 463, 480, 494	\tw@mk@test@input@sep 557, 591
\tw@declare@sytle	132	\tw@mk@warning 10, 193, 228, 503, 620, 760, 799
\tw@declare@sytle@extra@args	132	\tw@mk@warning@noline 10
\tw@default@input@sep 552, 562, 603, 612, 616	\tw@mk@winmenu@mac 798
\tw@default@post	100, 106, 133	\tw@set@tikz@colors 272
\tw@default@pre	100, 106, 139	\tw@typewriterkeys@curr@elem 464, 470, 472
\tw@default@sep	100, 106, 139	typewriterkeys (style) 7
\tw@define@mackey 665, 719, 731, 747, 762, 818, 832	U
\tw@define@menu@macro 554, 605, 614, 618	\usemenucolor 10, 95, 265, 273, 274, 275, 293, 298, 302, 305, 324, 329, 333, 336, 354, 359, 363, 366, 382, 385, 400, 406, 410, 426, 429, 443, 449, 452, 470, 472, 476, 481, 485, 509, 514, 531, 537, 543, 549
\tw@make@key@macro 640, 686, 702, 717, 729, 741,	W
		\winmenu 798