On the Toggle Switch Widget Package Tsw

by

Csaba Nemethi

csaba.nemethi@t-online.de

Contents

---

1. Abstract

Tsw is a **t**oggle **sw**itch widget package for Tcl/Tk 8.6 or higher, written in pure Tcl code. For version 8.6 (only) it requires that the tksvg extension can be loaded into the interpreter.

A **toggleswitch**, created with the **`tsw::toggleswitch`** command, is a mega-widget consisting of a horizontal **trough** (a fully rounded filled rectangle) and a **slider** (a filled circle contained in the trunk). It can have one of two possible **switch state**s: on or off. In the on state the slider is placed at the end of the trough, and in the off state at its beginning. The user can toggle between these two states with the mouse or the space key.

After a short description of the strait-forward Tsw API, the talk presents two demo scripts illustrating the typical steps needed to create and handle toggleswitch widgets used to trigger actions tied to boolean settings.

The talk concludes with a few tips related to choosing between toggleswitch and (ttk::)checkbutton widgets.

---

2. The tsw::toggleswitch Command

NAME

`tsw::toggleswitch` – Create and manipulate toggle switch widgets

SYNOPSIS

```
tsw::toggleswitch pathName ?options?
```

DESCRIPTION

A **toggleswitch** is a mega-widget consisting of a horizontal **trough** and a **slider**. The trough is a fully rounded filled rectangle, and the slider is a filled circle contained in the trough. Both elements are rendered using scaling-aware SVG images. Their dimensions depend on the display's scaling level, the current theme, and the value of the **`-size`** configuration option.

A toggleswitch widget can have one of two possible **switch state**s: on or off. In the on state the slider is placed at the end of the trough, and in the off state at its beginning. The user can toggle between these two states with the mouse or the `space` key.

You can use the **`switchstate`** subcommand to change or query the widget's switch state. By using the **`-command`** configuration option, you can specify a script to execute whenever the widget's switch state gets toggled. For compatibility with the (ttk::)checkbutton, toggleswitch widgets also support the **`-offvalue`**, **`-onvalue`**, and **`-variable`** options.

STANDARD OPTIONS

```
-cursor
```

WIDGET-SPECIFIC OPTIONS

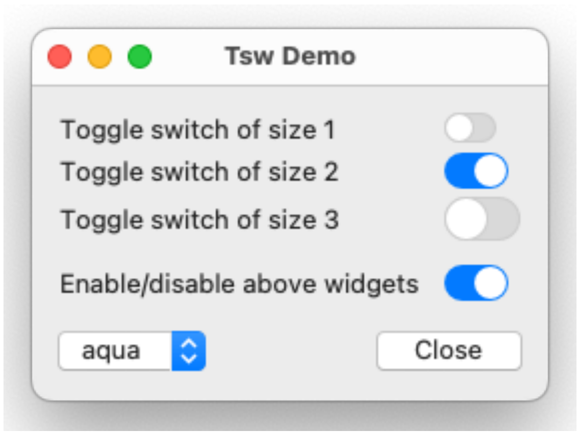
`-command` *`command`*
`-offvalue` *`value`*
`-onvalue` *`value`*
`-size 1|2|3`
`-takefocus 0|1|""``|`*`command`*
`-variable` *`variable`*

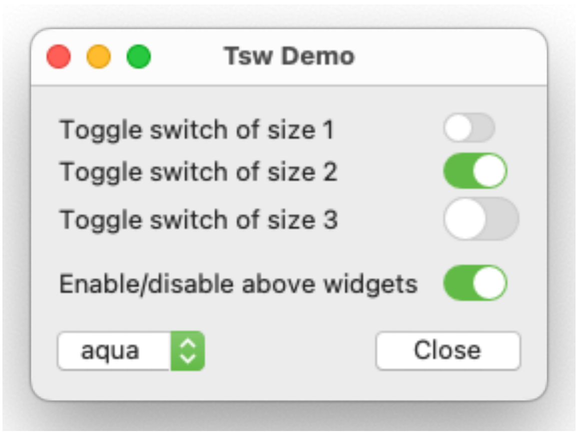
WIDGET COMMAND

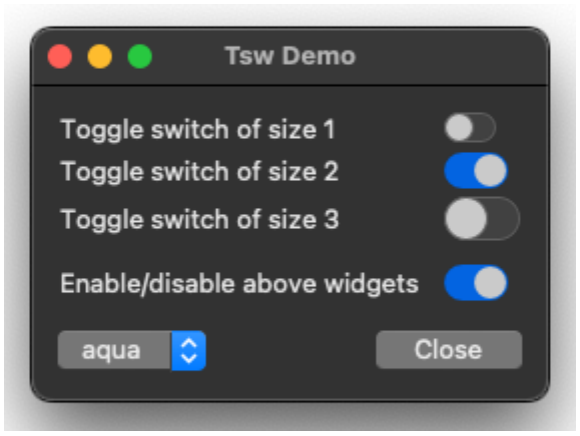
`pathName` **`attrib`** `?`*`name ?value name value ...??`*
`pathName` **`cget`** *`option`*
`pathName` **`configure`** `?`*`option ?value option value ...??`*
`pathName` **`hasattrib`** *`name`*
`pathName` **`identify`** `?`**`element`**`?` *`x y`*
`pathName` **`instate`** *`stateSpec ?script?`*
`pathName` **`state`** `?`*`stateSpec`*`?`
`pathName` **`style`**
`pathName` **`switchstate`** `?`*`boolean`*`?`
`pathName` **`toggle`**
`pathName` **`unsetattrib`** *`name`*

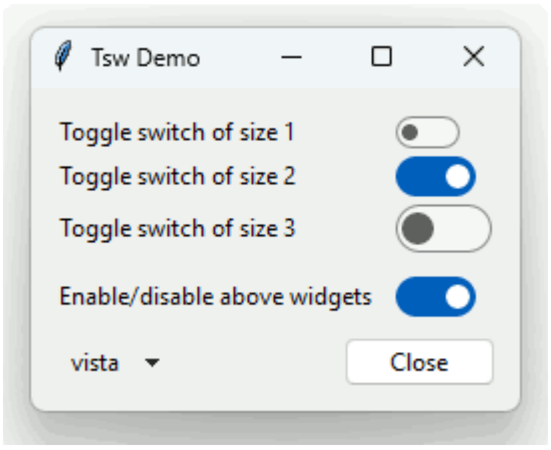
---

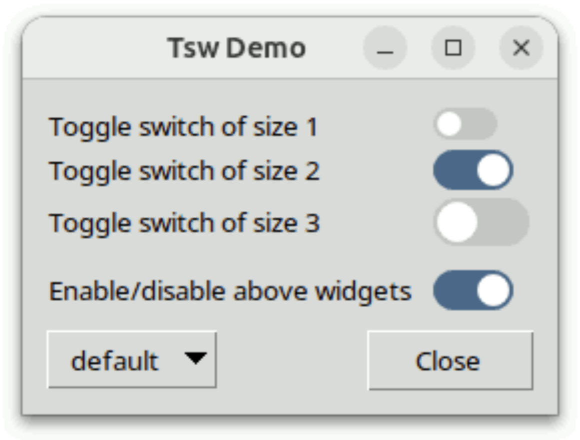
3. Tsw Demo

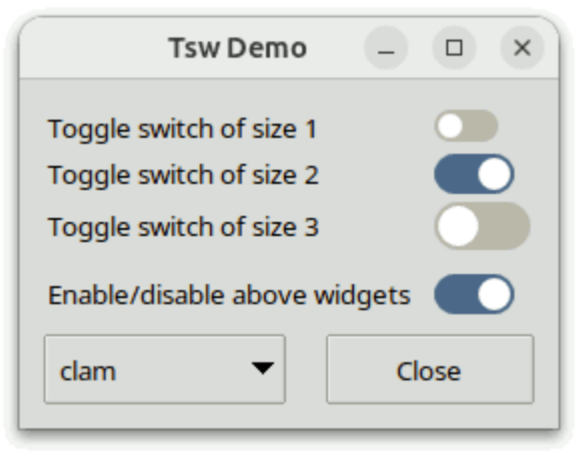

Tsw Demo
Toggle switch of size 1
Toggle switch of size 2
Toggle switch of size 3
Enable/disable above widgets
aqua
Close

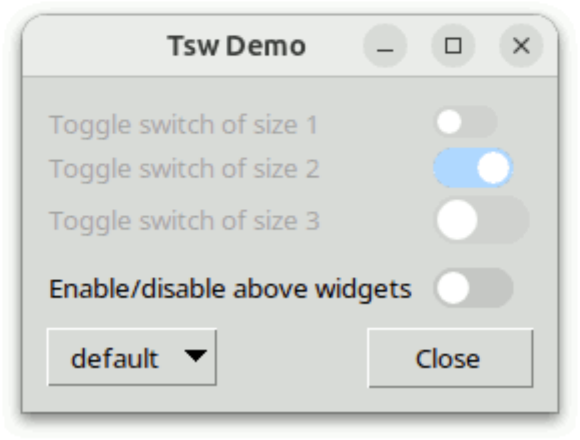

Tsw Demo
Toggle switch of size 1
Toggle switch of size 2
Toggle switch of size 3
Enable/disable above widgets
aqua
Close


Tsw Demo
Toggle switch of size 1
Toggle switch of size 2
Toggle switch of size 3
Enable/disable above widgets
aqua
Close


Tsw Demo
Toggle switch of size 1
Toggle switch of size 2
Toggle switch of size 3
Enable/disable above widgets
vista
Close

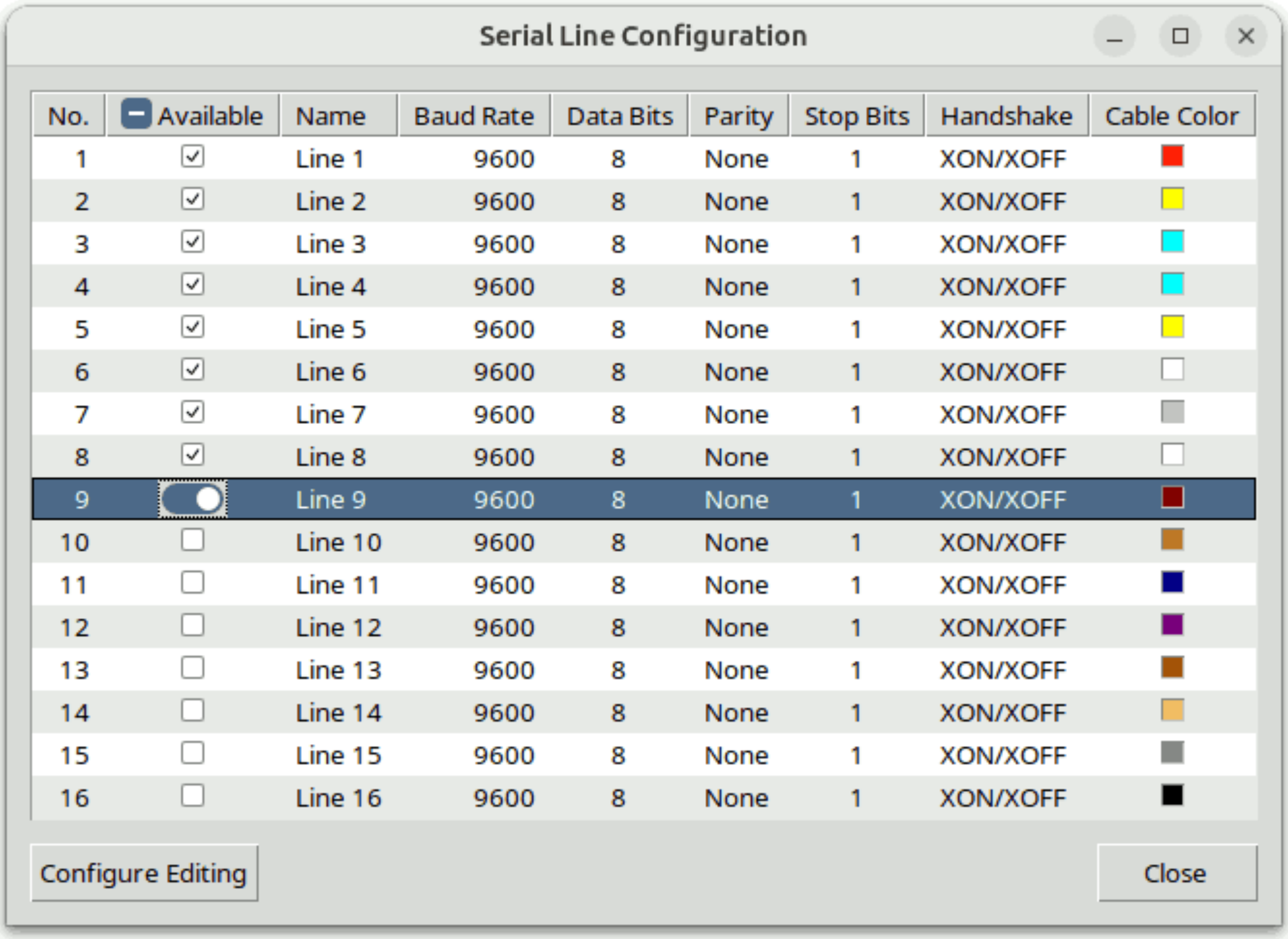

Tsw Demo
Toggle switch of size 1
Toggle switch of size 2
Toggle switch of size 3
Enable/disable above widgets
default
Close


Tsw Demo
Toggle switch of size 1
Toggle switch of size 2
Toggle switch of size 3
Enable/disable above widgets
clam
Close


Tsw Demo
Toggle switch of size 1
Toggle switch of size 2
Toggle switch of size 3
Enable/disable above widgets
default
Close

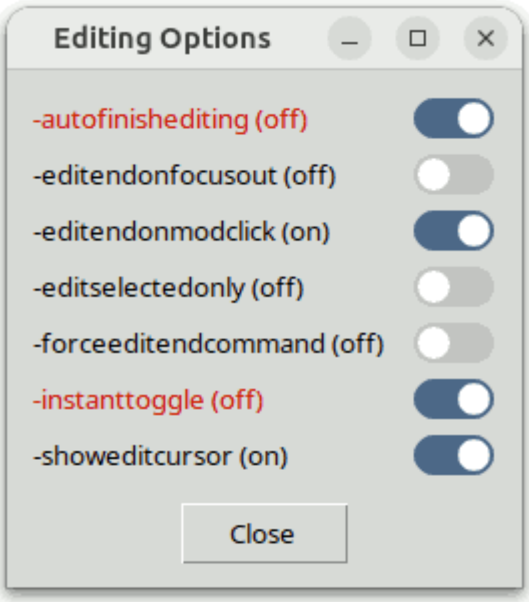
```
package require Tk
package require tsw

wm title . "Tsw Demo"

ttk::frame .tf
ttk::frame .bf

#
# Create 3 toggleswitch widgets having different values of the -size option
#
set l1 [ttk::label .tf.l1 -text "Toggle switch of size 1"]
set sw1 [tsw::toggleswitch .tf.sw1 -size 1]
set l2 [ttk::label .tf.l2 -text "Toggle switch of size 2"]
set sw2 [tsw::toggleswitch .tf.sw2 -size 2]
$sw2 switchstate 1
set l3 [ttk::label .tf.l3 -text "Toggle switch of size 3"]
set sw3 [tsw::toggleswitch .tf.sw3 -size 3]

#
# Create a toggleswitch widget of default size and set its -command option
#
set l4 [ttk::label .tf.l4 -text "Enable/disable above widgets"]
set sw4 [tsw::toggleswitch .tf.sw4]
$sw4 switchstate 1
$sw4 configure -command [list toggleWidgetsState $sw4]

. . .

#------------------------------------------------------------------------------
# toggleWidgetsState
#
# Enables/disables the widgets in the first 3 grid rows, depending on the
# switch state of the specified toggleswitch widget.
#------------------------------------------------------------------------------
proc toggleWidgetsState sw {
    global l1 l2 l3 sw1 sw2 sw3
    set stateSpec [expr {[$sw switchstate] ? "!disabled" : "disabled"}]
    foreach w [list $l1 $l2 $l3 $sw1 $sw2 $sw3] {
        $w state $stateSpec
    }
}
```

4. Tablelist Editing Options



Serial Line Configuration

No.	☐ Available	Name	Baud Rate	Data Bits	Parity	Stop Bits	Handshake	Cable Color
1	☑	Line 1	9600	8	None	1	XON/XOFF	🟥
2	☑	Line 2	9600	8	None	1	XON/XOFF	🟨
3	☑	Line 3	9600	8	None	1	XON/XOFF	🟦
4	☑	Line 4	9600	8	None	1	XON/XOFF	🟦
5	☑	Line 5	9600	8	None	1	XON/XOFF	🟨
6	☑	Line 6	9600	8	None	1	XON/XOFF	⬜
7	☑	Line 7	9600	8	None	1	XON/XOFF	⬜
8	☑	Line 8	9600	8	None	1	XON/XOFF	⬜
9	●	Line 9	9600	8	None	1	XON/XOFF	🟥
10	☐	Line 10	9600	8	None	1	XON/XOFF	🟧
11	☐	Line 11	9600	8	None	1	XON/XOFF	🟦
12	☐	Line 12	9600	8	None	1	XON/XOFF	🟪
13	☐	Line 13	9600	8	None	1	XON/XOFF	🟫
14	☐	Line 14	9600	8	None	1	XON/XOFF	🟧
15	☐	Line 15	9600	8	None	1	XON/XOFF	⬛
16	☐	Line 16	9600	8	None	1	XON/XOFF	⬛



Editing Options

-autofinishediting (off)	● on
-editendonfocusout (off)	○ off
-editendonmodclick (on)	● on
-editselectedonly (off)	○ off
-forceeditendcommand (off)	○ off
-instanttoggle (off)	● on
-showeditcursor (on)	● on

```
package require Tk
package require tsw
package require tablelist_tile

. . .

#------------------------------------------------------------------------------
# configEditing
#
# Configures the editing-related tablelist options having boolean values with
# the aid of toggleswitch widgets.
#------------------------------------------------------------------------------
proc configEditing tbl {
    set top .top
    if {[winfo exists $top]} {
        raise $top
        focus $top
        return ""
    }

    toplevel $top
    wm title $top "Editing Options"

    set tf [ttk::frame $top.tf]
    set bf [ttk::frame $top.bf]

    #
    # Create the widgets corresponding to the
    # editing-related options with boolean values
    #
    set row 0
    foreach opt {
        -autofinishediting
        -editendonfocusout
        -editendonmodclick
        -editselectedonly
        -forceeditendcommand
        -instanttoggle
        -showeditcursor
    } {
        lassign [$tbl configure $opt] option dbName dbClass default current
        set defaultStr [expr {$default ? "on" : "off"}]

        set l [ttk::label $tf.l$row -text "$opt ($defaultStr)"]
        if {$current != $default} {
            $l configure -foreground red2
        }
        grid $l -row $row -column 0 -sticky w -padx 9p -pady {0 3p}

        set sw [tsw::toggleswitch $tf.sw$row]
        $sw switchstate $current        ;# sets the switch state to $current
        $sw attrib default $default     ;# saves $default as attribute value
        $sw configure -command [list applySwitchState $sw $tbl $opt $l]
        grid $sw -row $row -column 1 -sticky w -padx {0 9p} -pady {0 3p}

        incr row
    }

    . . .
}
```

```
#------------------------------------------------------------------------------
# applySwitchState
#
# Sets the configuration option opt of the tablelist tbl and the foreground
# color of the ttk::label l according to the switch state of the toggleswitch
# widget sw.
#------------------------------------------------------------------------------
proc applySwitchState {sw tbl opt l} {
    set switchState [$sw switchstate]
    $tbl configure $opt $switchState

    set fgColor [expr {$switchState == [$sw attrib default] ? "" : "red2"}]
    $l configure -foreground $fgColor
}
```

5. Choosing between toggleswitch and (ttk::)checkbutton

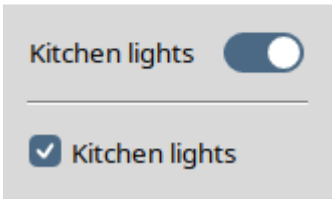
Based on the Microsoft Learn article *Toggle switches*. See

> https://learn.microsoft.com/en-us/windows/apps/design/controls/toggles

For some actions, either a toggleswitch or a checkbutton might work. To decide which control would work better, follow these tips:

- Use a toggleswitch for binary settings that work like actions by taking effect immediately after the user flips the switch.

  In the following example, for turning the kitchen lights on, you should use a toggleswitch rather than a checkbutton.

- For optional ("nice to have") items use checkbuttons.

- Use checkbuttons when the user can select multiple items that are related to a single setting or feature.

- Use a checkbutton when the user has to perform extra steps for changes to be effective. For example, if the user must click a "Submit" or "Next" button to apply changes, use a checkbutton.