Package: tickle (via r-universe)

September 16, 2024

Type Package

Title Easily Build Tcl/Tk UIs

Version 0.1.1

Author mikefc

Maintainer mikefc <mikefc@coolbutuseless.com>

Description Wrap tcltk to make GUI creation easier.

License MIT + file LICENSE

URL https://github.com/coolbutuseless/tickle,
https://coolbutuseless.github.io/package/tickle/

BugReports https://github.com/coolbutuseless/tickle/issues

Encoding UTF-8

LazyData true

RoxygenNote 7.1.2

Imports tcltk

Suggests ggplot2, knitr, magick, rmarkdown

VignetteBuilder knitr

Repository https://coolbutuseless.r-universe.dev

RemoteUrl https://github.com/coolbutuseless/tickle

RemoteRef HEAD

RemoteSha a7dc9204b87cbe0d174a62802c2d6dd35c368a71

Contents

.DollarNames.tic_ui 3
as.character.tic_ui 3
bind_event 4
canvas_arc 5
canvas_clear 7
canvas_image 7

canvas_line . . . 8
canvas_oval . . . 9
canvas_plot . . . 10
canvas_polygon . . . 10
canvas_rect . . . 11
canvas_save . . . 12
canvas_text . . . 12
extract_node_by_id . . . 13
format.reactive_value . . . 14
handle_images . . . 14
launch_idle_func . . . 15
list_font_families . . . 15
list_themes . . . 16
pack_opts . . . 16
popup_color_picker . . . 17
popup_messagebox . . . 17
popup_open_file . . . 18
print.tic_spec . . . 19
print.tic_ui . . . 19
reactive_lgl . . . 20
render_ui . . . 20
set_font_family . . . 21
set_font_scale . . . 21
set_theme . . . 22
tic_button . . . 22
tic_canvas . . . 24
tic_checkbutton . . . 25
tic_combobox . . . 27
tic_label . . . 29
tic_labelframe . . . 30
tic_menu . . . 31
tic_menubutton . . . 32
tic_menuitem . . . 34
tic_notebook . . . 35
tic_panedwindow . . . 36
tic_progressbar . . . 37
tic_radiobutton . . . 38
tic_row . . . 40
tic_separator . . . 41
tic_sizegrip . . . 42
tic_slider . . . 43
tic_spinbox . . . 44
tic_submenu . . . 46
tic_textbox . . . 47
tic_textentry . . . 48
tic_window . . . 49

Index **51**

.DollarNames.tic_ui *Autocomplete helper for* tic_ui *objects*

Description

This autocomplete helper removes 'type' and 'env' from the suggestions, and moves the suggestion of 'ID' to last in the list.

Usage

```
## S3 method for class 'tic_ui'
.DollarNames(x, pattern)
```

Arguments

x	object
pattern	current pattern to match

Details

This is because the user is most often going to be using this object to havigate the element tree, and will rarely need to access anything other than the named child nodes.

as.character.tic_ui *Create a character representation of a* tic_ui *object*

Description

Create a character representation of a tic_ui object

Usage

```
## S3 method for class 'tic_ui'
as.character(x, depth = 0L, ...)
```

Arguments

x	tic_ui object
depth	recusive depth
...	ignored

`bind_event` *Bind a command to a particular event*

Description

The bind command associates R functions with UI events.

Usage

```
bind_event(tag, event, command)

bind_opts(event, command)
```

Arguments

`tag`	the `tic_ui` object
`event`	the event to watch for on this object. The general form of an event is "modifiers-type-detail"
`command`	R function to run when this even occurs.

Details

`bind_opts` is used to define binding events within the UI spec.

After the UI is created (with a call to `win = render_ui(ui_spec)`), events can be bound with `bind_event(...)`.

examples

`"c"` keyboard character 'c'

`"Control-q"` Key combination CTRL+q

`"ButtonPress"` Any button press

`"KeyPress"` Any keypress

`"Double-Button-1"` Double click on Mouse Button number 1

modifiers

Control, Alt, Shift, Lock, Button1-Button5, Mod1-Mod5, Meta Double, Triple, Quadruple

events

Activate, ButtonPress, Button, ButtonRelease, Circulate, CirculateRequest, Configure, ConfigureRequest, Create, Deactivate, Destroy, Enter, Expose, FocusIn, FocusOut, Gravity, KeyPress, Key, KeyRelease, Leave, Map, MapRequest, Motion, MouseWheel, Property, Reparent, ResizeRequest, Unmap, Visibility

variables

Variables available to the command depend upon the event. See https://www.tcl.tk/man/tcl8.6/TkCmd/bind.html for the full list.

some variables are listed here:

- `b` The number of the button that was pressed or released. Valid only for ButtonPress and ButtonRelease events.
- `k` The keycode field from the event. Valid only for KeyPress and KeyRelease events.
- `K` The keysym corresponding to the event, substituted as a textual string. Valid only for KeyPress and KeyRelease events.
- `t` time stamp of the event
- `x,y` indicate the position of the mouse pointer relative to the UI window.
- `X, Y` indicate the position of the mouse pointer in absolute screen coordinates
- `D` This reports the delta value of a MouseWheel event. The delta value represents the rotation units the mouse wheel has been moved. The sign of the value represents the direction the mouse wheel was scrolled.

tcl/tk

See tcl/tk documentation for more information on binding commands to events https://www.tcl.tk/man/tcl8.6/TkCmd/bind.html

Examples

```
## Not run:
# Every mouse press prints coordinates
ui_spec <- tic_window()
win <- render_ui(ui_spec)
bind_event(win, "Button", function(t, x, y, ...) { message(t, ": ", x, ", ", y)})

## End(Not run)
```

---

canvas_arc *Draw an arc on a canvas*

Description

Draw an arc on a canvas

Usage

```
canvas_arc(
  canvas,
  x1,
  y1,
  x2,
  y2,
  start,
  extent,
  style,
  fill,
  outline,
  width,
  dash,
  ...
)
```

Arguments

canvas	a tic_ui 'canvas' element.
x1, y1, x2, y2	he coordinates of two diagonally opposite corners of a rectangular region enclosing the oval that defines the arc
start	starting angle of arc in degrees measured counter-clockwise from the "3 o'clock" position. Value in range [-360, 360]
extent	Size of te angle range occupied by the arc. Value in range [-360, 360]
style	how to draw the arc. One of: pieslice (default), chord, fill
	pieslice the arc's region is defined by a section of the oval's perimeter plus two line segments, one between the center of the oval and each end of the perimeter section
	chord the arc's region is defined by a section of the oval's perimeter plus a single line segment connecting the two end points of the perimeter section
	arc the arc's region consists of a section of the perimeter alone. In thiscase the fill option is ignored.
fill	line colour
outline	outline colour
width	line width e.g. width = 2
dash	Specifies the line's dash pattern. This should be a numeric vector with alternating lengths of "dash" and "space-between-dash". E.g. dash = c(6, 4, 2, 4) produces a dotted-dashed line
...	other line creation options. See https://www.tcl.tk/man/tcl8.6/TkCmd/canvas.html#M26

See Also

Other canvas: canvas_clear(), canvas_image(), canvas_line(), canvas_oval(), canvas_plot(), canvas_polygon(), canvas_rect(), canvas_save(), canvas_text(), tic_canvas()

canvas_clear *Clear the canvas of all objects*

Description

Clear the canvas of all objects

Usage

```
canvas_clear(canvas)
```

Arguments

canvas	a tic_ui 'canvas' element.

See Also

Other canvas: canvas_arc(), canvas_image(), canvas_line(), canvas_oval(), canvas_plot(), canvas_polygon(), canvas_rect(), canvas_save(), canvas_text(), tic_canvas()

canvas_image *Render an image a canvas*

Description

Render an image a canvas

Usage

```
canvas_image(canvas, x, y, image, anchor, ...)
```

Arguments

canvas	a tic_ui 'canvas' element.
x, y	coorindates for image
image	as loaded by load_tkimage() or otherwise manually created with tcltk
anchor	anchor point within image for positioning. default: center Possible values: n, s, e, w, ne, nw, se, sw, center
...	other line creation options. See https://www.tcl.tk/man/tcl8.6/TkCmd/canvas.html#M26

See Also

Other canvas: canvas_arc(), canvas_clear(), canvas_line(), canvas_oval(), canvas_plot(), canvas_polygon(), canvas_rect(), canvas_save(), canvas_text(), tic_canvas()

canvas_line *Draw a line on a canvas*

Description

Draw a line on a canvas

Usage

```
canvas_line(
  canvas,
  xs,
  ys,
  fill,
  width,
  arrow,
  smooth,
  capstyle,
  joinstyle,
  dash,
  ...
)
```

Arguments

canvas	a tic_ui 'canvas' element.
xs, ys	vectors of coordinates
fill	line colour
width	line width e.g. width = 2
arrow	where are arrowheads to be drawn? Default: 'none'. Possible values: 'none', 'first', 'last', 'both'
smooth	should the line be draw as quadratic beziers instead of line segements? logical. default: FALSE
capstyle	Specifies the ways in which caps are to be drawn at the endpoints of the line: Possible values: butt, projecting, or round. If this option is not specified then it defaults to butt. Where arrowheads are drawn the cap style is ignored.
joinstyle	Specifies the ways in which joints are to be drawn at the vertices of the line. Possible values: bevel, miter, or round). If this option is not specified then it defaults to round.
dash	Specifies the line's dash pattern. This should be a numeric vector with alternating lengths of "dash" and "space-between-dash". E.g. dash = c(6, 4, 2, 4) produces a dotted-dashed line
...	other line creation options. See https://www.tcl.tk/man/tcl8.6/TkCmd/canvas.html#M26

See Also

Other canvas: canvas_arc(), canvas_clear(), canvas_image(), canvas_oval(), canvas_plot(), canvas_polygon(), canvas_rect(), canvas_save(), canvas_text(), tic_canvas()

---

canvas_oval *Draw an oval on a canvas*

Description

Draw an oval on a canvas

Usage

```
canvas_oval(canvas, x1, y1, x2, y2, fill, outline, width, dash, ...)
```

Arguments

canvas	a tic_ui 'canvas' element.
x1, y1, x2, y2	corners of rectangle enclosing the oval
fill	line colour
outline	outline colour
width	line width e.g. width = 2
dash	Specifies the line's dash pattern. This should be a numeric vector with alternating lengths of "dash" and "space-between-dash". E.g. dash = c(6, 4, 2, 4) produces a dotted-dashed line
...	other line creation options. See https://www.tcl.tk/man/tcl8.6/TkCmd/canvas.html#M26

See Also

Other canvas: canvas_arc(), canvas_clear(), canvas_image(), canvas_line(), canvas_plot(), canvas_polygon(), canvas_rect(), canvas_save(), canvas_text(), tic_canvas()

canvas_plot *Render a plot to the canvas*

Description

Render a plot to the canvas

Usage

```
canvas_plot(canvas, plot, width, height, x = 0, y = 0, anchor = "nw")
```

Arguments

canvas	a tic_ui 'canvas' element.
plot	object to be plotted. Anything support by ggplot2::ggsave()
width, height	size of plot in pizels
x, y	coorindates for image. default (0, 0)
anchor	anchor point within image for positioning. default: nw Possible values: n, s, e, w, ne, nw, se, sw, center

See Also

Other canvas: canvas_arc(), canvas_clear(), canvas_image(), canvas_line(), canvas_oval(), canvas_polygon(), canvas_rect(), canvas_save(), canvas_text(), tic_canvas()

canvas_polygon *Draw a polygon on a canvas*

Description

Draw a polygon on a canvas

Usage

```
canvas_polygon(
  canvas,
  xs,
  ys,
  fill,
  outline,
  width,
  smooth,
  joinstyle,
  dash,
  ...
)
```

Arguments

canvas	a tic_ui 'canvas' element.
xs	vectors of coordinates
ys	vectors of coordinates
fill	line colour
outline	outline colour
width	line width e.g. width = 2
smooth	should the line be draw as quadratic beziers instead of line segements? logical. default: FALSE
joinstyle	Specifies the ways in which joints are to be drawn at the vertices of the line. Possible values: bevel, miter, or round). If this option is not specified then it defaults to round.
dash	Specifies the line's dash pattern. This should be a numeric vector with alternating lengths of "dash" and "space-between-dash". E.g. dash = c(6, 4, 2, 4) produces a dotted-dashed line
...	other line creation options. See https://www.tcl.tk/man/tcl8.6/TkCmd/canvas.html#M26

See Also

Other canvas: canvas_arc(), canvas_clear(), canvas_image(), canvas_line(), canvas_oval(), canvas_plot(), canvas_rect(), canvas_save(), canvas_text(), tic_canvas()

---

canvas_rect *Draw a polygon on a canvas*

Description

Draw a polygon on a canvas

Usage

```
canvas_rect(canvas, x1, y1, x2, y2, fill, outline, width, dash, ...)
```

Arguments

canvas	a tic_ui 'canvas' element.
x1, y1, x2, y2	corners of rectangle
fill	line colour
outline	outline colour
width	line width e.g. width = 2
dash	Specifies the line's dash pattern. This should be a numeric vector with alternating lengths of "dash" and "space-between-dash". E.g. dash = c(6, 4, 2, 4) produces a dotted-dashed line
...	other line creation options. See https://www.tcl.tk/man/tcl8.6/TkCmd/canvas.html#M26

See Also

Other canvas: canvas_arc(), canvas_clear(), canvas_image(), canvas_line(), canvas_oval(), canvas_plot(), canvas_polygon(), canvas_save(), canvas_text(), tic_canvas()

---

canvas_save *Save the contents of the canvas to file*

Description

tcl/tk only exports the canvas as a PS (Postscript) file. In this function, the magick package is used to read the postscript file and render it to an image file based upon the suffix supplied in the filename argument.

Usage

```
canvas_save(canvas, filename, ps_density = NULL, ...)
```

Arguments

canvas	a tic_ui 'canvas' element.
filename	where the image should be saved. The image suffix will be used as the format argument to the call to magick::image_write()
ps_density	DPI. Resolution to postscript document. This corresponds to the density argument in magick::image_read(). Default: NULL means to use whatever magick package defaults to.
...	all other arguments passed to magick::image_write()

See Also

Other canvas: canvas_arc(), canvas_clear(), canvas_image(), canvas_line(), canvas_oval(), canvas_plot(), canvas_polygon(), canvas_rect(), canvas_text(), tic_canvas()

---

canvas_text *Draw text on a canvas*

Description

Draw text on a canvas

Usage

```
canvas_text(canvas, x, y, text, fill, justify, angle, ...)
```

Arguments

canvas	a tic_ui 'canvas' element.
x, y	coorindates for text
text	string to display
fill	text colour
justify	one of 'left', 'right', 'center'. Default: left
angle	default: 0
...	other line creation options. See https://www.tcl.tk/man/tcl8.6/TkCmd/canvas.html#M26

See Also

Other canvas: canvas_arc(), canvas_clear(), canvas_image(), canvas_line(), canvas_oval(), canvas_plot(), canvas_polygon(), canvas_rect(), canvas_save(), tic_canvas()

---

extract_node_by_id *Extract a nide from a* tic_ui *tree*

Description

Extract a nide from a tic_ui tree

Usage

```
extract_node_by_id(ui, id)
```

Arguments

ui	a tic_ui object representing a UI
id	the id of the path. An ID string of the tkwin object. e.g. ".1.2.2.3.1". See the output of printing a tic_ui object to see the list of all ID in this UI.

format.reactive_value *Format/print a reactive value*

Description

Format/print a reactive value

Usage

```
## S3 method for class 'reactive_value'
format(x, ...)

## S3 method for class 'reactive_textbox'
format(x, ...)

## S3 method for class 'reactive_value'
print(x, ...)
```

Arguments

x	object
...	ignored

handle_images *Sanitize an argument list to replace any image paths with a tkimage object*

Description

Sanitize an argument list to replace any image paths with a tkimage object

Usage

```
handle_images(args)
```

Arguments

args	list of args as part of a tic_spec

Value

list of args with any image arguments replaced by a tkimage object

launch_idle_func *Setup an idle callback to a user function at the given FPS while the window is alive*

Description

To stop the function:

- Close the window it is attached to
- Use a global logical value that is consulted inside the user func to determine if any action should be taken

Usage

```
launch_idle_func(win, user_func, fps = 30, initial_delay = 100)
```

Arguments

win	top level window object
user_func	users R function. This function will be called within the idle loop within any argumnets
fps	desired frame rate. default: 30
initial_delay	initial delay before running function for the first time. In milliseconds. Default: 1000

list_font_families *List all the font families present*

Description

List all the font families present

Usage

```
list_font_families()
```

Value

character vector of names

list_themes *Get a list of active themes*

Description

Get a list of active themes

Usage

```
list_themes()
```

pack_opts *Create a list of pack options used during widget creation*

Description

These pack options specify how a widget is packed into its parent element.

Usage

```
pack_opts(anchor, expand, fill, ipadx, ipady, padx, pady, side, ...)
```

Arguments

anchor	Specify where to position content within its parent. Defaults to 'center'. The alternative is to specify a string consisting only of the letters 'n', 's', 'e', 'w', to indicate the compass direction to anchor to. e.g. 'w', or 'sw' for 'west' (left) or 'southwest' respectively.
expand	Should the content should be expanded to consume extra space in its parent container? Default: TRUE
fill	If parent size is larger than its requested dimensions, this option may be used to stretch the content. none No stretching of widget x Stretch the content horizontally to fill the space y Stretch the content vertically to fill the space both Stretch the content horizontally and vertically to fill the space
ipadx, ipady	How much horizontal/vertical *internal* padding to leave on the side of each element. If you are familiar with HTML/CSSThis is analogous to CSS margin.
padx, pady	How much horizontal/vertical *external* padding to leave on the side of each element. This may be two values in order to specify different padding for left vs right, or top vs bottom. If you are familiar with HTML/CSSThis is analogous to CSS margin.
side	Which side of the container the content will be packed against? Possible values: left, right, top, bottom
...	extra named args used by the packing spec.

tcl/tk

See tcl/tk documentation for more information on the packing specification https://www.tcl.tk/man/tcl8.6/TkCmd/pack.html

---

popup_color_picker *Create a popup window for choosing a colour*

Description

Create a popup window for choosing a colour

Usage

```
popup_color_picker(title = "Choose colourr", ...)
```

Arguments

title	Title to display on popup window
...	other named arguments used to initialise this widget

tcl/tk

See tcl/tk documentation for more information on this element https://www.tcl.tk/man/tcl8.6/TkCmd/chooseColor.html

Examples

```
## Not run:
popup_color_picker(message = 'hello')

## End(Not run)
```

---

popup_messagebox *Display a popup message box*

Description

Display a popup message box

Usage

```
popup_messagebox(message, type, title, ...)
```

Arguments

message	message to display. Required.
type	one of 'abortretryignore', 'ok', 'okcancel', 'retrycancel', 'yesno', 'yesnocancel'
title	window title
...	other options (experts)

tcl/tk

See tcl/tk documentation for more information on this element https://www.tcl.tk/man/tcl8.6/TkCmd/messageBox.html

Examples

```
## Not run:
popup_messagebox(message = 'hello')

## End(Not run)
```

---

popup_open_file *Dialogs for choosing a file to open and save*

Description

Dialogs for choosing a file to open and save

Usage

```
popup_open_file(multiple, ...)

popup_save_file(confirmoverwrite, ...)
```

Arguments

multiple	allow the user to choose multiple files? default: FALSE
...	other arguments. See tcltk documentation for all possible options
confirmoverwrite	If user selects a filename which already exists, then show another popup to confirm overwriting this file. Default: TRUE.

tcl/tk

See tcl/tk documentation for more information on this element https://www.tcl.tk/man/tcl8.6/TkCmd/getOpenFile.html

print.tic_spec *print a tic_spec*

Description

print a tic_spec

Usage

```
## S3 method for class 'tic_spec'
print(x, ...)
```

Arguments

x	tic_spec object
...	ignored

print.tic_ui *Print a* tic_ui *object*

Description

Print a tic_ui object

Usage

```
## S3 method for class 'tic_ui'
print(x, ...)
```

Arguments

x	tic_ui object
...	ignored

reactive_lgl *Create a reactive variable*

Description

Create a reactive variable

Usage

```
reactive_lgl(value = FALSE)

reactive_int(value = 0L)

reactive_chr(value = "")

reactive_dbl(value = 0)

reactive_textbox(value)
```

Arguments

value	initial value.

render_ui *Render a complete UI given a spec for a window with child elements*

Description

Render a complete UI given a spec for a window with child elements

Usage

```
render_ui(spec, parent = NULL)
```

Arguments

spec	UI spec as created using `tic_window()`, `tic_button()` etc
parent	parent object. The default 'NULL' value is only allowed for `tic_window()` objects. Otherwise this should be a `tclObj` holding a reference to a window. TODO: parent could also be something else done by a render_ui call?

Value

a nested named list of tcl objects making up the UI. For advanced users that want to do low-level manipulation of the window, these `tclObj` objects are a key argument to functions in the `tcltk` package.

The key side effect of this function is that a tcl/tk window will be opened and rendered to this specficiation.

---

set_font_family *Set the font family*

Description

Set the font family

Usage

```
set_font_family(body = NULL, headings = body)
```

Arguments

`body`	name of font family for body text, buttons, menus etc
`headings`	name of the font family for the h1-h5 headings for labels. by default this will be set to the same as the body font.

---

set_font_scale *Globally scale the fonts in the rendered UI*

Description

Globally scale the fonts in the rendered UI

Usage

```
set_font_scale(scale)
```

Arguments

`scale`	numeric scale factor. default 1.0

Value

none.

set_theme *Activate a known theme by name*

Description

For advanced users, you can load a theme from a file using `tcltk::tcl('source', theme_file)`, before calling `set_theme()`

Usage

```
set_theme(theme_name = "r-sun-valley-light")
```

Arguments

theme_name	name of theme. By default this will load the built-in theme developed for R. Use `list_themes()` to see what themes are available.

Value

none

Examples

```
{
## Not run:
list_themes()
set_theme('default')

## End(Not run)
}
```

tic_button *Button that runs a command when pressed*

Description

A button widget displays a textual label and/or image, and evaluates a command when pressed.

Usage

```
tic_button(
  text,
  command,
  textvariable,
  image,
  compound,
  width,
  bind = NULL,
  pack = NULL,
  ...
)
```

Arguments

`text`, `textvariable`
: Specifies a text string to be displayed inside the widget.
`text` Simple character string containing the name
`textvariable` Reactive variable containing a string

`command`
: Function to evaluate when the widget is invoked.

`image`
: pathname of image to display

`compound`
: Specifies how to display the image relative to the text, in the case both `text` and `image` are present. Possible values:
`text` Display text only
`image` Display image only
`top,bottom,left,right,center` Display image with this position relative to the text
`none` Display the image if present, otherwise the text

`width`
: If greater than zero, specifies how much space, in character widths, to allocate for the text label. If less than zero, specifies a minimum width. If zero or unspecified, the natural width of the text label is used.

`bind`
: bind commands to particular events on this element. This may be a single result of `bind_opts()` or a list of them for multiple events.

`pack`
: a named list of pack options for how to incorporate this element into its parent container. Default: NULL means to use the standard packing. See `pack_opts()` as a way of creating a valid list of pack options.

`...`
: Other arguments are parsed as follows:
`named arguments` Further options to be used during the creation of this widget. See the tcl/tk documentation for all arguments possible for this widget.
`unnamed arguments` Container widgets (e.g. `tic_frame()`) treat any unnamed arguments as child objects. Non-container widgets (e.g `tic_button()`) will raise an error if there are any unnamed widgets.

Value

A `tic_spec` object containing the widget specification.

tcl/tk

See tcl/tk documentation for more information on this element https://www.tcl.tk/man/tcl8.6/TkCmd/ttk_button.htm

See Also

Other widgets: tic_canvas(), tic_checkbutton(), tic_combobox(), tic_label(), tic_menubutton(), tic_menuitem(), tic_menu(), tic_progressbar(), tic_radiobutton(), tic_separator(), tic_sizegrip(), tic_slider(), tic_spinbox(), tic_submenu(), tic_textbox(), tic_textentry()

Examples

```
## Not run:
  tic_window(
    title = "Hello",
    tic_button(text = "Button", command = function() { message("Button clicked") })
  )

## End(Not run)
```

---

tic_canvas *Create drawing surface widget*

Description

This command will create a canvas widget as part of UI creation, but all the interesting things the user would want to do with a canvas are interactive things *after* the UI is available to the user.

Usage

```
tic_canvas(
  background = "#fafafa",
  scrollbars = FALSE,
  bind = NULL,
  pack = NULL,
  ...
)
```

Arguments

background	Background colour. Default: '#fafafa'
scrollbars	include scrollbars on the canvas? Default: FALSE. This option has not really been tested. Scrollbars still look funky.
bind	bind commands to particular events on this element. This may be a single result of bind_opts() or a list of them for multiple events.

pack — a named list of pack options for how to incorporate this element into its parent container. Default: NULL means to use the standard packing. See pack_opts() as a way of creating a valid list of pack options.

... — Other arguments are parsed as follows:

named arguments Further options to be used during the creation of this widget. See the tcl/tk documentation for all arguments possible for this widget.

unnamed arguments Container widgets (e.g. tic_frame()) treat any unnamed arguments as child objects. Non-container widgets (e.g tic_button()) will raise an error if there are any unnamed widgets.

Details

So to make actual use of canvas at the moment, you'll have to be prepared to write some code with the 'tcltk' package and read lots of documentation!

Value

handle on the tcl/tk object. TODO: better language needed here.

tcl/tk

See tcl/tk documentation for more information on this element https://www.tcl.tk/man/tcl8.6/TkCmd/canvas.html

See Also

Other widgets: tic_button(), tic_checkbutton(), tic_combobox(), tic_label(), tic_menubutton(), tic_menuitem(), tic_menu(), tic_progressbar(), tic_radiobutton(), tic_separator(), tic_sizegrip(), tic_slider(), tic_spinbox(), tic_submenu(), tic_textbox(), tic_textentry()

Other canvas: canvas_arc(), canvas_clear(), canvas_image(), canvas_line(), canvas_oval(), canvas_plot(), canvas_polygon(), canvas_rect(), canvas_save(), canvas_text()

---

tic_checkbutton *A button which toggles between two states*

Description

A checkbutton is used to show or change a setting. It has two states, selected and deselected.

Usage

```
tic_checkbutton(
  text,
  variable,
  command,
  textvariable,
  width,
```

```
  style,
  bind = NULL,
  pack = NULL,
  ...
)
```

Arguments

text	Specifies a text string to be displayed inside the widget. text Simple character string containing the name textvariable Reactive variable containing a string
variable	The variable containing the state of the button. Create this variable with reactive_lgl.
command	Function to evaluate when the widget is invoked.
textvariable	Specifies a text string to be displayed inside the widget. text Simple character string containing the name textvariable Reactive variable containing a string
width	If greater than zero, specifies how much space, in character widths, to allocate for the text label. If less than zero, specifies a minimum width. If zero or unspecified, the natural width of the text label is used.
style	Default behaviour is to use a checkbox stye button. With the build in theme in this package, two other styles are possible: "toggle The button will look like a regular button but will alternate between 'on' and 'off' states with each press 'switch' The button will look like a sliding switch which slides from one side to the other with each press
bind	bind commands to particular events on this element. This may be a single result of bind_opts() or a list of them for multiple events.
pack	a named list of pack options for how to incorporate this element into its parent container. Default: NULL means to use the standard packing. See pack_opts() as a way of creating a valid list of pack options.
...	Other arguments are parsed as follows: named arguments Further options to be used during the creation of this widget. See the tcl/tk documentation for all arguments possible for this widget. unnamed arguments Container widgets (e.g. tic_frame()) treat any unnamed arguments as child objects. Non-container widgets (e.g tic_button()) will raise an error if there are any unnamed widgets.

Details

The state of the checkbutton should be linked to a reactive_lgl logical variable using variable argument.

Value

handle on the tcl/tk object

tcl/tk

See tcl/tk documentation for more information on this element https://www.tcl.tk/man/tcl8.6/TkCmd/ttk_checkbutton.htm

See Also

Other widgets: tic_button(), tic_canvas(), tic_combobox(), tic_label(), tic_menubutton(), tic_menuitem(), tic_menu(), tic_progressbar(), tic_radiobutton(), tic_separator(), tic_sizegrip(), tic_slider(), tic_spinbox(), tic_submenu(), tic_textbox(), tic_textentry()

Examples

```
alert = reactive_lgl(FALSE)
tic_window(
  title = "demo",
  tic_checkbutton(text = "Click for Alert", variable = alert)
)
```

---

tic_combobox *Combobox: text field with popdown selection list*

Description

A combobox combines a text field with a pop-down list of values; the user may select the value of the text field from among the values in the list.

Usage

```
tic_combobox(
  values,
  textvariable,
  justify,
  state,
  width,
  bind = NULL,
  pack = NULL,
  ...
)
```

Arguments

values	Specifies the list of values to display in the drop-down listbox.
textvariable	Reactive value for the current selected value in this widget
justify	one of left, center, right
state	one of normal, readonly, disabled. Default: normal

`width` If greater than zero, specifies how much space, in character widths, to allocate for the text label. If less than zero, specifies a minimum width. If zero or unspecified, the natural width of the text label is used.

`bind` bind commands to particular events on this element. This may be a single result of `bind_opts()` or a list of them for multiple events.

`pack` a named list of pack options for how to incorporate this element into its parent container. Default: NULL means to use the standard packing. See `pack_opts()` as a way of creating a valid list of pack options.

`...` Other arguments are parsed as follows:

named `arguments` Further options to be used during the creation of this widget. See the tcl/tk documentation for all arguments possible for this widget.

unnamed `arguments` Container widgets (e.g. `tic_frame()`) treat any unnamed arguments as child objects. Non-container widgets (e.g `tic_button()`) will raise an error if there are any unnamed widgets.

Value

handle on the tcl/tk object

tcl/tk

See tcl/tk documentation for more information on this element https://www.tcl.tk/man/tcl8.6/TkCmd/ttk_combobox.htm

See Also

Other widgets: `tic_button()`, `tic_canvas()`, `tic_checkbutton()`, `tic_label()`, `tic_menubutton()`, `tic_menuitem()`, `tic_menu()`, `tic_progressbar()`, `tic_radiobutton()`, `tic_separator()`, `tic_sizegrip()`, `tic_slider()`, `tic_spinbox()`, `tic_submenu()`, `tic_textbox()`, `tic_textentry()`

Examples

```
opts <- c('alpha', 'bravo', 'charlie')
selected <- reactive_chr(opts[1])
tic_window(
  title = "Demo",
  tic_combobox(values = opts, textvariable = selected)
)
```

`tic_label` *Create a text label*

Description

Create a text label

Usage

```
tic_label(text, textvariable, width, style, bind = NULL, pack = NULL, ...)
```

Arguments

`text`	Specifies a text string to be displayed inside the widget. `text` Simple character string containing the name `textvariable` Reactive variable containing a string
`textvariable`	Specifies a text string to be displayed inside the widget. `text` Simple character string containing the name `textvariable` Reactive variable containing a string
`width`	If greater than zero, specifies how much space, in character widths, to allocate for the text label. If less than zero, specifies a minimum width. If zero or unspecified, the natural width of the text label is used.
`style`	specify the style for this label. Possible values are "h1" through "h5" which are analogues to the h1 to h5 headings in HTML, with "h1" being the largest heading size, down to "h5" being the smallest.
`bind`	bind commands to particular events on this element. This may be a single result of `bind_opts()` or a list of them for multiple events.
`pack`	a named list of pack options for how to incorporate this element into its parent container. Default: NULL means to use the standard packing. See `pack_opts()` as a way of creating a valid list of pack options.
`...`	Other arguments are parsed as follows: `named arguments` Further options to be used during the creation of this widget. See the tcl/tk documentation for all arguments possible for this widget. `unnamed arguments` Container widgets (e.g. `tic_frame()`) treat any unnamed arguments as child objects. Non-container widgets (e.g `tic_button()`) will raise an error if there are any unnamed widgets.

Value

handle on the tcl/tk object

tcl/tk

See tcl/tk documentation for more information on this element https://www.tcl.tk/man/tcl8.6/TkCmd/ttk_label.htm

See Also

Other widgets: tic_button(), tic_canvas(), tic_checkbutton(), tic_combobox(), tic_menubutton(), tic_menuitem(), tic_menu(), tic_progressbar(), tic_radiobutton(), tic_separator(), tic_sizegrip(), tic_slider(), tic_spinbox(), tic_submenu(), tic_textbox(), tic_textentry()

`tic_labelframe` *Container widgets with/without label label*

Description

A labelframe widget is a container used to group other widgets together.

Usage

```
tic_labelframe(
  ...,
  text,
  relief,
  borderwidth,
  bind = NULL,
  pack = NULL,
  pack_def = NULL
)

tic_frame(..., relief, borderwidth, bind = NULL, pack = NULL)
```

Arguments

`...`	Other arguments are parsed as follows: named `arguments` Further options to be used during the creation of this widget. See the tcl/tk documentation for all arguments possible for this widget. unnamed `arguments` Container widgets (e.g. `tic_frame()`) treat any unnamed arguments as child objects. Non-container widgets (e.g `tic_button()`) will raise an error if there are any unnamed widgets.
`text`	Label to display for this frame. Character string.
`relief`	border style. One of: flat, groove, raised, ridge, solid sunken. Defaults to: 'flat'
`borderwidth`	desired width of widget border
`bind`	bind commands to particular events on this element. This may be a single result of `bind_opts()` or a list of them for multiple events.
`pack`	a named list of pack options for how to incorporate this element into its parent container. Default: NULL means to use the standard packing. See `pack_opts()` as a way of creating a valid list of pack options.
`pack_def`	Default packing options for children of this object. This can be overriden by setting `pack` explicitly on child elemnets you want to control packing for.

Details

It has a label.

Value

handle on the tcl/tk object

tcl/tk

See tcl/tk documentation for more information on this element https://www.tcl.tk/man/tcl8.6/TkCmd/ttk_labelframe.htm

See Also

Other widgets containers: tic_window()

Other widgets containers: tic_window()

---

tic_menu *Create a toplevel menu bar.*

Description

This element must an immediate child of the main tic_window()

Usage

```
tic_menu(..., text, tearoff = FALSE, bind = NULL, pack = NULL)
```

Arguments

...	Other arguments are parsed as follows: named arguments Further options to be used during the creation of this widget. See the tcl/tk documentation for all arguments possible for this widget. unnamed arguments Container widgets (e.g. tic_frame()) treat any unnamed arguments as child objects. Non-container widgets (e.g tic_button()) will raise an error if there are any unnamed widgets.
text	Specifies a text string to be displayed inside the widget. text Simple character string containing the name textvariable Reactive variable containing a string
tearoff	Can the menu be torn off? Default: FALSE
bind	bind commands to particular events on this element. This may be a single result of bind_opts() or a list of them for multiple events.
pack	a named list of pack options for how to incorporate this element into its parent container. Default: NULL means to use the standard packing. See pack_opts() as a way of creating a valid list of pack options.

Value

handle on the tcl/tk object

tcl/tk

See tcl/tk documentation for more information on this element https://www.tcl.tk/man/tcl8.6/TkCmd/menu.htm

See Also

Other widgets: tic_button(), tic_canvas(), tic_checkbutton(), tic_combobox(), tic_label(), tic_menubutton(), tic_menuitem(), tic_progressbar(), tic_radiobutton(), tic_separator(), tic_sizegrip(), tic_slider(), tic_spinbox(), tic_submenu(), tic_textbox(), tic_textentry()

Examples

```
## Not run:
tic_window(
  tic_menu(
    text = "Press for menu",
    tic_menuitem("Run a command", menutype = "command", command = function() {
      message("This is where a command is run")
    }),
    tic_submenu(
      label = "Sub menu here",
      tic_menuitem("Run this", "command", command = function() { message("Hello")})
    )
  )
)

## End(Not run)
```

---

tic_menubutton *Button that pops down a menu when pressed*

Description

A menubutton widget displays a textual label and/or image, and displays a menu when pressed.

Usage

```
tic_menubutton(..., text, bind = NULL, pack = NULL)
```

Arguments

...	Other arguments are parsed as follows: named arguments Further options to be used during the creation of this widget. See the tcl/tk documentation for all arguments possible for this widget. unnamed arguments Container widgets (e.g. tic_frame()) treat any unnamed arguments as child objects. Non-container widgets (e.g tic_button()) will raise an error if there are any unnamed widgets.
text	Specifies a text string to be displayed inside the widget. text Simple character string containing the name textvariable Reactive variable containing a string
bind	bind commands to particular events on this element. This may be a single result of bind_opts() or a list of them for multiple events.
pack	how this object should be packed into its parent. pack has no effect for tic_menu() or tic_menu_item().

Details

Include items in this menu using tic_menuitem() and tic_submenu()

Value

handle on the tcl/tk object

tcl/tk

See tcl/tk documentation for more information on this element https://www.tcl.tk/man/tcl8.6/TkCmd/ttk_menubutton.htm

See Also

Other widgets: tic_button(), tic_canvas(), tic_checkbutton(), tic_combobox(), tic_label(), tic_menuitem(), tic_menu(), tic_progressbar(), tic_radiobutton(), tic_separator(), tic_sizegrip(), tic_slider(), tic_spinbox(), tic_submenu(), tic_textbox(), tic_textentry()

Examples

```
tic_menubutton(
  text = "Press for menu",
  tic_menuitem("Run a command", menutype = "command", command = function() {
    message("This is where a command is run")
  }),
  tic_submenu(
    label = "Sub menu here",
    tic_menuitem("Run this", "command", command = function() { message("Hello")})
  )
)
```

tic_menuitem *Create a menu item*

Description

This element must an immediate child of a tic_menubuton, tic_menu or tic_submenu

Usage

```
tic_menuitem(label, menutype, command, image, bind = NULL, pack = NULL, ...)
```

Arguments

label	text to display
menutype	one of the following:
	command When menu item is selected, run the function given by the command argument
	cascade Do not use. Use tic_submenu()to create a submenu.
	separator Horizontal separato between menu items
	checkbutton A menuitem which functions as a checkbutton. Use this in conjunction with variable to specify the reactive_lgl variable to store the state of the button (i.e. TRUE/FALSE). Use in conjunctionwith command argument to run a function each time the button is selected.
	radiobutton Same as the checkbutton menu item, but fo defining a mutually exclusive set of selection options. See tic_radiobutton() for more information and examples
command	Function to evaluate when the widget is invoked.
image	pathname of image to display
bind	bind commands to particular events on this element. This may be a single result of bind_opts() or a list of them for multiple events.
pack	a named list of pack options for how to incorporate this element into its parent container. Default: NULL means to use the standard packing. See pack_opts() as a way of creating a valid list of pack options.
...	Other arguments are parsed as follows:
	named arguments Further options to be used during the creation of this widget. See the tcl/tk documentation for all arguments possible for this widget.
	unnamed arguments Container widgets (e.g. tic_frame()) treat any unnamed arguments as child objects. Non-container widgets (e.g tic_button()) will raise an error if there are any unnamed widgets.

Value

handle on the tcl/tk object

tcl/tk

See tcl/tk documentation for more information on this element https://www.tcl.tk/man/tcl8.6/TkCmd/menu.htm

See Also

Other widgets: tic_button(), tic_canvas(), tic_checkbutton(), tic_combobox(), tic_label(), tic_menubutton(), tic_menu(), tic_progressbar(), tic_radiobutton(), tic_separator(), tic_sizegrip(), tic_slider(), tic_spinbox(), tic_submenu(), tic_textbox(), tic_textentry()

---

tic_notebook *Notebook: Multi-paned container window i.e. a window with multiple tabs.*

Description

A notebook widget manages a collection of windows and displays a single one at a time.

Usage

```
tic_notebook(..., labels, bind = NULL, pack = NULL)
```

Arguments

...	Other arguments are parsed as follows: named arguments Further options to be used during the creation of this widget. See the tcl/tk documentation for all arguments possible for this widget. unnamed arguments Container widgets (e.g. tic_frame()) treat any unnamed arguments as child objects. Non-container widgets (e.g tic_button()) will raise an error if there are any unnamed widgets.
labels	Labels to display at the top of each tab. Required.
bind	bind commands to particular events on this element. This may be a single result of bind_opts() or a list of them for multiple events.
pack	a named list of pack options for how to incorporate this element into its parent container. Default: NULL means to use the standard packing. See pack_opts() as a way of creating a valid list of pack options.

Details

Each content window is associated with a tab, which the user may select to change the currently-displayed window.

Value

handle on the tcl/tk object

tcl/tk

See tcl/tk documentation for more information on this element https://www.tcl.tk/man/tcl8.6/TkCmd/ttk_notebook.htm

---

`tic_panedwindow` *Panedwindow: Multi-pane container window*

Description

A ttk::panedwindow widget displays a number of subwindows, stacked either vertically or horizontally. The user may adjust the relative sizes of the subwindows by dragging the sash between panes.

Usage

```
tic_panedwindow(..., sizes = NULL, orient, bind = NULL, pack = NULL)
```

Arguments

`...`	Other arguments are parsed as follows: `named arguments` Further options to be used during the creation of this widget. See the tcl/tk documentation for all arguments possible for this widget. `unnamed arguments` Container widgets (e.g. `tic_frame()`) treat any unnamed arguments as child objects. Non-container widgets (e.g `tic_button()`) will raise an error if there are any unnamed widgets.
`sizes`	The relative sizes of the elements in this row or column
`orient`	orientataion. 'horizontal' or 'vertical' (default)
`bind`	bind commands to particular events on this element. This may be a single result of `bind_opts()` or a list of them for multiple events.
`pack`	a named list of pack options for how to incorporate this element into its parent container. Default: NULL means to use the standard packing. See `pack_opts()` as a way of creating a valid list of pack options.

Value

handle on the tcl/tk object

tcl/tk

See tcl/tk documentation for more information on this element https://www.tcl.tk/man/tcl8.6/TkCmd/ttk_panedwindow.htm

`tic_progressbar` *Progressbar: Provide progress feedback*

Description

A progressbar widget shows the status of a long-running operation.

Usage

```
tic_progressbar(variable, mode, maximum, bind = NULL, pack = NULL, ...)
```

Arguments

`variable`	reactive variable holding the progress value
`mode`	'determinate' or 'indeterminate'.
`maximum`	maximum value when in 'determinate' mode
`bind`	bind commands to particular events on this element. This may be a single result of `bind_opts()` or a list of them for multiple events.
`pack`	a named list of pack options for how to incorporate this element into its parent container. Default: NULL means to use the standard packing. See `pack_opts()` as a way of creating a valid list of pack options.
`...`	Other arguments are parsed as follows: `named arguments` Further options to be used during the creation of this widget. See the tcl/tk documentation for all arguments possible for this widget. `unnamed arguments` Container widgets (e.g. `tic_frame()`) treat any unnamed arguments as child objects. Non-container widgets (e.g `tic_button()`) will raise an error if there are any unnamed widgets.

Details

They can operate in two modes: determinate mode shows the amount completed relative to the total amount of work to be done, and indeterminate mode provides an animated display to let the user know that something is happening.

Value

handle on the tcl/tk object

tcl/tk

See tcl/tk documentation for more information on this element https://www.tcl.tk/man/tcl8.6/TkCmd/ttk_progressbar.htm

See Also

Other widgets: tic_button(), tic_canvas(), tic_checkbutton(), tic_combobox(), tic_label(), tic_menubutton(), tic_menuitem(), tic_menu(), tic_radiobutton(), tic_separator(), tic_sizegrip(), tic_slider(), tic_spinbox(), tic_submenu(), tic_textbox(), tic_textentry()

`tic_radiobutton` *Radiobutton: Mutually exclusive option widget*

Description

Radiobutton widgets are used in groups to show or change a set of mutually-exclusive options.

Usage

```
tic_radiobutton(
  text,
  value,
  variable,
  command,
  textvariable,
  width,
  bind = NULL,
  pack = NULL,
  ...
)
```

Arguments

`text`	Specifies a text string to be displayed inside the widget. `text` Simple character string containing the name `textvariable` Reactive variable containing a string
`value`	The value to store in the associated `variable` when the widget is selected.
`variable`	The reactive variable holding the state for a set of radiobuttons.
`command`	Function to evaluate when the widget is invoked.
`textvariable`	Specifies a text string to be displayed inside the widget. `text` Simple character string containing the name `textvariable` Reactive variable containing a string
`width`	If greater than zero, specifies how much space, in character widths, to allocate for the text label. If less than zero, specifies a minimum width. If zero or unspecified, the natural width of the text label is used.
`bind`	bind commands to particular events on this element. This may be a single result of `bind_opts()` or a list of them for multiple events.

pack — a named list of pack options for how to incorporate this element into its parent container. Default: NULL means to use the standard packing. See `pack_opts()` as a way of creating a valid list of pack options.

... — Other arguments are parsed as follows:

named `arguments` Further options to be used during the creation of this widget. See the tcl/tk documentation for all arguments possible for this widget.

unnamed `arguments` Container widgets (e.g. `tic_frame()`) treat any unnamed arguments as child objects. Non-container widgets (e.g `tic_button()`) will raise an error if there are any unnamed widgets.

Details

Radiobuttons are linked to a reactive variable, and have an associated value; when a radiobutton is clicked, it sets the variable to its associated value.

Value

handle on the tcl/tk object

tcl/tk

See tcl/tk documentation for more information on this element https://www.tcl.tk/man/tcl8.6/TkCmd/ttk_radiobutton.htm

See Also

Other widgets: `tic_button()`, `tic_canvas()`, `tic_checkbutton()`, `tic_combobox()`, `tic_label()`, `tic_menubutton()`, `tic_menuitem()`, `tic_menu()`, `tic_progressbar()`, `tic_separator()`, `tic_sizegrip()`, `tic_slider()`, `tic_spinbox()`, `tic_submenu()`, `tic_textbox()`, `tic_textentry()`

Examples

```
choices <- c('alpha', 'bravo', 'charlie')
state <- reactive_chr(choices[1])
tic_window(
  title = "demo",
  tic_radiobutton(text = choices[1], value = choices[1], variable = state),
  tic_radiobutton(text = choices[2], value = choices[2], variable = state),
  tic_radiobutton(text = choices[3], value = choices[3], variable = state)
)
```

tic_row *Create a container element oriented as a row or column of elements*

Description

This is not a direct analogy for a single tk element

Usage

```
tic_row(..., bind = NULL, pack = NULL, pack_def = NULL)

tic_col(..., bind = NULL, pack = NULL, pack_def = NULL)
```

Arguments

...	Other arguments are parsed as follows: named arguments Further options to be used during the creation of this widget. See the tcl/tk documentation for all arguments possible for this widget. unnamed arguments Container widgets (e.g. tic_frame()) treat any unnamed arguments as child objects. Non-container widgets (e.g tic_button()) will raise an error if there are any unnamed widgets.
bind	bind commands to particular events on this element. This may be a single result of bind_opts() or a list of them for multiple events.
pack	a named list of pack options for how to incorporate this element into its parent container. Default: NULL means to use the standard packing. See pack_opts() as a way of creating a valid list of pack options.
pack_def	Default packing options for children of this object. This can be overriden by setting pack explicitly on child elemnets you want to control packing for.

Value

handle on the tcl/tk object

tcl/tk

This is not a direct implementation for an existing tk element. It is implemented as a ttk::frame element with horizontal or vertical packing by default.

`tic_separator` *Separator: Separator bar*

Description

A separator widget displays a horizontal or vertical separator bar.

Usage

```
tic_separator(orient, bind = NULL, pack = list(fill = "x"), ...)
```

Arguments

`orient`	'horizontal' or 'vertical'.
`bind`	bind commands to particular events on this element. This may be a single result of `bind_opts()` or a list of them for multiple events.
`pack`	a named list of pack options for how to incorporate this element into its parent container. Default: NULL means to use the standard packing. See `pack_opts()` as a way of creating a valid list of pack options.
`...`	Other arguments are parsed as follows: `named arguments` Further options to be used during the creation of this widget. See the tcl/tk documentation for all arguments possible for this widget. `unnamed arguments` Container widgets (e.g. `tic_frame()`) treat any unnamed arguments as child objects. Non-container widgets (e.g `tic_button()`) will raise an error if there are any unnamed widgets.

Value

handle on the tcl/tk object. TODO: better language needed here.

tcl/tk

See tcl/tk documentation for more information on this element https://www.tcl.tk/man/tcl8.6/TkCmd/ttk_separator.htm

See Also

Other widgets: `tic_button()`, `tic_canvas()`, `tic_checkbutton()`, `tic_combobox()`, `tic_label()`, `tic_menubutton()`, `tic_menuitem()`, `tic_menu()`, `tic_progressbar()`, `tic_radiobutton()`, `tic_sizegrip()`, `tic_slider()`, `tic_spinbox()`, `tic_submenu()`, `tic_textbox()`, `tic_textentry()`

`tic_sizegrip` *Sizegrip: Bottom-right corner resize widget*

Description

A sizegrip widget (also known as a grow box) allows the user to resize the containing toplevel window by pressing and dragging the grip.

Usage

```
tic_sizegrip(bind = NULL, ...)
```

Arguments

`bind`	bind commands to particular events on this element. This may be a single result of `bind_opts()` or a list of them for multiple events.
`...`	Other arguments are parsed as follows:
	`named arguments` Further options to be used during the creation of this widget. See the tcl/tk documentation for all arguments possible for this widget.
	`unnamed arguments` Container widgets (e.g. `tic_frame()`) treat any unnamed arguments as child objects. Non-container widgets (e.g `tic_button()`) will raise an error if there are any unnamed widgets.

Details

On Mac OSX, toplevel windows automatically include a built-in size grip by default. Adding a ttk::sizegrip there is harmless, since the built-in grip will just mask the widget.

Value

handle on the tcl/tk object. TODO: better language needed here.

tcl/tk

See tcl/tk documentation for more information on this element https://www.tcl.tk/man/tcl8.6/TkCmd/ttk_sizegrip.htm

See Also

Other widgets: `tic_button()`, `tic_canvas()`, `tic_checkbutton()`, `tic_combobox()`, `tic_label()`, `tic_menubutton()`, `tic_menuitem()`, `tic_menu()`, `tic_progressbar()`, `tic_radiobutton()`, `tic_separator()`, `tic_slider()`, `tic_spinbox()`, `tic_submenu()`, `tic_textbox()`, `tic_textentry()`

`tic_slider` *Slider: Create and manipulate a slider widget*

Description

A slider widget is used to control the numeric value of a reactive variable that varies uniformly over some range.

Usage

```
tic_slider(
  variable,
  command,
  from = 0,
  to = 100,
  bind = NULL,
  pack = list(fill = "x"),
  ...
)
```

Arguments

`variable`	reactive variable holding the slider value.
`command`	function to be invoked when slider value changes
`from, to`	numerical limits of the slider
`bind`	bind commands to particular events on this element. This may be a single result of `bind_opts()` or a list of them for multiple events.
`pack`	a named list of pack options for how to incorporate this element into its parent container. Default: NULL means to use the standard packing. See `pack_opts()` as a way of creating a valid list of pack options.
`...`	Other arguments are parsed as follows: `named arguments` Further options to be used during the creation of this widget. See the tcl/tk documentation for all arguments possible for this widget. `unnamed arguments` Container widgets (e.g. `tic_frame()`) treat any unnamed arguments as child objects. Non-container widgets (e.g `tic_button()`) will raise an error if there are any unnamed widgets.

Details

The widget displays a slider that can be moved along over a trough, with the relative position of the slider over the trough indicating the value of the variable.

Note: In tcl/tk this widget is known as a `ttk::scale` widget.

Value

handle on the tcl/tk object. TODO: better language needed here.

tcl/tk

See tcl/tk documentation for more information on this element https://www.tcl.tk/man/tcl8.6/TkCmd/ttk_scale.htm

See Also

Other widgets: tic_button(), tic_canvas(), tic_checkbutton(), tic_combobox(), tic_label(), tic_menubutton(), tic_menuitem(), tic_menu(), tic_progressbar(), tic_radiobutton(), tic_separator(), tic_sizegrip(), tic_spinbox(), tic_submenu(), tic_textbox(), tic_textentry()

---

tic_spinbox *Spinbox: Selecting text field widget*

Description

A spinbox widget is a text-entry widget with built-in up and down buttons that are used to either

- modify a numeric value by setting `from, to, increment` arguments
- select among a set of values by using the `values` argument

Usage

```
tic_spinbox(
  values,
  textvariable,
  command,
  from,
  to,
  increment,
  width,
  bind = NULL,
  pack = NULL,
  ...
)
```

Arguments

`values`	Specifies the list of values to display in the spinbox.
`textvariable`	Reactive value for the value displayed in this widget
`command`	Function to evaluate when the widget is invoked.
`from, to, increment`	numeric values for the low value, high value and change in value when the up and down buttons are pressed.
`width`	If greater than zero, specifies how much space, in character widths, to allocate for the text label. If less than zero, specifies a minimum width. If zero or unspecified, the natural width of the text label is used.

bind — bind commands to particular events on this element. This may be a single result of bind_opts() or a list of them for multiple events.

pack — a named list of pack options for how to incorporate this element into its parent container. Default: NULL means to use the standard packing. See pack_opts() as a way of creating a valid list of pack options.

... — Other arguments are parsed as follows:

named arguments Further options to be used during the creation of this widget. See the tcl/tk documentation for all arguments possible for this widget.

unnamed arguments Container widgets (e.g. tic_frame()) treat any unnamed arguments as child objects. Non-container widgets (e.g tic_button()) will raise an error if there are any unnamed widgets.

Details

The widget implements all the features of the tic_entry() widget.

Value

handle on the tcl/tk object

tcl/tk

See tcl/tk documentation for more information on this element https://www.tcl.tk/man/tcl8.6/TkCmd/ttk_spinbox.htm

See Also

Other widgets: tic_button(), tic_canvas(), tic_checkbutton(), tic_combobox(), tic_label(), tic_menubutton(), tic_menuitem(), tic_menu(), tic_progressbar(), tic_radiobutton(), tic_separator(), tic_sizegrip(), tic_slider(), tic_submenu(), tic_textbox(), tic_textentry()

Examples

```
opts <- c('alpha', 'bravo', 'charlie')
selected <- reactive_chr(opts[1])
tic_window(
  title = "Demo",
  tic_spinbox(values = opts, textvariable = selected)
)
```

`tic_submenu` *Create a submenu under a menubutton or menu*

Description

Create a submenu under a menubutton or menu

Usage

```
tic_submenu(..., label, tearoff = FALSE, bind = NULL, pack = NULL)
```

Arguments

`...`	Other arguments are parsed as follows: `named arguments` Further options to be used during the creation of this widget. See the tcl/tk documentation for all arguments possible for this widget. `unnamed arguments` Container widgets (e.g. `tic_frame()`) treat any unnamed arguments as child objects. Non-container widgets (e.g `tic_button()`) will raise an error if there are any unnamed widgets.
`label`	submenu label
`tearoff`	FALSE
`bind`	bind commands to particular events on this element. This may be a single result of `bind_opts()` or a list of them for multiple events.
`pack`	a named list of pack options for how to incorporate this element into its parent container. Default: NULL means to use the standard packing. See `pack_opts()` as a way of creating a valid list of pack options.

Value

handle on the tcl/tk object

tcl/tk

See tcl/tk documentation for more information on this element https://www.tcl.tk/man/tcl8.6/TkCmd/menu.htm

See Also

Other widgets: `tic_button()`, `tic_canvas()`, `tic_checkbutton()`, `tic_combobox()`, `tic_label()`, `tic_menubutton()`, `tic_menuitem()`, `tic_menu()`, `tic_progressbar()`, `tic_radiobutton()`, `tic_separator()`, `tic_sizegrip()`, `tic_slider()`, `tic_spinbox()`, `tic_textbox()`, `tic_textentry()`

tic_textbox *textbox: multi-line text input*

Description

Note: The interface to this widget differs from the core tcl/tk widget in that

Usage

```
tic_textbox(variable, bind = NULL, pack = NULL, ...)
```

Arguments

variable	reactive variable which reads the contents of the textbox. This must be a `reactive_textbox()` variable.
bind	bind commands to particular events on this element. This may be a single result of `bind_opts()` or a list of them for multiple events.
pack	a named list of pack options for how to incorporate this element into its parent container. Default: NULL means to use the standard packing. See `pack_opts()` as a way of creating a valid list of pack options.
...	Other arguments are parsed as follows:
	named `arguments` Further options to be used during the creation of this widget. See the tcl/tk documentation for all arguments possible for this widget.
	unnamed `arguments` Container widgets (e.g. `tic_frame()`) treat any unnamed arguments as child objects. Non-container widgets (e.g `tic_button()`) will raise an error if there are any unnamed widgets.

Value

handle on the tcl/tk object. TODO: better language needed here.

tcl/tk

See tcl/tk documentation for more information on this element https://www.tcl.tk/man/tcl8.6/TkCmd/text.htm

See Also

Other widgets: `tic_button()`, `tic_canvas()`, `tic_checkbutton()`, `tic_combobox()`, `tic_label()`, `tic_menubutton()`, `tic_menuitem()`, `tic_menu()`, `tic_progressbar()`, `tic_radiobutton()`, `tic_separator()`, `tic_sizegrip()`, `tic_slider()`, `tic_spinbox()`, `tic_submenu()`, `tic_textentry()`

`tic_textentry` *Textentry: Editable text field widget*

Description

A textentry widget displays a one-line text string and allows that string to be edited by the user.

Usage

```
tic_textentry(
  textvariable,
  validate,
  validatecommand,
  width,
  bind = NULL,
  pack = NULL,
  ...
)
```

Arguments

`textvariable`	Reactive value for the value displayed in this widget
`validate`	when to vdlidate the contents of this text box. Possible values: none, key, focus, focusin, focusout, all. If this is not set then the `validatecommand` function will never be run.
`validatecommand`	function to call when validation event occurs. This function must return a non-NA boolean value.
`width`	If greater than zero, specifies how much space, in character widths, to allocate for the text label. If less than zero, specifies a minimum width. If zero or unspecified, the natural width of the text label is used.
`bind`	bind commands to particular events on this element. This may be a single result of `bind_opts()` or a list of them for multiple events.
`pack`	a named list of pack options for how to incorporate this element into its parent container. Default: NULL means to use the standard packing. See `pack_opts()` as a way of creating a valid list of pack options.
`...`	Other arguments are parsed as follows: `named arguments` Further options to be used during the creation of this widget. See the tcl/tk documentation for all arguments possible for this widget. `unnamed arguments` Container widgets (e.g. `tic_frame()`) treat any unnamed arguments as child objects. Non-container widgets (e.g `tic_button()`) will raise an error if there are any unnamed widgets.

Details

The value of the string is linked to a reacive variaable with the `textvariable` argument.

Note: in tcl/tk this object is known as `entry` widget, rather than `textentry`.

Value

handle on the tcl/tk object

tcl/tk

See tcl/tk documentation for more information on this element https://www.tcl.tk/man/tcl8.6/TkCmd/ttk_entry.htm

See Also

Other widgets: `tic_button()`, `tic_canvas()`, `tic_checkbutton()`, `tic_combobox()`, `tic_label()`, `tic_menubutton()`, `tic_menuitem()`, `tic_menu()`, `tic_progressbar()`, `tic_radiobutton()`, `tic_separator()`, `tic_sizegrip()`, `tic_slider()`, `tic_spinbox()`, `tic_submenu()`, `tic_textbox()`

`tic_window` *Create the top level window*

Description

Create the top level window

Usage

```
tic_window(
  ...,
  title = "{tickle}",
  width = NULL,
  height = NULL,
  idle_func = NULL,
  idle_fps = 30,
  bind = NULL,
  pack = NULL,
  pack_def = NULL
)
```

Arguments

`...`	Other arguments are parsed as follows:
	`named arguments` Further options to be used during the creation of this widget. See the tcl/tk documentation for all arguments possible for this widget.

unnamed arguments	Container widgets (e.g. tic_frame()) treat any unnamed arguments as child objects. Non-container widgets (e.g tic_button()) will raise an error if there are any unnamed widgets.
title	Window Title
width, height	width and height of window. If not given then UI will be automatically sized.
idle_func	callback function which will be run continually while this window is open.
idle_fps	the frame rate at which this idle functino should be called. Default: 30. Set to 'NA' for running as fast as possible.
bind	bind commands to particular events on this element. This may be a single result of bind_opts() or a list of them for multiple events.
pack	a named list of pack options for how to incorporate this element into its parent container. Default: NULL means to use the standard packing. See pack_opts() as a way of creating a valid list of pack options.
pack_def	Default packing options for children of this object. This can be overriden by setting pack explicitly on child elemnets you want to control packing for.

Value

the tcl/tk window handle

tcl/tk

See tcl/tk documentation for more information on this element https://www.tcl.tk/man/tcl8.6/TkCmd/toplevel.html

See Also

Other widgets containers: tic_labelframe()

Index

∗ **canvas**
  canvas_arc, 5
  canvas_clear, 7
  canvas_image, 7
  canvas_line, 8
  canvas_oval, 9
  canvas_plot, 10
  canvas_polygon, 10
  canvas_rect, 11
  canvas_save, 12
  canvas_text, 12
  tic_canvas, 24
∗ **themes**
  set_theme, 22
∗ **widgets containers**
  tic_labelframe, 30
  tic_window, 49
∗ **widgets**
  tic_button, 22
  tic_canvas, 24
  tic_checkbutton, 25
  tic_combobox, 27
  tic_label, 29
  tic_menu, 31
  tic_menubutton, 32
  tic_menuitem, 34
  tic_progressbar, 37
  tic_radiobutton, 38
  tic_separator, 41
  tic_sizegrip, 42
  tic_slider, 43
  tic_spinbox, 44
  tic_submenu, 46
  tic_textbox, 47
  tic_textentry, 48
.DollarNames.tic_ui, 3

as.character.tic_ui, 3

bind_event, 4
bind_opts *(*bind_event*)*, 4

canvas_arc, 5, *7*, *9–13*, *25*
canvas_clear, *6*, *7*, 7, *9–13*, *25*
canvas_image, *6*, *7*, 7, *9–13*, *25*
canvas_line, *6*, *7*, 8, *9–13*, *25*
canvas_oval, *6*, *7*, *9*, 9, *10–13*, *25*
canvas_plot, *6*, *7*, *9*, 10, *11–13*, *25*
canvas_polygon, *6*, *7*, *9*, *10*, 10, *12*, *13*, *25*
canvas_rect, *6*, *7*, *9–11*, 11, *12*, *13*, *25*
canvas_save, *6*, *7*, *9–12*, 12, *13*, *25*
canvas_text, *6*, *7*, *9–12*, 12, *25*

extract_node_by_id, 13

format.reactive_textbox
  *(*format.reactive_value*)*, 14
format.reactive_value, 14

handle_images, 14

launch_idle_func, 15
list_font_families, 15
list_themes, 16

pack_opts, 16
popup_color_picker, 17
popup_messagebox, 17
popup_open_file, 18
popup_save_file *(*popup_open_file*)*, 18
print.reactive_value
  *(*format.reactive_value*)*, 14
print.tic_spec, 19
print.tic_ui, 19

reactive_chr *(*reactive_lgl*)*, 20
reactive_dbl *(*reactive_lgl*)*, 20
reactive_int *(*reactive_lgl*)*, 20
reactive_lgl, 20
reactive_textbox *(*reactive_lgl*)*, 20
render_ui, 20

set_font_family, 21
set_font_scale, 21
set_theme, 22

tic_button, 22, *25*, *27, 28*, *30*, *32, 33*, *35*, *38, 39*, *41, 42*, *44–47*, *49*
tic_canvas, *6, 7*, *9–13*, *24*, 24, *27, 28*, *30*, *32, 33*, *35*, *38, 39*, *41, 42*, *44–47*, *49*
tic_checkbutton, *24, 25*, 25, *28*, *30*, *32, 33*, *35*, *38, 39*, *41, 42*, *44–47*, *49*
tic_col *(*tic_row*)*, 40
tic_combobox, *24, 25*, *27*, 27, *30*, *32, 33*, *35*, *38, 39*, *41, 42*, *44–47*, *49*
tic_frame *(*tic_labelframe*)*, 30
tic_label, *24, 25*, *27, 28*, 29, *32, 33*, *35*, *38, 39*, *41, 42*, *44–47*, *49*
tic_labelframe, 30, *50*
tic_menu, *24, 25*, *27, 28*, *30*, 31, *33*, *35*, *38, 39*, *41, 42*, *44–47*, *49*
tic_menubutton, *24, 25*, *27, 28*, *30*, *32*, 32, *35*, *38, 39*, *41, 42*, *44–47*, *49*
tic_menuitem, *24, 25*, *27, 28*, *30*, *32, 33*, 34, *38, 39*, *41, 42*, *44–47*, *49*
tic_notebook, 35
tic_panedwindow, 36
tic_progressbar, *24, 25*, *27, 28*, *30*, *32, 33*, *35*, 37, *39*, *41, 42*, *44–47*, *49*
tic_radiobutton, *24, 25*, *27, 28*, *30*, *32, 33*, *35*, *38*, 38, *41, 42*, *44–47*, *49*
tic_row, 40
tic_separator, *24, 25*, *27, 28*, *30*, *32, 33*, *35*, *38, 39*, 41, *42*, *44–47*, *49*
tic_sizegrip, *24, 25*, *27, 28*, *30*, *32, 33*, *35*, *38, 39*, *41*, 42, *44–47*, *49*
tic_slider, *24, 25*, *27, 28*, *30*, *32, 33*, *35*, *38, 39*, *41, 42*, 43, *45–47*, *49*
tic_spinbox, *24, 25*, *27, 28*, *30*, *32, 33*, *35*, *38, 39*, *41, 42*, *44*, 44, *46, 47*, *49*
tic_submenu, *24, 25*, *27, 28*, *30*, *32, 33*, *35*, *38, 39*, *41, 42*, *44, 45*, 46, *47*, *49*
tic_textbox, *24, 25*, *27, 28*, *30*, *32, 33*, *35*, *38, 39*, *41, 42*, *44–46*, 47, *49*
tic_textentry, *24, 25*, *27, 28*, *30*, *32, 33*, *35*, *38, 39*, *41, 42*, *44–47*, 48
tic_window, *31*, 49