Package ‘gWidgets2tcltk’

October 13, 2022

Version 1.0-8

Title Toolkit Implementation of gWidgets2 for tcltk

Author John Verzani

Maintainer John Verzani <jverzani@gmail.com>

URL https://github.com/jverzani/gWidgets2tcltk

Depends R (>= 2.12.0), methods, digest, memoise, tcltk(>= 2.7.0), gWidgets2(>= 1.0.7)

Suggests testthat

Description Port of the 'gWidgets2' API for the 'tcltk' package.

License GPL (>= 2)

LazyLoad yes

Collate 'List.R' 'gWidgets2tcltk-package.R' 'misc.R' 'tcltk-misc.R' 'dnd.R' 'GComponent.R' 'GContainer.R' 'GWidget.R' 'dialogs.R' 'gmenu.R' 'gaction.R' 'gbutton.R' 'gcalendar.R' 'gcheckbox.R' 'gtable.R' 'gcheckboxgroup.R' 'gcombobox.R' 'tablelist.R' 'gdf.R' 'gedit.R' 'gexpandgroup.R' 'gfile.R' 'gformlayout.R' 'ggroup.R' 'gframe.R' 'gimage.R' 'glabel.R' 'glayout.R' 'gnotebook.R' 'gpanedgroup.R' 'gprogressbar.R' 'gradio.R' 'gseparator.R' 'gslider.R' 'gspinbutton.R' 'gstackwidget.R' 'gstatusbar.R' 'gtext.R' 'gtimer.R' 'gtoolbar.R' 'gtree.R' 'gvarbrowser.R' 'gwindow.R' 'icons.R' 'tk2tip.R' 'zzz.R'

RoxygenNote 6.1.1

NeedsCompilation no

Repository CRAN

Date/Publication 2022-02-16 23:30:02 UTC

R topics documented:

`gWidgets2tcltk-package` *gWidgets2tcltk*

Description

The **gWidgets2** package provides a programming interface for making graphical user interfaces within R. The package is a rewrite of the **gWidgets** package, introducing a few external changes but a significant number of internal ones. The package relies on one of several underlying toolkit packages providing access to the graphical libraries. This package provides the implementation to interface with the underlying **tcltk** package.

Drag and drop in **tcltk** is not supported by any underlying toolkit functionality. As such, in **gWidgets2tcltk** we roll our own. The result is a little limited: a) you can't drop values from other applications b) drop targets aren't allowed to be picky about what they receive (not by mime-type anyways).

GComponent is a parent class for both GContainer and GWidget and inherits its primary interface from gWidgets2::BasicToolkitInterface.

The `GDf` class provides a means to edit a data frame. We use the add on TK code provided by tablelist as the underlying widget

The main reference methods `GTimer` are `start_timer` and `stop_timer`

Arguments

...	passed to constructor

Details

To implement drag and drop, we bind to the toplevel window the events: button1, motion and button1 release. The binding occurs not to to widget, but to the toplevel window containing the widget. This has some advantages, most imporantly it is unlikely the binding will be overwritten by usual gWidgets2tcltk programs. (Recall tkbind will only allow one binding per widget per signal).

The gWidgets2 way of doing drag and drop is implemented here. You add a drop source with a handler that returns the value of what you want to pass via the dnd process. Then you specify a widget as a drop target and give a handler. This handler receives the data through the `dropdata` component of the "h" argument. Here, the value is generated when the drop occurs, not when the drag initiates. Not sure this makes any difference, but it might.

Simply click on a row and the editor pops up as a modal dialog. The shortcut Shift+Enter will go onto the next case, saving changes provided the auto save featuer is employed.

There is no undo/redo support here. There is no support for editing rownames (just lazy at the moment, holler if you would like that). No support to change the dimensions of the data frame or edit factors, ...

Methods

`is_watching()` Are we watching for a drag?

`add_drag_motion(handler, action = NULL, ...)` Called when motion over widget occurs

`add_to_parent(parent, child, expand = NULL, fill = NULL, anchor = NULL, ...)` Add a child to parent if it is ia container and non null. Dispatches to add_child method of parent

`get_block()` Return surround block

`get_enabled()` is widget sensistive to user input

`get_index(drop = NULL, ...)` svalue; index=TRUE

`get_tk_id()` Return tk ID

`get_toplevel_tk_id()` return id of toplevel

`get_value(drop = NULL, ...)` Get main value of widget. From ‘svalue‘ when index = FALSE or NULL

`get_widget()` Return widget (not block)

`is_tkwidget()` Is widget older style widget

`is_ttkwidget()` Is widget new style widget?. Override in subclass if not

`set_enabled(value, ...)` specify with logical if widget is sensistive to user input

`set_size(value, ...)` Set widget size (size request), value=c(width=-1, height=-1)

`set_value(value, ..., drop = NULL)` for ‘svalue<-‘ when index = FALSE or NULL

`add(...)` add is just add_child

`child_bookkeeping(child)` Update parent property of child and children property of parent container

`get_widget()` Return widget (not block)

`connect_to_toolkit_signal(signal, decorator, emitter = handler_widget(), ...)` Connect signal of toolkit to notify observer

`remove_border()` Remove border by setting relief to none

`get_length(...)` Get length of object. Needed for sapply.

`save_data(nm, where)` Save data set

`set_interval(ms)` Set the interval. Need to stop and start active timer to implement.

`start_timer()` Start the timer

`stop_timer()` stop the timer

Author(s)

John Verzani <jverzani@gmail.com>

Maintainer: John Verzani <jverzani@gmail.com>

.gmessage.guiWidgetsToolkittcltk

toolkit implementation for gmessage

Description

toolkit implementation for gmessage

toolkit implementation for gconfirm

toolkit implmentation of ginput

toolkit implementation

toolkit implementation of galert

Toolkit constructor

Toolkit constructor

Toolkit constructor

Toolkit constructor

Toolkit XXX constructor

Toolkit constructor

Toolkit constructor

Toolkit constructor

Toolkit constructor

Toolkit gedit constructor

toolkit constructor

Toolkit implementation

Toolkit constructor

Toolkit constructor

toolkit constructor for ggroup

gframe constructor

Toolkit constructor

Toolkit label constructor

Toolkit constructor

Toolkit constructor

Toolkit constructor

Toolkit constructor

Toolkit constructor

Toolkit constructor

Toolkit constructor

Toolkit XXX constructor

Toolkit constructor

Toolkit constructor

toolkit implementation of gtext

S3 method for gtimer

Toolkit constructor

Toolkit constructor

toolkit constructor for gwindow

add stock icons

Returns list of stock ids

return stock id

return stock id from object

Usage

```
## S3 method for class 'guiWidgetsToolkittcltk'
.gmessage(toolkit, msg,
  title = "message", icon = c("info", "warning", "error", "question"),
  parent = NULL, ...)

## S3 method for class 'guiWidgetsToolkittcltk'
.gconfirm(toolkit, msg,
  title = "Confirm", icon = c("info", "warning", "error", "question"),
  parent = NULL, ...)

## S3 method for class 'guiWidgetsToolkittcltk'
.ginput(toolkit, msg, text = "",
  title = "Input", icon = c("info", "warning", "error", "question"),
  parent = NULL, ...)

## S3 method for class 'guiWidgetsToolkittcltk'
.gbasicdialog(toolkit, title = "Dialog",
  parent = NULL, do.buttons = TRUE, handler = NULL, action = NULL,
  ...)

## S3 method for class 'guiWidgetsToolkittcltk'
.galert(toolkit, msg, title = "message",
  delay = 3, parent = NULL, ...)

## S3 method for class 'guiWidgetsToolkittcltk'
.gmenu(toolkit, menu.list = list(),
  popup = FALSE, container = NULL, ...)

## S3 method for class 'guiWidgetsToolkittcltk'
.gaction(toolkit, label, tooltip = NULL,
  icon = NULL, key.accel = NULL, handler = NULL, action = NULL,
  parent = NULL, ...)

## S3 method for class 'guiWidgetsToolkittcltk'
.gbutton(toolkit, text, handler, action,
  container, ...)

## S3 method for class 'guiWidgetsToolkittcltk'
.gcalendar(toolkit, text = "",
  format = "%Y-%m-%d", handler = NULL, action = NULL,
  container = NULL, ...)

## S3 method for class 'guiWidgetsToolkittcltk'
.gcheckbox(toolkit, text,
  checked = FALSE, use.togglebutton = FALSE, handler = NULL,
  action = NULL, container = NULL, ...)

## S3 method for class 'guiWidgetsToolkittcltk'
.gtable(toolkit, items,
  multiple = FALSE, chosen.col = 1, icon.col = NULL,
  tooltip.col = NULL, handler = NULL, action = NULL,
  container = NULL, ...)

## S3 method for class 'guiWidgetsToolkittcltk'
.gcheckboxgroup(toolkit=NULL, items, checked = FALSE,horizontal = FALSE,
           use.table = FALSE, handler = NULL, action = NULL, container = NULL, ...)

## S3 method for class 'guiWidgetsToolkittcltk'
.gcombobox(toolkit, items, selected = 1,
  editable = FALSE, coerce.with = NULL, handler = NULL,
  action = NULL, container = NULL, ...)

## S3 method for class 'guiWidgetsToolkittcltk'
.gdf(toolkit, items = NULL,
  handler = NULL, action = NULL, container = NULL, ...)

## S3 method for class 'guiWidgetsToolkittcltk'
.gedit(toolkit, text = "", width = 25,
  coerce.with = NULL, initial.msg = initial.msg, handler = NULL,
  action = NULL, container = NULL, ...)

## S3 method for class 'guiWidgetsToolkittcltk'
.gexpandgroup(toolkit, text, markup,
  horizontal = TRUE, handler = NULL, action = NULL,
  container = NULL, ...)

## S3 method for class 'guiWidgetsToolkittcltk'
.gfile(toolkit, text = "",
  type = c("open", "save", "selectdir"), initial.filename = NULL,
  initial.dir = getwd(), filter = list(), multi = FALSE, ...)

## S3 method for class 'guiWidgetsToolkittcltk'
.gfilebrowse(toolkit, text = "",
  type = c("open", "save", "selectdir"), initial.filename = NULL,
  initial.dir = getwd(), filter = list(), quote = TRUE,
  handler = NULL, action = NULL, container = NULL, ...)

## S3 method for class 'guiWidgetsToolkittcltk'
.gformlayout(toolkit, align = "left",
  spacing = 5, container = NULL, ...)

## S3 method for class 'guiWidgetsToolkittcltk'
.ggroup(toolkit, horizontal = TRUE,
  spacing = 5, use.scrollwindow = FALSE, container = NULL, ...)

## S3 method for class 'guiWidgetsToolkittcltk'
.gframe(toolkit, text, markup, pos,
  horizontal = TRUE, spacing = 5, container = NULL, ...)

## S3 method for class 'guiWidgetsToolkittcltk'
.gimage(toolkit, filename = "",
  dirname = "", stock.id = NULL, size = "", handler = NULL,
  action = NULL, container = NULL, ...)

## S3 method for class 'guiWidgetsToolkittcltk'
.glabel(toolkit, text = "",
  markup = FALSE, editable = FALSE, handler = NULL, action = NULL,
  container = NULL, ...)

## S3 method for class 'guiWidgetsToolkittcltk'
.glayout(toolkit, homogeneous = FALSE,
  spacing = 10, container = NULL, ...)

## S3 method for class 'guiWidgetsToolkittcltk'
.gnotebook(toolkit, tab.pos = 3,
  container = NULL, ...)

## S3 method for class 'guiWidgetsToolkittcltk'
.gpanedgroup(toolkit, horizontal = TRUE,
  container = NULL, ...)

## S3 method for class 'guiWidgetsToolkittcltk'
```

```
.gprogressbar(toolkit, value, container,
  ...)

## S3 method for class 'guiWidgetsToolkittcltk'
.gradio(toolkit, items, selected = 1,
  horizontal = FALSE, handler = NULL, action = NULL,
  container = NULL, ...)

## S3 method for class 'guiWidgetsToolkittcltk'
.gseparator(toolkit, horizontal = TRUE,
  container = NULL, ...)

## S3 method for class 'guiWidgetsToolkittcltk'
.gslider(toolkit, from = 0, to = 100,
  by = 1, value = from, horizontal = TRUE, handler = NULL,
  action = NULL, container = NULL, ...)

## S3 method for class 'guiWidgetsToolkittcltk'
.gspinbutton(toolkit, from = 0,
  to = 10, by = 1, value = from, digits = 0, handler = NULL,
  action = NULL, container = NULL, ...)

## S3 method for class 'guiWidgetsToolkittcltk'
.gstackwidget(toolkit, container = NULL,
  ...)

## S3 method for class 'guiWidgetsToolkittcltk'
.gstatusbar(toolkit, text = "",
  container = NULL, ...)

## S3 method for class 'guiWidgetsToolkittcltk'
.gtext(toolkit, text = NULL,
  width = NULL, height = 300, font.attr = NULL, wrap = TRUE,
  handler = NULL, action = NULL, container = NULL, ...)

## S3 method for class 'guiWidgetsToolkittcltk'
.gtimer(toolkit, ms, FUN, data = NULL,
  one.shot = FALSE, start = TRUE)

## S3 method for class 'guiWidgetsToolkittcltk'
.gtree(toolkit, offspring = NULL,
  offspring.data = NULL, chosen.col = 1, offspring.col = 2,
  icon.col = NULL, tooltip.col = NULL, multiple = FALSE,
  handler = NULL, action = NULL, container = NULL, ...)

## S3 method for class 'guiWidgetsToolkittcltk'
.gvarbrowser(toolkit, handler = NULL,
  action = "summary", container = NULL, ...)
```

```
## S3 method for class 'guiWidgetsToolkittcltk'
.gwindow(toolkit, title,
  visible = visible, name, width, height, parent, handler, action, ...)

## S3 method for class 'guiWidgetsToolkittcltk'
.addStockIcons(toolkit, iconNames,
  iconFiles, ...)

## S3 method for class 'guiWidgetsToolkittcltk'
.getStockIcons(toolkit, ...)

## S3 method for class 'guiWidgetsToolkittcltk'
.getStockIconByName(toolkit,name, ...)

## S3 method for class 'guiWidgetsToolkittcltk'
.stockIconFromObject(toolkit,obj, ...)
```

Arguments

`toolkit`	toolkit
`msg`	Character. message to display.
`title`	Character. Title
`icon`	What icon to show
`parent`	Hint as to where to display
`...`	ignored
`text`	Character. Initial text
`do.buttons`	FALSE to suppress buttons when no parent
`handler`	handler called when `Ok` button invoked
`action`	passed to handler for OK button
`delay`	delay
`menu.list`	A list defining the menu structure. Named sub lists determine the submenu titles and structure. The list may have components of class: `GAction`, mapped to a button; `GSeparator`, mapped to a horizontal separator; `GRadio`, mapped to linked buttons; or `GCheckbox`, mapped to a checkbox button.
`popup`	logical. If true, make a popup window to be added through a handler call
`container`	A parent container. When a widget is created it can be incorporated into the widget heirarchy by passing in a parent container at construction time. (For some toolkits this is not optional, e.g. **gWidgets2tcltk** or **gWidgets2WWW2**.)
`label`	label for action
`tooltip`	toolktip for actin
`key.accel`	keyboard accelerator. If given, parent must be specified.
`format`	Date format

checked is button selected

use.togglebutton
Use a toggle button (shows depressed) not a check box

items data.frame specifies items for selection. May be a vector, matrix or data frame

multiple logical allow multiple selection

chosen.col which value from the row is returned by selection

icon.col NULL or integer. If latter, specifies column containing stock icon

tooltip.col NULL or integer. If latter, specifies column containing tooltip

horizontal logical. If true displayed horizontally, else vertically

use.table logical. If supported, and TRUE then uses a table widget with scrollbars

selected integer. Which item (by index) is selected. Use -1 for no selection

editable logical. Is user allowed to edit value

coerce.with A function of function name to be called before selected value is returned by svalue

width width of widget

initial.msg initial.msg

markup does label use markup (toolkit specific)

type type of browser: to open a file, to save a file or to select a directory

initial.filename
Suggested file name

initial.dir initial directory. If a filename is given, and is not an absolute name, this will be prepended. If filename given initial directory will be taken from that.

filter A filter specifiation. This can be a named character vector of file extensions or something toolkit specific. Here are some examples:

- characterc("csv"="csv","txt"="txt")
- RGtk2 Something like

```
list("All files" = list(patterns = c("*")),
     "R files" = list(patterns = c("*.R", "*.Rdata")),
     "text files" = list(mime.types = c("text/plain"))
    )
```

- tcltk
- Qt

multi Logical. Allow multiple files to be selected?

quote quote output

align alignment of label. Left justify or center balance. Leave as "default" for underlying toolkit default.

spacing spacing between columns

`use.scrollwindow`
logical. Either `TRUE`, `"TRUE"`, `FALSE`, `"FALSE"`, `"y"`, or `"x"`. For all toolkits a non-FALSE value will place the child components into a scrollable container. For some toolkits this will only be in the direction of packing. If the toolkit allows it (RGtk2), then values of `"x"` or `"y"` can be used to override the default scrolling directions. A box container with scrollwindows should have it size set either directly or through packing with `expand=TRUE` as its size request will not reflect the size of its child components.

`pos` position of label: 0=left, 1=right, some toolkit allow values in between

`filename` basename of file

`dirname` dirname of file

`stock.id` stock id of icon (if non NULL)

`size` size of icon when a stock id (toolkit dependent)

`homogeneous` are cells all the same size

`tab.pos` integer. Position of tabs, 1 on bottom, 2 left, 3 top, 4 right. (If supported)

`value` a list or menu bar specifying the new men ubar

`from` If a number of length one then a starting point, in which case to, by are passed to `seq`. Otherwise a sequence of values for which sort(unique(from)) will order

`to` ending point when from is starting point

`by` step size if not specified by `from`

`digits` digits

`height` height of widget (when width is specified)

`font.attr` font attributes for text buffer. One can also specify font attributes for insertion. The font attributes are specified with a list with named components, with names and values coming from:

- **weight** in c("light", "normal", "bold", "heavy")
- **style** inc("normal", "oblique", "italic")
- **family** in c("sans", "helvetica", "times", "monospace")
- **size** in c("xx-small", "x-small", "small", "medium", "large", "x-large", "xx-large")
- **foreground** a value in colors()
- **background** a value in colors()

`wrap` logical do lines wrap

`ms` interval in milliseconds

`FUN` FUnction to call. Has one argument, data passed in

`data` passed to function

`one.shot` logical. If TRUE, called just once, else repeats

`start` logical. If FALSE, started by `start_timer` OO method. (Call `obj$start_time()`).

`offspring` function. A function passed values `path` and `data`, the latter from `offspring.data`. The path is the current position of the parent item using the named keys from the chosen column.

`offspring.data`	Passed to second argument of `offspring` function. Used to parameterize a function call.
`offspring.col`	integer or column name. Points to column containing logical values indicating if a row has offspring.
`visible`	logical. If codeTRUE window is drawn when constructed. Otherwise, window can be drawn later using `visible<-`. This value can default to `FALSE` by setting the option: `options("gWidgets:gwindow-default-visible-is-false"=TRUE)`. There are advantages: windows can draw slowly when adding many items. With **gWidgets2RGtk2**, the `ggraphics` widget can like to be added to an undrawn widget as this avoids sizing issue.
`name`	name of icon
`iconNames`	names of icons
`iconFiles`	path of icons
`obj`	obj to get icon from

See Also

The documentation for this is found at `gbutton`.

The documentation for this is found at `gprogressbar`.

`GComponentObservable` *GComponentObservable adds the observable interface*

Description

GComponentObservable adds the observable interface

Usage

```
GComponentObservable(...)
```

Arguments

`...`	passed to constructor

GEdit-class *show the word list*

Description

show the word list

Arguments

`str`	a string. If missing do nothing, otherwise match against string to generate word list. Popup menu depending on length

Methods

`add_bindings()` Add bindings to the entry box

`clear_error()` Clear error message

`clear_init_txt(...)` clear out init text, set back to black

`find_match(x)` Find match in word list

`make_styles(bg = "#ff6622")` Create tcl styles, cf http://paste.tclers.tk/506

`set_error(msg)` Add error state and message to widget

`set_init_txt(msg)` set initial text, gray out

`set_invalid(value, msg)` Set widget as invalid or not

`set_validator(FUN)` Set a function to do the validation

`validate_input()` Return logical indicating if input is valid

getWidget.tkwin *method for getWidget defined in gWidgets2*

Description

method for getWidget defined in gWidgets2

Usage

```
## S3 method for class 'tkwin'
getWidget(obj)
```

Arguments

`obj`	object

GMenuBar *Toplevel menu bar*

Description

The GMenuBar class provides functionality for a top-level menubar. In **tlctk** the menubars can show either gaction items (proxied as buttons), gradio items or gcheckbox items. In the latter two cases, one uses the parent argument – not the container argument – to specify the parent container. Such items can also be shared with toolbars.

Usage

```
GMenuBar(...)
```

Arguments

...	passed to constructor

Examples

```
## Not run:
w <- gwindow("having fun?")
sb <- gstatusbar("Your message here...", cont=w)
g <- ggroup(cont=w, horizontal=FALSE)
f <- function(h,...) message(h$obj$get_value())

l <- list(file=gaction("file", handler=function(h,...) print("file"),
               key.accel="<Control-x><Control-s>", parent=w),
          ok=gaction("ok", icon="ok", handler=function(h,...) print("ok")),
          radio=list(
            rb=gradio(state.name[1:3], parent=w, handler=function(h,...)
               print(h$obj$get_value()))
            )
          sep=gseparator(vertical=TRUE),
       ,cb=gcheckbox("really", parent=w, handler=function(h,...) print(h$obj$get_value()))
          )
mlist <- list(File=l)
mb <- gmenu(mlist, cont=w)

## End(Not run)
```

`GSpinButton` *Spinbutton class*

Description

Spinbutton class

Usage

```
GSpinButton(...)
```

Arguments

`...`	passed to constructor `GSpinButton` is the base class for spin buttons. The widget is buggy in tcltk, atleast with the old non-themed style and a Mac running tcltk under X11. The bug is the thing keeps spinning when the buttons are pressed.

`GStatusBar` `GStatusBar` *is the base class for a status bar*

Description

The `GStatusBar` class inherits for `GBoxContainer` meaning it can be used as a parent container. As such, one can add additional widgets beyond the plain label that is the main property of this widget.

Usage

```
GStatusBar(...)
```

Arguments

`...`	passed to constructor

GToolBar-class *GToolBar is the base class for toolbars*

Description

The toolbar is a container, so can have other widgets added to it as though it were a box container. Buttons should be added as action items, so that they are rendered in the proper style. Check buttons should be given the argument `use.togglebutton`. Use `addSpring` to right align items.

Arguments

`...`	passed to constructor

Methods

`add_gaction_toolitem(obj)` Helper to add a gaction item

`add_gseparator_toolitem()` Helper to add a separator

`add_toolbar_items(items)` Map a toolbar list, a named list of gaction items or gsepartor items

`clear_toolbar()` Clear toolbar items

`get_widget()` What widget do we use for the parent of the children

`set_value(value, ...)` We can't really adjust spacing between children after they have been positioned.

guiWidgetsToolkittcltk-class *toolkit class for tlctk*

Description

toolkit class for tlctk

gwidgets2_tcltk_column_alignment

align a column based on the class of the content

Description

Gives visual difference to data based on its class

Usage

```
gwidgets2_tcltk_column_alignment(x)

## Default S3 method:
gwidgets2_tcltk_column_alignment(x)

## S3 method for class 'numeric'
gwidgets2_tcltk_column_alignment(x)

## S3 method for class 'logical'
gwidgets2_tcltk_column_alignment(x)
```

Arguments

x	column data to align

Value

anchor string

gwidgets2_tcltk_format_to_char

Format object to character class for inclusion in a table

Description

Gives chance to do more than `as.character`

Usage

```
gwidgets2_tcltk_format_to_char(x)

## Default S3 method:
gwidgets2_tcltk_format_to_char(x)

## S3 method for class 'factor'
```

```
gwidgets2_tcltk_format_to_char(x)

## S3 method for class 'integer'
gwidgets2_tcltk_format_to_char(x)

## S3 method for class 'numeric'
gwidgets2_tcltk_format_to_char(x)

## S3 method for class 'Date'
gwidgets2_tcltk_format_to_char(x)

## S3 method for class 'data.frame'
gwidgets2_tcltk_format_to_char(x)
```

Arguments

`x`	object to format to character class

Value

object of character class with possible formatting.

`List-class` *A list extension class.*

Description

Like a list, but has some methods. Completely superflous, but makes copying some code algorithms easier. We implement methods such as `append`, `push`, `pop` and `each` for iteration. As well, there are some lookup methods.

Arguments

`...`	passed to constructor

Methods

`contains(name)` TRUE if name is key in array

`core()` return list

`each(FUN, ...)` Iterator for lists, like sapply, but FUN gets passed index, key, and value

`flush(...)` Reset array, return contents as list

`get_by_name(name)` get item under name

`get_id()` Return an id, or name, for an object

`get_item(index)` Get item by index

`insert(x, name, index)` Insert item into List with 0 the head and index=len() the tail `len()` length

`pluck(id, FUN, ...)` Like ext.pluck. Returns array with 'id' extracted from each item in the List

`pop()` pop last element of list

`push(x, name)` Append x with optional name. If name not specified new id created. Returns name

makeCalendar *make a calendar...*

Description

make a calendar...

Usage

```
makeCalendar(date_var, widget, date, date_format = "%Y-%m-%d",
  set_value)
```

Arguments

`date_var` var

`widget` var

`date` var

`date_format` var

`set_value` var from chron with slight change to arguments

Index