Guile-GNOME: Gtk

version 2.15.93, updated 2 September 2007

Damon Chaplin
many others

This manual is for `(gnome gtk)` (version 2.15.93, updated 2 September 2007)

Short Contents

1 Overview

The GTK+ wrapper for Guile is a part of Guile-GNOME. Maybe write more here at some point.

2 GtkDialog

Create popup windows

2.1 Overview

Dialog boxes are a convenient way to prompt the user for a small amount of input, e.g. to display a message, ask a question, or anything else that does not require extensive effort on the user's part.

GTK+ treats a dialog as a window split vertically. The top section is a `<gtk-vbox>`, and is where widgets such as a `<gtk-label>` or a `<gtk-entry>` should be packed. The bottom area is known as the . This is generally used for packing buttons into the dialog which may perform functions such as cancel, ok, or apply. The two areas are separated by a `<gtk-hseparator>`.

`<gtk-dialog>` boxes are created with a call to `gtk-dialog-new` or `gtk-dialog-new-with-buttons`. `gtk-dialog-new-with-buttons` is recommended; it allows you to set the dialog title, some convenient flags, and add simple buttons.

If 'dialog' is a newly created dialog, the two primary areas of the window can be accessed as '`GTK_DIALOG(dialog)->vbox`' and '`GTK_DIALOG(dialog)->action_area`', as can be seen from the example, below.

A 'modal' dialog (that is, one which freezes the rest of the application from user input), can be created by calling `gtk-window-set-modal` on the dialog. Use the `gtk-window` macro to cast the widget returned from `gtk-dialog-new` into a `<gtk-window>`. When using `gtk-dialog-new-with-buttons` you can also pass the `<gtk-dialog-modal>` flag to make a dialog modal.

If you add buttons to `<gtk-dialog>` using `gtk-dialog-new-with-buttons`, `gtk-dialog-add-button`, `gtk-dialog-add-buttons`, or `gtk-dialog-add-action-widget`, clicking the button will emit a signal called `"response"` with a response ID that you specified. GTK+ will never assign a meaning to positive response IDs; these are entirely user-defined. But for convenience, you can use the response IDs in the `<gtk-response-type>` enumeration (these all have values less than zero). If a dialog receives a delete event, the `"response"` signal will be emitted with a response ID of `<gtk-response-delete-event>`.

If you want to block waiting for a dialog to return before returning control flow to your code, you can call `gtk-dialog-run`. This function enters a recursive main loop and waits for the user to respond to the dialog, returning the response ID corresponding to the button the user clicked.

For the simple dialog in the following example, in reality you'd probably use `<gtk-message-dialog>` to save yourself some effort. But you'd need to create the dialog contents manually if you had more than a simple message in the dialog.

```
/* Function to open a dialog box displaying the message provided. */

void quick_message (gchar *message) {
```

```
    GtkWidget *dialog, *label;

    /* Create the widgets */

    dialog = gtk_dialog_new_with_buttons ("Message",
                                          main_application_window,
                                          GTK_DIALOG_DESTROY_WITH_PARENT,
                                          GTK_STOCK_OK,
                                          GTK_RESPONSE_NONE,
                                          NULL);
    label = gtk_label_new (message);

    /* Ensure that the dialog box is destroyed when the user responds. */

    g_signal_connect_swapped (dialog,
                              "response",
                              G_CALLBACK (gtk_widget_destroy),
                              dialog);

    /* Add the label, and show everything we've added to the dialog. */

    gtk_container_add (GTK_CONTAINER (GTK_DIALOG(dialog)->vbox),
                       label);
    gtk_widget_show_all (dialog);
}
```

2.2 Usage

`<gtk-dialog>` [Class]

This `<gobject>` class defines the following properties:

`has-separator`
: The dialog has a separator bar above its buttons

`response` (*arg0* `<gint>`) [Signal on `<gtk-dialog>`]

Emitted when an action widget is clicked, the dialog receives a delete event, or the application programmer calls `gtk-dialog-response`. On a delete event, the response ID is `<gtk-response-none>`. Otherwise, it depends on which action widget was clicked.

`close` [Signal on `<gtk-dialog>`]

`gtk-dialog-run` (*self* `<gtk-dialog>`) ⇒ (*ret* `int`) [Function]

`run` [Method]

Blocks in a recursive main loop until the *dialog* either emits the response signal, or is destroyed. If the dialog is destroyed during the call to `gtk-dialog-run`, gtk_dialog_returns `<gtk-response-none>`. Otherwise, it returns the response ID

from the "response" signal emission. Before entering the recursive main loop, `gtk-dialog-run` calls `gtk-widget-show` on the dialog for you. Note that you still need to show any children of the dialog yourself.

During `gtk-dialog-run`, the default behavior of "delete_event" is disabled; if the dialog receives "delete_event", it will not be destroyed as windows usually are, and `gtk-dialog-run` will return `<gtk-response-delete-event>`. Also, during `gtk-dialog-run` the dialog will be modal. You can force `gtk-dialog-run` to return at any time by calling `gtk-dialog-response` to emit the "response" signal. Destroying the dialog during `gtk-dialog-run` is a very bad idea, because your post-run code won't know whether the dialog was destroyed or not.

After `gtk-dialog-run` returns, you are responsible for hiding or destroying the dialog if you wish to do so.

Typical usage of this function might be:

```
gint result = gtk_dialog_run (GTK_DIALOG (dialog));
switch (result)
  {
    case GTK_RESPONSE_ACCEPT:
       do_application_specific_something ();
       break;
    default:
       do_nothing_since_dialog_was_cancelled ();
       break;
  }
gtk_widget_destroy (dialog);
```

Note that even though the recursive main loop gives the effect of a modal dialog (it prevents the user from interacting with other windows in the same window group while the dialog is run), callbacks such as timeouts, IO channel watches, DND drops, etc, *will* be triggered during a `gtk-dialog-run` call.

dialog a `<gtk-dialog>`

ret response ID

`gtk-dialog-response` (*self* `<gtk-dialog>`) (*response_id* `int`) [Function]
`response` [Method]

Emits the "response" signal with the given response ID. Used to indicate that the user has responded to the dialog in some way; typically either you or `gtk-dialog-run` will be monitoring the "response" signal and take appropriate action.

dialog a `<gtk-dialog>`

response-id
response ID

`gtk-dialog-add-button` (*self* `<gtk-dialog>`) (*button_text* `mchars`) [Function]
(*response_id* `int`) ⇒ (*ret* `<gtk-widget>`)

`add-button` [Method]

Adds a button with the given text (or a stock button, if *button-text* is a stock ID) and sets things up so that clicking the button will emit the `"response"` signal with the given *response-id*. The button is appended to the end of the dialog's action area. The button widget is returned, but usually you don't need it.

dialog a `<gtk-dialog>`

button-text
text of button, or stock ID

response-id
response ID for the button

ret the button widget that was added

`gtk-dialog-add-action-widget` (*self* `<gtk-dialog>`) (*child* `<gtk-widget>`) (*response_id* `int`) [Function]

`add-action-widget` [Method]

Adds an activatable widget to the action area of a `<gtk-dialog>`, connecting a signal handler that will emit the `"response"` signal on the dialog when the widget is activated. The widget is appended to the end of the dialog's action area. If you want to add a non-activatable widget, simply pack it into the '`action_area`' field of the `<gtk-dialog>` struct.

dialog a `<gtk-dialog>`

child an activatable widget

response-id
response ID for *child*

`gtk-dialog-get-has-separator` (*self* `<gtk-dialog>`) ⇒ (*ret* `bool`) [Function]

`get-has-separator` [Method]

Accessor for whether the dialog has a separator.

dialog a `<gtk-dialog>`

ret '`#t`' if the dialog has a separator

`gtk-dialog-set-default-response` (*self* `<gtk-dialog>`) (*response_id* `int`) [Function]

`set-default-response` [Method]

Sets the last widget in the dialog's action area with the given *response-id* as the default widget for the dialog. Pressing `"Enter"` normally activates the default widget.

dialog a `<gtk-dialog>`

response-id
a response ID

`gtk-dialog-set-has-separator` (*self* `<gtk-dialog>`) (*setting* `bool`) [Function]

`set-has-separator` [Method]

Sets whether the dialog has a separator above the buttons. '`#t`' by default.

dialog a `<gtk-dialog>`

setting '`#t`' to have a separator

`gtk-dialog-set-response-sensitive` (*self* `<gtk-dialog>`) [Function]
(*response_id* `int`) (*setting* `bool`)

`set-response-sensitive` [Method]

Calls '`gtk_widget_set_sensitive (widget, setting)`' for each widget in the dialog's action area with the given *response-id*. A convenient way to sensitize/desensitize dialog buttons.

dialog a `<gtk-dialog>`

response-id
a response ID

setting '`#t`' for sensitive

`gtk-dialog-get-response-for-widget` (*self* `<gtk-dialog>`) [Function]
(*widget* `<gtk-widget>`) ⇒ (*ret* `int`)

`get-response-for-widget` [Method]

Gets the response id of a widget in the action area of a dialog.

dialog a `<gtk-dialog>`

widget a widget in the action area of *dialog*

ret the response id of *widget*, or '`GTK_RESPONSE_NONE`' if *widget* doesn't have a response id set.

Since 2.8

`gtk-alternative-dialog-button-order` (*screen* `<gdk-screen>`) [Function]
⇒ (*ret* `bool`)

Returns '`#t`' if dialogs are expected to use an alternative button order on the screen *screen*. See `gtk-dialog-set-alternative-button-order` for more details about alternative button order.

If you need to use this function, you should probably connect to the ::notify:gtk-alternative-button-order signal on the `<gtk-settings>` object associated to *screen*, in order to be notified if the button order setting changes.

screen a `<gdk-screen>`, or '`#f`' to use the default screen

ret Whether the alternative button order should be used

Since 2.6

3 GtkInvisible

A widget which is not displayed

3.1 Overview

The <gtk-invisible> widget is used internally in GTK+, and is probably not very useful for application developers.

It is used for reliable pointer grabs and selection handling in the code for drag-and-drop.

3.2 Usage

<gtk-invisible> [Class]

This <gobject> class defines the following properties:

screen The screen where this window will be displayed

gtk-invisible-new ⇒ (*ret* <gtk-widget>) [Function]

Creates a new <gtk-invisible>.

ret a new <gtk-invisible>.

gtk-invisible-new-for-screen (*screen* <gdk-screen>) [Function]
⇒ (*ret* <gtk-widget>)

Creates a new <gtk-invisible> object for a specified screen

screen a <gdk-screen> which identifies on which the new <gtk-invisible> will be created.

ret a newly created <gtk-invisible> object

Since 2.2

gtk-invisible-set-screen (*self* <gtk-invisible>) [Function]
(*screen* <gdk-screen>)

set-screen [Method]

Sets the <gdk-screen> where the <gtk-invisible> object will be displayed.

invisible a <gtk-invisible>.

screen a <gdk-screen>.

Since 2.2

gtk-invisible-get-screen (*self* <gtk-invisible>) [Function]
⇒ (*ret* <gdk-screen>)

get-screen [Method]

Returns the <gdk-screen> object associated with *invisible*

invisible a <gtk-invisible>.

ret the associated <gdk-screen>.

Since 2.2

4 GtkMessageDialog

A convenient message window

4.1 Overview

<gtk-message-dialog> presents a dialog with an image representing the type of message (Error, Question, etc.) alongside some message text. It's simply a convenience widget; you could construct the equivalent of <gtk-message-dialog> from <gtk-dialog> without too much effort, but <gtk-message-dialog> saves typing.

The easiest way to do a modal message dialog is to use gtk-dialog-run, though you can also pass in the 'GTK_DIALOG_MODAL' flag, gtk-dialog-run automatically makes the dialog modal and waits for the user to respond to it. gtk-dialog-run returns when any dialog button is clicked.

```
dialog = gtk_message_dialog_new (main_application_window,
                                 GTK_DIALOG_DESTROY_WITH_PARENT,
                                 GTK_MESSAGE_ERROR,
                                 GTK_BUTTONS_CLOSE,
                                 "Error loading file '%s': %s",
                                 filename, g_strerror (errno));
gtk_dialog_run (GTK_DIALOG (dialog));
gtk_widget_destroy (dialog);
```

You might do a non-modal <gtk-message-dialog> as follows:

```
dialog = gtk_message_dialog_new (main_application_window,
                                 GTK_DIALOG_DESTROY_WITH_PARENT,
                                 GTK_MESSAGE_ERROR,
                                 GTK_BUTTONS_CLOSE,
                                 "Error loading file '%s': %s",
                                 filename, g_strerror (errno));

/* Destroy the dialog when the user responds to it (e.g. clicks a button) */
g_signal_connect_swapped (dialog, "response",
                          G_CALLBACK (gtk_widget_destroy),
                          dialog);
```

4.2 Usage

<gtk-message-dialog> [Class]

This <gobject> class defines the following properties:

message-type
: The type of message

buttons
: The buttons shown in the message dialog

text
: The primary text of the message dialog

`use-markup`
: The primary text of the title includes Pango markup.

`secondary-text`
: The secondary text of the message dialog

`secondary-use-markup`
: The secondary text includes Pango markup.

`image`
: The image

`gtk-message-dialog-set-markup` (*self* `<gtk-message-dialog>`) (*str* `mchars`) [Function]

`set-markup` [Method]

Sets the text of the message dialog to be *str*, which is marked up with the Pango text markup language.

message-dialog
: a `<gtk-message-dialog>`

str
: markup string (see Pango markup format)

Since 2.4

`gtk-message-dialog-set-image` (*self* `<gtk-message-dialog>`) (*image* `<gtk-widget>`) [Function]

`set-image` [Method]

Sets the dialog's image to *image*.

dialog
: a `<gtk-message-dialog>`

image
: the image

Since 2.10

5 GtkWindow

Toplevel which can contain other widgets

5.1 Overview

5.2 Usage

`<gtk-window>` [Class]

This `<gobject>` class defines the following properties:

`type` The type of the window

`title` The title of the window

`role` Unique identifier for the window to be used when restoring a session

`startup-id`
Unique startup identifier for the window used by startup-notification

`allow-shrink`
If TRUE, the window has no mimimum size. Setting this to TRUE is 99% of the time a bad idea

`allow-grow`
If TRUE, users can expand the window beyond its minimum size

`resizable`
If TRUE, users can resize the window

`modal` If TRUE, the window is modal (other windows are not usable while this one is up)

`window-position`
The initial position of the window

`default-width`
The default width of the window, used when initially showing the window

`default-height`
The default height of the window, used when initially showing the window

`destroy-with-parent`
If this window should be destroyed when the parent is destroyed

`icon` Icon for this window

`icon-name`
Name of the themed icon for this window

`screen` The screen where this window will be displayed

`type-hint`
Hint to help the desktop environment understand what kind of window this is and how to treat it.

`skip-taskbar-hint`
: TRUE if the window should not be in the task bar.

`skip-pager-hint`
: TRUE if the window should not be in the pager.

`urgency-hint`
: TRUE if the window should be brought to the user's attention.

`accept-focus`
: TRUE if the window should receive the input focus.

`focus-on-map`
: TRUE if the window should receive the input focus when mapped.

`decorated`
: Whether the window should be decorated by the window manager

`deletable`
: Whether the window frame should have a close button

`gravity`
: The window gravity of the window

`transient-for`
: The transient parent of the dialog

`opacity`
: The opacity of the window, from 0 to 1

`is-active`
: Whether the toplevel is the current active window

`has-toplevel-focus`
: Whether the input focus is within this GtkWindow

`set-focus` (*arg0* `<gtk-widget>`) [Signal on `<gtk-window>`]

`frame-event` (*arg0* `<gdk-event>`) ⇒ `<gboolean>` [Signal on `<gtk-window>`]

`activate-focus` [Signal on `<gtk-window>`]

`activate-default` [Signal on `<gtk-window>`]

`keys-changed` [Signal on `<gtk-window>`]

`gtk-window-new` (*type* `<gtk-window-type>`) ⇒ (*ret* `<gtk-widget>`) [Function]

Creates a new `<gtk-window>`, which is a toplevel window that can contain other widgets. Nearly always, the type of the window should be `<gtk-window-toplevel>`. If you're implementing something like a popup menu from scratch (which is a bad idea, just use `<gtk-menu>`), you might use `<gtk-window-popup>`. `<gtk-window-popup>` is not for dialogs, though in some other toolkits dialogs are called "popups". In GTK+, `<gtk-window-popup>` means a pop-up menu or pop-up tooltip. On X11, popup windows are not controlled by the window manager.

If you simply want an undecorated window (no window borders), use `gtk-window-set-decorated`, don't use `<gtk-window-popup>`.

type type of window

ret a new `<gtk-window>`.

`gtk-window-set-title` (*self* `<gtk-window>`) (*title* `mchars`) [Function]
`set-title` [Method]

Sets the title of the `<gtk-window>`. The title of a window will be displayed in its title bar; on the X Window System, the title bar is rendered by the window manager, so exactly how the title appears to users may vary according to a user's exact configuration. The title should help a user distinguish this window from other windows they may have open. A good title might include the application name and current document filename, for example.

window a `<gtk-window>`

title title of the window

`gtk-window-set-wmclass` (*self* `<gtk-window>`) [Function]
(*wmclass_name* `mchars`) (*wmclass_class* `mchars`)
`set-wmclass` [Method]

Don't use this function. It sets the X Window System "class" and "name" hints for a window. According to the ICCCM, you should always set these to the same value for all windows in an application, and GTK+ sets them to that value by default, so calling this function is sort of pointless. However, you may want to call `gtk-window-set-role` on each window in your application, for the benefit of the session manager. Setting the role allows the window manager to restore window positions when loading a saved session.

window a `<gtk-window>`

wmclass-name
window name hint

wmclass-class
window class hint

`gtk-window-set-resizable` (*self* `<gtk-window>`) (*resizable* `bool`) [Function]
`set-resizable` [Method]

Sets whether the user can resize a window. Windows are user resizable by default.

window a `<gtk-window>`

resizable '`#t`' if the user can resize this window

`gtk-window-get-resizable` (*self* `<gtk-window>`) ⇒ (*ret* `bool`) [Function]
`get-resizable` [Method]

Gets the value set by `gtk-window-set-resizable`.

window a `<gtk-window>`

ret '`#t`' if the user can resize the window

`gtk-window-add-accel-group` (*self* `<gtk-window>`) [Function]
(*accel_group* `<gtk-accel-group>`)
`add-accel-group` [Method]

Associate *accel-group* with *window*, such that calling `gtk-accel-groups-activate` on *window* will activate accelerators in *accel-group*.

window window to attach accelerator group to

accel-group
a `<gtk-accel-group>`

`gtk-window-remove-accel-group` (*self* `<gtk-window>`) [Function]
(*accel_group* `<gtk-accel-group>`)
`remove-accel-group` [Method]

Reverses the effects of `gtk-window-add-accel-group`.

window a `<gtk-window>`

accel-group
a `<gtk-accel-group>`

`gtk-window-activate-focus` (*self* `<gtk-window>`) ⇒ (*ret* `bool`) [Function]
`activate-focus` [Method]

Activates the current focused widget within the window.

window a `<gtk-window>`

ret '`#t`' if a widget got activated.

`gtk-window-activate-default` (*self* `<gtk-window>`) ⇒ (*ret* `bool`) [Function]
`activate-default` [Method]

Activates the default widget for the window, unless the current focused widget has been configured to receive the default action (see `<gtk-receives-default>` in `<gtk-widget-flags>`), in which case the focused widget is activated.

window a `<gtk-window>`

ret '`#t`' if a widget got activated.

`gtk-window-set-modal` (*self* `<gtk-window>`) (*modal* `bool`) [Function]
`set-modal` [Method]

Sets a window modal or non-modal. Modal windows prevent interaction with other windows in the same application. To keep modal dialogs on top of main application windows, use `gtk-window-set-transient-for` to make the dialog transient for the parent; most window managers will then disallow lowering the dialog below the parent.

window a `<gtk-window>`

modal whether the window is modal

`gtk-window-set-default-size` (*self* `<gtk-window>`) (*width* `int`) [Function]
(*height* `int`)
`set-default-size` [Method]

Sets the default size of a window. If the window's "natural" size (its size request) is larger than the default, the default will be ignored. More generally, if the default size does not obey the geometry hints for the window (`gtk-window-set-geometry-hints` can be used to set these explicitly), the default size will be clamped to the nearest permitted size.

Unlike `gtk-widget-set-size-request`, which sets a size request for a widget and thus would keep users from shrinking the window, this function only sets the initial

size, just as if the user had resized the window themselves. Users can still shrink the window again as they normally would. Setting a default size of -1 means to use the "natural" default size (the size request of the window).

For more control over a window's initial size and how resizing works, investigate `gtk-window-set-geometry-hints`.

For some uses, `gtk-window-resize` is a more appropriate function. `gtk-window-resize` changes the current size of the window, rather than the size to be used on initial display. `gtk-window-resize` always affects the window itself, not the geometry widget.

The default size of a window only affects the first time a window is shown; if a window is hidden and re-shown, it will remember the size it had prior to hiding, rather than using the default size.

Windows can't actually be 0x0 in size, they must be at least 1x1, but passing 0 for *width* and *height* is OK, resulting in a 1x1 default size.

window a `<gtk-window>`

width width in pixels, or -1 to unset the default width

height height in pixels, or -1 to unset the default height

`gtk-window-set-geometry-hints` (*self* `<gtk-window>`) [Function]
(*geometry_widget* `<gtk-widget>`) (*geometry* `<gdk-geometry*>`)
(*geom_mask* `<gdk-window-hints>`)

`set-geometry-hints` [Method]

This function sets up hints about how a window can be resized by the user. You can set a minimum and maximum size; allowed resize increments (e.g. for xterm, you can only resize by the size of a character); aspect ratios; and more. See the `<gdk-geometry>` struct.

window a `<gtk-window>`

geometry-widget
widget the geometry hints will be applied to

geometry struct containing geometry information

geom-mask
mask indicating which struct fields should be paid attention to

`gtk-window-set-gravity` (*self* `<gtk-window>`) [Function]
(*gravity* `<gdk-gravity>`)

`set-gravity` [Method]

Window gravity defines the meaning of coordinates passed to `gtk-window-move`. See `gtk-window-move` and `<gdk-gravity>` for more details.

The default window gravity is `<gdk-gravity-north-west>` which will typically "do what you mean."

window a `<gtk-window>`

gravity window gravity

`gtk-window-get-gravity` (*self* `<gtk-window>`) [Function]
⇒ (*ret* `<gdk-gravity>`)

`get-gravity` [Method]

Gets the value set by `gtk-window-set-gravity`.

window a `<gtk-window>`

ret window gravity

`gtk-window-set-position` (*self* `<gtk-window>`) [Function]
(*position* `<gtk-window-position>`)

`set-position` [Method]

Sets a position constraint for this window. If the old or new constraint is '`GTK_WIN_POS_CENTER_ALWAYS`', this will also cause the window to be repositioned to satisfy the new constraint.

window a `<gtk-window>`.

position a position constraint.

`gtk-window-set-transient-for` (*self* `<gtk-window>`) [Function]
(*parent* `<gtk-window>`)

`set-transient-for` [Method]

Dialog windows should be set transient for the main application window they were spawned from. This allows window managers to e.g. keep the dialog on top of the main window, or center the dialog over the main window. `gtk-dialog-new-with-buttons` and other convenience functions in GTK+ will sometimes call `gtk-window-set-transient-for` on your behalf.

On Windows, this function will and put the child window on top of the parent, much as the window manager would have done on X.

window a `<gtk-window>`

parent parent window

`gtk-window-set-destroy-with-parent` (*self* `<gtk-window>`) [Function]
(*setting* `bool`)

`set-destroy-with-parent` [Method]

If *setting* is '`#t`', then destroying the transient parent of *window* will also destroy *window* itself. This is useful for dialogs that shouldn't persist beyond the lifetime of the main window they're associated with, for example.

window a `<gtk-window>`

setting whether to destroy *window* with its transient parent

`gtk-window-set-screen` (*self* `<gtk-window>`) (*screen* `<gdk-screen>`) [Function]

`set-screen` [Method]

Sets the `<gdk-screen>` where the *window* is displayed; if the window is already mapped, it will be unmapped, and then remapped on the new screen.

window a `<gtk-window>`.

screen a `<gdk-screen>`.

Since 2.2

`gtk-window-get-screen` (*self* `<gtk-window>`) ⇒ (*ret* `<gdk-screen>`) [Function]
`get-screen` [Method]

Returns the `<gdk-screen>` associated with *window*.

window a `<gtk-window>`.

ret a `<gdk-screen>`.

Since 2.2

`gtk-window-is-active` (*self* `<gtk-window>`) ⇒ (*ret* `bool`) [Function]
`is-active` [Method]

Returns whether the window is part of the current active toplevel. (That is, the toplevel window receiving keystrokes.) The return value is '`#t`' if the window is active toplevel itself, but also if it is, say, a `<gtk-plug>` embedded in the active toplevel. You might use this function if you wanted to draw a widget differently in an active window from a widget in an inactive window. See `gtk-window-has-toplevel-focus`

window a `<gtk-window>`

ret '`#t`' if the window part of the current active window.

Since 2.4

`gtk-window-has-toplevel-focus` (*self* `<gtk-window>`) ⇒ (*ret* `bool`) [Function]
`has-toplevel-focus` [Method]

Returns whether the input focus is within this GtkWindow. For real toplevel windows, this is identical to `gtk-window-is-active`, but for embedded windows, like `<gtk-plug>`, the results will differ.

window a `<gtk-window>`

ret '`#t`' if the input focus is within this GtkWindow

Since 2.4

`gtk-window-list-toplevels` ⇒ (*ret* `glist-of`) [Function]

Returns a list of all existing toplevel windows. The widgets in the list are not individually referenced. If you want to iterate through the list and perform actions involving callbacks that might destroy the widgets, you *must* call '`g_list_foreach (result, (GFunc)g_object_ref, NULL)`' first, and then unref all the widgets afterwards.

ret list of toplevel widgets

`gtk-window-add-mnemonic` (*self* `<gtk-window>`) [Function]
(*keyval* `unsigned-int`) (*target* `<gtk-widget>`)
`add-mnemonic` [Method]

Adds a mnemonic to this window.

window a `<gtk-window>`

keyval the mnemonic

target the widget that gets activated by the mnemonic

`gtk-window-remove-mnemonic` (*self* `<gtk-window>`) [Function]
(*keyval* `unsigned-int`) (*target* `<gtk-widget>`)
`remove-mnemonic` [Method]

Removes a mnemonic from this window.

window a `<gtk-window>`

keyval the mnemonic

target the widget that gets activated by the mnemonic

`gtk-window-mnemonic-activate` (*self* `<gtk-window>`) [Function]
(*keyval* `unsigned-int`) (*modifier* `<gdk-modifier-type>`) ⇒ (*ret* `bool`)
`mnemonic-activate` [Method]

Activates the targets associated with the mnemonic.

window a `<gtk-window>`

keyval the mnemonic

modifier the modifiers

ret '`#t`' if the activation is done.

`gtk-window-activate-key` (*self* `<gtk-window>`) [Function]
(*event* `<gdk-event-key>`) ⇒ (*ret* `bool`)
`activate-key` [Method]

Activates mnemonics and accelerators for this `<gtk-window>`. This is normally called by the default ::key_press_event handler for toplevel windows, however in some cases it may be useful to call this directly when overriding the standard key handling for a toplevel window.

window a `<gtk-window>`

event a `<gdk-event-key>`

ret '`#t`' if a mnemonic or accelerator was found and activated.

`gtk-window-propagate-key-event` (*self* `<gtk-window>`) [Function]
(*event* `<gdk-event-key>`) ⇒ (*ret* `bool`)
`propagate-key-event` [Method]

Propagate a key press or release event to the focus widget and up the focus container chain until a widget handles *event*. This is normally called by the default ::key_press_event and ::key_release_event handlers for toplevel windows, however in some cases it may be useful to call this directly when overriding the standard key handling for a toplevel window.

window a `<gtk-window>`

event a `<gdk-event-key>`

ret '`#t`' if a widget in the focus chain handled the event.

`gtk-window-get-focus` (*self* `<gtk-window>`) ⇒ (*ret* `<gtk-widget>`) [Function]
`get-focus` [Method]

Retrieves the current focused widget within the window. Note that this is the widget that would have the focus if the toplevel window focused; if the toplevel window is not focused then '`GTK_WIDGET_HAS_FOCUS (widget)`' will not be '`#t`' for the widget.

window a `<gtk-window>`

ret the currently focused widget, or '`#f`' if there is none.

`gtk-window-set-focus` (*self* `<gtk-window>`) (*focus* `<gtk-widget>`) [Function]
`set-focus` [Method]

If *focus* is not the current focus widget, and is focusable, sets it as the focus widget for the window. If *focus* is '`#f`', unsets the focus widget for this window. To set the focus to a particular widget in the toplevel, it is usually more convenient to use `gtk-widget-grab-focus` instead of this function.

window a `<gtk-window>`

focus widget to be the new focus widget, or '`#f`' to unset any focus widget for the toplevel window.

`gtk-window-set-default` (*self* `<gtk-window>`) [Function]
(*default_widget* `<gtk-widget>`)
`set-default` [Method]

The default widget is the widget that's activated when the user presses Enter in a dialog (for example). This function sets or unsets the default widget for a `<gtk-window>` about. When setting (rather than unsetting) the default widget it's generally easier to call `gtk-widget-grab-focus` on the widget. Before making a widget the default widget, you must set the `<gtk-can-default>` flag on the widget you'd like to make the default using `gtk-widget-set-flags`.

window a `<gtk-window>`

default-widget
widget to be the default, or '`#f`' to unset the default widget for the toplevel.

`gtk-window-present` (*self* `<gtk-window>`) [Function]
`present` [Method]

Presents a window to the user. This may mean raising the window in the stacking order, deiconifying it, moving it to the current desktop, and/or giving it the keyboard focus, possibly dependent on the user's platform, window manager, and preferences.

If *window* is hidden, this function calls `gtk-widget-show` as well.

This function should be used when the user tries to open a window that's already open. Say for example the preferences dialog is currently open, and the user chooses Preferences from the menu a second time; use `gtk-window-present` to move the already-open dialog where the user can see it.

If you are calling this function in response to a user interaction, it is preferable to use `gtk-window-present-with-time`.

window a `<gtk-window>`

`gtk-window-present-with-time` (*self* `<gtk-window>`) [Function]
(*timestamp* `unsigned-int32`)
`present-with-time` [Method]

Presents a window to the user in response to a user interaction. If you need to present a window without a timestamp, use `gtk-window-present`. See `gtk-window-present` for details.

window a `<gtk-window>`

timestamp
the timestamp of the user interaction (typically a button or key press event) which triggered this call

Since 2.8

`gtk-window-iconify` (*self* `<gtk-window>`) [Function]
`iconify` [Method]

Asks to iconify (i.e. minimize) the specified *window*. Note that you shouldn't assume the window is definitely iconified afterward, because other entities (e.g. the user or window manager) could deiconify it again, or there may not be a window manager in which case iconification isn't possible, etc. But normally the window will end up iconified. Just don't write code that crashes if not.

It's permitted to call this function before showing a window, in which case the window will be iconified before it ever appears onscreen.

You can track iconification via the "window_state_event" signal on `<gtk-widget>`.

window a `<gtk-window>`

`gtk-window-deiconify` (*self* `<gtk-window>`) [Function]
`deiconify` [Method]

Asks to deiconify (i.e. unminimize) the specified *window*. Note that you shouldn't assume the window is definitely deiconified afterward, because other entities (e.g. the user or window manager) could iconify it again before your code which assumes deiconification gets to run.

You can track iconification via the "window_state_event" signal on `<gtk-widget>`.

window a `<gtk-window>`

`gtk-window-stick` (*self* `<gtk-window>`) [Function]
`stick` [Method]

Asks to stick *window*, which means that it will appear on all user desktops. Note that you shouldn't assume the window is definitely stuck afterward, because other entities (e.g. the user or window manager) could unstick it again, and some window managers do not support sticking windows. But normally the window will end up stuck. Just don't write code that crashes if not.

It's permitted to call this function before showing a window.

You can track stickiness via the "window_state_event" signal on `<gtk-widget>`.

window a `<gtk-window>`

`gtk-window-unstick` (*self* `<gtk-window>`) [Function]
`unstick` [Method]

Asks to unstick *window*, which means that it will appear on only one of the user's desktops. Note that you shouldn't assume the window is definitely unstuck afterward, because other entities (e.g. the user or window manager) could stick it again. But normally the window will end up stuck. Just don't write code that crashes if not.

You can track stickiness via the "window_state_event" signal on `<gtk-widget>`.

window a `<gtk-window>`

`gtk-window-maximize` (*self* `<gtk-window>`) [Function]
`maximize` [Method]

Asks to maximize *window*, so that it becomes full-screen. Note that you shouldn't assume the window is definitely maximized afterward, because other entities (e.g. the user or window manager) could unmaximize it again, and not all window managers support maximization. But normally the window will end up maximized. Just don't write code that crashes if not.

It's permitted to call this function before showing a window, in which case the window will be maximized when it appears onscreen initially.

You can track maximization via the "window_state_event" signal on `<gtk-widget>`.

window a `<gtk-window>`

`gtk-window-unmaximize` (*self* `<gtk-window>`) [Function]
`unmaximize` [Method]

Asks to unmaximize *window*. Note that you shouldn't assume the window is definitely unmaximized afterward, because other entities (e.g. the user or window manager) could maximize it again, and not all window managers honor requests to unmaximize. But normally the window will end up unmaximized. Just don't write code that crashes if not.

You can track maximization via the "window_state_event" signal on `<gtk-widget>`.

window a `<gtk-window>`

`gtk-window-fullscreen` (*self* `<gtk-window>`) [Function]
`fullscreen` [Method]

Asks to place *window* in the fullscreen state. Note that you shouldn't assume the window is definitely full screen afterward, because other entities (e.g. the user or window manager) could unfullscreen it again, and not all window managers honor requests to fullscreen windows. But normally the window will end up fullscreen. Just don't write code that crashes if not.

You can track the fullscreen state via the "window_state_event" signal on `<gtk-widget>`.

window a `<gtk-window>`

Since 2.2

`gtk-window-unfullscreen` (*self* `<gtk-window>`) [Function]
`unfullscreen` [Method]

Asks to toggle off the fullscreen state for *window*. Note that you shouldn't assume the window is definitely not full screen afterward, because other entities (e.g. the user or window manager) could fullscreen it again, and not all window managers honor requests to unfullscreen windows. But normally the window will end up restored to its normal state. Just don't write code that crashes if not.

You can track the fullscreen state via the `"window_state_event"` signal on `<gtk-widget>`.

window a `<gtk-window>`

Since 2.2

`gtk-window-set-keep-above` (*self* `<gtk-window>`) (*setting* `bool`) [Function]
`set-keep-above` [Method]

Asks to keep *window* above, so that it stays on top. Note that you shouldn't assume the window is definitely above afterward, because other entities (e.g. the user or window manager) could not keep it above, and not all window managers support keeping windows above. But normally the window will end kept above. Just don't write code that crashes if not.

It's permitted to call this function before showing a window, in which case the window will be kept above when it appears onscreen initially.

You can track the above state via the `"window_state_event"` signal on `<gtk-widget>`.

Note that, according to the Extended Window Manager Hints specification, the above state is mainly meant for user preferences and should not be used by applications e.g. for drawing attention to their dialogs.

window a `<gtk-window>`

setting whether to keep *window* above other windows

Since 2.4

`gtk-window-set-keep-below` (*self* `<gtk-window>`) (*setting* `bool`) [Function]
`set-keep-below` [Method]

Asks to keep *window* below, so that it stays in bottom. Note that you shouldn't assume the window is definitely below afterward, because other entities (e.g. the user or window manager) could not keep it below, and not all window managers support putting windows below. But normally the window will be kept below. Just don't write code that crashes if not.

It's permitted to call this function before showing a window, in which case the window will be kept below when it appears onscreen initially.

You can track the below state via the `"window_state_event"` signal on `<gtk-widget>`.

Note that, according to the Extended Window Manager Hints specification, the above state is mainly meant for user preferences and should not be used by applications e.g. for drawing attention to their dialogs.

window a `<gtk-window>`

setting whether to keep *window* below other windows

Since 2.4

`gtk-window-begin-resize-drag` (*self* `<gtk-window>`) (*edge* `<gdk-window-edge>`) (*button* `int`) (*root_x* `int`) (*root_y* `int`) (*timestamp* `unsigned-int32`) [Function]

`begin-resize-drag` [Method]

Starts resizing a window. This function is used if an application has window resizing controls. When GDK can support it, the resize will be done using the standard mechanism for the window manager or windowing system. Otherwise, GDK will try to emulate window resizing, potentially not all that well, depending on the windowing system.

window a `<gtk-window>`

edge position of the resize control

button mouse button that initiated the drag

root-x X position where the user clicked to initiate the drag, in root window coordinates

root-y Y position where the user clicked to initiate the drag

timestamp timestamp from the click event that initiated the drag

`gtk-window-begin-move-drag` (*self* `<gtk-window>`) (*button* `int`) (*root_x* `int`) (*root_y* `int`) (*timestamp* `unsigned-int32`) [Function]

`begin-move-drag` [Method]

Starts moving a window. This function is used if an application has window movement grips. When GDK can support it, the window movement will be done using the standard mechanism for the window manager or windowing system. Otherwise, GDK will try to emulate window movement, potentially not all that well, depending on the windowing system.

window a `<gtk-window>`

button mouse button that initiated the drag

root-x X position where the user clicked to initiate the drag, in root window coordinates

root-y Y position where the user clicked to initiate the drag

timestamp timestamp from the click event that initiated the drag

`gtk-window-set-decorated` (*self* `<gtk-window>`) (*setting* `bool`) [Function]

`set-decorated` [Method]

By default, windows are decorated with a title bar, resize controls, etc. Some window managers allow GTK+ to disable these decorations, creating a borderless window. If you set the decorated property to '`#f`' using this function, GTK+ will do its best to

convince the window manager not to decorate the window. Depending on the system, this function may not have any effect when called on a window that is already visible, so you should call it before calling `gtk-window-show`.

On Windows, this function always works, since there's no window manager policy involved.

window a `<gtk-window>`

setting '`#t`' to decorate the window

`gtk-window-set-deletable` (*self* `<gtk-window>`) (*setting* `bool`) [Function]
`set-deletable` [Method]

By default, windows have a close button in the window frame. Some window managers allow GTK+ to disable this button. If you set the deletable property to '`#f`' using this function, GTK+ will do its best to convince the window manager not to show a close button. Depending on the system, this function may not have any effect when called on a window that is already visible, so you should call it before calling `gtk-window-show`.

On Windows, this function always works, since there's no window manager policy involved.

window a `<gtk-window>`

setting '`#t`' to decorate the window as deletable

Since 2.10

`gtk-window-set-frame-dimensions` (*self* `<gtk-window>`) (*left* `int`) [Function]
(*top* `int`) (*right* `int`) (*bottom* `int`)
`set-frame-dimensions` [Method]

(Note: this is a special-purpose function intended for the framebuffer port; see `gtk-window-set-has-frame`. It will have no effect on the window border drawn by the window manager, which is the normal case when using the X Window system.)

For windows with frames (see `gtk-window-set-has-frame`) this function can be used to change the size of the frame border.

window a `<gtk-window>` that has a frame

left The width of the left border

top The height of the top border

right The width of the right border

bottom The height of the bottom border

`gtk-window-set-has-frame` (*self* `<gtk-window>`) (*setting* `bool`) [Function]
`set-has-frame` [Method]

(Note: this is a special-purpose function for the framebuffer port, that causes GTK+ to draw its own window border. For most applications, you want `gtk-window-set-decorated` instead, which tells the window manager whether to draw the window border.)

If this function is called on a window with setting of '`#t`', before it is realized or showed, it will have a "frame" window around *window->window*, accessible in *window->frame*. Using the signal frame_event you can receive all events targeted at the frame.

This function is used by the linux-fb port to implement managed windows, but it could conceivably be used by X-programs that want to do their own window decorations.

window a `<gtk-window>`

setting a boolean

`gtk-window-set-mnemonic-modifier` (*self* `<gtk-window>`) (*modifier* `<gdk-modifier-type>`) [Function]

`set-mnemonic-modifier` [Method]

Sets the mnemonic modifier for this window.

window a `<gtk-window>`

modifier the modifier mask used to activate mnemonics on this window.

`gtk-window-set-role` (*self* `<gtk-window>`) (*role* `mchars`) [Function]

`set-role` [Method]

This function is only useful on X11, not with other GTK+ targets.

In combination with the window title, the window role allows a window manager to identify "the same" window when an application is restarted. So for example you might set the "toolbox" role on your app's toolbox window, so that when the user restarts their session, the window manager can put the toolbox back in the same place.

If a window already has a unique title, you don't need to set the role, since the WM can use the title to identify the window when restoring the session.

window a `<gtk-window>`

role unique identifier for the window to be used when restoring a session

`gtk-window-set-type-hint` (*self* `<gtk-window>`) (*hint* `<gdk-window-type-hint>`) [Function]

`set-type-hint` [Method]

By setting the type hint for the window, you allow the window manager to decorate and handle the window in a way which is suitable to the function of the window in your application.

This function should be called before the window becomes visible.

`gtk-dialog-new-with-buttons` and other convenience functions in GTK+ will sometimes call `gtk-window-set-type-hint` on your behalf.

window a `<gtk-window>`

hint the window type

`gtk-window-set-skip-taskbar-hint` (*self* `<gtk-window>`) (*setting* `bool`) [Function]

`set-skip-taskbar-hint` [Method]

Windows may set a hint asking the desktop environment not to display the window in the task bar. This function sets this hint.

window a `<gtk-window>`

setting '`#t`' to keep this window from appearing in the task bar

Since 2.2

`gtk-window-set-skip-pager-hint` (*self* `<gtk-window>`) (*setting* `bool`) [Function]
`set-skip-pager-hint` [Method]

Windows may set a hint asking the desktop environment not to display the window in the pager. This function sets this hint. (A "pager" is any desktop navigation tool such as a workspace switcher that displays a thumbnail representation of the windows on the screen.)

window a `<gtk-window>`

setting '`#t`' to keep this window from appearing in the pager

Since 2.2

`gtk-window-set-urgency-hint` (*self* `<gtk-window>`) (*setting* `bool`) [Function]
`set-urgency-hint` [Method]

Windows may set a hint asking the desktop environment to draw the users attention to the window. This function sets this hint.

window a `<gtk-window>`

setting '`#t`' to mark this window as urgent

Since 2.8

`gtk-window-set-accept-focus` (*self* `<gtk-window>`) (*setting* `bool`) [Function]
`set-accept-focus` [Method]

Windows may set a hint asking the desktop environment not to receive the input focus. This function sets this hint.

window a `<gtk-window>`

setting '`#t`' to let this window receive input focus

Since 2.4

`gtk-window-set-focus-on-map` (*self* `<gtk-window>`) (*setting* `bool`) [Function]
`set-focus-on-map` [Method]

Windows may set a hint asking the desktop environment not to receive the input focus when the window is mapped. This function sets this hint.

window a `<gtk-window>`

setting '`#t`' to let this window receive input focus on map

Since 2.6

`gtk-window-get-decorated` (*self* `<gtk-window>`) ⇒ (*ret* `bool`) [Function]
`get-decorated` [Method]

Returns whether the window has been set to have decorations such as a title bar via `gtk-window-set-decorated`.

window a `<gtk-window>`

ret '`#t`' if the window has been set to have decorations

`gtk-window-get-deletable` (*self* `<gtk-window>`) ⇒ (*ret* `bool`) [Function]
`get-deletable` [Method]
Returns whether the window has been set to have a close button via `gtk-window-set-deletable`.

window a `<gtk-window>`

ret '`#t`' if the window has been set to have a close button

Since 2.10

`gtk-window-get-default-icon-list` ⇒ (*ret* `glist-of`) [Function]
Gets the value set by `gtk-window-set-default-icon-list`. The list is a copy and should be freed with `g-list-free`, but the pixbufs in the list have not had their reference count incremented.

ret copy of default icon list

`gtk-window-get-default-size` (*self* `<gtk-window>`) ⇒ (*width* `int`) (*height* `int`) [Function]
`get-default-size` [Method]
Gets the default size of the window. A value of -1 for the width or height indicates that a default size has not been explicitly set for that dimension, so the "natural" size of the window will be used.

window a `<gtk-window>`

width location to store the default width, or '`#f`'

height location to store the default height, or '`#f`'

`gtk-window-get-destroy-with-parent` (*self* `<gtk-window>`) ⇒ (*ret* `bool`) [Function]
`get-destroy-with-parent` [Method]
Returns whether the window will be destroyed with its transient parent. See `gtk-window-set-destroy-with-parent`.

window a `<gtk-window>`

ret '`#t`' if the window will be destroyed with its transient parent.

`gtk-window-get-frame-dimensions` (*self* `<gtk-window>`) ⇒ (*left* `int`) (*top* `int`) (*right* `int`) (*bottom* `int`) [Function]
`get-frame-dimensions` [Method]
(Note: this is a special-purpose function intended for the framebuffer port; see `gtk-window-set-has-frame`. It will not return the size of the window border drawn by the window manager, which is the normal case when using a windowing system. See `gdk-window-get-frame-extents` to get the standard window border extents.)

Retrieves the dimensions of the frame window for this toplevel. See `gtk-window-set-has-frame`, `gtk-window-set-frame-dimensions`.

window a `<gtk-window>`

left location to store the width of the frame at the left, or '`#f`'

top location to store the height of the frame at the top, or '`#f`'

right location to store the width of the frame at the returns, or '`#f`'

bottom location to store the height of the frame at the bottom, or '`#f`'

`gtk-window-get-has-frame` (*self* `<gtk-window>`) ⇒ (*ret* `bool`) [Function]
`get-has-frame` [Method]

Accessor for whether the window has a frame window exterior to *window*->*window*. Gets the value set by `gtk-window-set-has-frame`.

window a `<gtk-window>`

ret '`#t`' if a frame has been added to the window via `gtk-window-set-has-frame`.

`gtk-window-get-icon` (*self* `<gtk-window>`) ⇒ (*ret* `<gdk-pixbuf>`) [Function]
`get-icon` [Method]

Gets the value set by `gtk-window-set-icon` (or if you've called `gtk-window-set-icon-list`, gets the first icon in the icon list).

window a `<gtk-window>`

ret icon for window

`gtk-window-get-icon-list` (*self* `<gtk-window>`) ⇒ (*ret* `glist-of`) [Function]
`get-icon-list` [Method]

Retrieves the list of icons set by `gtk-window-set-icon-list`. The list is copied, but the reference count on each member won't be incremented.

window a `<gtk-window>`

ret copy of window's icon list

`gtk-window-get-icon-name` (*self* `<gtk-window>`) ⇒ (*ret* `mchars`) [Function]
`get-icon-name` [Method]

Returns the name of the themed icon for the window, see `gtk-window-set-icon-name`.

window a `<gtk-window>`

ret the icon name or '`#f`' if the window has no themed icon

Since 2.6

`gtk-window-get-mnemonic-modifier` (*self* `<gtk-window>`) [Function]
⇒ (*ret* `<gdk-modifier-type>`)
`get-mnemonic-modifier` [Method]

Returns the mnemonic modifier for this window. See `gtk-window-set-mnemonic-modifier`.

window a `<gtk-window>`

ret the modifier mask used to activate mnemonics on this window.

`gtk-window-get-modal` (*self* `<gtk-window>`) ⇒ (*ret* `bool`) [Function]
`get-modal` [Method]

Returns whether the window is modal. See `gtk-window-set-modal`.

window a `<gtk-window>`

ret '`#t`' if the window is set to be modal and establishes a grab when shown

`gtk-window-get-position` (*self* `<gtk-window>`) ⇒ (*root_x* `int`) (*root_y* `int`) [Function]
`get-position` [Method]

This function returns the position you need to pass to `gtk-window-move` to keep *window* in its current position. This means that the meaning of the returned value varies with window gravity. See `gtk-window-move` for more details.

If you haven't changed the window gravity, its gravity will be `<gdk-gravity-north-west>`. This means that `gtk-window-get-position` gets the position of the top-left corner of the window manager frame for the window. `gtk-window-move` sets the position of this same top-left corner.

`gtk-window-get-position` is not 100% reliable because the X Window System does not specify a way to obtain the geometry of the decorations placed on a window by the window manager. Thus GTK+ is using a "best guess" that works with most window managers.

Moreover, nearly all window managers are historically broken with respect to their handling of window gravity. So moving a window to its current position as returned by `gtk-window-get-position` tends to result in moving the window slightly. Window managers are slowly getting better over time.

If a window has gravity `<gdk-gravity-static>` the window manager frame is not relevant, and thus `gtk-window-get-position` will always produce accurate results. However you can't use static gravity to do things like place a window in a corner of the screen, because static gravity ignores the window manager decorations.

If you are saving and restoring your application's window positions, you should know that it's impossible for applications to do this without getting it somewhat wrong because applications do not have sufficient knowledge of window manager state. The Correct Mechanism is to support the session management protocol (see the "GnomeClient" object in the GNOME libraries for example) and allow the window manager to save your window sizes and positions.

window a `<gtk-window>`

root-x return location for X coordinate of gravity-determined reference p\oint

root-y return location for Y coordinate of gravity-determined reference p\oint

`gtk-window-get-role` (*self* `<gtk-window>`) ⇒ (*ret* `mchars`) [Function]
`get-role` [Method]

Returns the role of the window. See `gtk-window-set-role` for further explanation.

window a `<gtk-window>`

ret the role of the window if set, or '`#f`'. The returned is owned by the widget and must not be modified or freed.

`gtk-window-get-size` (*self* `<gtk-window>`) ⇒ (*width* `int`) (*height* `int`) [Function]

`get-size` [Method]

Obtains the current size of *window*. If *window* is not onscreen, it returns the size GTK+ will suggest to the window manager for the initial window size (but this is not reliably the same as the size the window manager will actually select). The size obtained by `gtk-window-get-size` is the last size received in a `<gdk-event-configure>`, that is, GTK+ uses its locally-stored size, rather than querying the X server for the size. As a result, if you call `gtk-window-resize` then immediately call `gtk-window-get-size`, the size won't have taken effect yet. After the window manager processes the resize request, GTK+ receives notification that the size has changed via a configure event, and the size of the window gets updated.

Note 1: Nearly any use of this function creates a race condition, because the size of the window may change between the time that you get the size and the time that you perform some action assuming that size is the current size. To avoid race conditions, connect to `"configure_event"` on the window and adjust your size-dependent state to match the size delivered in the `<gdk-event-configure>`.

Note 2: The returned size does *not* include the size of the window manager decorations (aka the window frame or border). Those are not drawn by GTK+ and GTK+ has no reliable method of determining their size.

Note 3: If you are getting a window size in order to position the window onscreen, there may be a better way. The preferred way is to simply set the window's semantic type with `gtk-window-set-type-hint`, which allows the window manager to e.g. center dialogs. Also, if you set the transient parent of dialogs with `gtk-window-set-transient-for` window managers will often center the dialog over its parent window. It's much preferred to let the window manager handle these things rather than doing it yourself, because all apps will behave consistently and according to user prefs if the window manager handles it. Also, the window manager can take the size of the window decorations/border into account, while your application cannot.

In any case, if you insist on application-specified window positioning, there's *still* a better way than doing it yourself - `gtk-window-set-position` will frequently handle the details for you.

window a `<gtk-window>`

width return location for width, or '`#f`'

height return location for height, or '`#f`'

`gtk-window-get-title` (*self* `<gtk-window>`) ⇒ (*ret* `mchars`) [Function]

`get-title` [Method]

Retrieves the title of the window. See `gtk-window-set-title`.

window a `<gtk-window>`

ret the title of the window, or '`#f`' if none has been set explicitely. The returned string is owned by the widget and must not be modified or freed.

`gtk-window-get-transient-for` (*self* `<gtk-window>`) [Function]
⇒ (*ret* `<gtk-window>`)
`get-transient-for` [Method]

Fetches the transient parent for this window. See `gtk-window-set-transient-for`.

window a `<gtk-window>`

ret the transient parent for this window, or '`#f`' if no transient parent has been set.

`gtk-window-get-type-hint` (*self* `<gtk-window>`) [Function]
⇒ (*ret* `<gdk-window-type-hint>`)
`get-type-hint` [Method]

Gets the type hint for this window. See `gtk-window-set-type-hint`.

window a `<gtk-window>`

ret the type hint for *window*.

`gtk-window-get-skip-taskbar-hint` (*self* `<gtk-window>`) [Function]
⇒ (*ret* `bool`)
`get-skip-taskbar-hint` [Method]

Gets the value set by `gtk-window-set-skip-taskbar-hint`

window a `<gtk-window>`

ret '`#t`' if window shouldn't be in taskbar

Since 2.2

`gtk-window-get-skip-pager-hint` (*self* `<gtk-window>`) [Function]
⇒ (*ret* `bool`)
`get-skip-pager-hint` [Method]

Gets the value set by `gtk-window-set-skip-pager-hint`.

window a `<gtk-window>`

ret '`#t`' if window shouldn't be in pager

Since 2.2

`gtk-window-get-urgency-hint` (*self* `<gtk-window>`) ⇒ (*ret* `bool`) [Function]
`get-urgency-hint` [Method]

Gets the value set by `gtk-window-set-urgency-hint`

window a `<gtk-window>`

ret '`#t`' if window is urgent

Since 2.8

`gtk-window-get-accept-focus` (*self* `<gtk-window>`) ⇒ (*ret* `bool`) [Function]
`get-accept-focus` [Method]

Gets the value set by `gtk-window-set-accept-focus`.

window a `<gtk-window>`

ret '`#t`' if window should receive the input focus

Since 2.4

`gtk-window-get-focus-on-map` (*self* `<gtk-window>`) ⇒ (*ret* `bool`) [Function]
`get-focus-on-map` [Method]

Gets the value set by `gtk-window-set-focus-on-map`.

window a `<gtk-window>`

ret '`#t`' if window should receive the input focus when mapped.

Since 2.6

`gtk-window-get-group` (*self* `<gtk-window>`) [Function]
⇒ (*ret* `<gtk-window-group>`)
`get-group` [Method]

Returns the group for *window* or the default group, if *window* is '`#f`' or if *window* does not have an explicit window group.

window a `<gtk-window>`, or '`#f`'

ret the `<gtk-window-group>` for a window or the default group

Since 2.10

`gtk-window-move` (*self* `<gtk-window>`) (*x* `int`) (*y* `int`) [Function]
`move` [Method]

Asks the window manager to move *window* to the given position. Window managers are free to ignore this; most window managers ignore requests for initial window positions (instead using a user-defined placement algorithm) and honor requests after the window has already been shown.

Note: the position is the position of the gravity-determined reference point for the window. The gravity determines two things: first, the location of the reference point in root window coordinates; and second, which point on the window is positioned at the reference point.

By default the gravity is `<gdk-gravity-north-west>`, so the reference point is simply the *x*, *y* supplied to `gtk-window-move`. The top-left corner of the window decorations (aka window frame or border) will be placed at *x*, *y*. Therefore, to position a window at the top left of the screen, you want to use the default gravity (which is `<gdk-gravity-north-west>`) and move the window to 0,0.

To position a window at the bottom right corner of the screen, you would set `<gdk-gravity-south-east>`, which means that the reference point is at *x* `+` the window width and *y* `+` the window height, and the bottom-right corner of the window border will be placed at that reference point. So, to place a window in the bottom right corner you would first set gravity to south east, then write: '`gtk_window_move (window, gdk-screen-width - window_width, gdk-screen-height - window_height)`' (note that this example does not take multi-head scenarios into account).

The Extended Window Manager Hints specification at http://www.freedesktop.org/Standards/wm-spec has a nice table of gravities in the "implementation notes" section.

The `gtk-window-get-position` documentation may also be relevant.

window a `<gtk-window>`

x X coordinate to move window to

y Y coordinate to move window to

`gtk-window-parse-geometry` (*self* `<gtk-window>`) (*geometry* `mchars`) [Function]
⇒ (*ret* `bool`)

`parse-geometry` [Method]

Parses a standard X Window System geometry string - see the manual page for X (type 'man X') for details on this. `gtk-window-parse-geometry` does work on all GTK+ ports including Win32 but is primarily intended for an X environment.

If either a size or a position can be extracted from the geometry string, `gtk-window-parse-geometry` returns '`#t`' and calls `gtk-window-set-default-size` and/or `gtk-window-move` to resize/move the window.

If `gtk-window-parse-geometry` returns '`#t`', it will also set the `<gdk-hint-user-pos>` and/or `<gdk-hint-user-size>` hints indicating to the window manager that the size/position of the window was user-specified. This causes most window managers to honor the geometry.

Note that for `gtk-window-parse-geometry` to work as expected, it has to be called when the window has its "final" size, i.e. after calling `gtk-widget-show-all` on the contents and `gtk-window-set-geometry-hints` on the window.

```
#include <gtk/gtk.h>

static void
fill_with_content (GtkWidget *vbox)
{
  /* fill with content... */
}

int
main (int argc, char *argv[])
{
  GtkWidget *window, *vbox;
  GdkGeometry size_hints = {
    100, 50, 0, 0, 100, 50, 10, 10, 0.0, 0.0, GDK_GRAVITY_NORTH_WEST
  };

  gtk_init (&argc, &argv);

  window = gtk_window_new (GTK_WINDOW_TOPLEVEL);
  vbox = gtk_vbox_new (FALSE, 0);

  gtk_container_add (GTK_CONTAINER (window), vbox);
  fill_with_content (vbox);
  gtk_widget_show_all (vbox);

  gtk_window_set_geometry_hints (GTK_WINDOW (window),
```

```
            window,
          &size_hints,
          GDK_HINT_MIN_SIZE |
          GDK_HINT_BASE_SIZE |
          GDK_HINT_RESIZE_INC);

        if (argc > 1)
          {
            if (!gtk_window_parse_geometry (GTK_WINDOW (window), argv[1]))
              fprintf (stderr, "Failed to parse '%s'\n", argv[1]);
          }

        gtk_widget_show_all (window);
        gtk_main ();

        return 0;
      }
```

window a `<gtk-window>`

geometry geometry string

ret '`#t`' if string was parsed successfully

`gtk-window-reshow-with-initial-size` (*self* `<gtk-window>`) [Function]
`reshow-with-initial-size` [Method]

Hides *window*, then reshows it, resetting the default size and position of the window. Used by GUI builders only.

window a `<gtk-window>`

`gtk-window-resize` (*self* `<gtk-window>`) (*width* `int`) (*height* `int`) [Function]
`resize` [Method]

Resizes the window as if the user had done so, obeying geometry constraints. The default geometry constraint is that windows may not be smaller than their size request; to override this constraint, call `gtk-widget-set-size-request` to set the window's request to a smaller value.

If `gtk-window-resize` is called before showing a window for the first time, it overrides any default size set with `gtk-window-set-default-size`.

Windows may not be resized smaller than 1 by 1 pixels.

window a `<gtk-window>`

width width in pixels to resize the window to

height height in pixels to resize the window to

`gtk-window-set-default-icon-list` (*list* `glist-of`) [Function]

Sets an icon list to be used as fallback for windows that haven't had `gtk-window-set-icon-list` called on them to set up a window-specific icon list. This function allows you to set up the icon for all windows in your app at once.

See `gtk-window-set-icon-list` for more details.

list a list of `<gdk-pixbuf>`

`gtk-window-set-default-icon` (*icon* `<gdk-pixbuf>`) [Function]
Sets an icon to be used as fallback for windows that haven't had `gtk-window-set-icon` called on them from a pixbuf.

icon the icon

Since 2.4

`gtk-window-set-default-icon-name` (*name* `mchars`) [Function]
Sets an icon to be used as fallback for windows that haven't had `gtk-window-set-icon-list` called on them from a named themed icon, see `gtk-window-set-icon-name`.

name the name of the themed icon

Since 2.6

`gtk-window-set-icon` (*self* `<gtk-window>`) (*icon* `<gdk-pixbuf>`) [Function]
`set-icon` [Method]
Sets up the icon representing a `<gtk-window>`. This icon is used when the window is minimized (also known as iconified). Some window managers or desktop environments may also place it in the window frame, or display it in other contexts.

The icon should be provided in whatever size it was naturally drawn; that is, don't scale the image before passing it to GTK+. Scaling is postponed until the last minute, when the desired final size is known, to allow best quality.

If you have your icon hand-drawn in multiple sizes, use `gtk-window-set-icon-list`. Then the best size will be used.

This function is equivalent to calling `gtk-window-set-icon-list` with a 1-element list.

See also `gtk-window-set-default-icon-list` to set the icon for all windows in your application in one go.

window a `<gtk-window>`

icon icon image, or '`#f`'

`gtk-window-set-icon-list` (*self* `<gtk-window>`) (*list* `glist-of`) [Function]
`set-icon-list` [Method]
Sets up the icon representing a `<gtk-window>`. The icon is used when the window is minimized (also known as iconified). Some window managers or desktop environments may also place it in the window frame, or display it in other contexts.

`gtk-window-set-icon-list` allows you to pass in the same icon in several hand-drawn sizes. The list should contain the natural sizes your icon is available in; that is, don't scale the image before passing it to GTK+. Scaling is postponed until the last minute, when the desired final size is known, to allow best quality.

By passing several sizes, you may improve the final image quality of the icon, by reducing or eliminating automatic image scaling.

Recommended sizes to provide: 16x16, 32x32, 48x48 at minimum, and larger images (64x64, 128x128) if you have them.

See also `gtk-window-set-default-icon-list` to set the icon for all windows in your application in one go.

Note that transient windows (those who have been set transient for another window using `gtk-window-set-transient-for`) will inherit their icon from their transient parent. So there's no need to explicitly set the icon on transient windows.

window a `<gtk-window>`

list list of `<gdk-pixbuf>`

`gtk-window-set-icon-from-file` (*self* `<gtk-window>`) [Function]
(*filename* `mchars`) ⇒ (*ret* `bool`)

`set-icon-from-file` [Method]

Sets the icon for *window*. Warns on failure if *err* is '`#f`'.

This function is equivalent to calling `gtk-window-set-icon` with a pixbuf created by loading the image from *filename*.

window a `<gtk-window>`

filename location of icon file

err location to store error, or '`#f`'.

ret '`#t`' if setting the icon succeeded.

Since 2.2

`gtk-window-set-icon-name` (*self* `<gtk-window>`) (*name* `mchars`) [Function]

`set-icon-name` [Method]

Sets the icon for the window from a named themed icon. See the docs for `<gtk-icon-theme>` for more details.

Note that this has nothing to do with the WM_ICON_NAME property which is mentioned in the ICCCM.

window a `<gtk-window>`

name the name of the themed icon

Since 2.6

6 GtkWindowGroup

Limit the effect of grabs

6.1 Overview

6.2 Usage

`<gtk-window-group>` [Class]
This `<gobject>` class defines no properties, other than those defined by its superclasses.

`gtk-window-group-new` ⇒ (*ret* `<gtk-window-group>`) [Function]
Creates a new `<gtk-window-group>` object. Grabs added with `gtk-grab-add` only affect windows within the same `<gtk-window-group>`.

ret a new `<gtk-window-group>`.

`gtk-window-group-add-window` (*self* `<gtk-window-group>`) [Function]
(*window* `<gtk-window>`)
`add-window` [Method]
Adds a window to a `<gtk-window-group>`.

window-group
a `<gtk-window-group>`

window the `<gtk-window>` to add

`gtk-window-group-remove-window` (*self* `<gtk-window-group>`) [Function]
(*window* `<gtk-window>`)
`remove-window` [Method]
Removes a window from a `<gtk-window-group>`.

window-group
a `<gtk-window-group>`

window the `<gtk-window>` to remove

7 GtkAboutDialog

Display information about an application

7.1 Overview

The <gtk-about-dialog> offers a simple way to display information about a program like its logo, name, copyright, website and license. It is also possible to give credits to the authors, documenters, translators and artists who have worked on the program. An about dialog is typically opened when the user selects the 'About' option from the 'Help' menu. All parts of the dialog are optional.

About dialog often contain links and email addresses. <gtk-about-dialog> supports this by offering global hooks, which are called when the user clicks on a link or email address, see gtk-about-dialog-set-email-hook and gtk-about-dialog-set-url-hook. Email addresses in the authors, documenters and artists properties are recognized by looking for '<user@host>', URLs are recognized by looking for 'http://url', with 'url' extending to the next space, tab or line break.

To make constructing a <gtk-about-dialog> as convenient as possible, you can use the function gtk-show-about-dialog which constructs and shows a dialog and keeps it around so that it can be shown again.

7.2 Usage

<gtk-about-dialog> [Class]

This <gobject> class defines the following properties:

program-name
: The name of the program. If this is not set, it defaults to g_get_application_name()

version
: The version of the program

copyright
: Copyright information for the program

comments
: Comments about the program

website
: The URL for the link to the website of the program

website-label
: The label for the link to the website of the program. If this is not set, it defaults to the URL

license
: The license of the program

authors
: List of authors of the program

documenters
: List of people documenting the program

translator-credits
: Credits to the translators. This string should be marked as translatable

`artists` List of people who have contributed artwork to the program

`logo` A logo for the about box. If this is not set, it defaults to gtk_window_get_default_icon_list()

`logo-icon-name`
A named icon to use as the logo for the about box.

`wrap-license`
Whether to wrap the license text.

`gtk-about-dialog-new` ⇒ (*ret* `<gtk-widget>`) [Function]
Creates a new `<gtk-about-dialog>`.

ret a newly created `<gtk-about-dialog>`

Since 2.6

`gtk-about-dialog-get-name` (*self* `<gtk-about-dialog>`) [Function]
⇒ (*ret* `mchars`)
`get-name` [Method]
Returns the program name displayed in the about dialog.

about a `<gtk-about-dialog>`

ret The program name. The string is owned by the about dialog and must not be modified.

Since 2.6

`gtk-about-dialog-set-name` (*self* `<gtk-about-dialog>`) [Function]
(*name* `mchars`)
`set-name` [Method]
Sets the name to display in the about dialog. If this is not set, it defaults to `g-get-application-name`.

about a `<gtk-about-dialog>`

name the program name

Since 2.6

`gtk-about-dialog-get-version` (*self* `<gtk-about-dialog>`) [Function]
⇒ (*ret* `mchars`)
`get-version` [Method]
Returns the version string.

about a `<gtk-about-dialog>`

ret The version string. The string is owned by the about dialog and must not be modified.

Since 2.6

`gtk-about-dialog-set-version` (*self* `<gtk-about-dialog>`) [Function]
(*version* `mchars`)

`set-version` [Method]

Sets the version string to display in the about dialog.

about a `<gtk-about-dialog>`

version the version string

Since 2.6

`gtk-about-dialog-get-copyright` (*self* `<gtk-about-dialog>`) [Function]
⇒ (*ret* `mchars`)

`get-copyright` [Method]

Returns the copyright string.

about a `<gtk-about-dialog>`

ret The copyright string. The string is owned by the about dialog and must not be modified.

Since 2.6

`gtk-about-dialog-set-copyright` (*self* `<gtk-about-dialog>`) [Function]
(*copyright* `mchars`)

`set-copyright` [Method]

Sets the copyright string to display in the about dialog. This should be a short string of one or two lines.

about a `<gtk-about-dialog>`

copyright the copyright string

Since 2.6

`gtk-about-dialog-get-license` (*self* `<gtk-about-dialog>`) [Function]
⇒ (*ret* `mchars`)

`get-license` [Method]

Returns the license information.

about a `<gtk-about-dialog>`

ret The license information. The string is owned by the about dialog and must not be modified.

Since 2.6

`gtk-about-dialog-set-license` (*self* `<gtk-about-dialog>`) [Function]
(*license* `mchars`)

`set-license` [Method]

Sets the license information to be displayed in the secondary license dialog. If *license* is '`#f`', the license button is hidden.

about a `<gtk-about-dialog>`

license the license information or '`#f`'

Since 2.6

`gtk-about-dialog-get-wrap-license` (*self* `<gtk-about-dialog>`) [Function]
⇒ (*ret* `bool`)
`get-wrap-license` [Method]

Returns whether the license text in *about* is automatically wrapped.

about a `<gtk-about-dialog>`

ret '`#t`' if the license text is wrapped

Since 2.8

`gtk-about-dialog-set-wrap-license` (*self* `<gtk-about-dialog>`) [Function]
(*wrap_license* `bool`)
`set-wrap-license` [Method]

Sets whether the license text in *about* is automatically wrapped.

about a `<gtk-about-dialog>`

wrap-license
whether to wrap the license

Since 2.8

`gtk-about-dialog-get-website` (*self* `<gtk-about-dialog>`) [Function]
⇒ (*ret* `mchars`)
`get-website` [Method]

Returns the website URL.

about a `<gtk-about-dialog>`

ret The website URL. The string is owned by the about dialog and must not be modified.

Since 2.6

`gtk-about-dialog-set-website` (*self* `<gtk-about-dialog>`) [Function]
(*website* `mchars`)
`set-website` [Method]

Sets the URL to use for the website link.

about a `<gtk-about-dialog>`

website a URL string starting with `"http://"`

Since 2.6

`gtk-about-dialog-get-website-label` (*self* `<gtk-about-dialog>`) [Function]
⇒ (*ret* `mchars`)
`get-website-label` [Method]

Returns the label used for the website link.

about a `<gtk-about-dialog>`

ret The label used for the website link. The string is owned by the about dialog and must not be modified.

Since 2.6

`gtk-about-dialog-set-website-label` (*self* `<gtk-about-dialog>`) [Function]
(*website_label* `mchars`)

`set-website-label` [Method]

Sets the label to be used for the website link. It defaults to the website URL.

about a `<gtk-about-dialog>`

website-label
the label used for the website link

Since 2.6

`gtk-about-dialog-get-logo` (*self* `<gtk-about-dialog>`) [Function]
⇒ (*ret* `<gdk-pixbuf>`)

`get-logo` [Method]

Returns the pixbuf displayed as logo in the about dialog.

about a `<gtk-about-dialog>`

ret the pixbuf displayed as logo. The pixbuf is owned by the about dialog. If you want to keep a reference to it, you have to call `g-object-ref` on it.

Since 2.6

`gtk-about-dialog-set-logo` (*self* `<gtk-about-dialog>`) [Function]
(*logo* `<gdk-pixbuf>`)

`set-logo` [Method]

Sets the pixbuf to be displayed as logo in the about dialog. If it is '`#f`', the default window icon set with `gtk-window-set-default-icon` will be used.

about a `<gtk-about-dialog>`

logo a `<gdk-pixbuf>`, or '`#f`'

Since 2.6

`gtk-about-dialog-get-logo-icon-name` (*self* `<gtk-about-dialog>`) [Function]
⇒ (*ret* `mchars`)

`get-logo-icon-name` [Method]

Returns the icon name displayed as logo in the about dialog.

about a `<gtk-about-dialog>`

ret the icon name displayed as logo. The string is owned by the dialog. If you want to keep a reference to it, you have to call `g-strdup` on it.

Since 2.6

`gtk-about-dialog-set-logo-icon-name` (*self* `<gtk-about-dialog>`) [Function]
(*icon_name* `mchars`)

`set-logo-icon-name` [Method]

Sets the pixbuf to be displayed as logo in the about dialog. If it is '`#f`', the default window icon set with `gtk-window-set-default-icon` will be used.

about a `<gtk-about-dialog>`

icon-name an icon name, or '`#f`'

Since 2.6

8 GtkAssistant

A widget used to guide users through multi-step operations

8.1 Overview

A <gtk-assistant> is a widget used to represent a generally complex operation splitted in several steps, guiding the user through its pages and controlling the page flow to collect the necessary data.

8.2 Usage

<gtk-assistant> [Class]
This <gobject> class defines no properties, other than those defined by its superclasses.

close [Signal on <gtk-assistant>]
The ::close signal is emitted either when the close button of a summary page is clicked, or when the apply button in the last page in the flow (of type GTK_ASSISTANT_PAGE_CONFIRM) is clicked.

Since 2.10

cancel [Signal on <gtk-assistant>]
The ::cancel signal is emitted when then the cancel button is clicked.

Since 2.10

prepare (*arg0* <gtk-widget>) [Signal on <gtk-assistant>]
The ::prepared signal is emitted when a new page is set as the assistant's current page, before making the new page visible. A handler for this signal can do any preparation which are necessary before showing *page*.

Since 2.10

apply [Signal on <gtk-assistant>]
The ::apply signal is emitted when the apply button is clicked. The default behavior of the <gtk-assistant> is to switch to the page after the current page, unless the current page is the last one.

A handler for the ::apply signal should carry out the actions for which the wizard has collected data. If the action takes a long time to complete, you might consider to put a page of type GTK_ASSISTANT_PAGE_PROGRESS after the confirmation page and handle this operation within the ::prepare signal of the progress page.

Since 2.10

gtk-assistant-new ⇒ (*ret* <gtk-widget>) [Function]
Creates a new <gtk-assistant>.

ret a newly created <gtk-assistant>

Since 2.10

`gtk-assistant-get-current-page` (*self* `<gtk-assistant>`) ⇒ (*ret* `int`) [Function]
`get-current-page` [Method]

Returns the page number of the current page

assistant a `<gtk-assistant>`

ret The index (starting from 0) of the current page in the *assistant*, if the *assistant* has no pages, -1 will be returned

Since 2.10

`gtk-assistant-set-current-page` (*self* `<gtk-assistant>`) (*page_num* `int`) [Function]
`set-current-page` [Method]

Switches the page to *page-num*. Note that this will only be necessary in custom buttons, as the *assistant* flow can be set with `gtk-assistant-set-forward-page-func`.

assistant a `<gtk-assistant>`

page-num index of the page to switch to, starting from 0. If negative, the last page will be used. If greater than the number of pages in the *assistant*, nothing will be done.

Since 2.10

`gtk-assistant-get-n-pages` (*self* `<gtk-assistant>`) ⇒ (*ret* `int`) [Function]
`get-n-pages` [Method]

Returns the number of pages in the *assistant*

assistant a `<gtk-assistant>`

ret The number of pages in the *assistant*.

Since 2.10

`gtk-assistant-get-nth-page` (*self* `<gtk-assistant>`) (*page_num* `int`) ⇒ (*ret* `<gtk-widget>`) [Function]
`get-nth-page` [Method]

Returns the child widget contained in page number *page-num*.

assistant a `<gtk-assistant>`

page-num The index of a page in the *assistant*, or -1 to get the last page;

ret The child widget, or '`#f`' if *page-num* is out of bounds.

Since 2.10

`gtk-assistant-prepend-page` (*self* `<gtk-assistant>`) (*page* `<gtk-widget>`) ⇒ (*ret* `int`) [Function]
`prepend-page` [Method]

Prepends a page to the *assistant*.

assistant a `<gtk-assistant>`

page a `<gtk-widget>`

ret the index (starting at 0) of the inserted page

Since 2.10

`gtk-assistant-append-page` (*self* `<gtk-assistant>`) [Function]
(*page* `<gtk-widget>`) ⇒ (*ret* `int`)

`append-page` [Method]

Appends a page to the *assistant*.

assistant a `<gtk-assistant>`

page a `<gtk-widget>`

ret the index (starting at 0) of the inserted page

Since 2.10

`gtk-assistant-insert-page` (*self* `<gtk-assistant>`) [Function]
(*page* `<gtk-widget>`) (*position* `int`) ⇒ (*ret* `int`)

`insert-page` [Method]

Inserts a page in the *assistant* at a given position.

assistant a `<gtk-assistant>`

page a `<gtk-widget>`

position the index (starting at 0) at which to insert the page, or -1 to append the page to the *assistant*

ret the index (starting from 0) of the inserted page

Since 2.10

`gtk-assistant-set-page-type` (*self* `<gtk-assistant>`) [Function]
(*page* `<gtk-widget>`) (*type* `<gtk-assistant-page-type>`)

`set-page-type` [Method]

Sets the page type for *page*. The page type determines the page behavior in the *assistant*.

assistant a `<gtk-assistant>`

page a page of *assistant*

type the new type for *page*

Since 2.10

`gtk-assistant-get-page-type` (*self* `<gtk-assistant>`) [Function]
(*page* `<gtk-widget>`) ⇒ (*ret* `<gtk-assistant-page-type>`)

`get-page-type` [Method]

Gets the page type of *page*.

assistant a `<gtk-assistant>`

page a page of *assistant*

ret the page type of *page*.

Since 2.10

`gtk-assistant-set-page-title` (*self* `<gtk-assistant>`) [Function]
(*page* `<gtk-widget>`) (*title* `mchars`)

`set-page-title` [Method]

Sets a title for *page*. The title is displayed in the header area of the assistant when *page* is the current page.

assistant a `<gtk-assistant>`

page a page of *assistant*

title the new title for *page*

Since 2.10

`gtk-assistant-get-page-title` (*self* `<gtk-assistant>`) [Function]
(*page* `<gtk-widget>`) ⇒ (*ret* `mchars`)

`get-page-title` [Method]

Gets the title for *page*.

assistant a `<gtk-assistant>`

page a page of *assistant*

ret the title for *page*.

Since 2.10

`gtk-assistant-set-page-header-image` (*self* `<gtk-assistant>`) [Function]
(*page* `<gtk-widget>`) (*pixbuf* `<gdk-pixbuf>`)

`set-page-header-image` [Method]

Sets a header image for *page*. This image is displayed in the header area of the assistant when *page* is the current page.

assistant a `<gtk-assistant>`

page a page of *assistant*

pixbuf the new header image *page*

Since 2.10

`gtk-assistant-get-page-header-image` (*self* `<gtk-assistant>`) [Function]
(*page* `<gtk-widget>`) ⇒ (*ret* `<gdk-pixbuf>`)

`get-page-header-image` [Method]

Gets the header image for *page*.

assistant a `<gtk-assistant>`

page a page of *assistant*

ret the header image for *page*, or '`#f`' if there's no header image for the page.

Since 2.10

`gtk-assistant-set-page-side-image` (*self* `<gtk-assistant>`) [Function]
(*page* `<gtk-widget>`) (*pixbuf* `<gdk-pixbuf>`)

`set-page-side-image` [Method]

Sets a header image for *page*. This image is displayed in the side area of the assistant when *page* is the current page.

assistant a `<gtk-assistant>`

page a page of *assistant*

pixbuf the new header image *page*

Since 2.10

`gtk-assistant-get-page-side-image` (*self* `<gtk-assistant>`) [Function]
(*page* `<gtk-widget>`) ⇒ (*ret* `<gdk-pixbuf>`)

`get-page-side-image` [Method]

Gets the header image for *page*.

assistant a `<gtk-assistant>`

page a page of *assistant*

ret the side image for *page*, or '`#f`' if there's no side image for the page.

Since 2.10

`gtk-assistant-set-page-complete` (*self* `<gtk-assistant>`) [Function]
(*page* `<gtk-widget>`) (*complete* `bool`)

`set-page-complete` [Method]

Sets whether *page* contents are complete. This will make *assistant* update the buttons state to be able to continue the task.

assistant a `<gtk-assistant>`

page a page of *assistant*

complete the completeness status of the page

Since 2.10

`gtk-assistant-get-page-complete` (*self* `<gtk-assistant>`) [Function]
(*page* `<gtk-widget>`) ⇒ (*ret* `bool`)

`get-page-complete` [Method]

Gets whether *page* is complete..

assistant a `<gtk-assistant>`

page a page of *assistant*

ret '`#t`' if *page* is complete.

Since 2.10

`gtk-assistant-add-action-widget` (*self* `<gtk-assistant>`) [Function]
(*child* `<gtk-widget>`)

`add-action-widget` [Method]

Adds a widget to the action area of a `<gtk-assistant>`.

assistant a `<gtk-assistant>`

child a `<gtk-widget>`

Since 2.10

`gtk-assistant-remove-action-widget` (*self* `<gtk-assistant>`) [Function]
(*child* `<gtk-widget>`)
`remove-action-widget` [Method]

Removes a widget from the action area of a `<gtk-assistant>`.

assistant a `<gtk-assistant>`

child a `<gtk-widget>`

Since 2.10

`gtk-assistant-update-buttons-state` (*self* `<gtk-assistant>`) [Function]
`update-buttons-state` [Method]

Forces *assistant* to recompute the buttons state.

GTK+ automatically takes care of this in most situations, e.g. when the user goes to a different page, or when the visibility or completeness of a page changes.

One situation where it can be necessary to call this function is when changing a value on the current page affects the future page flow of the assistant.

assistant a `<gtk-assistant>`

Since 2.10

9 GtkAccelLabel

A label which displays an accelerator key on the right of the text

9.1 Overview

The <gtk-accel-label> widget is a subclass of <gtk-label> that also displays an accelerator key on the right of the label text, e.g. 'Ctl+S'. It is commonly used in menus to show the keyboard short-cuts for commands.

The accelerator key to display is not set explicitly. Instead, the <gtk-accel-label> displays the accelerators which have been added to a particular widget. This widget is set by calling gtk-accel-label-set-accel-widget.

For example, a <gtk-menu-item> widget may have an accelerator added to emit the "activate" signal when the 'Ctl+S' key combination is pressed. A <gtk-accel-label> is created and added to the <gtk-menu-item>, and gtk-accel-label-set-accel-widget is called with the <gtk-menu-item> as the second argument. The <gtk-accel-label> will now display 'Ctl+S' after its label.

Note that creating a <gtk-menu-item> with gtk-menu-item-new-with-label (or one of the similar functions for <gtk-check-menu-item> and <gtk-radio-menu-item>) automatically adds a <gtk-accel-label> to the <gtk-menu-item> and calls gtk-accel-label-set-accel-widget to set it up for you.

A <gtk-accel-label> will only display accelerators which have 'GTK_ACCEL_VISIBLE' set (see <gtk-accel-flags>). A <gtk-accel-label> can display multiple accelerators and even signal names, though it is almost always used to display just one accelerator key.

```
GtkWidget *save_item;
GtkAccelGroup *accel_group;

/* Create a GtkAccelGroup and add it to the window. */
accel_group = gtk_accel_group_new ();
gtk_window_add_accel_group (GTK_WINDOW (window), accel_group);

/* Create the menu item using the convenience function. */
save_item = gtk_menu_item_new_with_label ("Save");
gtk_widget_show (save_item);
gtk_container_add (GTK_CONTAINER (menu), save_item);

/* Now add the accelerator to the GtkMenuItem. Note that since we called
   gtk_menu_item_new_with_label() to create the GtkMenuItem the
   GtkAccelLabel is automatically set up to display the GtkMenuItem
   accelerators. We just need to make sure we use GTK_ACCEL_VISIBLE here. */
gtk_widget_add_accelerator (save_item, "activate", accel_group,
                            GDK_s, GDK_CONTROL_MASK, GTK_ACCEL_VISIBLE);
```

9.2 Usage

`<gtk-accel-label>` [Class]

This `<gobject>` class defines the following properties:

`accel-closure`
: The closure to be monitored for accelerator changes

`accel-widget`
: The widget to be monitored for accelerator changes

`gtk-accel-label-new` (*string* `mchars`) ⇒ (*ret* `<gtk-widget>`) [Function]

Creates a new `<gtk-accel-label>`.

string the label string. Must be non-'`#f`'.

ret a new `<gtk-accel-label>`.

`gtk-accel-label-set-accel-closure` (*self* `<gtk-accel-label>`) [Function]
(*accel_closure* `<gclosure>`)
`set-accel-closure` [Method]

Sets the closure to be monitored by this accelerator label. The closure must be connected to an accelerator group; see `gtk-accel-group-connect`.

accel-label a `<gtk-accel-label>`

accel-closure
: the closure to monitor for accelerator changes.

`gtk-accel-label-get-accel-widget` (*self* `<gtk-accel-label>`) [Function]
⇒ (*ret* `<gtk-widget>`)
`get-accel-widget` [Method]

Fetches the widget monitored by this accelerator label. See `gtk-accel-label-set-accel-widget`.

accel-label a `<gtk-accel-label>`

ret the object monitored by the accelerator label, or '`#f`'.

`gtk-accel-label-set-accel-widget` (*self* `<gtk-accel-label>`) [Function]
(*accel_widget* `<gtk-widget>`)
`set-accel-widget` [Method]

Sets the widget to be monitored by this accelerator label.

accel-label a `<gtk-accel-label>`

accel-widget
: the widget to be monitored.

`gtk-accel-label-get-accel-width` (*self* `<gtk-accel-label>`) [Function]
⇒ (*ret* `unsigned-int`)
`get-accel-width` [Method]

Returns the width needed to display the accelerator key(s). This is used by menus to align all of the `<gtk-menu-item>` widgets, and shouldn't be needed by applications.

accel-label a `<gtk-accel-label>`.

ret the width needed to display the accelerator key(s).

`gtk-accel-label-refetch` (*self* `<gtk-accel-label>`) ⇒ (*ret* `bool`) [Function]
`refetch` [Method]

Recreates the string representing the accelerator keys. This should not be needed since the string is automatically updated whenever accelerators are added or removed from the associated widget.

accel-label a `<gtk-accel-label>`.

ret always returns '`#f`'.

10 GtkImage

A widget displaying an image

10.1 Overview

The <gtk-image> widget displays an image. Various kinds of object can be displayed as an image; most typically, you would load a <gdk-pixbuf> ("pixel buffer") from a file, and then display that. There's a convenience function to do this, gtk-image-new-from-file, used as follows: If the file isn't loaded successfully, the image will contain a "broken image" icon similar to that used in many web browsers. If you want to handle errors in loading the file yourself, for example by displaying an error message, then load the image with gdk-pixbuf-new-from-file, then create the <gtk-image> with gtk-image-new-from-pixbuf.

```
GtkWidget *image;
image = gtk_image_new_from_file ("myfile.png");
```

The image file may contain an animation, if so the <gtk-image> will display an animation (<gdk-pixbuf-animation>) instead of a static image.

<gtk-image> is a subclass of <gtk-misc>, which implies that you can align it (center, left, right) and add padding to it, using <gtk-misc> methods.

<gtk-image> is a "no window" widget (has no <gdk-window> of its own), so by default does not receive events. If you want to receive events on the image, such as button clicks, place the image inside a <gtk-event-box>, then connect to the event signals on the event box.

```
static gboolean
button_press_callback (GtkWidget      *event_box,
                       GdkEventButton *event,
                       gpointer        data)
{
  g_print ("Event box clicked at coordinates %f,%f\n",
           event->x, event->y);

  /* Returning TRUE means we handled the event, so the signal
   * emission should be stopped (don't call any further
   * callbacks that may be connected). Return FALSE
   * to continue invoking callbacks.
   */
  return TRUE;
}

static GtkWidget*
create_image (void)
{
  GtkWidget *image;
  GtkWidget *event_box;
```

```
  image = gtk_image_new_from_file ("myfile.png");

  event_box = gtk_event_box_new ();

  gtk_container_add (GTK_CONTAINER (event_box), image);

  g_signal_connect (G_OBJECT (event_box),
                    "button_press_event",
                    G_CALLBACK (button_press_callback),
                    image);

  return image;
}
```

When handling events on the event box, keep in mind that coordinates in the image may be different from event box coordinates due to the alignment and padding settings on the image (see <gtk-misc>). The simplest way to solve this is to set the alignment to 0.0 (left/top), and set the padding to zero. Then the origin of the image will be the same as the origin of the event box.

Sometimes an application will want to avoid depending on external data files, such as image files. GTK+ comes with a program to avoid this, called . This program allows you to convert an image into a C variable declaration, which can then be loaded into a <gdk-pixbuf> using gdk-pixbuf-new-from-inline.

10.2 Usage

<gtk-image> [Class]

This <gobject> class defines the following properties:

pixbuf A GdkPixbuf to display

pixmap A GdkPixmap to display

image A GdkImage to display

mask Mask bitmap to use with GdkImage or GdkPixmap

file Filename to load and display

stock Stock ID for a stock image to display

icon-set Icon set to display

icon-size
Symbolic size to use for stock icon, icon set or named icon

pixel-size
Pixel size to use for named icon

pixbuf-animation
GdkPixbufAnimation to display

`icon-name`
The name of the icon from the icon theme

`storage-type`
The representation being used for image data

`gtk-image-get-icon-set` (*self* `<gtk-image>`) [Function]
(*icon_set* `<gtk-icon-set**>`) (*size* `<gtk-icon-size*>`)
`get-icon-set` [Method]
Gets the icon set and size being displayed by the `<gtk-image>`. The storage type of the image must be '`GTK_IMAGE_EMPTY`' or '`GTK_IMAGE_ICON_SET`' (see `gtk-image-get-storage-type`).

image a `<gtk-image>`

icon-set location to store a `<gtk-icon-set>`

size location to store a stock icon size

`gtk-image-get-image` (*self* `<gtk-image>`) [Function]
(*gdk_image* `<gdk-image**>`) (*mask* `<gdk-bitmap**>`)
`get-image` [Method]
Gets the `<gdk-image>` and mask being displayed by the `<gtk-image>`. The storage type of the image must be '`GTK_IMAGE_EMPTY`' or '`GTK_IMAGE_IMAGE`' (see `gtk-image-get-storage-type`). The caller of this function does not own a reference to the returned image and mask.

image a `<gtk-image>`

gdk-image return location for a `<gtk-image>`

mask return location for a `<gdk-bitmap>`

`gtk-image-get-pixbuf` (*self* `<gtk-image>`) ⇒ (*ret* `<gdk-pixbuf>`) [Function]
`get-pixbuf` [Method]
Gets the `<gdk-pixbuf>` being displayed by the `<gtk-image>`. The storage type of the image must be '`GTK_IMAGE_EMPTY`' or '`GTK_IMAGE_PIXBUF`' (see `gtk-image-get-storage-type`). The caller of this function does not own a reference to the returned pixbuf.

image a `<gtk-image>`

ret the displayed pixbuf, or '`#f`' if the image is empty

`gtk-image-get-pixmap` (*self* `<gtk-image>`) [Function]
(*pixmap* `<gdk-pixmap**>`) (*mask* `<gdk-bitmap**>`)
`get-pixmap` [Method]
Gets the pixmap and mask being displayed by the `<gtk-image>`. The storage type of the image must be '`GTK_IMAGE_EMPTY`' or '`GTK_IMAGE_PIXMAP`' (see `gtk-image-get-storage-type`). The caller of this function does not own a reference to the returned pixmap and mask.

image a `<gtk-image>`

pixmap location to store the pixmap, or '`#f`'

mask location to store the mask, or '`#f`'

`gtk-image-get-stock` (*self* `<gtk-image>`) (*size* `<gtk-icon-size*>`) [Function]
⇒ (*stock_id* `mchars`)

`get-stock` [Method]

Gets the stock icon name and size being displayed by the `<gtk-image>`. The storage type of the image must be '`GTK_IMAGE_EMPTY`' or '`GTK_IMAGE_STOCK`' (see `gtk-image-get-storage-type`). The returned string is owned by the `<gtk-image>` and should not be freed.

image a `<gtk-image>`

stock-id place to store a stock icon name

size place to store a stock icon size

`gtk-image-get-animation` (*self* `<gtk-image>`) [Function]
⇒ (*ret* `<gdk-pixbuf-animation>`)

`get-animation` [Method]

Gets the `<gdk-pixbuf-animation>` being displayed by the `<gtk-image>`. The storage type of the image must be '`GTK_IMAGE_EMPTY`' or '`GTK_IMAGE_ANIMATION`' (see `gtk-image-get-storage-type`). The caller of this function does not own a reference to the returned animation.

image a `<gtk-image>`

ret the displayed animation, or '`#f`' if the image is empty

`gtk-image-get-icon-name` (*self* `<gtk-image>`) [Function]
(*icon_name* `<gchar**>`) (*size* `<gtk-icon-size*>`)

`get-icon-name` [Method]

Gets the icon name and size being displayed by the `<gtk-image>`. The storage type of the image must be '`GTK_IMAGE_EMPTY`' or '`GTK_IMAGE_ICON_NAME`' (see `gtk-image-get-storage-type`). The returned string is owned by the `<gtk-image>` and should not be freed.

image a `<gtk-image>`

icon-name place to store an icon name

size place to store an icon size

Since 2.6

`gtk-image-get-storage-type` (*self* `<gtk-image>`) [Function]
⇒ (*ret* `<gtk-image-type>`)

`get-storage-type` [Method]

Gets the type of representation being used by the `<gtk-image>` to store image data. If the `<gtk-image>` has no image data, the return value will be '`GTK_IMAGE_EMPTY`'.

image a `<gtk-image>`

ret image representation being used

`gtk-image-new-from-file` (*filename* `mchars`) ⇒ (*ret* `<gtk-widget>`) [Function]

Creates a new `<gtk-image>` displaying the file *filename*. If the file isn't found or can't be loaded, the resulting `<gtk-image>` will display a `"broken image"` icon. This function never returns '`#f`', it always returns a valid `<gtk-image>` widget.

If the file contains an animation, the image will contain an animation.

If you need to detect failures to load the file, use `gdk-pixbuf-new-from-file` to load the file yourself, then create the `<gtk-image>` from the pixbuf. (Or for animations, use `gdk-pixbuf-animation-new-from-file`).

The storage type (`gtk-image-get-storage-type`) of the returned image is not defined, it will be whatever is appropriate for displaying the file.

filename a filename

ret a new `<gtk-image>`

`gtk-image-new-from-icon-set` (*icon_set* `<gtk-icon-set>`) [Function]
(*size* `<gtk-icon-size>`) ⇒ (*ret* `<gtk-widget>`)

Creates a `<gtk-image>` displaying an icon set. Sample stock sizes are `<gtk-icon-size-menu>`, `<gtk-icon-size-small-toolbar>`. Instead of using this function, usually it's better to create a `<gtk-icon-factory>`, put your icon sets in the icon factory, add the icon factory to the list of default factories with `gtk-icon-factory-add-default`, and then use `gtk-image-new-from-stock`. This will allow themes to override the icon you ship with your application.

The `<gtk-image>` does not assume a reference to the icon set; you still need to unref it if you own references. `<gtk-image>` will add its own reference rather than adopting yours.

icon-set a `<gtk-icon-set>`

size a stock icon size

ret a new `<gtk-image>`

`gtk-image-new-from-image` (*image* `<gdk-image>`) [Function]
(*mask* `<gdk-bitmap*>`) ⇒ (*ret* `<gtk-widget>`)

Creates a `<gtk-image>` widget displaying a *image* with a *mask*. A `<gdk-image>` is a client-side image buffer in the pixel format of the current display. The `<gtk-image>` does not assume a reference to the image or mask; you still need to unref them if you own references. `<gtk-image>` will add its own reference rather than adopting yours.

image a `<gdk-image>`, or '`#f`'

mask a `<gdk-bitmap>`, or '`#f`'

ret a new `<gtk-image>`

`gtk-image-new-from-pixbuf` (*pixbuf* `<gdk-pixbuf>`) [Function]
⇒ (*ret* `<gtk-widget>`)

Creates a new `<gtk-image>` displaying *pixbuf*. The `<gtk-image>` does not assume a reference to the pixbuf; you still need to unref it if you own references. `<gtk-image>` will add its own reference rather than adopting yours.

Note that this function just creates an <gtk-image> from the pixbuf. The <gtk-image> created will not react to state changes. Should you want that, you should use gtk-image-new-from-icon-set.

pixbuf a <gdk-pixbuf>, or '#f'

ret a new <gtk-image>

gtk-image-new-from-pixmap (*pixmap* <gdk-pixmap*>) [Function]
(*mask* <gdk-bitmap*>) ⇒ (*ret* <gtk-widget>)

Creates a <gtk-image> widget displaying *pixmap* with a *mask*. A <gdk-pixmap> is a server-side image buffer in the pixel format of the current display. The <gtk-image> does not assume a reference to the pixmap or mask; you still need to unref them if you own references. <gtk-image> will add its own reference rather than adopting yours.

pixmap a <gdk-pixmap>, or '#f'

mask a <gdk-bitmap>, or '#f'

ret a new <gtk-image>

gtk-image-new-from-stock (*stock_id* mchars) [Function]
(*size* <gtk-icon-size>) ⇒ (*ret* <gtk-widget>)

Creates a <gtk-image> displaying a stock icon. Sample stock icon names are <gtk-stock-open>, <gtk-stock-quit>. Sample stock sizes are <gtk-icon-size-menu>, <gtk-icon-size-small-toolbar>. If the stock icon name isn't known, the image will be empty. You can register your own stock icon names, see gtk-icon-factory-add-default and gtk-icon-factory-add.

stock-id a stock icon name

size a stock icon size

ret a new <gtk-image> displaying the stock icon

gtk-image-new-from-animation [Function]
(*animation* <gdk-pixbuf-animation>) ⇒ (*ret* <gtk-widget>)

Creates a <gtk-image> displaying the given animation. The <gtk-image> does not assume a reference to the animation; you still need to unref it if you own references. <gtk-image> will add its own reference rather than adopting yours.

Note that the animation frames are shown using a timeout with <g-priority-default>. When using animations to indicate busyness, keep in mind that the animation will only be shown if the main loop is not busy with something that has a higher priority.

animation an animation

ret a new <gtk-image> widget

gtk-image-new-from-icon-name (*icon_name* mchars) [Function]
(*size* <gtk-icon-size>) ⇒ (*ret* <gtk-widget>)

Creates a <gtk-image> displaying an icon from the current icon theme. If the icon name isn't known, a "broken image" icon will be displayed instead. If the current icon theme is changed, the icon will be updated appropriately.

icon-name an icon name

size a stock icon size

ret a new `<gtk-image>` displaying the themed icon

Since 2.6

`gtk-image-set-from-file` (*self* `<gtk-image>`) (*filename* `mchars`) [Function]
`set-from-file` [Method]

See `gtk-image-new-from-file` for details.

image a `<gtk-image>`

filename a filename or '`#f`'

`gtk-image-set-from-icon-set` (*self* `<gtk-image>`) [Function]
(*icon_set* `<gtk-icon-set>`) (*size* `<gtk-icon-size>`)
`set-from-icon-set` [Method]

See `gtk-image-new-from-icon-set` for details.

image a `<gtk-image>`

icon-set a `<gtk-icon-set>`

size a stock icon size

`gtk-image-set-from-image` (*self* `<gtk-image>`) [Function]
(*gdk_image* `<gdk-image>`) (*mask* `<gdk-bitmap*>`)
`set-from-image` [Method]

See `gtk-image-new-from-image` for details.

image a `<gtk-image>`

gdk-image a `<gdk-image>` or '`#f`'

mask a `<gdk-bitmap>` or '`#f`'

`gtk-image-set-from-pixbuf` (*self* `<gtk-image>`) [Function]
(*pixbuf* `<gdk-pixbuf>`)
`set-from-pixbuf` [Method]

See `gtk-image-new-from-pixbuf` for details.

image a `<gtk-image>`

pixbuf a `<gdk-pixbuf>` or '`#f`'

`gtk-image-set-from-pixmap` (*self* `<gtk-image>`) [Function]
(*pixmap* `<gdk-pixmap*>`) (*mask* `<gdk-bitmap*>`)
`set-from-pixmap` [Method]

See `gtk-image-new-from-pixmap` for details.

image a `<gtk-image>`

pixmap a `<gdk-pixmap>` or '`#f`'

mask a `<gdk-bitmap>` or '`#f`'

`gtk-image-set-from-stock` (*self* `<gtk-image>`) (*stock_id* `mchars`) (*size* `<gtk-icon-size>`) [Function]
`set-from-stock` [Method]

See `gtk-image-new-from-stock` for details.

image a `<gtk-image>`

stock-id a stock icon name

size a stock icon size

`gtk-image-set-from-animation` (*self* `<gtk-image>`) (*animation* `<gdk-pixbuf-animation>`) [Function]
`set-from-animation` [Method]

Causes the `<gtk-image>` to display the given animation (or display nothing, if you set the animation to '`#f`').

image a `<gtk-image>`

animation the `<gdk-pixbuf-animation>`

`gtk-image-set-from-icon-name` (*self* `<gtk-image>`) (*icon_name* `mchars`) (*size* `<gtk-icon-size>`) [Function]
`set-from-icon-name` [Method]

See `gtk-image-new-from-icon-name` for details.

image a `<gtk-image>`

icon-name an icon name

size an icon size

Since 2.6

`gtk-image-clear` (*self* `<gtk-image>`) [Function]
`clear` [Method]

Resets the image to be empty.

image a `<gtk-image>`

Since 2.8

`gtk-image-new` ⇒ (*ret* `<gtk-widget>`) [Function]

Creates a new empty `<gtk-image>` widget.

ret a newly created `<gtk-image>` widget.

`gtk-image-set-pixel-size` (*self* `<gtk-image>`) (*pixel_size* `int`) [Function]
`set-pixel-size` [Method]

Sets the pixel size to use for named icons. If the pixel size is set to a value != -1, it is used instead of the icon size set by `gtk-image-set-from-icon-name`.

image a `<gtk-image>`

pixel-size the new pixel size

Since 2.6

`gtk-image-get-pixel-size` (*self* `<gtk-image>`) ⇒ (*ret* `int`) [Function]
`get-pixel-size` [Method]

Gets the pixel size used for named icons.

image a `<gtk-image>`

ret the pixel size used for named icons.

Since 2.6

11 GtkLabel

A widget that displays a small to medium amount of text

11.1 Overview

The `<gtk-label>` widget displays a small amount of text. As the name implies, most labels are used to label another widget such as a `<gtk-button>`, a `<gtk-menu-item>`, or a `<gtk-option-menu>`.

11.2 Mnemonics

Labels may contain *mnemonics*. Mnemonics are underlined characters in the label, used for keyboard navigation. Mnemonics are created by providing a string with an underscore before the mnemonic character, such as '`"_File"`', to the functions `gtk-label-new-with-mnemonic` or `gtk-label-set-text-with-mnemonic`.

Mnemonics automatically activate any activatable widget the label is inside, such as a `<gtk-button>`; if the label is not inside the mnemonic's target widget, you have to tell the label about the target using `gtk-label-set-mnemonic-widget`. Here's a simple example where the label is inside a button: There's a convenience function to create buttons with a mnemonic label already inside: To create a mnemonic for a widget alongside the label, such as a `<gtk-entry>`, you have to point the label at the entry with `gtk-label-set-mnemonic-widget`:

```
/* Pressing Alt+H will activate this button */
button = gtk_button_new ();
label = gtk_label_new_with_mnemonic ("_Hello");
gtk_container_add (GTK_CONTAINER (button), label);

/* Pressing Alt+H will activate this button */
button = gtk_button_new_with_mnemonic ("_Hello");

/* Pressing Alt+H will focus the entry */
entry = gtk_entry_new ();
label = gtk_label_new_with_mnemonic ("_Hello");
gtk_label_set_mnemonic_widget (GTK_LABEL (label), entry);
```

11.3 Markup (styled text)

To make it easy to format text in a label (changing colors, fonts, etc.), label text can be provided in a simple markup format. Here's how to create a label with a small font: (See complete documentation of available tags in the Pango manual.)

```
label = gtk_label_new (NULL);
gtk_label_set_markup (GTK_LABEL (label), "<small>Small text</small>");
```

The markup passed to `gtk-label-set-markup` must be valid; for example, literal </>/& characters must be escaped as <, >, and &. If you pass text obtained from the

user, file, or a network to `gtk-label-set-markup`, you'll want to escape it with `g-markup-escape-text` or `g-markup-printf-escaped`.

Markup strings are just a convenient way to set the `<pango-attr-list>` on a label; `gtk-label-set-attributes` may be a simpler way to set attributes in some cases. Be careful though; `<pango-attr-list>` tends to cause internationalization problems, unless you're applying attributes to the entire string (i.e. unless you set the range of each attribute to [0, G_MAXINT)). The reason is that specifying the start_index and end_index for a `<pango-attribute>` requires knowledge of the exact string being displayed, so translations will cause problems.

11.4 Selectable labels

Labels can be made selectable with `gtk-label-set-selectable`. Selectable labels allow the user to copy the label contents to the clipboard. Only labels that contain useful-to-copy information — such as error messages — should be made selectable.

11.5 Text layout

A label can contain any number of paragraphs, but will have performance problems if it contains more than a small number. Paragraphs are separated by newlines or other paragraph separators understood by Pango.

Labels can automatically wrap text if you call `gtk-label-set-line-wrap`.

`gtk-label-set-justify` sets how the lines in a label align with one another. If you want to set how the label as a whole aligns in its available space, see `gtk-misc-set-alignment`.

11.6 Usage

`<gtk-label>` [Class]

This `<gobject>` class defines the following properties:

`label` The text of the label

`attributes`
A list of style attributes to apply to the text of the label

`use-markup`
The text of the label includes XML markup. See pango_parse_markup()

`use-underline`
If set, an underline in the text indicates the next character should be used for the mnemonic accelerator key

`justify` The alignment of the lines in the text of the label relative to each other. This does NOT affect the alignment of the label within its allocation. See GtkMisc::xalign for that

`pattern` A string with _ characters in positions correspond to characters in the text to underline

`wrap` If set, wrap lines if the text becomes too wide

`wrap-mode`
: If wrap is set, controls how linewrapping is done

`selectable`
: Whether the label text can be selected with the mouse

`mnemonic-keyval`
: The mnemonic accelerator key for this label

`mnemonic-widget`
: The widget to be activated when the label's mnemonic key is pressed

`cursor-position`
: The current position of the insertion cursor in chars

`selection-bound`
: The position of the opposite end of the selection from the cursor in chars

`ellipsize`
: The preferred place to ellipsize the string, if the label does not have enough room to display the entire string

`width-chars`
: The desired width of the label, in characters

`single-line-mode`
: Whether the label is in single line mode

`angle`
: Angle at which the label is rotated

`max-width-chars`
: The desired maximum width of the label, in characters

`move-cursor` (*arg0* `<gtk-movement-step>`) (*arg1* `<gint>`) (*arg2* `<gboolean>`) [Signal on `<gtk-label>`]

`copy-clipboard` [Signal on `<gtk-label>`]

`populate-popup` (*arg0* `<gtk-menu>`) [Signal on `<gtk-label>`]

`gtk-label-new` (*str* `mchars`) ⇒ (*ret* `<gtk-widget>`) [Function]

Creates a new label with the given text inside it. You can pass '`#f`' to get an empty label widget.

str
: The text of the label

ret
: the new `<gtk-label>`

`gtk-label-set-text` (*self* `<gtk-label>`) (*str* `mchars`) [Function]
`set-text` [Method]

Sets the text within the `<gtk-label>` widget. It overwrites any text that was there before.

This will also clear any previously set mnemonic accelerators.

label
: a `<gtk-label>`

str
: The text you want to set.

`gtk-label-set-attributes` (*self* `<gtk-label>`) [Function]
(*attrs* `<pango-attr-list>`)
`set-attributes` [Method]

Sets a `<pango-attr-list>`; the attributes in the list are applied to the label text. The attributes set with this function will be ignored if the "use_underline" property or the "use_markup" property is '`#t`'.

label a `<gtk-label>`

attrs a `<pango-attr-list>`

`gtk-label-set-markup` (*self* `<gtk-label>`) (*str* `mchars`) [Function]
`set-markup` [Method]

Parses *str* which is marked up with the Pango text markup language, setting the label's text and attribute list based on the parse results. If the *str* is external data, you may need to escape it with `g-markup-escape-text` or `g-markup-printf-escaped`:

```
char *markup;

markup = g_markup_printf_escaped ("<span style=\"italic\">%s</span>", str);
gtk_label_set_markup (GTK_LABEL (label), markup);
g_free (markup);
```

label a `<gtk-label>`

str a markup string (see Pango markup format)

`gtk-label-set-markup-with-mnemonic` (*self* `<gtk-label>`) [Function]
(*str* `mchars`)
`set-markup-with-mnemonic` [Method]

Parses *str* which is marked up with the Pango text markup language, setting the label's text and attribute list based on the parse results. If characters in *str* are preceded by an underscore, they are underlined indicating that they represent a keyboard accelerator called a mnemonic.

The mnemonic key can be used to activate another widget, chosen automatically, or explicitly using `gtk-label-set-mnemonic-widget`.

label a `<gtk-label>`

str a markup string (see Pango markup format)

`gtk-label-set-pattern` (*self* `<gtk-label>`) (*pattern* `mchars`) [Function]
`set-pattern` [Method]

The pattern of underlines you want under the existing text within the `<gtk-label>` widget. For example if the current text of the label says "FooBarBaz" passing a pattern of "`___ ___`" will underline "Foo" and "Baz" but not "Bar".

label The `<gtk-label>` you want to set the pattern to.

pattern The pattern as described above.

`gtk-label-set-justify` (*self* `<gtk-label>`) [Function]
(*jtype* `<gtk-justification>`)
`set-justify` [Method]

Sets the alignment of the lines in the text of the label relative to each other. '`GTK_JUSTIFY_LEFT`' is the default value when the widget is first created with `gtk-label-new`. If you instead want to set the alignment of the label as a whole, use `gtk-misc-set-alignment` instead. `gtk-label-set-justify` has no effect on labels containing only a single line.

label a `<gtk-label>`

jtype a `<gtk-justification>`

`gtk-label-set-ellipsize` (*self* `<gtk-label>`) [Function]
(*mode* `<pango-ellipsize-mode>`)
`set-ellipsize` [Method]

Sets the mode used to ellipsize (add an ellipsis: `"..."`) to the text if there is not enough space to render the entire string.

label a `<gtk-label>`

mode a `<pango-ellipsize-mode>`

Since 2.6

`gtk-label-set-width-chars` (*self* `<gtk-label>`) (*n_chars* `int`) [Function]
`set-width-chars` [Method]

Sets the desired width in characters of *label* to *n-chars*.

label a `<gtk-label>`

n-chars the new desired width, in characters.

Since 2.6

`gtk-label-set-max-width-chars` (*self* `<gtk-label>`) (*n_chars* `int`) [Function]
`set-max-width-chars` [Method]

Sets the desired maximum width in characters of *label* to *n-chars*.

label a `<gtk-label>`

n-chars the new desired maximum width, in characters.

Since 2.6

`gtk-label-set-line-wrap` (*self* `<gtk-label>`) (*wrap* `bool`) [Function]
`set-line-wrap` [Method]

Toggles line wrapping within the `<gtk-label>` widget. '`#t`' makes it break lines if text exceeds the widget's size. '`#f`' lets the text get cut off by the edge of the widget if it exceeds the widget size.

Note that setting line wrapping to '`#t`' does not make the label wrap at its parent container's width, because GTK+ widgets conceptually can't make their requisition depend on the parent container's size. For a label that wraps at a specific position, set the label's width using `gtk-widget-set-size-request`.

label a `<gtk-label>`

wrap the setting

`gtk-label-set-line-wrap-mode` (*self* `<gtk-label>`) [Function]
(*wrap_mode* `<pango-wrap-mode>`)
`set-line-wrap-mode` [Method]

If line wrapping is on (see `gtk-label-set-line-wrap`) this controls how the line wrapping is done. The default is '`PANGO_WRAP_WORD`' which means wrap on word boundaries.

label a `<gtk-label>`

wrap-mode
the line wrapping mode

Since 2.10

`gtk-label-get-layout-offsets` (*self* `<gtk-label>`) ⇒ (*x* `int`) [Function]
(*y* `int`)
`get-layout-offsets` [Method]

Obtains the coordinates where the label will draw the `<pango-layout>` representing the text in the label; useful to convert mouse events into coordinates inside the `<pango-layout>`, e.g. to take some action if some part of the label is clicked. Of course you will need to create a `<gtk-event-box>` to receive the events, and pack the label inside it, since labels are a `<gtk-no-window>` widget. Remember when using the `<pango-layout>` functions you need to convert to and from pixels using `pango-pixels` or `<pango-scale>`.

label a `<gtk-label>`

x location to store X offset of layout, or '`#f`'

y location to store Y offset of layout, or '`#f`'

`gtk-label-get-mnemonic-keyval` (*self* `<gtk-label>`) [Function]
⇒ (*ret* `unsigned-int`)
`get-mnemonic-keyval` [Method]

If the label has been set so that it has an mnemonic key this function returns the keyval used for the mnemonic accelerator. If there is no mnemonic set up it returns `<gdk--void-symbol>`.

label a `<gtk-label>`

ret GDK keyval usable for accelerators, or `<gdk--void-symbol>`

`gtk-label-get-selectable` (*self* `<gtk-label>`) ⇒ (*ret* `bool`) [Function]
`get-selectable` [Method]

Gets the value set by `gtk-label-set-selectable`.

label a `<gtk-label>`

ret '`#t`' if the user can copy text from the label

`gtk-label-get-text` (*self* `<gtk-label>`) ⇒ (*ret* `mchars`) [Function]
`get-text` [Method]

Fetches the text from a label widget, as displayed on the screen. This does not include any embedded underlines indicating mnemonics or Pango markup. (See `gtk-label-get-label`)

label a `<gtk-label>`

ret the text in the label widget. This is the internal string used by the label, and must not be modified.

`gtk-label-new-with-mnemonic` (*str* `mchars`) ⇒ (*ret* `<gtk-widget>`) [Function]

Creates a new `<gtk-label>`, containing the text in *str*.

If characters in *str* are preceded by an underscore, they are underlined. If you need a literal underscore character in a label, use '__' (two underscores). The first underlined character represents a keyboard accelerator called a mnemonic. The mnemonic key can be used to activate another widget, chosen automatically, or explicitly using `gtk-label-set-mnemonic-widget`.

If `gtk-label-set-mnemonic-widget` is not called, then the first activatable ancestor of the `<gtk-label>` will be chosen as the mnemonic widget. For instance, if the label is inside a button or menu item, the button or menu item will automatically become the mnemonic widget and be activated by the mnemonic.

str The text of the label, with an underscore in front of the mnemonic character

ret the new `<gtk-label>`

`gtk-label-select-region` (*self* `<gtk-label>`) (*start_offset* `int`) (*end_offset* `int`) [Function]
`select-region` [Method]

Selects a range of characters in the label, if the label is selectable. See `gtk-label-set-selectable`. If the label is not selectable, this function has no effect. If *start-offset* or *end-offset* are -1, then the end of the label will be substituted.

label a `<gtk-label>`

start-offset start offset (in characters not bytes)

end-offset end offset (in characters not bytes)

`gtk-label-set-mnemonic-widget` (*self* `<gtk-label>`) (*widget* `<gtk-widget>`) [Function]
`set-mnemonic-widget` [Method]

If the label has been set so that it has an mnemonic key (using i.e. `gtk-label-set-markup-with-mnemonic`, `gtk-label-set-text-with-mnemonic`, `gtk-label-new-with-mnemonic` or the "use_underline" property) the label can be associated with a widget that is the target of the mnemonic. When the label is inside a widget (like a `<gtk-button>` or a `<gtk-notebook>` tab) it is automatically associated with the correct widget, but sometimes (i.e. when the target is a `<gtk-entry>` next to the label) you need to set it explicitly using this function.

The target widget will be accelerated by emitting "mnemonic_activate" on it. The default handler for this signal will activate the widget if there are no mnemonic collisions and toggle focus between the colliding widgets otherwise.

label a `<gtk-label>`

widget the target `<gtk-widget>`

`gtk-label-set-selectable` (*self* `<gtk-label>`) (*setting* `bool`) [Function]
`set-selectable` [Method]

Selectable labels allow the user to select text from the label, for copy-and-paste.

label a `<gtk-label>`

setting '`#t`' to allow selecting text in the label

`gtk-label-set-text-with-mnemonic` (*self* `<gtk-label>`) [Function]
(*str* `mchars`)
`set-text-with-mnemonic` [Method]

Sets the label's text from the string *str*. If characters in *str* are preceded by an underscore, they are underlined indicating that they represent a keyboard accelerator called a mnemonic. The mnemonic key can be used to activate another widget, chosen automatically, or explicitly using `gtk-label-set-mnemonic-widget`.

label a `<gtk-label>`

str a string

`gtk-label-get-attributes` (*self* `<gtk-label>`) [Function]
⇒ (*ret* `<pango-attr-list>`)
`get-attributes` [Method]

Gets the attribute list that was set on the label using `gtk-label-set-attributes`, if any. This function does not reflect attributes that come from the labels markup (see `gtk-label-set-markup`). If you want to get the effective attributes for the label, use pango_layout_get_attribute (gtk_label_get_layout (label)).

label a `<gtk-label>`

ret the attribute list, or '`#f`' if none was set.

`gtk-label-get-justify` (*self* `<gtk-label>`) [Function]
⇒ (*ret* `<gtk-justification>`)
`get-justify` [Method]

Returns the justification of the label. See `gtk-label-set-justify`.

label a `<gtk-label>`

ret `<gtk-justification>`

`gtk-label-get-ellipsize` (*self* `<gtk-label>`) [Function]
⇒ (*ret* `<pango-ellipsize-mode>`)
`get-ellipsize` [Method]

Returns the ellipsizing position of the label. See `gtk-label-set-ellipsize`.

label a `<gtk-label>`

ret `<pango-ellipsize-mode>`

Since 2.6

`gtk-label-get-width-chars` (*self* `<gtk-label>`) ⇒ (*ret* `int`) [Function]
`get-width-chars` [Method]
Retrieves the desired width of *label*, in characters. See `gtk-label-set-width-chars`.

label a `<gtk-label>`

ret the width of the label in characters.

Since 2.6

`gtk-label-get-max-width-chars` (*self* `<gtk-label>`) ⇒ (*ret* `int`) [Function]
`get-max-width-chars` [Method]
Retrieves the desired maximum width of *label*, in characters. See `gtk-label-set-width-chars`.

label a `<gtk-label>`

ret the maximum width of the label in characters.

Since 2.6

`gtk-label-get-label` (*self* `<gtk-label>`) ⇒ (*ret* `mchars`) [Function]
`get-label` [Method]
Fetches the text from a label widget including any embedded underlines indicating mnemonics and Pango markup. (See `gtk-label-get-text`).

label a `<gtk-label>`

ret the text of the label widget. This string is owned by the widget and must not be modified or freed.

`gtk-label-get-layout` (*self* `<gtk-label>`) [Function]
⇒ (*ret* `<pango-layout*>`)
`get-layout` [Method]
Gets the `<pango-layout>` used to display the label. The layout is useful to e.g. convert text positions to pixel positions, in combination with `gtk-label-get-layout-offsets`. The returned layout is owned by the label so need not be freed by the caller.

label a `<gtk-label>`

ret the `<pango-layout>` for this label

`gtk-label-get-line-wrap` (*self* `<gtk-label>`) ⇒ (*ret* `bool`) [Function]
`get-line-wrap` [Method]
Returns whether lines in the label are automatically wrapped. See `gtk-label-set-line-wrap`.

label a `<gtk-label>`

ret '`#t`' if the lines of the label are automatically wrapped.

`gtk-label-get-line-wrap-mode` (*self* `<gtk-label>`) [Function]
⇒ (*ret* `<pango-wrap-mode>`)
`get-line-wrap-mode` [Method]

Returns line wrap mode used by the label. See `gtk-label-set-line-wrap-mode`.

label a `<gtk-label>`

ret '`#t`' if the lines of the label are automatically wrapped.

Since 2.10

`gtk-label-get-mnemonic-widget` (*self* `<gtk-label>`) [Function]
⇒ (*ret* `<gtk-widget>`)
`get-mnemonic-widget` [Method]

Retrieves the target of the mnemonic (keyboard shortcut) of this label. See `gtk-label-set-mnemonic-widget`.

label a `<gtk-label>`

ret the target of the label's mnemonic, or '`#f`' if none has been set and the default algorithm will be used.

`gtk-label-get-selection-bounds` (*self* `<gtk-label>`) ⇒ (*ret* `bool`) [Function]
(*start* `int`) (*end* `int`)
`get-selection-bounds` [Method]

Gets the selected range of characters in the label, returning '`#t`' if there's a selection.

label a `<gtk-label>`

start return location for start of selection, as a character offset

end return location for end of selection, as a character offset

ret '`#t`' if selection is non-empty

`gtk-label-get-use-markup` (*self* `<gtk-label>`) ⇒ (*ret* `bool`) [Function]
`get-use-markup` [Method]

Returns whether the label's text is interpreted as marked up with the Pango text markup language. See `gtk-label-set-use-markup`.

label a `<gtk-label>`

ret '`#t`' if the label's text will be parsed for markup.

`gtk-label-get-use-underline` (*self* `<gtk-label>`) ⇒ (*ret* `bool`) [Function]
`get-use-underline` [Method]

Returns whether an embedded underline in the label indicates a mnemonic. See `gtk-label-set-use-underline`.

label a `<gtk-label>`

ret '`#t`' whether an embedded underline in the label indicates the mnemonic accelerator keys.

`gtk-label-get-single-line-mode` (*self* `<gtk-label>`) ⇒ (*ret* `bool`) [Function]
`get-single-line-mode` [Method]

Returns whether the label is in single line mode.

label a `<gtk-label>`

ret '`#t`' when the label is in single line mode.

Since 2.6

`gtk-label-get-angle` (*self* `<gtk-label>`) ⇒ (*ret* `double`) [Function]
`get-angle` [Method]

Gets the angle of rotation for the label. See gtk_label_set_angle.

label a `<gtk-label>`

ret the angle of rotation for the label

Since 2.6

`gtk-label-set-label` (*self* `<gtk-label>`) (*str* `mchars`) [Function]
`set-label` [Method]

Sets the text of the label. The label is interpreted as including embedded underlines and/or Pango markup depending on the values of label->use_underline and label->use_markup.

label a `<gtk-label>`

str the new text to set for the label

`gtk-label-set-use-markup` (*self* `<gtk-label>`) (*setting* `bool`) [Function]
`set-use-markup` [Method]

Sets whether the text of the label contains markup in Pango's text markup language. See `gtk-label-set-markup`.

label a `<gtk-label>`

setting '`#t`' if the label's text should be parsed for markup.

`gtk-label-set-use-underline` (*self* `<gtk-label>`) (*setting* `bool`) [Function]
`set-use-underline` [Method]

If true, an underline in the text indicates the next character should be used for the mnemonic accelerator key.

label a `<gtk-label>`

setting '`#t`' if underlines in the text indicate mnemonics

`gtk-label-set-single-line-mode` (*self* `<gtk-label>`) [Function]
(*single_line_mode* `bool`)
`set-single-line-mode` [Method]

Sets whether the label is in single line mode.

label a `<gtk-label>`

single-line-mode
'`#t`' if the label should be in single line mode

Since 2.6

`gtk-label-set-angle` (*self* `<gtk-label>`) (*angle* `double`) [Function]
`set-angle` [Method]

Sets the angle of rotation for the label. An angle of 90 reads from from bottom to top, an angle of 270, from top to bottom. The angle setting for the label is ignored if the label is selectable, wrapped, or ellipsized.

label a `<gtk-label>`

angle the angle that the baseline of the label makes with the horizontal, in degrees, measured counterclockwise

Since 2.6

12 GtkProgressBar

A widget which indicates progress visually

12.1 Overview

The <gtk-progress-bar> is typically used to display the progress of a long running operation. It provides a visual clue that processing is underway. The <gtk-progress-bar> can be used in two different modes: percentage mode and activity mode.

When an application can determine how much work needs to take place (e.g. read a fixed number of bytes from a file) and can monitor its progress, it can use the <gtk-progress-bar> in percentage mode and the user sees a growing bar indicating the percentage of the work that has been completed. In this mode, the application is required to call gtk-progress-bar-set-fraction periodically to update the progress bar.

When an application has no accurate way of knowing the amount of work to do, it can use the <gtk-progress-bar> in activity mode, which shows activity by a block moving back and forth within the progress area. In this mode, the application is required to call gtk-progress-bar-pulse perodically to update the progress bar.

There is quite a bit of flexibility provided to control the appearance of the <gtk-progress-bar>. Functions are provided to control the orientation of the bar, optional text can be displayed along with the bar, and the step size used in activity mode can be set.

The <gtk-progress-bar>/<gtk-progress> API in GTK 1.2 was bloated, needlessly complex and hard to use properly. Therefore <gtk-progress> has been deprecated completely and the <gtk-progress-bar> API has been reduced to the following 10 functions: gtk-progress-bar-new, gtk-progress-bar-pulse, gtk-progress-bar-set-text, gtk-progress-bar-set-fraction, gtk-progress-bar-set-pulse-step, gtk-progress-bar-set-orientation, gtk-progress-bar-get-text, gtk-progress-bar-get-fraction, gtk-progress-bar-get-pulse-step, gtk-progress-bar-get-orientation. These have been grouped at the beginning of this section, followed by a large chunk of deprecated 1.2 compatibility functions.

12.2 Usage

<gtk-progress-bar> [Class]

This <gobject> class defines the following properties:

`fraction` The fraction of total work that has been completed

`pulse-step`
The fraction of total progress to move the bouncing block when pulsed

`orientation`
Orientation and growth direction of the progress bar

`text` Text to be displayed in the progress bar

`ellipsize`
The preferred place to ellipsize the string, if the progress bar does not have enough room to display the entire string, if at all.

`adjustment`
: The GtkAdjustment connected to the progress bar (Deprecated)

`bar-style`
: Specifies the visual style of the bar in percentage mode (Deprecated)

`activity-step`
: The increment used for each iteration in activity mode (Deprecated)

`activity-blocks`
: The number of blocks which can fit in the progress bar area in activity mode (Deprecated)

`discrete-blocks`
: The number of discrete blocks in a progress bar (when shown in the discrete style)

`gtk-progress-bar-new` ⇒ (*ret* `<gtk-widget>`) [Function]

Creates a new `<gtk-progress-bar>`.

ret a `<gtk-progress-bar>`.

`gtk-progress-bar-pulse` (*self* `<gtk-progress-bar>`) [Function]
`pulse` [Method]

Indicates that some progress is made, but you don't know how much. Causes the progress bar to enter "activity mode," where a block bounces back and forth. Each call to `gtk-progress-bar-pulse` causes the block to move by a little bit (the amount of movement per pulse is determined by `gtk-progress-bar-set-pulse-step`).

pbar a `<gtk-progress-bar>`

`gtk-progress-bar-set-text` (*self* `<gtk-progress-bar>`) [Function]
(*text* `mchars`)
`set-text` [Method]

Causes the given *text* to appear superimposed on the progress bar.

pbar a `<gtk-progress-bar>`

text a UTF-8 string, or '`#f`'

`gtk-progress-bar-set-fraction` (*self* `<gtk-progress-bar>`) [Function]
(*fraction* `double`)
`set-fraction` [Method]

Causes the progress bar to "fill in" the given fraction of the bar. The fraction should be between 0.0 and 1.0, inclusive.

pbar a `<gtk-progress-bar>`

fraction fraction of the task that's been completed

`gtk-progress-bar-set-pulse-step` (*self* `<gtk-progress-bar>`) [Function]
(*fraction* `double`)
`set-pulse-step` [Method]

Sets the fraction of total progress bar length to move the bouncing block for each call to `gtk-progress-bar-pulse`.

pbar a `<gtk-progress-bar>`

fraction fraction between 0.0 and 1.0

`gtk-progress-bar-set-orientation` (*self* `<gtk-progress-bar>`) [Function]
(*orientation* `<gtk-progress-bar-orientation>`)
`set-orientation` [Method]

Causes the progress bar to switch to a different orientation (left-to-right, right-to-left, top-to-bottom, or bottom-to-top).

pbar a `<gtk-progress-bar>`

orientation
orientation of the progress bar

`gtk-progress-bar-set-ellipsize` (*self* `<gtk-progress-bar>`) [Function]
(*mode* `<pango-ellipsize-mode>`)
`set-ellipsize` [Method]

Sets the mode used to ellipsize (add an ellipsis: `"..."`) the text if there is not enough space to render the entire string.

pbar a `<gtk-progress-bar>`

mode a `<pango-ellipsize-mode>`

Since 2.6

`gtk-progress-bar-get-text` (*self* `<gtk-progress-bar>`) [Function]
⇒ (*ret* `mchars`)
`get-text` [Method]

Retrieves the text displayed superimposed on the progress bar, if any, otherwise '`#f`'. The return value is a reference to the text, not a copy of it, so will become invalid if you change the text in the progress bar.

pbar a `<gtk-progress-bar>`

ret text, or '`#f`'; this string is owned by the widget and should not be modified or freed.

`gtk-progress-bar-get-fraction` (*self* `<gtk-progress-bar>`) [Function]
⇒ (*ret* `double`)
`get-fraction` [Method]

Returns the current fraction of the task that's been completed.

pbar a `<gtk-progress-bar>`

ret a fraction from 0.0 to 1.0

`gtk-progress-bar-get-pulse-step` (*self* `<gtk-progress-bar>`) [Function]
⇒ (*ret* `double`)
`get-pulse-step` [Method]

Retrieves the pulse step set with `gtk-progress-bar-set-pulse-step`

pbar a `<gtk-progress-bar>`

ret a fraction from 0.0 to 1.0

`gtk-progress-bar-get-ellipsize` (*self* `<gtk-progress-bar>`) [Function]
⇒ (*ret* `<pango-ellipsize-mode>`)

`get-ellipsize` [Method]

Returns the ellipsizing position of the progressbar. See `gtk-progress-bar-set-ellipsize`.

pbar a `<gtk-progress-bar>`

ret `<pango-ellipsize-mode>`

Since 2.6

13 GtkStatusbar

Report messages of minor importance to the user

13.1 Overview

A <gtk-statusbar> is usually placed along the bottom of an application's main <gtk-window>. It may provide a regular commentary of the application's status (as is usually the case in a web browser, for example), or may be used to simply output a message when the status changes, (when an upload is complete in an FTP client, for example). It may also have a resize grip (a triangular area in the lower right corner) which can be clicked on to resize the window containing the statusbar.

Status bars in Gtk+ maintain a stack of messages. The message at the top of the each bar's stack is the one that will currently be displayed.

Any messages added to a statusbar's stack must specify a *context_id* that is used to uniquely identify the source of a message. This context_id can be generated by gtk-statusbar-get-context-id, given a message and the statusbar that it will be added to. Note that messages are stored in a stack, and when choosing which message to display, the stack structure is adhered to, regardless of the context identifier of a message.

Status bars are created using gtk-statusbar-new.

Messages are added to the bar's stack with gtk-statusbar-push.

The message at the top of the stack can be removed using gtk-statusbar-pop. A message can be removed from anywhere in the stack if its message_id was recorded at the time it was added. This is done using gtk-statusbar-remove.

13.2 Usage

<gtk-statusbar> [Class]
This <gobject> class defines the following properties:

has-resize-grip
Whether the statusbar has a grip for resizing the toplevel

text-pushed (*arg0* <guint>) (*arg1* <gchararray>) [Signal on <gtk-statusbar>]
Is emitted whenever a new message gets pushed onto a statusbar's stack.

text-popped (*arg0* <guint>) (*arg1* <gchararray>) [Signal on <gtk-statusbar>]
Is emitted whenever a new message is popped off a statusbar's stack.

gtk-statusbar-new ⇒ (*ret* <gtk-widget>) [Function]
Creates a new <gtk-statusbar> ready for messages.

ret the new <gtk-statusbar>.

gtk-statusbar-get-context-id (*self* <gtk-statusbar>) [Function]
(*context_description* mchars) ⇒ (*ret* unsigned-int)
get-context-id [Method]
Returns a new context identifier, given a description of the actual context.

statusbar a `<gtk-statusbar>`.

context-description
textual description of what context the new message is being used in.

ret an integer id.

`gtk-statusbar-push` (*self* `<gtk-statusbar>`) [Function]
(*context_id* `unsigned-int`) (*text* `mchars`) ⇒ (*ret* `unsigned-int`)

`push` [Method]

Pushes a new message onto a statusbar's stack.

statusbar a `<gtk-statusbar>`.

context-id the message's context id, as returned by `gtk-statusbar-get-context-id`.

text the message to add to the statusbar.

ret the message's new message id for use with `gtk-statusbar-remove`.

`gtk-statusbar-pop` (*self* `<gtk-statusbar>`) [Function]
(*context_id* `unsigned-int`)

`pop` [Method]

Removes the message at the top of a `<gtk-status-bar>`'s stack.

statusbar a `<gtk-status-bar>`.

context-id a context identifier.

`gtk-statusbar-remove` (*self* `<gtk-statusbar>`) [Function]
(*context_id* `unsigned-int`) (*message_id* `unsigned-int`)

`remove` [Method]

Forces the removal of a message from a statusbar's stack. The exact *context-id* and *message-id* must be specified.

statusbar a `<gtk-status-bar>`.

context-id a context identifier.

message-id
a message identifier, as returned by `gtk-statusbar-push`.

`gtk-statusbar-set-has-resize-grip` (*self* `<gtk-statusbar>`) [Function]
(*setting* `bool`)

`set-has-resize-grip` [Method]

Sets whether the statusbar has a resize grip. '`#t`' by default.

statusbar a `<gtk-status-bar>`.

setting '`#t`' to have a resize grip.

`gtk-statusbar-get-has-resize-grip` (*self* `<gtk-statusbar>`) [Function]
⇒ (*ret* `bool`)

`get-has-resize-grip` [Method]

Returns whether the statusbar has a resize grip.

statusbar a `<gtk-status-bar>`.

ret '`#t`' if the statusbar has a resize grip.

14 GtkStatusIcon

Display an icon in the system tray

14.1 Overview

The "system tray" or notification area is normally used for transient icons that indicate some special state. For example, a system tray icon might appear to tell the user that they have new mail, or have an incoming instant message, or something along those lines. The basic idea is that creating an icon in the notification area is less annoying than popping up a dialog.

A `<gtk-status-icon>` object can be used to display an icon in a "system tray". The icon can have a tooltip, and the user can interact with it by activating it or popping up a context menu. Critical information should not solely be displayed in a `<gtk-status-icon>`, since it may not be visible (e.g. when the user doesn't have a notification area on his panel). This can be checked with `gtk-status-icon-is-embedded`.

On X11, the implementation follows the freedesktop.org "System Tray" specification. Implementations of the "tray" side of this specification can be found e.g. in the GNOME and KDE panel applications.

Note that a GtkStatusIcon is *not* a widget, but just a `<gobject>`. Making it a widget would be impractical, since the system tray on Win32 doesn't allow to embed arbitrary widgets.

14.2 Usage

`<gtk-status-icon>` [Class]

This `<gobject>` class defines the following properties:

`pixbuf` A GdkPixbuf to display

`file` Filename to load and display

`stock` Stock ID for a stock image to display

`icon-name`
The name of the icon from the icon theme

`storage-type`
The representation being used for image data

`size` The size of the icon

`screen` The screen where this status icon will be displayed

`visible` Whether or not the status icon is visible

`orientation`
The orientation of the tray

`embedded` Whether or not the status icon is embedded

`blinking` Whether or not the status icon is blinking

`size-changed` (*arg0* `<gint>`) ⇒ `<gboolean>` [Signal on `<gtk-status-icon>`]

Gets emitted when the size available for the image changes, e.g. because the notification area got resized.

Since 2.10

`popup-menu` (*arg0* `<guint>`) (*arg1* `<guint>`) [Signal on `<gtk-status-icon>`]

Gets emitted when the user brings up the context menu of the status icon. Whether status icons can have context menus and how these are activated is platform-dependent.

The *button* and *activate-timeout* parameters should be passed as the last to arguments to `gtk-menu-popup`.

Since 2.10

`activate` [Signal on `<gtk-status-icon>`]

Gets emitted when the user activates the status icon. If and how status icons can activated is platform-dependent.

Since 2.10

`gtk-status-icon-new` ⇒ (*ret* `<gtk-status-icon>`) [Function]

Creates an empty status icon object.

ret a new `<gtk-status-icon>`

Since 2.10

`gtk-status-icon-new-from-pixbuf` (*pixbuf* `<gdk-pixbuf>`) [Function]
⇒ (*ret* `<gtk-status-icon>`)

Creates a status icon displaying *pixbuf*.

The image will be scaled down to fit in the available space in the notification area, if necessary.

pixbuf a `<gdk-pixbuf>`

ret a new `<gtk-status-icon>`

Since 2.10

`gtk-status-icon-new-from-file` (*filename* `mchars`) [Function]
⇒ (*ret* `<gtk-status-icon>`)

Creates a status icon displaying the file *filename*.

The image will be scaled down to fit in the available space in the notification area, if necessary.

filename a filename

ret a new `<gtk-status-icon>`

Since 2.10

`gtk-status-icon-new-from-stock` (*stock_id* `mchars`) [Function]
⇒ (*ret* `<gtk-status-icon>`)

Creates a status icon displaying a stock icon. Sample stock icon names are `<gtk-stock-open>`, `<gtk-stock-quit>`. You can register your own stock icon names, see `gtk-icon-factory-add-default` and `gtk-icon-factory-add`.

stock-id a stock icon id

ret a new `<gtk-status-icon>`

Since 2.10

`gtk-status-icon-new-from-icon-name` (*icon_name* `mchars`) [Function]
⇒ (*ret* `<gtk-status-icon>`)

Creates a status icon displaying an icon from the current icon theme. If the current icon theme is changed, the icon will be updated appropriately.

icon-name an icon name

ret a new `<gtk-status-icon>`

Since 2.10

`gtk-status-icon-set-from-pixbuf` (*self* `<gtk-status-icon>`) [Function]
(*pixbuf* `<gdk-pixbuf>`)
`set-from-pixbuf` [Method]

Makes *status-icon* display *pixbuf*. See `gtk-status-icon-new-from-pixbuf` for details.

status-icon
a `<gtk-status-icon>`

pixbuf a `<gdk-pixbuf>` or '`#f`'

Since 2.10

`gtk-status-icon-set-from-file` (*self* `<gtk-status-icon>`) [Function]
(*filename* `mchars`)
`set-from-file` [Method]

Makes *status-icon* display the file *filename*. See `gtk-status-icon-new-from-file` for details.

status-icon
a `<gtk-status-icon>`

filename a filename

Since 2.10

`gtk-status-icon-set-from-stock` (*self* `<gtk-status-icon>`) [Function]
(*stock_id* `mchars`)
`set-from-stock` [Method]

Makes *status-icon* display the stock icon with the id *stock-id*. See `gtk-status-icon-new-from-stock` for details.

status-icon
: a <gtk-status-icon>

stock-id
: a stock icon id

Since 2.10

gtk-status-icon-set-from-icon-name (*self* <gtk-status-icon>) (*icon_name* mchars) [Function]

set-from-icon-name [Method]

Makes *status-icon* display the icon named *icon-name* from the current icon theme. See gtk-status-icon-new-from-icon-name for details.

status-icon
: a <gtk-status-icon>

icon-name
: an icon name

Since 2.10

gtk-status-icon-get-storage-type (*self* <gtk-status-icon>) ⇒ (*ret* <gtk-image-type>) [Function]

get-storage-type [Method]

Gets the type of representation being used by the <gtk-status-icon> to store image data. If the <gtk-status-icon> has no image data, the return value will be 'GTK_IMAGE_EMPTY'.

status-icon
: a <gtk-status-icon>

ret
: the image representation being used

Since 2.10

gtk-status-icon-get-pixbuf (*self* <gtk-status-icon>) ⇒ (*ret* <gdk-pixbuf>) [Function]

get-pixbuf [Method]

Gets the <gdk-pixbuf> being displayed by the <gtk-status-icon>. The storage type of the status icon must be 'GTK_IMAGE_EMPTY' or 'GTK_IMAGE_PIXBUF' (see gtk-status-icon-get-storage-type). The caller of this function does not own a reference to the returned pixbuf.

status-icon
: a <gtk-status-icon>

ret
: the displayed pixbuf, or '#f' if the image is empty.

Since 2.10

gtk-status-icon-get-stock (*self* <gtk-status-icon>) ⇒ (*ret* mchars) [Function]

get-stock [Method]

Gets the id of the stock icon being displayed by the <gtk-status-icon>. The storage type of the status icon must be 'GTK_IMAGE_EMPTY' or 'GTK_IMAGE_STOCK' (see gtk-status-icon-get-storage-type). The returned string is owned by the <gtk-status-icon> and should not be freed or modified.

status-icon
a `<gtk-status-icon>`

ret stock id of the displayed stock icon, or '`#f`' if the image is empty.

Since 2.10

`gtk-status-icon-get-icon-name` (*self* `<gtk-status-icon>`) ⇒ (*ret* `mchars`) [Function]
`get-icon-name` [Method]

Gets the name of the icon being displayed by the `<gtk-status-icon>`. The storage type of the status icon must be '`GTK_IMAGE_EMPTY`' or '`GTK_IMAGE_ICON_NAME`' (see `gtk-status-icon-get-storage-type`). The returned string is owned by the `<gtk-status-icon>` and should not be freed or modified.

status-icon
a `<gtk-status-icon>`

ret name of the displayed icon, or '`#f`' if the image is empty.

Since 2.10

`gtk-status-icon-get-size` (*self* `<gtk-status-icon>`) ⇒ (*ret* `int`) [Function]
`get-size` [Method]

Gets the size in pixels that is available for the image. Stock icons and named icons adapt their size automatically if the size of the notification area changes. For other storage types, the size-changed signal can be used to react to size changes.

status-icon
a `<gtk-status-icon>`

ret the size that is available for the image

Since 2.10

`gtk-status-icon-set-tooltip` (*self* `<gtk-status-icon>`) (*tooltip_text* `mchars`) [Function]
`set-tooltip` [Method]

Sets the tooltip of the status icon.

status-icon
a `<gtk-status-icon>`

tooltip-text
the tooltip text, or '`#f`'

Since 2.10

`gtk-status-icon-set-visible` (*self* `<gtk-status-icon>`) (*visible* `bool`) [Function]
`set-visible` [Method]

Shows or hides a status icon.

status-icon
a `<gtk-status-icon>`

visible '`#t`' to show the status icon, '`#f`' to hide it

Since 2.10

`gtk-status-icon-get-visible` (*self* `<gtk-status-icon>`) [Function]
⇒ (*ret* `bool`)

`get-visible` [Method]

Returns whether the status icon is visible or not. Note that being visible does not guarantee that the user can actually see the icon, see also `gtk-status-icon-is-embedded`.

status-icon
a `<gtk-status-icon>`

ret '`#t`' if the status icon is visible

Since 2.10

`gtk-status-icon-set-blinking` (*self* `<gtk-status-icon>`) [Function]
(*blinking* `bool`)

`set-blinking` [Method]

Makes the status icon start or stop blinking. Note that blinking user interface elements may be problematic for some users, and thus may be turned off, in which case this setting has no effect.

status-icon
a `<gtk-status-icon>`

blinking '`#t`' to turn blinking on, '`#f`' to turn it off

Since 2.10

`gtk-status-icon-get-blinking` (*self* `<gtk-status-icon>`) [Function]
⇒ (*ret* `bool`)

`get-blinking` [Method]

Returns whether the icon is blinking, see `gtk-status-icon-set-blinking`.

status-icon
a `<gtk-status-icon>`

ret '`#t`' if the icon is blinking

Since 2.10

`gtk-status-icon-is-embedded` (*self* `<gtk-status-icon>`) [Function]
⇒ (*ret* `bool`)

`is-embedded` [Method]

Returns whether the status icon is embedded in a notification area.

status-icon
a `<gtk-status-icon>`

ret '`#t`' if the status icon is embedded in a notification area.

Since 2.10

`gtk-status-icon-position-menu` (*menu* `<gtk-menu>`) [Function]
(*user_data* `<gpointer>`) ⇒ (*x* `int`) (*y* `int`) (*push_in* `bool`)

Menu positioning function to use with `gtk-menu-popup` to position *menu* aligned to the status icon *user-data*.

menu the `<gtk-menu>`

x return location for the x position

y return location for the y position

push-in return location for whether the menu should be pushed in to be completely inside the screen instead of just clamped to the size to the screen.

user-data the status icon to position the menu on

Since 2.10

`gtk-status-icon-get-geometry` (*self* `<gtk-status-icon>`) [Function]
(*screen* `<gdk-screen**>`) (*area* `<gdk-rectangle>`)
(*orientation* `<gtk-orientation*>`) ⇒ (*ret* `bool`)

`get-geometry` [Method]

Obtains information about the location of the status icon on screen. This information can be used to e.g. position popups like notification bubbles.

See `gtk-status-icon-position-menu` for a more convenient alternative for positioning menus.

Note that some platforms do not allow GTK+ to provide this information, and even on platforms that do allow it, the information is not reliable unless the status icon is embedded in a notification area, see `gtk-status-icon-is-embedded`.

status-icon
a `<gtk-status-icon>`

screen return location for the screen, or '`#f`' if the information is not needed

area return location for the area occupied by the status icon, or '`#f`'

orientation
return location for the orientation of the panel in which the status icon is embedded, or '`#f`'. A panel at the top or bottom of the screen is horizontal, a panel at the left or right is vertical.

ret '`#t`' if the location information has been filled in

Since 2.10

15 GtkButton

A widget that creates a signal when clicked on

15.1 Overview

The <gtk-button> widget is generally used to attach a function to that is called when the button is pressed. The various signals and how to use them are outlined below.

The <gtk-button> widget can hold any valid child widget. That is it can hold most any other standard <gtk-widget>. The most commonly used child is the <gtk-label>.

15.2 Usage

<gtk-button> [Class]

This <gobject> class defines the following properties:

label Text of the label widget inside the button, if the button contains a label widget

image Child widget to appear next to the button text

relief The border relief style

use-underline
If set, an underline in the text indicates the next character should be used for the mnemonic accelerator key

use-stock
If set, the label is used to pick a stock item instead of being displayed

focus-on-click
Whether the button grabs focus when it is clicked with the mouse

xalign Horizontal position of child in available space. 0.0 is left aligned, 1.0 is right aligned

yalign Vertical position of child in available space. 0.0 is top aligned, 1.0 is bottom aligned

image-position
The position of the image relative to the text

activate [Signal on <gtk-button>]

The "activate" signal on GtkButton is an action signal and emitting it causes the button to animate press then release. Applications should never connect to this signal, but use the "clicked" signal.

pressed [Signal on <gtk-button>]

Emitted when the button is pressed.

deprecated: Use the GtkWidget::button-press-event signal.

released [Signal on <gtk-button>]

Emitted when the button is released.

deprecated: Use the GtkWidget::button-release-event signal.

`clicked` [Signal on `<gtk-button>`]
Emitted when the button has been activated (pressed and released).

`enter` [Signal on `<gtk-button>`]
Emitted when the pointer enters the button.

deprecated: Use the GtkWidget::enter-notify-event signal.

`leave` [Signal on `<gtk-button>`]
Emitted when the pointer leaves the button.

deprecated: Use the GtkWidget::leave-notify-event signal.

`gtk-button-new` ⇒ (*ret* `<gtk-widget>`) [Function]
Creates a new `<gtk-button>` widget. To add a child widget to the button, use `gtk-container-add`.

ret The newly created `<gtk-button>` widget.

`gtk-button-new-with-label` (*label* `mchars`) ⇒ (*ret* `<gtk-widget>`) [Function]
Creates a `<gtk-button>` widget with a `<gtk-label>` child containing the given text.

label The text you want the `<gtk-label>` to hold.

ret The newly created `<gtk-button>` widget.

`gtk-button-new-with-mnemonic` (*label* `mchars`) [Function]
⇒ (*ret* `<gtk-widget>`)
Creates a new `<gtk-button>` containing a label. If characters in *label* are preceded by an underscore, they are underlined. If you need a literal underscore character in a label, use '__' (two underscores). The first underlined character represents a keyboard accelerator called a mnemonic. Pressing Alt and that key activates the button.

label The text of the button, with an underscore in front of the mnemonic character

ret a new `<gtk-button>`

`gtk-button-new-from-stock` (*stock_id* `mchars`) [Function]
⇒ (*ret* `<gtk-widget>`)
Creates a new `<gtk-button>` containing the image and text from a stock item. Some stock ids have preprocessor macros like `<gtk-stock-ok>` and `<gtk-stock-apply>`.

If *stock-id* is unknown, then it will be treated as a mnemonic label (as for `gtk-button-new-with-mnemonic`).

stock-id the name of the stock item

ret a new `<gtk-button>`

`gtk-button-pressed` (*self* `<gtk-button>`) [Function]
`pressed` [Method]
Emits a `<gtk-button::pressed>` signal to the given `<gtk-button>`.

button The `<gtk-button>` you want to send the signal to.

`gtk-button-released` (*self* `<gtk-button>`) [Function]
`released` [Method]

Emits a `<gtk-button::released>` signal to the given `<gtk-button>`.

button The `<gtk-button>` you want to send the signal to.

`gtk-button-clicked` (*self* `<gtk-button>`) [Function]
`clicked` [Method]

Emits a `<gtk-button::clicked>` signal to the given `<gtk-button>`.

button The `<gtk-button>` you want to send the signal to.

`gtk-button-enter` (*self* `<gtk-button>`) [Function]
`enter` [Method]

Emits a `<gtk-button::enter>` signal to the given `<gtk-button>`.

button The `<gtk-button>` you want to send the signal to.

`gtk-button-leave` (*self* `<gtk-button>`) [Function]
`leave` [Method]

Emits a `<gtk-button::leave>` signal to the given `<gtk-button>`.

button The `<gtk-button>` you want to send the signal to.

`gtk-button-set-relief` (*self* `<gtk-button>`) [Function]
(*newstyle* `<gtk-relief-style>`)
`set-relief` [Method]

Sets the relief style of the edges of the given `<gtk-button>` widget. Three styles exist, GTK_RELIEF_NORMAL, GTK_RELIEF_HALF, GTK_RELIEF_NONE. The default style is, as one can guess, GTK_RELIEF_NORMAL.

button The `<gtk-button>` you want to set relief styles of.

newstyle The GtkReliefStyle as described above.

`gtk-button-get-relief` (*self* `<gtk-button>`) [Function]
⇒ (*ret* `<gtk-relief-style>`)
`get-relief` [Method]

Returns the current relief style of the given `<gtk-button>`.

button The `<gtk-button>` you want the `<gtk-relief-style>` from.

ret The current `<gtk-relief-style>`

`gtk-button-get-label` (*self* `<gtk-button>`) ⇒ (*ret* `mchars`) [Function]
`get-label` [Method]

Fetches the text from the label of the button, as set by `gtk-button-set-label`. If the label text has not been set the return value will be '`#f`'. This will be the case if you create an empty button with `gtk-button-new` to use as a container.

button a `<gtk-button>`

ret The text of the label widget. This string is owned by the widget and must not be modified or freed.

`gtk-button-set-label` (*self* `<gtk-button>`) (*label* `mchars`) [Function]
`set-label` [Method]

Sets the text of the label of the button to *str*. This text is also used to select the stock item if `gtk-button-set-use-stock` is used.

This will also clear any previously set labels.

button a `<gtk-button>`

label a string

`gtk-button-get-use-stock` (*self* `<gtk-button>`) ⇒ (*ret* `bool`) [Function]
`get-use-stock` [Method]

Returns whether the button label is a stock item.

button a `<gtk-button>`

ret '`#t`' if the button label is used to select a stock item instead of being used directly as the label text.

`gtk-button-set-use-stock` (*self* `<gtk-button>`) (*use_stock* `bool`) [Function]
`set-use-stock` [Method]

If true, the label set on the button is used as a stock id to select the stock item for the button.

button a `<gtk-button>`

use-stock '`#t`' if the button should use a stock item

`gtk-button-get-use-underline` (*self* `<gtk-button>`) ⇒ (*ret* `bool`) [Function]
`get-use-underline` [Method]

Returns whether an embedded underline in the button label indicates a mnemonic. See `gtk-button-set-use-underline`.

button a `<gtk-button>`

ret '`#t`' if an embedded underline in the button label indicates the mnemonic accelerator keys.

`gtk-button-set-use-underline` (*self* `<gtk-button>`) [Function]
(*use_underline* `bool`)
`set-use-underline` [Method]

If true, an underline in the text of the button label indicates the next character should be used for the mnemonic accelerator key.

button a `<gtk-button>`

use-underline
'`#t`' if underlines in the text indicate mnemonics

`gtk-button-set-focus-on-click` (*self* `<gtk-button>`) [Function]
(*focus_on_click* `bool`)
`set-focus-on-click` [Method]

Sets whether the button will grab focus when it is clicked with the mouse. Making mouse clicks not grab focus is useful in places like toolbars where you don't want the keyboard focus removed from the main area of the application.

button a `<gtk-button>`

focus-on-click
whether the button grabs focus when clicked with the mouse

Since 2.4

`gtk-button-get-focus-on-click` (*self* `<gtk-button>`) ⇒ (*ret* `bool`) [Function]
`get-focus-on-click` [Method]
Returns whether the button grabs focus when it is clicked with the mouse. See `gtk-button-set-focus-on-click`.

button a `<gtk-button>`

ret '`#t`' if the button grabs focus when it is clicked with the mouse.

Since 2.4

`gtk-button-set-alignment` (*self* `<gtk-button>`) (*xalign* `float`) (*yalign* `float`) [Function]
`set-alignment` [Method]
Sets the alignment of the child. This property has no effect unless the child is a `<gtk-misc>` or a `<gtk-aligment>`.

button a `<gtk-button>`

xalign the horizontal position of the child, 0.0 is left aligned, 1.0 is right aligned

yalign the vertical position of the child, 0.0 is top aligned, 1.0 is bottom aligned

Since 2.4

`gtk-button-get-alignment` (*self* `<gtk-button>`) ⇒ (*xalign* `float`) (*yalign* `float`) [Function]
`get-alignment` [Method]
Gets the alignment of the child in the button.

button a `<gtk-button>`

xalign return location for horizontal alignment

yalign return location for vertical alignment

Since 2.4

`gtk-button-set-image` (*self* `<gtk-button>`) (*image* `<gtk-widget>`) [Function]
`set-image` [Method]
Set the image of *button* to the given widget. Note that it depends on the gtk-button-images setting whether the image will be displayed or not, you don't have to call `gtk-widget-show` on *image* yourself.

button a `<gtk-button>`

image a widget to set as the image for the button

Since 2.6

`gtk-button-get-image` (*self* `<gtk-button>`) ⇒ (*ret* `<gtk-widget>`) [Function]
`get-image` [Method]

Gets the widget that is currenty set as the image of *button*. This may have been explicitly set by `gtk-button-set-image` or constructed by `gtk-button-new-from-stock`.

button a `<gtk-button>`

ret a `<gtk-widget>` or '`#f`' in case there is no image

Since 2.6

`gtk-button-set-image-position` (*self* `<gtk-button>`) [Function]
(*position* `<gtk-position-type>`)
`set-image-position` [Method]

Sets the position of the image relative to the text inside the button.

button a `<gtk-button>`

position the position

Since 2.10

`gtk-button-get-image-position` (*self* `<gtk-button>`) [Function]
⇒ (*ret* `<gtk-position-type>`)
`get-image-position` [Method]

Gets the position of the image relative to the text inside the button.

button a `<gtk-button>`

ret the position

Since 2.10

16 GtkCheckButton

Create widgets with a discrete toggle button

16.1 Overview

A <gtk-check-button> places a discrete <gtk-toggle-button> next to a widget, (usually a <gtk-label>). See the section on <gtk-toggle-button> widgets for more information about toggle/check buttons.

The important signal ('toggled') is also inherited from <gtk-toggle-button>.

16.2 Usage

<gtk-check-button> [Class]

This <gobject> class defines no properties, other than those defined by its superclasses.

gtk-check-button-new ⇒ (*ret* <gtk-widget>) [Function]

Creates a new <gtk-check-button>.

ret a <gtk-widget>.

gtk-check-button-new-with-label (*label* mchars) [Function]
⇒ (*ret* <gtk-widget>)

Creates a new <gtk-check-button> with a <gtk-label> to the right of it.

label the text for the check button.

ret a <gtk-widget>.

gtk-check-button-new-with-mnemonic (*label* mchars) [Function]
⇒ (*ret* <gtk-widget>)

Creates a new <gtk-check-button> containing a label. The label will be created using gtk-label-new-with-mnemonic, so underscores in *label* indicate the mnemonic for the check button.

label The text of the button, with an underscore in front of the mnemonic character

ret a new <gtk-check-button>

17 GtkRadioButton

A choice from multiple check buttons

17.1 Overview

A single radio button performs the same basic function as a <gtk-check-button>, as its position in the object hierarchy reflects. It is only when multiple radio buttons are grouped together that they become a different user interface component in their own right.

Every radio button is a member of some group of radio buttons. When one is selected, all other radio buttons in the same group are deselected. A <gtk-radio-button> is one way of giving the user a choice from many options.

Radio button widgets are created with gtk-radio-button-new, passing NULL as the argument if this is the first radio button in a group. In subsequent calls, the group you wish to add this button to should be passed as an argument. Optionally, gtk-radio-button-new-with-label can be used if you want a text label on the radio button.

Alternatively, when adding widgets to an existing group of radio buttons, use gtk-radio-button-new-from-widget with a <gtk-radio-button> that already has a group assigned to it. The convenience function gtk-radio-button-new-with-label-from-widget is also provided.

To retrieve the group a <gtk-radio-button> is assigned to, use gtk-radio-button-get-group.

To remove a <gtk-radio-button> from one group and make it part of a new one, use gtk-radio-button-set-group.

The group list does not need to be freed, as each <gtk-radio-button> will remove itself and its list item when it is destroyed.

```
void create_radio_buttons (void) {

   GtkWidget *window, *radio1, *radio2, *box, *entry;
   window = gtk_window_new (GTK_WINDOW_TOPLEVEL);
   box = gtk_vbox_new (TRUE, 2);

   /* Create a radio button with a GtkEntry widget */
   radio1 = gtk_radio_button_new (NULL);
   entry = gtk_entry_new ();
   gtk_container_add (GTK_CONTAINER (radio1), entry);


   /* Create a radio button with a label */
   radio2 = gtk_radio_button_new_with_label_from_widget (GTK_RADIO_BUTTON (radio1),
"I'm the second radio button.");

   /* Pack them into a box, then show all the widgets */
   gtk_box_pack_start (GTK_BOX (box), radio1, TRUE, TRUE, 2);
```

```
    gtk_box_pack_start (GTK_BOX (box), radio2, TRUE, TRUE, 2);
    gtk_container_add (GTK_CONTAINER (window), box);
    gtk_widget_show_all (window);
    return;
}
```

When an unselected button in the group is clicked the clicked button receives the "toggled" signal, as does the previously selected button. Inside the "toggled" handler, `gtk-toggle-button-get-active` can be used to determine if the button has been selected or deselected.

17.2 Usage

`<gtk-radio-button>` [Class]

This `<gobject>` class defines the following properties:

`group` The radio button whose group this widget belongs to.

`group-changed` [Signal on `<gtk-radio-button>`]

Emitted when the group of radio buttons that a radio button belongs to changes. This is emitted when a radio button switches from being alone to being part of a group of 2 or more buttons, or vice-versa, and when a buttton is moved from one group of 2 or more buttons to a different one, but not when the composition of the group that a button belongs to changes.

Since 2.4

`gtk-radio-button-new` (*group* `<gtk-radio-group*>`) [Function]
⇒ (*ret* `<gtk-widget>`)

Creates a new `<gtk-radio-button>`. To be of any practical value, a widget should then be packed into the radio button.

group an existing radio button group, or '`#f`' if you are creating a new group.

ret a new radio button.

`gtk-radio-button-new-from-widget` (*group* `<gtk-radio-button>`) [Function]
⇒ (*ret* `<gtk-widget>`)

Creates a new `<gtk-radio-button>`, adding it to the same group as *group*. As with `gtk-radio-button-new`, a widget should be packed into the radio button.

group an existing `<gtk-radio-button>`.

ret a new radio button.

`gtk-radio-button-new-with-label` (*group* `<gtk-radio-group*>`) [Function]
(*label* `mchars`) ⇒ (*ret* `<gtk-widget>`)

Creates a new `<gtk-radio-button>` with a text label.

group an existing radio button group, or '`#f`' if you are creating a new group.

label the text label to display next to the radio button.

ret a new radio button.

`gtk-radio-button-new-with-mnemonic` [Function]
(*group* `<gtk-radio-group*>`) (*label* `mchars`) ⇒ (*ret* `<gtk-widget>`)

Creates a new `<gtk-radio-button>` containing a label, adding it to the same group as *group*. The label will be created using `gtk-label-new-with-mnemonic`, so underscores in *label* indicate the mnemonic for the button.

group the radio button group

label the text of the button, with an underscore in front of the mnemonic character

ret a new `<gtk-radio-button>`

`gtk-radio-button-set-group` (*self* `<gtk-radio-button>`) [Function]
(*group* `<gtk-radio-group*>`)

`set-group` [Method]

Sets a `<gtk-radio-button>`'s group. It should be noted that this does not change the layout of your interface in any way, so if you are changing the group, it is likely you will need to re-arrange the user interface to reflect these changes.

radio-button
a `<gtk-radio-button>`.

group an existing radio button group, such as one returned from `gtk-radio-button-get-group`.

`gtk-radio-button-get-group` (*self* `<gtk-radio-button>`) [Function]
⇒ (*ret* `<gtk-radio-group*>`)

`get-group` [Method]

Retrieves the group assigned to a radio button.

radio-button
a `<gtk-radio-button>`.

ret a linked list containing all the radio buttons in the same group as *radio-button*.

18 GtkToggleButton

Create buttons which retain their state

18.1 Overview

A <gtk-toggle-button> is a <gtk-button> which will remain 'pressed-in' when clicked. Clicking again will cause the toggle button to return to its normal state.

A toggle button is created by calling either gtk-toggle-button-new or gtk-toggle-button-new-with-label. If using the former, it is advisable to pack a widget, (such as a <gtk-label> and/or a <gtk-pixmap>), into the toggle button's container. (See <gtk-button> for more information).

The state of a <gtk-toggle-button> can be set specifically using gtk-toggle-button-set-active, and retrieved using gtk-toggle-button-get-active.

To simply switch the state of a toggle button, use gtk_toggle_button_toggled.

```
void make_toggles (void) {
   GtkWidget *dialog, *toggle1, *toggle2;

   dialog = gtk_dialog_new ();
   toggle1 = gtk_toggle_button_new_with_label ("Hi, i'm a toggle button.");

   /* Makes this toggle button invisible */
   gtk_toggle_button_set_mode (GTK_TOGGLE_BUTTON (toggle1), TRUE);

   g_signal_connect (toggle1, "toggled",
                     G_CALLBACK (output_state), NULL);
   gtk_box_pack_start (GTK_BOX (GTK_DIALOG (dialog)->action_area),
                       toggle1, FALSE, FALSE, 2);

   toggle2 = gtk_toggle_button_new_with_label ("Hi, i'm another toggle button.");
   gtk_toggle_button_set_mode (GTK_TOGGLE_BUTTON (toggle2), FALSE);
   g_signal_connect (toggle2, "toggled",
                     G_CALLBACK (output_state), NULL);
   gtk_box_pack_start (GTK_BOX (GTK_DIALOG (dialog)->action_area),
                       toggle2, FALSE, FALSE, 2);

   gtk_widget_show_all (dialog);
}
```

18.2 Usage

<gtk-toggle-button> [Class]

This <gobject> class defines the following properties:

`active` If the toggle button should be pressed in or not

`inconsistent`
If the toggle button is in an "in between" state

`draw-indicator`
If the toggle part of the button is displayed

`toggled` [Signal on `<gtk-toggle-button>`]
Should be connected if you wish to perform an action whenever the `<gtk-toggle-button>`'s state is changed.

`gtk-toggle-button-new` ⇒ (*ret* `<gtk-widget>`) [Function]
Creates a new toggle button. A widget should be packed into the button, as in `gtk-button-new`.

ret a new toggle button.

`gtk-toggle-button-new-with-label` (*label* `mchars`) [Function]
⇒ (*ret* `<gtk-widget>`)
Creates a new toggle button with a text label.

label a string containing the message to be placed in the toggle button.

ret a new toggle button.

`gtk-toggle-button-new-with-mnemonic` (*label* `mchars`) [Function]
⇒ (*ret* `<gtk-widget>`)
Creates a new `<gtk-toggle-button>` containing a label. The label will be created using `gtk-label-new-with-mnemonic`, so underscores in *label* indicate the mnemonic for the button.

label the text of the button, with an underscore in front of the mnemonic character

ret a new `<gtk-toggle-button>`

`gtk-toggle-button-set-mode` (*self* `<gtk-toggle-button>`) [Function]
(*draw_indicator* `bool`)
`set-mode` [Method]
Sets whether the button is displayed as a separate indicator and label. You can call this function on a checkbutton or a radiobutton with *draw-indicator* = '`#f`' to make the button look like a normal button

This function only affects instances of classes like `<gtk-check-button>` and `<gtk-radio-button>` that derive from `<gtk-toggle-button>`, not instances of `<gtk-toggle-button>` itself.

toggle-button
a `<gtk-toggle-button>`

draw-indicator
if '`#t`', draw the button as a separate indicator and label; if '`#f`', draw the button like a normal button

`gtk-toggle-button-get-mode` (*self* `<gtk-toggle-button>`) [Function]
⇒ (*ret* `bool`)
`get-mode` [Method]

Retrieves whether the button is displayed as a separate indicator and label. See `gtk-toggle-button-set-mode`.

toggle-button
a `<gtk-toggle-button>`

ret '`#t`' if the togglebutton is drawn as a separate indicator and label.

`gtk-toggle-button-toggled` (*self* `<gtk-toggle-button>`) [Function]
`toggled` [Method]

Emits the toggled signal on the `<gtk-toggle-button>`. There is no good reason for an application ever to call this function.

toggle-button
a `<gtk-toggle-button>`.

`gtk-toggle-button-get-active` (*self* `<gtk-toggle-button>`) [Function]
⇒ (*ret* `bool`)
`get-active` [Method]

Queries a `<gtk-toggle-button>` and returns its current state. Returns '`#t`' if the toggle button is pressed in and '`#f`' if it is raised.

toggle-button
a `<gtk-toggle-button>`.

ret a `<gboolean>` value.

`gtk-toggle-button-set-active` (*self* `<gtk-toggle-button>`) [Function]
(*is_active* `bool`)
`set-active` [Method]

Sets the status of the toggle button. Set to '`#t`' if you want the GtkToggleButton to be 'pressed in', and '`#f`' to raise it. This action causes the toggled signal to be emitted.

toggle-button
a `<gtk-toggle-button>`.

is-active '`#t`' or '`#f`'.

`gtk-toggle-button-get-inconsistent` (*self* `<gtk-toggle-button>`) [Function]
⇒ (*ret* `bool`)
`get-inconsistent` [Method]

Gets the value set by `gtk-toggle-button-set-inconsistent`.

toggle-button
a `<gtk-toggle-button>`

ret '`#t`' if the button is displayed as inconsistent, '`#f`' otherwise

`gtk-toggle-button-set-inconsistent` (*self* `<gtk-toggle-button>`) [Function]
(*setting* `bool`)
`set-inconsistent` [Method]

If the user has selected a range of elements (such as some text or spreadsheet cells) that are affected by a toggle button, and the current values in that range are inconsistent, you may want to display the toggle in an "in between" state. This function turns on "in between" display. Normally you would turn off the inconsistent state again if the user toggles the toggle button. This has to be done manually, `gtk-toggle-button-set-inconsistent` only affects visual appearance, it doesn't affect the semantics of the button.

toggle-button
: a `<gtk-toggle-button>`

setting
: '`#t`' if state is inconsistent

19 GtkLinkButton

Create buttons bound to a URL

19.1 Overview

A <gtk-link-button> is a <gtk-button> with a hyperlink, similar to the one used by web browsers, which triggers an action when clicked. It is useful to show quick links to resources.

A link button is created by calling either gtk-link-button-new or gtk-link-button-new-with-label. If using the former, the URI you pass to the constructor is used as a label for the widget.

The URI bound to a <gtk-link-button> can be set specifically using gtk-link-button-set-uri, and retrieved using gtk-link-button-get-uri.

<gtk-link-button> offers a global hook, which is called when the used clicks on it: see gtk-link-button-set-uri-hook.

<gtk-link-button> was added in GTK+ 2.10.

19.2 Usage

<gtk-link-button> [Class]

This <gobject> class defines the following properties:

uri The URI bound to this button

gtk-link-button-new (*uri* mchars) ⇒ (*ret* <gtk-widget>) [Function]

Creates a new <gtk-link-button> with the URI as its text.

uri a valid URI

ret a new link button widget.

Since 2.10

gtk-link-button-new-with-label (*uri* mchars) (*label* mchars) [Function]
⇒ (*ret* <gtk-widget>)

Creates a new <gtk-link-button> containing a label.

uri a valid URI

label the text of the button

ret a new link button widget.

Since 2.10

gtk-link-button-get-uri (*self* <gtk-link-button>) [Function]
⇒ (*ret* mchars)

get-uri [Method]

Retrieves the URI set using gtk-link-button-set-uri.

link-button
a <gtk-link-button>

ret a valid URI. The returned string is owned by the link button and should not be modified or freed.

Since 2.10

`gtk-link-button-set-uri` (*self* `<gtk-link-button>`) (*uri* `mchars`) [Function]
`set-uri` [Method]

Sets *uri* as the URI where the `<gtk-link-button>` points.

link-button
a `<gtk-link-button>`

uri a valid URI

Since 2.10

20 GtkEntry

A single line text entry field

20.1 Overview

The <gtk-entry> widget is a single line text entry widget. A fairly large set of key bindings are supported by default. If the entered text is longer than the allocation of the widget, the widget will scroll so that the cursor position is visible.

20.2 Usage

<gtk-entry> [Class]

This <gobject> class defines the following properties:

cursor-position
: The current position of the insertion cursor in chars

selection-bound
: The position of the opposite end of the selection from the cursor in chars

editable
: Whether the entry contents can be edited

max-length
: Maximum number of characters for this entry. Zero if no maximum

visibility
: FALSE displays the "invisible char" instead of the actual text (password mode)

has-frame
: FALSE removes outside bevel from entry

inner-border
: Border between text and frame. Overrides the inner-border style property

invisible-char
: The character to use when masking entry contents (in "password mode")

activates-default
: Whether to activate the default widget (such as the default button in a dialog) when Enter is pressed

width-chars
: Number of characters to leave space for in the entry

scroll-offset
: Number of pixels of the entry scrolled off the screen to the left

text
: The contents of the entry

xalign
: The horizontal alignment, from 0 (left) to 1 (right). Reversed for RTL layouts.

truncate-multiline
: Whether to truncate multiline pastes to one line.

`shadow-type`
Which kind of shadow to draw around the entry when has-frame is set

`move-cursor` (*arg0* `<gtk-movement-step>`) (*arg1* `<gint>`) (*arg2* `<gboolean>`) [Signal on `<gtk-entry>`]

`copy-clipboard` [Signal on `<gtk-entry>`]

`populate-popup` (*arg0* `<gtk-menu>`) [Signal on `<gtk-entry>`]

`activate` [Signal on `<gtk-entry>`]

`insert-at-cursor` (*arg0* `<gchararray>`) [Signal on `<gtk-entry>`]

`delete-from-cursor` (*arg0* `<gtk-delete-type>`) (*arg1* `<gint>`) [Signal on `<gtk-entry>`]

`backspace` [Signal on `<gtk-entry>`]

`cut-clipboard` [Signal on `<gtk-entry>`]

`paste-clipboard` [Signal on `<gtk-entry>`]

`toggle-overwrite` [Signal on `<gtk-entry>`]

`gtk-entry-set-text` (*self* `<gtk-entry>`) (*text* `mchars`) [Function]
`set-text` [Method]
Sets the text in the widget to the given value, replacing the current contents.

entry a `<gtk-entry>`.

text the new text.

`gtk-entry-get-text` (*self* `<gtk-entry>`) ⇒ (*ret* `mchars`) [Function]
`get-text` [Method]
Retrieves the contents of the entry widget. See also `gtk-editable-get-chars`.

entry a `<gtk-entry>`

ret a pointer to the contents of the widget as a string. This string points to internally allocated storage in the widget and must not be freed, modified or stored.

`gtk-entry-set-visibility` (*self* `<gtk-entry>`) (*visible* `bool`) [Function]
`set-visibility` [Method]
Sets whether the contents of the entry are visible or not. When visibility is set to '`#f`', characters are displayed as the invisible char, and will also appear that way when the text in the entry widget is copied elsewhere.

The default invisible char is the asterisk '*', but it can be changed with `gtk-entry-set-invisible-char`.

entry a `<gtk-entry>`.

visible '`#t`' if the contents of the entry are displayed as plaintext.

`gtk-entry-set-invisible-char` (*self* `<gtk-entry>`) [Function]
(*ch* `unsigned-int32`)
`set-invisible-char` [Method]

Sets the character to use in place of the actual text when `gtk-entry-set-visibility` has been called to set text visibility to '`#f`'. i.e. this is the character used in "password mode" to show the user how many characters have been typed. The default invisible char is an asterisk ('*'). If you set the invisible char to 0, then the user will get no feedback at all; there will be no text on the screen as they type.

entry a `<gtk-entry>`

ch a Unicode character

`gtk-entry-set-max-length` (*self* `<gtk-entry>`) (*max* `int`) [Function]
`set-max-length` [Method]

Sets the maximum allowed length of the contents of the widget. If the current contents are longer than the given length, then they will be truncated to fit.

entry a `<gtk-entry>`.

max the maximum length of the entry, or 0 for no maximum. (other than the maximum length of entries.) The value passed in will be clamped to the range 0-65536.

`gtk-entry-get-activates-default` (*self* `<gtk-entry>`) [Function]
⇒ (*ret* `bool`)
`get-activates-default` [Method]

Retrieves the value set by `gtk-entry-set-activates-default`.

entry a `<gtk-entry>`

ret '`#t`' if the entry will activate the default widget

`gtk-entry-get-has-frame` (*self* `<gtk-entry>`) ⇒ (*ret* `bool`) [Function]
`get-has-frame` [Method]

Gets the value set by `gtk-entry-set-has-frame`.

entry a `<gtk-entry>`

ret whether the entry has a beveled frame

`gtk-entry-get-inner-border` (*self* `<gtk-entry>`) [Function]
⇒ (*ret* `<gtk-border*>`)
`get-inner-border` [Method]

This function returns the entry's inner-border property. See `gtk-entry-set-inner-border` for more information.

entry a `<gtk-entry>`

ret the entry's `<gtk-border>`, or '`#f`' if none was set.

Since 2.10

`gtk-entry-get-width-chars` (*self* `<gtk-entry>`) ⇒ (*ret* `int`) [Function]
`get-width-chars` [Method]

Gets the value set by `gtk-entry-set-width-chars`.

entry a `<gtk-entry>`

ret number of chars to request space for, or negative if unset

`gtk-entry-set-activates-default` (*self* `<gtk-entry>`) (*setting* `bool`) [Function]
`set-activates-default` [Method]

If *setting* is '`#t`', pressing Enter in the *entry* will activate the default widget for the window containing the entry. This usually means that the dialog box containing the entry will be closed, since the default widget is usually one of the dialog buttons.

(For experts: if *setting* is '`#t`', the entry calls `gtk-window-activate-default` on the window containing the entry, in the default handler for the "activate" signal.)

entry a `<gtk-entry>`

setting '`#t`' to activate window's default widget on Enter keypress

`gtk-entry-set-has-frame` (*self* `<gtk-entry>`) (*setting* `bool`) [Function]
`set-has-frame` [Method]

Sets whether the entry has a beveled frame around it.

entry a `<gtk-entry>`

setting new value

`gtk-entry-set-inner-border` (*self* `<gtk-entry>`) (*border* `<gtk-border*>`) [Function]
`set-inner-border` [Method]

Sets '`entry`''s inner-border property to '`border`', or clears it if '`#f`' is passed. The inner-border is the area around the entry's text, but inside its frame.

If set, this property overrides the inner-border style property. Overriding the style-provided border is useful when you want to do in-place editing of some text in a canvas or list widget, where pixel-exact positioning of the entry is important.

entry a `<gtk-entry>`

border a `<gtk-border>`, or '`#f`'

Since 2.10

`gtk-entry-set-width-chars` (*self* `<gtk-entry>`) (*n_chars* `int`) [Function]
`set-width-chars` [Method]

Changes the size request of the entry to be about the right size for *n-chars* characters. Note that it changes the size *request*, the size can still be affected by how you pack the widget into containers. If *n-chars* is -1, the size reverts to the default entry size.

entry a `<gtk-entry>`

n-chars width in chars

`gtk-entry-get-invisible-char` (*self* `<gtk-entry>`) [Function]
⇒ (*ret* `unsigned-int32`)
`get-invisible-char` [Method]

Retrieves the character displayed in place of the real characters for entries with visisbility set to false. See `gtk-entry-set-invisible-char`.

entry a `<gtk-entry>`

ret the current invisible char, or 0, if the entry does not show invisible text at all.

`gtk-entry-set-alignment` (*self* `<gtk-entry>`) (*xalign* `float`) [Function]
`set-alignment` [Method]

Sets the alignment for the contents of the entry. This controls the horizontal positioning of the contents when the displayed text is shorter than the width of the entry.

entry a `<gtk-entry>`

xalign The horizontal alignment, from 0 (left) to 1 (right). Reversed for RTL layouts

Since 2.4

`gtk-entry-get-alignment` (*self* `<gtk-entry>`) ⇒ (*ret* `float`) [Function]
`get-alignment` [Method]

Gets the value set by `gtk-entry-set-alignment`.

entry a `<gtk-entry>`

ret the alignment

Since 2.4

`gtk-entry-get-layout` (*self* `<gtk-entry>`) [Function]
⇒ (*ret* `<pango-layout*>`)
`get-layout` [Method]

Gets the `<pango-layout>` used to display the entry. The layout is useful to e.g. convert text positions to pixel positions, in combination with `gtk-entry-get-layout-offsets`. The returned layout is owned by the entry and must not be modified or freed by the caller.

Keep in mind that the layout text may contain a preedit string, so `gtk-entry-layout-index-to-text-index` and `gtk-entry-text-index-to-layout-index` are needed to convert byte indices in the layout to byte indices in the entry contents.

entry a `<gtk-entry>`

ret the `<pango-layout>` for this entry

`gtk-entry-get-layout-offsets` (*self* `<gtk-entry>`) ⇒ (*x* `int`) [Function]
(*y* `int`)
`get-layout-offsets` [Method]

Obtains the position of the `<pango-layout>` used to render text in the entry, in widget coordinates. Useful if you want to line up the text in an entry with some other text, e.g. when using the entry to implement editable cells in a sheet widget.

Also useful to convert mouse events into coordinates inside the <pango-layout>, e.g. to take some action if some part of the entry text is clicked.

Note that as the user scrolls around in the entry the offsets will change; you'll need to connect to the "notify::scroll-offset" signal to track this. Remember when using the <pango-layout> functions you need to convert to and from pixels using pango-pixels or <pango-scale>.

Keep in mind that the layout text may contain a preedit string, so gtk-entry-layout-index-to-text-index and gtk-entry-text-index-to-layout-index are needed to convert byte indices in the layout to byte indices in the entry contents.

entry a <gtk-entry>

x location to store X offset of layout, or '#f'

y location to store Y offset of layout, or '#f'

gtk-entry-get-max-length (*self* <gtk-entry>) ⇒ (*ret* int) [Function]
get-max-length [Method]

Retrieves the maximum allowed length of the text in *entry*. See gtk-entry-set-max-length.

entry a <gtk-entry>

ret the maximum allowed number of characters in <gtk-entry>, or 0 if there is no maximum.

gtk-entry-get-visibility (*self* <gtk-entry>) ⇒ (*ret* bool) [Function]
get-visibility [Method]

Retrieves whether the text in *entry* is visible. See gtk-entry-set-visibility.

entry a <gtk-entry>

ret '#t' if the text is currently visible

gtk-entry-set-completion (*self* <gtk-entry>) [Function]
(*completion* <gtk-entry-completion>)
set-completion [Method]

Sets *completion* to be the auxiliary completion object to use with *entry*. All further configuration of the completion mechanism is done on *completion* using the <gtk-entry-completion> API. Completion is disabled if *completion* is set to '#f'.

entry A <gtk-entry>.

completion
The <gtk-entry-completion> or '#f'.

Since 2.4

gtk-entry-get-completion (*self* <gtk-entry>) [Function]
⇒ (*ret* <gtk-entry-completion>)
get-completion [Method]

Returns the auxiliary completion object currently in use by *entry*.

entry A <gtk-entry>.

ret The auxiliary completion object currently in use by *entry*.

Since 2.4

21 GtkEntryCompletion

Completion functionality for GtkEntry

21.1 Overview

<gtk-entry-completion> is an auxiliary object to be used in conjunction with <gtk-entry> to provide the completion functionality. It implements the <gtk-cell-layout> interface, to allow the user to add extra cells to the <gtk-tree-view> with completion matches.

"Completion functionality" means that when the user modifies the text in the entry, <gtk-entry-completion> checks which rows in the model match the current content of the entry, and displays a list of matches. By default, the matching is done by comparing the entry text case-insensitively against the text column of the model (see gtk-entry-completion-set-text-column), but this can be overridden with a custom match function (see gtk-entry-completion-set-match-func).

When the user selects a completion, the content of the entry is updated. By default, the content of the entry is replaced by the text column of the model, but this can be overridden by connecting to the ::match-selected signal and updating the entry in the signal handler. Note that you should return '#t' from the signal handler to suppress the default behaviour.

To add completion functionality to an entry, use gtk-entry-set-completion.

In addition to regular completion matches, which will be inserted into the entry when they are selected, <gtk-entry-completion> also allows to display "actions" in the popup window. Their appearance is similar to menuitems, to differentiate them clearly from completion strings. When an action is selected, the ::action-activated signal is emitted.

21.2 Usage

<gtk-entry-completion> [Class]

This <gobject> class defines the following properties:

model The model to find matches in

minimum-key-length
Minimum length of the search key in order to look up matches

text-column
The column of the model containing the strings.

inline-completion
Whether the common prefix should be inserted automatically

popup-completion
Whether the completions should be shown in a popup window

popup-set-width
If TRUE, the popup window will have the same size as the entry

popup-single-match
If TRUE, the popup window will appear for a single match.

`inline-selection`
Your description here

`insert-prefix` (*arg0* `<gchararray>`) [Signal on `<gtk-entry-completion>`]
⇒ `<gboolean>`
Gets emitted when the inline autocompletion is triggered. The default behaviour is to make the entry display the whole prefix and select the newly inserted part.

Applications may connect to this signal in order to insert only a smaller part of the *prefix* into the entry - e.g. the entry used in the `<gtk-file-chooser>` inserts only the part of the prefix up to the next '/'.

Since 2.6

`match-selected` (*arg0* `<gtk-tree-model>`) [Signal on `<gtk-entry-completion>`]
(*arg1* `<gtk-tree-iter>`) ⇒ `<gboolean>`
Gets emitted when a match from the list is selected. The default behaviour is to replace the contents of the entry with the contents of the text column in the row pointed to by *iter*.

Since 2.4

`cursor-on-match` (*arg0* `<gtk-tree-model>`) [Signal on `<gtk-entry-completion>`]
(*arg1* `<gtk-tree-iter>`) ⇒ `<gboolean>`
undocumented

`action-activated` (*arg0* `<gint>`) [Signal on `<gtk-entry-completion>`]
Gets emitted when an action is activated.

Since 2.4

`gtk-entry-completion-new` ⇒ (*ret* `<gtk-entry-completion>`) [Function]
Creates a new `<gtk-entry-completion>` object.

ret A newly created `<gtk-entry-completion>` object.

Since 2.4

`gtk-entry-completion-get-entry` (*self* `<gtk-entry-completion>`) [Function]
⇒ (*ret* `<gtk-widget>`)
`get-entry` [Method]
Gets the entry *completion* has been attached to.

completion
A `<gtk-entry-completion>`.

ret The entry *completion* has been attached to.

Since 2.4

`gtk-entry-completion-set-model` (*self* `<gtk-entry-completion>`) [Function]
(*model* `<gtk-tree-model>`)
`set-model` [Method]
Sets the model for a `<gtk-entry-completion>`. If *completion* already has a model set, it will remove it before setting the new model. If model is '`#f`', then it will unset the model.

completion
A <gtk-entry-completion>.

model The <gtk-tree-model>.

Since 2.4

gtk-entry-completion-get-model (*self* <gtk-entry-completion>) [Function]
⇒ (*ret* <gtk-tree-model>)

get-model [Method]

Returns the model the <gtk-entry-completion> is using as data source. Returns '#f' if the model is unset.

completion
A <gtk-entry-completion>.

ret A <gtk-tree-model>, or '#f' if none is currently being used.

Since 2.4

gtk-entry-completion-set-match-func [Function]
(*self* <gtk-entry-completion>)
(*func* <gtk-entry-completion-match-func>) (*func_data* <gpointer>)
(*func_notify* <g-destroy-notify>)

set-match-func [Method]

Sets the match function for *completion* to be *func*. The match function is used to determine if a row should or should not be in the completion list.

completion
A <gtk-entry-completion>.

func The <gtk-entry-completion-match-func> to use.

func-data The user data for *func*.

func-notify
Destroy notifier for *func-data*.

Since 2.4

gtk-entry-completion-complete (*self* <gtk-entry-completion>) [Function]

complete [Method]

Requests a completion operation, or in other words a refiltering of the current list with completions, using the current key. The completion list view will be updated accordingly.

completion
A <gtk-entry-completion>.

Since 2.4

gtk-entry-completion-insert-prefix [Function]
(*self* <gtk-entry-completion>)

insert-prefix [Method]

Requests a prefix insertion.

completion
a `<gtk-entry-completion>`

Since 2.6

`gtk-entry-completion-delete-action` [Function]
(*self* `<gtk-entry-completion>`) (*index* `int`)
`delete-action` [Method]

Deletes the action at *index* from *completion*'s action list.

completion
A `<gtk-entry-completion>`.

index The index of the item to Delete.

Since 2.4

22 GtkHScale

A horizontal slider widget for selecting a value from a range

22.1 Overview

The `<gtk-hscale>` widget is used to allow the user to select a value using a horizontal slider. To create one, use `gtk-hscale-new-with-range`.

The position to show the current value, and the number of decimal places shown can be set using the parent `<gtk-scale>` class's functions.

22.2 Usage

`<gtk-hscale>` [Class]

This `<gobject>` class defines no properties, other than those defined by its superclasses.

`gtk-hscale-new` (*adjustment* `<gtk-adjustment>`) [Function]
⇒ (*ret* `<gtk-widget>`)

Creates a new `<gtk-hscale>`.

adjustment
: the `<gtk-adjustment>` which sets the range of the scale.

ret
: a new `<gtk-hscale>`.

`gtk-hscale-new-with-range` (*min* `double`) (*max* `double`) [Function]
(*step* `double`) ⇒ (*ret* `<gtk-widget>`)

Creates a new horizontal scale widget that lets the user input a number between *min* and *max* (including *min* and *max*) with the increment *step*. *step* must be nonzero; it's the distance the slider moves when using the arrow keys to adjust the scale value.

Note that the way in which the precision is derived works best if *step* is a power of ten. If the resulting precision is not suitable for your needs, use `gtk-scale-set-digits` to correct it.

min
: minimum value

max
: maximum value

step
: step increment (tick size) used with keyboard shortcuts

ret
: a new `<gtk-hscale>`

23 GtkVScale

A vertical slider widget for selecting a value from a range

23.1 Overview

The `<gtk-vscale>` widget is used to allow the user to select a value using a vertical slider. To create one, use `gtk-hscale-new-with-range`.

The position to show the current value, and the number of decimal places shown can be set using the parent `<gtk-scale>` class's functions.

23.2 Usage

`<gtk-vscale>` [Class]

This `<gobject>` class defines no properties, other than those defined by its superclasses.

`gtk-vscale-new` (*adjustment* `<gtk-adjustment>`) [Function]
⇒ (*ret* `<gtk-widget>`)

Creates a new `<gtk-vscale>`.

adjustment
: the `<gtk-adjustment>` which sets the range of the scale.

ret
: a new `<gtk-vscale>`.

`gtk-vscale-new-with-range` (*min* `double`) (*max* `double`) [Function]
(*step* `double`) ⇒ (*ret* `<gtk-widget>`)

Creates a new vertical scale widget that lets the user input a number between *min* and *max* (including *min* and *max*) with the increment *step*. *step* must be nonzero; it's the distance the slider moves when using the arrow keys to adjust the scale value.

Note that the way in which the precision is derived works best if *step* is a power of ten. If the resulting precision is not suitable for your needs, use `gtk-scale-set-digits` to correct it.

min
: minimum value

max
: maximum value

step
: step increment (tick size) used with keyboard shortcuts

ret
: a new `<gtk-vscale>`

24 GtkSpinButton

Retrieve an integer or floating-point number from the user

24.1 Overview

A <gtk-spin-button> is an ideal way to allow the user to set the value of some attribute. Rather than having to directly type a number into a <gtk-entry>, <gtk-spin-button> allows the user to click on one of two arrows to increment or decrement the displayed value. A value can still be typed in, with the bonus that it can be checked to ensure it is in a given range.

The main properties of a <gtk-spin-button> are through a <gtk-adjustment>. See the <gtk-adjustment> section for more details about an adjustment's properties.

```
/* Provides a function to retrieve an integer value from a GtkSpinButton
 * and creates a spin button to model percentage values.
 */

gint grab_int_value (GtkSpinButton *a_spinner, gpointer user_data) {
   return gtk_spin_button_get_value_as_int (a_spinner);
}

void create_integer_spin_button (void) {

   GtkWidget *window, *spinner;
   GtkAdjustment *spinner_adj;

   spinner_adj = (GtkAdjustment *) gtk_adjustment_new (50.0, 0.0, 100.0, 1.0, 5.0, 5.0

   window = gtk_window_new (GTK_WINDOW_TOPLEVEL);
   gtk_container_set_border_width (GTK_CONTAINER (window), 5);

   /* creates the spinner, with no decimal places */
   spinner = gtk_spin_button_new (spinner_adj, 1.0, 0);
   gtk_container_add (GTK_CONTAINER (window), spinner);

   gtk_widget_show_all (window);
   return;
}



/* Provides a function to retrieve a floating point value from a
 * GtkSpinButton, and creates a high precision spin button.
 */
```

```
gfloat grab_int_value (GtkSpinButton *a_spinner, gpointer user_data) {
   return gtk_spin_button_get_value (a_spinner);
}

void create_floating_spin_button (void) {

   GtkWidget *window, *spinner;
   GtkAdjustment *spinner_adj;

   spinner_adj = (GtkAdjustment *) gtk_adjustment_new (2.500, 0.0, 5.0, 0.001, 0.1, 0.

   window = gtk_window_new (GTK_WINDOW_TOPLEVEL);
   gtk_container_set_border_width (GTK_CONTAINER (window), 5);

   /* creates the spinner, with three decimal places */
   spinner = gtk_spin_button_new (spinner_adj, 0.001, 3);
   gtk_container_add (GTK_CONTAINER (window), spinner);

   gtk_widget_show_all (window);
   return;
}
```

24.2 Usage

`<gtk-spin-button>` [Class]

This `<gobject>` class defines the following properties:

`adjustment`
: The adjustment that holds the value of the spinbutton

`climb-rate`
: The acceleration rate when you hold down a button

`digits` The number of decimal places to display

`snap-to-ticks`
: Whether erroneous values are automatically changed to a spin button's nearest step increment

`numeric` Whether non-numeric characters should be ignored

`wrap` Whether a spin button should wrap upon reaching its limits

`update-policy`
: Whether the spin button should update always, or only when the value is legal

`value` Reads the current value, or sets a new value

`value-changed` [Signal on `<gtk-spin-button>`]

`change-value` (*arg0* `<gtk-scroll-type>`) [Signal on `<gtk-spin-button>`]

`input` (*arg0* `<gpointer>`) ⇒ `<gint>` [Signal on `<gtk-spin-button>`]

`output` ⇒ `<gboolean>` [Signal on `<gtk-spin-button>`]

`wrapped` [Signal on `<gtk-spin-button>`]

The wrapped signal is emitted right after the spinbutton wraps from its maximum to minimum value or vice-versa.

Since 2.10

`gtk-spin-button-configure` (*self* `<gtk-spin-button>`) [Function]
(*adjustment* `<gtk-adjustment>`) (*climb_rate* `double`)
(*digits* `unsigned-int`)

`configure` [Method]

Changes the properties of an existing spin button. The adjustment, climb rate, and number of decimal places are all changed accordingly, after this function call.

spin-button
a `<gtk-spin-button>`.

adjustment
a `<gtk-adjustment>`.

climb-rate the new climb rate.

digits the number of decimal places to display in the spin button.

`gtk-spin-button-new` (*adjustment* `<gtk-adjustment>`) [Function]
(*climb_rate* `double`) (*digits* `unsigned-int`) ⇒ (*ret* `<gtk-widget>`)

Creates a new `<gtk-spin-button>`.

adjustment
the `<gtk-adjustment>` object that this spin button should use.

climb-rate specifies how much the spin button changes when an arrow is clicked on.

digits the number of decimal places to display.

ret The new spin button as a `<gtk-widget>`.

`gtk-spin-button-new-with-range` (*min* `double`) (*max* `double`) [Function]
(*step* `double`) ⇒ (*ret* `<gtk-widget>`)

This is a convenience constructor that allows creation of a numeric `<gtk-spin-button>` without manually creating an adjustment. The value is initially set to the minimum value and a page increment of 10 * *step* is the default. The precision of the spin button is equivalent to the precision of *step*.

Note that the way in which the precision is derived works best if *step* is a power of ten. If the resulting precision is not suitable for your needs, use `gtk-spin-button-set-digits` to correct it.

min Minimum allowable value

max Maximum allowable value

step Increment added or subtracted by spinning the widget

ret The new spin button as a `<gtk-widget>`.

`gtk-spin-button-set-adjustment` (*self* `<gtk-spin-button>`) [Function]
(*adjustment* `<gtk-adjustment>`)
`set-adjustment` [Method]

Replaces the `<gtk-adjustment>` associated with *spin-button*.

spin-button
a `<gtk-spin-button>`

adjustment
a `<gtk-adjustment>` to replace the existing adjustment

`gtk-spin-button-get-adjustment` (*self* `<gtk-spin-button>`) [Function]
⇒ (*ret* `<gtk-adjustment>`)
`get-adjustment` [Method]

Get the adjustment associated with a `<gtk-spin-button>`

spin-button
a `<gtk-spin-button>`

ret the `<gtk-adjustment>` of *spin-button*

`gtk-spin-button-set-digits` (*self* `<gtk-spin-button>`) [Function]
(*digits* `unsigned-int`)
`set-digits` [Method]

Set the precision to be displayed by *spin-button*. Up to 20 digit precision is allowed.

spin-button
a `<gtk-spin-button>`

digits the number of digits after the decimal point to be displayed for the spin button's value

`gtk-spin-button-set-increments` (*self* `<gtk-spin-button>`) [Function]
(*step* `double`) (*page* `double`)
`set-increments` [Method]

Sets the step and page increments for spin_button. This affects how quickly the value changes when the spin button's arrows are activated.

spin-button
a `<gtk-spin-button>`

step increment applied for a button 1 press.

page increment applied for a button 2 press.

`gtk-spin-button-set-range` (*self* `<gtk-spin-button>`) [Function]
(*min* `double`) (*max* `double`)
`set-range` [Method]

Sets the minimum and maximum allowable values for *spin-button*

spin-button
: a `<gtk-spin-button>`

min
: minimum allowable value

max
: maximum allowable value

`gtk-spin-button-get-value-as-int` (*self* `<gtk-spin-button>`) [Function]
⇒ (*ret* `int`)

`get-value-as-int` [Method]

Get the value *spin-button* represented as an integer.

spin-button
: a `<gtk-spin-button>`

ret
: the value of *spin-button*

`gtk-spin-button-set-value` (*self* `<gtk-spin-button>`) [Function]
(*value* `double`)

`set-value` [Method]

Set the value of *spin-button*.

spin-button
: a `<gtk-spin-button>`

value
: the new value

`gtk-spin-button-set-update-policy` (*self* `<gtk-spin-button>`) [Function]
(*policy* `<gtk-spin-button-update-policy>`)

`set-update-policy` [Method]

Sets the update behavior of a spin button. This determines whether the spin button is always updated or only when a valid value is set.

spin-button
: a `<gtk-spin-button>`

policy
: a `<gtk-spin-button-update-policy>` value

`gtk-spin-button-set-numeric` (*self* `<gtk-spin-button>`) [Function]
(*numeric* `bool`)

`set-numeric` [Method]

Sets the flag that determines if non-numeric text can be typed into the spin button.

spin-button
: a `<gtk-spin-button>`

numeric
: flag indicating if only numeric entry is allowed.

`gtk-spin-button-spin` (*self* `<gtk-spin-button>`) [Function]
(*direction* `<gtk-spin-type>`) (*increment* `double`)

`spin` [Method]

Increment or decrement a spin button's value in a specified direction by a specified amount.

spin-button
a `<gtk-spin-button>`

direction a `<gtk-spin-type>` indicating the direction to spin.

increment step increment to apply in the specified direction.

`gtk-spin-button-set-wrap` (*self* `<gtk-spin-button>`) (*wrap* `bool`) [Function]
`set-wrap` [Method]
Sets the flag that determines if a spin button value wraps around to the opposite limit when the upper or lower limit of the range is exceeded.

spin-button
a `<gtk-spin-button>`

wrap a flag indicating if wrapping behavior is performed.

`gtk-spin-button-set-snap-to-ticks` (*self* `<gtk-spin-button>`) [Function]
(*snap_to_ticks* `bool`)
`set-snap-to-ticks` [Method]
Sets the policy as to whether values are corrected to the nearest step increment when a spin button is activated after providing an invalid value.

spin-button
a `<gtk-spin-button>`

snap-to-ticks
a flag indicating if invalid values should be corrected.

`gtk-spin-button-update` (*self* `<gtk-spin-button>`) [Function]
`update` [Method]
Manually force an update of the spin button.

spin-button
a `<gtk-spin-button>`

`gtk-spin-button-get-digits` (*self* `<gtk-spin-button>`) [Function]
⇒ (*ret* `unsigned-int`)
`get-digits` [Method]
Fetches the precision of *spin-button*. See `gtk-spin-button-set-digits`.

spin-button
a `<gtk-spin-button>`

ret the current precision

`gtk-spin-button-get-increments` (*self* `<gtk-spin-button>`) [Function]
⇒ (*step* `double`) (*page* `double`)
`get-increments` [Method]
Gets the current step and page the increments used by *spin-button*. See `gtk-spin-button-set-increments`.

spin-button
a `<gtk-spin-button>`

step location to store step increment, or '`#f`'

page location to store page increment, or '`#f`'

`gtk-spin-button-get-numeric` (*self* `<gtk-spin-button>`) ⇒ (*ret* `bool`) [Function]

`get-numeric` [Method]

Returns whether non-numeric text can be typed into the spin button. See `gtk-spin-button-set-numeric`.

spin-button a `<gtk-spin-button>`

ret '`#t`' if only numeric text can be entered

`gtk-spin-button-get-range` (*self* `<gtk-spin-button>`) ⇒ (*min* `double`) (*max* `double`) [Function]

`get-range` [Method]

Gets the range allowed for *spin-button*. See `gtk-spin-button-set-range`.

spin-button a `<gtk-spin-button>`

min location to store minimum allowed value, or '`#f`'

max location to store maximum allowed value, or '`#f`'

`gtk-spin-button-get-snap-to-ticks` (*self* `<gtk-spin-button>`) ⇒ (*ret* `bool`) [Function]

`get-snap-to-ticks` [Method]

Returns whether the values are corrected to the nearest step. See `gtk-spin-button-set-snap-to-ticks`.

spin-button a `<gtk-spin-button>`

ret '`#t`' if values are snapped to the nearest step.

`gtk-spin-button-get-value` (*self* `<gtk-spin-button>`) ⇒ (*ret* `double`) [Function]

`get-value` [Method]

Get the value in the *spin-button*.

spin-button a `<gtk-spin-button>`

ret the value of *spin-button*

`gtk-spin-button-get-wrap` (*self* `<gtk-spin-button>`) ⇒ (*ret* `bool`) [Function]

`get-wrap` [Method]

Returns whether the spin button's value wraps around to the opposite limit when the upper or lower limit of the range is exceeded. See `gtk-spin-button-set-wrap`.

spin-button a `<gtk-spin-button>`

ret '`#t`' if the spin button wraps around

25 GtkEditable

Interface for text-editing widgets

25.1 Overview

The <gtk-editable> interface is an interface which should be implemented by text editing widgets, such as <gtk-entry> and <gtk-text>. It contains functions for generically manipulating an editable widget, a large number of action signals used for key bindings, and several signals that an application can connect to to modify the behavior of a widget.

As an example of the latter usage, by connecting the following handler to "insert_text", an application can convert all entry into a widget into uppercase.

```
#include <ctype.h>

void
insert_text_handler (GtkEditable *editable,
                     const gchar *text,
                     gint         length,
                     gint        *position,
                     gpointer     data)
{
  int i;
  gchar *result = g_utf8_strup (text, length);

  g_signal_handlers_block_by_func (editable,
   (gpointer) insert_text_handler, data);
  gtk_editable_insert_text (editable, result, length, position);
  g_signal_handlers_unblock_by_func (editable,
                                     (gpointer) insert_text_handler, data);

  g_signal_stop_emission_by_name (editable, "insert_text");

  g_free (result);
}
```

25.2 Usage

<gtk-editable> [Class]

This <gobject> class defines no properties, other than those defined by its superclasses.

changed [Signal on <gtk-editable>]

Indicates that the user has changed the contents of the widget.

`insert-text` (*arg0* `<gchararray>`) (*arg1* `<gint>`) (*arg2* `<gpointer>`) [Signal on `<gtk-editable>`]

This signal is emitted when text is inserted into the widget by the user. The default handler for this signal will normally be responsible for inserting the text, so by connecting to this signal and then stopping the signal with `gtk-signal-emit-stop`, it is possible to modify the inserted text, or prevent it from being inserted entirely.

`delete-text` (*arg0* `<gint>`) (*arg1* `<gint>`) [Signal on `<gtk-editable>`]

This signal is emitted when text is deleted from the widget by the user. The default handler for this signal will normally be responsible for inserting the text, so by connecting to this signal and then stopping the signal with `gtk-signal-emit-stop`, it is possible to modify the inserted text, or prevent it from being inserted entirely. The *start-pos* and *end-pos* parameters are interpreted as for `gtk-editable-delete-text`

`gtk-editable-select-region` (*self* `<gtk-editable>`) (*start* `int`) (*end* `int`) [Function]

`select-region` [Method]

Selects a region of text. The characters that are selected are those characters at positions from *start-pos* up to, but not including *end-pos*. If *end-pos* is negative, then the the characters selected will be those characters from *start-pos* to the end of the text.

editable a `<gtk-editable>` widget.

start the starting position.

end the end position.

`gtk-editable-get-selection-bounds` (*self* `<gtk-editable>`) ⇒ (*ret* `bool`) (*start* `int`) (*end* `int`) [Function]

`get-selection-bounds` [Method]

Gets the current selection bounds, if there is a selection.

editable a `<gtk-editable>` widget.

start location to store the starting position, or '`#f`'.

end location to store the end position, or '`#f`'.

ret '`#t`' if there is a selection.

`gtk-editable-insert-text` (*self* `<gtk-editable>`) (*new_text* `mchars`) (*position* `int`) ⇒ (*ret* `int`) [Function]

`insert-text` [Method]

Inserts text at a given position.

editable a `<gtk-editable>` widget.

new-text the text to insert.

new-text-length
the length of the text to insert, in bytes

position an inout parameter. The caller initializes it to the position at which to insert the text. After the call it points at the position after the newly inserted text.

`gtk-editable-delete-text` (*self* `<gtk-editable>`) (*start_pos* `int`) (*end_pos* `int`) [Function]
`delete-text` [Method]

Deletes a sequence of characters. The characters that are deleted are those characters at positions from *start-pos* up to, but not including *end-pos*. If *end-pos* is negative, then the the characters deleted will be those characters from *start-pos* to the end of the text.

editable a `<gtk-editable>` widget.

start-pos the starting position.

end-pos the end position.

`gtk-editable-get-chars` (*self* `<gtk-editable>`) (*start_pos* `int`) (*end_pos* `int`) ⇒ (*ret* `mchars`) [Function]
`get-chars` [Method]

Retrieves a sequence of characters. The characters that are retrieved are those characters at positions from *start-pos* up to, but not including *end-pos*. If *end-pos* is negative, then the the characters retrieved will be those characters from *start-pos* to the end of the text.

editable a `<gtk-editable>` widget.

start-pos the starting position.

end-pos the end position.

ret the characters in the indicated region. The result must be freed with `g-free` when the application is finished with it.

`gtk-editable-cut-clipboard` (*self* `<gtk-editable>`) [Function]
`cut-clipboard` [Method]

Causes the characters in the current selection to be copied to the clipboard and then deleted from the widget.

editable a `<gtk-editable>` widget.

`gtk-editable-copy-clipboard` (*self* `<gtk-editable>`) [Function]
`copy-clipboard` [Method]

Causes the characters in the current selection to be copied to the clipboard.

editable a `<gtk-editable>` widget.

`gtk-editable-paste-clipboard` (*self* `<gtk-editable>`) [Function]
`paste-clipboard` [Method]

Causes the contents of the clipboard to be pasted into the given widget at the current cursor position.

editable a `<gtk-editable>` widget.

`gtk-editable-delete-selection` (*self* `<gtk-editable>`) [Function]
`delete-selection` [Method]

Deletes the current contents of the widgets selection and disclaims the selection.

editable a `<gtk-editable>` widget.

`gtk-editable-set-position` (*self* `<gtk-editable>`) (*position* `int`) [Function]
`set-position` [Method]

Sets the cursor position.

editable a `<gtk-editable>` widget.

position the position of the cursor. The cursor is displayed before the character with the given (base 0) index in the widget. The value must be less than or equal to the number of characters in the widget. A value of -1 indicates that the position should be set after the last character in the entry. Note that this position is in characters, not in bytes.

`gtk-editable-get-position` (*self* `<gtk-editable>`) ⇒ (*ret* `int`) [Function]
`get-position` [Method]

Retrieves the current cursor position.

editable a `<gtk-editable>` widget.

ret the position of the cursor. The cursor is displayed before the character with the given (base 0) index in the widget. The value will be less than or equal to the number of characters in the widget. Note that this position is in characters, not in bytes.

`gtk-editable-set-editable` (*self* `<gtk-editable>`) [Function]
(*is_editable* `bool`)
`set-editable` [Method]

Determines if the user can edit the text in the editable widget or not.

editable a `<gtk-editable>` widget.

is-editable '`#t`' if the user is allowed to edit the text in the widget.

`gtk-editable-get-editable` (*self* `<gtk-editable>`) ⇒ (*ret* `bool`) [Function]
`get-editable` [Method]

Retrieves whether *editable* is editable. See `gtk-editable-set-editable`.

editable a `<gtk-editable>`

ret '`#t`' if *editable* is editable.

26 GtkTextIter

Text buffer iterator

26.1 Overview

You may wish to begin by reading the text widget conceptual overview which gives an overview of all the objects and data types related to the text widget and how they work together.

26.2 Usage

`<gtk-text-iter>` [Class]

`gtk-text-iter-get-buffer` (*self* `<gtk-text-iter>`) [Function]
⇒ (*ret* `<gtk-text-buffer>`)

Returns the `<gtk-text-buffer>` this iterator is associated with.

iter an iterator

ret the buffer

`gtk-text-iter-copy` (*self* `<gtk-text-iter>`) [Function]
⇒ (*ret* `<gtk-text-iter>`)

Creates a dynamically-allocated copy of an iterator. This function is not useful in applications, because iterators can be copied with a simple assignment (‘`GtkTextIter i = j;`’). The function is used by language bindings.

iter an iterator

ret a copy of the *iter*, free with `gtk-text-iter-free`

`gtk-text-iter-get-offset` (*self* `<gtk-text-iter>`) ⇒ (*ret* `int`) [Function]

Returns the character offset of an iterator. Each character in a `<gtk-text-buffer>` has an offset, starting with 0 for the first character in the buffer. Use `gtk-text-buffer-get-iter-at-offset` to convert an offset back into an iterator.

iter an iterator

ret a character offset

`gtk-text-iter-get-line` (*self* `<gtk-text-iter>`) ⇒ (*ret* `int`) [Function]

Returns the line number containing the iterator. Lines in a `<gtk-text-buffer>` are numbered beginning with 0 for the first line in the buffer.

iter an iterator

ret a line number

`gtk-text-iter-get-line-offset` (*self* `<gtk-text-iter>`) [Function]
⇒ (*ret* `int`)

Returns the character offset of the iterator, counting from the start of a newline-terminated line. The first character on the line has offset 0.

iter an iterator

ret offset from start of line

`gtk-text-iter-get-line-index` (*self* `<gtk-text-iter>`) [Function]
⇒ (*ret* `int`)

Returns the byte index of the iterator, counting from the start of a newline-terminated line. Remember that `<gtk-text-buffer>` encodes text in UTF-8, and that characters can require a variable number of bytes to represent.

iter an iterator

ret distance from start of line, in bytes

`gtk-text-iter-get-char` (*self* `<gtk-text-iter>`) [Function]
⇒ (*ret* `unsigned-int32`)

Returns the Unicode character at this iterator. (Equivalent to operator* on a C++ iterator.) If the element at this iterator is a non-character element, such as an image embedded in the buffer, the Unicode "unknown" character 0xFFFC is returned. If invoked on the end iterator, zero is returned; zero is not a valid Unicode character. So you can write a loop which ends when `gtk-text-iter-get-char` returns 0.

iter an iterator

ret a Unicode character, or 0 if *iter* is not dereferenceable

`gtk-text-iter-get-slice` (*self* `<gtk-text-iter>`) [Function]
(*end* `<gtk-text-iter>`) ⇒ (*ret* `mchars`)

Returns the text in the given range. A "slice" is an array of characters encoded in UTF-8 format, including the Unicode "unknown" character 0xFFFC for iterable non-character elements in the buffer, such as images. Because images are encoded in the slice, byte and character offsets in the returned array will correspond to byte offsets in the text buffer. Note that 0xFFFC can occur in normal text as well, so it is not a reliable indicator that a pixbuf or widget is in the buffer.

start iterator at start of a range

end iterator at end of a range

ret slice of text from the buffer

`gtk-text-iter-get-text` (*self* `<gtk-text-iter>`) [Function]
(*end* `<gtk-text-iter>`) ⇒ (*ret* `mchars`)

Returns *text* in the given range. If the range contains non-text elements such as images, the character and byte offsets in the returned string will not correspond to character and byte offsets in the buffer. If you want offsets to correspond, see `gtk-text-iter-get-slice`.

start iterator at start of a range

end iterator at end of a range

ret array of characters from the buffer

`gtk-text-iter-get-visible-slice` (*self* `<gtk-text-iter>`) [Function]
(*end* `<gtk-text-iter>`) ⇒ (*ret* `mchars`)

Like `gtk-text-iter-get-slice`, but invisible text is not included. Invisible text is usually invisible because a `<gtk-text-tag>` with the "invisible" attribute turned on has been applied to it.

start iterator at start of range

end iterator at end of range

ret slice of text from the buffer

`gtk-text-iter-get-visible-text` (*self* `<gtk-text-iter>`) [Function]
(*end* `<gtk-text-iter>`) ⇒ (*ret* `mchars`)

Like `gtk-text-iter-get-text`, but invisible text is not included. Invisible text is usually invisible because a `<gtk-text-tag>` with the "invisible" attribute turned on has been applied to it.

start iterator at start of range

end iterator at end of range

ret string containing visible text in the range

`gtk-text-iter-get-pixbuf` (*self* `<gtk-text-iter>`) [Function]
⇒ (*ret* `<gdk-pixbuf>`)

If the element at *iter* is a pixbuf, the pixbuf is returned (with no new reference count added). Otherwise, '`#f`' is returned.

iter an iterator

ret the pixbuf at *iter*

`gtk-text-iter-get-marks` (*self* `<gtk-text-iter>`) [Function]
⇒ (*ret* `gslist-of`)

Returns a list of all `<gtk-text-mark>` at this location. Because marks are not iterable (they don't take up any "space" in the buffer, they are just marks in between iterable locations), multiple marks can exist in the same place. The returned list is not in any meaningful order.

iter an iterator

ret list of `<gtk-text-mark>`

`gtk-text-iter-get-toggled-tags` (*self* `<gtk-text-iter>`) [Function]
(*toggled_on* `bool`) ⇒ (*ret* `gslist-of`)

Returns a list of `<gtk-text-tag>` that are toggled on or off at this point. (If *toggled-on* is '`#t`', the list contains tags that are toggled on.) If a tag is toggled on at *iter*, then some non-empty range of characters following *iter* has that tag applied to it. If a tag is toggled off, then some non-empty range following *iter* does *not* have the tag applied to it.

iter an iterator

toggled-on '`#t`' to get toggled-on tags

ret tags toggled at this point

`gtk-text-iter-get-child-anchor` (*self* `<gtk-text-iter>`) [Function]
⇒ (*ret* `<gtk-text-child-anchor>`)

If the location at *iter* contains a child anchor, the anchor is returned (with no new reference count added). Otherwise, '`#f`' is returned.

iter an iterator

ret the anchor at *iter*

`gtk-text-iter-begins-tag` (*self* `<gtk-text-iter>`) [Function]
(*tag* `<gtk-text-tag>`) ⇒ (*ret* `bool`)

Returns '`#t`' if *tag* is toggled on at exactly this point. If *tag* is '`#f`', returns '`#t`' if any tag is toggled on at this point. Note that the `gtk-text-iter-begins-tag` returns '`#t`' if *iter* is the *start* of the tagged range; `gtk-text-iter-has-tag` tells you whether an iterator is *within* a tagged range.

iter an iterator

tag a `<gtk-text-tag>`, or '`#f`'

ret whether *iter* is the start of a range tagged with *tag*

`gtk-text-iter-ends-tag` (*self* `<gtk-text-iter>`) [Function]
(*tag* `<gtk-text-tag>`) ⇒ (*ret* `bool`)

Returns '`#t`' if *tag* is toggled off at exactly this point. If *tag* is '`#f`', returns '`#t`' if any tag is toggled off at this point. Note that the `gtk-text-iter-ends-tag` returns '`#t`' if *iter* is the *end* of the tagged range; `gtk-text-iter-has-tag` tells you whether an iterator is *within* a tagged range.

iter an iterator

tag a `<gtk-text-tag>`, or '`#f`'

ret whether *iter* is the end of a range tagged with *tag*

`gtk-text-iter-toggles-tag` (*self* `<gtk-text-iter>`) [Function]
(*tag* `<gtk-text-tag>`) ⇒ (*ret* `bool`)

This is equivalent to (`gtk-text-iter-begins-tag || gtk-text-iter-ends-tag`), i.e. it tells you whether a range with *tag* applied to it begins *or* ends at *iter*.

iter an iterator

tag a `<gtk-text-tag>`, or '`#f`'

ret whether *tag* is toggled on or off at *iter*

`gtk-text-iter-has-tag` (*self* `<gtk-text-iter>`) [Function]
(*tag* `<gtk-text-tag>`) ⇒ (*ret* `bool`)

Returns '`#t`' if *iter* is within a range tagged with *tag*.

iter an iterator

tag a `<gtk-text-tag>`

ret whether *iter* is tagged with *tag*

`gtk-text-iter-get-tags` (*self* `<gtk-text-iter>`) [Function]
⇒ (*ret* `gslist-of`)

Returns a list of tags that apply to *iter*, in ascending order of priority (highest-priority tags are last). The `<gtk-text-tag>` in the list don't have a reference added, but you have to free the list itself.

iter a `<gtk-text-iter>`

ret list of `<gtk-text-tag>`

`gtk-text-iter-editable` (*self* `<gtk-text-iter>`) [Function]
(*default_setting* `bool`) ⇒ (*ret* `bool`)

Returns whether the character at *iter* is within an editable region of text. Non-editable text is "locked" and can't be changed by the user via `<gtk-text-view>`. This function is simply a convenience wrapper around `gtk-text-iter-get-attributes`. If no tags applied to this text affect editability, *default-setting* will be returned.

You don't want to use this function to decide whether text can be inserted at *iter*, because for insertion you don't want to know whether the char at *iter* is inside an editable range, you want to know whether a new character inserted at *iter* would be inside an editable range. Use `gtk-text-iter-can-insert` to handle this case.

iter an iterator

default-setting
'`#t`' if text is editable by default

ret whether *iter* is inside an editable range

`gtk-text-iter-can-insert` (*self* `<gtk-text-iter>`) [Function]
(*default_editability* `bool`) ⇒ (*ret* `bool`)

Considering the default editability of the buffer, and tags that affect editability, determines whether text inserted at *iter* would be editable. If text inserted at *iter* would be editable then the user should be allowed to insert text at *iter*. `gtk-text-buffer-insert-interactive` uses this function to decide whether insertions are allowed at a given position.

iter an iterator

default-editability
'`#t`' if text is editable by default

ret whether text inserted at *iter* would be editable

`gtk-text-iter-starts-word` (*self* `<gtk-text-iter>`) ⇒ (*ret* `bool`) [Function]

Determines whether *iter* begins a natural-language word. Word breaks are determined by Pango and should be correct for nearly any language (if not, the correct fix would be to the Pango word break algorithms).

iter a `<gtk-text-iter>`

ret '`#t`' if *iter* is at the start of a word

`gtk-text-iter-ends-word` (*self* `<gtk-text-iter>`) ⇒ (*ret* `bool`) [Function]

Determines whether *iter* ends a natural-language word. Word breaks are determined by Pango and should be correct for nearly any language (if not, the correct fix would be to the Pango word break algorithms).

iter a `<gtk-text-iter>`

ret '`#t`' if *iter* is at the end of a word

`gtk-text-iter-inside-word` (*self* `<gtk-text-iter>`) ⇒ (*ret* `bool`) [Function]
Determines whether *iter* is inside a natural-language word (as opposed to say inside some whitespace). Word breaks are determined by Pango and should be correct for nearly any language (if not, the correct fix would be to the Pango word break algorithms).

iter a `<gtk-text-iter>`

ret '`#t`' if *iter* is inside a word

`gtk-text-iter-starts-line` (*self* `<gtk-text-iter>`) ⇒ (*ret* `bool`) [Function]
Returns '`#t`' if *iter* begins a paragraph, i.e. if `gtk-text-iter-get-line-offset` would return 0. However this function is potentially more efficient than `gtk-text-iter-get-line-offset` because it doesn't have to compute the offset, it just has to see whether it's 0.

iter an iterator

ret whether *iter* begins a line

`gtk-text-iter-ends-line` (*self* `<gtk-text-iter>`) ⇒ (*ret* `bool`) [Function]
Returns '`#t`' if *iter* points to the start of the paragraph delimiter characters for a line (delimiters will be either a newline, a carriage return, a carriage return followed by a newline, or a Unicode paragraph separator character). Note that an iterator pointing to the \n of a \r\n pair will not be counted as the end of a line, the line ends before the \r. The end iterator is considered to be at the end of a line, even though there are no paragraph delimiter chars there.

iter an iterator

ret whether *iter* is at the end of a line

`gtk-text-iter-starts-sentence` (*self* `<gtk-text-iter>`) [Function]
⇒ (*ret* `bool`)
Determines whether *iter* begins a sentence. Sentence boundaries are determined by Pango and should be correct for nearly any language (if not, the correct fix would be to the Pango text boundary algorithms).

iter a `<gtk-text-iter>`

ret '`#t`' if *iter* is at the start of a sentence.

`gtk-text-iter-ends-sentence` (*self* `<gtk-text-iter>`) [Function]
⇒ (*ret* `bool`)
Determines whether *iter* ends a sentence. Sentence boundaries are determined by Pango and should be correct for nearly any language (if not, the correct fix would be to the Pango text boundary algorithms).

iter a `<gtk-text-iter>`

ret '`#t`' if *iter* is at the end of a sentence.

`gtk-text-iter-inside-sentence` (*self* `<gtk-text-iter>`) [Function]
⇒ (*ret* `bool`)

Determines whether *iter* is inside a sentence (as opposed to in between two sentences, e.g. after a period and before the first letter of the next sentence). Sentence boundaries are determined by Pango and should be correct for nearly any language (if not, the correct fix would be to the Pango text boundary algorithms).

iter a `<gtk-text-iter>`

ret '`#t`' if *iter* is inside a sentence.

`gtk-text-iter-is-cursor-position` (*self* `<gtk-text-iter>`) [Function]
⇒ (*ret* `bool`)

See `gtk-text-iter-forward-cursor-position` or `<pango-log-attr>` or `pango-break` for details on what a cursor position is.

iter a `<gtk-text-iter>`

ret '`#t`' if the cursor can be placed at *iter*

`gtk-text-iter-get-chars-in-line` (*self* `<gtk-text-iter>`) [Function]
⇒ (*ret* `int`)

Returns the number of characters in the line containing *iter*, including the paragraph delimiters.

iter an iterator

ret number of characters in the line

`gtk-text-iter-get-bytes-in-line` (*self* `<gtk-text-iter>`) [Function]
⇒ (*ret* `int`)

Returns the number of bytes in the line containing *iter*, including the paragraph delimiters.

iter an iterator

ret number of bytes in the line

`gtk-text-iter-get-attributes` (*self* `<gtk-text-iter>`) [Function]
(*values* `<gtk-text-attributes>`) ⇒ (*ret* `bool`)

Computes the effect of any tags applied to this spot in the text. The *values* parameter should be initialized to the default settings you wish to use if no tags are in effect. You'd typically obtain the defaults from `gtk-text-view-get-default-attributes`.

`gtk-text-iter-get-attributes` will modify *values*, applying the effects of any tags present at *iter*. If any tags affected *values*, the function returns '`#t`'.

iter an iterator

values a `<gtk-text-attributes>` to be filled in

ret '`#t`' if *values* was modified

`gtk-text-iter-get-language` (*self* `<gtk-text-iter>`) [Function]
⇒ (*ret* `<pango-language>`)

A convenience wrapper around `gtk-text-iter-get-attributes`, which returns the language in effect at *iter*. If no tags affecting language apply to *iter*, the return value is identical to that of `gtk-get-default-language`.

iter an iterator

ret language in effect at *iter*

`gtk-text-iter-is-end` (*self* `<gtk-text-iter>`) ⇒ (*ret* `bool`) [Function]

Returns '`#t`' if *iter* is the end iterator, i.e. one past the last dereferenceable iterator in the buffer. `gtk-text-iter-is-end` is the most efficient way to check whether an iterator is the end iterator.

iter an iterator

ret whether *iter* is the end iterator

`gtk-text-iter-is-start` (*self* `<gtk-text-iter>`) ⇒ (*ret* `bool`) [Function]

Returns '`#t`' if *iter* is the first iterator in the buffer, that is if *iter* has a character offset of 0.

iter an iterator

ret whether *iter* is the first in the buffer

`gtk-text-iter-forward-char` (*self* `<gtk-text-iter>`) ⇒ (*ret* `bool`) [Function]

Moves *iter* forward by one character offset. Note that images embedded in the buffer occupy 1 character slot, so `gtk-text-iter-forward-char` may actually move onto an image instead of a character, if you have images in your buffer. If *iter* is the end iterator or one character before it, *iter* will now point at the end iterator, and `gtk-text-iter-forward-char` returns '`#f`' for convenience when writing loops.

iter an iterator

ret whether *iter* moved and is dereferenceable

`gtk-text-iter-backward-char` (*self* `<gtk-text-iter>`) [Function]
⇒ (*ret* `bool`)

Moves backward by one character offset. Returns '`#t`' if movement was possible; if *iter* was the first in the buffer (character offset 0), `gtk-text-iter-backward-char` returns '`#f`' for convenience when writing loops.

iter an iterator

ret whether movement was possible

`gtk-text-iter-forward-chars` (*self* `<gtk-text-iter>`) (*count* `int`) [Function]
⇒ (*ret* `bool`)

Moves *count* characters if possible (if *count* would move past the start or end of the buffer, moves to the start or end of the buffer). The return value indicates whether the new position of *iter* is different from its original position, and dereferenceable (the last iterator in the buffer is not dereferenceable). If *count* is 0, the function does nothing and returns '`#f`'.

iter an iterator

count number of characters to move, may be negative

ret whether *iter* moved and is dereferenceable

`gtk-text-iter-backward-chars` (*self* `<gtk-text-iter>`) (*count* `int`) [Function]
⇒ (*ret* `bool`)

Moves *count* characters backward, if possible (if *count* would move past the start or end of the buffer, moves to the start or end of the buffer). The return value indicates whether the iterator moved onto a dereferenceable position; if the iterator didn't move, or moved onto the end iterator, then '`#f`' is returned. If *count* is 0, the function does nothing and returns '`#f`'.

iter an iterator

count number of characters to move

ret whether *iter* moved and is dereferenceable

`gtk-text-iter-forward-line` (*self* `<gtk-text-iter>`) ⇒ (*ret* `bool`) [Function]

Moves *iter* to the start of the next line. Returns '`#t`' if there was a next line to move to, and '`#f`' if *iter* was simply moved to the end of the buffer and is now not dereferenceable, or if *iter* was already at the end of the buffer.

iter an iterator

ret whether *iter* can be dereferenced

`gtk-text-iter-backward-line` (*self* `<gtk-text-iter>`) [Function]
⇒ (*ret* `bool`)

Moves *iter* to the start of the previous line. Returns '`#t`' if *iter* could be moved; i.e. if *iter* was at character offset 0, this function returns '`#f`'. Therefore if *iter* was already on line 0, but not at the start of the line, *iter* is snapped to the start of the line and the function returns '`#t`'. (Note that this implies that in a loop calling this function, the line number may not change on every iteration, if your first iteration is on line 0.)

iter an iterator

ret whether *iter* moved

`gtk-text-iter-forward-lines` (*self* `<gtk-text-iter>`) (*count* `int`) [Function]
⇒ (*ret* `bool`)

Moves *count* lines forward, if possible (if *count* would move past the start or end of the buffer, moves to the start or end of the buffer). The return value indicates whether the iterator moved onto a dereferenceable position; if the iterator didn't move, or moved onto the end iterator, then '`#f`' is returned. If *count* is 0, the function does nothing and returns '`#f`'. If *count* is negative, moves backward by 0 - *count* lines.

iter a `<gtk-text-iter>`

count number of lines to move forward

ret whether *iter* moved and is dereferenceable

`gtk-text-iter-backward-lines` (*self* `<gtk-text-iter>`) (*count* `int`) ⇒ (*ret* `bool`) [Function]

Moves *count* lines backward, if possible (if *count* would move past the start or end of the buffer, moves to the start or end of the buffer). The return value indicates whether the iterator moved onto a dereferenceable position; if the iterator didn't move, or moved onto the end iterator, then '`#f`' is returned. If *count* is 0, the function does nothing and returns '`#f`'. If *count* is negative, moves forward by 0 - *count* lines.

iter a `<gtk-text-iter>`

count number of lines to move backward

ret whether *iter* moved and is dereferenceable

`gtk-text-iter-forward-word-ends` (*self* `<gtk-text-iter>`) (*count* `int`) ⇒ (*ret* `bool`) [Function]

Calls `gtk-text-iter-forward-word-end` up to *count* times.

iter a `<gtk-text-iter>`

count number of times to move

ret '`#t`' if *iter* moved and is not the end iterator

`gtk-text-iter-backward-word-starts` (*self* `<gtk-text-iter>`) (*count* `int`) ⇒ (*ret* `bool`) [Function]

Calls `gtk-text-iter-backward-word-start` up to *count* times.

iter a `<gtk-text-iter>`

count number of times to move

ret '`#t`' if *iter* moved and is not the end iterator

`gtk-text-iter-forward-word-end` (*self* `<gtk-text-iter>`) ⇒ (*ret* `bool`) [Function]

Moves forward to the next word end. (If *iter* is currently on a word end, moves forward to the next one after that.) Word breaks are determined by Pango and should be correct for nearly any language (if not, the correct fix would be to the Pango word break algorithms).

iter a `<gtk-text-iter>`

ret '`#t`' if *iter* moved and is not the end iterator

`gtk-text-iter-backward-word-start` (*self* `<gtk-text-iter>`) ⇒ (*ret* `bool`) [Function]

Moves backward to the previous word start. (If *iter* is currently on a word start, moves backward to the next one after that.) Word breaks are determined by Pango and should be correct for nearly any language (if not, the correct fix would be to the Pango word break algorithms).

iter a `<gtk-text-iter>`

ret '`#t`' if *iter* moved and is not the end iterator

`gtk-text-iter-forward-sentence-end` (*self* `<gtk-text-iter>`) [Function]
⇒ (*ret* `bool`)

Moves forward to the next sentence end. (If *iter* is at the end of a sentence, moves to the next end of sentence.) Sentence boundaries are determined by Pango and should be correct for nearly any language (if not, the correct fix would be to the Pango text boundary algorithms).

iter a `<gtk-text-iter>`

ret '`#t`' if *iter* moved and is not the end iterator

`gtk-text-iter-forward-sentence-ends` (*self* `<gtk-text-iter>`) [Function]
(*count* `int`) ⇒ (*ret* `bool`)

Calls `gtk-text-iter-forward-sentence-end`*count* times (or until `gtk-text-iter-forward-sentence-end` returns '`#f`'). If *count* is negative, moves backward instead of forward.

iter a `<gtk-text-iter>`

count number of sentences to move

ret '`#t`' if *iter* moved and is not the end iterator

`gtk-text-iter-forward-visible-line` (*self* `<gtk-text-iter>`) [Function]
⇒ (*ret* `bool`)

Moves *iter* to the start of the next visible line. Returns '`#t`' if there was a next line to move to, and '`#f`' if *iter* was simply moved to the end of the buffer and is now not dereferenceable, or if *iter* was already at the end of the buffer.

iter an iterator

ret whether *iter* can be dereferenced

Since 2.8

`gtk-text-iter-backward-visible-line` (*self* `<gtk-text-iter>`) [Function]
⇒ (*ret* `bool`)

Moves *iter* to the start of the previous visible line. Returns '`#t`' if *iter* could be moved; i.e. if *iter* was at character offset 0, this function returns '`#f`'. Therefore if *iter* was already on line 0, but not at the start of the line, *iter* is snapped to the start of the line and the function returns '`#t`'. (Note that this implies that in a loop calling this function, the line number may not change on every iteration, if your first iteration is on line 0.)

iter an iterator

ret whether *iter* moved

Since 2.8

`gtk-text-iter-forward-visible-lines` (*self* `<gtk-text-iter>`) [Function]
(*count* `int`) ⇒ (*ret* `bool`)

Moves *count* visible lines forward, if possible (if *count* would move past the start or end of the buffer, moves to the start or end of the buffer). The return value

indicates whether the iterator moved onto a dereferenceable position; if the iterator didn't move, or moved onto the end iterator, then '`#f`' is returned. If *count* is 0, the function does nothing and returns '`#f`'. If *count* is negative, moves backward by 0 - *count* lines.

iter a `<gtk-text-iter>`

count number of lines to move forward

ret whether *iter* moved and is dereferenceable

Since 2.8

`gtk-text-iter-set-offset` (*self* `<gtk-text-iter>`) (*char_offset* `int`) [Function]
Sets *iter* to point to *char-offset*. *char-offset* counts from the start of the entire text buffer, starting with 0.

iter a `<gtk-text-iter>`

char-offset a character number

`gtk-text-iter-set-line` (*self* `<gtk-text-iter>`) (*line_number* `int`) [Function]
Moves iterator *iter* to the start of the line *line-number*. If *line-number* is negative or larger than the number of lines in the buffer, moves *iter* to the start of the last line in the buffer.

iter a `<gtk-text-iter>`

line-number
line number (counted from 0)

`gtk-text-iter-set-line-offset` (*self* `<gtk-text-iter>`) [Function]
(*char_on_line* `int`)
Moves *iter* within a line, to a new *character* (not byte) offset. The given character offset must be less than or equal to the number of characters in the line; if equal, *iter* moves to the start of the next line. See `gtk-text-iter-set-line-index` if you have a byte index rather than a character offset.

iter a `<gtk-text-iter>`

char-on-line
a character offset relative to the start of *iter*'s current line

`gtk-text-iter-set-line-index` (*self* `<gtk-text-iter>`) [Function]
(*byte_on_line* `int`)
Same as `gtk-text-iter-set-line-offset`, but works with a *byte* index. The given byte index must be at the start of a character, it can't be in the middle of a UTF-8 encoded character.

iter a `<gtk-text-iter>`

byte-on-line
a byte index relative to the start of *iter*'s current line

`gtk-text-iter-forward-to-end` (*self* `<gtk-text-iter>`) [Function]

Moves *iter* forward to the "end iterator," which points one past the last valid character in the buffer. `gtk-text-iter-get-char` called on the end iterator returns 0, which is convenient for writing loops.

iter a `<gtk-text-iter>`

`gtk-text-iter-forward-to-line-end` (*self* `<gtk-text-iter>`) [Function]
⇒ (*ret* `bool`)

Moves the iterator to point to the paragraph delimiter characters, which will be either a newline, a carriage return, a carriage return/newline in sequence, or the Unicode paragraph separator character. If the iterator is already at the paragraph delimiter characters, moves to the paragraph delimiter characters for the next line. If *iter* is on the last line in the buffer, which does not end in paragraph delimiters, moves to the end iterator (end of the last line), and returns '`#f`'.

iter a `<gtk-text-iter>`

ret '`#t`' if we moved and the new location is not the end iterator

`gtk-text-iter-forward-to-tag-toggle` (*self* `<gtk-text-iter>`) [Function]
(*tag* `<gtk-text-tag>`) ⇒ (*ret* `bool`)

Moves forward to the next toggle (on or off) of the `<gtk-text-tag>`*tag*, or to the next toggle of any tag if *tag* is '`#f`'. If no matching tag toggles are found, returns '`#f`', otherwise '`#t`'. Does not return toggles located at *iter*, only toggles after *iter*. Sets *iter* to the location of the toggle, or to the end of the buffer if no toggle is found.

iter a `<gtk-text-iter>`

tag a `<gtk-text-tag>`, or '`#f`'

ret whether we found a tag toggle after *iter*

`gtk-text-iter-forward-find-char` (*self* `<gtk-text-iter>`) [Function]
(*pred* `<gtk-text-char-predicate>`) (*user_data* `<gpointer>`)
(*limit* `<gtk-text-iter>`) ⇒ (*ret* `bool`)

Advances *iter*, calling *pred* on each character. If *pred* returns '`#t`', returns '`#t`' and stops scanning. If *pred* never returns '`#t`', *iter* is set to *limit* if *limit* is non-'`#f`', otherwise to the end iterator.

iter a `<gtk-text-iter>`

pred a function to be called on each character

user-data user data for *pred*

limit search limit, or '`#f`' for none

ret whether a match was found

`gtk-text-iter-backward-find-char` (*self* `<gtk-text-iter>`) [Function]
(*pred* `<gtk-text-char-predicate>`) (*user_data* `<gpointer>`)
(*limit* `<gtk-text-iter>`) ⇒ (*ret* `bool`)

Same as `gtk-text-iter-forward-find-char`, but goes backward from *iter*.

iter a `<gtk-text-iter>`

pred function to be called on each character

user-data user data for *pred*

limit search limit, or '`#f`' for none

ret whether a match was found

`gtk-text-iter-forward-search` (*self* `<gtk-text-iter>`) [Function]
(*str* `mchars`) (*flags* `<gtk-text-search-flags>`)
(*match_start* `<gtk-text-iter>`) (*match_end* `<gtk-text-iter>`)
(*limit* `<gtk-text-iter>`) ⇒ (*ret* `bool`)

Searches forward for *str*. Any match is returned by setting *match-start* to the first character of the match and *match-end* to the first character after the match. The search will not continue past *limit*. Note that a search is a linear or O(n) operation, so you may wish to use *limit* to avoid locking up your UI on large buffers.

If the `<gtk-text-search-visible-only>` flag is present, the match may have invisible text interspersed in *str*. i.e. *str* will be a possibly-noncontiguous subsequence of the matched range. similarly, if you specify `<gtk-text-search-text-only>`, the match may have pixbufs or child widgets mixed inside the matched range. If these flags are not given, the match must be exact; the special 0xFFFC character in *str* will match embedded pixbufs or child widgets.

iter start of search

str a search string

flags flags affecting how the search is done

match-start
return location for start of match, or '`#f`'

match-end
return location for end of match, or '`#f`'

limit bound for the search, or '`#f`' for the end of the buffer

ret whether a match was found

`gtk-text-iter-backward-search` (*self* `<gtk-text-iter>`) [Function]
(*str* `mchars`) (*flags* `<gtk-text-search-flags>`)
(*match_start* `<gtk-text-iter>`) (*match_end* `<gtk-text-iter>`)
(*limit* `<gtk-text-iter>`) ⇒ (*ret* `bool`)

Same as `gtk-text-iter-forward-search`, but moves backward.

iter a `<gtk-text-iter>` where the search begins

str search string

flags bitmask of flags affecting the search

match-start
return location for start of match, or '`#f`'

match-end
: return location for end of match, or '`#f`'

limit
: location of last possible *match-start*, or '`#f`' for start of buffer

ret
: whether a match was found

`gtk-text-iter-equal` (*self* `<gtk-text-iter>`) [Function]
(*rhs* `<gtk-text-iter>`) ⇒ (*ret* `bool`)

Tests whether two iterators are equal, using the fastest possible mechanism. This function is very fast; you can expect it to perform better than e.g. getting the character offset for each iterator and comparing the offsets yourself. Also, it's a bit faster than `gtk-text-iter-compare`.

lhs
: a `<gtk-text-iter>`

rhs
: another `<gtk-text-iter>`

ret
: '`#t`' if the iterators point to the same place in the buffer

`gtk-text-iter-compare` (*self* `<gtk-text-iter>`) [Function]
(*rhs* `<gtk-text-iter>`) ⇒ (*ret* `int`)

A `qsort`-style function that returns negative if *lhs* is less than *rhs*, positive if *lhs* is greater than *rhs*, and 0 if they're equal. Ordering is in character offset order, i.e. the first character in the buffer is less than the second character in the buffer.

lhs
: a `<gtk-text-iter>`

rhs
: another `<gtk-text-iter>`

ret
: -1 if *lhs* is less than *rhs*, 1 if *lhs* is greater, 0 if they are equal

`gtk-text-iter-in-range` (*self* `<gtk-text-iter>`) [Function]
(*start* `<gtk-text-iter>`) (*end* `<gtk-text-iter>`) ⇒ (*ret* `bool`)

Checks whether *iter* falls in the range [*start*, *end*). *start* and *end* must be in ascending order.

iter
: a `<gtk-text-iter>`

start
: start of range

end
: end of range

ret
: '`#t`' if *iter* is in the range

`gtk-text-iter-order` (*self* `<gtk-text-iter>`) [Function]
(*second* `<gtk-text-iter>`)

Swaps the value of *first* and *second* if *second* comes before *first* in the buffer. That is, ensures that *first* and *second* are in sequence. Most text buffer functions that take a range call this automatically on your behalf, so there's no real reason to call it yourself in those cases. There are some exceptions, such as `gtk-text-iter-in-range`, that expect a pre-sorted range.

first
: a `<gtk-text-iter>`

second
: another `<gtk-text-iter>`

27 GtkTextMark

A position in the buffer preserved across buffer modifications

27.1 Overview

You may wish to begin by reading the text widget conceptual overview which gives an overview of all the objects and data types related to the text widget and how they work together.

A `<gtk-text-mark>` is like a bookmark in a text buffer; it preserves a position in the text. You can convert the mark to an iterator using `gtk-text-buffer-get-iter-at-mark`. Unlike iterators, marks remain valid across buffer mutations, because their behavior is defined when text is inserted or deleted. When text containing a mark is deleted, the mark remains in the position originally occupied by the deleted text. When text is inserted at a mark, a mark with *left gravity* will be moved to the beginning of the newly-inserted text, and a mark with *right gravity* will be moved to the end.

"left" and "right" here refer to logical direction (left is the toward the start of the buffer); in some languages such as Hebrew the logically-leftmost text is not actually on the left when displayed.

Marks are reference counted, but the reference count only controls the validity of the memory; marks can be deleted from the buffer at any time with `gtk-text-buffer-delete-mark`. Once deleted from the buffer, a mark is essentially useless.

Marks optionally have names; these can be convenient to avoid passing the `<gtk-text-mark>` object around.

Marks are typically created using the `gtk-text-buffer-create-mark` function.

27.2 Usage

`<gtk-text-mark>` [Class]

This `<gobject>` class defines the following properties:

`name` Mark name

`left-gravity`
Whether the mark has left gravity

`gtk-text-mark-set-visible` (*self* `<gtk-text-mark>`) (*setting* `bool`) [Function]
`set-visible` [Method]

Sets the visibility of *mark*; the insertion point is normally visible, i.e. you can see it as a vertical bar. Also, the text widget uses a visible mark to indicate where a drop will occur when dragging-and-dropping text. Most other marks are not visible. Marks are not visible by default.

mark a `<gtk-text-mark>`

setting visibility of mark

`gtk-text-mark-get-visible` (*self* `<gtk-text-mark>`) ⇒ (*ret* `bool`) [Function]
`get-visible` [Method]

Returns '`#t`' if the mark is visible (i.e. a cursor is displayed for it)

mark a `<gtk-text-mark>`

ret '`#t`' if visible

`gtk-text-mark-get-deleted` (*self* `<gtk-text-mark>`) ⇒ (*ret* `bool`) [Function]
`get-deleted` [Method]

Returns '`#t`' if the mark has been removed from its buffer with `gtk-text-buffer-delete-mark`. Marks can't be used once deleted.

mark a `<gtk-text-mark>`

ret whether the mark is deleted

`gtk-text-mark-get-name` (*self* `<gtk-text-mark>`) ⇒ (*ret* `mchars`) [Function]
`get-name` [Method]

Returns the mark name; returns NULL for anonymous marks.

mark a `<gtk-text-mark>`

ret mark name

`gtk-text-mark-get-buffer` (*self* `<gtk-text-mark>`) [Function]
⇒ (*ret* `<gtk-text-buffer>`)
`get-buffer` [Method]

Gets the buffer this mark is located inside, or NULL if the mark is deleted.

mark a `<gtk-text-mark>`

ret the mark's `<gtk-text-buffer>`

`gtk-text-mark-get-left-gravity` (*self* `<gtk-text-mark>`) [Function]
⇒ (*ret* `bool`)
`get-left-gravity` [Method]

Determines whether the mark has left gravity.

mark a `<gtk-text-mark>`

ret '`#t`' if the mark has left gravity, '`#f`' otherwise

28 GtkTextBuffer

Stores attributed text for display in a

28.1 Overview

You may wish to begin by reading the text widget conceptual overview which gives an overview of all the objects and data types related to the text widget and how they work together.

28.2 Usage

`<gtk-text-buffer>` [Class]

This `<gobject>` class defines the following properties:

`tag-table`
: Text Tag Table

`text`
: Current text of the buffer

`has-selection`
: Whether the buffer has some text currently selected

`cursor-position`
: The position of the insert mark (as offset from the beginning of the buffer)

`copy-target-list`
: The list of targets this buffer supports for clipboard copying and DND source

`paste-target-list`
: The list of targets this buffer supports for clipboard pasting and DND destination

`changed` [Signal on `<gtk-text-buffer>`]

The changed signal is emitted when the content of a `<gtk-text-buffer>` has changed.

`insert-text` (*arg0* `<gtk-text-iter>`) (*arg1* `<gchararray>`) (*arg2* `<gint>`) [Signal on `<gtk-text-buffer>`]

The insert_text signal is emitted to insert text in a `<gtk-text-buffer>`. Insertion actually occurs in the default handler.

Note that if your handler runs before the default handler it must not invalidate the *location* iter (or has to revalidate it). The default signal handler revalidates it to point to the end of the inserted text.

See also: `gtk-text-buffer-insert`, `gtk-text-buffer-insert-range`.

`insert-pixbuf` (*arg0* `<gtk-text-iter>`) (*arg1* `<gdk-pixbuf>`) [Signal on `<gtk-text-buffer>`]

The insert_pixbuf signal is emitted to insert a `<gdk-pixbuf>` in a `<gtk-text-buffer>`. Insertion actually occurs in the default handler.

Note that if your handler runs before the default handler it must not invalidate the *location* iter (or has to revalidate it). The default signal handler revalidates it to be placed after the inserted *pixbuf*.

See also: `gtk-text-buffer-insert-pixbuf`.

`insert-child-anchor` (*arg0* `<gtk-text-iter>`) [Signal on `<gtk-text-buffer>`]
(*arg1* `<gtk-text-child-anchor>`)

The insert_child_anchor signal is emitted to insert a `<gtk-text-child-anchor>` in a `<gtk-text-buffer>`. Insertion actually occurs in the default handler.

Note that if your handler runs before the default handler it must not invalidate the *location* iter (or has to revalidate it). The default signal handler revalidates it to be placed after the inserted *anchor*.

See also: `gtk-text-buffer-insert-child-anchor`.

`delete-range` (*arg0* `<gtk-text-iter>`) [Signal on `<gtk-text-buffer>`]
(*arg1* `<gtk-text-iter>`)

The delete_range signal is emitted to delete a range from a `<gtk-text-buffer>`.

Note that if your handler runs before the default handler it must not invalidate the *start* and *end* iters (or has to revalidate them). The default signal handler revalidates the *start* and *end* iters to both point point to the location where text was deleted. Handlers which run after the default handler (see `g-signal-connect-after`) do not have access to the deleted text.

See also: `gtk-text-buffer-delete`.

`modified-changed` [Signal on `<gtk-text-buffer>`]

The modified_changed signal is emitted when the modified bit of a `<gtk-text-buffer>` flips.

See also: `gtk-text-buffer-set-modified`.

`mark-set` (*arg0* `<gtk-text-iter>`) [Signal on `<gtk-text-buffer>`]
(*arg1* `<gtk-text-mark>`)

The mark_set signal is emitted as notification after a `<gtk-text-mark>` is set.

See also: `gtk-text-buffer-create-mark`, `gtk-text-buffer-move-mark`.

`mark-deleted` (*arg0* `<gtk-text-mark>`) [Signal on `<gtk-text-buffer>`]

The mark_deleted signal is emitted as notification after a `<gtk-text-mark>` is deleted.

See also: `gtk-text-buffer-delete-mark`.

`apply-tag` (*arg0* `<gtk-text-tag>`) [Signal on `<gtk-text-buffer>`]
(*arg1* `<gtk-text-iter>`) (*arg2* `<gtk-text-iter>`)

The apply_tag signal is emitted to apply a tag to a range of text in a `<gtk-text-buffer>`. Applying actually occurs in the default handler.

Note that if your handler runs before the default handler it must not invalidate the *start* and *end* iters (or has to revalidate them).

See also: `gtk-text-buffer-apply-tag`, `gtk-text-buffer-insert-with-tags`, `gtk-text-buffer-insert-range`.

`remove-tag` (*arg0* `<gtk-text-tag>`) [Signal on `<gtk-text-buffer>`]
(*arg1* `<gtk-text-iter>`) (*arg2* `<gtk-text-iter>`)

The remove_tag signal is emitted to remove all occurrences of *tag* from a range of text in a `<gtk-text-buffer>`. Removal actually occurs in the default handler.

Note that if your handler runs before the default handler it must not invalidate the *start* and *end* iters (or has to revalidate them).

See also: `gtk-text-buffer-remove-tag`.

`begin-user-action` [Signal on `<gtk-text-buffer>`]

The begin_user_action signal is emitted at the beginning of a single user-visible operation on a `<gtk-text-buffer>`.

See also: `gtk-text-buffer-begin-user-action`, `gtk-text-buffer-insert-interactive`, `gtk-text-buffer-insert-range-interactive`, `gtk-text-buffer-delete-interactive`, `gtk-text-buffer-backspace`, `gtk-text-buffer-delete-selection`.

`end-user-action` [Signal on `<gtk-text-buffer>`]

The end_user_action signal is emitted at the end of a single user-visible operation `<gtk-text-buffer>`.

See also: `gtk-text-buffer-end-user-action`, `gtk-text-buffer-insert-interactive`, `gtk-text-buffer-insert-range-interactive`, `gtk-text-buffer-delete-interactive`, `gtk-text-buffer-backspace`, `gtk-text-buffer-delete-selection`, `gtk-text-buffer-backspace`.

`gtk-text-buffer-new` (*table* `<gtk-text-tag-table>`) [Function]
⇒ (*ret* `<gtk-text-buffer>`)

Creates a new text buffer.

table a tag table, or NULL to create a new one

ret a new text buffer

`gtk-text-buffer-get-line-count` (*self* `<gtk-text-buffer>`) [Function]
⇒ (*ret* `int`)

`get-line-count` [Method]

Obtains the number of lines in the buffer. This value is cached, so the function is very fast.

buffer a `<gtk-text-buffer>`

ret number of lines in the buffer

`gtk-text-buffer-get-char-count` (*self* `<gtk-text-buffer>`) [Function]
⇒ (*ret* `int`)

`get-char-count` [Method]

Gets the number of characters in the buffer; note that characters and bytes are not the same, you can't e.g. expect the contents of the buffer in string form to be this many bytes long. The character count is cached, so this function is very fast.

buffer a `<gtk-text-buffer>`

ret number of characters in the buffer

`gtk-text-buffer-get-tag-table` (*self* `<gtk-text-buffer>`) [Function]
⇒ (*ret* `<gtk-text-tag-table>`)

`get-tag-table` [Method]

Get the `<gtk-text-tag-table>` associated with this buffer.

buffer a `<gtk-text-buffer>`

ret the buffer's tag table

`gtk-text-buffer-insert` (*self* `<gtk-text-buffer>`) [Function]
(*iter* `<gtk-text-iter>`) (*stext* `scm`)

`insert` [Method]

Inserts *len* bytes of *text* at position *iter*. If *len* is -1, *text* must be nul-terminated and will be inserted in its entirety. Emits the "insert_text" signal; insertion actually occurs in the default handler for the signal. *iter* is invalidated when insertion occurs (because the buffer contents change), but the default signal handler revalidates it to point to the end of the inserted text.

buffer a `<gtk-text-buffer>`

iter a position in the buffer

text UTF-8 format text to insert

len length of text in bytes, or -1

`gtk-text-buffer-insert-at-cursor` (*self* `<gtk-text-buffer>`) [Function]
(*stext* `scm`)

`insert-at-cursor` [Method]

Simply calls `gtk-text-buffer-insert`, using the current cursor position as the insertion point.

buffer a `<gtk-text-buffer>`

text some text in UTF-8 format

len length of text, in bytes

`gtk-text-buffer-insert-interactive` (*self* `<gtk-text-buffer>`) [Function]
(*iter* `<gtk-text-iter>`) (*stext* `scm`) (*default_editable* `bool`) ⇒ (*ret* `bool`)

`insert-interactive` [Method]

Like `gtk-text-buffer-insert`, but the insertion will not occur if *iter* is at a non-editable location in the buffer. Usually you want to prevent insertions at ineditable locations if the insertion results from a user action (is interactive).

default-editable indicates the editability of text that doesn't have a tag affecting editability applied to it. Typically the result of `gtk-text-view-get-editable` is appropriate here.

buffer a `<gtk-text-buffer>`

iter a position in *buffer*

text some UTF-8 text

len length of text in bytes, or -1

default-editable
default editability of buffer

ret whether text was actually inserted

`gtk-text-buffer-insert-range` (*self* `<gtk-text-buffer>`) [Function]
(*iter* `<gtk-text-iter>`) (*start* `<gtk-text-iter>`) (*end* `<gtk-text-iter>`)
`insert-range` [Method]

Copies text, tags, and pixbufs between *start* and *end* (the order of *start* and *end* doesn't matter) and inserts the copy at *iter*. Used instead of simply getting/inserting text because it preserves images and tags. If *start* and *end* are in a different buffer from *buffer*, the two buffers must share the same tag table.

Implemented via emissions of the insert_text and apply_tag signals, so expect those.

buffer a `<gtk-text-buffer>`

iter a position in *buffer*

start a position in a `<gtk-text-buffer>`

end another position in the same buffer as *start*

`gtk-text-buffer-insert-with-tags` (*self* `<gtk-text-buffer>`) [Function]
(*iter* `<gtk-text-iter>`) (*stext* `scm`) (*tag_list* `glist-of`)
`insert-with-tags` [Method]

Inserts *text* into *buffer* at *iter*, applying the list of tags to the newly-inserted text. The last tag specified must be NULL to terminate the list. Equivalent to calling `gtk-text-buffer-insert`, then `gtk-text-buffer-apply-tag` on the inserted text; `gtk-text-buffer-insert-with-tags` is just a convenience function.

buffer a `<gtk-text-buffer>`

iter an iterator in *buffer*

text UTF-8 text

len length of *text*, or -1

first-tag first tag to apply to *text*

... NULL-terminated list of tags to apply

`gtk-text-buffer-delete` (*self* `<gtk-text-buffer>`) [Function]
(*start* `<gtk-text-iter>`) (*end* `<gtk-text-iter>`)
`delete` [Method]

Deletes text between *start* and *end*. The order of *start* and *end* is not actually relevant; `gtk-text-buffer-delete` will reorder them. This function actually emits the "delete_range" signal, and the default handler of that signal deletes the text. Because the buffer is modified, all outstanding iterators become invalid after calling this function; however, the *start* and *end* will be re-initialized to point to the location where text was deleted.

buffer a `<gtk-text-buffer>`

start a position in *buffer*

end another position in *buffer*

`gtk-text-buffer-delete-interactive` (*self* `<gtk-text-buffer>`) [Function]
(*start_iter* `<gtk-text-iter>`) (*end_iter* `<gtk-text-iter>`)
(*default_editable* `bool`) ⇒ (*ret* `bool`)

`delete-interactive` [Method]

Deletes all *editable* text in the given range. Calls `gtk-text-buffer-delete` for each editable sub-range of [*start*,*end*). *start* and *end* are revalidated to point to the location of the last deleted range, or left untouched if no text was deleted.

buffer a `<gtk-text-buffer>`

start-iter start of range to delete

end-iter end of range

default-editable
whether the buffer is editable by default

ret whether some text was actually deleted

`gtk-text-buffer-backspace` (*self* `<gtk-text-buffer>`) [Function]
(*iter* `<gtk-text-iter>`) (*interactive* `bool`) (*default_editable* `bool`)
⇒ (*ret* `bool`)

`backspace` [Method]

Performs the appropriate action as if the user hit the delete key with the cursor at the position specified by *iter*. In the normal case a single character will be deleted, but when combining accents are involved, more than one character can be deleted, and when precomposed character and accent combinations are involved, less than one character will be deleted.

Because the buffer is modified, all outstanding iterators become invalid after calling this function; however, the *iter* will be re-initialized to point to the location where text was deleted.

buffer a `<gtk-text-buffer>`

iter a position in *buffer*

interactive
whether the deletion is caused by user interaction

default-editable
whether the buffer is editable by default

ret '`#t`' if the buffer was modified

Since 2.6

`gtk-text-buffer-set-text` (*self* `<gtk-text-buffer>`) (*stext* `scm`) [Function]

`set-text` [Method]

Deletes current contents of *buffer*, and inserts *text* instead. If *len* is -1, *text* must be nul-terminated. *text* must be valid UTF-8.

buffer a `<gtk-text-buffer>`

text UTF-8 text to insert

len length of *text* in bytes

`gtk-text-buffer-get-text` (*self* `<gtk-text-buffer>`) [Function]
(*start* `<gtk-text-iter>`) (*end* `<gtk-text-iter>`)
(*include_hidden_chars* `bool`) ⇒ (*ret* `mchars`)

`get-text` [Method]

Returns the text in the range [*start*,*end*). Excludes undisplayed text (text marked with tags that set the invisibility attribute) if *include-hidden-chars* is '`#f`'. Does not include characters representing embedded images, so byte and character indexes into the returned string do *not* correspond to byte and character indexes into the buffer. Contrast with `gtk-text-buffer-get-slice`.

buffer a `<gtk-text-buffer>`

start start of a range

end end of a range

include-hidden-chars
whether to include invisible text

ret an allocated UTF-8 string

`gtk-text-buffer-get-slice` (*self* `<gtk-text-buffer>`) [Function]
(*start* `<gtk-text-iter>`) (*end* `<gtk-text-iter>`)
(*include_hidden_chars* `bool`) ⇒ (*ret* `mchars`)

`get-slice` [Method]

Returns the text in the range [*start*,*end*). Excludes undisplayed text (text marked with tags that set the invisibility attribute) if *include-hidden-chars* is '`#f`'. The returned string includes a 0xFFFC character whenever the buffer contains embedded images, so byte and character indexes into the returned string *do* correspond to byte and character indexes into the buffer. Contrast with `gtk-text-buffer-get-text`. Note that 0xFFFC can occur in normal text as well, so it is not a reliable indicator that a pixbuf or widget is in the buffer.

buffer a `<gtk-text-buffer>`

start start of a range

end end of a range

include-hidden-chars
whether to include invisible text

ret an allocated UTF-8 string

`gtk-text-buffer-insert-pixbuf` (*self* `<gtk-text-buffer>`) [Function]
(*iter* `<gtk-text-iter>`) (*pixbuf* `<gdk-pixbuf>`)

`insert-pixbuf` [Method]

Inserts an image into the text buffer at *iter*. The image will be counted as one character in character counts, and when obtaining the buffer contents as a string, will be represented by the Unicode "object replacement character" 0xFFFC. Note that the "slice" variants for obtaining portions of the buffer as a string include this character for pixbufs, but the "text" variants do not. e.g. see `gtk-text-buffer-get-slice` and `gtk-text-buffer-get-text`.

buffer a `<gtk-text-buffer>`

iter location to insert the pixbuf

pixbuf a `<gdk-pixbuf>`

`gtk-text-buffer-insert-child-anchor` (*self* `<gtk-text-buffer>`) [Function]
(*iter* `<gtk-text-iter>`) (*anchor* `<gtk-text-child-anchor>`)

`insert-child-anchor` [Method]

Inserts a child widget anchor into the text buffer at *iter*. The anchor will be counted as one character in character counts, and when obtaining the buffer contents as a string, will be represented by the Unicode "object replacement character" 0xFFFC. Note that the "slice" variants for obtaining portions of the buffer as a string include this character for child anchors, but the "text" variants do not. e.g. see `gtk-text-buffer-get-slice` and `gtk-text-buffer-get-text`. Consider `gtk-text-buffer-create-child-anchor` as a more convenient alternative to this function. The buffer will add a reference to the anchor, so you can unref it after insertion.

buffer a `<gtk-text-buffer>`

iter location to insert the anchor

anchor a `<gtk-text-child-anchor>`

`gtk-text-buffer-create-child-anchor` (*self* `<gtk-text-buffer>`) [Function]
(*iter* `<gtk-text-iter>`) ⇒ (*ret* `<gtk-text-child-anchor>`)

`create-child-anchor` [Method]

This is a convenience function which simply creates a child anchor with `gtk-text-child-anchor-new` and inserts it into the buffer with `gtk-text-buffer-insert-child-anchor`. The new anchor is owned by the buffer; no reference count is returned to the caller of `gtk-text-buffer-create-child-anchor`.

buffer a `<gtk-text-buffer>`

iter location in the buffer

ret the created child anchor

`gtk-text-buffer-create-mark` (*self* `<gtk-text-buffer>`) [Function]
(*mark_name* `mchars`) (*where* `<gtk-text-iter>`) (*left_gravity* `bool`)
⇒ (*ret* `<gtk-text-mark>`)

`create-mark` [Method]

Creates a mark at position *where*. If *mark-name* is '`#f`', the mark is anonymous; otherwise, the mark can be retrieved by name using `gtk-text-buffer-get-mark`. If a mark has left gravity, and text is inserted at the mark's current location, the mark will be moved to the left of the newly-inserted text. If the mark has right gravity (*left-gravity* = '`#f`'), the mark will end up on the right of newly-inserted text. The standard left-to-right cursor is a mark with right gravity (when you type, the cursor stays on the right side of the text you're typing).

The caller of this function does *not* own a reference to the returned `<gtk-text-mark>`, so you can ignore the return value if you like. Marks are owned by the buffer and go away when the buffer does.

Emits the "mark_set" signal as notification of the mark's initial placement.

buffer a `<gtk-text-buffer>`

mark-name name for mark, or '`#f`'

where location to place mark

left-gravity whether the mark has left gravity

ret the new `<gtk-text-mark>` object

`gtk-text-buffer-move-mark` (*self* `<gtk-text-buffer>`) (*mark* `<gtk-text-mark>`) (*where* `<gtk-text-iter>`) [Function]

`move-mark` [Method]

Moves *mark* to the new location *where*. Emits the "mark_set" signal as notification of the move.

buffer a `<gtk-text-buffer>`

mark a `<gtk-text-mark>`

where new location for *mark* in *buffer*

`gtk-text-buffer-move-mark-by-name` (*self* `<gtk-text-buffer>`) (*name* `mchars`) (*where* `<gtk-text-iter>`) [Function]

`move-mark-by-name` [Method]

Moves the mark named *name* (which must exist) to location *where*. See `gtk-text-buffer-move-mark` for details.

buffer a `<gtk-text-buffer>`

name name of a mark

where new location for mark

`gtk-text-buffer-delete-mark` (*self* `<gtk-text-buffer>`) (*mark* `<gtk-text-mark>`) [Function]

`delete-mark` [Method]

Deletes *mark*, so that it's no longer located anywhere in the buffer. Removes the reference the buffer holds to the mark, so if you haven't called `g-object-ref` on the mark, it will be freed. Even if the mark isn't freed, most operations on *mark* become invalid. There is no way to undelete a mark. `gtk-text-mark-get-deleted` will return TRUE after this function has been called on a mark; `gtk-text-mark-get-deleted` indicates that a mark no longer belongs to a buffer. The "mark_deleted" signal will be emitted as notification after the mark is deleted.

buffer a `<gtk-text-buffer>`

mark a `<gtk-text-mark>` in *buffer*

`gtk-text-buffer-delete-mark-by-name` (*self* `<gtk-text-buffer>`) (*name* `mchars`) [Function]

`delete-mark-by-name` [Method]

Deletes the mark named *name*; the mark must exist. See `gtk-text-buffer-delete-mark` for details.

buffer a `<gtk-text-buffer>`

name name of a mark in *buffer*

`gtk-text-buffer-get-mark` (*self* `<gtk-text-buffer>`) [Function]
(*name* `mchars`) ⇒ (*ret* `<gtk-text-mark>`)

`get-mark` [Method]

Returns the mark named *name* in buffer *buffer*, or NULL if no such mark exists in the buffer.

buffer a `<gtk-text-buffer>`

name a mark name

ret a `<gtk-text-mark>`, or NULL

`gtk-text-buffer-get-insert` (*self* `<gtk-text-buffer>`) [Function]
⇒ (*ret* `<gtk-text-mark>`)

`get-insert` [Method]

Returns the mark that represents the cursor (insertion point). Equivalent to calling `gtk-text-buffer-get-mark` to get the mark named "insert", but very slightly more efficient, and involves less typing.

buffer a `<gtk-text-buffer>`

ret insertion point mark

`gtk-text-buffer-get-selection-bound` (*self* `<gtk-text-buffer>`) [Function]
⇒ (*ret* `<gtk-text-mark>`)

`get-selection-bound` [Method]

Returns the mark that represents the selection bound. Equivalent to calling `gtk-text-buffer-get-mark` to get the mark named "selection_bound", but very slightly more efficient, and involves less typing.

The currently-selected text in *buffer* is the region between the "selection_bound" and "insert" marks. If "selection_bound" and "insert" are in the same place, then there is no current selection. `gtk-text-buffer-get-selection-bounds` is another convenient function for handling the selection, if you just want to know whether there's a selection and what its bounds are.

buffer a `<gtk-text-buffer>`

ret selection bound mark

`gtk-text-buffer-get-has-selection` (*self* `<gtk-text-buffer>`) [Function]
⇒ (*ret* `bool`)

`get-has-selection` [Method]

Indicates whether the buffer has some text currently selected.

buffer a `<gtk-text-buffer>`

ret '`#t`' if the there is text selected

Since 2.10

`gtk-text-buffer-place-cursor` (*self* `<gtk-text-buffer>`) [Function]
(*where* `<gtk-text-iter>`)
`place-cursor` [Method]

This function moves the "insert" and "selection_bound" marks simultaneously. If you move them to the same place in two steps with `gtk-text-buffer-move-mark`, you will temporarily select a region in between their old and new locations, which can be pretty inefficient since the temporarily-selected region will force stuff to be recalculated. This function moves them as a unit, which can be optimized.

buffer a `<gtk-text-buffer>`

where where to put the cursor

`gtk-text-buffer-select-range` (*self* `<gtk-text-buffer>`) [Function]
(*ins* `<gtk-text-iter>`) (*bound* `<gtk-text-iter>`)
`select-range` [Method]

This function moves the "insert" and "selection_bound" marks simultaneously. If you move them in two steps with `gtk-text-buffer-move-mark`, you will temporarily select a region in between their old and new locations, which can be pretty inefficient since the temporarily-selected region will force stuff to be recalculated. This function moves them as a unit, which can be optimized.

buffer a `<gtk-text-buffer>`

ins where to put the "insert" mark

bound where to put the "selection_bound" mark

Since 2.4

`gtk-text-buffer-apply-tag` (*self* `<gtk-text-buffer>`) [Function]
(*tag* `<gtk-text-tag>`) (*start* `<gtk-text-iter>`) (*end* `<gtk-text-iter>`)
`apply-tag` [Method]

Emits the "apply_tag" signal on *buffer*. The default handler for the signal applies *tag* to the given range. *start* and *end* do not have to be in order.

buffer a `<gtk-text-buffer>`

tag a `<gtk-text-tag>`

start one bound of range to be tagged

end other bound of range to be tagged

`gtk-text-buffer-remove-tag` (*self* `<gtk-text-buffer>`) [Function]
(*tag* `<gtk-text-tag>`) (*start* `<gtk-text-iter>`) (*end* `<gtk-text-iter>`)
`remove-tag` [Method]

Emits the "remove_tag" signal. The default handler for the signal removes all occurrences of *tag* from the given range. *start* and *end* don't have to be in order.

buffer a `<gtk-text-buffer>`

tag a `<gtk-text-tag>`

start one bound of range to be untagged

end other bound of range to be untagged

`gtk-text-buffer-apply-tag-by-name` (*self* `<gtk-text-buffer>`) [Function]
(*name* `mchars`) (*start* `<gtk-text-iter>`) (*end* `<gtk-text-iter>`)
`apply-tag-by-name` [Method]

Calls `gtk-text-tag-table-lookup` on the buffer's tag table to get a `<gtk-text-tag>`, then calls `gtk-text-buffer-apply-tag`.

buffer a `<gtk-text-buffer>`

name name of a named `<gtk-text-tag>`

start one bound of range to be tagged

end other bound of range to be tagged

`gtk-text-buffer-remove-tag-by-name` (*self* `<gtk-text-buffer>`) [Function]
(*name* `mchars`) (*start* `<gtk-text-iter>`) (*end* `<gtk-text-iter>`)
`remove-tag-by-name` [Method]

Calls `gtk-text-tag-table-lookup` on the buffer's tag table to get a `<gtk-text-tag>`, then calls `gtk-text-buffer-remove-tag`.

buffer a `<gtk-text-buffer>`

name name of a `<gtk-text-tag>`

start one bound of range to be untagged

end other bound of range to be untagged

`gtk-text-buffer-remove-all-tags` (*self* `<gtk-text-buffer>`) [Function]
(*start* `<gtk-text-iter>`) (*end* `<gtk-text-iter>`)
`remove-all-tags` [Method]

Removes all tags in the range between *start* and *end*. Be careful with this function; it could remove tags added in code unrelated to the code you're currently writing. That is, using this function is probably a bad idea if you have two or more unrelated code sections that add tags.

buffer a `<gtk-text-buffer>`

start one bound of range to be untagged

end other bound of range to be untagged

`gtk-text-buffer-get-iter-at-offset` (*self* `<gtk-text-buffer>`) [Function]
(*char_offset* `int`) ⇒ (*ret* `<gtk-text-iter>`)
`get-iter-at-offset` [Method]

Initializes *iter* to a position *char-offset* chars from the start of the entire buffer. If *char-offset* is -1 or greater than the number of characters in the buffer, *iter* is initialized to the end iterator, the iterator one past the last valid character in the buffer.

buffer a `<gtk-text-buffer>`

iter iterator to initialize

char-offset char offset from start of buffer, counting from 0, or -1

`gtk-text-buffer-get-iter-at-line` (*self* `<gtk-text-buffer>`) [Function]
(*line_number* `int`) ⇒ (*ret* `<gtk-text-iter>`)

`get-iter-at-line` [Method]

Initializes *iter* to the start of the given line.

buffer a `<gtk-text-buffer>`

iter iterator to initialize

line-number
line number counting from 0

`gtk-text-buffer-get-iter-at-mark` (*self* `<gtk-text-buffer>`) [Function]
(*mark* `<gtk-text-mark>`) ⇒ (*ret* `<gtk-text-iter>`)

`get-iter-at-mark` [Method]

Initializes *iter* with the current position of *mark*.

buffer a `<gtk-text-buffer>`

iter iterator to initialize

mark a `<gtk-text-mark>` in *buffer*

`gtk-text-buffer-get-start-iter` (*self* `<gtk-text-buffer>`) [Function]
⇒ (*ret* `<gtk-text-iter>`)

`get-start-iter` [Method]

Initialized *iter* with the first position in the text buffer. This is the same as using `gtk-text-buffer-get-iter-at-offset` to get the iter at character offset 0.

buffer a `<gtk-text-buffer>`

iter iterator to initialize

`gtk-text-buffer-get-end-iter` (*self* `<gtk-text-buffer>`) [Function]
⇒ (*ret* `<gtk-text-iter>`)

`get-end-iter` [Method]

Initializes *iter* with the "end iterator," one past the last valid character in the text buffer. If dereferenced with `gtk-text-iter-get-char`, the end iterator has a character value of 0. The entire buffer lies in the range from the first position in the buffer (call `gtk-text-buffer-get-start-iter` to get character position 0) to the end iterator.

buffer a `<gtk-text-buffer>`

iter iterator to initialize

`gtk-text-buffer-get-bounds` (*self* `<gtk-text-buffer>`) [Function]
⇒ (*ret* `scm`)

`get-bounds` [Method]

Retrieves the first and last iterators in the buffer, i.e. the entire buffer lies within the range [*start*,*end*).

buffer a `<gtk-text-buffer>`

start iterator to initialize with first position in the buffer

end iterator to initialize with the end iterator

`gtk-text-buffer-get-modified` (*self* `<gtk-text-buffer>`) [Function]
⇒ (*ret* `bool`)
`get-modified` [Method]

Indicates whether the buffer has been modified since the last call to `gtk-text-buffer-set-modified` set the modification flag to '`#f`'. Used for example to enable a "save" function in a text editor.

buffer a `<gtk-text-buffer>`

ret '`#t`' if the buffer has been modified

`gtk-text-buffer-set-modified` (*self* `<gtk-text-buffer>`) [Function]
(*setting* `bool`)
`set-modified` [Method]

Used to keep track of whether the buffer has been modified since the last time it was saved. Whenever the buffer is saved to disk, call gtk_text_buffer_set_modified (*buffer*, FALSE). When the buffer is modified, it will automatically toggled on the modified bit again. When the modified bit flips, the buffer emits a "modified_changed" signal.

buffer a `<gtk-text-buffer>`

setting modification flag setting

`gtk-text-buffer-delete-selection` (*self* `<gtk-text-buffer>`) [Function]
(*interactive* `bool`) (*default_editable* `bool`) ⇒ (*ret* `bool`)
`delete-selection` [Method]

Deletes the range between the "insert" and "selection_bound" marks, that is, the currently-selected text. If *interactive* is '`#t`', the editability of the selection will be considered (users can't delete uneditable text).

buffer a `<gtk-text-buffer>`

interactive
whether the deletion is caused by user interaction

default-editable
whether the buffer is editable by default

ret whether there was a non-empty selection to delete

`gtk-text-buffer-paste-clipboard` (*self* `<gtk-text-buffer>`) [Function]
(*clipboard* `<gtk-clipboard*>`) (*override_location* `<gtk-text-iter>`)
(*default_editable* `bool`)
`paste-clipboard` [Method]

Pastes the contents of a clipboard at the insertion point, or at *override-location*. (Note: pasting is asynchronous, that is, we'll ask for the paste data and return, and at some point later after the main loop runs, the paste data will be inserted.)

buffer a `<gtk-text-buffer>`

clipboard the `<gtk-clipboard>` to paste from

override-location
location to insert pasted text, or '`#f`' for at the cursor

default-editable
whether the buffer is editable by default

`gtk-text-buffer-copy-clipboard` (*self* `<gtk-text-buffer>`) [Function]
(*clipboard* `<gtk-clipboard*>`)
`copy-clipboard` [Method]
Copies the currently-selected text to a clipboard.

buffer a `<gtk-text-buffer>`

clipboard the `<gtk-clipboard>` object to copy to.

`gtk-text-buffer-cut-clipboard` (*self* `<gtk-text-buffer>`) [Function]
(*clipboard* `<gtk-clipboard*>`) (*default_editable* `bool`)
`cut-clipboard` [Method]
Copies the currently-selected text to a clipboard, then deletes said text if it's editable.

buffer a `<gtk-text-buffer>`

clipboard the `<gtk-clipboard>` object to cut to.

default-editable
default editability of the buffer

`gtk-text-buffer-begin-user-action` (*self* `<gtk-text-buffer>`) [Function]
`begin-user-action` [Method]
Called to indicate that the buffer operations between here and a call to `gtk-text-buffer-end-user-action` are part of a single user-visible operation. The operations between `gtk-text-buffer-begin-user-action` and `gtk-text-buffer-end-user-action` can then be grouped when creating an undo stack. `<gtk-text-buffer>` maintains a count of calls to `gtk-text-buffer-begin-user-action` that have not been closed with a call to `gtk-text-buffer-end-user-action`, and emits the "begin_user_action" and "end_user_action" signals only for the outermost pair of calls. This allows you to build user actions from other user actions.

The "interactive" buffer mutation functions, such as `gtk-text-buffer-insert-interactive`, automatically call begin/end user action around the buffer operations they perform, so there's no need to add extra calls if you user action consists solely of a single call to one of those functions.

buffer a `<gtk-text-buffer>`

`gtk-text-buffer-end-user-action` (*self* `<gtk-text-buffer>`) [Function]
`end-user-action` [Method]
Should be paired with a call to `gtk-text-buffer-begin-user-action`. See that function for a full explanation.

buffer a `<gtk-text-buffer>`

`gtk-text-buffer-deserialize` (*self* `<gtk-text-buffer>`) [Function]
(*content_buffer* `<gtk-text-buffer>`) (*format* `<gdk-atom>`)
(*iter* `<gtk-text-iter>`) (*data* `<guint8*>`) (*length* `size_t`) ⇒ (*ret* `bool`)
`deserialize` [Method]
This function deserializes rich text in format '`format`' and inserts it at '`iter`'.

‘format’s to be used must be registered using gtk-text-buffer-register-deserialize-format or gtk-text-buffer-register-deserialize-tagset beforehand.

register-buffer
: the <gtk-text-buffer>‘format’ is registered with

content-buffer
: the <gtk-text-buffer> to deserialize into

format
: the rich text format to use for deserializing

iter
: insertion point for the deserialized text

data
: data to deserialize

length
: length of ‘data’

error
: return loaction for a <g-error>

ret
: ‘#t’ on success, ‘#f’ otherwise.

Since 2.10

gtk-text-buffer-serialize (*self* <gtk-text-buffer>) (*content_buffer* <gtk-text-buffer>) (*format* <gdk-atom>) (*start* <gtk-text-iter>) (*end* <gtk-text-iter>) ⇒ (*ret* <guint8*>) (*length* size_t) [Function]

serialize [Method]

This function serializes the portion of text between ‘start’ and ‘end’ in the rich text format represented by ‘format’.

‘format’s to be used must be registered using gtk-text-buffer-register-serialize-format or gtk-text-buffer-register-serialize-tagset beforehand.

register-buffer
: the <gtk-text-buffer>‘format’ is registered with

content-buffer
: the <gtk-text-buffer> to serialize

format
: the rich text format to use for serializing

start
: start of block of text to serialize

end
: end of block of test to serialize

length
: return location for the length of the serialized data

ret
: the serialized data, encoded as ‘format’

Since 2.10

29 GtkTextTag

A tag that can be applied to text in a

29.1 Overview

You may wish to begin by reading the text widget conceptual overview which gives an overview of all the objects and data types related to the text widget and how they work together.

Tags should be in the `<gtk-text-tag-table>` for a given `<gtk-text-buffer>` before using them with that buffer.

`gtk-text-buffer-create-tag` is the best way to create tags. See for numerous examples.

The "invisible" property was not implemented for GTK+ 2.0; it's planned to be implemented in future releases.

29.2 Usage

`<gtk-text-tag>` [Class]

This `<gobject>` class defines the following properties:

`name` Name used to refer to the text tag. NULL for anonymous tags

`background`
Background color as a string

`foreground`
Foreground color as a string

`background-gdk`
Background color as a (possibly unallocated) GdkColor

`foreground-gdk`
Foreground color as a (possibly unallocated) GdkColor

`background-stipple`
Bitmap to use as a mask when drawing the text background

`foreground-stipple`
Bitmap to use as a mask when drawing the text foreground

`font` Font description as a string, e.g. "Sans Italic 12"

`font-desc`
Font description as a PangoFontDescription struct

`family` Name of the font family, e.g. Sans, Helvetica, Times, Monospace

`style` Font style as a PangoStyle, e.g. PANGO_STYLE_ITALIC

`variant` Font variant as a PangoVariant, e.g. PANGO_VARIANT_SMALL_CAPS

`weight` Font weight as an integer, see predefined values in PangoWeight; for example, PANGO_WEIGHT_BOLD

`stretch` Font stretch as a PangoStretch, e.g. PANGO_STRETCH_CONDENSED

`size` Font size in Pango units

`size-points`
Font size in points

`scale` Font size as a scale factor relative to the default font size. This properly adapts to theme changes etc. so is recommended. Pango predefines some scales such as PANGO_SCALE_X_LARGE

`pixels-above-lines`
Pixels of blank space above paragraphs

`pixels-below-lines`
Pixels of blank space below paragraphs

`pixels-inside-wrap`
Pixels of blank space between wrapped lines in a paragraph

`editable` Whether the text can be modified by the user

`wrap-mode`
Whether to wrap lines never, at word boundaries, or at character boundaries

`justification`
Left, right, or center justification

`direction`
Text direction, e.g. right-to-left or left-to-right

`left-margin`
Width of the left margin in pixels

`indent` Amount to indent the paragraph, in pixels

`strikethrough`
Whether to strike through the text

`right-margin`
Width of the right margin in pixels

`underline`
Style of underline for this text

`rise` Offset of text above the baseline (below the baseline if rise is negative) in Pango units

`background-full-height`
Whether the background color fills the entire line height or only the height of the tagged characters

`language` The language this text is in, as an ISO code. Pango can use this as a hint when rendering the text. If not set, an appropriate default will be used.

`tabs` Custom tabs for this text

`invisible`
: Whether this text is hidden.

`paragraph-background`
: Paragraph background color as a string

`paragraph-background-gdk`
: Paragraph background color as a (possibly unallocated) GdkColor

`accumulative-margin`
: Whether left and right margins accumulate.

`background-set`
: Whether this tag affects the background color

`foreground-set`
: Whether this tag affects the foreground color

`background-stipple-set`
: Whether this tag affects the background stipple

`foreground-stipple-set`
: Whether this tag affects the foreground stipple

`family-set`
: Whether this tag affects the font family

`style-set`
: Whether this tag affects the font style

`variant-set`
: Whether this tag affects the font variant

`weight-set`
: Whether this tag affects the font weight

`stretch-set`
: Whether this tag affects the font stretch

`size-set` Whether this tag affects the font size

`scale-set`
: Whether this tag scales the font size by a factor

`pixels-above-lines-set`
: Whether this tag affects the number of pixels above lines

`pixels-below-lines-set`
: Whether this tag affects the number of pixels above lines

`pixels-inside-wrap-set`
: Whether this tag affects the number of pixels between wrapped lines

`editable-set`
: Whether this tag affects text editability

`wrap-mode-set`
: Whether this tag affects line wrap mode

`justification-set`
: Whether this tag affects paragraph justification

`left-margin-set`
: Whether this tag affects the left margin

`indent-set`
: Whether this tag affects indentation

`strikethrough-set`
: Whether this tag affects strikethrough

`right-margin-set`
: Whether this tag affects the right margin

`underline-set`
: Whether this tag affects underlining

`rise-set` Whether this tag affects the rise

`background-full-height-set`
: Whether this tag affects background height

`language-set`
: Whether this tag affects the language the text is rendered as

`tabs-set` Whether this tag affects tabs

`invisible-set`
: Whether this tag affects text visibility

`paragraph-background-set`
: Whether this tag affects the paragraph background color

`event` (*arg0* `<gobject>`) (*arg1* `<gdk-event>`) (*arg2* `<gtk-text-iter>`) ⇒ `<gboolean>` [Signal on `<gtk-text-tag>`]

`<gtk-text-attributes>` [Class]

`gtk-text-tag-new` (*name* `mchars`) ⇒ (*ret* `<gtk-text-tag>`) [Function]

Creates a `<gtk-text-tag>`. Configure the tag using object arguments, i.e. using `g-object-set`.

name tag name, or '`#f`'

ret a new `<gtk-text-tag>`

`gtk-text-tag-get-priority` (*self* `<gtk-text-tag>`) ⇒ (*ret* `int`) [Function]
`get-priority` [Method]

Get the tag priority.

tag a `<gtk-text-tag>`

ret The tag's priority.

`gtk-text-tag-set-priority` (*self* `<gtk-text-tag>`) (*priority* `int`) [Function]
`set-priority` [Method]

Sets the priority of a `<gtk-text-tag>`. Valid priorities are start at 0 and go to one less than `gtk-text-tag-table-get-size`. Each tag in a table has a unique priority; setting the priority of one tag shifts the priorities of all the other tags in the table to maintain a unique priority for each tag. Higher priority tags "win" if two tags both set the same text attribute. When adding a tag to a tag table, it will be assigned the highest priority in the table by default; so normally the precedence of a set of tags is the order in which they were added to the table, or created with `gtk-text-buffer-create-tag`, which adds the tag to the buffer's table automatically.

tag a `<gtk-text-tag>`

priority the new priority

`gtk-text-tag-event` (*self* `<gtk-text-tag>`) (*event_object* `<gobject>`) (*event* `<gdk-event>`) (*iter* `<gtk-text-iter>`) ⇒ (*ret* `bool`) [Function]
`event` [Method]

Emits the "event" signal on the `<gtk-text-tag>`.

tag a `<gtk-text-tag>`

event-object object that received the event, such as a widget

event the event

iter location where the event was received

ret result of signal emission (whether the event was handled)

`gtk-text-attributes-new` ⇒ (*ret* `<gtk-text-attributes>`) [Function]

Creates a `<gtk-text-attributes>`, which describes a set of properties on some text.

ret a new `<gtk-text-attributes>`

`gtk-text-attributes-copy` (*self* `<gtk-text-attributes>`) ⇒ (*ret* `<gtk-text-attributes>`) [Function]

Copies *src* and returns a new `<gtk-text-attributes>`.

src a `<gtk-text-attributes>` to be copied

ret a copy of *src*

`gtk-text-attributes-copy-values` (*self* `<gtk-text-attributes>`) (*dest* `<gtk-text-attributes>`) [Function]

Copies the values from *src* to *dest* so that *dest* has the same values as *src*. Frees existing values in *dest*.

src a `<gtk-text-attributes>`

dest another `<gtk-text-attributes>`

30 GtkTextTagTable

Collection of tags that can be used together

30.1 Overview

You may wish to begin by reading the text widget conceptual overview which gives an overview of all the objects and data types related to the text widget and how they work together.

30.2 Usage

`<gtk-text-tag-table>` [Class]

This `<gobject>` class defines no properties, other than those defined by its superclasses.

`tag-changed` (*arg0* `<gtk-text-tag>`) [Signal on `<gtk-text-tag-table>`]
(*arg1* `<gboolean>`)

`tag-added` (*arg0* `<gtk-text-tag>`) [Signal on `<gtk-text-tag-table>`]

`tag-removed` (*arg0* `<gtk-text-tag>`) [Signal on `<gtk-text-tag-table>`]

`gtk-text-tag-table-new` ⇒ (*ret* `<gtk-text-tag-table>`) [Function]

Creates a new `<gtk-text-tag-table>`. The table contains no tags by default.

ret a new `<gtk-text-tag-table>`

`gtk-text-tag-table-add` (*self* `<gtk-text-tag-table>`) [Function]
(*tag* `<gtk-text-tag>`)

`add` [Method]

Add a tag to the table. The tag is assigned the highest priority in the table.

tag must not be in a tag table already, and may not have the same name as an already-added tag.

table a `<gtk-text-tag-table>`

tag a `<gtk-text-tag>`

`gtk-text-tag-table-remove` (*self* `<gtk-text-tag-table>`) [Function]
(*tag* `<gtk-text-tag>`)

`remove` [Method]

Remove a tag from the table. This will remove the table's reference to the tag, so be careful - the tag will end up destroyed if you don't have a reference to it.

table a `<gtk-text-tag-table>`

tag a `<gtk-text-tag>`

`gtk-text-tag-table-lookup` (*self* `<gtk-text-tag-table>`) [Function]
(*name* `mchars`) ⇒ (*ret* `<gtk-text-tag>`)

`lookup` [Method]

Look up a named tag.

table a `<gtk-text-tag-table>`

name name of a tag

ret The tag, or '`#f`' if none by that name is in the table.

`gtk-text-tag-table-get-size` (*self* `<gtk-text-tag-table>`) [Function]
⇒ (*ret* `int`)

`get-size` [Method]

Returns the size of the table (number of tags)

table a `<gtk-text-tag-table>`

ret number of tags in *table*

31 GtkTextView

Widget that displays a

31.1 Overview

You may wish to begin by reading the text widget conceptual overview which gives an overview of all the objects and data types related to the text widget and how they work together.

31.2 Usage

`<gtk-text-view>` [Class]

This `<gobject>` class defines the following properties:

`pixels-above-lines`
: Pixels of blank space above paragraphs

`pixels-below-lines`
: Pixels of blank space below paragraphs

`pixels-inside-wrap`
: Pixels of blank space between wrapped lines in a paragraph

`editable` Whether the text can be modified by the user

`wrap-mode`
: Whether to wrap lines never, at word boundaries, or at character boundaries

`justification`
: Left, right, or center justification

`left-margin`
: Width of the left margin in pixels

`right-margin`
: Width of the right margin in pixels

`indent` Amount to indent the paragraph, in pixels

`tabs` Custom tabs for this text

`cursor-visible`
: If the insertion cursor is shown

`buffer` The buffer which is displayed

`overwrite`
: Whether entered text overwrites existing contents

`accepts-tab`
: Whether Tab will result in a tab character being entered

`move-cursor` (*arg0* `<gtk-movement-step>`) [Signal on `<gtk-text-view>`]
(*arg1* `<gint>`) (*arg2* `<gboolean>`)
The ::move-cursor signal is a keybinding signal which gets emitted when the user initiates a cursor movement.

Applications should not connect to it, but may emit it with `g-signal-emit-by-name` if they need to control scrolling programmatically.

`copy-clipboard` [Signal on `<gtk-text-view>`]

`populate-popup` (*arg0* `<gtk-menu>`) [Signal on `<gtk-text-view>`]

`insert-at-cursor` (*arg0* `<gchararray>`) [Signal on `<gtk-text-view>`]

`delete-from-cursor` (*arg0* `<gtk-delete-type>`) [Signal on `<gtk-text-view>`]
(*arg1* `<gint>`)

`backspace` [Signal on `<gtk-text-view>`]

`cut-clipboard` [Signal on `<gtk-text-view>`]

`paste-clipboard` [Signal on `<gtk-text-view>`]

`toggle-overwrite` [Signal on `<gtk-text-view>`]

`set-scroll-adjustments` [Signal on `<gtk-text-view>`]
(*arg0* `<gtk-adjustment>`) (*arg1* `<gtk-adjustment>`)

`select-all` (*arg0* `<gboolean>`) [Signal on `<gtk-text-view>`]

`page-horizontally` (*arg0* `<gint>`) [Signal on `<gtk-text-view>`]
(*arg1* `<gboolean>`)

`move-viewport` (*arg0* `<gtk-scroll-step>`) [Signal on `<gtk-text-view>`]
(*arg1* `<gint>`)

`set-anchor` [Signal on `<gtk-text-view>`]

`toggle-cursor-visible` [Signal on `<gtk-text-view>`]
undocumented

`<gtk-text-child-anchor>` [Class]
This `<gobject>` class defines no properties, other than those defined by its superclasses.

`gtk-text-view-new-with-buffer` (*buffer* `<gtk-text-buffer>`) [Function]
⇒ (*ret* `<gtk-widget>`)
Creates a new `<gtk-text-view>` widget displaying the buffer *buffer*. One buffer can be shared among many widgets. *buffer* may be NULL to create a default buffer, in which case this function is equivalent to `gtk-text-view-new`. The text view adds its own reference count to the buffer; it does not take over an existing reference.

buffer a `<gtk-text-buffer>`

ret a new `<gtk-text-view>`.

`gtk-text-view-set-buffer` (*self* `<gtk-text-view>`) [Function]
(*buffer* `<gtk-text-buffer>`)

`set-buffer` [Method]

Sets *buffer* as the buffer being displayed by *text-view*. The previous buffer displayed by the text view is unreferenced, and a reference is added to *buffer*. If you owned a reference to *buffer* before passing it to this function, you must remove that reference yourself; `<gtk-text-view>` will not "adopt" it.

text-view a `<gtk-text-view>`

buffer a `<gtk-text-buffer>`

`gtk-text-view-get-buffer` (*self* `<gtk-text-view>`) [Function]
⇒ (*ret* `<gtk-text-buffer>`)

`get-buffer` [Method]

Returns the `<gtk-text-buffer>` being displayed by this text view. The reference count on the buffer is not incremented; the caller of this function won't own a new reference.

text-view a `<gtk-text-view>`

ret a `<gtk-text-buffer>`

`gtk-text-view-scroll-to-mark` (*self* `<gtk-text-view>`) [Function]
(*mark* `<gtk-text-mark>`) (*within_margin* `double`) (*use_align* `bool`)
(*xalign* `double`) (*yalign* `double`)

`scroll-to-mark` [Method]

Scrolls *text-view* so that *mark* is on the screen in the position indicated by *xalign* and *yalign*. An alignment of 0.0 indicates left or top, 1.0 indicates right or bottom, 0.5 means center. If *use-align* is '`#f`', the text scrolls the minimal distance to get the mark onscreen, possibly not scrolling at all. The effective screen for purposes of this function is reduced by a margin of size *within-margin*.

text-view a `<gtk-text-view>`

mark a `<gtk-text-mark>`

within-margin
margin as a [0.0,0.5) fraction of screen size

use-align whether to use alignment arguments (if '`#f`', just get the mark onscreen)

xalign horizontal alignment of mark within visible area.

yalign vertical alignment of mark within visible area

`gtk-text-view-scroll-to-iter` (*self* `<gtk-text-view>`) [Function]
(*iter* `<gtk-text-iter>`) (*within_margin* `double`) (*use_align* `bool`)
(*xalign* `double`) (*yalign* `double`) ⇒ (*ret* `bool`)

`scroll-to-iter` [Method]

Scrolls *text-view* so that *iter* is on the screen in the position indicated by *xalign* and *yalign*. An alignment of 0.0 indicates left or top, 1.0 indicates right or bottom, 0.5 means center. If *use-align* is '`#f`', the text scrolls the minimal distance to get the

mark onscreen, possibly not scrolling at all. The effective screen for purposes of this function is reduced by a margin of size *within-margin*. NOTE: This function uses the currently-computed height of the lines in the text buffer. Note that line heights are computed in an idle handler; so this function may not have the desired effect if it's called before the height computations. To avoid oddness, consider using `gtk-text-view-scroll-to-mark` which saves a point to be scrolled to after line validation.

text-view a `<gtk-text-view>`

iter a `<gtk-text-iter>`

within-margin
margin as a [0.0,0.5) fraction of screen size

use-align whether to use alignment arguments (if '`#f`', just get the mark onscreen)

xalign horizontal alignment of mark within visible area.

yalign vertical alignment of mark within visible area

ret '`#t`' if scrolling occurred

`gtk-text-view-scroll-mark-onscreen` (*self* `<gtk-text-view>`) [Function]
(*mark* `<gtk-text-mark>`)

`scroll-mark-onscreen` [Method]

Scrolls *text-view* the minimum distance such that *mark* is contained within the visible area of the widget.

text-view a `<gtk-text-view>`

mark a mark in the buffer for *text-view*

`gtk-text-view-move-mark-onscreen` (*self* `<gtk-text-view>`) [Function]
(*mark* `<gtk-text-mark>`) ⇒ (*ret* `bool`)

`move-mark-onscreen` [Method]

Moves a mark within the buffer so that it's located within the currently-visible text area.

text-view a `<gtk-text-view>`

mark a `<gtk-text-mark>`

ret '`#t`' if the mark moved (wasn't already onscreen)

`gtk-text-view-place-cursor-onscreen` (*self* `<gtk-text-view>`) [Function]
⇒ (*ret* `bool`)

`place-cursor-onscreen` [Method]

Moves the cursor to the currently visible region of the buffer, it it isn't there already.

text-view a `<gtk-text-view>`

ret TRUE if the cursor had to be moved.

`gtk-text-view-get-visible-rect` (*self* `<gtk-text-view>`) [Function]
(*visible_rect* `<gdk-rectangle>`)

`get-visible-rect` [Method]

Fills *visible-rect* with the currently-visible region of the buffer, in buffer coordinates. Convert to window coordinates with `gtk-text-view-buffer-to-window-coords`.

text-view a `<gtk-text-view>`

visible-rect
rectangle to fill

`gtk-text-view-get-iter-location` (*self* `<gtk-text-view>`) [Function]
(*iter* `<gtk-text-iter>`) (*location* `<gdk-rectangle>`)
`get-iter-location` [Method]

Gets a rectangle which roughly contains the character at *iter*. The rectangle position is in buffer coordinates; use `gtk-text-view-buffer-to-window-coords` to convert these coordinates to coordinates for one of the windows in the text view.

text-view a `<gtk-text-view>`

iter a `<gtk-text-iter>`

location bounds of the character at *iter*

`gtk-text-view-get-line-at-y` (*self* `<gtk-text-view>`) [Function]
(*target_iter* `<gtk-text-iter>`) (*y* `int`) ⇒ (*line_top* `int`)
`get-line-at-y` [Method]

Gets the `<gtk-text-iter>` at the start of the line containing the coordinate *y*. *y* is in buffer coordinates, convert from window coordinates with `gtk-text-view-window-to-buffer-coords`. If non-'`#f`', *line-top* will be filled with the coordinate of the top edge of the line.

text-view a `<gtk-text-view>`

target-iter a `<gtk-text-iter>`

y a y coordinate

line-top return location for top coordinate of the line

`gtk-text-view-get-line-yrange` (*self* `<gtk-text-view>`) [Function]
(*iter* `<gtk-text-iter>`) ⇒ (*y* `int`) (*height* `int`)
`get-line-yrange` [Method]

Gets the y coordinate of the top of the line containing *iter*, and the height of the line. The coordinate is a buffer coordinate; convert to window coordinates with `gtk-text-view-buffer-to-window-coords`.

text-view a `<gtk-text-view>`

iter a `<gtk-text-iter>`

y return location for a y coordinate

height return location for a height

`gtk-text-view-get-iter-at-location` (*self* `<gtk-text-view>`) [Function]
(*iter* `<gtk-text-iter>`) (*x* `int`) (*y* `int`)
`get-iter-at-location` [Method]

Retrieves the iterator at buffer coordinates *x* and *y*. Buffer coordinates are coordinates for the entire buffer, not just the currently-displayed portion. If you have coordinates from an event, you have to convert those to buffer coordinates with `gtk-text-view-window-to-buffer-coords`.

text-view a `<gtk-text-view>`

iter a `<gtk-text-iter>`

x x position, in buffer coordinates

y y position, in buffer coordinates

`gtk-text-view-get-window` (*self* `<gtk-text-view>`) [Function]
(*win* `<gtk-text-window-type>`) ⇒ (*ret* `<gdk-window*>`)

`get-window` [Method]

Retrieves the `<gdk-window>` corresponding to an area of the text view; possible windows include the overall widget window, child windows on the left, right, top, bottom, and the window that displays the text buffer. Windows are '`#f`' and nonexistent if their width or height is 0, and are nonexistent before the widget has been realized.

text-view a `<gtk-text-view>`

win window to get

ret a `<gdk-window>`, or '`#f`'

`gtk-text-view-get-window-type` (*self* `<gtk-text-view>`) [Function]
(*window* `<gdk-window*>`) ⇒ (*ret* `<gtk-text-window-type>`)

`get-window-type` [Method]

Usually used to find out which window an event corresponds to. If you connect to an event signal on *text-view*, this function should be called on '`event->window`' to see which window it was.

text-view a `<gtk-text-view>`

window a window type

ret the window type.

`gtk-text-view-forward-display-line` (*self* `<gtk-text-view>`) [Function]
(*iter* `<gtk-text-iter>`) ⇒ (*ret* `bool`)

`forward-display-line` [Method]

Moves the given *iter* forward by one display (wrapped) line. A display line is different from a paragraph. Paragraphs are separated by newlines or other paragraph separator characters. Display lines are created by line-wrapping a paragraph. If wrapping is turned off, display lines and paragraphs will be the same. Display lines are divided differently for each view, since they depend on the view's width; paragraphs are the same in all views, since they depend on the contents of the `<gtk-text-buffer>`.

text-view a `<gtk-text-view>`

iter a `<gtk-text-iter>`

ret '`#t`' if *iter* was moved and is not on the end iterator

`gtk-text-view-backward-display-line` (*self* `<gtk-text-view>`) [Function]
(*iter* `<gtk-text-iter>`) ⇒ (*ret* `bool`)

`backward-display-line` [Method]

Moves the given *iter* backward by one display (wrapped) line. A display line is different from a paragraph. Paragraphs are separated by newlines or other paragraph

separator characters. Display lines are created by line-wrapping a paragraph. If wrapping is turned off, display lines and paragraphs will be the same. Display lines are divided differently for each view, since they depend on the view's width; paragraphs are the same in all views, since they depend on the contents of the `<gtk-text-buffer>`.

text-view a `<gtk-text-view>`

iter a `<gtk-text-iter>`

ret '`#t`' if *iter* was moved and is not on the end iterator

`gtk-text-view-starts-display-line` (*self* `<gtk-text-view>`) [Function]
(*iter* `<gtk-text-iter>`) ⇒ (*ret* `bool`)

`starts-display-line` [Method]

Determines whether *iter* is at the start of a display line. See `gtk-text-view-forward-display-line` for an explanation of display lines vs. paragraphs.

text-view a `<gtk-text-view>`

iter a `<gtk-text-iter>`

ret '`#t`' if *iter* begins a wrapped line

`gtk-text-view-move-visually` (*self* `<gtk-text-view>`) [Function]
(*iter* `<gtk-text-iter>`) (*count* `int`) ⇒ (*ret* `bool`)

`move-visually` [Method]

Move the iterator a given number of characters visually, treating it as the strong cursor position. If *count* is positive, then the new strong cursor position will be *count* positions to the right of the old cursor position. If *count* is negative then the new strong cursor position will be *count* positions to the left of the old cursor position.

In the presence of bidirection text, the correspondence between logical and visual order will depend on the direction of the current run, and there may be jumps when the cursor is moved off of the end of a run.

text-view a `<gtk-text-view>`

iter a `<gtk-text-iter>`

count number of characters to move (negative moves left, positive moves right)

ret '`#t`' if *iter* moved and is not on the end iterator

`gtk-text-view-add-child-at-anchor` (*self* `<gtk-text-view>`) [Function]
(*child* `<gtk-widget>`) (*anchor* `<gtk-text-child-anchor>`)

`add-child-at-anchor` [Method]

Adds a child widget in the text buffer, at the given *anchor*.

text-view a `<gtk-text-view>`

child a `<gtk-widget>`

anchor a `<gtk-text-child-anchor>` in the `<gtk-text-buffer>` for *text-view*

`gtk-text-child-anchor-new` ⇒ (*ret* `<gtk-text-child-anchor>`) [Function]

Creates a new `<gtk-text-child-anchor>`. Usually you would then insert it into a `<gtk-text-buffer>` with `gtk-text-buffer-insert-child-anchor`. To perform the creation and insertion in one step, use the convenience function `gtk-text-buffer-create-child-anchor`.

ret a new `<gtk-text-child-anchor>`

`gtk-text-child-anchor-get-widgets` [Function]
(*self* `<gtk-text-child-anchor>`) ⇒ (*ret* `glist-of`)

`get-widgets` [Method]

Gets a list of all widgets anchored at this child anchor. The returned list should be freed with `g-list-free`.

anchor a `<gtk-text-child-anchor>`

ret list of widgets anchored at *anchor*

`gtk-text-child-anchor-get-deleted` [Function]
(*self* `<gtk-text-child-anchor>`) ⇒ (*ret* `bool`)

`get-deleted` [Method]

Determines whether a child anchor has been deleted from the buffer. Keep in mind that the child anchor will be unreferenced when removed from the buffer, so you need to hold your own reference (with `g-object-ref`) if you plan to use this function — otherwise all deleted child anchors will also be finalized.

anchor a `<gtk-text-child-anchor>`

ret '`#t`' if the child anchor has been deleted from its buffer

`gtk-text-view-add-child-in-window` (*self* `<gtk-text-view>`) [Function]
(*child* `<gtk-widget>`) (*which_window* `<gtk-text-window-type>`)
(*xpos* `int`) (*ypos* `int`)

`add-child-in-window` [Method]

Adds a child at fixed coordinates in one of the text widget's windows. The window must have nonzero size (see `gtk-text-view-set-border-window-size`). Note that the child coordinates are given relative to the `<gdk-window>` in question, and that these coordinates have no sane relationship to scrolling. When placing a child in `<gtk-text-window-widget>`, scrolling is irrelevant, the child floats above all scrollable areas. But when placing a child in one of the scrollable windows (border windows or text window), you'll need to compute the child's correct position in buffer coordinates any time scrolling occurs or buffer changes occur, and then call `gtk-text-view-move-child` to update the child's position. Unfortunately there's no good way to detect that scrolling has occurred, using the current API; a possible hack would be to update all child positions when the scroll adjustments change or the text buffer changes. See bug 64518 on bugzilla.gnome.org for status of fixing this issue.

text-view a `<gtk-text-view>`

child a `<gtk-widget>`

which-window
which window the child should appear in

xpos X position of child in window coordinates

ypos Y position of child in window coordinates

`gtk-text-view-move-child` (*self* `<gtk-text-view>`) (*child* `<gtk-widget>`) (*xpos* `int`) (*ypos* `int`) [Function]
`move-child` [Method]

Updates the position of a child, as for `gtk-text-view-add-child-in-window`.

text-view a `<gtk-text-view>`

child child widget already added to the text view

xpos new X position in window coordinates

ypos new Y position in window coordinates

`gtk-text-view-set-wrap-mode` (*self* `<gtk-text-view>`) (*wrap_mode* `<gtk-wrap-mode>`) [Function]
`set-wrap-mode` [Method]

Sets the line wrapping for the view.

text-view a `<gtk-text-view>`

wrap-mode a `<gtk-wrap-mode>`

`gtk-text-view-get-wrap-mode` (*self* `<gtk-text-view>`) ⇒ (*ret* `<gtk-wrap-mode>`) [Function]
`get-wrap-mode` [Method]

Gets the line wrapping for the view.

text-view a `<gtk-text-view>`

ret the line wrap setting

`gtk-text-view-set-editable` (*self* `<gtk-text-view>`) (*setting* `bool`) [Function]
`set-editable` [Method]

Sets the default editability of the `<gtk-text-view>`. You can override this default setting with tags in the buffer, using the "editable" attribute of tags.

text-view a `<gtk-text-view>`

setting whether it's editable

`gtk-text-view-get-editable` (*self* `<gtk-text-view>`) ⇒ (*ret* `bool`) [Function]
`get-editable` [Method]

Returns the default editability of the `<gtk-text-view>`. Tags in the buffer may override this setting for some ranges of text.

text-view a `<gtk-text-view>`

ret whether text is editable by default

`gtk-text-view-set-cursor-visible` (*self* `<gtk-text-view>`) [Function]
(*setting* `bool`)
`set-cursor-visible` [Method]
Toggles whether the insertion point is displayed. A buffer with no editable text probably shouldn't have a visible cursor, so you may want to turn the cursor off.

text-view a `<gtk-text-view>`

setting whether to show the insertion cursor

`gtk-text-view-get-cursor-visible` (*self* `<gtk-text-view>`) [Function]
⇒ (*ret* `bool`)
`get-cursor-visible` [Method]
Find out whether the cursor is being displayed.

text-view a `<gtk-text-view>`

ret whether the insertion mark is visible

`gtk-text-view-set-overwrite` (*self* `<gtk-text-view>`) [Function]
(*overwrite* `bool`)
`set-overwrite` [Method]
Changes the `<gtk-text-view>` overwrite mode.

text-view a `<gtk-text-view>`

overwrite '`#t`' to turn on overwrite mode, '`#f`' to turn it off

Since 2.4

`gtk-text-view-get-overwrite` (*self* `<gtk-text-view>`) [Function]
⇒ (*ret* `bool`)
`get-overwrite` [Method]
Returns whether the `<gtk-text-view>` is in overwrite mode or not.

text-view a `<gtk-text-view>`

ret whether *text-view* is in overwrite mode or not.

Since 2.4

`gtk-text-view-set-justification` (*self* `<gtk-text-view>`) [Function]
(*justification* `<gtk-justification>`)
`set-justification` [Method]
Sets the default justification of text in *text-view*. Tags in the view's buffer may override the default.

text-view a `<gtk-text-view>`

justification
justification

`gtk-text-view-get-justification` (*self* `<gtk-text-view>`) [Function]
⇒ (*ret* `<gtk-justification>`)
`get-justification` [Method]
Gets the default justification of paragraphs in *text-view*. Tags in the buffer may override the default.

text-view a `<gtk-text-view>`

ret default justification

`gtk-text-view-set-left-margin` (*self* `<gtk-text-view>`) [Function]
(*left_margin* `int`)
`set-left-margin` [Method]
Sets the default left margin for text in *text-view*. Tags in the buffer may override the default.

text-view a `<gtk-text-view>`

left-margin
left margin in pixels

`gtk-text-view-get-left-margin` (*self* `<gtk-text-view>`) [Function]
⇒ (*ret* `int`)
`get-left-margin` [Method]
Gets the default left margin size of paragraphs in the *text-view*. Tags in the buffer may override the default.

text-view a `<gtk-text-view>`

ret left margin in pixels

`gtk-text-view-set-right-margin` (*self* `<gtk-text-view>`) [Function]
(*right_margin* `int`)
`set-right-margin` [Method]
Sets the default right margin for text in the text view. Tags in the buffer may override the default.

text-view a `<gtk-text-view>`

right-margin
right margin in pixels

`gtk-text-view-get-right-margin` (*self* `<gtk-text-view>`) [Function]
⇒ (*ret* `int`)
`get-right-margin` [Method]
Gets the default right margin for text in *text-view*. Tags in the buffer may override the default.

text-view a `<gtk-text-view>`

ret right margin in pixels

`gtk-text-view-set-indent` (*self* `<gtk-text-view>`) (*indent* `int`) [Function]
`set-indent` [Method]
Sets the default indentation for paragraphs in *text-view*. Tags in the buffer may override the default.

text-view a `<gtk-text-view>`

indent indentation in pixels

`gtk-text-view-get-indent` (*self* `<gtk-text-view>`) ⇒ (*ret* `int`) [Function]
`get-indent` [Method]

Gets the default indentation of paragraphs in *text-view*. Tags in the view's buffer may override the default. The indentation may be negative.

text-view a `<gtk-text-view>`

ret number of pixels of indentation

`gtk-text-view-set-tabs` (*self* `<gtk-text-view>`) [Function]
(*tabs* `<pango-tab-array>`)
`set-tabs` [Method]

Sets the default tab stops for paragraphs in *text-view*. Tags in the buffer may override the default.

text-view a `<gtk-text-view>`

tabs tabs as a `<pango-tab-array>`

`gtk-text-view-get-tabs` (*self* `<gtk-text-view>`) [Function]
⇒ (*ret* `<pango-tab-array>`)
`get-tabs` [Method]

Gets the default tabs for *text-view*. Tags in the buffer may override the defaults. The returned array will be '`#f`' if `"standard"` (8-space) tabs are used. Free the return value with `pango-tab-array-free`.

text-view a `<gtk-text-view>`

ret copy of default tab array, or '`#f`' if `"standard"` tabs are used; must be freed with `pango-tab-array-free`.

`gtk-text-view-set-accepts-tab` (*self* `<gtk-text-view>`) [Function]
(*accepts_tab* `bool`)
`set-accepts-tab` [Method]

Sets the behavior of the text widget when the Tab key is pressed. If *accepts-tab* is '`#t`' a tab character is inserted. If *accepts-tab* is '`#f`' the keyboard focus is moved to the next widget in the focus chain.

text-view A `<gtk-text-view>`

accepts-tab
'`#t`' if pressing the Tab key should insert a tab character, '`#f`', if pressing the Tab key should move the keyboard focus.

Since 2.4

`gtk-text-view-get-accepts-tab` (*self* `<gtk-text-view>`) [Function]
⇒ (*ret* `bool`)
`get-accepts-tab` [Method]

Returns whether pressing the Tab key inserts a tab characters. `gtk-text-view-set-accepts-tab`.

text-view A `<gtk-text-view>`

ret '`#t`' if pressing the Tab key inserts a tab character, '`#f`' if pressing the Tab key moves the keyboard focus.

Since 2.4

32 GtkTreeModel

The tree interface used by

32.1 Overview

The <gtk-tree-model> interface defines a generic tree interface for use by the <gtk-tree-view> widget. It is an abstract interface, and is designed to be usable with any appropriate data structure. The programmer just has to implement this interface on their own data type for it to be viewable by a <gtk-tree-view> widget.

The model is represented as a hierarchical tree of strongly-typed, columned data. In other words, the model can be seen as a tree where every node has different values depending on which column is being queried. The type of data found in a column is determined by using the GType system (ie. <g-type-int>, <gtk-type-button>, <g-type-pointer>, etc.). The types are homogeneous per column across all nodes. It is important to note that this interface only provides a way of examining a model and observing changes. The implementation of each individual model decides how and if changes are made.

In order to make life simpler for programmers who do not need to write their own specialized model, two generic models are provided — the <gtk-tree-store> and the <gtk-list-store>. To use these, the developer simply pushes data into these models as necessary. These models provide the data structure as well as all appropriate tree interfaces. As a result, implementing drag and drop, sorting, and storing data is trivial. For the vast majority of trees and lists, these two models are sufficient.

Models are accessed on a node/column level of granularity. One can query for the value of a model at a certain node and a certain column on that node. There are two structures used to reference a particular node in a model. They are the <gtk-tree-path> and the <gtk-tree-iter> Most of the interface consists of operations on a <gtk-tree-iter>.

Here, is short for

A path is essentially a potential node. It is a location on a model that may or may not actually correspond to a node on a specific model. The <gtk-tree-path> struct can be converted into either an array of unsigned integers or a string. The string form is a list of numbers separated by a colon. Each number refers to the offset at that level. Thus, the path refers to the root node and the path refers to the fifth child of the third node.

By contrast, a <gtk-tree-iter> is a reference to a specific node on a specific model. It is a generic struct with an integer and three generic pointers. These are filled in by the model in a model-specific way. One can convert a path to an iterator by calling gtk-tree-model-get-iter. These iterators are the primary way of accessing a model and are similar to the iterators used by <gtk-text-buffer>. They are generally statically allocated on the stack and only used for a short time. The model interface defines a set of operations using them for navigating the model.

It is expected that models fill in the iterator with private data. For example, the <gtk-list-store> model, which is internally a simple linked list, stores a list node in one of the pointers. The <gtk-tree-model-sort> stores an array and an offset in two of the pointers. Additionally, there is an integer field. This field is generally filled with a unique stamp per model. This stamp is for catching errors resulting from using invalid iterators with a model.

The lifecycle of an iterator can be a little confusing at first. Iterators are expected to always be valid for as long as the model is unchanged (and doesn't emit a signal). The model is considered to own all outstanding iterators and nothing needs to be done to free them from the user's point of view. Additionally, some models guarantee that an iterator is valid for as long as the node it refers to is valid (most notably the `<gtk-tree-store>` and `<gtk-list-store>`). Although generally uninteresting, as one always has to allow for the case where iterators do not persist beyond a signal, some very important performance enhancements were made in the sort model. As a result, the `<gtk-tree-model-iters-persist>` flag was added to indicate this behavior.

To help show some common operation of a model, some examples are provided. The first example shows three ways of getting the iter at the location . While the first method shown is easier, the second is much more common, as you often get paths from callbacks.

```
/* Three ways of getting the iter pointing to the location
 */
{
  GtkTreePath *path;
  GtkTreeIter iter;
  GtkTreeIter parent_iter;

  /* get the iterator from a string */
  gtk_tree_model_get_iter_from_string (model, &iter, "3:2:5");

  /* get the iterator from a path */
  path = gtk_tree_path_new_from_string ("3:2:5");
  gtk_tree_model_get_iter (model, &iter, path);
  gtk_tree_path_free (path);


  /* walk the tree to find the iterator */
  gtk_tree_model_iter_nth_child (model, &iter, NULL, 3);
  parent_iter = iter;
  gtk_tree_model_iter_nth_child (model, &iter, &parent_iter, 2);
  parent_iter = iter;
  gtk_tree_model_iter_nth_child (model, &iter, &parent_iter, 5);
}
```

This second example shows a quick way of iterating through a list and getting a string and an integer from each row. The `populate-model` function used below is not shown, as it is specific to the `<gtk-list-store>`. For information on how to write such a function, see the `<gtk-list-store>` documentation.

```
enum
{
  STRING_COLUMN,
  INT_COLUMN,
  N_COLUMNS
```

```
};

{
  GtkTreeModel *list_store;
  GtkTreeIter iter;
  gboolean valid;
  gint row_count = 0;

  /* make a new list_store */
  list_store = gtk_list_store_new (N_COLUMNS, G_TYPE_STRING, G_TYPE_INT);

  /* Fill the list store with data */
  populate_model (list_store);

  /* Get the first iter in the list */
  valid = gtk_tree_model_get_iter_first (list_store, &iter);

  while (valid)
    {
      /* Walk through the list, reading each row */
      gchar *str_data;
      gint   int_data;

      /* Make sure you terminate calls to gtk_tree_model_get()
       * with a '-1' value
       */
      gtk_tree_model_get (list_store, &iter,
                          STRING_COLUMN, &str_data,
                          INT_COLUMN, &int_data,
                          -1);

      /* Do something with the data */
      g_print ("Row %d: (%s,%d)\n", row_count, str_data, int_data);
      g_free (str_data);

      row_count ++;
      valid = gtk_tree_model_iter_next (list_store, &iter);
    }
}
```

32.2 Usage

`<gtk-tree-model>` [Class]

This `<gobject>` class defines no properties, other than those defined by its superclasses.

`row-changed` (*arg0* `<gtk-tree-path>`) [Signal on `<gtk-tree-model>`]
(*arg1* `<gtk-tree-iter>`)

This signal is emitted when a row in the model has changed.

`row-inserted` (*arg0* `<gtk-tree-path>`) [Signal on `<gtk-tree-model>`]
(*arg1* `<gtk-tree-iter>`)

This signal is emitted when a new row has been inserted in the model.

Note that the row may still be empty at this point, since it is a common pattern to first insert an empty row, and then fill it with the desired values.

`row-has-child-toggled` (*arg0* `<gtk-tree-path>`) [Signal on `<gtk-tree-model>`]
(*arg1* `<gtk-tree-iter>`)

This signal is emitted when a row has gotten the first child row or lost its last child row.

`row-deleted` (*arg0* `<gtk-tree-path>`) [Signal on `<gtk-tree-model>`]

This signal is emitted when a row has been deleted.

Note that no iterator is passed to the signal handler, since the row is already deleted.

Implementations of GtkTreeModel must emit row-deleted *before* removing the node from its internal data structures. This is because models and views which access and monitor this model might have references on the node which need to be released in the row-deleted handler.

`rows-reordered` (*arg0* `<gtk-tree-path>`) [Signal on `<gtk-tree-model>`]
(*arg1* `<gtk-tree-iter>`) (*arg2* `<gpointer>`)

This signal is emitted when the children of a node in the `<gtk-tree-model>` have been reordered.

Note that this signal is *not* emitted when rows are reordered by DND, since this is implemented by removing and then reinserting the row.

`<gtk-tree-iter>` [Class]

`<gtk-tree-path>` [Class]

`gtk-tree-path-new-from-string` (*path* `mchars`) (*path* `mchars`) [Function]
⇒ (*ret* `<gtk-tree-path>`)

Creates a new `<gtk-tree-path>` initialized to *path*. *path* is expected to be a colon separated list of numbers. For example, the string `"10:4:0"` would create a path of depth 3 pointing to the 11th child of the root node, the 5th child of that 11th child, and the 1st child of that 5th child. If an invalid path string is passed in, '`#f`' is returned.

path The string representation of a path.

ret A newly-created `<gtk-tree-path>`, or '`#f`'

`gtk-tree-path-append-index` (*self* `<gtk-tree-path>`) (*index* `int`) [Function]

Appends a new index to a path. As a result, the depth of the path is increased.

path A `<gtk-tree-path>`.

index The index.

`gtk-tree-path-prepend-index` (*self* `<gtk-tree-path>`) (*index* `int`) [Function]

Prepends a new index to a path. As a result, the depth of the path is increased.

path A `<gtk-tree-path>`.

index The index.

`gtk-tree-path-copy` (*self* `<gtk-tree-path>`) [Function]
⇒ (*ret* `<gtk-tree-path>`)

Creates a new `<gtk-tree-path>` as a copy of *path*.

path A `<gtk-tree-path>`.

ret A new `<gtk-tree-path>`.

`gtk-tree-row-reference-new` (*model* `<gtk-tree-model>`) [Function]
(*path* `<gtk-tree-path>`) ⇒ (*ret* `<gtk-tree-row-reference*>`)

Creates a row reference based on *path*. This reference will keep pointing to the node pointed to by *path*, so long as it exists. It listens to all signals emitted by *model*, and updates its path appropriately. If *path* isn't a valid path in *model*, then '`#f`' is returned.

model A `<gtk-tree-model>`

path A valid `<gtk-tree-path>` to monitor

ret A newly allocated `<gtk-tree-row-reference>`, or '`#f`'

`gtk-tree-row-reference-new-proxy` (*proxy* `<gobject>`) [Function]
(*model* `<gtk-tree-model>`) (*path* `<gtk-tree-path>`)
⇒ (*ret* `<gtk-tree-row-reference*>`)

You do not need to use this function. Creates a row reference based on *path*. This reference will keep pointing to the node pointed to by *path*, so long as it exists. If *path* isn't a valid path in *model*, then '`#f`' is returned. However, unlike references created with `gtk-tree-row-reference-new`, it does not listen to the model for changes. The creator of the row reference must do this explicitly using `gtk-tree-row-reference-inserted`, `gtk-tree-row-reference-deleted`, `gtk-tree-row-reference-reordered`.

These functions must be called exactly once per proxy when the corresponding signal on the model is emitted. This single call updates all row references for that proxy. Since built-in GTK+ objects like `<gtk-tree-view>` already use this mechanism internally, using them as the proxy object will produce unpredictable results. Further more, passing the same object as *model* and *proxy* doesn't work for reasons of internal implementation.

This type of row reference is primarily meant by structures that need to carefully monitor exactly when a row reference updates itself, and is not generally needed by most applications.

proxy A proxy `<gobject>`

model A `<gtk-tree-model>`

path A valid `<gtk-tree-path>` to monitor

ret A newly allocated `<gtk-tree-row-reference>`, or '`#f`'

`gtk-tree-row-reference-get-model` [Function]
(*self* `<gtk-tree-row-reference*>`) ⇒ (*ret* `<gtk-tree-model>`)
Returns the model that the row reference is monitoring.

reference A `<gtk-tree-row-reference>`

ret the model

Since 2.8

`gtk-tree-row-reference-get-path` [Function]
(*self* `<gtk-tree-row-reference*>`) ⇒ (*ret* `<gtk-tree-path>`)
Returns a path that the row reference currently points to, or '`#f`' if the path pointed to is no longer valid.

reference A `<gtk-tree-row-reference>`

ret A current path, or '`#f`'.

`gtk-tree-row-reference-valid` (*self* `<gtk-tree-row-reference*>`) [Function]
⇒ (*ret* `bool`)
Returns '`#t`' if the *reference* is non-'`#f`' and refers to a current valid path.

reference A `<gtk-tree-row-reference>`, or '`#f`'

ret '`#t`' if *reference* points to a valid path.

`gtk-tree-row-reference-inserted` (*proxy* `<gobject>`) [Function]
(*path* `<gtk-tree-path>`)
Lets a set of row reference created by `gtk-tree-row-reference-new-proxy` know that the model emitted the "row_inserted" signal.

proxy A `<gobject>`

path The row position that was inserted

`gtk-tree-row-reference-deleted` (*proxy* `<gobject>`) [Function]
(*path* `<gtk-tree-path>`)
Lets a set of row reference created by `gtk-tree-row-reference-new-proxy` know that the model emitted the "row_deleted" signal.

proxy A `<gobject>`

path The path position that was deleted

`gtk-tree-row-reference-reordered` (*proxy* `<gobject>`) [Function]
(*path* `<gtk-tree-path>`) (*iter* `<gtk-tree-iter>`) ⇒ (*new_order* `int`)
Lets a set of row reference created by `gtk-tree-row-reference-new-proxy` know that the model emitted the "rows_reordered" signal.

proxy A `<gobject>`

path The parent path of the reordered signal

iter The iter pointing to the parent of the reordered

new-order The new order of rows

`gtk-tree-iter-copy` (*self* `<gtk-tree-iter>`) [Function]
⇒ (*ret* `<gtk-tree-iter>`)

Creates a dynamically allocated tree iterator as a copy of *iter*. This function is not intended for use in applications, because you can just copy the structs by value (‘`GtkTreeIter new_iter = iter;`’). You must free this iter with `gtk-tree-iter-free`.

iter A `<gtk-tree-iter>`.

ret a newly-allocated copy of *iter*.

`gtk-tree-model-get-flags` (*self* `<gtk-tree-model>`) [Function]
⇒ (*ret* `<gtk-tree-model-flags>`)

`get-flags` [Method]

Returns a set of flags supported by this interface. The flags are a bitwise combination of `<gtk-tree-model-flags>`. The flags supported should not change during the lifecycle of the *tree-model*.

tree-model
A `<gtk-tree-model>`.

ret The flags supported by this interface.

`gtk-tree-model-get-n-columns` (*self* `<gtk-tree-model>`) [Function]
⇒ (*ret* `int`)

`get-n-columns` [Method]

Returns the number of columns supported by *tree-model*.

tree-model
A `<gtk-tree-model>`.

ret The number of columns.

`gtk-tree-model-get-column-type` (*self* `<gtk-tree-model>`) [Function]
(*index* `int`) ⇒ (*ret* `<gtype>`)

`get-column-type` [Method]

Returns the type of the column.

tree-model
A `<gtk-tree-model>`.

index The column index.

ret The type of the column.

`gtk-tree-model-get-iter` (*self* `<gtk-tree-model>`) [Function]
(*path* `<gtk-tree-path>`) ⇒ (*ret* `<gtk-tree-iter>`)

`get-iter` [Method]

Sets *iter* to a valid iterator pointing to *path*.

tree-model
A `<gtk-tree-model>`.

iter The uninitialized `<gtk-tree-iter>`.

path The `<gtk-tree-path>`.

ret '`#t`', if *iter* was set.

`gtk-tree-model-get-iter-first` (*self* `<gtk-tree-model>`) [Function]
⇒ (*ret* `<gtk-tree-iter>`)

`get-iter-first` [Method]

Initializes *iter* with the first iterator in the tree (the one at the path `"0"`) and returns '`#t`'. Returns '`#f`' if the tree is empty.

tree-model
A `<gtk-tree-model>`.

iter The uninitialized `<gtk-tree-iter>`.

ret '`#t`', if *iter* was set.

`gtk-tree-model-get-path` (*self* `<gtk-tree-model>`) [Function]
(*iter* `<gtk-tree-iter>`) ⇒ (*ret* `<gtk-tree-path>`)

`get-path` [Method]

Returns a newly-created `<gtk-tree-path>` referenced by *iter*. This path should be freed with `gtk-tree-path-free`.

tree-model
A `<gtk-tree-model>`.

iter The `<gtk-tree-iter>`.

ret a newly-created `<gtk-tree-path>`.

`gtk-tree-model-get-value` (*self* `<gtk-tree-model>`) [Function]
(*iter* `<gtk-tree-iter>`) (*column* `int`) ⇒ (*ret* `scm`)

`get-value` [Method]

Sets initializes and sets *value* to that at *column*. When done with *value*, `g-value-unset` needs to be called to free any allocated memory.

tree-model
A `<gtk-tree-model>`.

iter The `<gtk-tree-iter>`.

column The column to lookup the value at.

value An empty `<gvalue>` to set.

`gtk-tree-model-iter-next` (*self* `<gtk-tree-model>`) [Function]
(*iter* `<gtk-tree-iter>`) ⇒ (*ret* `<gtk-tree-iter>`)

`iter-next` [Method]

Sets *iter* to point to the node following it at the current level. If there is no next *iter*, '`#f`' is returned and *iter* is set to be invalid.

tree-model
A `<gtk-tree-model>`.

iter The `<gtk-tree-iter>`.

ret '`#t`' if *iter* has been changed to the next node.

`gtk-tree-model-iter-children` (*self* `<gtk-tree-model>`) [Function]
(*parent* `<gtk-tree-iter>`) ⇒ (*ret* `glist-of`)
`iter-children` [Method]

Sets *iter* to point to the first child of *parent*. If *parent* has no children, '`#f`' is returned and *iter* is set to be invalid. *parent* will remain a valid node after this function has been called.

If *parent* is '`#f`' returns the first node, equivalent to '`gtk_tree_model_get_iter_first (tree_model, iter);`'

tree-model
A `<gtk-tree-model>`.

iter The new `<gtk-tree-iter>` to be set to the child.

parent The `<gtk-tree-iter>`, or '`#f`'

ret '`#t`', if *child* has been set to the first child.

`gtk-tree-model-iter-has-child` (*self* `<gtk-tree-model>`) [Function]
(*iter* `<gtk-tree-iter>`) ⇒ (*ret* `bool`)
`iter-has-child` [Method]

Returns '`#t`' if *iter* has children, '`#f`' otherwise.

tree-model
A `<gtk-tree-model>`.

iter The `<gtk-tree-iter>` to test for children.

ret '`#t`' if *iter* has children.

`gtk-tree-model-iter-n-children` (*self* `<gtk-tree-model>`) [Function]
(*iter* `<gtk-tree-iter>`) ⇒ (*ret* `int`)
`iter-n-children` [Method]

Returns the number of children that *iter* has. As a special case, if *iter* is '`#f`', then the number of toplevel nodes is returned.

tree-model
A `<gtk-tree-model>`.

iter The `<gtk-tree-iter>`, or '`#f`'.

ret The number of children of *iter*.

`gtk-tree-model-iter-nth-child` (*self* `<gtk-tree-model>`) [Function]
(*parent* `<gtk-tree-iter>`) (*n* `int`) ⇒ (*ret* `<gtk-tree-iter>`)
`iter-nth-child` [Method]

Sets *iter* to be the child of *parent*, using the given index. The first index is 0. If *n* is too big, or *parent* has no children, *iter* is set to an invalid iterator and '`#f`' is returned. *parent* will remain a valid node after this function has been called. As a special case, if *parent* is '`#f`', then the *n*th root node is set.

tree-model
A `<gtk-tree-model>`.

iter The <gtk-tree-iter> to set to the nth child.

parent The <gtk-tree-iter> to get the child from, or '#f'.

n Then index of the desired child.

ret '#t', if *parent* has an *n*th child.

gtk-tree-model-iter-parent (*self* <gtk-tree-model>) [Function]
(*child* <gtk-tree-iter>) ⇒ (*ret* <gtk-tree-iter>)

iter-parent [Method]

Sets *iter* to be the parent of *child*. If *child* is at the toplevel, and doesn't have a parent, then *iter* is set to an invalid iterator and '#f' is returned. *child* will remain a valid node after this function has been called.

tree-model
A <gtk-tree-model>

iter The new <gtk-tree-iter> to set to the parent.

child The <gtk-tree-iter>.

ret '#t', if *iter* is set to the parent of *child*.

gtk-tree-model-get-string-from-iter (*self* <gtk-tree-model>) [Function]
(*iter* <gtk-tree-iter>) ⇒ (*ret* mchars)

get-string-from-iter [Method]

Generates a string representation of the iter. This string is a ':' separated list of numbers. For example, "4:10:0:3" would be an acceptable return value for this string.

tree-model
A <gtk-tree-model>.

iter An <gtk-tree-iter>.

ret A newly-allocated string. Must be freed with g-free.

Since 2.2

gtk-tree-model-ref-node (*self* <gtk-tree-model>) [Function]
(*iter* <gtk-tree-iter>)

ref-node [Method]

Lets the tree ref the node. This is an optional method for models to implement. To be more specific, models may ignore this call as it exists primarily for performance reasons.

This function is primarily meant as a way for views to let caching model know when nodes are being displayed (and hence, whether or not to cache that node.) For example, a file-system based model would not want to keep the entire file-hierarchy in memory, just the sections that are currently being displayed by every current view.

A model should be expected to be able to get an iter independent of its reffed state.

tree-model
A <gtk-tree-model>.

iter The <gtk-tree-iter>.

`gtk-tree-model-unref-node` (*self* `<gtk-tree-model>`) [Function]
(*iter* `<gtk-tree-iter>`)
`unref-node` [Method]

Lets the tree unref the node. This is an optional method for models to implement. To be more specific, models may ignore this call as it exists primarily for performance reasons.

For more information on what this means, see `gtk-tree-model-ref-node`. Please note that nodes that are deleted are not unreffed.

tree-model A `<gtk-tree-model>`.

iter The `<gtk-tree-iter>`.

`gtk-tree-model-row-changed` (*self* `<gtk-tree-model>`) [Function]
(*path* `<gtk-tree-path>`) (*iter* `<gtk-tree-iter>`)
`row-changed` [Method]

Emits the "row_changed" signal on *tree-model*.

tree-model A `<gtk-tree-model>`

path A `<gtk-tree-path>` pointing to the changed row

iter A valid `<gtk-tree-iter>` pointing to the changed row

`gtk-tree-model-row-inserted` (*self* `<gtk-tree-model>`) [Function]
(*path* `<gtk-tree-path>`) (*iter* `<gtk-tree-iter>`)
`row-inserted` [Method]

Emits the "row_inserted" signal on *tree-model*

tree-model A `<gtk-tree-model>`

path A `<gtk-tree-path>` pointing to the inserted row

iter A valid `<gtk-tree-iter>` pointing to the inserted row

`gtk-tree-model-row-deleted` (*self* `<gtk-tree-model>`) [Function]
(*path* `<gtk-tree-path>`)
`row-deleted` [Method]

Emits the "row_deleted" signal on *tree-model*. This should be called by models after a row has been removed. The location pointed to by *path* should be the location that the row previously was at. It may not be a valid location anymore.

tree-model A `<gtk-tree-model>`

path A `<gtk-tree-path>` pointing to the previous location of the deleted row.

`gtk-tree-model-rows-reordered` (*self* `<gtk-tree-model>`) [Function]
(*path* `<gtk-tree-path>`) (*iter* `<gtk-tree-iter>`) ⇒ (*new_order* `int`)
`rows-reordered` [Method]

Emits the "rows_reordered" signal on *tree-model*. This should be called by models when their rows have been reordered.

tree-model
: A `<gtk-tree-model>`

path
: A `<gtk-tree-path>` pointing to the tree node whose children have been reordered

iter
: A valid `<gtk-tree-iter>` pointing to the node whose children have been reordered, or '`#f`' if the depth of *path* is 0.

new-order
: an array of integers mapping the current position of each child to its old position before the re-ordering, i.e. *new-order*'`[newpos] = oldpos`'.

33 GtkTreeSelection

The selection object for

33.1 Overview

The <gtk-tree-selection> object is a helper object to manage the selection for a <gtk-tree-view> widget. The <gtk-tree-selection> object is automatically created when a new <gtk-tree-view> widget is created, and cannot exist independentally of this widget. The primary reason the <gtk-tree-selection> objects exists is for cleanliness of code and API. That is, there is no conceptual reason all these functions could not be methods on the <gtk-tree-view> widget instead of a separate function.

The <gtk-tree-selection> object is gotten from a <gtk-tree-view> by calling gtk-tree-view-get-selection. It can be manipulated to check the selection status of the tree, as well as select and deselect individual rows. Selection is done completely view side. As a result, multiple views of the same model can have completely different selections. Additionally, you cannot change the selection of a row on the model that is not currently displayed by the view without expanding its parents first.

One of the important things to remember when monitoring the selection of a view is that the "changed" signal is mostly a hint. That is, it may only emit one signal when a range of rows is selected. Additionally, it may on occasion emit a "changed" signal when nothing has happened (mostly as a result of programmers calling select_row on an already selected row).

33.2 Usage

<gtk-tree-selection> [Class]

This <gobject> class defines no properties, other than those defined by its superclasses.

changed [Signal on <gtk-tree-selection>]

Emitted whenever the selection has (possibly) changed. Please note that this signal is mostly a hint. It may only be emitted once when a range of rows are selected, and it may occasionally be emitted when nothing has happened.

gtk-tree-selection-set-mode (*self* <gtk-tree-selection>) (*type* <gtk-selection-mode>) [Function]

set-mode [Method]

Sets the selection mode of the *selection*. If the previous type was <gtk-selection-multiple>, then the anchor is kept selected, if it was previously selected.

selection A <gtk-tree-selection>.

type The selection mode

gtk-tree-selection-get-mode (*self* <gtk-tree-selection>) ⇒ (*ret* <gtk-selection-mode>) [Function]

get-mode [Method]

Gets the selection mode for *selection*. See gtk-tree-selection-set-mode.

selection a <gtk-tree-selection>

ret the current selection mode

`gtk-tree-selection-get-user-data` (*self* <gtk-tree-selection>) [Function]
⇒ (*ret* <gpointer>)
`get-user-data` [Method]
Returns the user data for the selection function.

selection A <gtk-tree-selection>.

ret The user data.

`gtk-tree-selection-get-tree-view` (*self* <gtk-tree-selection>) [Function]
⇒ (*ret* <gtk-tree-view>)
`get-tree-view` [Method]
Returns the tree view associated with *selection*.

selection A <gtk-tree-selection>

ret A <gtk-tree-view>

`gtk-tree-selection-get-selected` (*self* <gtk-tree-selection>) [Function]
⇒ (*ret* scm)
`get-selected` [Method]
Sets *iter* to the currently selected node if *selection* is set to <gtk-selection-single> or <gtk-selection-browse>. *iter* may be NULL if you just want to test if *selection* has any selected nodes. *model* is filled with the current model as a convenience. This function will not work if you use *selection* is <gtk-selection-multiple>.

selection A <gtk-tree-selection>.

model A pointer to set to the <gtk-tree-model>, or NULL.

iter The <gtk-tree-iter>, or NULL.

ret TRUE, if there is a selected node.

`gtk-tree-selection-select-path` (*self* <gtk-tree-selection>) [Function]
(*path* <gtk-tree-path>)
`select-path` [Method]
Select the row at *path*.

selection A <gtk-tree-selection>.

path The <gtk-tree-path> to be selected.

`gtk-tree-selection-unselect-path` (*self* <gtk-tree-selection>) [Function]
(*path* <gtk-tree-path>)
`unselect-path` [Method]
Unselects the row at *path*.

selection A <gtk-tree-selection>.

path The <gtk-tree-path> to be unselected.

`gtk-tree-selection-path-is-selected` [Function]
(*self* `<gtk-tree-selection>`) (*path* `<gtk-tree-path>`) ⇒ (*ret* `bool`)
`path-is-selected` [Method]

Returns '`#t`' if the row pointed to by *path* is currently selected. If *path* does not point to a valid location, '`#f`' is returned

selection A `<gtk-tree-selection>`.

path A `<gtk-tree-path>` to check selection on.

ret '`#t`' if *path* is selected.

`gtk-tree-selection-select-iter` (*self* `<gtk-tree-selection>`) [Function]
(*iter* `<gtk-tree-iter>`)
`select-iter` [Method]

Selects the specified iterator.

selection A `<gtk-tree-selection>`.

iter The `<gtk-tree-iter>` to be selected.

`gtk-tree-selection-unselect-iter` (*self* `<gtk-tree-selection>`) [Function]
(*iter* `<gtk-tree-iter>`)
`unselect-iter` [Method]

Unselects the specified iterator.

selection A `<gtk-tree-selection>`.

iter The `<gtk-tree-iter>` to be unselected.

`gtk-tree-selection-iter-is-selected` [Function]
(*self* `<gtk-tree-selection>`) (*iter* `<gtk-tree-iter>`) ⇒ (*ret* `bool`)
`iter-is-selected` [Method]

Returns '`#t`' if the row at *iter* is currently selected.

selection A `<gtk-tree-selection>`

iter A valid `<gtk-tree-iter>`

ret '`#t`', if *iter* is selected

`gtk-tree-selection-select-all` (*self* `<gtk-tree-selection>`) [Function]
`select-all` [Method]

Selects all the nodes. *selection* must be set to `<gtk-selection-multiple>` mode.

selection A `<gtk-tree-selection>`.

`gtk-tree-selection-unselect-all` (*self* `<gtk-tree-selection>`) [Function]
`unselect-all` [Method]

Unselects all the nodes.

selection A `<gtk-tree-selection>`.

`gtk-tree-selection-select-range` (*self* `<gtk-tree-selection>`) [Function]
(*start_path* `<gtk-tree-path>`) (*end_path* `<gtk-tree-path>`)
`select-range` [Method]

Selects a range of nodes, determined by *start-path* and *end-path* inclusive. *selection* must be set to `<gtk-selection-multiple>` mode.

selection A `<gtk-tree-selection>`.

start-path The initial node of the range.

end-path The final node of the range.

`gtk-tree-selection-unselect-range` (*self* `<gtk-tree-selection>`) [Function]
(*start_path* `<gtk-tree-path>`) (*end_path* `<gtk-tree-path>`)
`unselect-range` [Method]

Unselects a range of nodes, determined by *start-path* and *end-path* inclusive.

selection A `<gtk-tree-selection>`.

start-path The initial node of the range.

end-path The initial node of the range.

Since 2.2

34 GtkTreeViewColumn

A visible column in a widget

34.1 Overview

The GtkTreeViewColumn object represents a visible column in a `<gtk-tree-view>` widget. It allows to set properties of the column header, and functions as a holding pen for the cell renderers which determine how the data in the column is displayed.

Please refer to the tree widget conceptual overview for an overview of all the objects and data types related to the tree widget and how they work together.

34.2 Usage

`<gtk-tree-view-column>` [Class]

This `<gobject>` class defines the following properties:

`visible` Whether to display the column

`resizable`
Column is user-resizable

`width` Current width of the column

`spacing` Space which is inserted between cells

`sizing` Resize mode of the column

`fixed-width`
Current fixed width of the column

`min-width`
Minimum allowed width of the column

`max-width`
Maximum allowed width of the column

`title` Title to appear in column header

`expand` Column gets share of extra width allocated to the widget

`clickable`
Whether the header can be clicked

`widget` Widget to put in column header button instead of column title

`alignment`
X Alignment of the column header text or widget

`reorderable`
Whether the column can be reordered around the headers

`sort-indicator`
Whether to show a sort indicator

`sort-order`
Sort direction the sort indicator should indicate

`clicked` [Signal on `<gtk-tree-view-column>`]

`gtk-tree-view-column-new` ⇒ (*ret* `<gtk-tree-view-column>`) [Function]
Creates a new `<gtk-tree-view-column>`.

ret A newly created `<gtk-tree-view-column>`.

`gtk-tree-view-column-pack-start` (*self* `<gtk-tree-view-column>`) [Function]
(*cell* `<gtk-cell-renderer>`) (*expand* `bool`)
`pack-start` [Method]
Packs the *cell* into the beginning of the column. If *expand* is '`#f`', then the *cell* is allocated no more space than it needs. Any unused space is divided evenly between cells for which *expand* is '`#t`'.

tree-column
A `<gtk-tree-view-column>`.

cell The `<gtk-cell-renderer>`.

expand '`#t`' if *cell* is to be given extra space allocated to *tree-column*.

`gtk-tree-view-column-pack-end` (*self* `<gtk-tree-view-column>`) [Function]
(*cell* `<gtk-cell-renderer>`) (*expand* `bool`)
`pack-end` [Method]
Adds the *cell* to end of the column. If *expand* is '`#f`', then the *cell* is allocated no more space than it needs. Any unused space is divided evenly between cells for which *expand* is '`#t`'.

tree-column
A `<gtk-tree-view-column>`.

cell The `<gtk-cell-renderer>`.

expand '`#t`' if *cell* is to be given extra space allocated to *tree-column*.

`gtk-tree-view-column-clear` (*self* `<gtk-tree-view-column>`) [Function]
`clear` [Method]
Unsets all the mappings on all renderers on the *tree-column*.

tree-column
A `<gtk-tree-view-column>`

`gtk-tree-view-column-add-attribute` [Function]
(*self* `<gtk-tree-view-column>`) (*cell_renderer* `<gtk-cell-renderer>`)
(*attribute* `mchars`) (*column* `int`)
`add-attribute` [Method]
Adds an attribute mapping to the list in *tree-column*. The *column* is the column of the model to get a value from, and the *attribute* is the parameter on *cell-renderer* to be set from the value. So for example if column 2 of the model contains strings, you could have the "text" attribute of a `<gtk-cell-renderer-text>` get its values from column 2.

tree-column
A `<gtk-tree-view-column>`.

cell-renderer
: the `<gtk-cell-renderer>` to set attributes on

attribute
: An attribute on the renderer

column
: The column position on the model to get the attribute from.

`gtk-tree-view-column-set-spacing` [Function]
(*self* `<gtk-tree-view-column>`) (*spacing* `int`)

`set-spacing` [Method]

Sets the spacing field of *tree-column*, which is the number of pixels to place between cell renderers packed into it.

tree-column
: A `<gtk-tree-view-column>`.

spacing
: distance between cell renderers in pixels.

`gtk-tree-view-column-get-spacing` [Function]
(*self* `<gtk-tree-view-column>`) ⇒ (*ret* `int`)

`get-spacing` [Method]

Returns the spacing of *tree-column*.

tree-column
: A `<gtk-tree-view-column>`.

ret
: the spacing of *tree-column*.

`gtk-tree-view-column-set-visible` [Function]
(*self* `<gtk-tree-view-column>`) (*visible* `bool`)

`set-visible` [Method]

Sets the visibility of *tree-column*.

tree-column
: A `<gtk-tree-view-column>`.

visible
: '`#t`' if the *tree-column* is visible.

`gtk-tree-view-column-get-visible` [Function]
(*self* `<gtk-tree-view-column>`) ⇒ (*ret* `bool`)

`get-visible` [Method]

Returns '`#t`' if *tree-column* is visible.

tree-column
: A `<gtk-tree-view-column>`.

ret
: whether the column is visible or not. If it is visible, then the tree will show the column.

`gtk-tree-view-column-set-resizable` [Function]
(*self* `<gtk-tree-view-column>`) (*resizable* `bool`)

`set-resizable` [Method]

If *resizable* is '`#t`', then the user can explicitly resize the column by grabbing the outer edge of the column button. If resizable is '`#t`' and sizing mode of the column is `<gtk-tree-view-column-autosize>`, then the sizing mode is changed to `<gtk-tree-view-column-grow-only>`.

tree-column
A `<gtk-tree-view-column>`

resizable '`#t`', if the column can be resized

`gtk-tree-view-column-get-resizable` [Function]
(*self* `<gtk-tree-view-column>`) ⇒ (*ret* `bool`)
`get-resizable` [Method]
Returns '`#t`' if the *tree-column* can be resized by the end user.

tree-column
A `<gtk-tree-view-column>`

ret '`#t`', if the *tree-column* can be resized.

`gtk-tree-view-column-set-sizing` (*self* `<gtk-tree-view-column>`) [Function]
(*type* `<gtk-tree-view-column-sizing>`)
`set-sizing` [Method]
Sets the growth behavior of *tree-column* to *type*.

tree-column
A `<gtk-tree-view-column>`.

type The `<gtk-tree-view-column-sizing>`.

`gtk-tree-view-column-get-sizing` (*self* `<gtk-tree-view-column>`) [Function]
⇒ (*ret* `<gtk-tree-view-column-sizing>`)
`get-sizing` [Method]
Returns the current type of *tree-column*.

tree-column
A `<gtk-tree-view-column>`.

ret The type of *tree-column*.

`gtk-tree-view-column-get-width` (*self* `<gtk-tree-view-column>`) [Function]
⇒ (*ret* `int`)
`get-width` [Method]
Returns the current size of *tree-column* in pixels.

tree-column
A `<gtk-tree-view-column>`.

ret The current width of *tree-column*.

`gtk-tree-view-column-set-min-width` [Function]
(*self* `<gtk-tree-view-column>`) (*min_width* `int`)
`set-min-width` [Method]
Sets the minimum width of the *tree-column*. If *min-width* is -1, then the minimum width is unset.

tree-column
A `<gtk-tree-view-column>`.

min-width The minimum width of the column in pixels, or -1.

`gtk-tree-view-column-get-min-width` [Function]
(*self* `<gtk-tree-view-column>`) ⇒ (*ret* `int`)
`get-min-width` [Method]

Returns the minimum width in pixels of the *tree-column*, or -1 if no minimum width is set.

tree-column
A `<gtk-tree-view-column>`.

ret The minimum width of the *tree-column*.

`gtk-tree-view-column-set-max-width` [Function]
(*self* `<gtk-tree-view-column>`) (*max_width* `int`)
`set-max-width` [Method]

Sets the maximum width of the *tree-column*. If *max-width* is -1, then the maximum width is unset. Note, the column can actually be wider than max width if it's the last column in a view. In this case, the column expands to fill any extra space.

tree-column
A `<gtk-tree-view-column>`.

max-width
The maximum width of the column in pixels, or -1.

`gtk-tree-view-column-get-max-width` [Function]
(*self* `<gtk-tree-view-column>`) ⇒ (*ret* `int`)
`get-max-width` [Method]

Returns the maximum width in pixels of the *tree-column*, or -1 if no maximum width is set.

tree-column
A `<gtk-tree-view-column>`.

ret The maximum width of the *tree-column*.

`gtk-tree-view-column-clicked` (*self* `<gtk-tree-view-column>`) [Function]
`clicked` [Method]

Emits the "clicked" signal on the column. This function will only work if *tree-column* is clickable.

tree-column
a `<gtk-tree-view-column>`

`gtk-tree-view-column-set-title` (*self* `<gtk-tree-view-column>`) [Function]
(*title* `mchars`)
`set-title` [Method]

Sets the title of the *tree-column*. If a custom widget has been set, then this value is ignored.

tree-column
A `<gtk-tree-view-column>`.

title The title of the *tree-column*.

`gtk-tree-view-column-get-title` (*self* `<gtk-tree-view-column>`) [Function]
⇒ (*ret* `mchars`)

`get-title` [Method]

Returns the title of the widget.

tree-column
A `<gtk-tree-view-column>`.

ret the title of the column. This string should not be modified or freed.

`gtk-tree-view-column-set-expand` (*self* `<gtk-tree-view-column>`) [Function]
(*expand* `bool`)

`set-expand` [Method]

Sets the column to take available extra space. This space is shared equally amongst all columns that have the expand set to '`#t`'. If no column has this option set, then the last column gets all extra space. By default, every column is created with this '`#f`'.

tree-column
A `<gtk-tree-view-column>`

expand '`#t`' if the column should take available extra space, '`#f`' if not

Since 2.4

`gtk-tree-view-column-get-expand` (*self* `<gtk-tree-view-column>`) [Function]
⇒ (*ret* `bool`)

`get-expand` [Method]

Return '`#t`' if the column expands to take any available space.

tree-column
a `<gtk-tree-view-column>`

ret '`#t`', if the column expands

Since 2.4

`gtk-tree-view-column-set-clickable` [Function]
(*self* `<gtk-tree-view-column>`) (*clickable* `bool`)

`set-clickable` [Method]

Sets the header to be active if *active* is '`#t`'. When the header is active, then it can take keyboard focus, and can be clicked.

tree-column
A `<gtk-tree-view-column>`.

clickable '`#t`' if the header is active.

`gtk-tree-view-column-get-clickable` [Function]
(*self* `<gtk-tree-view-column>`) ⇒ (*ret* `bool`)

`get-clickable` [Method]

Returns '`#t`' if the user can click on the header for the column.

tree-column
a `<gtk-tree-view-column>`

ret '`#t`' if user can click the column header.

`gtk-tree-view-column-set-widget` (*self* `<gtk-tree-view-column>`) [Function]
(*widget* `<gtk-widget>`)
`set-widget` [Method]

Sets the widget in the header to be *widget*. If widget is '`#f`', then the header button is set with a `<gtk-label>` set to the title of *tree-column*.

tree-column
A `<gtk-tree-view-column>`.

widget A child `<gtk-widget>`, or '`#f`'.

`gtk-tree-view-column-get-widget` (*self* `<gtk-tree-view-column>`) [Function]
⇒ (*ret* `<gtk-widget>`)
`get-widget` [Method]

Returns the `<gtk-widget>` in the button on the column header. If a custom widget has not been set then '`#f`' is returned.

tree-column
A `<gtk-tree-view-column>`.

ret The `<gtk-widget>` in the column header, or '`#f`'

`gtk-tree-view-column-set-alignment` [Function]
(*self* `<gtk-tree-view-column>`) (*xalign* `float`)
`set-alignment` [Method]

Sets the alignment of the title or custom widget inside the column header. The alignment determines its location inside the button – 0.0 for left, 0.5 for center, 1.0 for right.

tree-column
A `<gtk-tree-view-column>`.

xalign The alignment, which is between [0.0 and 1.0] inclusive.

`gtk-tree-view-column-get-alignment` [Function]
(*self* `<gtk-tree-view-column>`) ⇒ (*ret* `float`)
`get-alignment` [Method]

Returns the current x alignment of *tree-column*. This value can range between 0.0 and 1.0.

tree-column
A `<gtk-tree-view-column>`.

ret The current alignent of *tree-column*.

`gtk-tree-view-column-set-sort-order` [Function]
(*self* `<gtk-tree-view-column>`) (*order* `<gtk-sort-type>`)
`set-sort-order` [Method]

Changes the appearance of the sort indicator.

This *does not* actually sort the model. Use `gtk-tree-view-column-set-sort-column-id` if you want automatic sorting support. This function is primarily for custom sorting behavior, and should be used in conjunction with `gtk-tree-sortable-set-sort-column` to do that. For custom models, the mechanism will vary.

The sort indicator changes direction to indicate normal sort or reverse sort. Note that you must have the sort indicator enabled to see anything when calling this function; see `gtk-tree-view-column-set-sort-indicator`.

tree-column
: a `<gtk-tree-view-column>`

order
: sort order that the sort indicator should indicate

`gtk-tree-view-column-get-sort-order` [Function]
(*self* `<gtk-tree-view-column>`) ⇒ (*ret* `<gtk-sort-type>`)

`get-sort-order` [Method]

Gets the value set by `gtk-tree-view-column-set-sort-order`.

tree-column
: a `<gtk-tree-view-column>`

ret
: the sort order the sort indicator is indicating

`gtk-tree-view-column-focus-cell` (*self* `<gtk-tree-view-column>`) [Function]
(*cell* `<gtk-cell-renderer>`)

`focus-cell` [Method]

Sets the current keyboard focus to be at *cell*, if the column contains 2 or more editable and activatable cells.

tree-column
: A `<gtk-tree-view-column>`

cell
: A `<gtk-cell-renderer>`

Since 2.2

`gtk-tree-view-column-queue-resize` [Function]
(*self* `<gtk-tree-view-column>`)

`queue-resize` [Method]

Flags the column, and the cell renderers added to this column, to have their sizes renegotiated.

tree-column
: A `<gtk-tree-view-column>`

Since 2.8

35 GtkTreeView

A widget for displaying both trees and lists

35.1 Overview

Widget that displays any object that implements the GtkTreeModel interface.

Please refer to the tree widget conceptual overview for an overview of all the objects and data types related to the tree widget and how they work together.

35.2 Usage

`<gtk-tree-view>` [Class]

This `<gobject>` class defines the following properties:

`model` The model for the tree view

`hadjustment`
Horizontal Adjustment for the widget

`vadjustment`
Vertical Adjustment for the widget

`headers-visible`
Show the column header buttons

`headers-clickable`
Column headers respond to click events

`expander-column`
Set the column for the expander column

`reorderable`
View is reorderable

`rules-hint`
Set a hint to the theme engine to draw rows in alternating colors

`enable-search`
View allows user to search through columns interactively

`search-column`
Model column to search through when searching through code

`fixed-height-mode`
Speeds up GtkTreeView by assuming that all rows have the same height

`hover-selection`
Whether the selection should follow the pointer

`hover-expand`
Whether rows should be expanded/collapsed when the pointer moves over them

`show-expanders`
: View has expanders

`level-indentation`
: Extra indentation for each level

`rubber-banding`
: Whether to enable selection of multiple items by dragging the mouse pointer

`enable-grid-lines`
: Whether grid lines should be drawn in the tree view

`enable-tree-lines`
: Whether tree lines should be drawn in the tree view

`tooltip-column`
: The column in the model containing the tooltip texts for the rows

`move-cursor` (*arg0* `<gtk-movement-step>`) [Signal on `<gtk-tree-view>`]
(*arg1* `<gint>`) ⇒ `<gboolean>`

`set-scroll-adjustments` [Signal on `<gtk-tree-view>`]
(*arg0* `<gtk-adjustment>`) (*arg1* `<gtk-adjustment>`)

`select-all` ⇒ `<gboolean>` [Signal on `<gtk-tree-view>`]

`unselect-all` ⇒ `<gboolean>` [Signal on `<gtk-tree-view>`]

`row-activated` (*arg0* `<gtk-tree-path>`) [Signal on `<gtk-tree-view>`]
(*arg1* `<gtk-tree-view-column>`)

The "row-activated" signal is emitted when the method `gtk-tree-view-row-activated` is called or the user double clicks a treeview row. It is also emitted when a non-editable row is selected and one of the keys: Space, Shift+Space, Return or Enter is pressed.

For selection handling refer to the tree widget conceptual overview as well as `<gtk-tree-selection>`.

`test-expand-row` (*arg0* `<gtk-tree-iter>`) [Signal on `<gtk-tree-view>`]
(*arg1* `<gtk-tree-path>`) ⇒ `<gboolean>`

The given row is about to be expanded (show its children nodes). Use this signal if you need to control the expandability of individual rows.

`test-collapse-row` (*arg0* `<gtk-tree-iter>`) [Signal on `<gtk-tree-view>`]
(*arg1* `<gtk-tree-path>`) ⇒ `<gboolean>`

The given row is about to be collapsed (hide its children nodes). Use this signal if you need to control the collapsibility of individual rows.

`row-expanded` (*arg0* `<gtk-tree-iter>`) [Signal on `<gtk-tree-view>`]
(*arg1* `<gtk-tree-path>`)

The given row has been expanded (child nodes are shown).

`row-collapsed` (*arg0* `<gtk-tree-iter>`) [Signal on `<gtk-tree-view>`]
(*arg1* `<gtk-tree-path>`)

The given row has been collapsed (child nodes are hidden).

`columns-changed` [Signal on `<gtk-tree-view>`]
The number of columns of the treeview has changed.

`cursor-changed` [Signal on `<gtk-tree-view>`]
The position of the cursor (focused cell) has changed.

`select-cursor-row` (*arg0* `<gboolean>`) [Signal on `<gtk-tree-view>`]
⇒ `<gboolean>`

`toggle-cursor-row` ⇒ `<gboolean>` [Signal on `<gtk-tree-view>`]

`expand-collapse-cursor-row` (*arg0* `<gboolean>`) [Signal on `<gtk-tree-view>`]
(*arg1* `<gboolean>`) (*arg2* `<gboolean>`) ⇒ `<gboolean>`

`select-cursor-parent` ⇒ `<gboolean>` [Signal on `<gtk-tree-view>`]

`start-interactive-search` ⇒ `<gboolean>` [Signal on `<gtk-tree-view>`]

`gtk-tree-view-new` ⇒ (*ret* `<gtk-widget>`) [Function]
Creates a new `<gtk-tree-view>` widget.

ret A newly created `<gtk-tree-view>` widget.

`gtk-tree-view-new-with-model` (*model* `<gtk-tree-model>`) [Function]
⇒ (*ret* `<gtk-widget>`)
Creates a new `<gtk-tree-view>` widget with the model initialized to *model*.

model the model.

ret A newly created `<gtk-tree-view>` widget.

`gtk-tree-view-get-model` (*self* `<gtk-tree-view>`) [Function]
⇒ (*ret* `<gtk-tree-model>`)
`get-model` [Method]
Returns the model the `<gtk-tree-view>` is based on. Returns '`#f`' if the model is unset.

tree-view a `<gtk-tree-view>`

ret A `<gtk-tree-model>`, or '`#f`' if none is currently being used.

`gtk-tree-view-set-model` (*self* `<gtk-tree-view>`) [Function]
(*model* `<gtk-tree-model>`)
`set-model` [Method]
Sets the model for a `<gtk-tree-view>`. If the *tree-view* already has a model set, it will remove it before setting the new model. If *model* is '`#f`', then it will unset the old model.

tree-view A `<gtk-tree-node>`.

model The model.

`gtk-tree-view-get-selection` (*self* `<gtk-tree-view>`) [Function]
⇒ (*ret* `<gtk-tree-selection>`)
`get-selection` [Method]
Gets the `<gtk-tree-selection>` associated with *tree-view*.

tree-view A `<gtk-tree-view>`.

ret A `<gtk-tree-selection>` object.

`gtk-tree-view-get-hadjustment` (*self* `<gtk-tree-view>`) [Function]
⇒ (*ret* `<gtk-adjustment>`)
`get-hadjustment` [Method]

Gets the `<gtk-adjustment>` currently being used for the horizontal aspect.

tree-view A `<gtk-tree-view>`

ret A `<gtk-adjustment>` object, or '`#f`' if none is currently being used.

`gtk-tree-view-set-hadjustment` (*self* `<gtk-tree-view>`) [Function]
(*adjustment* `<gtk-adjustment>`)
`set-hadjustment` [Method]

Sets the `<gtk-adjustment>` for the current horizontal aspect.

tree-view A `<gtk-tree-view>`

adjustment
The `<gtk-adjustment>` to set, or '`#f`'

`gtk-tree-view-get-vadjustment` (*self* `<gtk-tree-view>`) [Function]
⇒ (*ret* `<gtk-adjustment>`)
`get-vadjustment` [Method]

Gets the `<gtk-adjustment>` currently being used for the vertical aspect.

tree-view A `<gtk-tree-view>`

ret A `<gtk-adjustment>` object, or '`#f`' if none is currently being used.

`gtk-tree-view-set-vadjustment` (*self* `<gtk-tree-view>`) [Function]
(*adjustment* `<gtk-adjustment>`)
`set-vadjustment` [Method]

Sets the `<gtk-adjustment>` for the current vertical aspect.

tree-view A `<gtk-tree-view>`

adjustment
The `<gtk-adjustment>` to set, or '`#f`'

`gtk-tree-view-get-headers-visible` (*self* `<gtk-tree-view>`) [Function]
⇒ (*ret* `bool`)
`get-headers-visible` [Method]

Returns '`#t`' if the headers on the *tree-view* are visible.

tree-view A `<gtk-tree-view>`.

ret Whether the headers are visible or not.

`gtk-tree-view-set-headers-visible` (*self* `<gtk-tree-view>`) [Function]
(*headers_visible* `bool`)
`set-headers-visible` [Method]

Sets the visibility state of the headers.

tree-view A `<gtk-tree-view>`.

headers-visible
'`#t`' if the headers are visible

`gtk-tree-view-columns-autosize` (*self* `<gtk-tree-view>`) [Function]
`columns-autosize` [Method]
Resizes all columns to their optimal width. Only works after the treeview has been realized.

tree-view A `<gtk-tree-view>`.

`gtk-tree-view-get-headers-clickable` (*self* `<gtk-tree-view>`) [Function]
⇒ (*ret* `bool`)
`get-headers-clickable` [Method]
Returns whether all header columns are clickable.

tree-view A `<gtk-tree-view>`.

ret '`#t`' if all header columns are clickable, otherwise '`#f`'

Since 2.10

`gtk-tree-view-set-headers-clickable` (*self* `<gtk-tree-view>`) [Function]
(*setting* `bool`)
`set-headers-clickable` [Method]
Allow the column title buttons to be clicked.

tree-view A `<gtk-tree-view>`.

setting '`#t`' if the columns are clickable.

`gtk-tree-view-set-rules-hint` (*self* `<gtk-tree-view>`) [Function]
(*setting* `bool`)
`set-rules-hint` [Method]
This function tells GTK+ that the user interface for your application requires users to read across tree rows and associate cells with one another. By default, GTK+ will then render the tree with alternating row colors. Do *not* use it just because you prefer the appearance of the ruled tree; that's a question for the theme. Some themes will draw tree rows in alternating colors even when rules are turned off, and users who prefer that appearance all the time can choose those themes. You should call this function only as a *semantic* hint to the theme engine that your tree makes alternating colors useful from a functional standpoint (since it has lots of columns, generally).

tree-view a `<gtk-tree-view>`

setting '`#t`' if the tree requires reading across rows

`gtk-tree-view-get-rules-hint` (*self* `<gtk-tree-view>`) [Function]
⇒ (*ret* `bool`)
`get-rules-hint` [Method]
Gets the setting set by `gtk-tree-view-set-rules-hint`.

tree-view a `<gtk-tree-view>`

ret '`#t`' if rules are useful for the user of this tree

`gtk-tree-view-append-column` (*self* `<gtk-tree-view>`) [Function]
(*column* `<gtk-tree-view-column>`) ⇒ (*ret* `int`)

`append-column` [Method]

Appends *column* to the list of columns. If *tree-view* has "fixed_height" mode enabled, then *column* must have its "sizing" property set to be GTK_TREE_VIEW_COLUMN_FIXED.

tree-view A `<gtk-tree-view>`.

column The `<gtk-tree-view-column>` to add.

ret The number of columns in *tree-view* after appending.

`gtk-tree-view-remove-column` (*self* `<gtk-tree-view>`) [Function]
(*column* `<gtk-tree-view-column>`) ⇒ (*ret* `int`)

`remove-column` [Method]

Removes *column* from *tree-view*.

tree-view A `<gtk-tree-view>`.

column The `<gtk-tree-view-column>` to remove.

ret The number of columns in *tree-view* after removing.

`gtk-tree-view-insert-column` (*self* `<gtk-tree-view>`) [Function]
(*column* `<gtk-tree-view-column>`) (*position* `int`) ⇒ (*ret* `int`)

`insert-column` [Method]

This inserts the *column* into the *tree-view* at *position*. If *position* is -1, then the column is inserted at the end. If *tree-view* has "fixed_height" mode enabled, then *column* must have its "sizing" property set to be GTK_TREE_VIEW_COLUMN_FIXED.

tree-view A `<gtk-tree-view>`.

column The `<gtk-tree-view-column>` to be inserted.

position The position to insert *column* in.

ret The number of columns in *tree-view* after insertion.

`gtk-tree-view-get-column` (*self* `<gtk-tree-view>`) (*n* `int`) [Function]
⇒ (*ret* `<gtk-tree-view-column>`)

`get-column` [Method]

Gets the `<gtk-tree-view-column>` at the given position in the `<tree-view>`.

tree-view A `<gtk-tree-view>`.

n The position of the column, counting from 0.

ret The `<gtk-tree-view-column>`, or '`#f`' if the position is outside the range of columns.

`gtk-tree-view-get-columns` (*self* `<gtk-tree-view>`) [Function]
⇒ (*ret* `glist-of`)

`get-columns` [Method]

Returns a `<g-list>` of all the `<gtk-tree-view-column>` s currently in *tree-view*. The returned list must be freed with `g-list-free`.

tree-view A `<gtk-tree-view>`

ret A list of `<gtk-tree-view-column>` s

`gtk-tree-view-move-column-after` (*self* `<gtk-tree-view>`) [Function]
(*column* `<gtk-tree-view-column>`)
(*base_column* `<gtk-tree-view-column>`)
`move-column-after` [Method]

Moves *column* to be after to *base-column*. If *base-column* is '`#f`', then *column* is placed in the first position.

tree-view A `<gtk-tree-view>`

column The `<gtk-tree-view-column>` to be moved.

base-column
The `<gtk-tree-view-column>` to be moved relative to, or '`#f`'.

`gtk-tree-view-set-expander-column` (*self* `<gtk-tree-view>`) [Function]
(*column* `<gtk-tree-view-column>`)
`set-expander-column` [Method]

Sets the column to draw the expander arrow at. It must be in *tree-view*. If *column* is '`#f`', then the expander arrow is always at the first visible column.

If you do not want expander arrow to appear in your tree, set the expander column to a hidden column.

tree-view A `<gtk-tree-view>`

column '`#f`', or the column to draw the expander arrow at.

`gtk-tree-view-get-expander-column` (*self* `<gtk-tree-view>`) [Function]
⇒ (*ret* `<gtk-tree-view-column>`)
`get-expander-column` [Method]

Returns the column that is the current expander column. This column has the expander arrow drawn next to it.

tree-view A `<gtk-tree-view>`

ret The expander column.

`gtk-tree-view-scroll-to-point` (*self* `<gtk-tree-view>`) [Function]
(*tree_x* `int`) (*tree_y* `int`)
`scroll-to-point` [Method]

Scrolls the tree view such that the top-left corner of the visible area is *tree-x*, *tree-y*, where *tree-x* and *tree-y* are specified in tree window coordinates. The *tree-view* must be realized before this function is called. If it isn't, you probably want to be using `gtk-tree-view-scroll-to-cell`.

If either *tree-x* or *tree-y* are -1, then that direction isn't scrolled.

tree-view a `<gtk-tree-view>`

tree-x X coordinate of new top-left pixel of visible area, or -1

tree-y Y coordinate of new top-left pixel of visible area, or -1

`gtk-tree-view-scroll-to-cell` (*self* `<gtk-tree-view>`) [Function]
(*path* `<gtk-tree-path>`) (*column* `<gtk-tree-view-column>`)
(*use_align* `bool`) (*row_align* `float`) (*col_align* `float`)

`scroll-to-cell` [Method]

Moves the alignments of *tree-view* to the position specified by *column* and *path*. If *column* is '`#f`', then no horizontal scrolling occurs. Likewise, if *path* is '`#f`' no vertical scrolling occurs. At a minimum, one of *column* or *path* need to be non-'`#f`'. *row-align* determines where the row is placed, and *col-align* determines where *column* is placed. Both are expected to be between 0.0 and 1.0. 0.0 means left/top alignment, 1.0 means right/bottom alignment, 0.5 means center.

If *use-align* is '`#f`', then the alignment arguments are ignored, and the tree does the minimum amount of work to scroll the cell onto the screen. This means that the cell will be scrolled to the edge closest to its current position. If the cell is currently visible on the screen, nothing is done.

This function only works if the model is set, and *path* is a valid row on the model. If the model changes before the *tree-view* is realized, the centered path will be modified to reflect this change.

tree-view A `<gtk-tree-view>`.

path The path of the row to move to, or '`#f`'.

column The `<gtk-tree-view-column>` to move horizontally to, or '`#f`'.

use-align whether to use alignment arguments, or '`#f`'.

row-align The vertical alignment of the row specified by *path*.

col-align The horizontal alignment of the column specified by *column*.

`gtk-tree-view-set-cursor` (*self* `<gtk-tree-view>`) [Function]
(*path* `<gtk-tree-path>`) (*focus_column* `<gtk-tree-view-column>`)
(*start_editing* `bool`)

`set-cursor` [Method]

Sets the current keyboard focus to be at *path*, and selects it. This is useful when you want to focus the user's attention on a particular row. If *focus-column* is not '`#f`', then focus is given to the column specified by it. Additionally, if *focus-column* is specified, and *start-editing* is '`#t`', then editing should be started in the specified cell. This function is often followed by *gtk-widget-grab-focus* (*tree-view*) in order to give keyboard focus to the widget. Please note that editing can only happen when the widget is realized.

tree-view A `<gtk-tree-view>`

path A `<gtk-tree-path>`

focus-column
A `<gtk-tree-view-column>`, or '`#f`'

start-editing
'`#t`' if the specified cell should start being edited.

`gtk-tree-view-set-cursor-on-cell` (*self* `<gtk-tree-view>`) [Function]
(*path* `<gtk-tree-path>`) (*focus_column* `<gtk-tree-view-column>`)
(*focus_cell* `<gtk-cell-renderer>`) (*start_editing* `bool`)

`set-cursor-on-cell` [Method]

Sets the current keyboard focus to be at *path*, and selects it. This is useful when you want to focus the user's attention on a particular row. If *focus-column* is not '`#f`', then focus is given to the column specified by it. If *focus-column* and *focus-cell* are not '`#f`', and *focus-column* contains 2 or more editable or activatable cells, then focus is given to the cell specified by *focus-cell*. Additionally, if *focus-column* is specified, and *start-editing* is '`#t`', then editing should be started in the specified cell. This function is often followed by *gtk-widget-grab-focus* (*tree-view*) in order to give keyboard focus to the widget. Please note that editing can only happen when the widget is realized.

tree-view A `<gtk-tree-view>`

path A `<gtk-tree-path>`

focus-column
A `<gtk-tree-view-column>`, or '`#f`'

focus-cell A `<gtk-cell-renderer>`, or '`#f`'

start-editing
'`#t`' if the specified cell should start being edited.

Since 2.2

`gtk-tree-view-get-cursor` (*self* `<gtk-tree-view>`) [Function]
(*path* `<gtk-tree-path**>`) (*focus_column* `<gtk-tree-view-column**>`)

`get-cursor` [Method]

Fills in *path* and *focus-column* with the current path and focus column. If the cursor isn't currently set, then **path* will be '`#f`'. If no column currently has focus, then **focus-column* will be '`#f`'.

The returned `<gtk-tree-path>` must be freed with `gtk-tree-path-free` when you are done with it.

tree-view A `<gtk-tree-view>`

path A pointer to be filled with the current cursor path, or '`#f`'

focus-column
A pointer to be filled with the current focus column, or '`#f`'

`gtk-tree-view-row-activated` (*self* `<gtk-tree-view>`) [Function]
(*path* `<gtk-tree-path>`) (*column* `<gtk-tree-view-column>`)

`row-activated` [Method]

Activates the cell determined by *path* and *column*.

tree-view A `<gtk-tree-view>`

path The `<gtk-tree-path>` to be activated.

column The `<gtk-tree-view-column>` to be activated.

`gtk-tree-view-expand-all` (*self* `<gtk-tree-view>`) [Function]
`expand-all` [Method]

Recursively expands all nodes in the *tree-view*.

tree-view A `<gtk-tree-view>`.

`gtk-tree-view-collapse-all` (*self* `<gtk-tree-view>`) [Function]
`collapse-all` [Method]

Recursively collapses all visible, expanded nodes in *tree-view*.

tree-view A `<gtk-tree-view>`.

`gtk-tree-view-expand-to-path` (*self* `<gtk-tree-view>`) [Function]
(*path* `<gtk-tree-path>`)
`expand-to-path` [Method]

Expands the row at *path*. This will also expand all parent rows of *path* as necessary.

tree-view A `<gtk-tree-view>`.

path path to a row.

Since 2.2

`gtk-tree-view-expand-row` (*self* `<gtk-tree-view>`) [Function]
(*path* `<gtk-tree-path>`) (*open_all* `bool`) ⇒ (*ret* `bool`)
`expand-row` [Method]

Opens the row so its children are visible.

tree-view a `<gtk-tree-view>`

path path to a row

open-all whether to recursively expand, or just expand immediate children

ret '`#t`' if the row existed and had children

`gtk-tree-view-collapse-row` (*self* `<gtk-tree-view>`) [Function]
(*path* `<gtk-tree-path>`) ⇒ (*ret* `bool`)
`collapse-row` [Method]

Collapses a row (hides its child rows, if they exist).

tree-view a `<gtk-tree-view>`

path path to a row in the *tree-view*

ret '`#t`' if the row was collapsed.

`gtk-tree-view-map-expanded-rows` (*self* `<gtk-tree-view>`) [Function]
(*func* `<gtk-tree-view-mapping-func>`) (*data* `<gpointer>`)
`map-expanded-rows` [Method]

Calls *func* on all expanded rows.

tree-view A `<gtk-tree-view>`

func A function to be called

data User data to be passed to the function.

`gtk-tree-view-row-expanded` (*self* `<gtk-tree-view>`) [Function]
(*path* `<gtk-tree-path>`) ⇒ (*ret* `bool`)

`row-expanded` [Method]

Returns '`#t`' if the node pointed to by *path* is expanded in *tree-view*.

tree-view A `<gtk-tree-view>`.

path A `<gtk-tree-path>` to test expansion state.

ret '`#t`' if `<path>` is expanded.

`gtk-tree-view-set-reorderable` (*self* `<gtk-tree-view>`) [Function]
(*reorderable* `bool`)

`set-reorderable` [Method]

This function is a convenience function to allow you to reorder models that support the `<gtk-drag-source-iface>` and the `<gtk-drag-dest-iface>`. Both `<gtk-tree-store>` and `<gtk-list-store>` support these. If *reorderable* is '`#t`', then the user can reorder the model by dragging and dropping rows. The developer can listen to these changes by connecting to the model's row_inserted and row_deleted signals.

This function does not give you any degree of control over the order – any reordering is allowed. If more control is needed, you should probably handle drag and drop manually.

tree-view A `<gtk-tree-view>`.

reorderable
'`#t`', if the tree can be reordered.

`gtk-tree-view-get-reorderable` (*self* `<gtk-tree-view>`) [Function]
⇒ (*ret* `bool`)

`get-reorderable` [Method]

Retrieves whether the user can reorder the tree via drag-and-drop. See `gtk-tree-view-set-reorderable`.

tree-view a `<gtk-tree-view>`

ret '`#t`' if the tree can be reordered.

`gtk-tree-view-get-path-at-pos` (*self* `<gtk-tree-view>`) (*x* `int`) [Function]
(*y* `int`) (*path* `<gtk-tree-path**>`) (*column* `<gtk-tree-view-column**>`)
⇒ (*ret* `bool`) (*cell_x* `int`) (*cell_y* `int`)

`get-path-at-pos` [Method]

Finds the path at the point (*x*, *y*), relative to widget coordinates. That is, *x* and *y* are relative to an events coordinates. *x* and *y* must come from an event on the *tree-view* only where '`event->window == gtk_tree_view_get_bin ()`'. It is primarily for things like popup menus. If *path* is non-'`#f`', then it will be filled with the `<gtk-tree-path>` at that point. This path should be freed with `gtk-tree-path-free`. If *column* is non-'`#f`', then it will be filled with the column at that point. *cell-x* and *cell-y* return the coordinates relative to the cell background (i.e. the *background-area* passed to `gtk-cell-renderer-render`). This function is only meaningful if *tree-view* is realized.

tree-view A <gtk-tree-view>.

x The x position to be identified.

y The y position to be identified.

path A pointer to a <gtk-tree-path> pointer to be filled in, or '#f'

column A pointer to a <gtk-tree-view-column> pointer to be filled in, or '#f'

cell-x A pointer where the X coordinate relative to the cell can be placed, or '#f'

cell-y A pointer where the Y coordinate relative to the cell can be placed, or '#f'

ret '#t' if a row exists at that coordinate.

`gtk-tree-view-get-cell-area` (*self* <gtk-tree-view>) [Function]
(*path* <gtk-tree-path>) (*column* <gtk-tree-view-column>)
(*rect* <gdk-rectangle>)

`get-cell-area` [Method]

Fills the bounding rectangle in tree window coordinates for the cell at the row specified by *path* and the column specified by *column*. If *path* is '#f', or points to a path not currently displayed, the *y* and *height* fields of the rectangle will be filled with 0. If *column* is '#f', the *x* and *width* fields will be filled with 0. The sum of all cell rects does not cover the entire tree; there are extra pixels in between rows, for example. The returned rectangle is equivalent to the *cell-area* passed to `gtk-cell-renderer-render`. This function is only valid if *tree-view* is realized.

tree-view a <gtk-tree-view>

path a <gtk-tree-path> for the row, or '#f' to get only horizontal coordinates

column a <gtk-tree-view-column> for the column, or '#f' to get only vertical coordinates

rect rectangle to fill with cell rect

`gtk-tree-view-get-background-area` (*self* <gtk-tree-view>) [Function]
(*path* <gtk-tree-path>) (*column* <gtk-tree-view-column>)
(*rect* <gdk-rectangle>)

`get-background-area` [Method]

Fills the bounding rectangle in tree window coordinates for the cell at the row specified by *path* and the column specified by *column*. If *path* is '#f', or points to a node not found in the tree, the *y* and *height* fields of the rectangle will be filled with 0. If *column* is '#f', the *x* and *width* fields will be filled with 0. The returned rectangle is equivalent to the *background-area* passed to `gtk-cell-renderer-render`. These background areas tile to cover the entire tree window (except for the area used for header buttons). Contrast with the *cell-area*, returned by `gtk-tree-view-get-cell-area`, which returns only the cell itself, excluding surrounding borders and the tree expander area.

tree-view a <gtk-tree-view>

path a `<gtk-tree-path>` for the row, or '`#f`' to get only horizontal coordinates

column a `<gtk-tree-view-column>` for the column, or '`#f`' to get only vertical coordiantes

rect rectangle to fill with cell background rect

`gtk-tree-view-get-visible-rect` (*self* `<gtk-tree-view>`) [Function]
(*visible_rect* `<gdk-rectangle>`)

`get-visible-rect` [Method]

Fills *visible-rect* with the currently-visible region of the buffer, in tree coordinates. Convert to widget coordinates with `gtk-tree-view-tree-to-widget-coords`. Tree coordinates start at 0,0 for row 0 of the tree, and cover the entire scrollable area of the tree.

tree-view a `<gtk-tree-view>`

visible-rect rectangle to fill

`gtk-tree-view-get-visible-range` (*self* `<gtk-tree-view>`) [Function]
(*start_path* `<gtk-tree-path**>`) (*end_path* `<gtk-tree-path**>`)
⇒ (*ret* `bool`)

`get-visible-range` [Method]

Sets *start-path* and *end-path* to be the first and last visible path. Note that there may be invisible paths in between.

The paths should be freed with `gtk-tree-path-free` after use.

tree-view A `<gtk-tree-view>`

start-path Return location for start of region, or '`#f`'.

end-path Return location for end of region, or '`#f`'.

ret '`#t`', if valid paths were placed in *start-path* and *end-path*.

Since 2.8

`gtk-tree-view-get-bin-window` (*self* `<gtk-tree-view>`) [Function]
⇒ (*ret* `<gdk-window*>`)

`get-bin-window` [Method]

Returns the window that *tree-view* renders to. This is used primarily to compare to '`event->window`' to confirm that the event on *tree-view* is on the right window.

tree-view A `<gtk-tree-view>`

ret A `<gdk-window>`, or '`#f`' when *tree-view* hasn't been realized yet

`gtk-tree-view-widget-to-tree-coords` (*self* `<gtk-tree-view>`) [Function]
(*wx* `int`) (*wy* `int`) ⇒ (*tx* `int`) (*ty* `int`)

`widget-to-tree-coords` [Method]

Converts widget coordinates to coordinates for the tree window (the full scrollable area of the tree).

tree-view a `<gtk-tree-view>`

wx widget X coordinate

wy widget Y coordinate

tx return location for tree X coordinate

ty return location for tree Y coordinate

`gtk-tree-view-tree-to-widget-coords` (*self* `<gtk-tree-view>`) [Function]
(*tx* `int`) (*ty* `int`) ⇒ (*wx* `int`) (*wy* `int`)
`tree-to-widget-coords` [Method]

Converts tree coordinates (coordinates in full scrollable area of the tree) to widget coordinates.

tree-view a `<gtk-tree-view>`

tx tree X coordinate

ty tree Y coordinate

wx return location for widget X coordinate

wy return location for widget Y coordinate

`gtk-tree-view-unset-rows-drag-dest` (*self* `<gtk-tree-view>`) [Function]
`unset-rows-drag-dest` [Method]

Undoes the effect of `gtk-tree-view-enable-model-drag-dest`.

tree-view a `<gtk-tree-view>`

`gtk-tree-view-set-drag-dest-row` (*self* `<gtk-tree-view>`) [Function]
(*path* `<gtk-tree-path>`) (*pos* `<gtk-tree-view-drop-position>`)
`set-drag-dest-row` [Method]

Sets the row that is highlighted for feedback.

tree-view a `<gtk-tree-view>`

path The path of the row to highlight, or '`#f`'.

pos Specifies whether to drop before, after or into the row

`gtk-tree-view-get-drag-dest-row` (*self* `<gtk-tree-view>`) [Function]
(*path* `<gtk-tree-path**>`) (*pos* `<gtk-tree-view-drop-position*>`)
`get-drag-dest-row` [Method]

Gets information about the row that is highlighted for feedback.

tree-view a `<gtk-tree-view>`

path Return location for the path of the highlighted row, or '`#f`'.

pos Return location for the drop position, or '`#f`'

`gtk-tree-view-get-dest-row-at-pos` (*self* `<gtk-tree-view>`) [Function]
(*drag_x* `int`) (*drag_y* `int`) (*path* `<gtk-tree-path**>`)
(*pos* `<gtk-tree-view-drop-position*>`) ⇒ (*ret* `bool`)
`get-dest-row-at-pos` [Method]

Determines the destination row for a given position.

tree-view a `<gtk-tree-view>`

drag-x the position to determine the destination row for

drag-y the position to determine the destination row for

path Return location for the path of the highlighted row, or '`#f`'.

pos Return location for the drop position, or '`#f`'

ret whether there is a row at the given position.

`gtk-tree-view-create-row-drag-icon` (*self* `<gtk-tree-view>`) [Function]
(*path* `<gtk-tree-path>`) ⇒ (*ret* `<gdk-pixmap*>`)

`create-row-drag-icon` [Method]

Creates a `<gdk-pixmap>` representation of the row at *path*. This image is used for a drag icon.

tree-view a `<gtk-tree-view>`

path a `<gtk-tree-path>` in *tree-view*

ret a newly-allocated pixmap of the drag icon.

`gtk-tree-view-set-enable-search` (*self* `<gtk-tree-view>`) [Function]
(*enable_search* `bool`)

`set-enable-search` [Method]

If *enable-search* is set, then the user can type in text to search through the tree interactively (this is sometimes called "typeahead find").

Note that even if this is '`#f`', the user can still initiate a search using the "start-interactive-search" key binding.

tree-view A `<gtk-tree-view>`

enable-search
'`#t`', if the user can search interactively

`gtk-tree-view-get-enable-search` (*self* `<gtk-tree-view>`) [Function]
⇒ (*ret* `bool`)

`get-enable-search` [Method]

Returns whether or not the tree allows to start interactive searching by typing in text.

tree-view A `<gtk-tree-view>`

ret whether or not to let the user search interactively

`gtk-tree-view-get-search-column` (*self* `<gtk-tree-view>`) [Function]
⇒ (*ret* `int`)

`get-search-column` [Method]

Gets the column searched on by the interactive search code.

tree-view A `<gtk-tree-view>`

ret the column the interactive search code searches in.

`gtk-tree-view-set-search-column` (*self* `<gtk-tree-view>`) [Function]
(*column* `int`)

`set-search-column` [Method]

Sets *column* as the column where the interactive search code should search in.

If the sort column is set, users can use the "start-interactive-search" key binding to bring up search popup. The enable-search property controls whether simply typing text will also start an interactive search.

Note that *column* refers to a column of the model.

tree-view A `<gtk-tree-view>`

column the column of the model to search in, or -1 to disable searching

`gtk-tree-view-set-search-equal-func` (*self* `<gtk-tree-view>`) [Function]
(*search_equal_func* `<gtk-tree-view-search-equal-func>`)
(*search_user_data* `<gpointer>`) (*search_destroy* `<gtk-destroy-notify>`)

`set-search-equal-func` [Method]

Sets the compare function for the interactive search capabilities; note that somewhat like `strcmp` returning 0 for equality `<gtk-tree-view-search-equal-func>` returns '`#f`' on matches.

tree-view A `<gtk-tree-view>`

search-equal-func
the compare function to use during the search

search-user-data
user data to pass to *search-equal-func*, or '`#f`'

search-destroy
Destroy notifier for *search-user-data*, or '`#f`'

`gtk-tree-view-get-search-entry` (*self* `<gtk-tree-view>`) [Function]
⇒ (*ret* `<gtk-entry>`)

`get-search-entry` [Method]

Returns the GtkEntry which is currently in use as interactive search entry for *tree-view*. In case the built-in entry is being used, '`#f`' will be returned.

tree-view A `<gtk-tree-view>`

ret the entry currently in use as search entry.

Since 2.10

`gtk-tree-view-set-search-entry` (*self* `<gtk-tree-view>`) [Function]
(*entry* `<gtk-entry>`)

`set-search-entry` [Method]

Sets the entry which the interactive search code will use for this *tree-view*. This is useful when you want to provide a search entry in our interface at all time at a fixed position. Passing '`#f`' for *entry* will make the interactive search code use the built-in popup entry again.

tree-view A `<gtk-tree-view>`

entry the entry the interactive search code of *tree-view* should use or '`#f`'

Since 2.10

`gtk-tree-view-get-fixed-height-mode` (*self* `<gtk-tree-view>`) [Function]
⇒ (*ret* `bool`)
`get-fixed-height-mode` [Method]

Returns whether fixed height mode is turned on for *tree-view*.

tree-view a `<gtk-tree-view>`

ret '`#t`' if *tree-view* is in fixed height mode

Since 2.6

`gtk-tree-view-set-fixed-height-mode` (*self* `<gtk-tree-view>`) [Function]
(*enable* `bool`)
`set-fixed-height-mode` [Method]

Enables or disables the fixed height mode of *tree-view*. Fixed height mode speeds up `<gtk-tree-view>` by assuming that all rows have the same height. Only enable this option if all rows are the same height and all columns are of type '`GTK_TREE_VIEW_COLUMN_FIXED`'.

tree-view a `<gtk-tree-view>`

enable '`#t`' to enable fixed height mode

Since 2.6

`gtk-tree-view-get-hover-selection` (*self* `<gtk-tree-view>`) [Function]
⇒ (*ret* `bool`)
`get-hover-selection` [Method]

Returns whether hover selection mode is turned on for *tree-view*.

tree-view a `<gtk-tree-view>`

ret '`#t`' if *tree-view* is in hover selection mode

Since 2.6

`gtk-tree-view-set-hover-selection` (*self* `<gtk-tree-view>`) [Function]
(*hover* `bool`)
`set-hover-selection` [Method]

Enables of disables the hover selection mode of *tree-view*. Hover selection makes the selected row follow the pointer. Currently, this works only for the selection modes '`GTK_SELECTION_SINGLE`' and '`GTK_SELECTION_BROWSE`'.

tree-view a `<gtk-tree-view>`

hover '`#t`' to enable hover selection mode

Since 2.6

`gtk-tree-view-get-hover-expand` (*self* `<gtk-tree-view>`) [Function]
⇒ (*ret* `bool`)
`get-hover-expand` [Method]

Returns whether hover expansion mode is turned on for *tree-view*.

tree-view a `<gtk-tree-view>`

ret '`#t`' if *tree-view* is in hover expansion mode

Since 2.6

`gtk-tree-view-set-hover-expand` (*self* `<gtk-tree-view>`) [Function]
(*expand* `bool`)
`set-hover-expand` [Method]

Enables of disables the hover expansion mode of *tree-view*. Hover expansion makes rows expand or collaps if the pointer moves over them.

tree-view a `<gtk-tree-view>`

expand '`#t`' to enable hover selection mode

Since 2.6

`gtk-tree-view-get-rubber-banding` (*self* `<gtk-tree-view>`) [Function]
⇒ (*ret* `bool`)
`get-rubber-banding` [Method]

Returns whether rubber banding is turned on for *tree-view*. If the selection mode is `<gtk-selection-multiple>`, rubber banding will allow the user to select multiple rows by dragging the mouse.

tree-view a `<gtk-tree-view>`

ret '`#t`' if rubber banding in *tree-view* is enabled.

Since 2.10

`gtk-tree-view-set-rubber-banding` (*self* `<gtk-tree-view>`) [Function]
(*enable* `bool`)
`set-rubber-banding` [Method]

Enables or disables rubber banding in *tree-view*. If the selection mode is `<gtk-selection-multiple>`, rubber banding will allow the user to select multiple rows by dragging the mouse.

tree-view a `<gtk-tree-view>`

enable '`#t`' to enable rubber banding

Since 2.10

`gtk-tree-view-get-enable-tree-lines` (*self* `<gtk-tree-view>`) [Function]
⇒ (*ret* `bool`)
`get-enable-tree-lines` [Method]

Returns whether or not tree lines are drawn in *tree-view*.

tree-view a `<gtk-tree-view>`.

ret '`#t`' if tree lines are drawn in *tree-view*, '`#f`' otherwise.

Since 2.10

`gtk-tree-view-set-enable-tree-lines` (*self* `<gtk-tree-view>`) [Function]
(*enabled* `bool`)
`set-enable-tree-lines` [Method]

Sets whether to draw lines interconnecting the expanders in *tree-view*. This does not have any visible effects for lists.

tree-view a `<gtk-tree-view>`

enabled '`#t`' to enable tree line drawing, '`#f`' otherwise.

Since 2.10

`gtk-tree-view-get-grid-lines` (*self* `<gtk-tree-view>`) [Function]
⇒ (*ret* `<gtk-tree-view-grid-lines>`)
`get-grid-lines` [Method]

Returns which grid lines are enabled in *tree-view*.

tree-view a `<gtk-tree-view>`

ret a `<gtk-tree-view-grid-lines>` value indicating which grid lines are enabled.

Since 2.10

`gtk-tree-view-set-grid-lines` (*self* `<gtk-tree-view>`) [Function]
(*grid_lines* `<gtk-tree-view-grid-lines>`)
`set-grid-lines` [Method]

Sets which grid lines to draw in *tree-view*.

tree-view a `<gtk-tree-view>`

grid-lines a `<gtk-tree-view-grid-lines>` value indicating which grid lines to enable.

Since 2.10

36 GtkTreeView drag-and-drop

Interfaces for drag-and-drop support in GtkTreeView

36.1 Overview

GTK+ supports Drag-and-Drop in tree views with a high-level and a low-level API.

The low-level API consists of the GTK+ DND API, augmented by some treeview utility functions: `gtk-tree-view-set-drag-dest-row`, `gtk-tree-view-get-drag-dest-row`, `gtk-tree-view-get-dest-row-at-pos`, `gtk-tree-view-create-row-drag-icon`, `gtk-tree-set-row-drag-data` and `gtk-tree-get-row-drag-data`. This API leaves a lot of flexibility, but nothing is done automatically, and implementing advanced features like hover-to-open-rows or autoscrolling on top of this API is a lot of work.

On the other hand, if you write to the high-level API, then all the bookkeeping of rows is done for you, as well as things like hover-to-open and auto-scroll, but your models have to implement the `<gtk-tree-drag-source>` and `<gtk-tree-drag-dest>` interfaces.

36.2 Usage

`<gtk-tree-drag-source>` [Class]

This `<gobject>` class defines no properties, other than those defined by its superclasses.

`<gtk-tree-drag-dest>` [Class]

This `<gobject>` class defines no properties, other than those defined by its superclasses.

`gtk-tree-drag-source-drag-data-get` [Function]

(*self* `<gtk-tree-drag-source>`) (*path* `<gtk-tree-path>`) (*selection_data* `<gtk-selection-data>`) ⇒ (*ret* `bool`)

`drag-data-get` [Method]

Asks the `<gtk-tree-drag-source>` to fill in *selection-data* with a representation of the row at *path*. *selection-data->target* gives the required type of the data. Should robustly handle a *path* no longer found in the model!

drag-source
: a `<gtk-tree-drag-source>`

path
: row that was dragged

selection-data
: a `<gtk-selection-data>` to fill with data from the dragged row

ret
: '`#t`' if data of the required type was provided

`gtk-tree-drag-source-row-draggable` [Function]

(*self* `<gtk-tree-drag-source>`) (*path* `<gtk-tree-path>`) ⇒ (*ret* `bool`)

`row-draggable` [Method]

Asks the `<gtk-tree-drag-source>` whether a particular row can be used as the source of a DND operation. If the source doesn't implement this interface, the row is assumed draggable.

drag-source
: a `<gtk-tree-drag-source>`

path
: row on which user is initiating a drag

ret
: '`#t`' if the row can be dragged

`gtk-tree-set-row-drag-data` (*self* `<gtk-selection-data>`) [Function]
(*tree_model* `<gtk-tree-model>`) (*path* `<gtk-tree-path>`) ⇒ (*ret* `bool`)

Sets selection data of target type '`GTK_TREE_MODEL_ROW`'. Normally used in a drag_data_get handler.

selection-data
: some `<gtk-selection-data>`

tree-model
: a `<gtk-tree-model>`

path
: a row in *tree-model*

ret
: '`#t`' if the `<gtk-selection-data>` had the proper target type to allow us to set a tree row

`gtk-tree-get-row-drag-data` (*self* `<gtk-selection-data>`) [Function]
(*tree_model* `<gtk-tree-model**>`) (*path* `<gtk-tree-path**>`)
⇒ (*ret* `bool`)

Obtains a *tree-model* and *path* from selection data of target type '`GTK_TREE_MODEL_ROW`'. Normally called from a drag_data_received handler. This function can only be used if *selection-data* originates from the same process that's calling this function, because a pointer to the tree model is being passed around. If you aren't in the same process, then you'll get memory corruption. In the `<gtk-tree-drag-dest>` drag_data_received handler, you can assume that selection data of type '`GTK_TREE_MODEL_ROW`' is in from the current process. The returned path must be freed with `gtk-tree-path-free`.

selection-data
: a `<gtk-selection-data>`

tree-model
: a `<gtk-tree-model>`

path
: row in *tree-model*

ret
: '`#t`' if *selection-data* had target type '`GTK_TREE_MODEL_ROW`' and is otherwise valid

37 GtkCellView

A widget displaying a single row of a GtkTreeModel

37.1 Overview

A <gtk-cell-view> displays a single row of a <gtk-tree-model>, using cell renderers just like <gtk-tree-view>. <gtk-cell-view> doesn't support some of the more complex features of <gtk-tree-view>, like cell editing and drag and drop.

37.2 Usage

<gtk-cell-view> [Class]

This <gobject> class defines the following properties:

background
: Background color as a string

background-gdk
: Background color as a GdkColor

background-set
: Whether this tag affects the background color

model
: The model for cell view

gtk-cell-view-new ⇒ (*ret* <gtk-widget>) [Function]

Creates a new <gtk-cell-view> widget.

ret A newly created <gtk-cell-view> widget.

Since 2.6

gtk-cell-view-new-with-text (*text* mchars) ⇒ (*ret* <gtk-widget>) [Function]

Creates a new <gtk-cell-view> widget, adds a <gtk-cell-renderer-text> to it, and makes its show *text*.

text the text to display in the cell view

ret A newly created <gtk-cell-view> widget.

Since 2.6

gtk-cell-view-new-with-markup (*markup* mchars) [Function]
⇒ (*ret* <gtk-widget>)

Creates a new <gtk-cell-view> widget, adds a <gtk-cell-renderer-text> to it, and makes its show *markup*. The text can text can be marked up with the Pango text markup language.

markup the text to display in the cell view

ret A newly created <gtk-cell-view> widget.

Since 2.6

`gtk-cell-view-new-with-pixbuf` (*pixbuf* `<gdk-pixbuf>`) [Function]
⇒ (*ret* `<gtk-widget>`)

Creates a new `<gtk-cell-view>` widget, adds a `<gtk-cell-renderer-pixbuf>` to it, and makes its show *pixbuf*.

pixbuf the image to display in the cell view

ret A newly created `<gtk-cell-view>` widget.

Since 2.6

`gtk-cell-view-set-model` (*self* `<gtk-cell-view>`) [Function]
(*model* `<gtk-tree-model>`)

`set-model` [Method]

Sets the model for *cell-view*. If *cell-view* already has a model set, it will remove it before setting the new model. If *model* is '`#f`', then it will unset the old model.

cell-view a `<gtk-cell-view>`

model a `<gtk-tree-model>`

Since 2.6

`gtk-cell-view-set-displayed-row` (*self* `<gtk-cell-view>`) [Function]
(*path* `<gtk-tree-path>`)

`set-displayed-row` [Method]

Sets the row of the model that is currently displayed by the `<gtk-cell-view>`. If the path is unset, then the contents of the cellview "stick" at their last value; this is not normally a desired result, but may be a needed intermediate state if say, the model for the `<gtk-cell-view>` becomes temporarily empty.

cell-view a `<gtk-cell-view>`

path a `<gtk-tree-path>` or '`#f`' to unset.

Since 2.6

`gtk-cell-view-get-displayed-row` (*self* `<gtk-cell-view>`) [Function]
⇒ (*ret* `<gtk-tree-path>`)

`get-displayed-row` [Method]

Returns a `<gtk-tree-path>` referring to the currently displayed row. If no row is currently displayed, '`#f`' is returned.

cell-view a `<gtk-cell-view>`

ret the currently displayed row or '`#f`'

Since 2.6

`gtk-cell-view-get-size-of-row` (*self* `<gtk-cell-view>`) [Function]
(*path* `<gtk-tree-path>`) (*requisition* `<gtk-requisition>`) ⇒ (*ret* `bool`)

`get-size-of-row` [Method]

Sets *requisition* to the size needed by *cell-view* to display the model row pointed to by *path*.

cell-view a `<gtk-cell-view>`

path a `<gtk-tree-path>`

requisition return location for the size

ret '`#t`'

Since 2.6

`gtk-cell-view-set-background-color` (*self* `<gtk-cell-view>`) (*color* `<gdk-color>`) [Function]

`set-background-color` [Method]

Sets the background color of *view*.

cell-view a `<gtk-cell-view>`

color the new background color

Since 2.6

`gtk-cell-view-get-cell-renderers` (*self* `<gtk-cell-view>`) ⇒ (*ret* `glist-of`) [Function]

`get-cell-renderers` [Method]

Returns the cell renderers which have been added to *cell-view*.

cell-view a `<gtk-cell-view>`

ret a list of cell renderers. The list, but not the renderers has been newly allocated and should be freed with `g-list-free` when no longer needed.

Since 2.6

38 GtkIconView

A widget which displays a list of icons in a grid

38.1 Overview

<gtk-icon-view> provides an alternative view on a list model. It displays the model as a grid of icons with labels. Like <gtk-tree-view>, it allows to select one or multiple items (depending on the selection mode, see gtk-icon-view-set-selection-mode). In addition to selection with the arrow keys, <gtk-icon-view> supports rubberband selection, which is controlled by dragging the pointer.

38.2 Usage

<gtk-icon-view> [Class]

This <gobject> class defines the following properties:

pixbuf-column
: Model column used to retrieve the icon pixbuf from

text-column
: Model column used to retrieve the text from

markup-column
: Model column used to retrieve the text if using Pango markup

selection-mode
: The selection mode

orientation
: How the text and icon of each item are positioned relative to each other

model
: The model for the icon view

columns
: Number of columns to display

item-width
: The width used for each item

spacing
: Space which is inserted between cells of an item

row-spacing
: Space which is inserted between grid rows

column-spacing
: Space which is inserted between grid columns

margin
: Space which is inserted at the edges of the icon view

reorderable
: View is reorderable

tooltip-column
: The column in the model containing the tooltip texts for the items

`move-cursor` (*arg0* `<gtk-movement-step>`) [Signal on `<gtk-icon-view>`]
(*arg1* `<gint>`) ⇒ `<gboolean>`

`selection-changed` [Signal on `<gtk-icon-view>`]

`set-scroll-adjustments` [Signal on `<gtk-icon-view>`]
(*arg0* `<gtk-adjustment>`) (*arg1* `<gtk-adjustment>`)

`item-activated` (*arg0* `<gtk-tree-path>`) [Signal on `<gtk-icon-view>`]

`select-all` [Signal on `<gtk-icon-view>`]

`unselect-all` [Signal on `<gtk-icon-view>`]

`select-cursor-item` [Signal on `<gtk-icon-view>`]

`toggle-cursor-item` [Signal on `<gtk-icon-view>`]

`activate-cursor-item` ⇒ `<gboolean>` [Signal on `<gtk-icon-view>`]

`gtk-icon-view-new` ⇒ (*ret* `<gtk-widget>`) [Function]

Creates a new `<gtk-icon-view>` widget

ret A newly created `<gtk-icon-view>` widget

Since 2.6

`gtk-icon-view-new-with-model` (*model* `<gtk-tree-model>`) [Function]
⇒ (*ret* `<gtk-widget>`)

Creates a new `<gtk-icon-view>` widget with the model *model*.

model The model.

ret A newly created `<gtk-icon-view>` widget.

Since 2.6

`gtk-icon-view-set-model` (*self* `<gtk-icon-view>`) [Function]
(*model* `<gtk-tree-model>`)

`set-model` [Method]

Sets the model for a `<gtk-icon-view>`. If the *icon-view* already has a model set, it will remove it before setting the new model. If *model* is '`#f`', then it will unset the old model.

icon-view A `<gtk-icon-view>`.

model The model.

Since 2.6

`gtk-icon-view-get-model` (*self* `<gtk-icon-view>`) [Function]
⇒ (*ret* `<gtk-tree-model>`)

`get-model` [Method]

Returns the model the `<gtk-icon-view>` is based on. Returns '`#f`' if the model is unset.

icon-view a `<gtk-icon-view>`

ret A `<gtk-tree-model>`, or '`#f`' if none is currently being used.

Since 2.6

`gtk-icon-view-set-text-column` (*self* `<gtk-icon-view>`) [Function]
(*column* `int`)
`set-text-column` [Method]

Sets the column with text for *icon-view* to be *column*. The text column must be of type `<g-type-string>`.

icon-view A `<gtk-icon-view>`.

column A column in the currently used model.

Since 2.6

`gtk-icon-view-get-text-column` (*self* `<gtk-icon-view>`) [Function]
⇒ (*ret* `int`)
`get-text-column` [Method]

Returns the column with text for *icon-view*.

icon-view A `<gtk-icon-view>`.

ret the text column, or -1 if it's unset.

Since 2.6

`gtk-icon-view-set-markup-column` (*self* `<gtk-icon-view>`) [Function]
(*column* `int`)
`set-markup-column` [Method]

Sets the column with markup information for *icon-view* to be *column*. The markup column must be of type `<g-type-string>`. If the markup column is set to something, it overrides the text column set by `gtk-icon-view-set-text-column`.

icon-view A `<gtk-icon-view>`.

column A column in the currently used model.

Since 2.6

`gtk-icon-view-get-markup-column` (*self* `<gtk-icon-view>`) [Function]
⇒ (*ret* `int`)
`get-markup-column` [Method]

Returns the column with markup text for *icon-view*.

icon-view A `<gtk-icon-view>`.

ret the markup column, or -1 if it's unset.

Since 2.6

`gtk-icon-view-set-pixbuf-column` (*self* `<gtk-icon-view>`) [Function]
(*column* `int`)
`set-pixbuf-column` [Method]

Sets the column with pixbufs for *icon-view* to be *column*. The pixbuf column must be of type `<gdk-type-pixbuf>`

icon-view A `<gtk-icon-view>`.

column A column in the currently used model.

Since 2.6

`gtk-icon-view-get-pixbuf-column` (*self* `<gtk-icon-view>`) [Function]
⇒ (*ret* `int`)

`get-pixbuf-column` [Method]

Returns the column with pixbufs for *icon-view*.

icon-view A `<gtk-icon-view>`.

ret the pixbuf column, or -1 if it's unset.

Since 2.6

`gtk-icon-view-get-path-at-pos` (*self* `<gtk-icon-view>`) (*x* `int`) [Function]
(*y* `int`) ⇒ (*ret* `<gtk-tree-path>`)

`get-path-at-pos` [Method]

Finds the path at the point (*x*, *y*), relative to widget coordinates. See `gtk-icon-view-get-item-at-pos`, if you are also interested in the cell at the specified position.

icon-view A `<gtk-icon-view>`.

x The x position to be identified

y The y position to be identified

ret The `<gtk-tree-path>` corresponding to the icon or '`#f`' if no icon exists at that position.

Since 2.6

`gtk-icon-view-get-item-at-pos` (*self* `<gtk-icon-view>`) (*x* `int`) [Function]
(*y* `int`) (*path* `<gtk-tree-path**>`) (*cell* `<gtk-cell-renderer**>`)
⇒ (*ret* `bool`)

`get-item-at-pos` [Method]

Finds the path at the point (*x*, *y*), relative to widget coordinates. In contrast to `gtk-icon-view-get-path-at-pos`, this function also obtains the cell at the specified position. The returned path should be freed with `gtk-tree-path-free`.

icon-view A `<gtk-icon-view>`.

x The x position to be identified

y The y position to be identified

path Return location for the path, or '`#f`'

cell Return location for the renderer responsible for the cell at (*x*, *y*), or '`#f`'

ret '`#t`' if an item exists at the specified position

Since 2.8

`gtk-icon-view-set-cursor` (*self* `<gtk-icon-view>`) [Function]
(*path* `<gtk-tree-path>`) (*cell* `<gtk-cell-renderer>`) (*start_editing* `bool`)

`set-cursor` [Method]

Sets the current keyboard focus to be at *path*, and selects it. This is useful when you want to focus the user's attention on a particular item. If *cell* is not '`#f`', then focus is given to the cell specified by it. Additionally, if *start-editing* is '`#t`', then editing should be started in the specified cell.

This function is often followed by '`gtk_widget_grab_focus (icon_view)`' in order to give keyboard focus to the widget. Please note that editing can only happen when the widget is realized.

icon-view A `<gtk-icon-view>`

path A `<gtk-tree-path>`

cell One of the cell renderers of *icon-view*, or '`#f`'

start-editing
'`#t`' if the specified cell should start being edited.

Since 2.8

`gtk-icon-view-get-cursor` (*self* `<gtk-icon-view>`) [Function]
(*path* `<gtk-tree-path**>`) (*cell* `<gtk-cell-renderer**>`) ⇒ (*ret* `bool`)
`get-cursor` [Method]

Fills in *path* and *cell* with the current cursor path and cell. If the cursor isn't currently set, then **path* will be '`#f`'. If no cell currently has focus, then **cell* will be '`#f`'.

The returned `<gtk-tree-path>` must be freed with `gtk-tree-path-free`.

icon-view A `<gtk-icon-view>`

path Return location for the current cursor path, or '`#f`'

cell Return location the current focus cell, or '`#f`'

ret '`#t`' if the cursor is set.

Since 2.8

`gtk-icon-view-set-selection-mode` (*self* `<gtk-icon-view>`) [Function]
(*mode* `<gtk-selection-mode>`)
`set-selection-mode` [Method]

Sets the selection mode of the *icon-view*.

icon-view A `<gtk-icon-view>`.

mode The selection mode

Since 2.6

`gtk-icon-view-get-selection-mode` (*self* `<gtk-icon-view>`) [Function]
⇒ (*ret* `<gtk-selection-mode>`)
`get-selection-mode` [Method]

Gets the selection mode of the *icon-view*.

icon-view A `<gtk-icon-view>`.

ret the current selection mode

Since 2.6

`gtk-icon-view-set-orientation` (*self* `<gtk-icon-view>`) (*orientation* `<gtk-orientation>`) [Function]
`set-orientation` [Method]

Sets the ::orientation property which determines whether the labels are drawn beside the icons instead of below.

icon-view a `<gtk-icon-view>`

orientation the relative position of texts and icons

Since 2.6

`gtk-icon-view-get-orientation` (*self* `<gtk-icon-view>`) ⇒ (*ret* `<gtk-orientation>`) [Function]
`get-orientation` [Method]

Returns the value of the ::orientation property which determines whether the labels are drawn beside the icons instead of below.

icon-view a `<gtk-icon-view>`

ret the relative position of texts and icons

Since 2.6

`gtk-icon-view-set-columns` (*self* `<gtk-icon-view>`) (*columns* `int`) [Function]
`set-columns` [Method]

Sets the ::columns property which determines in how many columns the icons are arranged. If *columns* is -1, the number of columns will be chosen automatically to fill the available area.

icon-view a `<gtk-icon-view>`

columns the number of columns

Since 2.6

`gtk-icon-view-get-columns` (*self* `<gtk-icon-view>`) ⇒ (*ret* `int`) [Function]
`get-columns` [Method]

Returns the value of the ::columns property.

icon-view a `<gtk-icon-view>`

ret the number of columns, or -1

Since 2.6

`gtk-icon-view-set-item-width` (*self* `<gtk-icon-view>`) (*item_width* `int`) [Function]
`set-item-width` [Method]

Sets the ::item-width property which specifies the width to use for each item. If it is set to -1, the icon view will automatically determine a suitable item size.

icon-view a `<gtk-icon-view>`

item-width the width for each item

Since 2.6

`gtk-icon-view-get-item-width` (*self* `<gtk-icon-view>`) ⇒ (*ret* `int`) [Function]
`get-item-width` [Method]

Returns the value of the ::item-width property.

icon-view a `<gtk-icon-view>`

ret the width of a single item, or -1

Since 2.6

`gtk-icon-view-set-spacing` (*self* `<gtk-icon-view>`) (*spacing* `int`) [Function]
`set-spacing` [Method]

Sets the ::spacing property which specifies the space which is inserted between the cells (i.e. the icon and the text) of an item.

icon-view a `<gtk-icon-view>`

spacing the spacing

Since 2.6

`gtk-icon-view-get-spacing` (*self* `<gtk-icon-view>`) ⇒ (*ret* `int`) [Function]
`get-spacing` [Method]

Returns the value of the ::spacing property.

icon-view a `<gtk-icon-view>`

ret the space between cells

Since 2.6

`gtk-icon-view-set-row-spacing` (*self* `<gtk-icon-view>`) (*row_spacing* `int`) [Function]
`set-row-spacing` [Method]

Sets the ::row-spacing property which specifies the space which is inserted between the rows of the icon view.

icon-view a `<gtk-icon-view>`

row-spacing the row spacing

Since 2.6

`gtk-icon-view-get-row-spacing` (*self* `<gtk-icon-view>`) ⇒ (*ret* `int`) [Function]
`get-row-spacing` [Method]

Returns the value of the ::row-spacing property.

icon-view a `<gtk-icon-view>`

ret the space between rows

Since 2.6

`gtk-icon-view-set-column-spacing` (*self* `<gtk-icon-view>`) (*column_spacing* `int`) [Function]
`set-column-spacing` [Method]

Sets the ::column-spacing property which specifies the space which is inserted between the columns of the icon view.

icon-view a `<gtk-icon-view>`

column-spacing
the column spacing

Since 2.6

`gtk-icon-view-get-column-spacing` (*self* `<gtk-icon-view>`) ⇒ (*ret* `int`) [Function]
`get-column-spacing` [Method]

Returns the value of the ::column-spacing property.

icon-view a `<gtk-icon-view>`

ret the space between columns

Since 2.6

`gtk-icon-view-set-margin` (*self* `<gtk-icon-view>`) (*margin* `int`) [Function]
`set-margin` [Method]

Sets the ::margin property which specifies the space which is inserted at the top, bottom, left and right of the icon view.

icon-view a `<gtk-icon-view>`

margin the margin

Since 2.6

`gtk-icon-view-get-margin` (*self* `<gtk-icon-view>`) ⇒ (*ret* `int`) [Function]
`get-margin` [Method]

Returns the value of the ::margin property.

icon-view a `<gtk-icon-view>`

ret the space at the borders

Since 2.6

`gtk-icon-view-select-path` (*self* `<gtk-icon-view>`) (*path* `<gtk-tree-path>`) [Function]
`select-path` [Method]

Selects the row at *path*.

icon-view A `<gtk-icon-view>`.

path The `<gtk-tree-path>` to be selected.

Since 2.6

`gtk-icon-view-unselect-path` (*self* `<gtk-icon-view>`) [Function]
(*path* `<gtk-tree-path>`)
`unselect-path` [Method]

Unselects the row at *path*.

icon-view A `<gtk-icon-view>`.

path The `<gtk-tree-path>` to be unselected.

Since 2.6

`gtk-icon-view-path-is-selected` (*self* `<gtk-icon-view>`) [Function]
(*path* `<gtk-tree-path>`) ⇒ (*ret* `bool`)
`path-is-selected` [Method]

Returns '`#t`' if the icon pointed to by *path* is currently selected. If *icon* does not point to a valid location, '`#f`' is returned.

icon-view A `<gtk-icon-view>`.

path A `<gtk-tree-path>` to check selection on.

ret '`#t`' if *path* is selected.

Since 2.6

`gtk-icon-view-get-selected-items` (*self* `<gtk-icon-view>`) [Function]
⇒ (*ret* `glist-of`)
`get-selected-items` [Method]

Creates a list of paths of all selected items. Additionally, if you are planning on modifying the model after calling this function, you may want to convert the returned list into a list of `<gtk-tree-row-reference>`s. To do this, you can use `gtk-tree-row-reference-new`.

To free the return value, use:

```
g_list_foreach (list, gtk_tree_path_free, NULL);
g_list_free (list);
```

icon-view A `<gtk-icon-view>`.

ret A `<g-list>` containing a `<gtk-tree-path>` for each selected row.

Since 2.6

`gtk-icon-view-select-all` (*self* `<gtk-icon-view>`) [Function]
`select-all` [Method]

Selects all the icons. *icon-view* must has its selection mode set to `<gtk-selection-multiple>`.

icon-view A `<gtk-icon-view>`.

Since 2.6

`gtk-icon-view-unselect-all` (*self* `<gtk-icon-view>`) [Function]
`unselect-all` [Method]

Unselects all the icons.

icon-view A `<gtk-icon-view>`.

Since 2.6

`gtk-icon-view-item-activated` (*self* `<gtk-icon-view>`) [Function]
(*path* `<gtk-tree-path>`)

`item-activated` [Method]

Activates the item determined by *path*.

icon-view A `<gtk-icon-view>`

path The `<gtk-tree-path>` to be activated

Since 2.6

`gtk-icon-view-scroll-to-path` (*self* `<gtk-icon-view>`) [Function]
(*path* `<gtk-tree-path>`) (*use_align* `bool`) (*row_align* `float`)
(*col_align* `float`)

`scroll-to-path` [Method]

Moves the alignments of *icon-view* to the position specified by *path*. *row-align* determines where the row is placed, and *col-align* determines where *column* is placed. Both are expected to be between 0.0 and 1.0. 0.0 means left/top alignment, 1.0 means right/bottom alignment, 0.5 means center.

If *use-align* is '`#f`', then the alignment arguments are ignored, and the tree does the minimum amount of work to scroll the item onto the screen. This means that the item will be scrolled to the edge closest to its current position. If the item is currently visible on the screen, nothing is done.

This function only works if the model is set, and *path* is a valid row on the model. If the model changes before the *icon-view* is realized, the centered path will be modified to reflect this change.

icon-view A `<gtk-icon-view>`.

path The path of the item to move to.

use-align whether to use alignment arguments, or '`#f`'.

row-align The vertical alignment of the item specified by *path*.

col-align The horizontal alignment of the item specified by *path*.

Since 2.8

`gtk-icon-view-get-visible-range` (*self* `<gtk-icon-view>`) [Function]
(*start_path* `<gtk-tree-path**>`) (*end_path* `<gtk-tree-path**>`)
⇒ (*ret* `bool`)

`get-visible-range` [Method]

Sets *start-path* and *end-path* to be the first and last visible path. Note that there may be invisible paths in between.

Both paths should be freed with `gtk-tree-path-free` after use.

icon-view A `<gtk-icon-view>`

start-path Return location for start of region, or '`#f`'

end-path Return location for end of region, or '`#f`'

ret '`#t`', if valid paths were placed in *start-path* and *end-path*

Since 2.8

`gtk-icon-view-unset-model-drag-dest` (*self* `<gtk-icon-view>`) [Function]
`unset-model-drag-dest` [Method]

Undoes the effect of `gtk-icon-view-enable-model-drag-dest`.

icon-view a `<gtk-icon-view>`

Since 2.8

`gtk-icon-view-set-reorderable` (*self* `<gtk-icon-view>`) [Function]
(*reorderable* `bool`)
`set-reorderable` [Method]

This function is a convenience function to allow you to reorder models that support the `<gtk-tree-drag-source-iface>` and the `<gtk-tree-drag-dest-iface>`. Both `<gtk-tree-store>` and `<gtk-list-store>` support these. If *reorderable* is '`#t`', then the user can reorder the model by dragging and dropping rows. The developer can listen to these changes by connecting to the model's row_inserted and row_deleted signals.

This function does not give you any degree of control over the order – any reordering is allowed. If more control is needed, you should probably handle drag and drop manually.

icon-view A `<gtk-icon-view>`.

reorderable
'`#t`', if the list of items can be reordered.

Since 2.8

`gtk-icon-view-get-reorderable` (*self* `<gtk-icon-view>`) [Function]
⇒ (*ret* `bool`)
`get-reorderable` [Method]

Retrieves whether the user can reorder the list via drag-and-drop. See `gtk-icon-view-set-reorderable`.

icon-view a `<gtk-icon-view>`

ret '`#t`' if the list can be reordered.

Since 2.8

`gtk-icon-view-set-drag-dest-item` (*self* `<gtk-icon-view>`) [Function]
(*path* `<gtk-tree-path>`) (*pos* `<gtk-icon-view-drop-position>`)
`set-drag-dest-item` [Method]

Sets the item that is highlighted for feedback.

icon-view a `<gtk-icon-view>`

path The path of the item to highlight, or '`#f`'.

pos Specifies where to drop, relative to the item

Since 2.8

`gtk-icon-view-get-drag-dest-item` (*self* `<gtk-icon-view>`) [Function]
(*path* `<gtk-tree-path**>`) (*pos* `<gtk-icon-view-drop-position*>`)
`get-drag-dest-item` [Method]

Gets information about the item that is highlighted for feedback.

icon-view a `<gtk-icon-view>`

path Return location for the path of the highlighted item, or '`#f`'.

pos Return location for the drop position, or '`#f`'

Since 2.8

`gtk-icon-view-get-dest-item-at-pos` (*self* `<gtk-icon-view>`) [Function]
(*drag_x* `int`) (*drag_y* `int`) (*path* `<gtk-tree-path**>`)
(*pos* `<gtk-icon-view-drop-position*>`) ⇒ (*ret* `bool`)
`get-dest-item-at-pos` [Method]

Determines the destination item for a given position.

icon-view a `<gtk-icon-view>`

drag-x the position to determine the destination item for

drag-y the position to determine the destination item for

path Return location for the path of the item, or '`#f`'.

pos Return location for the drop position, or '`#f`'

ret whether there is an item at the given position.

Since 2.8

`gtk-icon-view-create-drag-icon` (*self* `<gtk-icon-view>`) [Function]
(*path* `<gtk-tree-path>`) ⇒ (*ret* `<gdk-pixmap*>`)
`create-drag-icon` [Method]

Creates a `<gdk-pixmap>` representation of the item at *path*. This image is used for a drag icon.

icon-view a `<gtk-icon-view>`

path a `<gtk-tree-path>` in *icon-view*

ret a newly-allocated pixmap of the drag icon.

Since 2.8

39 GtkTreeSortable

The interface for sortable models used by GtkTreeView

39.1 Overview

`<gtk-tree-sortable>` is an interface to be implemented by tree models which support sorting. The `<gtk-tree-view>` uses the methods provided by this interface to sort the model.

39.2 Usage

`<gtk-tree-sortable>` [Class]

This `<gobject>` class defines no properties, other than those defined by its superclasses.

`sort-column-changed` [Signal on `<gtk-tree-sortable>`]

`gtk-tree-sortable-set-sort-func` (*self* `<gtk-tree-sortable>`) [Function]
(*sort_column_id* `int`) (*sort_func* `<gtk-tree-iter-compare-func>`)
(*user_data* `<gpointer>`) (*destroy* `<gtk-destroy-notify>`)

`set-sort-func` [Method]

Sets the comparison function used when sorting to be *sort-func*. If the current sort column id of *sortable* is the same as *sort-column-id*, then the model will sort using this function.

sortable A `<gtk-tree-sortable>`

sort-column-id
the sort column id to set the function for

sort-func The comparison function

user-data User data to pass to *sort-func*, or '`#f`'

destroy Destroy notifier of *user-data*, or '`#f`'

40 GtkTreeModelSort

A GtkTreeModel which makes an underlying tree model sortable

40.1 Overview

The <gtk-tree-model-sort> is a model which implements the <gtk-tree-sortable> interface. It does not hold any data itself, but rather is created with a child model and proxies its data. It has identical column types to this child model, and the changes in the child are propagated. The primary purpose of this model is to provide a way to sort a different model without modifying it. Note that the sort function used by <gtk-tree-model-sort> is not guaranteed to be stable.

The use of this is best demonstrated through an example. In the following sample code we create two <gtk-tree-view> widgets each with a view of the same data. As the model is wrapped here by a <gtk-tree-model-sort>, the two <gtk-tree-view>s can each sort their view of the data without affecting the other. By contrast, if we simply put the same model in each widget, then sorting the first would sort the second.

```
{
  GtkTreeView *tree_view1;
  GtkTreeView *tree_view2;
  GtkTreeModel *sort_model1;
  GtkTreeModel *sort_model2;
  GtkTreeModel *child_model;

  /* get the child model */
  child_model = get_my_model ();

  /* Create the first tree */
  sort_model1 = gtk_tree_model_sort_new_with_model (child_model);
  tree_view1 = gtk_tree_view_new_with_model (sort_model1);

  /* Create the second tree */
  sort_model2 = gtk_tree_model_sort_new_with_model (child_model);
  tree_view2 = gtk_tree_view_new_with_model (sort_model2);

  /* Now we can sort the two models independently */
  gtk_tree_sortable_set_sort_column_id (GTK_TREE_SORTABLE (sort_model1),
                                        COLUMN_1, GTK_SORT_ASCENDING);
  gtk_tree_sortable_set_sort_column_id (GTK_TREE_SORTABLE (sort_model2),
                                        COLUMN_1, GTK_SORT_DESCENDING);
}
```

To demonstrate how to access the underlying child model from the sort model, the next example will be a callback for the <gtk-tree-selection> "changed" signal. In this callback, we get a string from COLUMN_1 of the model. We then modify the string, find the same selected row on the child model, and change the row there.

```
void
selection_changed (GtkTreeSelection *selection, gpointer data)
{
  GtkTreeModel *sort_model = NULL;
  GtkTreeModel *child_model;
  GtkTreeIter sort_iter;
  GtkTreeIter child_iter;
  char *some_data = NULL;
  char *modified_data;

  /* Get the current selected row and the model. */
  if (! gtk_tree_selection_get_selected (selection,
                                         &sort_model,
                                         &sort_iter))
    return;


  /* Look up the current value on the selected row and get a new value
   * to change it to.
   */
  gtk_tree_model_get (GTK_TREE_MODEL (sort_model), &sort_iter,
                      COLUMN_1, &some_data,
                      -1);

  modified_data = change_the_data (some_data);
  g_free (some_data);

  /* Get an iterator on the child model, instead of the sort model. */
  gtk_tree_model_sort_convert_iter_to_child_iter (GTK_TREE_MODEL_SORT (sort_model),
                                                  &child_iter,
                                                  &sort_iter);

  /* Get the child model and change the value of the row.  In this
   * example, the child model is a GtkListStore.  It could be any other
   * type of model, though.
   */
  child_model = gtk_tree_model_sort_get_model (GTK_TREE_MODEL_SORT (sort_model));
  gtk_list_store_set (GTK_LIST_STORE (child_model), &child_iter,
                      COLUMN_1, &modified_data,
                      -1);
  g_free (modified_data);
}
```

40.2 Usage

`<gtk-tree-model-sort>` [Class]

This `<gobject>` class defines the following properties:

`model` The model for the TreeModelSort to sort

`gtk-tree-model-sort-new-with-model` [Function]
(*child_model* `<gtk-tree-model>`) ⇒ (*ret* `<gtk-tree-model>`)

Creates a new `<gtk-tree-model>`, with *child-model* as the child model.

child-model
A `<gtk-tree-model>`

ret A new `<gtk-tree-model>`.

`gtk-tree-model-sort-get-model` (*self* `<gtk-tree-model-sort>`) [Function]
⇒ (*ret* `<gtk-tree-model>`)
`get-model` [Method]

Returns the model the `<gtk-tree-model-sort>` is sorting.

tree-model
a `<gtk-tree-model-sort>`

ret the "child model" being sorted

`gtk-tree-model-sort-clear-cache` (*self* `<gtk-tree-model-sort>`) [Function]
`clear-cache` [Method]

This function should almost never be called. It clears the *tree-model-sort* of any cached iterators that haven't been reffed with `gtk-tree-model-ref-node`. This might be useful if the child model being sorted is static (and doesn't change often) and there has been a lot of unreffed access to nodes. As a side effect of this function, all unreffed iters will be invalid.

tree-model-sort
A `<gtk-tree-model-sort>`

`gtk-tree-model-sort-iter-is-valid` [Function]
(*self* `<gtk-tree-model-sort>`) (*iter* `<gtk-tree-iter>`) ⇒ (*ret* `bool`)
`iter-is-valid` [Method]

This function is slow. Only use it for debugging and/or testing purposes.

Checks if the given iter is a valid iter for this `<gtk-tree-model-sort>`.

tree-model-sort
A `<gtk-tree-model-sort>`.

iter A `<gtk-tree-iter>`.

ret '`#t`' if the iter is valid, '`#f`' if the iter is invalid.

Since 2.2

41 GtkTreeModelFilter

A GtkTreeModel which hides parts of an underlying tree model

41.1 Overview

A <gtk-tree-model-filter> is a tree model which wraps another tree model, and can do the following things:

Filter specific rows, based on data from a "visible column", a column storing booleans indicating whether the row should be filtered or not, or based on the return value of a "visible function", which gets a model, iter and user_data and returns a boolean indicating whether the row should be filtered or not.

Modify the "appearance" of the model, using a modify function. This is extremely powerful and allows for just changing some values and also for creating a completely different model based on the given child model.

Set a different root node, also known as a "virtual root". You can pass in a <gtk-tree-path> indicating the root node for the filter at construction time.

41.2 Usage

<gtk-tree-model-filter> [Class]

This <gobject> class defines the following properties:

child-model
: The model for the filtermodel to filter

virtual-root
: The virtual root (relative to the child model) for this filtermodel

gtk-tree-model-filter-new (*self* <gtk-tree-model>) [Function]
(*root* <gtk-tree-path>) ⇒ (*ret* <gtk-tree-model>)

filter-new [Method]

Creates a new <gtk-tree-model>, with *child-model* as the child_model and *root* as the virtual root.

child-model
: A <gtk-tree-model>.

root
: A <gtk-tree-path> or '#f'.

ret
: A new <gtk-tree-model>.

Since 2.4

gtk-tree-model-filter-get-model [Function]
(*self* <gtk-tree-model-filter>) ⇒ (*ret* <gtk-tree-model>)

get-model [Method]

Returns a pointer to the child model of *filter*.

filter
: A <gtk-tree-model-filter>.

ret
: A pointer to a <gtk-tree-model>.

Since 2.4

`gtk-tree-model-filter-refilter` (*self* `<gtk-tree-model-filter>`) [Function]
`refilter` [Method]

Emits ::row_changed for each row in the child model, which causes the filter to re-evaluate whether a row is visible or not.

filter A `<gtk-tree-model-filter>`.

Since 2.4

`gtk-tree-model-filter-clear-cache` [Function]
(*self* `<gtk-tree-model-filter>`)
`clear-cache` [Method]

This function should almost never be called. It clears the *filter* of any cached iterators that haven't been reffed with `gtk-tree-model-ref-node`. This might be useful if the child model being filtered is static (and doesn't change often) and there has been a lot of unreffed access to nodes. As a side effect of this function, all unreffed iters will be invalid.

filter A `<gtk-tree-model-filter>`.

Since 2.4

42 GtkCellLayout

An interface for packing cells

42.1 Overview

`<gtk-cell-layout>` is an interface to be implemented by all objects which want to provide a `<gtk-tree-view-column-like>` API for packing cells, setting attributes and data funcs.

One of the notable features provided by implementations of GtkCellLayout are *attributes*. Attributes let you set the properties in flexible ways. They can just be set to constant values like regular properties. But they can also be mapped to a column of the underlying tree model with `gtk-cell-layout-set-attributes`, which means that the value of the attribute can change from cell to cell as they are rendered by the cell renderer. Finally, it is possible to specify a function with `gtk-cell-layout-set-cell-data-func` that is called to determine the value of the attribute for each cell that is rendered.

42.2 Usage

`gtk-cell-layout-pack-start` (*self* `<gtk-cell-layout*>`) [Function]
(*cell* `<gtk-cell-renderer>`) (*expand* `bool`)

Packs the *cell* into the beginning of *cell-layout*. If *expand* is '`#f`', then the *cell* is allocated no more space than it needs. Any unused space is divided evenly between cells for which *expand* is '`#t`'.

Note that reusing the same cell renderer is not supported.

cell-layout A `<gtk-cell-layout>`.

cell A `<gtk-cell-renderer>`.

expand '`#t`' if *cell* is to be given extra space allocated to *cell-layout*.

Since 2.4

`gtk-cell-layout-pack-end` (*self* `<gtk-cell-layout*>`) [Function]
(*cell* `<gtk-cell-renderer>`) (*expand* `bool`)

Adds the *cell* to the end of *cell-layout*. If *expand* is '`#f`', then the *cell* is allocated no more space than it needs. Any unused space is divided evenly between cells for which *expand* is '`#t`'.

Note that reusing the same cell renderer is not supported.

cell-layout A `<gtk-cell-layout>`.

cell A `<gtk-cell-renderer>`.

expand '`#t`' if *cell* is to be given extra space allocated to *cell-layout*.

Since 2.4

`gtk-cell-layout-reorder` (*self* `<gtk-cell-layout*>`) [Function]
(*cell* `<gtk-cell-renderer>`) (*position* `int`)

Re-inserts *cell* at *position*. Note that *cell* has already to be packed into *cell-layout* for this to function properly.

cell-layout A `<gtk-cell-layout>`.

cell A `<gtk-cell-renderer>` to reorder.

position New position to insert *cell* at.

Since 2.4

`gtk-cell-layout-clear` (*self* `<gtk-cell-layout*>`) [Function]

Unsets all the mappings on all renderers on *cell-layout* and removes all renderers from *cell-layout*.

cell-layout A `<gtk-cell-layout>`.

Since 2.4

`gtk-cell-layout-add-attribute` (*self* `<gtk-cell-layout*>`) [Function]
(*cell* `<gtk-cell-renderer>`) (*attribute* `mchars`) (*column* `int`)

Adds an attribute mapping to the list in *cell-layout*. The *column* is the column of the model to get a value from, and the *attribute* is the parameter on *cell* to be set from the value. So for example if column 2 of the model contains strings, you could have the "text" attribute of a `<gtk-cell-renderer-text>` get its values from column 2.

cell-layout A `<gtk-cell-layout>`.

cell A `<gtk-cell-renderer>`.

attribute An attribute on the renderer.

column The column position on the model to get the attribute from.

Since 2.4

`gtk-cell-layout-set-cell-data-func` (*self* `<gtk-cell-layout*>`) [Function]
(*cell* `<gtk-cell-renderer>`) (*func* `<gtk-cell-layout-data-func>`)
(*func_data* `<gpointer>`) (*destroy* `<g-destroy-notify>`)

Sets the `<gtk-cell-layout-data-func>` to use for *cell-layout*. This function is used instead of the standard attributes mapping for setting the column value, and should set the value of *cell-layout*'s cell renderer(s) as appropriate. *func* may be '`#f`' to remove and older one.

cell-layout A `<gtk-cell-layout>`.

cell A `<gtk-cell-renderer>`.

func The `<gtk-cell-layout-data-func>` to use.

func-data The user data for *func*.

destroy The destroy notification for *func-data*.

Since 2.4

`gtk-cell-layout-clear-attributes` (*self* `<gtk-cell-layout*>`) [Function]
(*cell* `<gtk-cell-renderer>`)

Clears all existing attributes previously set with `gtk-cell-layout-set-attributes`.

cell-layout A `<gtk-cell-layout>`.

cell A `<gtk-cell-renderer>` to clear the attribute mapping on.

Since 2.4

43 GtkCellRenderer

An object for rendering a single cell on a

43.1 Overview

The <gtk-cell-renderer> is a base class of a set of objects used for rendering a cell to a <gdk-drawable>. These objects are used primarily by the <gtk-tree-view> widget, though they aren't tied to them in any specific way. It is worth noting that <gtk-cell-renderer> is not a <gtk-widget> and cannot be treated as such.

The primary use of a <gtk-cell-renderer> is for drawing a certain graphical elements on a <gdk-drawable>. Typically, one cell renderer is used to draw many cells on the screen. To this extent, it isn't expected that a CellRenderer keep any permanent state around. Instead, any state is set just prior to use using <gobject>s property system. Then, the cell is measured using gtk-cell-renderer-get-size. Finally, the cell is rendered in the correct location using gtk-cell-renderer-render.

There are a number of rules that must be followed when writing a new <gtk-cell-renderer>. First and formost, it's important that a certain set of properties will always yield a cell renderer of the same size, barring a <gtk-style> change. The <gtk-cell-renderer> also has a number of generic properties that are expected to be honored by all children.

Beyond merely rendering a cell, cell renderers can optionally provide active user interface elements. A cell renderer can be *activatable* like <gtk-cell-renderer-toggle>, which toggles when it gets activated by a mouse click, or it can be *editable* like <gtk-cell-renderer-text>, which allows the user to edit the text using a <gtk-entry>. To make a cell renderer activatable or editable, you have to implement the *activate* or *start-editing* virtual functions, respectively.

43.2 Usage

<gtk-cell-renderer> [Class]

This <gobject> class defines the following properties:

mode	Editable mode of the CellRenderer
visible	Display the cell
sensitive	Display the cell sensitive
xalign	The x-align
yalign	The y-align
xpad	The xpad
ypad	The ypad
width	The fixed width
height	The fixed height

`is-expander`
: Row has children

`is-expanded`
: Row is an expander row, and is expanded

`cell-background`
: Cell background color as a string

`cell-background-gdk`
: Cell background color as a GdkColor

`cell-background-set`
: Whether this tag affects the cell background color

`editing-canceled` [Signal on `<gtk-cell-renderer>`]

This signal gets emitted when the user cancels the process of editing a cell. For example, an editable cell renderer could be written to cancel editing when the user presses Escape.

See also: `gtk-cell-renderer-editing-canceled`

Since 2.4

`editing-started` [Signal on `<gtk-cell-renderer>`]

(*arg0* `<gtk-cell-editable>`) (*arg1* `<gchararray>`)

This signal gets emitted when a cell starts to be edited. The indended use of this signal is to do special setup on *editable*, e.g. adding a `<gtk-entry-completion>` or setting up additional columns in a `<gtk-combo-box>`.

Note that GTK+ doesn't guarantee that cell renderers will continue to use the same kind of widget for editing in future releases, therefore you should check the type of *editable* before doing any specific setup, as in the following example:

```
static void
text_editing_started (GtkCellRenderer *cell,
                      GtkCellEditable *editable,
                      const gchar     *path,
                      gpointer         data)
{
  if (GTK_IS_ENTRY (editable))
    {
      GtkEntry *entry = GTK_ENTRY (editable);

      /* ... create a GtkEntryCompletion */

      gtk_entry_set_completion (entry, completion);
    }
}
```

Since 2.6

`gtk-cell-renderer-render` (*self* `<gtk-cell-renderer>`) [Function]
(*window* `<gdk-window*>`) (*widget* `<gtk-widget>`)
(*background_area* `<gdk-rectangle>`) (*cell_area* `<gdk-rectangle>`)
(*expose_area* `<gdk-rectangle>`) (*flags* `<gtk-cell-renderer-state>`)

`render` [Method]

Invokes the virtual render function of the `<gtk-cell-renderer>`. The three passed-in rectangles are areas of *window*. Most renderers will draw within *cell-area*; the xalign, yalign, xpad, and ypad fields of the `<gtk-cell-renderer>` should be honored with respect to *cell-area*. *background-area* includes the blank space around the cell, and also the area containing the tree expander; so the *background-area* rectangles for all cells tile to cover the entire *window*. *expose-area* is a clip rectangle.

cell a `<gtk-cell-renderer>`

window a `<gdk-drawable>` to draw to

widget the widget owning *window*

background-area
entire cell area (including tree expanders and maybe padding on the sides)

cell-area area normally rendered by a cell renderer

expose-area
area that actually needs updating

flags flags that affect rendering

`gtk-cell-renderer-activate` (*self* `<gtk-cell-renderer>`) [Function]
(*event* `<gdk-event>`) (*widget* `<gtk-widget>`) (*path* `mchars`)
(*background_area* `<gdk-rectangle>`) (*cell_area* `<gdk-rectangle>`)
(*flags* `<gtk-cell-renderer-state>`) ⇒ (*ret* `bool`)

`activate` [Method]

Passes an activate event to the cell renderer for possible processing. Some cell renderers may use events; for example, `<gtk-cell-renderer-toggle>` toggles when it gets a mouse click.

cell a `<gtk-cell-renderer>`

event a `<gdk-event>`

widget widget that received the event

path widget-dependent string representation of the event location; e.g. for `<gtk-tree-view>`, a string representation of `<gtk-tree-path>`

background-area
background area as passed to *gtk-cell-renderer-render*

cell-area cell area as passed to *gtk-cell-renderer-render*

flags render flags

ret '`#t`' if the event was consumed/handled

`gtk-cell-renderer-start-editing` (*self* `<gtk-cell-renderer>`) [Function]
(*event* `<gdk-event>`) (*widget* `<gtk-widget>`) (*path* `mchars`)
(*background_area* `<gdk-rectangle>`) (*cell_area* `<gdk-rectangle>`)
(*flags* `<gtk-cell-renderer-state>`) ⇒ (*ret* `<gtk-cell-editable>`)

`start-editing` [Method]

Passes an activate event to the cell renderer for possible processing.

cell a `<gtk-cell-renderer>`

event a `<gdk-event>`

widget widget that received the event

path widget-dependent string representation of the event location; e.g. for `<gtk-tree-view>`, a string representation of `<gtk-tree-path>`

background-area background area as passed to *gtk-cell-renderer-render*

cell-area cell area as passed to *gtk-cell-renderer-render*

flags render flags

ret A new `<gtk-cell-editable>`, or '`#f`'

`gtk-cell-renderer-editing-canceled` (*self* `<gtk-cell-renderer>`) [Function]

`editing-canceled` [Method]

'`gtk_cell_renderer_editing_canceled`' has been deprecated since version 2.6 and should not be used in newly-written code. Use `gtk-cell-renderer-stop-editing` instead

Causes the cell renderer to emit the "editing-canceled" signal. This function is for use only by implementations of cell renderers that need to notify the client program that an editing process was canceled and the changes were not committed.

cell A `<gtk-cell-renderer>`

Since 2.4

`gtk-cell-renderer-stop-editing` (*self* `<gtk-cell-renderer>`) [Function]
(*canceled* `bool`)

`stop-editing` [Method]

Informs the cell renderer that the editing is stopped. If *canceled* is '`#t`', the cell renderer will emit the "editing-canceled" signal. This function should be called by cell renderer implementations in response to the "editing-done" signal of `<gtk-cell-editable>`.

cell A `<gtk-cell-renderer>`

canceled '`#t`' if the editing has been canceled

Since 2.6

`gtk-cell-renderer-get-fixed-size` (*self* `<gtk-cell-renderer>`) [Function]
⇒ (*width* `int`) (*height* `int`)

`get-fixed-size` [Method]

Fills in *width* and *height* with the appropriate size of *cell*.

cell A `<gtk-cell-renderer>`

width location to fill in with the fixed width of the widget, or '`#f`'

height location to fill in with the fixed height of the widget, or '`#f`'

`gtk-cell-renderer-set-fixed-size` (*self* `<gtk-cell-renderer>`) (*width* `int`) (*height* `int`) [Function]

`set-fixed-size` [Method]

Sets the renderer size to be explicit, independent of the properties set.

cell A `<gtk-cell-renderer>`

width the width of the cell renderer, or -1

height the height of the cell renderer, or -1

44 GtkCellEditable

Interface for widgets which can are used for editing cells

44.1 Overview

The <gtk-cell-editable> interface must be implemented for widgets to be usable when editing the contents of a <gtk-tree-view> cell.

44.2 Usage

<gtk-cell-editable> [Class]
This <gobject> class defines no properties, other than those defined by its superclasses.

editing-done [Signal on <gtk-cell-editable>]

remove-widget [Signal on <gtk-cell-editable>]

gtk-cell-editable-start-editing (*self* <gtk-cell-editable>) [Function]
(*event* <gdk-event>)
start-editing [Method]
Begins editing on a *cell-editable*. *event* is the <gdk-event> that began the editing process. It may be '#f', in the instance that editing was initiated through programatic means.

cell-editable
A <gtk-cell-editable>

event A <gdk-event>, or '#f'

gtk-cell-editable-editing-done (*self* <gtk-cell-editable>) [Function]
editing-done [Method]
Emits the "editing_done" signal. This signal is a sign for the cell renderer to update its value from the cell.

cell-editable
A <gtk-tree-editable>

gtk-cell-editable-remove-widget (*self* <gtk-cell-editable>) [Function]
remove-widget [Method]
Emits the "remove_widget" signal. This signal is meant to indicate that the cell is finished editing, and the widget may now be destroyed.

cell-editable
A <gtk-tree-editable>

45 GtkCellRendererAccel

Renders a keyboard accelerator in a cell

45.1 Overview

<gtk-cell-renderer-accel> displays a keyboard accelerator (i.e. a key combination like <Control>-a). If the cell renderer is editable, the accelerator can be changed by simply typing the new combination.

The <gtk-cell-renderer-accel> cell renderer was added in GTK+ 2.10.

45.2 Usage

<gtk-cell-renderer-accel> [Class]

This <gobject> class defines the following properties:

accel-key
: The keyval of the accelerator

accel-mods
: The modifier mask of the accelerator

keycode
: The hardware keycode of the accelerator

accel-mode
: The type of accelerators

accel-edited (*arg0* <gchararray>) [Signal on <gtk-cell-renderer-accel>]
(*arg1* <guint>) (*arg2* <gdk-modifier-type>) (*arg3* <guint>)

Gets emitted when the user has selected a new accelerator.

Since 2.10

accel-cleared (*arg0* <gchararray>) [Signal on <gtk-cell-renderer-accel>]

Gets emitted when the user has removed the accelerator.

Since 2.10

gtk-cell-renderer-accel-new ⇒ (*ret* <gtk-cell-renderer>) [Function]

Creates a new <gtk-cell-renderer-accel>.

ret
: the new cell renderer

Since 2.10

46 GtkCellRendererCombo

Renders a combobox in a cell

46.1 Overview

<gtk-cell-renderer-combo> renders text in a cell like <gtk-cell-renderer-text> from which it is derived. But while <gtk-cell-renderer-text> offers a simple entry to edit the text, <gtk-cell-renderer-combo> offers a <gtk-combo-box> or <gtk-combo-box-entry> widget to edit the text. The values to display in the combo box are taken from the tree model specified in the model property.

The combo cell renderer takes care of adding a text cell renderer to the combo box and sets it to display the column specified by its text-column property. Further properties of the comnbo box can be set in a handler for the editing-started signal.

The <gtk-cell-renderer-combo> cell renderer was added in GTK+ 2.6.

46.2 Usage

<gtk-cell-renderer-combo> [Class]

This <gobject> class defines the following properties:

`model` The model containing the possible values for the combo box

`text-column`
A column in the data source model to get the strings from

`has-entry`
If FALSE, don't allow to enter strings other than the chosen ones

gtk-cell-renderer-combo-new ⇒ (*ret* <gtk-cell-renderer>) [Function]

Creates a new <gtk-cell-renderer-combo>. Adjust how text is drawn using object properties. Object properties can be set globally (with g-object-set). Also, with <gtk-tree-view-column>, you can bind a property to a value in a <gtk-tree-model>. For example, you can bind the "text" property on the cell renderer to a string value in the model, thus rendering a different string in each row of the <gtk-tree-view>.

ret the new cell renderer

Since 2.6

47 GtkCellRendererPixbuf

Renders a pixbuf in a cell

47.1 Overview

A <gtk-cell-renderer-pixbuf> can be used to render an image in a cell. It allows to render either a given <gdk-pixbuf> (set via the pixbuf property) or a stock icon (set via the stock-id property).

To support the tree view, <gtk-cell-renderer-pixbuf> also supports rendering two alternative pixbufs, when the is-expander property is '#t'. If the is-expanded property is '#t' and the pixbuf-expander-open property is set to a pixbuf, it renders that pixbuf, if the is-expanded property is '#f' and the pixbuf-expander-closed property is set to a pixbuf, it renders that one.

47.2 Usage

<gtk-cell-renderer-pixbuf> [Class]

This <gobject> class defines the following properties:

pixbuf The pixbuf to render

pixbuf-expander-open
Pixbuf for open expander

pixbuf-expander-closed
Pixbuf for closed expander

stock-id The stock ID of the stock icon to render

stock-size
The GtkIconSize value that specifies the size of the rendered icon

stock-detail
Render detail to pass to the theme engine

follow-state
Whether the rendered pixbuf should be colorized according to the state

icon-name
The name of the icon from the icon theme

gtk-cell-renderer-pixbuf-new ⇒ (*ret* <gtk-cell-renderer>) [Function]

Creates a new <gtk-cell-renderer-pixbuf>. Adjust rendering parameters using object properties. Object properties can be set globally (with g-object-set). Also, with <gtk-tree-view-column>, you can bind a property to a value in a <gtk-tree-model>. For example, you can bind the "pixbuf" property on the cell renderer to a pixbuf value in the model, thus rendering a different image in each row of the <gtk-tree-view>.

ret the new cell renderer

48 GtkCellRendererProgress

Renders numbers as progress bars

48.1 Overview

`<gtk-cell-renderer-progress>` renders a numeric value as a progress par in a cell. Additionally, it can display a text on top of the progress bar.

The `<gtk-cell-renderer-progress>` cell renderer was added in GTK+ 2.6.

48.2 Usage

`<gtk-cell-renderer-progress>` [Class]

This `<gobject>` class defines the following properties:

`value` Value of the progress bar

`text` Text on the progress bar

`pulse` Set this to positive values to indicate that some progress is made, but you don't know how much.

`text-xalign`
The horizontal text alignment, from 0 (left) to 1 (right). Reversed for RTL layouts.

`text-yalign`
The vertical text alignment, from 0 (top) to 1 (bottom).

`orientation`
Orientation and growth direction of the progress bar

`gtk-cell-renderer-progress-new` ⇒ (*ret* `<gtk-cell-renderer>`) [Function]

Creates a new `<gtk-cell-renderer-progress>`.

ret the new cell renderer

Since 2.6

49 GtkCellRendererSpin

Renders a spin button in a cell

49.1 Overview

`<gtk-cell-renderer-spin>` renders text in a cell like `<gtk-cell-renderer-text>` from which it is derived. But while `<gtk-cell-renderer-text>` offers a simple entry to edit the text, `<gtk-cell-renderer-spin>` offers a `<gtk-spin-button>` widget. Of course, that means that the text has to be parseable as a floating point number.

The range of the spinbutton is taken from the adjustment property of the cell renderer, which can be set explicitly or mapped to a column in the tree model, like all properties of cell renders. `<gtk-cell-renderer-spin>` also has properties for the climb rate and the number of digits to display. Other `<gtk-spin-button>` properties can be set in a handler for the start-editing signal.

The `<gtk-cell-renderer-spin>` cell renderer was added in GTK+ 2.10.

49.2 Usage

`<gtk-cell-renderer-spin>` [Class]

This `<gobject>` class defines the following properties:

`adjustment`
: The adjustment that holds the value of the spinbutton.

`climb-rate`
: The acceleration rate when you hold down a button

`digits`
: The number of decimal places to display

`gtk-cell-renderer-spin-new` ⇒ (*ret* `<gtk-cell-renderer>`) [Function]

Creates a new `<gtk-cell-renderer-spin>`.

ret
: a new `<gtk-cell-renderer-spin>`

Since 2.10

50 GtkCellRendererText

Renders text in a cell

50.1 Overview

A <gtk-cell-renderer-text> renders a given text in its cell, using the font, color and style information provided by its properties. The text will be ellipsized if it is too long and the ellipsize property allows it.

If the mode is 'GTK_CELL_RENDERER_MODE_EDITABLE', the <gtk-cell-renderer-text> allows to edit its text using an entry.

50.2 Usage

<gtk-cell-renderer-text> [Class]

This <gobject> class defines the following properties:

`text` Text to render

`markup` Marked up text to render

`attributes`
A list of style attributes to apply to the text of the renderer

`single-paragraph-mode`
Whether or not to keep all text in a single paragraph

`width-chars`
The desired width of the label, in characters

`wrap-width`
The width at which the text is wrapped

`alignment`
How to align the lines

`background`
Background color as a string

`foreground`
Foreground color as a string

`background-gdk`
Background color as a GdkColor

`foreground-gdk`
Foreground color as a GdkColor

`font` Font description as a string, e.g. "Sans Italic 12"

`font-desc`
Font description as a PangoFontDescription struct

`family` Name of the font family, e.g. Sans, Helvetica, Times, Monospace

`style` Font style

`variant` Font variant

`weight` Font weight

`stretch` Font stretch

`size` Font size

`size-points`
Font size in points

`scale` Font scaling factor

`editable` Whether the text can be modified by the user

`strikethrough`
Whether to strike through the text

`underline`
Style of underline for this text

`rise` Offset of text above the baseline (below the baseline if rise is negative)

`language` The language this text is in, as an ISO code. Pango can use this as a hint when rendering the text. If you don't understand this parameter, you probably don't need it

`ellipsize`
The preferred place to ellipsize the string, if the cell renderer does not have enough room to display the entire string

`wrap-mode`
How to break the string into multiple lines, if the cell renderer does not have enough room to display the entire string

`background-set`
Whether this tag affects the background color

`foreground-set`
Whether this tag affects the foreground color

`family-set`
Whether this tag affects the font family

`style-set`
Whether this tag affects the font style

`variant-set`
Whether this tag affects the font variant

`weight-set`
Whether this tag affects the font weight

`stretch-set`
Whether this tag affects the font stretch

`size-set` Whether this tag affects the font size

`scale-set`
: Whether this tag scales the font size by a factor

`editable-set`
: Whether this tag affects text editability

`strikethrough-set`
: Whether this tag affects strikethrough

`underline-set`
: Whether this tag affects underlining

`rise-set` Whether this tag affects the rise

`language-set`
: Whether this tag affects the language the text is rendered as

`ellipsize-set`
: Whether this tag affects the ellipsize mode

`align-set`
: Whether this tag affects the alignment mode

`edited` (*arg0* `<gchararray>`) [Signal on `<gtk-cell-renderer-text>`]
(*arg1* `<gchararray>`)

This signal is emitted after *renderer* has been edited.

`gtk-cell-renderer-text-new` ⇒ (*ret* `<gtk-cell-renderer>`) [Function]

Creates a new `<gtk-cell-renderer-text>`. Adjust how text is drawn using object properties. Object properties can be set globally (with `g-object-set`). Also, with `<gtk-tree-view-column>`, you can bind a property to a value in a `<gtk-tree-model>`. For example, you can bind the "text" property on the cell renderer to a string value in the model, thus rendering a different string in each row of the `<gtk-tree-view>`

ret the new cell renderer

51 GtkCellRendererToggle

Renders a toggle button in a cell

51.1 Overview

`<gtk-cell-renderer-toggle>` renders a toggle button in a cell. The button is drawn as a radio- or checkbutton, depending on the radio property. When activated, it emits the toggled signal.

51.2 Usage

`<gtk-cell-renderer-toggle>` [Class]
This `<gobject>` class defines the following properties:

`activatable`
The toggle button can be activated

`active` The toggle state of the button

`radio` Draw the toggle button as a radio button

`inconsistent`
The inconsistent state of the button

`indicator-size`
Size of check or radio indicator

`toggled` (*arg0* `<gchararray>`) [Signal on `<gtk-cell-renderer-toggle>`]
The ::toggled signal is emitted when the cell is toggled.

`gtk-cell-renderer-toggle-new` ⇒ (*ret* `<gtk-cell-renderer>`) [Function]
Creates a new `<gtk-cell-renderer-toggle>`. Adjust rendering parameters using object properties. Object properties can be set globally (with `g-object-set`). Also, with `<gtk-tree-view-column>`, you can bind a property to a value in a `<gtk-tree-model>`. For example, you can bind the `"active"` property on the cell renderer to a boolean value in the model, thus causing the check button to reflect the state of the model.

ret the new cell renderer

`gtk-cell-renderer-toggle-get-radio` [Function]
(*self* `<gtk-cell-renderer-toggle>`) ⇒ (*ret* `bool`)
`get-radio` [Method]
Returns whether we're rendering radio toggles rather than checkboxes.

toggle a `<gtk-cell-renderer-toggle>`

ret '`#t`' if we're rendering radio toggles rather than checkboxes

`gtk-cell-renderer-toggle-set-radio` [Function]
(*self* `<gtk-cell-renderer-toggle>`) (*radio* `bool`)

`set-radio` [Method]

If *radio* is '`#t`', the cell renderer renders a radio toggle (i.e. a toggle in a group of mutually-exclusive toggles). If '`#f`', it renders a check toggle (a standalone boolean option). This can be set globally for the cell renderer, or changed just before rendering each cell in the model (for `<gtk-tree-view>`, you set up a per-row setting using `<gtk-tree-view-column>` to associate model columns with cell renderer properties).

toggle a `<gtk-cell-renderer-toggle>`

radio '`#t`' to make the toggle look like a radio button

`gtk-cell-renderer-toggle-get-active` [Function]
(*self* `<gtk-cell-renderer-toggle>`) ⇒ (*ret* `bool`)

`get-active` [Method]

Returns whether the cell renderer is active. See `gtk-cell-renderer-toggle-set-active`.

toggle a `<gtk-cell-renderer-toggle>`

ret '`#t`' if the cell renderer is active.

`gtk-cell-renderer-toggle-set-active` [Function]
(*self* `<gtk-cell-renderer-toggle>`) (*setting* `bool`)

`set-active` [Method]

Activates or deactivates a cell renderer.

toggle a `<gtk-cell-renderer-toggle>`.

setting the value to set.

52 GtkListStore

A list-like data structure that can be used with the

52.1 Overview

The <gtk-list-store> object is a list model for use with a <gtk-tree-view> widget. It implements the <gtk-tree-model> interface, and consequentialy, can use all of the methods available there. It also implements the <gtk-tree-sortable> interface so it can be sorted by the view. Finally, it also implements the tree drag and drop interfaces.

The <gtk-list-store> can accept most GObject types as a column type, though it can't accept all custom types. Internally, it will keep a copy of data passed in (such as a string or a boxed pointer). Columns that accept <gobject>s are handled a little differently. The <gtk-list-store> will keep a reference to the object instead of copying the value. As a result, if the object is modified, it is up to the application writer to call *gtk-tree-model-row-changed* to emit the "row_changed" signal. This most commonly affects lists with <gdk-pixbuf>s stored.

```
enum {
  COLUMN_STRING,
  COLUMN_INT,
  COLUMN_BOOLEAN,
  N_COLUMNS
};

{
  GtkListStore *list_store;
  GtkTreePath *path;
  GtkTreeIter iter;
  gint i;

  list_store = gtk_list_store_new (N_COLUMNS,
                                   G_TYPE_STRING,
                                   G_TYPE_INT,
                                   G_TYPE_BOOLEAN);

  for (i = 0; i < 10; i++)
    {
      gchar *some_data;

      some_data = get_some_data (i);

      /* Add a new row to the model */
      gtk_list_store_append (list_store, &iter);
      gtk_list_store_set (list_store, &iter,
                          COLUMN_STRING, some_data,
                          COLUMN_INT, i,
```

```
                              COLUMN_BOOLEAN,  FALSE,
                              -1);

          /* As the store will keep a copy of the string internally, we
           * free some_data.
           */
          g_free (some_data);
        }

      /* Modify a particular row */
      path = gtk_tree_path_new_from_string ("4");
      gtk_tree_model_get_iter (GTK_TREE_MODEL (list_store),
                               &iter,
                               path);
      gtk_tree_path_free (path);
      gtk_list_store_set (list_store, &iter,
                          COLUMN_BOOLEAN, TRUE,
                          -1);
    }
```

52.2 Performance Considerations

Internally, the <gtk-list-store> was implemented with a linked list with a tail pointer prior to GTK+ 2.6. As a result, it was fast at data insertion and deletion, and not fast at random data access. The <gtk-list-store> sets the <gtk-tree-model-iters-persist> flag, which means that <gtk-tree-iter>s can be cached while the row exists. Thus, if access to a particular row is needed often and your code is expected to run on older versions of GTK+, it is worth keeping the iter around.

It is important to note that only the methods *gtk-list-store-insert-with-values* and *gtk-list-store-insert-with-valuesv* are atomic, in the sense that the row is being appended to the store and the values filled in in a single operation with regard to <gtk-tree-model> signaling. In contrast, using e.g. *gtk-list-store-append* and then *gtk-list-store-set* will first create a row, which triggers the "row_inserted" <gtk-tree-model> signal on <gtk-list-store>. The row, however, is still empty, and any signal handler connecting to "row_inserted" on this particular store should be prepared for the situation that the row might be empty. This is especially important if you are wrapping the <gtk-list-store> inside a <gtk-tree-model-filter> and are using a <gtk-tree-model-filter-visible-func>. Using any of the non-atomic operations to append rows to the <gtk-list-store> will cause the <gtk-tree-model-filter-visible-func> to be visited with an empty row first; the function must be prepared for that.

52.3 Usage

<gtk-list-store> [Class]

This <gobject> class defines no properties, other than those defined by its superclasses.

`gtk-list-store-new` (*types* `scm`) ⇒ (*ret* `<gtk-list-store>`) [Function]

Creates a new list store as with *n-columns* columns each of the types passed in. Note that only types derived from standard GObject fundamental types are supported.

As an example, '`gtk_tree_store_new (3, G_TYPE_INT, G_TYPE_STRING, GDK_TYPE_PIXBUF);`' will create a new `<gtk-list-store>` with three columns, of type int, string and `<gdk-pixbuf>` respectively.

n-columns number of columns in the list store

... all `<g-type>` types for the columns, from first to last

ret a new `<gtk-list-store>`

`gtk-list-store-newv` (*n_columns* `int`) (*types* `<g-type*>`) [Function]
⇒ (*ret* `<gtk-list-store>`)

Non-vararg creation function. Used primarily by language bindings.

n-columns number of columns in the list store

types an array of `<g-type>` types for the columns, from first to last

ret a new `<gtk-list-store>`

`gtk-list-store-set-value` (*self* `<gtk-list-store>`) [Function]
(*iter* `<gtk-tree-iter>`) (*column* `int`) (*value* `scm`)

`set-value` [Method]

Sets the data in the cell specified by *iter* and *column*. The type of *value* must be convertible to the type of the column.

list-store A `<gtk-list-store>`

iter A valid `<gtk-tree-iter>` for the row being modified

column column number to modify

value new value for the cell

`gtk-list-store-remove` (*self* `<gtk-list-store>`) [Function]
(*iter* `<gtk-tree-iter>`) ⇒ (*ret* `<gtk-tree-iter>`)

`remove` [Method]

Removes the given row from the list store. After being removed, *iter* is set to be the next valid row, or invalidated if it pointed to the last row in *list-store*.

list-store A `<gtk-list-store>`

iter A valid `<gtk-tree-iter>`

ret '`#t`' if *iter* is valid, '`#f`' if not.

`gtk-list-store-insert` (*self* `<gtk-list-store>`) (*position* `int`) [Function]
⇒ (*ret* `<gtk-tree-iter>`)

`insert` [Method]

Creates a new row at *position*. *iter* will be changed to point to this new row. If *position* is larger than the number of rows on the list, then the new row will be appended to the list. The row will be empty after this function is called. To fill in values, you need to call `gtk-list-store-set` or `gtk-list-store-set-value`.

list-store A `<gtk-list-store>`

iter An unset `<gtk-tree-iter>` to set to the new row

position position to insert the new row

`gtk-list-store-insert-before` (*self* `<gtk-list-store>`) [Function]
(*sibling* `<gtk-tree-iter>`) ⇒ (*ret* `<gtk-tree-iter>`)

`insert-before` [Method]

Inserts a new row before *sibling*. If *sibling* is '`#f`', then the row will be appended to the end of the list. *iter* will be changed to point to this new row. The row will be empty after this function is called. To fill in values, you need to call `gtk-list-store-set` or `gtk-list-store-set-value`.

list-store A `<gtk-list-store>`

iter An unset `<gtk-tree-iter>` to set to the new row

sibling A valid `<gtk-tree-iter>`, or '`#f`'

`gtk-list-store-insert-after` (*self* `<gtk-list-store>`) [Function]
(*sibling* `<gtk-tree-iter>`) ⇒ (*ret* `<gtk-tree-iter>`)

`insert-after` [Method]

Inserts a new row after *sibling*. If *sibling* is '`#f`', then the row will be prepended to the beginning of the list. *iter* will be changed to point to this new row. The row will be empty after this function is called. To fill in values, you need to call `gtk-list-store-set` or `gtk-list-store-set-value`.

list-store A `<gtk-list-store>`

iter An unset `<gtk-tree-iter>` to set to the new row

sibling A valid `<gtk-tree-iter>`, or '`#f`'

`gtk-list-store-prepend` (*self* `<gtk-list-store>`) [Function]
⇒ (*ret* `<gtk-tree-iter>`)

`prepend` [Method]

Prepends a new row to *list-store*. *iter* will be changed to point to this new row. The row will be empty after this function is called. To fill in values, you need to call `gtk-list-store-set` or `gtk-list-store-set-value`.

list-store A `<gtk-list-store>`

iter An unset `<gtk-tree-iter>` to set to the prepend row

`gtk-list-store-append` (*self* `<gtk-list-store>`) [Function]
⇒ (*ret* `<gtk-tree-iter>`)

`append` [Method]

Appends a new row to *list-store*. *iter* will be changed to point to this new row. The row will be empty after this function is called. To fill in values, you need to call `gtk-list-store-set` or `gtk-list-store-set-value`.

list-store A `<gtk-list-store>`

iter An unset `<gtk-tree-iter>` to set to the appended row

`gtk-list-store-clear` (*self* `<gtk-list-store>`) [Function]
`clear` [Method]

Removes all rows from the list store.

list-store a `<gtk-list-store>`.

`gtk-list-store-iter-is-valid` (*self* `<gtk-list-store>`) [Function]
(*iter* `<gtk-tree-iter>`) ⇒ (*ret* `bool`)
`iter-is-valid` [Method]

Checks if the given iter is a valid iter for this `<gtk-list-store>`.

list-store A `<gtk-list-store>`.

iter A `<gtk-tree-iter>`.

ret '`#t`' if the iter is valid, '`#f`' if the iter is invalid.

Since 2.2

`gtk-list-store-reorder` (*self* `<gtk-list-store>`) [Function]
⇒ (*new_order* `int`)
`reorder` [Method]

Reorders *store* to follow the order indicated by *new-order*. Note that this function only works with unsorted stores.

store A `<gtk-list-store>`.

new-order an array of integers mapping the new position of each child to its old position before the re-ordering, i.e. *new-order*'`[newpos] = oldpos`'.

Since 2.2

`gtk-list-store-swap` (*self* `<gtk-list-store>`) [Function]
(*a* `<gtk-tree-iter>`) (*b* `<gtk-tree-iter>`)
`swap` [Method]

Swaps *a* and *b* in *store*. Note that this function only works with unsorted stores.

store A `<gtk-list-store>`.

a A `<gtk-tree-iter>`.

b Another `<gtk-tree-iter>`.

Since 2.2

`gtk-list-store-move-before` (*self* `<gtk-list-store>`) [Function]
(*iter* `<gtk-tree-iter>`) (*position* `<gtk-tree-iter>`)
`move-before` [Method]

Moves *iter* in *store* to the position before *position*. Note that this function only works with unsorted stores. If *position* is '`#f`', *iter* will be moved to the end of the list.

store A `<gtk-list-store>`.

iter A `<gtk-tree-iter>`.

position A `<gtk-tree-iter>`, or '`#f`'.

Since 2.2

`gtk-list-store-move-after` (*self* `<gtk-list-store>`) [Function]
(*iter* `<gtk-tree-iter>`) (*position* `<gtk-tree-iter>`)
`move-after` [Method]

Moves *iter* in *store* to the position after *position*. Note that this function only works with unsorted stores. If *position* is '`#f`', *iter* will be moved to the start of the list.

store A `<gtk-list-store>`.

iter A `<gtk-tree-iter>`.

position A `<gtk-tree-iter>` or '`#f`'.

Since 2.2

53 GtkTreeStore

A tree-like data structure that can be used with the

53.1 Overview

The <gtk-tree-store> object is a list model for use with a <gtk-tree-view> widget. It implements the <gtk-tree-model> interface, and consequentialy, can use all of the methods available there. It also implements the <gtk-tree-sortable> interface so it can be sorted by the view. Finally, it also implements the tree drag and drop interfaces.

53.2 Usage

<gtk-tree-store> [Class]

This <gobject> class defines no properties, other than those defined by its superclasses.

gtk-tree-store-new (*types* scm) ⇒ (*ret* <gtk-tree-store>) [Function]

Creates a new tree store as with *n-columns* columns each of the types passed in. Note that only types derived from standard GObject fundamental types are supported.

As an example, 'gtk_tree_store_new (3, G_TYPE_INT, G_TYPE_STRING, GDK_TYPE_PIXBUF);' will create a new <gtk-tree-store> with three columns, of type <int>, <string> and <gdk-pixbuf> respectively.

n-columns number of columns in the tree store

... all <g-type> types for the columns, from first to last

ret a new <gtk-tree-store>

gtk-tree-store-newv (*n_columns* int) (*types* <g-type*>) [Function]
⇒ (*ret* <gtk-tree-store>)

Non vararg creation function. Used primarily by language bindings.

n-columns number of columns in the tree store

types an array of <g-type> types for the columns, from first to last

ret a new <gtk-tree-store>

gtk-tree-store-set-value (*self* <gtk-tree-store>) [Function]
(*iter* <gtk-tree-iter>) (*column* int) (*value* scm)

set-value [Method]

Sets the data in the cell specified by *iter* and *column*. The type of *value* must be convertible to the type of the column.

tree-store a <gtk-tree-store>

iter A valid <gtk-tree-iter> for the row being modified

column column number to modify

value new value for the cell

`gtk-tree-store-remove` (*self* `<gtk-tree-store>`) [Function]
(*iter* `<gtk-tree-iter>`) ⇒ (*ret* `<gtk-tree-iter>`)

`remove` [Method]

Removes *iter* from *tree-store*. After being removed, *iter* is set to the next valid row at that level, or invalidated if it previously pointed to the last one.

tree-store A `<gtk-tree-store>`

iter A valid `<gtk-tree-iter>`

ret '`#t`' if *iter* is still valid, '`#f`' if not.

`gtk-tree-store-insert` (*self* `<gtk-tree-store>`) [Function]
(*parent* `<gtk-tree-iter>`) (*position* `int`) ⇒ (*ret* `<gtk-tree-iter>`)

`insert` [Method]

Creates a new row at *position*. If parent is non-'`#f`', then the row will be made a child of *parent*. Otherwise, the row will be created at the toplevel. If *position* is larger than the number of rows at that level, then the new row will be inserted to the end of the list. *iter* will be changed to point to this new row. The row will be empty after this function is called. To fill in values, you need to call `gtk-tree-store-set` or `gtk-tree-store-set-value`.

tree-store A `<gtk-tree-store>`

iter An unset `<gtk-tree-iter>` to set to the new row

parent A valid `<gtk-tree-iter>`, or '`#f`'

position position to insert the new row

`gtk-tree-store-insert-before` (*self* `<gtk-tree-store>`) [Function]
(*parent* `<gtk-tree-iter>`) (*sibling* `<gtk-tree-iter>`)
⇒ (*ret* `<gtk-tree-iter>`)

`insert-before` [Method]

Inserts a new row before *sibling*. If *sibling* is '`#f`', then the row will be appended to *parent* 's children. If *parent* and *sibling* are '`#f`', then the row will be appended to the toplevel. If both *sibling* and *parent* are set, then *parent* must be the parent of *sibling*. When *sibling* is set, *parent* is optional.

iter will be changed to point to this new row. The row will be empty after this function is called. To fill in values, you need to call `gtk-tree-store-set` or `gtk-tree-store-set-value`.

tree-store A `<gtk-tree-store>`

iter An unset `<gtk-tree-iter>` to set to the new row

parent A valid `<gtk-tree-iter>`, or '`#f`'

sibling A valid `<gtk-tree-iter>`, or '`#f`'

`gtk-tree-store-insert-after` (*self* `<gtk-tree-store>`) [Function]
(*parent* `<gtk-tree-iter>`) (*sibling* `<gtk-tree-iter>`)
⇒ (*ret* `<gtk-tree-iter>`)

`insert-after` [Method]

Inserts a new row after *sibling*. If *sibling* is '`#f`', then the row will be prepended to *parent* 's children. If *parent* and *sibling* are '`#f`', then the row will be prepended to the toplevel. If both *sibling* and *parent* are set, then *parent* must be the parent of *sibling*. When *sibling* is set, *parent* is optional.

iter will be changed to point to this new row. The row will be empty after this function is called. To fill in values, you need to call `gtk-tree-store-set` or `gtk-tree-store-set-value`.

tree-store A `<gtk-tree-store>`

iter An unset `<gtk-tree-iter>` to set to the new row

parent A valid `<gtk-tree-iter>`, or '`#f`'

sibling A valid `<gtk-tree-iter>`, or '`#f`'

`gtk-tree-store-prepend` (*self* `<gtk-tree-store>`) [Function]
(*parent* `<gtk-tree-iter>`) ⇒ (*ret* `<gtk-tree-iter>`)

`prepend` [Method]

Prepends a new row to *tree-store*. If *parent* is non-'`#f`', then it will prepend the new row before the first child of *parent*, otherwise it will prepend a row to the top level. *iter* will be changed to point to this new row. The row will be empty after this function is called. To fill in values, you need to call `gtk-tree-store-set` or `gtk-tree-store-set-value`.

tree-store A `<gtk-tree-store>`

iter An unset `<gtk-tree-iter>` to set to the prepended row

parent A valid `<gtk-tree-iter>`, or '`#f`'

`gtk-tree-store-append` (*self* `<gtk-tree-store>`) [Function]
(*parent* `<gtk-tree-iter>`) ⇒ (*ret* `<gtk-tree-iter>`)

`append` [Method]

Appends a new row to *tree-store*. If *parent* is non-'`#f`', then it will append the new row after the last child of *parent*, otherwise it will append a row to the top level. *iter* will be changed to point to this new row. The row will be empty after this function is called. To fill in values, you need to call `gtk-tree-store-set` or `gtk-tree-store-set-value`.

tree-store A `<gtk-tree-store>`

iter An unset `<gtk-tree-iter>` to set to the appended row

parent A valid `<gtk-tree-iter>`, or '`#f`'

`gtk-tree-store-is-ancestor` (*self* `<gtk-tree-store>`) [Function]
(*iter* `<gtk-tree-iter>`) (*descendant* `<gtk-tree-iter>`) ⇒ (*ret* `bool`)

`is-ancestor` [Method]

Returns '`#t`' if *iter* is an ancestor of *descendant*. That is, *iter* is the parent (or grandparent or great-grandparent) of *descendant*.

tree-store A `<gtk-tree-store>`

iter A valid `<gtk-tree-iter>`

descendant
A valid `<gtk-tree-iter>`

ret '`#t`', if *iter* is an ancestor of *descendant*

`gtk-tree-store-iter-depth` (*self* `<gtk-tree-store>`) [Function]
(*iter* `<gtk-tree-iter>`) ⇒ (*ret* `int`)
`iter-depth` [Method]
Returns the depth of *iter*. This will be 0 for anything on the root level, 1 for anything down a level, etc.

tree-store A `<gtk-tree-store>`

iter A valid `<gtk-tree-iter>`

ret The depth of *iter*

`gtk-tree-store-clear` (*self* `<gtk-tree-store>`) [Function]
`clear` [Method]
Removes all rows from *tree-store*

tree-store a `<gtk-tree-store>`

`gtk-tree-store-iter-is-valid` (*self* `<gtk-tree-store>`) [Function]
(*iter* `<gtk-tree-iter>`) ⇒ (*ret* `bool`)
`iter-is-valid` [Method]
WARNING: This function is slow. Only use it for debugging and/or testing purposes.

Checks if the given iter is a valid iter for this `<gtk-tree-store>`.

tree-store A `<gtk-tree-store>`.

iter A `<gtk-tree-iter>`.

ret '`#t`' if the iter is valid, '`#f`' if the iter is invalid.

Since 2.2

`gtk-tree-store-reorder` (*self* `<gtk-tree-store>`) [Function]
(*parent* `<gtk-tree-iter>`) ⇒ (*new_order* `int`)
`reorder` [Method]
Reorders the children of *parent* in *tree-store* to follow the order indicated by *new-order*. Note that this function only works with unsorted stores.

tree-store A `<gtk-tree-store>`.

parent A `<gtk-tree-iter>`.

new-order an array of integers mapping the new position of each child to its old position before the re-ordering, i.e. *new-order*'`[newpos] = oldpos`'.

Since 2.2

`gtk-tree-store-swap` (*self* `<gtk-tree-store>`) [Function]
(*a* `<gtk-tree-iter>`) (*b* `<gtk-tree-iter>`)

`swap` [Method]

Swaps *a* and *b* in the same level of *tree-store*. Note that this function only works with unsorted stores.

tree-store A `<gtk-tree-store>`.

a A `<gtk-tree-iter>`.

b Another `<gtk-tree-iter>`.

Since 2.2

`gtk-tree-store-move-before` (*self* `<gtk-tree-store>`) [Function]
(*iter* `<gtk-tree-iter>`) (*position* `<gtk-tree-iter>`)

`move-before` [Method]

Moves *iter* in *tree-store* to the position before *position*. *iter* and *position* should be in the same level. Note that this function only works with unsorted stores. If *position* is '`#f`', *iter* will be moved to the end of the level.

tree-store A `<gtk-tree-store>`.

iter A `<gtk-tree-iter>`.

position A `<gtk-tree-iter>` or '`#f`'.

Since 2.2

`gtk-tree-store-move-after` (*self* `<gtk-tree-store>`) [Function]
(*iter* `<gtk-tree-iter>`) (*position* `<gtk-tree-iter>`)

`move-after` [Method]

Moves *iter* in *tree-store* to the position after *position*. *iter* and *position* should be in the same level. Note that this function only works with unsorted stores. If *position* is '`#f`', *iter* will be moved to the start of the level.

tree-store A `<gtk-tree-store>`.

iter A `<gtk-tree-iter>`.

position A `<gtk-tree-iter>`.

Since 2.2

54 GtkComboBox

A widget used to choose from a list of items

54.1 Overview

A <gtk-combo-box> is a widget that allows the user to choose from a list of valid choices. The <gtk-combo-box> displays the selected choice. When activated, the <gtk-combo-box> displays a popup which allows the user to make a new choice. The style in which the selected value is displayed, and the style of the popup is determined by the current theme. It may be similar to a <gtk-option-menu>, or similar to a Windows-style combo box.

Unlike its predecessors <gtk-combo> and <gtk-option-menu>, the <gtk-combo-box> uses the model-view pattern; the list of valid choices is specified in the form of a tree model, and the display of the choices can be adapted to the data in the model by using cell renderers, as you would in a tree view. This is possible since <gtk-combo-box> implements the <gtk-cell-layout> interface. The tree model holding the valid choices is not restricted to a flat list, it can be a real tree, and the popup will reflect the tree structure.

In addition to the model-view API, <gtk-combo-box> offers a simple API which is suitable for text-only combo boxes, and hides the complexity of managing the data in a model. It consists of the functions gtk-combo-box-new-text, gtk-combo-box-append-text, gtk-combo-box-insert-text, gtk-combo-box-prepend-text, gtk-combo-box-remove-text and gtk-combo-box-get-active-text.

54.2 Usage

<gtk-combo-box> [Class]

This <gobject> class defines the following properties:

model The model for the combo box

wrap-width
Wrap width for laying out the items in a grid

row-span-column
TreeModel column containing the row span values

column-span-column
TreeModel column containing the column span values

active The item which is currently active

add-tearoffs
Whether dropdowns should have a tearoff menu item

tearoff-title
A title that may be displayed by the window manager when the popup is torn-off

has-frame
Whether the combo box draws a frame around the child

`focus-on-click`
: Whether the combo box grabs focus when it is clicked with the mouse

`popup-shown`
: Whether the combo's dropdown is shown

`changed` [Signal on `<gtk-combo-box>`]

The changed signal is emitted when the active item is changed. The can be due to the user selecting a different item from the list, or due to a call to `gtk-combo-box-set-active-iter`. It will also be emitted while typing into a GtkComboBoxEntry, as well as when selecting an item from the GtkComboBoxEntry's list.

Since 2.4

`move-active` (*arg0* `<gtk-scroll-type>`) [Signal on `<gtk-combo-box>`]

undocumented

`popup` [Signal on `<gtk-combo-box>`]

undocumented

`popdown` ⇒ `<gboolean>` [Signal on `<gtk-combo-box>`]

undocumented

`gtk-combo-box-new` ⇒ (*ret* `<gtk-widget>`) [Function]

Creates a new empty `<gtk-combo-box>`.

ret A new `<gtk-combo-box>`.

Since 2.4

`gtk-combo-box-new-with-model` (*model* `<gtk-tree-model>`) [Function]
⇒ (*ret* `<gtk-widget>`)

Creates a new `<gtk-combo-box>` with the model initialized to *model*.

model A `<gtk-tree-model>`.

ret A new `<gtk-combo-box>`.

Since 2.4

`gtk-combo-box-get-wrap-width` (*self* `<gtk-combo-box>`) [Function]
⇒ (*ret* `int`)

`get-wrap-width` [Method]

Returns the wrap width which is used to determine the number of columns for the popup menu. If the wrap width is larger than 1, the combo box is in table mode.

combo-box
: A `<gtk-combo-box>`.

ret the wrap width.

Since 2.6

`gtk-combo-box-set-wrap-width` (*self* `<gtk-combo-box>`) (*width* `int`) [Function]
`set-wrap-width` [Method]

Sets the wrap width of *combo-box* to be *width*. The wrap width is basically the preferred number of columns when you want the popup to be layed out in a table.

combo-box
A `<gtk-combo-box>`.

width Preferred number of columns.

Since 2.4

`gtk-combo-box-get-row-span-column` (*self* `<gtk-combo-box>`) [Function]
⇒ (*ret* `int`)
`get-row-span-column` [Method]

Returns the column with row span information for *combo-box*.

combo-box
A `<gtk-combo-box>`.

ret the row span column.

Since 2.6

`gtk-combo-box-set-row-span-column` (*self* `<gtk-combo-box>`) [Function]
(*row_span* `int`)
`set-row-span-column` [Method]

Sets the column with row span information for *combo-box* to be *row-span*. The row span column contains integers which indicate how many rows an item should span.

combo-box
A `<gtk-combo-box>`.

row-span A column in the model passed during construction.

Since 2.4

`gtk-combo-box-get-active` (*self* `<gtk-combo-box>`) ⇒ (*ret* `int`) [Function]
`get-active` [Method]

Returns the index of the currently active item, or -1 if there's no active item. If the model is a non-flat treemodel, and the active item is not an immediate child of the root of the tree, this function returns '`gtk_tree_path_get_indices (path)[0]`', where '`path`' is the `<gtk-tree-path>` of the active item.

combo-box
A `<gtk-combo-box>`.

ret An integer which is the index of the currently active item, or -1 if there's no active item.

Since 2.4

`gtk-combo-box-set-active` (*self* `<gtk-combo-box>`) (*index_* `int`) [Function]
`set-active` [Method]

Sets the active item of *combo-box* to be the item at *index*.

combo-box
A `<gtk-combo-box>`.

index An index in the model passed during construction, or -1 to have no active item.

Since 2.4

`gtk-combo-box-get-active-iter` (*self* `<gtk-combo-box>`) [Function]
(*iter* `<gtk-tree-iter>`) ⇒ (*ret* `bool`)

`get-active-iter` [Method]

Sets *iter* to point to the current active item, if it exists.

combo-box
A `<gtk-combo-box>`

iter The uninitialized `<gtk-tree-iter>`.

ret '`#t`', if *iter* was set

Since 2.4

`gtk-combo-box-set-active-iter` (*self* `<gtk-combo-box>`) [Function]
(*iter* `<gtk-tree-iter>`)

`set-active-iter` [Method]

Sets the current active item to be the one referenced by *iter*. *iter* must correspond to a path of depth one.

combo-box
A `<gtk-combo-box>`

iter The `<gtk-tree-iter>`.

Since 2.4

`gtk-combo-box-get-model` (*self* `<gtk-combo-box>`) [Function]
⇒ (*ret* `<gtk-tree-model>`)

`get-model` [Method]

Returns the `<gtk-tree-model>` which is acting as data source for *combo-box*.

combo-box
A `<gtk-combo-box>`.

ret A `<gtk-tree-model>` which was passed during construction.

Since 2.4

`gtk-combo-box-set-model` (*self* `<gtk-combo-box>`) [Function]
(*model* `<gtk-tree-model>`)

`set-model` [Method]

Sets the model used by *combo-box* to be *model*. Will unset a previously set model (if applicable). If model is '`#f`', then it will unset the model.

Note that this function does not clear the cell renderers, you have to call `gtk-combo-box-cell-layout-clear` yourself if you need to set up different cell renderers for the new model.

combo-box
A `<gtk-combo-box>`.

model A `<gtk-tree-model>`.

Since 2.4

`gtk-combo-box-new-text` ⇒ (*ret* `<gtk-widget>`) [Function]
Convenience function which constructs a new text combo box, which is a `<gtk-combo-box>` just displaying strings. If you use this function to create a text combo box, you should only manipulate its data source with the following convenience functions: `gtk-combo-box-append-text`, `gtk-combo-box-insert-text`, `gtk-combo-box-prepend-text` and `gtk-combo-box-remove-text`.

ret A new text combo box.

Since 2.4

`gtk-combo-box-append-text` (*self* `<gtk-combo-box>`) (*text* `mchars`) [Function]
`append-text` [Method]
Appends *string* to the list of strings stored in *combo-box*. Note that you can only use this function with combo boxes constructed with `gtk-combo-box-new-text`.

combo-box
A `<gtk-combo-box>` constructed using `gtk-combo-box-new-text`.

text A string.

Since 2.4

`gtk-combo-box-insert-text` (*self* `<gtk-combo-box>`) (*position* `int`) [Function]
(*text* `mchars`)
`insert-text` [Method]
Inserts *string* at *position* in the list of strings stored in *combo-box*. Note that you can only use this function with combo boxes constructed with `gtk-combo-box-new-text`.

combo-box
A `<gtk-combo-box>` constructed using `gtk-combo-box-new-text`.

position An index to insert *text*.

text A string.

Since 2.4

`gtk-combo-box-prepend-text` (*self* `<gtk-combo-box>`) (*text* `mchars`) [Function]
`prepend-text` [Method]
Prepends *string* to the list of strings stored in *combo-box*. Note that you can only use this function with combo boxes constructed with `gtk-combo-box-new-text`.

combo-box
A `<gtk-combo-box>` constructed with `gtk-combo-box-new-text`.

text A string.

Since 2.4

`gtk-combo-box-remove-text` (*self* `<gtk-combo-box>`) (*position* `int`) [Function]
`remove-text` [Method]

Removes the string at *position* from *combo-box*. Note that you can only use this function with combo boxes constructed with `gtk-combo-box-new-text`.

combo-box
A `<gtk-combo-box>` constructed with `gtk-combo-box-new-text`.

position Index of the item to remove.

Since 2.4

`gtk-combo-box-get-active-text` (*self* `<gtk-combo-box>`) [Function]
⇒ (*ret* `mchars`)
`get-active-text` [Method]

Returns the currently active string in *combo-box* or '`#f`' if none is selected. Note that you can only use this function with combo boxes constructed with `gtk-combo-box-new-text` and with `<gtk-combo-box-entry>`s.

combo-box
A `<gtk-combo-box>` constructed with `gtk-combo-box-new-text`.

ret a newly allocated string containing the currently active text.

Since 2.6

`gtk-combo-box-popup` (*self* `<gtk-combo-box>`) [Function]
`popup` [Method]

Pops up the menu or dropdown list of *combo-box*.

This function is mostly intended for use by accessibility technologies; applications should have little use for it.

combo-box
a `<gtk-combo-box>`

Since 2.4

`gtk-combo-box-popdown` (*self* `<gtk-combo-box>`) [Function]
`popdown` [Method]

Hides the menu or dropdown list of *combo-box*.

This function is mostly intended for use by accessibility technologies; applications should have little use for it.

combo-box
a `<gtk-combo-box>`

Since 2.4

`gtk-combo-box-get-popup-accessible` (*self* `<gtk-combo-box>`) [Function]
⇒ (*ret* `<atk-object>`)
`get-popup-accessible` [Method]

Gets the accessible object corresponding to the combo box's popup.

This function is mostly intended for use by accessibility technologies; applications should have little use for it.

combo-box
a `<gtk-combo-box>`

ret the accessible object corresponding to the combo box's popup.

Since 2.6

`gtk-combo-box-set-add-tearoffs` (*self* `<gtk-combo-box>`) (*add_tearoffs* `bool`) [Function]
`set-add-tearoffs` [Method]

Sets whether the popup menu should have a tearoff menu item.

combo-box
a `<gtk-combo-box>`

add-tearoffs
'`#t`' to add tearoff menu items

Since 2.6

`gtk-combo-box-get-add-tearoffs` (*self* `<gtk-combo-box>`) ⇒ (*ret* `bool`) [Function]
`get-add-tearoffs` [Method]

Gets the current value of the :add-tearoffs property.

combo-box
a `<gtk-combo-box>`

ret the current value of the :add-tearoffs property.

`gtk-combo-box-set-title` (*self* `<gtk-combo-box>`) (*title* `mchars`) [Function]
`set-title` [Method]

Sets the menu's title in tearoff mode.

combo-box
a `<gtk-combo-box>`

title a title for the menu in tearoff mode.

Since 2.10

`gtk-combo-box-get-title` (*self* `<gtk-combo-box>`) ⇒ (*ret* `mchars`) [Function]
`get-title` [Method]

Gets the current title of the menu in tearoff mode. See `gtk-combo-box-set-add-tearoffs`.

combo-box
a `<gtk-combo-box>`

ret the menu's title in tearoff mode. This is an internal copy of the string which must not be freed.

Since 2.10

`gtk-combo-box-set-focus-on-click` (*self* `<gtk-combo-box>`) [Function]
(*focus_on_click* `bool`)

`set-focus-on-click` [Method]

Sets whether the combo box will grab focus when it is clicked with the mouse. Making mouse clicks not grab focus is useful in places like toolbars where you don't want the keyboard focus removed from the main area of the application.

combo a `<gtk-combo-box>`

focus-on-click
whether the combo box grabs focus when clicked with the mouse

Since 2.6

`gtk-combo-box-get-focus-on-click` (*self* `<gtk-combo-box>`) [Function]
⇒ (*ret* `bool`)

`get-focus-on-click` [Method]

Returns whether the combo box grabs focus when it is clicked with the mouse. See `gtk-combo-box-set-focus-on-click`.

combo a `<gtk-combo-box>`

ret '`#t`' if the combo box grabs focus when it is clicked with the mouse.

Since 2.6

55 GtkComboBoxEntry

A text entry field with a dropdown list

55.1 Overview

A <gtk-combo-box-entry> is a widget that allows the user to choose from a list of valid choices or enter a different value. It is very similar to a <gtk-combo-box>, but it displays the selected value in an entry to allow modifying it.

In contrast to a <gtk-combo-box>, the underlying model of a <gtk-combo-box-entry> must always have a text column (see gtk-combo-box-entry-set-text-column), and the entry will show the content of the text column in the selected row. To get the text from the entry, use gtk-combo-box-get-active-text.

The changed signal will be emitted while typing into a GtkComboBoxEntry, as well as when selecting an item from the GtkComboBoxEntry's list. Use gtk-combo-box-get-active or gtk-combo-box-get-active-iter to discover whether an item was actually selected from the list.

Connect to the activate signal of the GtkEntry (use gtk-bin-get-child) to detect when the user actually finishes entering text.

The convenience API to construct simple text-only <gtk-combo-box>es can also be used with <gtk-combo-box-entry>s which have been constructed with gtk-combo-box-entry-new-text.

55.2 Usage

<gtk-combo-box-entry> [Class]

This <gobject> class defines the following properties:

text-column

A column in the data source model to get the strings from

gtk-combo-box-entry-new ⇒ (*ret* <gtk-widget>) [Function]

Creates a new <gtk-combo-box-entry> which has a <gtk-entry> as child. After construction, you should set a model using gtk-combo-box-set-model and a text_column * using gtk-combo-box-entry-set-text-column.

ret A new <gtk-combo-box-entry>.

Since 2.4

gtk-combo-box-entry-new-with-model (*model* <gtk-tree-model>) [Function]
(*text_column* int) ⇒ (*ret* <gtk-widget>)

Creates a new <gtk-combo-box-entry> which has a <gtk-entry> as child and a list of strings as popup. You can get the <gtk-entry> from a <gtk-combo-box-entry> using GTK_ENTRY (GTK_BIN (combo_box_entry)->child). To add and remove strings from the list, just modify *model* using its data manipulation API.

model A <gtk-tree-model>.

text-column
A column in *model* to get the strings from.

ret A new `<gtk-combo-box-entry>`.

Since 2.4

`gtk-combo-box-entry-new-text` ⇒ (*ret* `<gtk-widget>`) [Function]
Convenience function which constructs a new editable text combo box, which is a `<gtk-combo-box-entry>` just displaying strings. If you use this function to create a text combo box, you should only manipulate its data source with the following convenience functions: `gtk-combo-box-append-text`, `gtk-combo-box-insert-text`, `gtk-combo-box-prepend-text` and `gtk-combo-box-remove-text`.

ret A new text `<gtk-combo-box-entry>`.

Since 2.4

`gtk-combo-box-entry-set-text-column` [Function]
(*self* `<gtk-combo-box-entry>`) (*text_column* `int`)
`set-text-column` [Method]
Sets the model column which *entry-box* should use to get strings from to be *text-column*.

entry-box A `<gtk-combo-box-entry>`.

text-column
A column in *model* to get the strings from.

Since 2.4

`gtk-combo-box-entry-get-text-column` [Function]
(*self* `<gtk-combo-box-entry>`) ⇒ (*ret* `int`)
`get-text-column` [Method]
Returns the column which *entry-box* is using to get the strings from.

entry-box A `<gtk-combo-box-entry>`.

ret A column in the data source model of *entry-box*.

Since 2.4

56 GtkMenu

A menu widget

56.1 Overview

A <gtk-menu> is a <gtk-menu-shell> that implements a drop down menu consisting of a list of <gtk-menu-item> objects which can be navigated and activated by the user to perform application functions.

A <gtk-menu> is most commonly dropped down by activating a <gtk-menu-item> in a <gtk-menu-bar> or popped up by activating a <gtk-menu-item> in another <gtk-menu>.

A <gtk-menu> can also be popped up by activating a <gtk-option-menu>. Other composite widgets such as the <gtk-notebook> can pop up a <gtk-menu> as well.

Applications can display a <gtk-menu> as a popup menu by calling the gtk-menu-popup function. The example below shows how an application can pop up a menu when the 3rd mouse button is pressed.

```
    /* connect our handler which will popup the menu */
    g_signal_connect_swapped (window, "button_press_event",
G_CALLBACK (my_popup_handler), menu);

static gint
my_popup_handler (GtkWidget *widget, GdkEvent *event)
{
  GtkMenu *menu;
  GdkEventButton *event_button;

  g_return_val_if_fail (widget != NULL, FALSE);
  g_return_val_if_fail (GTK_IS_MENU (widget), FALSE);
  g_return_val_if_fail (event != NULL, FALSE);

  /* The "widget" is the menu that was supplied when
   * g_signal_connect_swapped() was called.
   */
  menu = GTK_MENU (widget);

  if (event->type == GDK_BUTTON_PRESS)
    {
      event_button = (GdkEventButton *) event;
      if (event_button->button == 3)
{
  gtk_menu_popup (menu, NULL, NULL, NULL, NULL,
  event_button->button, event_button->time);
  return TRUE;
}
    }
```

```
  return FALSE;
}
```

56.2 Usage

`<gtk-menu>` [Class]

This `<gobject>` class defines the following properties:

`tearoff-state`
: A boolean that indicates whether the menu is torn-off

`tearoff-title`
: A title that may be displayed by the window manager when this menu is torn-off

`move-scroll` (*arg0* `<gtk-scroll-type>`) [Signal on `<gtk-menu>`]

`gtk-menu-new` ⇒ (*ret* `<gtk-widget>`) [Function]

Creates a new `<gtk-menu>`.

ret a new `<gtk-menu>`.

`gtk-menu-set-screen` (*self* `<gtk-menu>`) (*screen* `<gdk-screen>`) [Function]
`set-screen` [Method]

Sets the `<gdk-screen>` on which the menu will be displayed.

menu a `<gtk-menu>`.

screen a `<gdk-screen>`, or '`#f`' if the screen should be determined by the widget the menu is attached to.

Since 2.2

`gtk-menu-reorder-child` (*self* `<gtk-menu>`) (*child* `<gtk-widget>`) [Function]
(*position* `int`)
`reorder-child` [Method]

Moves a `<gtk-menu-item>` to a new position within the `<gtk-menu>`.

menu a `<gtk-menu>`.

child the `<gtk-menu-item>` to move.

position the new position to place *child*. Positions are numbered from 0 to n-1.

`gtk-menu-attach` (*self* `<gtk-menu>`) (*child* `<gtk-widget>`) [Function]
(*left_attach* `unsigned-int`) (*right_attach* `unsigned-int`)
(*top_attach* `unsigned-int`) (*bottom_attach* `unsigned-int`)
`attach` [Method]

Adds a new `<gtk-menu-item>` to a (table) menu. The number of 'cells' that an item will occupy is specified by *left-attach*, *right-attach*, *top-attach* and *bottom-attach*. These each represent the leftmost, rightmost, uppermost and lower column and row numbers of the table. (Columns and rows are indexed from zero).

Note that this function is not related to `gtk-menu-detach`.

menu a `<gtk-menu>`.

child a `<gtk-menu-item>`.

left-attach The column number to attach the left side of the item to.

right-attach
The column number to attach the right side of the item to.

top-attach The row number to attach the top of the item to.

bottom-attach
The row number to attach the bottom of the item to.

Since 2.4

`gtk-menu-popup` (*self* `<gtk-menu>`) (*parent_menu_shell* `<gtk-widget>`) [Function]
(*parent_menu_item* `<gtk-widget>`) (*func* `<gtk-menu-position-func>`)
(*data* `<gpointer>`) (*button* `unsigned-int`)
(*activate_time* `unsigned-int32`)

`popup` [Method]

Displays a menu and makes it available for selection. Applications can use this function to display context-sensitive menus, and will typically supply '`#f`' for the *parent-menu-shell*, *parent-menu-item*, *func* and *data* parameters. The default menu positioning function will position the menu at the current mouse cursor position.

The *button* parameter should be the mouse button pressed to initiate the menu popup. If the menu popup was initiated by something other than a mouse button press, such as a mouse button release or a keypress, *button* should be 0.

The *activate-time* parameter should be the time stamp of the event that initiated the popup. If such an event is not available, use `gtk-get-current-event-time` instead.

menu a `<gtk-menu>`.

parent-menu-shell
the menu shell containing the triggering menu item, or '`#f`'

parent-menu-item
the menu item whose activation triggered the popup, or '`#f`'

func a user supplied function used to position the menu, or '`#f`'

data user supplied data to be passed to *func*.

button the mouse button which was pressed to initiate the event.

activate-time
the time at which the activation event occurred.

`gtk-menu-set-accel-group` (*self* `<gtk-menu>`) [Function]
(*accel_group* `<gtk-accel-group>`)

`set-accel-group` [Method]

Set the `<gtk-accel-group>` which holds global accelerators for the menu. This accelerator group needs to also be added to all windows that this menu is being used in with `gtk-window-add-accel-group`, in order for those windows to support all the accelerators contained in this group.

menu a `<gtk-menu>`.

accel-group
the `<gtk-accel-group>` to be associated with the menu.

`gtk-menu-get-accel-group` (*self* `<gtk-menu>`) [Function]
⇒ (*ret* `<gtk-accel-group>`)
`get-accel-group` [Method]
Gets the `<gtk-accel-group>` which holds global accelerators for the menu. See `gtk-menu-set-accel-group`.

menu a `<gtk-menu>`.

ret the `<gtk-accel-group>` associated with the menu.

`gtk-menu-set-accel-path` (*self* `<gtk-menu>`) (*accel_path* `mchars`) [Function]
`set-accel-path` [Method]
Sets an accelerator path for this menu from which accelerator paths for its immediate children, its menu items, can be constructed. The main purpose of this function is to spare the programmer the inconvenience of having to call `gtk-menu-item-set-accel-path` on each menu item that should support runtime user changable accelerators. Instead, by just calling `gtk-menu-set-accel-path` on their parent, each menu item of this menu, that contains a label describing its purpose, automatically gets an accel path assigned. For example, a menu containing menu items `"New"` and `"Exit"`, will, after '`gtk_menu_set_accel_path (menu, "<Gnumeric-Sheet>/File");`' has been called, assign its items the accel paths: '`"<Gnumeric-Sheet>/File/New"`' and '`"<Gnumeric-Sheet>/File/Exit"`'. Assigning accel paths to menu items then enables the user to change their accelerators at runtime. More details about accelerator paths and their default setups can be found at `gtk-accel-map-add-entry`.

menu a valid `<gtk-menu>`

accel-path a valid accelerator path

`gtk-menu-set-title` (*self* `<gtk-menu>`) (*title* `mchars`) [Function]
`set-title` [Method]
Sets the title string for the menu. The title is displayed when the menu is shown as a tearoff menu. If *title* is '`#f`', the menu will see if it is attached to a parent menu item, and if so it will try to use the same text as that menu item's label.

menu a `<gtk-menu>`

title a string containing the title for the menu.

`gtk-menu-get-tearoff-state` (*self* `<gtk-menu>`) ⇒ (*ret* `bool`) [Function]
`get-tearoff-state` [Method]
Returns whether the menu is torn off. See `gtk-menu-set-tearoff-state`.

menu a `<gtk-menu>`

ret '`#t`' if the menu is currently torn off.

`gtk-menu-get-title` (*self* `<gtk-menu>`) ⇒ (*ret* `mchars`) [Function]
`get-title` [Method]

Returns the title of the menu. See `gtk-menu-set-title`.

menu a `<gtk-menu>`

ret the title of the menu, or '`#f`' if the menu has no title set on it. This string is owned by the widget and should not be modified or freed.

`gtk-menu-popdown` (*self* `<gtk-menu>`) [Function]
`popdown` [Method]

Removes the menu from the screen.

menu a `<gtk-menu>`.

`gtk-menu-reposition` (*self* `<gtk-menu>`) [Function]
`reposition` [Method]

Repositions the menu according to its position function.

menu a `<gtk-menu>`.

`gtk-menu-get-active` (*self* `<gtk-menu>`) ⇒ (*ret* `<gtk-widget>`) [Function]
`get-active` [Method]

Returns the selected menu item from the menu. This is used by the `<gtk-option-menu>`.

menu a `<gtk-menu>`.

ret the `<gtk-menu-item>` that was last selected in the menu. If a selection has not yet been made, the first menu item is selected.

`gtk-menu-set-active` (*self* `<gtk-menu>`) (*index* `unsigned-int`) [Function]
`set-active` [Method]

Selects the specified menu item within the menu. This is used by the `<gtk-option-menu>` and should not be used by anyone else.

menu a `<gtk-menu>`.

index the index of the menu item to select. Index values are from 0 to n-1.

`gtk-menu-set-tearoff-state` (*self* `<gtk-menu>`) (*torn_off* `bool`) [Function]
`set-tearoff-state` [Method]

Changes the tearoff state of the menu. A menu is normally displayed as drop down menu which persists as long as the menu is active. It can also be displayed as a tearoff menu which persists until it is closed or reattached.

menu a `<gtk-menu>`.

torn-off If '`#t`', menu is displayed as a tearoff menu.

`gtk-menu-attach-to-widget` (*self* `<gtk-menu>`) [Function]
(*attach_widget* `<gtk-widget>`) (*detacher* `<gtk-menu-detach-func>`)
`attach-to-widget` [Method]

Attaches the menu to the widget and provides a callback function that will be invoked when the menu calls `gtk-menu-detach` during its destruction.

menu a `<gtk-menu>`.

attach-widget
the `<gtk-widget>` that the menu will be attached to.

detacher the user supplied callback function that will be called when the menu calls `gtk-menu-detach`.

`gtk-menu-detach` (*self* `<gtk-menu>`) [Function]
`detach` [Method]
Detaches the menu from the widget to which it had been attached. This function will call the callback function, *detacher*, provided when the `gtk-menu-attach-to-widget` function was called.

menu a `<gtk-menu>`.

`gtk-menu-get-attach-widget` (*self* `<gtk-menu>`) [Function]
⇒ (*ret* `<gtk-widget>`)
`get-attach-widget` [Method]
Returns the `<gtk-widget>` that the menu is attached to.

menu a `<gtk-menu>`.

ret the `<gtk-widget>` that the menu is attached to.

`gtk-menu-get-for-attach-widget` (*widget* `<gtk-widget>`) [Function]
⇒ (*ret* `glist-of`)
Returns a list of the menus which are attached to this widget. This list is owned by GTK+ and must not be modified.

widget a `<gtk-widget>`

ret the list of menus attached to his widget.

Since 2.6

`gtk-menu-set-monitor` (*self* `<gtk-menu>`) (*monitor_num* `int`) [Function]
`set-monitor` [Method]
Informs GTK+ on which monitor a menu should be popped up. See `gdk-screen-get-monitor-geometry`.

This function should be called from a `<gtk-menu-position-func>` if the menu should not appear on the same monitor as the pointer. This information can't be reliably inferred from the coordinates returned by a `<gtk-menu-position-func>`, since, for very long menus, these coordinates may extend beyond the monitor boundaries or even the screen boundaries.

menu a `<gtk-menu>`

monitor-num
the number of the monitor on which the menu should be popped up

Since 2.4

57 GtkMenuBar

A subclass widget for which holds widgets

57.1 Overview

The <gtk-menu-bar> is a subclass of <gtk-menu-shell> which contains one to many <gtk-menu-item>. The result is a standard menu bar which can hold many menu items. <gtk-menu-bar> allows for a shadow type to be set for aesthetic purposes. The shadow types are defined in the <gtk-menu-bar-set-shadow-type> function.

57.2 Usage

<gtk-menu-bar> [Class]

This <gobject> class defines the following properties:

pack-direction
: The pack direction of the menubar

child-pack-direction
: The child pack direction of the menubar

gtk-menu-bar-new ⇒ (*ret* <gtk-widget>) [Function]

Creates the new <gtk-menu-bar>

ret the <gtk-menu-bar>

gtk-menu-bar-set-pack-direction (*self* <gtk-menu-bar>) [Function]
(*pack_dir* <gtk-pack-direction>)
set-pack-direction [Method]

Sets how items should be packed inside a menubar.

menubar a <gtk-menu-bar>.

pack-dir a new <gtk-pack-direction>.

Since 2.8

gtk-menu-bar-get-pack-direction (*self* <gtk-menu-bar>) [Function]
⇒ (*ret* <gtk-pack-direction>)
get-pack-direction [Method]

Retrieves the current pack direction of the menubar. See gtk-menu-bar-set-pack-direction.

menubar a <gtk-menu-bar>

ret the pack direction

Since 2.8

58 GtkMenuItem

The widget used for item in menus

58.1 Overview

The <gtk-menu-item> widget and the derived widgets are the only valid childs for menus. Their function is to correctly handle highlighting, alignment, events and submenus.

As it derives from <gtk-bin> it can hold any valid child widget, altough only a few are really useful.

58.2 Usage

<gtk-menu-item> [Class]
This <gobject> class defines the following properties:

submenu The submenu attached to the menu item, or NULL if it has none

activate [Signal on <gtk-menu-item>]
Emitted when the item is activated.

activate-item [Signal on <gtk-menu-item>]
Emitted when the item is activated, but also if the menu item has a submenu. For normal applications, the relevant signal is "activate".

toggle-size-request (*arg0* <gpointer>) [Signal on <gtk-menu-item>]

toggle-size-allocate (*arg0* <gint>) [Signal on <gtk-menu-item>]

gtk-menu-item-new ⇒ (*ret* <gtk-widget>) [Function]
Creates a new <gtk-menu-item>.

ret the newly created <gtk-menu-item>

gtk-menu-item-new-with-label (*label* mchars) [Function]
⇒ (*ret* <gtk-widget>)
Creates a new <gtk-menu-item> whose child is a <gtk-label>.

label the text for the label

ret the newly created <gtk-menu-item>

gtk-menu-item-new-with-mnemonic (*label* mchars) [Function]
⇒ (*ret* <gtk-widget>)
Creates a new <gtk-menu-item> containing a label. The label will be created using gtk-label-new-with-mnemonic, so underscores in *label* indicate the mnemonic for the menu item.

label The text of the button, with an underscore in front of the mnemonic character

ret a new <gtk-menu-item>

`gtk-menu-item-set-right-justified` (*self* `<gtk-menu-item>`) [Function]
(*right_justified* `bool`)
`set-right-justified` [Method]

Sets whether the menu item appears justified at the right side of a menu bar. This was traditionally done for `"Help"` menu items, but is now considered a bad idea. (If the widget layout is reversed for a right-to-left language like Hebrew or Arabic, right-justified-menu-items appear at the left.)

menu-item
a `<gtk-menu-item>`.

right-justified
if '`#t`' the menu item will appear at the far right if added to a menu bar.

`gtk-menu-item-set-submenu` (*self* `<gtk-menu-item>`) [Function]
(*submenu* `<gtk-widget>`)
`set-submenu` [Method]

Sets the widget submenu, or changes it.

menu-item
the menu item widget

submenu the submenu

`gtk-menu-item-set-accel-path` (*self* `<gtk-menu-item>`) [Function]
(*accel_path* `mchars`)
`set-accel-path` [Method]

Set the accelerator path on *menu-item*, through which runtime changes of the menu item's accelerator caused by the user can be identified and saved to persistant storage (see `gtk-accel-map-save` on this). To setup a default accelerator for this menu item, call `gtk-accel-map-add-entry` with the same *accel-path*. See also `gtk-accel-map-add-entry` on the specifics of accelerator paths, and `gtk-menu-set-accel-path` for a more convenient variant of this function.

This function is basically a convenience wrapper that handles calling `gtk-widget-set-accel-path` with the appropriate accelerator group for the menu item.

Note that you do need to set an accelerator on the parent menu with `gtk-menu-set-accel-group` for this to work.

menu-item
a valid `<gtk-menu-item>`

accel-path accelerator path, corresponding to this menu item's functionality, or '`#f`' to unset the current path.

`gtk-menu-item-remove-submenu` (*self* `<gtk-menu-item>`) [Function]
`remove-submenu` [Method]

Removes the widget's submenu.

menu-item
the menu item widget

`gtk-menu-item-select` (*self* `<gtk-menu-item>`) [Function]
`select` [Method]
Emits the "select" signal on the given item. Behaves exactly like `<gtk-item-select>`.

menu-item
the menu item

`gtk-menu-item-deselect` (*self* `<gtk-menu-item>`) [Function]
`deselect` [Method]
Emits the "deselect" signal on the given item. Behaves exactly like `<gtk-item-deselect>`.

menu-item
the menu item

`gtk-menu-item-activate` (*self* `<gtk-menu-item>`) [Function]
`activate` [Method]
Emits the "activate" signal on the given item

menu-item
the menu item

`gtk-menu-item-toggle-size-request` (*self* `<gtk-menu-item>`) [Function]
⇒ (*requisition* `int`)
`toggle-size-request` [Method]
Emits the "toggle_size_request" signal on the given item.

menu-item
the menu item

requisition the requisition to use as signal data.

`gtk-menu-item-toggle-size-allocate` (*self* `<gtk-menu-item>`) [Function]
(*allocation* `int`)
`toggle-size-allocate` [Method]
Emits the "toggle_size_allocate" signal on the given item.

menu-item
the menu item.

allocation the allocation to use as signal data.

`gtk-menu-item-get-right-justified` (*self* `<gtk-menu-item>`) [Function]
⇒ (*ret* `bool`)
`get-right-justified` [Method]
Gets whether the menu item appears justified at the right side of the menu bar.

menu-item
a `<gtk-menu-item>`

ret '`#t`' if the menu item will appear at the far right if added to a menu bar.

`gtk-menu-item-get-submenu` (*self* `<gtk-menu-item>`) [Function]
⇒ (*ret* `<gtk-widget>`)

`get-submenu` [Method]

Gets the submenu underneath this menu item, if any. See `gtk-menu-item-set-submenu`.

menu-item
a `<gtk-menu-item>`

ret submenu for this menu item, or '`#f`' if none.

59 GtkMenuShell

A base class for menu objects

59.1 Overview

A <gtk-menu-shell> is the abstract base class used to derive the <gtk-menu> and <gtk-menu-bar> subclasses.

A <gtk-menu-shell> is a container of <gtk-menu-item> objects arranged in a list which can be navigated, selected, and activated by the user to perform application functions. A <gtk-menu-item> can have a submenu associated with it, allowing for nested hierarchical menus.

59.2 Usage

<gtk-menu-shell> [Class]

This <gobject> class defines the following properties:

take-focus

A boolean that determines whether the menu grabs the keyboard focus

cancel [Signal on <gtk-menu-shell>]

An action signal which cancels the selection within the menu shell. Causes the GtkMenuShell::selection-done signal to be emitted.

deactivate [Signal on <gtk-menu-shell>]

This signal is emitted when a menu shell is deactivated.

selection-done [Signal on <gtk-menu-shell>]

This signal is emitted when a selection has been completed within a menu shell.

move-current [Signal on <gtk-menu-shell>]

(*arg0* <gtk-menu-direction-type>)

An action signal which moves the current menu item in the direction specified by *direction*.

activate-current (*arg0* <gboolean>) [Signal on <gtk-menu-shell>]

An action signal that activates the current menu item within the menu shell.

cycle-focus (*arg0* <gtk-direction-type>) [Signal on <gtk-menu-shell>]

move-selected (*arg0* <gint>) ⇒ <gboolean> [Signal on <gtk-menu-shell>]

undocumented

gtk-menu-shell-append (*self* <gtk-menu-shell>) [Function]

(*child* <gtk-widget>)

append [Method]

Adds a new <gtk-menu-item> to the end of the menu shell's item list.

menu-shell

a <gtk-menu-shell>.

child The <gtk-menu-item> to add.

`gtk-menu-shell-prepend` (*self* `<gtk-menu-shell>`) [Function]
(*child* `<gtk-widget>`)
`prepend` [Method]

Adds a new `<gtk-menu-item>` to the beginning of the menu shell's item list.

menu-shell
a `<gtk-menu-shell>`.

child The `<gtk-menu-item>` to add.

`gtk-menu-shell-insert` (*self* `<gtk-menu-shell>`) [Function]
(*child* `<gtk-widget>`) (*position* `int`)
`insert` [Method]

Adds a new `<gtk-menu-item>` to the menu shell's item list at the position indicated by *position*.

menu-shell
a `<gtk-menu-shell>`.

child The `<gtk-menu-item>` to add.

position The position in the item list where *child* is added. Positions are numbered from 0 to n-1.

`gtk-menu-shell-deactivate` (*self* `<gtk-menu-shell>`) [Function]
`deactivate` [Method]

Deactivates the menu shell. Typically this results in the menu shell being erased from the screen.

menu-shell
a `<gtk-menu-shell>`.

`gtk-menu-shell-select-item` (*self* `<gtk-menu-shell>`) [Function]
(*menu_item* `<gtk-widget>`)
`select-item` [Method]

Selects the menu item from the menu shell.

menu-shell
a `<gtk-menu-shell>`.

menu-item
The `<gtk-menu-item>` to select.

`gtk-menu-shell-select-first` (*self* `<gtk-menu-shell>`) [Function]
(*search_sensitive* `bool`)
`select-first` [Method]

Select the first visible or selectable child of the menu shell; don't select tearoff items unless the only item is a tearoff item.

menu-shell
a `<gtk-menu-shell>`

search-sensitive
if '`#t`', search for the first selectable menu item, otherwise select nothing if the first item isn't sensitive. This should be '`#f`' if the menu is being popped up initially.

Since 2.2

`gtk-menu-shell-deselect` (*self* `<gtk-menu-shell>`) [Function]
`deselect` [Method]

Deselects the currently selected item from the menu shell, if any.

menu-shell
a `<gtk-menu-shell>`.

`gtk-menu-shell-activate-item` (*self* `<gtk-menu-shell>`) [Function]
(*menu_item* `<gtk-widget>`) (*force_deactivate* `bool`)
`activate-item` [Method]

Activates the menu item within the menu shell.

menu-shell
a `<gtk-menu-shell>`.

menu-item
The `<gtk-menu-item>` to activate.

force-deactivate
If TRUE, force the deactivation of the menu shell after the menu item is activated.

`gtk-menu-shell-cancel` (*self* `<gtk-menu-shell>`) [Function]
`cancel` [Method]

Cancels the selection within the menu shell.

menu-shell
a `<gtk-menu-shell>`

Since 2.4

`gtk-menu-shell-set-take-focus` (*self* `<gtk-menu-shell>`) [Function]
(*take_focus* `bool`)
`set-take-focus` [Method]

If *take-focus* is '`#t`' (the default) the menu shell will take the keyboard focus so that it will receive all keyboard events which is needed to enable keyboard navigation in menus.

Setting *take-focus* to '`#f`' is useful only for special applications like virtual keyboard implementations which should not take keyboard focus.

The *take-focus* state of a menu or menu bar is automatically propagated to submenus whenever a submenu is popped up, so you don't have to worry about recursively setting it for your entire menu hierarchy. Only when programmatically picking a submenu and popping it up manually, the *take-focus* property of the submenu needs to be set explicitely.

Note that setting it to '`#f`' has side-effects:

If the focus is in some other app, it keeps the focus and keynav in the menu doesn't work. Consequently, keynav on the menu will only work if the focus is on some toplevel owned by the onscreen keyboard.

To avoid confusing the user, menus with *take-focus* set to '`#f`' should not display mnemonics or accelerators, since it cannot be guaranteed that they will work.

See also `gdk-keyboard-grab`

menu-shell
: a `<gtk-menu-shell>`

take-focus
: '`#t`' if the menu shell should take the keyboard focus on popup.

Since 2.8

`gtk-menu-shell-get-take-focus` (*self* `<gtk-menu-shell>`) [Function]
⇒ (*ret* `bool`)

`get-take-focus` [Method]

Returns '`#t`' if the menu shell will take the keyboard focus on popup.

menu-shell
: a `<gtk-menu-shell>`

ret
: '`#t`' if the menu shell will take the keyboard focus on popup.

Since 2.8

60 GtkImageMenuItem

A menu item with an icon

60.1 Overview

A GtkImageMenuItem is a menu item which has an icon next to the text label.

Note that the user can disable display of menu icons, so make sure to still fill in the text label.

60.2 Usage

`<gtk-image-menu-item>` [Class]

This `<gobject>` class defines the following properties:

`image` Child widget to appear next to the menu text

`gtk-image-menu-item-set-image` (*self* `<gtk-image-menu-item>`) (*image* `<gtk-widget>`) [Function]

`set-image` [Method]

Sets the image of *image-menu-item* to the given widget. Note that it depends on the show-menu-images setting whether the image will be displayed or not.

image-menu-item
a `<gtk-image-menu-item>`.

image a widget to set as the image for the menu item.

`gtk-image-menu-item-get-image` (*self* `<gtk-image-menu-item>`) ⇒ (*ret* `<gtk-widget>`) [Function]

`get-image` [Method]

Gets the widget that is currently set as the image of *image-menu-item*. See `gtk-image-menu-item-set-image`.

image-menu-item
a `<gtk-image-menu-item>`.

ret the widget set as image of *image-menu-item*.

`gtk-image-menu-item-new` ⇒ (*ret* `<gtk-widget>`) [Function]

Creates a new `<gtk-image-menu-item>` with an empty label.

ret a new `<gtk-image-menu-item>`.

`gtk-image-menu-item-new-from-stock` (*stock_id* `mchars`) (*accel_group* `<gtk-accel-group>`) ⇒ (*ret* `<gtk-widget>`) [Function]

Creates a new `<gtk-image-menu-item>` containing the image and text from a stock item. Some stock ids have preprocessor macros like `<gtk-stock-ok>` and `<gtk-stock-apply>`.

If you want this menu item to have changeable accelerators, then pass in '`#f`' for accel_group. Next call `gtk-menu-item-set-accel-path` with an appropriate path for the menu item, use `gtk-stock-lookup` to look up the standard accelerator for the stock item, and if one is found, call `gtk-accel-map-add-entry` to register it.

stock-id the name of the stock item.

accel-group
the `<gtk-accel-group>` to add the menu items accelerator to, or '`#f`'.

ret a new `<gtk-image-menu-item>`.

`gtk-image-menu-item-new-with-label` (*label* `mchars`) [Function]
⇒ (*ret* `<gtk-widget>`)

Creates a new `<gtk-image-menu-item>` containing a label.

label the text of the menu item.

ret a new `<gtk-image-menu-item>`.

61 GtkRadioMenuItem

A choice from multiple check menu items

61.1 Overview

A radio menu item is a check menu item that belongs to a group. At each instant exactly one of the radio menu items from a group is selected.

The group list does not need to be freed, as each <gtk-radio-menu-item> will remove itself and its list item when it is destroyed.

The correct way to create a group of radio menu items is approximatively this:

```
GSList *group = NULL;
GtkWidget *item;
gint i;

for (i = 0; i < 5; i++)
{
  item = gtk_radio_menu_item_new_with_label (group, "This is an example");
  group = gtk_radio_menu_item_get_group (GTK_RADIO_MENU_ITEM (item));
  if (i == 1)
    gtk_check_menu_item_set_active (GTK_CHECK_MENU_ITEM (item), TRUE);
}
```

61.2 Usage

<gtk-radio-menu-item> [Class]

This <gobject> class defines the following properties:

group The radio menu item whose group this widget belongs to.

group-changed [Signal on <gtk-radio-menu-item>]

gtk-radio-menu-item-new (*group* <gtk-radio-group*>) [Function]
⇒ (*ret* <gtk-widget>)

Creates a new <gtk-radio-menu-item>.

group the group to which the radio menu item is to be attached

ret a new <gtk-radio-menu-item>

gtk-radio-menu-item-new-with-label [Function]
(*group* <gtk-radio-group*>) (*label* mchars) ⇒ (*ret* <gtk-widget>)

Creates a new <gtk-radio-menu-item> whose child is a simple <gtk-label>.

group the group to which the radio menu item is to be attached

label the text for the label

ret a new <gtk-radio-menu-item>

`gtk-radio-menu-item-new-from-widget` [Function]
(*group* `<gtk-radio-menu-item>`) ⇒ (*ret* `<gtk-widget>`)

Creates a new `<gtk-radio-menu-item>` adding it to the same group as *group*.

group An existing `<gtk-radio-menu-item>`

ret The new `<gtk-radio-menu-item>`

Since 2.4

`gtk-radio-menu-item-set-group` (*self* `<gtk-radio-menu-item>`) [Function]
(*group* `<gtk-radio-group*>`)

`set-group` [Method]

Sets the group of a radio menu item, or changes it.

radio-menu-item
a `<gtk-radio-menu-item>`.

group the new group.

`gtk-radio-menu-item-get-group` (*self* `<gtk-radio-menu-item>`) [Function]
⇒ (*ret* `<gtk-radio-group*>`)

`get-group` [Method]

Returns the group to which the radio menu item belongs, as a `<g-list>` of `<gtk-radio-menu-item>`. The list belongs to GTK+ and should not be freed.

radio-menu-item
a `<gtk-radio-menu-item>`.

ret the group of *radio-menu-item*.

62 GtkCheckMenuItem

A menu item with a check box

62.1 Overview

A <gtk-check-menu-item> is a menu item that maintains the state of a boolean value in addition to a <gtk-menu-item>'s usual role in activating application code.

A check box indicating the state of the boolean value is displayed at the left side of the <gtk-menu-item>. Activating the <gtk-menu-item> toggles the value.

62.2 Usage

<gtk-check-menu-item> [Class]

This <gobject> class defines the following properties:

`active` Whether the menu item is checked

`inconsistent`
Whether to display an "inconsistent" state

`draw-as-radio`
Whether the menu item looks like a radio menu item

toggled [Signal on <gtk-check-menu-item>]

This signal is emitted when the state of the check box is changed.

A signal handler can examine the field of the <gtk-check-menu-item> struct to discover the new state.

gtk-check-menu-item-new ⇒ (*ret* <gtk-widget>) [Function]

Creates a new <gtk-check-menu-item>.

ret a new <gtk-check-menu-item>.

gtk-check-menu-item-new-with-label (*label* mchars) [Function]
⇒ (*ret* <gtk-widget>)

Creates a new <gtk-check-menu-item> with a label.

label the string to use for the label.

ret a new <gtk-check-menu-item>.

gtk-check-menu-item-get-active (*self* <gtk-check-menu-item>) [Function]
⇒ (*ret* bool)

get-active [Method]

Returns whether the check menu item is active. See gtk-check-menu-item-set-active.

check-menu-item
a <gtk-check-menu-item>

ret '#t' if the menu item is checked.

`gtk-check-menu-item-set-active` (*self* `<gtk-check-menu-item>`) [Function]
(*is_active* `bool`)

`set-active` [Method]

Sets the active state of the menu item's check box.

check-menu-item
a `<gtk-check-menu-item>`.

is-active boolean value indicating whether the check box is active.

`gtk-check-menu-item-toggled` (*self* `<gtk-check-menu-item>`) [Function]

`toggled` [Method]

Emits the GtkCheckMenuItem::toggled signal.

check-menu-item
a `<gtk-check-menu-item>`.

63 GtkSeparatorMenuItem

A separator used in menus

63.1 Overview

The `<gtk-separator-menu-item>` is a separator used to group items within a menu. It displays a horizontal line with a shadow to make it appear sunken into the interface.

63.2 Usage

`<gtk-separator-menu-item>` [Class]

This `<gobject>` class defines no properties, other than those defined by its superclasses.

`gtk-separator-menu-item-new` ⇒ (*ret* `<gtk-widget>`) [Function]

Creates a new `<gtk-separator-menu-item>`.

ret a new `<gtk-separator-menu-item>`.

64 GtkTearoffMenuItem

A menu item used to tear off and reattach its menu

64.1 Overview

A <gtk-tearoff-menu-item> is a special <gtk-menu-item> which is used to tear off and reattach its menu.

When its menu is shown normally, the <gtk-tearoff-menu-item> is drawn as a dotted line indicating that the menu can be torn off. Activating it causes its menu to be torn off and displayed in its own window as a tearoff menu.

When its menu is shown as a tearoff menu, the <gtk-tearoff-menu-item> is drawn as a dotted line which has a left pointing arrow graphic indicating that the tearoff menu can be reattached. Activating it will erase the tearoff menu window.

64.2 Usage

<gtk-tearoff-menu-item> [Class]
This <gobject> class defines no properties, other than those defined by its superclasses.

gtk-tearoff-menu-item-new ⇒ (*ret* <gtk-widget>) [Function]
Creates a new <gtk-tearoff-menu-item>.

ret a new <gtk-tearoff-menu-item>.

65 GtkToolbar

Create bars of buttons and other widgets

65.1 Overview

A toolbar is created with a call to `gtk-toolbar-new`.

A toolbar can contain instances of a subclass of `<gtk-tool-item>`. To add a `<gtk-tool-item>` to the a toolbar, use `gtk-toolbar-insert`. To remove an item from the toolbar use `gtk-container-remove`. To add a button to the toolbar, add an instance of `<gtk-tool-button>`.

Toolbar items can be visually grouped by adding instances of `<gtk-separator-tool-item>` to the toolbar. If a `<gtk-separator-tool-item>` has the `"expand"` property set to `#t` and the `"draw"` property set to `#f` the effect is to force all following items to the end of the toolbar.

Creating a context menu for the toolbar can be done by connecting to the `<gtk-toolbar::popup-context-menu>` signal.

65.2 Usage

`<gtk-toolbar>` [Class]

This `<gobject>` class defines the following properties:

`orientation`
: The orientation of the toolbar

`toolbar-style`
: How to draw the toolbar

`show-arrow`
: If an arrow should be shown if the toolbar doesn't fit

`tooltips` If the tooltips of the toolbar should be active or not

`icon-size`
: Size of icons in this toolbar

`icon-size-set`
: Whether the icon-size property has been set

`orientation-changed` (*arg0* `<gtk-orientation>`) [Signal on `<gtk-toolbar>`]

Emitted when the orientation of the toolbar changes.

`style-changed` (*arg0* `<gtk-toolbar-style>`) [Signal on `<gtk-toolbar>`]

Emitted when the style of the toolbar changes.

`popup-context-menu` (*arg0* `<gint>`) (*arg1* `<gint>`) (*arg2* `<gint>`) ⇒ `<gboolean>` [Signal on `<gtk-toolbar>`]

Emitted when the user right-clicks the toolbar or uses the keybinding to display a popup menu.

Application developers should handle this signal if they want to display a context menu on the toolbar. The context-menu should appear at the coordinates given by *x* and *y*. The mouse button number is given by the *button* parameter. If the menu was popped up using the keybaord, *button* is -1.

`focus-home-or-end` (*arg0* `<gboolean>`) [Signal on `<gtk-toolbar>`]
⇒ `<gboolean>`

A keybinding signal used internally by GTK+. This signal can't be used in application code

`gtk-toolbar-new` ⇒ (*ret* `<gtk-widget>`) [Function]

Creates a new toolbar.

ret the newly-created toolbar.

`gtk-toolbar-insert` (*self* `<gtk-toolbar>`) (*item* `<gtk-tool-item>`) [Function]
(*pos* `int`)
`insert` [Method]

Insert a `<gtk-tool-item>` into the toolbar at position *pos*. If *pos* is 0 the item is prepended to the start of the toolbar. If *pos* is negative, the item is appended to the end of the toolbar.

toolbar a `<gtk-toolbar>`

item a `<gtk-tool-item>`

pos the position of the new item

Since 2.4

`gtk-toolbar-get-item-index` (*self* `<gtk-toolbar>`) [Function]
(*item* `<gtk-tool-item>`) ⇒ (*ret* `int`)
`get-item-index` [Method]

Returns the position of *item* on the toolbar, starting from 0. It is an error if *item* is not a child of the toolbar.

toolbar a `<gtk-toolbar>`

item a `<gtk-tool-item>` that is a child of *toolbar*

ret the position of item on the toolbar.

Since 2.4

`gtk-toolbar-get-n-items` (*self* `<gtk-toolbar>`) ⇒ (*ret* `int`) [Function]
`get-n-items` [Method]

Returns the number of items on the toolbar.

toolbar a `<gtk-toolbar>`

ret the number of items on the toolbar

Since 2.4

`gtk-toolbar-get-nth-item` (*self* `<gtk-toolbar>`) (*n* `int`) [Function]
⇒ (*ret* `<gtk-tool-item>`)

`get-nth-item` [Method]

Returns the *n*'th item on *toolbar*, or '`#f`' if the toolbar does not contain an *n*'th item.

toolbar a `<gtk-toolbar>`

n A position on the toolbar

ret The *n*'th `<gtk-tool-item>` on *toolbar*, or '`#f`' if there isn't an *n*'th item.

Since 2.4

`gtk-toolbar-get-drop-index` (*self* `<gtk-toolbar>`) (*x* `int`) (*y* `int`) [Function]
⇒ (*ret* `int`)

`get-drop-index` [Method]

Returns the position corresponding to the indicated point on *toolbar*. This is useful when dragging items to the toolbar: this function returns the position a new item should be inserted.

x and *y* are in *toolbar* coordinates.

toolbar a `<gtk-toolbar>`

x x coordinate of a point on the toolbar

y y coordinate of a point on the toolbar

ret The position corresponding to the point (*x*, *y*) on the toolbar.

Since 2.4

`gtk-toolbar-set-drop-highlight-item` (*self* `<gtk-toolbar>`) [Function]
(*tool_item* `<gtk-tool-item>`) (*index_* `int`)

`set-drop-highlight-item` [Method]

Highlights *toolbar* to give an idea of what it would look like if *item* was added to *toolbar* at the position indicated by *index*. If *item* is '`#f`', highlighting is turned off. In that case *index* is ignored.

The *tool-item* passed to this function must not be part of any widget hierarchy. When an item is set as drop highlight item it can not added to any widget hierarchy or used as highlight item for another toolbar.

toolbar a `<gtk-toolbar>`

tool-item a `<gtk-tool-item>`, or '`#f`' to turn of highlighting

index a position on *toolbar*

Since 2.4

`gtk-toolbar-set-show-arrow` (*self* `<gtk-toolbar>`) [Function]
(*show_arrow* `bool`)

`set-show-arrow` [Method]

Sets whether to show an overflow menu when *toolbar* doesn't have room for all items on it. If '`#t`', items that there are not room are available through an overflow menu.

toolbar a <gtk-toolbar>

show-arrow
Whether to show an overflow menu

Since 2.4

`gtk-toolbar-set-orientation` (*self* <gtk-toolbar>) (*orientation* <gtk-orientation>) [Function]
`set-orientation` [Method]
Sets whether a toolbar should appear horizontally or vertically.

toolbar a <gtk-toolbar>.

orientation
a new <gtk-orientation>.

`gtk-toolbar-set-tooltips` (*self* <gtk-toolbar>) (*enable* bool) [Function]
`set-tooltips` [Method]
Sets if the tooltips of a toolbar should be active or not.

toolbar a <gtk-toolbar>.

enable set to '#f' to disable the tooltips, or '#t' to enable them.

`gtk-toolbar-get-show-arrow` (*self* <gtk-toolbar>) ⇒ (*ret* bool) [Function]
`get-show-arrow` [Method]
Returns whether the toolbar has an overflow menu. See `gtk-toolbar-set-show-arrow`.

toolbar a <gtk-toolbar>

ret '#t' if the toolbar has an overflow menu.

Since 2.4

`gtk-toolbar-get-orientation` (*self* <gtk-toolbar>) ⇒ (*ret* <gtk-orientation>) [Function]
`get-orientation` [Method]
Retrieves the current orientation of the toolbar. See `gtk-toolbar-set-orientation`.

toolbar a <gtk-toolbar>

ret the orientation

`gtk-toolbar-get-style` (*self* <gtk-toolbar>) ⇒ (*ret* <gtk-toolbar-style>) [Function]
`get-style` [Method]
Retrieves whether the toolbar has text, icons, or both . See `gtk-toolbar-set-style`.

toolbar a <gtk-toolbar>

ret the current style of *toolbar*

`gtk-toolbar-get-icon-size` (*self* `<gtk-toolbar>`) [Function]
⇒ (*ret* `<gtk-icon-size>`)
`get-icon-size` [Method]

Retrieves the icon size for the toolbar. See `gtk-toolbar-set-icon-size`.

toolbar a `<gtk-toolbar>`

ret the current icon size for the icons on the toolbar.

`gtk-toolbar-get-tooltips` (*self* `<gtk-toolbar>`) ⇒ (*ret* `bool`) [Function]
`get-tooltips` [Method]

Retrieves whether tooltips are enabled. See `gtk-toolbar-set-tooltips`.

toolbar a `<gtk-toolbar>`

ret '`#t`' if tooltips are enabled

`gtk-toolbar-get-relief-style` (*self* `<gtk-toolbar>`) [Function]
⇒ (*ret* `<gtk-relief-style>`)
`get-relief-style` [Method]

Returns the relief style of buttons on *toolbar*. See `gtk-button-set-relief`.

toolbar a `<gtk-toolbar>`

ret The relief style of buttons on *toolbar*.

Since 2.4

`gtk-toolbar-set-style` (*self* `<gtk-toolbar>`) [Function]
(*style* `<gtk-toolbar-style>`)
`set-style` [Method]

Alters the view of *toolbar* to display either icons only, text only, or both.

toolbar a `<gtk-toolbar>`.

style the new style for *toolbar*.

`gtk-toolbar-unset-style` (*self* `<gtk-toolbar>`) [Function]
`unset-style` [Method]

Unsets a toolbar style set with `gtk-toolbar-set-style`, so that user preferences will be used to determine the toolbar style.

toolbar a `<gtk-toolbar>`

66 GtkToolItem

The base class of widgets that can be added to GtkToolbar

66.1 Overview

<gtk-tool-item>s are widgets that can appear on a toolbar. To create a toolbar item that contain something else than a button, use gtk-tool-item-new. Use gtk-container-add to add a child widget to the tool item.

For toolbar items that contain buttons, see the <gtk-tool-button>, <gtk-toggle-tool-button> and <gtk-radio-tool-button> classes.

66.2 Usage

<gtk-tool-item> [Class]

This <gobject> class defines the following properties:

visible-horizontal
: Whether the toolbar item is visible when the toolbar is in a horizontal orientation.

visible-vertical
: Whether the toolbar item is visible when the toolbar is in a vertical orientation.

is-important
: Whether the toolbar item is considered important. When TRUE, toolbar buttons show text in GTK_TOOLBAR_BOTH_HORIZ mode

create-menu-proxy ⇒ <gboolean> [Signal on <gtk-tool-item>]

This signal is emitted when the toolbar needs information from *tool-item* about whether the item should appear in the toolbar overflow menu. In response the tool item should either

The toolbar may cache the result of this signal. When the tool item changes how it will respond to this signal it must call gtk-tool-item-rebuild-menu to invalidate the cache and ensure that the toolbar rebuilds its overflow menu.

toolbar-reconfigured [Signal on <gtk-tool-item>]

This signal is emitted when some property of the toolbar that the item is a child of changes. For custom subclasses of <gtk-tool-item>, the default handler of this signal use the functions to find out what the toolbar should look like and change themselves accordingly.

set-tooltip (*arg0* <gtk-tooltips>) [Signal on <gtk-tool-item>]
(*arg1* <gchararray>) (*arg2* <gchararray>) ⇒ <gboolean>

This signal is emitted when the toolitem's tooltip changes. Application developers can use gtk-tool-item-set-tooltip to set the item's tooltip.

gtk-tool-item-new ⇒ (*ret* <gtk-tool-item>) [Function]

Creates a new <gtk-tool-item>

ret the new `<gtk-tool-item>`

Since 2.4

`gtk-tool-item-set-homogeneous` (*self* `<gtk-tool-item>`) (*homogeneous* `bool`) [Function]
`set-homogeneous` [Method]

Sets whether *tool-item* is to be allocated the same size as other homogeneous items. The effect is that all homogeneous items will have the same width as the widest of the items.

tool-item a `<gtk-tool-item:>`

homogeneous whether *tool-item* is the same size as other homogeneous items

Since 2.4

`gtk-tool-item-get-homogeneous` (*self* `<gtk-tool-item>`) ⇒ (*ret* `bool`) [Function]
`get-homogeneous` [Method]

Returns whether *tool-item* is the same size as other homogeneous items. See `gtk-tool-item-set-homogeneous`.

tool-item a `<gtk-tool-item:>`

ret '`#t`' if the item is the same size as other homogeneous item.s

Since 2.4

`gtk-tool-item-set-expand` (*self* `<gtk-tool-item>`) (*expand* `bool`) [Function]
`set-expand` [Method]

Sets whether *tool-item* is allocated extra space when there is more room on the toolbar then needed for the items. The effect is that the item gets bigger when the toolbar gets bigger and smaller when the toolbar gets smaller.

tool-item a `<gtk-tool-item:>`

expand Whether *tool-item* is allocated extra space

Since 2.4

`gtk-tool-item-get-expand` (*self* `<gtk-tool-item>`) ⇒ (*ret* `bool`) [Function]
`get-expand` [Method]

Returns whether *tool-item* is allocated extra space. See `gtk-tool-item-set-expand`.

tool-item a `<gtk-tool-item:>`

ret '`#t`' if *tool-item* is allocated extra space.

Since 2.4

`gtk-tool-item-set-tooltip` (*self* `<gtk-tool-item>`) (*tooltips* `<gtk-tooltips>`) (*tip_text* `mchars`) (*tip_private* `mchars`) [Function]
`set-tooltip` [Method]

Sets the `<gtk-tooltips>` object to be used for *tool-item*, the text to be displayed as tooltip on the item and the private text to be used. See `gtk-tooltips-set-tip`.

tool-item a `<gtk-tool-item:>`

tooltips The `<gtk-tooltips>` object to be used

tip-text text to be used as tooltip text for *tool-item*

tip-private
text to be used as private tooltip text

Since 2.4

`gtk-tool-item-set-use-drag-window` (*self* `<gtk-tool-item>`) [Function]
(*use_drag_window* `bool`)

`set-use-drag-window` [Method]

Sets whether *toolitem* has a drag window. When '`#t`' the toolitem can be used as a drag source through `gtk-drag-source-set`. When *toolitem* has a drag window it will intercept all events, even those that would otherwise be sent to a child of *toolitem*.

toolitem a `<gtk-tool-item>`

use-drag-window
Whether *toolitem* has a drag window.

Since 2.4

`gtk-tool-item-get-use-drag-window` (*self* `<gtk-tool-item>`) [Function]
⇒ (*ret* `bool`)

`get-use-drag-window` [Method]

Returns whether *toolitem* has a drag window. See `gtk-tool-item-set-use-drag-window`.

toolitem a `<gtk-tool-item>`

ret '`#t`' if *toolitem* uses a drag window.

Since 2.4

`gtk-tool-item-set-visible-vertical` (*self* `<gtk-tool-item>`) [Function]
(*visible_vertical* `bool`)

`set-visible-vertical` [Method]

Sets whether *toolitem* is visible when the toolbar is docked vertically. Some tool items, such as text entries, are too wide to be useful on a vertically docked toolbar. If *visible-vertical* is '`#f`'*toolitem* will not appear on toolbars that are docked vertically.

toolitem a `<gtk-tool-item>`

visible-vertical
whether *toolitem* is visible when the toolbar is in vertical mode

Since 2.4

`gtk-tool-item-get-visible-vertical` (*self* `<gtk-tool-item>`) [Function]
⇒ (*ret* `bool`)

`get-visible-vertical` [Method]

Returns whether *toolitem* is visible when the toolbar is docked vertically. See `gtk-tool-item-set-visible-vertical`.

toolitem a `<gtk-tool-item>`

ret Whether *toolitem* is visible when the toolbar is docked vertically

Since 2.4

`gtk-tool-item-set-is-important` (*self* `<gtk-tool-item>`) [Function]
(*is_important* `bool`)

`set-is-important` [Method]

Sets whether *tool-item* should be considered important. The `<gtk-tool-button>` class uses this property to determine whether to show or hide its label when the toolbar style is '`GTK_TOOLBAR_BOTH_HORIZ`'. The result is that only tool buttons with the "is_important" property set have labels, an effect known as "priority text"

tool-item a `<gtk-tool-item>`

is-important
whether the tool item should be considered important

Since 2.4

`gtk-tool-item-get-is-important` (*self* `<gtk-tool-item>`) [Function]
⇒ (*ret* `bool`)

`get-is-important` [Method]

Returns whether *tool-item* is considered important. See `gtk-tool-item-set-is-important`

tool-item a `<gtk-tool-item>`

ret '`#t`' if *tool-item* is considered important.

Since 2.4

`gtk-tool-item-get-icon-size` (*self* `<gtk-tool-item>`) [Function]
⇒ (*ret* `<gtk-icon-size>`)

`get-icon-size` [Method]

Returns the icon size used for *tool-item*. Custom subclasses of `<gtk-tool-item>` should call this function to find out what size icons they should use.

tool-item a `<gtk-tool-item:>`

ret a `<gtk-icon-size>` indicating the icon size used for *tool-item*

Since 2.4

`gtk-tool-item-get-orientation` (*self* `<gtk-tool-item>`) [Function]
⇒ (*ret* `<gtk-orientation>`)

`get-orientation` [Method]

Returns the orientation used for *tool-item*. Custom subclasses of `<gtk-tool-item>` should call this function to find out what size icons they should use.

tool-item a `<gtk-tool-item:>`

ret a `<gtk-orientation>` indicating the orientation used for *tool-item*

Since 2.4

`gtk-tool-item-get-toolbar-style` (*self* `<gtk-tool-item>`) [Function]
⇒ (*ret* `<gtk-toolbar-style>`)

`get-toolbar-style` [Method]

Returns the toolbar style used for *tool-item*. Custom subclasses of `<gtk-tool-item>` should call this function in the handler of the GtkToolItem::toolbar_reconfigured signal to find out in what style the toolbar is displayed and change themselves accordingly

Possibilities are:

tool-item a `<gtk-tool-item:>`

ret A `<gtk-toolbar-style>` indicating the toolbar style used for *tool-item*.

Since 2.4

`gtk-tool-item-get-relief-style` (*self* `<gtk-tool-item>`) [Function]
⇒ (*ret* `<gtk-relief-style>`)

`get-relief-style` [Method]

Returns the relief style of *tool-item*. See `gtk-button-set-relief-style`. Custom subclasses of `<gtk-tool-item>` should call this function in the handler of the `<gtk-tool-item::toolbar-reconfigured>` signal to find out the relief style of buttons.

tool-item a `<gtk-tool-item:>`

ret a `<gtk-relief-style>` indicating the relief style used for *tool-item*.

Since 2.4

`gtk-tool-item-get-proxy-menu-item` (*self* `<gtk-tool-item>`) [Function]
(*menu_item_id* `mchars`) ⇒ (*ret* `<gtk-widget>`)

`get-proxy-menu-item` [Method]

If *menu-item-id* matches the string passed to `gtk-tool-item-set-proxy-menu-item` return the corresponding `<gtk-menu-item>`.

Custom subclasses of `<gtk-tool-item>` should use this function to update their menu item when the `<gtk-tool-item>` changes. That the *menu-item-id*s must match ensures that a `<gtk-tool-item>` will not inadvertently change a menu item that they did not create.

tool-item a `<gtk-tool-item:>`

menu-item-id
a string used to identify the menu item

ret The `<gtk-menu-item>` passed to `gtk-tool-item-set-proxy-menu-item`, if the *menu-item-id*s match.

Since 2.4

`gtk-tool-item-set-proxy-menu-item` (*self* `<gtk-tool-item>`) [Function]
(*menu_item_id* `mchars`) (*menu_item* `<gtk-widget>`)

`set-proxy-menu-item` [Method]

Sets the `<gtk-menu-item>` used in the toolbar overflow menu. The *menu-item-id* is used to identify the caller of this function and should also be used with `gtk-tool-item-get-proxy-menu-item`.

tool-item a `<gtk-tool-item:>`

menu-item-id
a string used to identify *menu-item*

menu-item
a `<gtk-menu-item>` to be used in the overflow menu

Since 2.4

`gtk-tool-item-rebuild-menu` (*self* `<gtk-tool-item>`) [Function]
`rebuild-menu` [Method]

Calling this function signals to the toolbar that the overflow menu item for *tool-item* has changed. If the overflow menu is visible when this function it called, the menu will be rebuilt.

The function must be called when the tool item changes what it will do in response to the "create_menu_proxy" signal.

tool-item a `<gtk-tool-item>`

Since 2.6

67 GtkSeparatorToolItem

A toolbar item that separates groups of other toolbar items

67.1 Overview

A <gtk-separator-item> is a <gtk-tool-item> that separates groups of other <gtk-tool-items>. Depending on the theme, a <gtk-separator-tool-item> will often look like a vertical line on horizontally docked toolbars.

If the property "expand" is #t and the property "draw" is #f, a <gtk-separator-tool-item> will act as a "spring" that forces other items to the ends of the toolbar.

Use gtk-separator-tool-item-new to create a new <gtk-separator-tool-item>.

67.2 Usage

<gtk-separator-tool-item> [Class]

This <gobject> class defines the following properties:

draw Whether the separator is drawn, or just blank

gtk-separator-tool-item-new ⇒ (*ret* <gtk-tool-item>) [Function]

Create a new <gtk-separator-tool-item>

ret the new <gtk-separator-tool-item>

Since 2.4

gtk-separator-tool-item-set-draw [Function]
(*self* <gtk-separator-tool-item>) (*draw* bool)

set-draw [Method]

When *separator-tool-items* is drawn as a vertical line, or just blank. Setting this #f along with gtk-tool-item-set-expand is useful to create an item that forces following items to the end of the toolbar.

item a <gtk-separator-tool-item>

draw whether *separator-tool-item* is drawn as a vertical line

Since 2.4

gtk-separator-tool-item-get-draw [Function]
(*self* <gtk-separator-tool-item>) ⇒ (*ret* bool)

get-draw [Method]

Returns whether *separator-tool-item* is drawn as a line, or just blank. See gtk-separator-tool-item-set-draw.

item a <gtk-separator-tool-item>

ret #t if *separator-tool-item* is drawn as a line, or just blank.

Since 2.4

68 GtkToolButton

A GtkToolItem subclass that displays buttons

68.1 Overview

`<gtk-tool-button>`s are `<gtk-tool-items>` containing buttons.

Use `gtk-tool-button-new` to create a new `<gtk-tool-button>`. Use `gtk-tool-button-new-with-stock` to create a `<gtk-tool-button>` containing a stock item.

The label of a `<gtk-tool-button>` is determined by the properties `"label_widget"`, `"label"`, and `"stock_id"`. If `"label_widget"` is non-'`#f`', then that widget is used as the label. Otherwise, if `"label"` is non-'`#f`', that string is used as the label. Otherwise, if `"stock_id"` is non-'`#f`', the label is determined by the stock item. Otherwise, the button does not have a label.

The icon of a `<gtk-tool-button>` is determined by the properties `"icon_widget"` and `"stock_id"`. If `"icon_widget"` is non-'`#f`', then that widget is used as the icon. Otherwise, if `"stock_id"` is non-'`#f`', the icon is determined by the stock item. Otherwise, the button does not have a label.

68.2 Usage

`<gtk-tool-button>` [Class]

This `<gobject>` class defines the following properties:

`label` Text to show in the item.

`use-underline`
: If set, an underline in the label property indicates that the next character should be used for the mnemonic accelerator key in the overflow menu

`label-widget`
: Widget to use as the item label

`stock-id` The stock icon displayed on the item

`icon-name`
: The name of the themed icon displayed on the item

`icon-widget`
: Icon widget to display in the item

`clicked` [Signal on `<gtk-tool-button>`]

This signal is emitted when the tool button is clicked with the mouse or activated with the keyboard.

`gtk-tool-button-new` (*icon_widget* `<gtk-widget>`) (*label* `mchars`) [Function]
⇒ (*ret* `<gtk-tool-item>`)

Creates a new '`GtkToolButton`' using *icon-widget* as icon and *label* as label.

icon-widget
: a widget that will be used as icon widget, or '`#f`'

label a string that will be used as label, or '`#f`'

ret A new `<gtk-tool-button>`

Since 2.4

`gtk-tool-button-new-from-stock` (*stock_id* `mchars`) [Function]
⇒ (*ret* `<gtk-tool-item>`)

Creates a new `<gtk-tool-button>` containing the image and text from a stock item. Some stock ids have preprocessor macros like `<gtk-stock-ok>` and `<gtk-stock-apply>`.

It is an error if *stock-id* is not a name of a stock item.

stock-id the name of the stock item

ret A new `<gtk-tool-button>`

Since 2.4

`gtk-tool-button-set-label` (*self* `<gtk-tool-button>`) [Function]
(*label* `mchars`)

`set-label` [Method]

Sets *label* as the label used for the tool button. The "label" property only has an effect if not overridden by a non-'`#f`' "label_widget" property. If both the "label_widget" and "label" properties are '`#f`', the label is determined by the "stock_id" property. If the "stock_id" property is also '`#f`', *button* will not have a label.

button a `<gtk-tool-button>`

label a string that will be used as label, or '`#f`'.

Since 2.4

`gtk-tool-button-get-label` (*self* `<gtk-tool-button>`) [Function]
⇒ (*ret* `mchars`)

`get-label` [Method]

Returns the label used by the tool button, or '`#f`' if the tool button doesn't have a label. or uses a the label from a stock item. The returned string is owned by GTK+, and must not be modified or freed.

button a `<gtk-tool-button>`

ret The label, or '`#f`'

Since 2.4

`gtk-tool-button-set-use-underline` (*self* `<gtk-tool-button>`) [Function]
(*use_underline* `bool`)

`set-use-underline` [Method]

If set, an underline in the label property indicates that the next character should be used for the mnemonic accelerator key in the overflow menu. For example, if the label property is "_Open" and *use-underline* is '`#t`', the label on the tool button will be "Open" and the item on the overflow menu will have an underlined 'O'.

Labels shown on tool buttons never have mnemonics on them; this property only affects the menu item on the overflow menu.

button a `<gtk-tool-button>`

use-underline
whether the button label has the form `"_Open"`

Since 2.4

`gtk-tool-button-get-use-underline` (*self* `<gtk-tool-button>`) [Function]
⇒ (*ret* `bool`)
`get-use-underline` [Method]

Returns whether underscores in the label property are used as mnemonics on menu items on the overflow menu. See `gtk-tool-button-set-use-underline`.

button a `<gtk-tool-button>`

ret '`#t`' if underscores in the label property are used as mnemonics on menu items on the overflow menu.

Since 2.4

`gtk-tool-button-set-stock-id` (*self* `<gtk-tool-button>`) [Function]
(*stock_id* `mchars`)
`set-stock-id` [Method]

Sets the name of the stock item. See `gtk-tool-button-new-from-stock`. The stock_id property only has an effect if not overridden by non-'`#f`' `"label"` and `"icon_widget"` properties.

button a `<gtk-tool-button>`

stock-id a name of a stock item, or '`#f`'

Since 2.4

`gtk-tool-button-get-stock-id` (*self* `<gtk-tool-button>`) [Function]
⇒ (*ret* `mchars`)
`get-stock-id` [Method]

Returns the name of the stock item. See `gtk-tool-button-set-stock-id`. The returned string is owned by GTK+ and must not be freed or modifed.

button a `<gtk-tool-button>`

ret the name of the stock item for *button*.

Since 2.4

`gtk-tool-button-set-icon-name` (*self* `<gtk-tool-button>`) [Function]
(*icon_name* `mchars`)
`set-icon-name` [Method]

Sets the icon for the tool button from a named themed icon. See the docs for `<gtk-icon-theme>` for more details. The `"icon_name"` property only has an effect if not overridden by non-'`#f`' `"label"`, `"icon_widget"` and `"stock_id"` properties.

button a `<gtk-tool-button>`

icon-name the name of the themed icon

Since 2.8

`gtk-tool-button-get-icon-name` (*self* `<gtk-tool-button>`) [Function]
⇒ (*ret* `mchars`)
`get-icon-name` [Method]

Returns the name of the themed icon for the tool button, see `gtk-tool-button-set-icon-name`.

button a `<gtk-tool-button>`

ret the icon name or '`#f`' if the tool button has no themed icon

Since 2.8

`gtk-tool-button-set-icon-widget` (*self* `<gtk-tool-button>`) [Function]
(*icon_widget* `<gtk-widget>`)
`set-icon-widget` [Method]

Sets *icon* as the widget used as icon on *button*. If *icon-widget* is '`#f`' the icon is determined by the "stock_id" property. If the "stock_id" property is also '`#f`', *button* will not have an icon.

button a `<gtk-tool-button>`

icon-widget
the widget used as icon, or '`#f`'

Since 2.4

`gtk-tool-button-get-icon-widget` (*self* `<gtk-tool-button>`) [Function]
⇒ (*ret* `<gtk-widget>`)
`get-icon-widget` [Method]

Return the widget used as icon widget on *button*. See `gtk-tool-button-set-icon-widget`.

button a `<gtk-tool-button>`

ret The widget used as icon on *button*, or '`#f`'.

Since 2.4

`gtk-tool-button-set-label-widget` (*self* `<gtk-tool-button>`) [Function]
(*label_widget* `<gtk-widget>`)
`set-label-widget` [Method]

Sets *label-widget* as the widget that will be used as the label for *button*. If *label-widget* is '`#f`' the "label" property is used as label. If "label" is also '`#f`', the label in the stock item determined by the "stock_id" property is used as label. If "stock_id" is also '`#f`', *button* does not have a label.

button a `<gtk-tool-button>`

label-widget
the widget used as label, or '`#f`'

Since 2.4

`gtk-tool-button-get-label-widget` (*self* `<gtk-tool-button>`) [Function]
⇒ (*ret* `<gtk-widget>`)

`get-label-widget` [Method]

Returns the widget used as label on *button*. See `gtk-tool-button-set-label-widget`.

button a `<gtk-tool-button>`

ret The widget used as label on *button*, or '`#f`'.

Since 2.4

69 GtkMenuToolButton

A GtkToolItem containing a button with an additional dropdown menu

69.1 Overview

A <gtk-menu-tool-button> is a <gtk-tool-item> that contains a button and a small additional button with an arrow. When clicked, the arrow button pops up a dropdown menu.

Use gtk-menu-tool-button-new to create a new <gtk-menu-tool-button>. Use gtk-menu-tool-button-new-from-stock to create a new <gtk-menu-tool-button> containing a stock item.

69.2 Usage

<gtk-menu-tool-button> [Class]

This <gobject> class defines the following properties:

menu The dropdown menu

show-menu [Signal on <gtk-menu-tool-button>]

gtk-menu-tool-button-new (*icon_widget* <gtk-widget>) [Function]
(*label* mchars) ⇒ (*ret* <gtk-tool-item>)

Creates a new <gtk-menu-tool-button> using *icon-widget* as icon and *label* as label.

icon-widget
a widget that will be used as icon widget, or '#f'

label a string that will be used as label, or '#f'

ret the new <gtk-menu-tool-button>

Since 2.6

gtk-menu-tool-button-new-from-stock (*stock_id* mchars) [Function]
⇒ (*ret* <gtk-tool-item>)

Creates a new <gtk-menu-tool-button>. The new <gtk-menu-tool-button> will contain an icon and label from the stock item indicated by *stock-id*.

stock-id the name of a stock item

ret the new <gtk-menu-tool-button>

Since 2.6

gtk-menu-tool-button-set-menu (*self* <gtk-menu-tool-button>) [Function]
(*menu* <gtk-widget>)

set-menu [Method]

Sets the <gtk-menu> that is popped up when the user clicks on the arrow. If *menu* is NULL, the arrow button becomes insensitive.

button a <gtk-menu-tool-button>

menu the <gtk-menu> associated with <gtk-menu-tool-button>

Since 2.6

`gtk-menu-tool-button-get-menu` (*self* `<gtk-menu-tool-button>`) [Function]
⇒ (*ret* `<gtk-widget>`)

`get-menu` [Method]

Gets the `<gtk-menu>` associated with `<gtk-menu-tool-button>`.

button a `<gtk-menu-tool-button>`

ret the `<gtk-menu>` associated with `<gtk-menu-tool-button>`

Since 2.6

70 GtkToggleToolButton

A GtkToolItem containing a toggle button

70.1 Overview

A `<gtk-toggle-tool-button>` is a `<gtk-tool-item>` that contains a toggle button.

Use `gtk-toggle-tool-button-new` to create a new `<gtk-toggle-tool-button>`. Use `gtk-toggle-tool-button-new-from-stock` to create a new `<gtk-toggle-tool-button>` containing a stock item.

70.2 Usage

`<gtk-toggle-tool-button>` [Class]

This `<gobject>` class defines the following properties:

`active` If the toggle button should be pressed in or not

`toggled` [Signal on `<gtk-toggle-tool-button>`]

Emitted whenever the toggle tool button changes state.

`gtk-toggle-tool-button-new` ⇒ (*ret* `<gtk-tool-item>`) [Function]

Returns a new `<gtk-toggle-tool-button>`

ret a newly created `<gtk-toggle-tool-button>`

Since 2.4

`gtk-toggle-tool-button-set-active` [Function]
(*self* `<gtk-toggle-tool-button>`) (*is_active* `bool`)
`set-active` [Method]

Sets the status of the toggle tool button. Set to '`#t`' if you want the GtkToggleButton to be 'pressed in', and '`#f`' to raise it. This action causes the toggled signal to be emitted.

button a `<gtk-toggle-tool-button>`

is-active whether *button* should be active

Since 2.4

`gtk-toggle-tool-button-get-active` [Function]
(*self* `<gtk-toggle-tool-button>`) ⇒ (*ret* `bool`)
`get-active` [Method]

Queries a `<gtk-toggle-tool-button>` and returns its current state. Returns '`#t`' if the toggle button is pressed in and '`#f`' if it is raised.

button a `<gtk-toggle-tool-button>`

ret '`#t`' if the toggle tool button is pressed in, '`#f`' if not

Since 2.4

71 GtkRadioToolButton

A toolbar item that contains a radio button

71.1 Overview

A <gtk-radio-tool-button> is a <gtk-tool-item> that contains a radio button, that is, a button that is part of a group of toggle buttons where only one button can be active at a time.

Use gtk-radio-tool-button-new to create a new <gtk-radio-tool-button>. use gtk-radio-tool-button-new-from-widget to create a new <gtk-radio-tool-button> that is part of the same group as an existing <gtk-radio-tool-button>. Use gtk-radio-tool-button-new-from-stock or gtk-radio-tool-button-new-from-widget-with-stock to create a new <gtk-radio-tool-button> containing a stock item.

71.2 Usage

<gtk-radio-tool-button> [Class]

This <gobject> class defines the following properties:

group The radio tool button whose group this button belongs to.

gtk-radio-tool-button-new (*group* <gtk-radio-group*>) [Function]
⇒ (*ret* <gtk-tool-item>)

Creates a new <gtk-radio-tool-button>, adding it to *group*.

group An existing radio button group, or '#f' if you are creating a new group

ret The new <gtk-radio-tool-button>

Since 2.4

gtk-radio-tool-button-get-group [Function]
(*self* <gtk-radio-tool-button>) ⇒ (*ret* <gtk-radio-group*>)

get-group [Method]

Returns the radio button group *button* belongs to.

button a <gtk-radio-tool-button>

ret The group *button* belongs to.

Since 2.4

gtk-radio-tool-button-set-group [Function]
(*self* <gtk-radio-tool-button>) (*group* <gtk-radio-group*>)

set-group [Method]

Adds *button* to *group*, removing it from the group it belonged to before.

button a <gtk-radio-tool-button>

group an existing radio button group

Since 2.4

72 GtkUIManager

Constructing menus and toolbars from an XML description

72.1 Overview

A <gtk-ui-manager> constructs a user interface (menus and toolbars) from one or more UI definitions, which reference actions from one or more action groups.

72.2 UI Definitions

The UI definitions are specified in an XML format which can be roughly described by the following DTD. There are some additional restrictions beyond those specified in the DTD, e.g. every toolitem must have a toolbar in its anchestry and every menuitem must have a menubar or popup in its anchestry. Since a <g-markup> parser is used to parse the UI description, it must not only be valid XML, but valid <g-markup>.

```
<!ELEMENT ui          (menubar|toolbar|popup|accelerator)* >
<!ELEMENT menubar     (menuitem|separator|placeholder|menu)* >
<!ELEMENT menu        (menuitem|separator|placeholder|menu)* >
<!ELEMENT popup       (menuitem|separator|placeholder|menu)* >
<!ELEMENT toolbar     (toolitem|separator|placeholder)* >
<!ELEMENT placeholder (menuitem|toolitem|separator|placeholder|menu)* >
<!ELEMENT menuitem     EMPTY >
<!ELEMENT toolitem     (menu?) >
<!ELEMENT separator    EMPTY >
<!ELEMENT accelerator  EMPTY >
<!ATTLIST menubar      name                  &#x0023;IMPLIED
                       action                &#x0023;IMPLIED >
<!ATTLIST toolbar      name                  &#x0023;IMPLIED
                       action                &#x0023;IMPLIED >
<!ATTLIST popup        name                  &#x0023;IMPLIED
                       action                &#x0023;IMPLIED >
<!ATTLIST placeholder  name                  &#x0023;IMPLIED
                       action                &#x0023;IMPLIED >
<!ATTLIST separator    name                  &#x0023;IMPLIED
                       action                &#x0023;IMPLIED
                       expand   (true|false) &#x0023;IMPLIED >
<!ATTLIST menu         name                  &#x0023;IMPLIED
                       action                &#x0023;REQUIRED
                       position (top|bot)    &#x0023;IMPLIED >
<!ATTLIST menuitem     name                  &#x0023;IMPLIED
                       action                &#x0023;REQUIRED
                       position (top|bot)    &#x0023;IMPLIED >
<!ATTLIST toolitem     name                  &#x0023;IMPLIED
                       action                &#x0023;REQUIRED
                       position (top|bot)    &#x0023;IMPLIED >
```

```
<!ATTLIST accelerator  name                    &#x0023;IMPLIED
                       action                  &#x0023;REQUIRED >
```

If a name is not specified, it defaults to the action. If an action is not specified either, the element name is used. The name and action attributes must not contain '/' characters after parsing (since that would mess up path lookup) and must be usable as XML attributes when enclosed in doublequotes, thus they must not '"' characters or references to the " entity.

```
<ui>
  <menubar>
    <menu name="FileMenu" action="FileMenuAction">
      <menuitem name="New" action="New2Action" />
      <placeholder name="FileMenuAdditions" />
    </menu>
    <menu name="JustifyMenu" action="JustifyMenuAction">
      <menuitem name="Left" action="justify-left"/>
      <menuitem name="Centre" action="justify-center"/>
      <menuitem name="Right" action="justify-right"/>
      <menuitem name="Fill" action="justify-fill"/>
    </menu>
  </menubar>
  <toolbar action="toolbar1">
    <placeholder name="JustifyToolItems">
      <separator/>
      <toolitem name="Left" action="justify-left"/>
      <toolitem name="Centre" action="justify-center"/>
      <toolitem name="Right" action="justify-right"/>
      <toolitem name="Fill" action="justify-fill"/>
      <separator/>
    </placeholder>
  </toolbar>
</ui>
```

The constructed widget hierarchy is very similar to the element tree of the XML, with the exception that placeholders are merged into their parents. The correspondence of XML elements to widgets should be almost obvious:

toolbar

popup

menu

menuitem

toolitem

separator

accelerator

a `<gtk-menu-bar>`

a <gtk-toolbar>

a toplevel <gtk-menu>

a <gtk-menu> attached to a menuitem

a <gtk-menu-item> subclass, the exact type depends on the action

a <gtk-tool-item> subclass, the exact type depends on the action. Note that toolitem elements may contain a menu element, but only if their associated action specifies a <gtk-menu-tool-button> as proxy.

a <gtk-separator-menu-item> or <gtk-separator-tool-item>

a keyboard accelerator

The "position" attribute determines where a constructed widget is positioned wrt. to its siblings in the partially constructed tree. If it is "top", the widget is prepended, otherwise it is appended.

72.3 UI Merging

The most remarkable feature of <gtk-ui-manager> is that it can overlay a set of menuitems and toolitems over another one, and demerge them later.

Merging is done based on the names of the XML elements. Each element is identified by a path which consists of the names of its anchestors, separated by slashes. For example, the menuitem named "Left" in the example above has the path '/ui/menubar/JustifyMenu/Left' and the toolitem with the same name has path '/ui/toolbar1/JustifyToolItems/Left'.

72.4 Accelerators

Every action has an accelerator path. Accelerators are installed together with menuitem proxies, but they can also be explicitly added with <accelerator> elements in the UI definition. This makes it possible to have accelerators for actions even if they have no visible proxies.

72.5 Smart Separators

The separators created by <gtk-ui-manager> are "smart", i.e. they do not show up in the UI unless they end up between two visible menu or tool items. Separators which are located at the very beginning or end of the menu or toolbar containing them, or multiple separators next to each other, are hidden. This is a useful feature, since the merging of UI elements from multiple sources can make it hard or impossible to determine in advance whether a separator will end up in such an unfortunate position.

For separators in toolbars, you can set 'expand="true"' to turn them from a small, visible separator to an expanding, invisible one. Toolitems following an expanding separator are effectively right-aligned.

72.6 Empty Menus

Submenus pose similar problems to separators inconnection with merging. It is impossible to know in advance whether they will end up empty after merging. <gtk-ui-manager> offers

two ways to treat empty submenus: The behaviour is chosen based on the "hide_if_empty" property of the action to which the submenu is associated.

make them disappear by hiding the menu item they're attached to

add an insensitive "Empty" item

72.7 Usage

`<gtk-ui-manager>` [Class]

This `<gobject>` class defines the following properties:

`add-tearoffs`
: Whether tearoff menu items should be added to menus

`ui`
: An XML string describing the merged UI

`connect-proxy` (*arg0* `<gtk-action>`) (*arg1* `<gtk-widget>`) [Signal on `<gtk-ui-manager>`]

The connect_proxy signal is emitted after connecting a proxy to an action in the group.

This is intended for simple customizations for which a custom action class would be too clumsy, e.g. showing tooltips for menuitems in the statusbar.

Since 2.4

`disconnect-proxy` (*arg0* `<gtk-action>`) (*arg1* `<gtk-widget>`) [Signal on `<gtk-ui-manager>`]

The disconnect_proxy signal is emitted after disconnecting a proxy from an action in the group.

Since 2.4

`pre-activate` (*arg0* `<gtk-action>`) [Signal on `<gtk-ui-manager>`]

The pre_activate signal is emitted just before the *action* is activated.

This is intended for applications to get notification just before any action is activated.

Since 2.4

`post-activate` (*arg0* `<gtk-action>`) [Signal on `<gtk-ui-manager>`]

The post_activate signal is emitted just after the *action* is activated.

This is intended for applications to get notification just after any action is activated.

Since 2.4

`add-widget` (*arg0* `<gtk-widget>`) [Signal on `<gtk-ui-manager>`]

The add_widget signal is emitted for each generated menubar and toolbar. It is not emitted for generated popup menus, which can be obtained by `gtk-ui-manager-get-widget`.

Since 2.4

`actions-changed` [Signal on `<gtk-ui-manager>`]

The "actions-changed" signal is emitted whenever the set of actions changes.

Since 2.4

`gtk-ui-manager-new` ⇒ (*ret* `<gtk-ui-manager>`) [Function]

Creates a new ui manager object.

ret a new ui manager object.

Since 2.4

`gtk-ui-manager-set-add-tearoffs` (*self* `<gtk-ui-manager>`) [Function]
(*add_tearoffs* `bool`)
`set-add-tearoffs` [Method]

Sets the "add_tearoffs" property, which controls whether menus generated by this `<gtk-ui-manager>` will have tearoff menu items.

Note that this only affects regular menus. Generated popup menus never have tearoff menu items.

self a `<gtk-ui-manager>`

add-tearoffs
whether tearoff menu items are added

Since 2.4

`gtk-ui-manager-get-add-tearoffs` (*self* `<gtk-ui-manager>`) [Function]
⇒ (*ret* `bool`)
`get-add-tearoffs` [Method]

Returns whether menus generated by this `<gtk-ui-manager>` will have tearoff menu items.

self a `<gtk-ui-manager>`

ret whether tearoff menu items are added

Since 2.4

`gtk-ui-manager-insert-action-group` (*self* `<gtk-ui-manager>`) [Function]
(*action_group* `<gtk-action-group>`) (*pos* `int`)
`insert-action-group` [Method]

Inserts an action group into the list of action groups associated with *self*. Actions in earlier groups hide actions with the same name in later groups.

self a `<gtk-ui-manager>` object

action-group
the action group to be inserted

pos the position at which the group will be inserted.

Since 2.4

`gtk-ui-manager-remove-action-group` (*self* `<gtk-ui-manager>`) [Function]
(*action_group* `<gtk-action-group>`)
`remove-action-group` [Method]

Removes an action group from the list of action groups associated with *self*.

self a `<gtk-ui-manager>` object

action-group
: the action group to be removed

Since 2.4

`gtk-ui-manager-get-action-groups` (*self* `<gtk-ui-manager>`) ⇒ (*ret* `glist-of`) [Function]
`get-action-groups` [Method]

Returns the list of action groups associated with *self*.

self : a `<gtk-ui-manager>` object

ret : a `<g-list>` of action groups. The list is owned by GTK+ and should not be modified.

Since 2.4

`gtk-ui-manager-get-accel-group` (*self* `<gtk-ui-manager>`) ⇒ (*ret* `<gtk-accel-group>`) [Function]
`get-accel-group` [Method]

Returns the `<gtk-accel-group>` associated with *self*.

self : a `<gtk-ui-manager>` object

ret : the `<gtk-accel-group>`.

Since 2.4

`gtk-ui-manager-get-widget` (*self* `<gtk-ui-manager>`) (*path* `mchars`) ⇒ (*ret* `<gtk-widget>`) [Function]
`get-widget` [Method]

Looks up a widget by following a path. The path consists of the names specified in the XML description of the UI. separated by '/'. Elements which don't have a name or action attribute in the XML (e.g. <popup>) can be addressed by their XML element name (e.g. "popup"). The root element ("/ui") can be omitted in the path.

Note that the widget found by following a path that ends in a <menu> element is the menuitem to which the menu is attached, not the menu itself.

Also note that the widgets constructed by a ui manager are not tied to the lifecycle of the ui manager. If you add the widgets returned by this function to some container or explicitly ref them, they will survive the destruction of the ui manager.

self : a `<gtk-ui-manager>`

path : a path

ret : the widget found by following the path, or '`#f`' if no widget was found.

Since 2.4

`gtk-ui-manager-get-toplevels` (*self* `<gtk-ui-manager>`) (*types* `<gtk-ui-manager-item-type>`) ⇒ (*ret* `gslist-of`) [Function]
`get-toplevels` [Method]

Obtains a list of all toplevel widgets of the requested types.

self : a `<gtk-ui-manager>`

types specifies the types of toplevel widgets to include. Allowed types are `<gtk-ui-manager-menubar>`, `<gtk-ui-manager-toolbar>` and `<gtk-ui-manager-popup>`.

ret a newly-allocated of all toplevel widgets of the requested types.

Since 2.4

`gtk-ui-manager-get-action` (*self* `<gtk-ui-manager>`) (*path* `mchars`) ⇒ (*ret* `<gtk-action>`) [Function]

`get-action` [Method]

Looks up an action by following a path. See `gtk-ui-manager-get-widget` for more information about paths.

self a `<gtk-ui-manager>`

path a path

ret the action whose proxy widget is found by following the path, or '`#f`' if no widget was found.

Since 2.4

`gtk-ui-manager-add-ui-from-string` (*self* `<gtk-ui-manager>`) (*buffer* `mchars`) (*length* `ssize_t`) ⇒ (*ret* `unsigned-int`) [Function]

`add-ui-from-string` [Method]

Parses a string containing a UI definition and merges it with the current contents of *self*. An enclosing `<ui>` element is added if it is missing.

self a `<gtk-ui-manager>` object

buffer the string to parse

length the length of *buffer* (may be -1 if *buffer* is nul-terminated)

error return location for an error

ret The merge id for the merged UI. The merge id can be used to unmerge the UI with `gtk-ui-manager-remove-ui`. If an error occurred, the return value is 0.

Since 2.4

`gtk-ui-manager-add-ui-from-file` (*self* `<gtk-ui-manager>`) (*filename* `mchars`) ⇒ (*ret* `unsigned-int`) [Function]

`add-ui-from-file` [Method]

Parses a file containing a UI definition and merges it with the current contents of *self*.

self a `<gtk-ui-manager>` object

filename the name of the file to parse

error return location for an error

ret The merge id for the merged UI. The merge id can be used to unmerge the UI with `gtk-ui-manager-remove-ui`. If an error occurred, the return value is 0.

Since 2.4

`gtk-ui-manager-new-merge-id` (*self* `<gtk-ui-manager>`) [Function]
⇒ (*ret* `unsigned-int`)
`new-merge-id` [Method]

Returns an unused merge id, suitable for use with `gtk-ui-manager-add-ui`.

self a `<gtk-ui-manager>`

ret an unused merge id.

Since 2.4

`gtk-ui-manager-add-ui` (*self* `<gtk-ui-manager>`) [Function]
(*merge_id* `unsigned-int`) (*path* `mchars`) (*name* `mchars`) (*action* `mchars`)
(*type* `<gtk-ui-manager-item-type>`) (*top* `bool`)
`add-ui` [Method]

Adds a UI element to the current contents of *self*.

If *type* is '`GTK_UI_MANAGER_AUTO`', GTK+ inserts a menuitem, toolitem or separator if such an element can be inserted at the place determined by *path*. Otherwise *type* must indicate an element that can be inserted at the place determined by *path*.

If *path* points to a menuitem or toolitem, the new element will be inserted before or after this item, depending on *top*.

self a `<gtk-ui-manager>`

merge-id the merge id for the merged UI, see `gtk-ui-manager-new-merge-id`

path a path

name the name for the added UI element

action the name of the action to be proxied, or '`#f`' to add a separator

type the type of UI element to add.

top if '`#t`', the UI element is added before its siblings, otherwise it is added after its siblings.

Since 2.4

`gtk-ui-manager-remove-ui` (*self* `<gtk-ui-manager>`) [Function]
(*merge_id* `unsigned-int`)
`remove-ui` [Method]

Unmerges the part of *self*'s content identified by *merge-id*.

self a `<gtk-ui-manager>` object

merge-id a merge id as returned by `gtk-ui-manager-add-ui-from-string`

Since 2.4

`gtk-ui-manager-get-ui` (*self* `<gtk-ui-manager>`) ⇒ (*ret* `mchars`) [Function]
`get-ui` [Method]

Creates a UI definition of the merged UI.

self a `<gtk-ui-manager>`

ret A newly allocated string containing an XML representation of the merged UI.

Since 2.4

`gtk-ui-manager-ensure-update` (*self* `<gtk-ui-manager>`) [Function]
`ensure-update` [Method]

Makes sure that all pending updates to the UI have been completed.

This may occasionally be necessary, since `<gtk-ui-manager>` updates the UI in an idle function. A typical example where this function is useful is to enforce that the menubar and toolbar have been added to the main window before showing it:

```
gtk_container_add (GTK_CONTAINER (window), vbox);
g_signal_connect (merge, "add_widget",
                  G_CALLBACK (add_widget), vbox);
gtk_ui_manager_add_ui_from_file (merge, "my-menus");
gtk_ui_manager_add_ui_from_file (merge, "my-toolbars");
gtk_ui_manager_ensure_update (merge);
gtk_widget_show (window);
```

self a `<gtk-ui-manager>`

Since 2.4

73 GtkActionGroup

A group of actions

73.1 Overview

Actions are organised into groups. An action group is essentially a map from names to <gtk-action> objects.

All actions that would make sense to use in a particular context should be in a single group. Multiple action groups may be used for a particular user interface. In fact, it is expected that most nontrivial applications will make use of multiple groups. For example, in an application that can edit multiple documents, one group holding global actions (e.g. quit, about, new), and one group per document holding actions that act on that document (eg. save, cut/copy/paste, etc). Each window's menus would be constructed from a combination of two action groups.

Accelerators are handled by the GTK+ accelerator map. All actions are assigned an accelerator path (which normally has the form '<Actions>//') and a shortcut is associated with this accelerator path. All menuitems and toolitems take on this accelerator path. The GTK+ accelerator map code makes sure that the correct shortcut is displayed next to the menu item.

73.2 Usage

<gtk-action-group> [Class]

This <gobject> class defines the following properties:

`name` A name for the action group.

`sensitive`
Whether the action group is enabled.

`visible` Whether the action group is visible.

`connect-proxy` (*arg0* <gtk-action>) [Signal on <gtk-action-group>]
(*arg1* <gtk-widget>)

The connect_proxy signal is emitted after connecting a proxy to an action in the group. Note that the proxy may have been connected to a different action before.

This is intended for simple customizations for which a custom action class would be too clumsy, e.g. showing tooltips for menuitems in the statusbar.

<gtk-ui-manager> proxies the signal and provides global notification just before any action is connected to a proxy, which is probably more convenient to use.

Since 2.4

`disconnect-proxy` (*arg0* <gtk-action>) [Signal on <gtk-action-group>]
(*arg1* <gtk-widget>)

The disconnect_proxy signal is emitted after disconnecting a proxy from an action in the group.

<gtk-ui-manager> proxies the signal and provides global notification just before any action is connected to a proxy, which is probably more convenient to use.

Since 2.4

`pre-activate` (*arg0* `<gtk-action>`) [Signal on `<gtk-action-group>`]
The pre_activate signal is emitted just before the *action* in the *action-group* is activated

This is intended for `<gtk-ui-manager>` to proxy the signal and provide global notification just before any action is activated.

Since 2.4

`post-activate` (*arg0* `<gtk-action>`) [Signal on `<gtk-action-group>`]
The post_activate signal is emitted just after the *action* in the *action-group* is activated

This is intended for `<gtk-ui-manager>` to proxy the signal and provide global notification just after any action is activated.

Since 2.4

`gtk-action-group-new` (*name* `mchars`) ⇒ (*ret* `<gtk-action-group>`) [Function]
Creates a new `<gtk-action-group>` object. The name of the action group is used when associating keybindings with the actions.

name the name of the action group.

ret the new `<gtk-action-group>`

Since 2.4

`gtk-action-group-get-name` (*self* `<gtk-action-group>`) [Function]
⇒ (*ret* `mchars`)
`get-name` [Method]
Gets the name of the action group.

action-group
the action group

ret the name of the action group.

Since 2.4

`gtk-action-group-get-sensitive` (*self* `<gtk-action-group>`) [Function]
⇒ (*ret* `bool`)
`get-sensitive` [Method]
Returns '`#t`' if the group is sensitive. The constituent actions can only be logically sensitive (see `gtk-action-is-sensitive`) if they are sensitive (see `gtk-action-get-sensitive`) and their group is sensitive.

action-group
the action group

ret '`#t`' if the group is sensitive.

Since 2.4

`gtk-action-group-set-sensitive` (*self* `<gtk-action-group>`) [Function]
(*sensitive* `bool`)
`set-sensitive` [Method]
Changes the sensitivity of *action-group*

action-group
: the action group

sensitive
: new sensitivity

Since 2.4

`gtk-action-group-get-visible` (*self* `<gtk-action-group>`) ⇒ (*ret* `bool`) [Function]

`get-visible` [Method]

Returns '`#t`' if the group is visible. The constituent actions can only be logically visible (see `gtk-action-is-visible`) if they are visible (see `gtk-action-get-visible`) and their group is visible.

action-group
: the action group

ret
: '`#t`' if the group is visible.

Since 2.4

`gtk-action-group-set-visible` (*self* `<gtk-action-group>`) (*visible* `bool`) [Function]

`set-visible` [Method]

Changes the visible of *action-group*.

action-group
: the action group

visible
: new visiblity

Since 2.4

`gtk-action-group-get-action` (*self* `<gtk-action-group>`) (*action_name* `mchars`) ⇒ (*ret* `<gtk-action>`) [Function]

`get-action` [Method]

Looks up an action in the action group by name.

action-group
: the action group

action-name
: the name of the action

ret
: the action, or '`#f`' if no action by that name exists

Since 2.4

`gtk-action-group-list-actions` (*self* `<gtk-action-group>`) ⇒ (*ret* `glist-of`) [Function]

`list-actions` [Method]

Lists the actions in the action group.

action-group
: the action group

ret an allocated list of the action objects in the action group

Since 2.4

`gtk-action-group-add-action` (*self* `<gtk-action-group>`) [Function]
(*action* `<gtk-action>`)
`add-action` [Method]

Adds an action object to the action group. Note that this function does not set up the accel path of the action, which can lead to problems if a user tries to modify the accelerator of a menuitem associated with the action. Therefore you must either set the accel path yourself with `gtk-action-set-accel-path`, or use '`gtk_action_group_add_action_with_accel (..., NULL)`'.

action-group
the action group

action an action

Since 2.4

`gtk-action-group-remove-action` (*self* `<gtk-action-group>`) [Function]
(*action* `<gtk-action>`)
`remove-action` [Method]

Removes an action object from the action group.

action-group
the action group

action an action

Since 2.4

`gtk-action-group-add-actions` (*self* `<gtk-action-group>`) [Function]
(*entries* `scm`)
`add-actions` [Method]

This is a convenience function to create a number of actions and add them to the action group.

The "activate" signals of the actions are connected to the callbacks and their accel paths are set to '`<Actions>//`'.

action-group
the action group

entries an array of action descriptions

n-entries the number of entries

user-data data to pass to the action callbacks

Since 2.4

`gtk-action-group-add-actions-full` (*self* `<gtk-action-group>`) [Function]
(*entries* `<gtk-action-entry*>`) (*n_entries* `unsigned-int`)
(*user_data* `<gpointer>`) (*destroy* `<g-destroy-notify>`)

`add-actions-full` [Method]

This variant of `gtk-action-group-add-actions` adds a `<g-destroy-notify>` callback for *user-data*.

action-group the action group

entries an array of action descriptions

n-entries the number of entries

user-data data to pass to the action callbacks

destroy destroy notification callback for *user-data*

Since 2.4

`gtk-action-group-add-toggle-actions` (*self* `<gtk-action-group>`) [Function]
(*entries* `scm`)

`add-toggle-actions` [Method]

This is a convenience function to create a number of toggle actions and add them to the action group.

The "activate" signals of the actions are connected to the callbacks and their accel paths are set to '`<Actions>//`'.

action-group the action group

entries an array of toggle action descriptions

n-entries the number of entries

user-data data to pass to the action callbacks

Since 2.4

`gtk-action-group-add-radio-actions` (*self* `<gtk-action-group>`) [Function]
(*entries* `scm`) (*value* `int`) (*on_change* `scm`)

`add-radio-actions` [Method]

This is a convenience routine to create a group of radio actions and add them to the action group.

The "changed" signal of the first radio action is connected to the *on-change* callback and the accel paths of the actions are set to '`<Actions>//`'.

action-group the action group

entries an array of radio action descriptions

n-entries the number of entries

value the value of the action to activate initially, or -1 if no action should be activated

on-change the callback to connect to the changed signal

user-data data to pass to the action callbacks

Since 2.4

`gtk-action-group-set-translate-func` (*self* `<gtk-action-group>`) [Function]
(*func* `<gtk-translate-func>`) (*data* `<gpointer>`)
(*notify* `<gtk-destroy-notify>`)

`set-translate-func` [Method]

Sets a function to be used for translating the *label* and *tooltip* of `<gtk-action-group-entry>`s added by `gtk-action-group-add-actions`.

If you're using `gettext`, it is enough to set the translation domain with `gtk-action-group-set-translation-domain`.

action-group
: a `<gtk-action-group>`

func
: a `<gtk-translate-func>`

data
: data to be passed to *func* and *notify*

notify
: a `<gtk-destroy-notify>` function to be called when *action-group* is destroyed and when the translation function is changed again

Since 2.4

`gtk-action-group-translate-string` (*self* `<gtk-action-group>`) [Function]
(*string* `mchars`) ⇒ (*ret* `mchars`)

`translate-string` [Method]

Translates a string using the specified `translate-func`. This is mainly intended for language bindings.

action-group
: a `<gtk-action-group>`

string
: a string

ret
: the translation of *string*

Since 2.6

74 GtkAction

An action which can be triggered by a menu or toolbar item

74.1 Overview

Actions represent operations that the user can be perform, along with some information how it should be presented in the interface. Each action provides methods to create icons, menu items and toolbar items representing itself.

As well as the callback that is called when the action gets activated, the following also gets associated with the action: The action will also have some state information:

a name (not translated, for path lookup)

a label (translated, for display)

an accelerator

whether label indicates a stock id

a tooltip (optional, translated)

a toolbar label (optional, shorter than label)

visible (shown/hidden)

sensitive (enabled/disabled)

Apart from regular actions, there are toggle actions, which can be toggled between two states and radio actions, of which only one in a group can be in the "active" state. Other actions can be implemented as `<gtk-action>` subclasses.

Each action can have one or more proxy menu item, toolbar button or other proxy widgets. Proxies mirror the state of the action (text label, tooltip, icon, visible, sensitive, etc), and should change when the action's state changes. When the proxy is activated, it should activate its action.

74.2 Usage

`<gtk-action>` [Class]

This `<gobject>` class defines the following properties:

`name` A unique name for the action.

`label` The label used for menu items and buttons that activate this action.

`short-label`
A shorter label that may be used on toolbar buttons.

`tooltip` A tooltip for this action.

`stock-id` The stock icon displayed in widgets representing this action.

`icon-name`
The name of the icon from the icon theme

`visible-horizontal`
Whether the toolbar item is visible when the toolbar is in a horizontal orientation.

`visible-vertical`
: Whether the toolbar item is visible when the toolbar is in a vertical orientation.

`visible-overflown`
: When TRUE, toolitem proxies for this action are represented in the toolbar overflow menu.

`is-important`
: Whether the action is considered important. When TRUE, toolitem proxies for this action show text in GTK_TOOLBAR_BOTH_HORIZ mode.

`hide-if-empty`
: When TRUE, empty menu proxies for this action are hidden.

`sensitive`
: Whether the action is enabled.

`visible`
: Whether the action is visible.

`action-group`
: The GtkActionGroup this GtkAction is associated with, or NULL (for internal use).

`activate` [Signal on `<gtk-action>`]

The "activate" signal is emitted when the action is activated.

Since 2.4

`gtk-action-new` (*name* `mchars`) (*label* `mchars`) (*tooltip* `mchars`) (*stock_id* `mchars`) ⇒ (*ret* `<gtk-action>`) [Function]

Creates a new `<gtk-action>` object. To add the action to a `<gtk-action-group>` and set the accelerator for the action, call `gtk-action-group-add-action-with-accel`. See *(the missing figure, XML-UI* for information on allowed action names.

name
: A unique name for the action

label
: the label displayed in menu items and on buttons

tooltip
: a tooltip for the action

stock-id
: the stock icon to display in widgets representing the action

ret
: a new `<gtk-action>`

Since 2.4

`gtk-action-get-name` (*self* `<gtk-action>`) ⇒ (*ret* `mchars`) [Function]
`get-name` [Method]

Returns the name of the action.

action
: the action object

ret
: the name of the action. The string belongs to GTK+ and should not be freed.

Since 2.4

`gtk-action-is-sensitive` (*self* `<gtk-action>`) ⇒ (*ret* `bool`) [Function]
`is-sensitive` [Method]

Returns whether the action is effectively sensitive.

action the action object

ret '`#t`' if the action and its associated action group are both sensitive.

Since 2.4

`gtk-action-get-sensitive` (*self* `<gtk-action>`) ⇒ (*ret* `bool`) [Function]
`get-sensitive` [Method]

Returns whether the action itself is sensitive. Note that this doesn't necessarily mean effective sensitivity. See `gtk-action-is-sensitive` for that.

action the action object

ret '`#t`' if the action itself is sensitive.

Since 2.4

`gtk-action-set-sensitive` (*self* `<gtk-action>`) (*sensitive* `bool`) [Function]
`set-sensitive` [Method]

Sets the ::sensitive property of the action to *sensitive*. Note that this doesn't necessarily mean effective sensitivity. See `gtk-action-is-sensitive` for that.

action the action object

sensitive '`#t`' to make the action sensitive

Since 2.6

`gtk-action-is-visible` (*self* `<gtk-action>`) ⇒ (*ret* `bool`) [Function]
`is-visible` [Method]

Returns whether the action is effectively visible.

action the action object

ret '`#t`' if the action and its associated action group are both visible.

Since 2.4

`gtk-action-get-visible` (*self* `<gtk-action>`) ⇒ (*ret* `bool`) [Function]
`get-visible` [Method]

Returns whether the action itself is visible. Note that this doesn't necessarily mean effective visibility. See `gtk-action-is-sensitive` for that.

action the action object

ret '`#t`' if the action itself is visible.

Since 2.4

`gtk-action-set-visible` (*self* `<gtk-action>`) (*visible* `bool`) [Function]
`set-visible` [Method]

Sets the ::visible property of the action to *visible*. Note that this doesn't necessarily mean effective visibility. See `gtk-action-is-visible` for that.

action the action object

visible '`#t`' to make the action visible

Since 2.6

`gtk-action-activate` (*self* `<gtk-action>`) [Function]
`activate` [Method]

Emits the "activate" signal on the specified action, if it isn't insensitive. This gets called by the proxy widgets when they get activated.

It can also be used to manually activate an action.

action the action object

Since 2.4

`gtk-action-create-icon` (*self* `<gtk-action>`) [Function]
(*icon_size* `<gtk-icon-size>`) ⇒ (*ret* `<gtk-widget>`)
`create-icon` [Method]

This function is intended for use by action implementations to create icons displayed in the proxy widgets.

action the action object

icon-size the size of the icon that should be created.

ret a widget that displays the icon for this action.

Since 2.4

`gtk-action-create-menu-item` (*self* `<gtk-action>`) [Function]
⇒ (*ret* `<gtk-widget>`)
`create-menu-item` [Method]

Creates a menu item widget that proxies for the given action.

action the action object

ret a menu item connected to the action.

Since 2.4

`gtk-action-create-tool-item` (*self* `<gtk-action>`) [Function]
⇒ (*ret* `<gtk-widget>`)
`create-tool-item` [Method]

Creates a toolbar item widget that proxies for the given action.

action the action object

ret a toolbar item connected to the action.

Since 2.4

`gtk-action-connect-proxy` (*self* `<gtk-action>`) [Function]
(*proxy* `<gtk-widget>`)

`connect-proxy` [Method]

Connects a widget to an action object as a proxy. Synchronises various properties of the action with the widget (such as label text, icon, tooltip, etc), and attaches a callback so that the action gets activated when the proxy widget does.

If the widget is already connected to an action, it is disconnected first.

action the action object

proxy the proxy widget

Since 2.4

`gtk-action-disconnect-proxy` (*self* `<gtk-action>`) [Function]
(*proxy* `<gtk-widget>`)
`disconnect-proxy` [Method]

Disconnects a proxy widget from an action. Does *not* destroy the widget, however.

action the action object

proxy the proxy widget

Since 2.4

`gtk-action-get-proxies` (*self* `<gtk-action>`) ⇒ (*ret* `gslist-of`) [Function]
`get-proxies` [Method]

Returns the proxy widgets for an action. See also `gtk-widget-get-action`.

action the action object

ret a `<gs-list>` of proxy widgets. The list is owned by GTK+ and must not be modified.

Since 2.4

`gtk-action-connect-accelerator` (*self* `<gtk-action>`) [Function]
`connect-accelerator` [Method]

Installs the accelerator for *action* if *action* has an accel path and group. See `gtk-action-set-accel-path` and `gtk-action-set-accel-group`

Since multiple proxies may independently trigger the installation of the accelerator, the *action* counts the number of times this function has been called and doesn't remove the accelerator until `gtk-action-disconnect-accelerator` has been called as many times.

action a `<gtk-action>`

Since 2.4

`gtk-action-disconnect-accelerator` (*self* `<gtk-action>`) [Function]
`disconnect-accelerator` [Method]

Undoes the effect of one call to `gtk-action-connect-accelerator`.

action a `<gtk-action>`

Since 2.4

`gtk-action-block-activate-from` (*self* `<gtk-action>`) [Function]
(*proxy* `<gtk-widget>`)
`block-activate-from` [Method]

Disables calls to the `gtk-action-activate` function by signals on the given proxy widget. This is used to break notification loops for things like check or radio actions.

This function is intended for use by action implementations.

action the action object

proxy a proxy widget

Since 2.4

`gtk-action-unblock-activate-from` (*self* `<gtk-action>`) [Function]
(*proxy* `<gtk-widget>`)
`unblock-activate-from` [Method]

Re-enables calls to the `gtk-action-activate` function by signals on the given proxy widget. This undoes the blocking done by `gtk-action-block-activate-from`.

This function is intended for use by action implementations.

action the action object

proxy a proxy widget

Since 2.4

`gtk-action-get-accel-path` (*self* `<gtk-action>`) ⇒ (*ret* `mchars`) [Function]
`get-accel-path` [Method]

Returns the accel path for this action.

action the action object

ret the accel path for this action, or '`#f`' if none is set. The returned string is owned by GTK+ and must not be freed or modified.

Since 2.6

`gtk-action-set-accel-path` (*self* `<gtk-action>`) [Function]
(*accel_path* `mchars`)
`set-accel-path` [Method]

Sets the accel path for this action. All proxy widgets associated with the action will have this accel path, so that their accelerators are consistent.

action the action object

accel-path the accelerator path

Since 2.4

`gtk-action-get-accel-closure` (*self* `<gtk-action>`) [Function]
⇒ (*ret* `<gclosure>`)
`get-accel-closure` [Method]

Returns the accel closure for this action.

action the action object

ret the accel closure for this action. The returned closure is owned by GTK+ and must not be unreffed or modified.

Since 2.8

`gtk-action-set-accel-group` (*self* `<gtk-action>`) [Function]
(*accel_group* `<gtk-accel-group>`)
`set-accel-group` [Method]

Sets the `<gtk-accel-group>` in which the accelerator for this action will be installed.

action the action object

accel-group
a `<gtk-accel-group>` or '`#f`'

Since 2.4

75 GtkToggleAction

An action which can be toggled between two states

75.1 Overview

A <gtk-toggle-action> corresponds roughly to a <gtk-check-menu-item>. It has an "active" state specifying whether the action has been checked or not.

75.2 Usage

<gtk-toggle-action> [Class]

This <gobject> class defines the following properties:

draw-as-radio
: Whether the proxies for this action look like radio action proxies

active
: If the toggle action should be active in or not

toggled [Signal on <gtk-toggle-action>]

gtk-toggle-action-new (*name* mchars) (*label* mchars) [Function]
(*tooltip* mchars) (*stock_id* mchars) ⇒ (*ret* <gtk-toggle-action>)

Creates a new <gtk-toggle-action> object. To add the action to a <gtk-action-group> and set the accelerator for the action, call gtk-action-group-add-action-with-accel.

name
: A unique name for the action

label
: The label displayed in menu items and on buttons

tooltip
: A tooltip for the action

stock-id
: The stock icon to display in widgets representing the action

ret
: a new <gtk-toggle-action>

Since 2.4

gtk-toggle-action-toggled (*self* <gtk-toggle-action>) [Function]
toggled [Method]

Emits the "toggled" signal on the toggle action.

action
: the action object

Since 2.4

gtk-toggle-action-set-active (*self* <gtk-toggle-action>) [Function]
(*is_active* bool)
set-active [Method]

Sets the checked state on the toggle action.

action
: the action object

is-active
: whether the action should be checked or not

Since 2.4

`gtk-toggle-action-get-active` (*self* `<gtk-toggle-action>`) ⇒ (*ret* `bool`) [Function]

`get-active` [Method]

Returns the checked state of the toggle action.

action the action object

ret the checked state of the toggle action

Since 2.4

`gtk-toggle-action-set-draw-as-radio` (*self* `<gtk-toggle-action>`) (*draw_as_radio* `bool`) [Function]

`set-draw-as-radio` [Method]

Sets whether the action should have proxies like a radio action.

action the action object

draw-as-radio whether the action should have proxies like a radio action

Since 2.4

`gtk-toggle-action-get-draw-as-radio` (*self* `<gtk-toggle-action>`) ⇒ (*ret* `bool`) [Function]

`get-draw-as-radio` [Method]

Returns whether the action should have proxies like a radio action.

action the action object

ret whether the action should have proxies like a radio action.

Since 2.4

76 GtkRadioAction

An action of which only one in a group can be active

76.1 Overview

A <gtk-radio-action> is similar to <gtk-radio-menu-item>. A number of radio actions can be linked together so that only one may be active at any one time.

76.2 Usage

<gtk-radio-action> [Class]

This <gobject> class defines the following properties:

value The value returned by gtk_radio_action_get_current_value() when this action is the current action of its group.

group The radio action whose group this action belongs to.

current-value
The value property of the currently active member of the group to which this action belongs.

changed (*arg0* <gtk-radio-action>) [Signal on <gtk-radio-action>]

The ::changed signal is emitted on every member of a radio group when the active member is changed. The signal gets emitted after the ::activate signals for the previous and current active members.

Since 2.4

gtk-radio-action-new (*name* mchars) (*label* mchars) (*tooltip* mchars) (*stock_id* mchars) (*value* int) ⇒ (*ret* <gtk-radio-action>) [Function]

Creates a new <gtk-radio-action> object. To add the action to a <gtk-action-group> and set the accelerator for the action, call gtk-action-group-add-action-with-accel.

name A unique name for the action

label The label displayed in menu items and on buttons

tooltip A tooltip for this action

stock-id The stock icon to display in widgets representing this action

value The value which gtk-radio-action-get-current-value should return if this action is selected.

ret a new <gtk-radio-action>

Since 2.4

gtk-radio-action-get-group (*self* <gtk-radio-action>) ⇒ (*ret* <gtk-radio-group*>) [Function]

`get-group` [Method]

Returns the list representing the radio group for this object. Note that the returned list is only valid until the next change to the group.

A common way to set up a group of radio group is the following:

```
GSList *group = NULL;
GtkRadioAction *action;

while (/* more actions to add */)
  {
     action = gtk_radio_action_new (...);

     gtk_radio_action_set_group (action, group);
     group = gtk_radio_action_get_group (action);
  }
```

action the action object

ret the list representing the radio group for this object

Since 2.4

`gtk-radio-action-set-group` (*self* `<gtk-radio-action>`) [Function]
(*group* `<gtk-radio-group*>`)

`set-group` [Method]

Sets the radio group for the radio action object.

action the action object

group a list representing a radio group

Since 2.4

`gtk-radio-action-get-current-value` (*self* `<gtk-radio-action>`) [Function]
⇒ (*ret* `int`)

`get-current-value` [Method]

Obtains the value property of the currently active member of the group to which *action* belongs.

action a `<gtk-radio-action>`

ret The value of the currently active group member

Since 2.4

`gtk-radio-action-set-current-value` (*self* `<gtk-radio-action>`) [Function]
(*current_value* `int`)

`set-current-value` [Method]

Sets the currently active group member to the member with value property *current-value*.

action a `<gtk-radio-action>`

current-value
: the new value

Since 2.10

77 GtkColorButton

A button to launch a color selection dialog

77.1 Overview

The <gtk-color-button> is a button which displays the currently selected color an allows to open a color selection dialog to change the color. It is suitable widget for selecting a color in a preference dialog.

77.2 Usage

<gtk-color-button> [Class]

This <gobject> class defines the following properties:

use-alpha
: Whether or not to give the color an alpha value

title
: The title of the color selection dialog

color
: The selected color

alpha
: The selected opacity value (0 fully transparent, 65535 fully opaque)

color-set [Signal on <gtk-color-button>]

The ::color-set signal is emitted when the user selects a color. When handling this signal, use gtk-color-button-get-color and gtk-color-button-get-alpha to find out which color was just selected.

Note that this signal is only emitted when the *user* changes the color. If you need to react to programmatic color changes as well, use the notify::color signal.

Since 2.4

gtk-color-button-new ⇒ (*ret* <gtk-widget>) [Function]

Creates a new color button. This returns a widget in the form of a small button containing a swatch representing the current selected color. When the button is clicked, a color-selection dialog will open, allowing the user to select a color. The swatch will be updated to reflect the new color when the user finishes.

ret
: a new color button.

Since 2.4

gtk-color-button-new-with-color (*color* <gdk-color>) [Function]
⇒ (*ret* <gtk-widget>)

Creates a new color button.

color
: A <gdk-color> to set the current color with.

ret
: a new color button.

Since 2.4

`gtk-color-button-set-color` (*self* `<gtk-color-button>`) [Function]
(*color* `<gdk-color>`)
`set-color` [Method]

Sets the current color to be *color*.

color-button
a `<gtk-color-button>`.

color A `<gdk-color>` to set the current color with.

Since 2.4

`gtk-color-button-get-color` (*self* `<gtk-color-button>`) [Function]
(*color* `<gdk-color>`)
`get-color` [Method]

Sets *color* to be the current color in the `<gtk-color-button>` widget.

color-button
a `<gtk-color-button>`.

color a `<gdk-color>` to fill in with the current color.

Since 2.4

`gtk-color-button-set-alpha` (*self* `<gtk-color-button>`) [Function]
(*alpha* `unsigned-int16`)
`set-alpha` [Method]

Sets the current opacity to be *alpha*.

color-button
a `<gtk-color-button>`.

alpha an integer between 0 and 65535.

Since 2.4

`gtk-color-button-get-alpha` (*self* `<gtk-color-button>`) [Function]
⇒ (*ret* `unsigned-int16`)
`get-alpha` [Method]

Returns the current alpha value.

color-button
a `<gtk-color-button>`.

ret an integer between 0 and 65535.

Since 2.4

`gtk-color-button-set-use-alpha` (*self* `<gtk-color-button>`) [Function]
(*use_alpha* `bool`)
`set-use-alpha` [Method]

Sets whether or not the color button should use the alpha channel.

color-button
a `<gtk-color-button>`.

use-alpha '`#t`' if color button should use alpha channel, '`#f`' if not.

Since 2.4

`gtk-color-button-get-use-alpha` (*self* `<gtk-color-button>`) [Function]
⇒ (*ret* `bool`)

`get-use-alpha` [Method]

Does the color selection dialog use the alpha channel?

color-button
a `<gtk-color-button>`.

ret '`#t`' if the color sample uses alpha channel, '`#f`' if not.

Since 2.4

`gtk-color-button-set-title` (*self* `<gtk-color-button>`) [Function]
(*title* `mchars`)

`set-title` [Method]

Sets the title for the color selection dialog.

color-button
a `<gtk-color-button>`

title String containing new window title.

Since 2.4

`gtk-color-button-get-title` (*self* `<gtk-color-button>`) [Function]
⇒ (*ret* `mchars`)

`get-title` [Method]

Gets the title of the color selection dialog.

color-button
a `<gtk-color-button>`

ret An internal string, do not free the return value

Since 2.4

78 GtkColorSelection

A widget used to select a color

78.1 Overview

The <gtk-color-selection> is a widget that is used to select a color. It consists of a color wheel and number of sliders and entry boxes for color parameters such as hue, saturation, value, red, green, blue, and opacity. It is found on the standard color selection dialog box <gtk-color-selection-dialog>.

78.2 Usage

<gtk-color-selection> [Class]

This <gobject> class defines the following properties:

has-palette
: Whether a palette should be used

has-opacity-control
: Whether the color selector should allow setting opacity

current-color
: The current color

current-alpha
: The current opacity value (0 fully transparent, 65535 fully opaque)

color-changed [Signal on <gtk-color-selection>]

This signal is emitted when the color changes in the <gtk-color-selection> according to its update policy.

gtk-color-selection-new ⇒ (*ret* <gtk-widget>) [Function]

Creates a new GtkColorSelection.

ret a new <gtk-color-selection>

gtk-color-selection-set-has-palette [Function]
(*self* <gtk-color-selection>) (*has_palette* bool)

set-has-palette [Method]

Shows and hides the palette based upon the value of *has-palette*.

colorsel a <gtk-color-selection>.

has-palette
: '#t' if palette is to be visible, '#f' otherwise.

gtk-color-selection-get-has-palette [Function]
(*self* <gtk-color-selection>) ⇒ (*ret* bool)

get-has-palette [Method]

Determines whether the color selector has a color palette.

colorsel a <gtk-color-selection>.

ret '#t' if the selector has a palette. '#f' if it hasn't.

`gtk-color-selection-is-adjusting` (*self* `<gtk-color-selection>`) [Function]
⇒ (*ret* `bool`)

`is-adjusting` [Method]

Gets the current state of the *colorsel*.

colorsel a `<gtk-color-selection>`.

ret '`#t`' if the user is currently dragging a color around, and '`#f`' if the selection has stopped.

79 GtkColorSelectionDialog

A standard dialog box for selecting a color

79.1 Overview

The `<gtk-color-selection-dialog>` provides a standard dialog which allows the user to select a color much like the `<gtk-file-selection>` provides a standard dialog for file selection.

79.2 Usage

`<gtk-color-selection-dialog>` [Class]

This `<gobject>` class defines no properties, other than those defined by its superclasses.

`gtk-color-selection-dialog-new` (*title* `mchars`) [Function]
⇒ (*ret* `<gtk-widget>`)

Creates a new `<gtk-color-selection-dialog>`.

title a string containing the title text for the dialog.

ret a `<gtk-color-selection-dialog>`.

80 GtkFileSelection

Prompt the user for a file or directory name

80.1 Overview

`<gtk-file-selection>` should be used to retrieve file or directory names from the user. It will create a new dialog window containing a directory list, and a file list corresponding to the current working directory. The filesystem can be navigated using the directory list or the drop-down history menu. Alternatively, the TAB key can be used to navigate using filename completion - common in text based editors such as emacs and jed.

File selection dialogs are created with a call to `gtk-file-selection-new`.

The default filename can be set using `gtk-file-selection-set-filename` and the selected filename retrieved using `gtk-file-selection-get-filename`.

Use `gtk-file-selection-complete` to display files and directories that match a given pattern. This can be used for example, to show only *.txt files, or only files beginning with gtk*.

Simple file operations; create directory, delete file, and rename file, are available from buttons at the top of the dialog. These can be hidden using `gtk-file-selection-hide-fileop-buttons` and shown again using `gtk-file-selection-show-fileop-buttons`.

```
/* The file selection widget and the string to store the chosen filename */

void store_filename (GtkWidget *widget, gpointer user_data) {
   GtkWidget *file_selector = GTK_WIDGET (user_data);
   const gchar *selected_filename;

   selected_filename = gtk_file_selection_get_filename (GTK_FILE_SELECTION (file_selec
   g_print ("Selected filename: %s\n", selected_filename);
}

void create_file_selection (void) {

   GtkWidget *file_selector;

   /* Create the selector */

   file_selector = gtk_file_selection_new ("Please select a file for editing.");

   g_signal_connect (GTK_FILE_SELECTION (file_selector)->ok_button,
                     "clicked",
                     G_CALLBACK (store_filename),
                     file_selector);

   /* Ensure that the dialog box is destroyed when the user clicks a button. */
```

```
    g_signal_connect_swapped (GTK_FILE_SELECTION (file_selector)->ok_button,
                              "clicked",
                              G_CALLBACK (gtk_widget_destroy),
                              file_selector);

    g_signal_connect_swapped (GTK_FILE_SELECTION (file_selector)->cancel_button,
                              "clicked",
                              G_CALLBACK (gtk_widget_destroy),
                              file_selector);

    /* Display that dialog */

    gtk_widget_show (file_selector);
}
```

80.2 Usage

`<gtk-file-selection>` [Class]

This `<gobject>` class defines the following properties:

`show-fileops`
: Whether buttons for creating/manipulating files should be displayed

`filename`
: The currently selected filename

`select-multiple`
: Whether to allow multiple files to be selected

`gtk-file-selection-new` (*title* `mchars`) ⇒ (*ret* `<gtk-widget>`) [Function]

Creates a new file selection dialog box. By default it will contain a `<gtk-tree-view>` of the application's current working directory, and a file listing. Operation buttons that allow the user to create a directory, delete files and rename files, are also present.

title
: a message that will be placed in the file requestor's titlebar.

ret
: the new file selection.

`gtk-file-selection-set-filename` (*self* `<gtk-file-selection>`) [Function]
(*filename* `mchars`)

`set-filename` [Method]

Sets a default path for the file requestor. If *filename* includes a directory path, then the requestor will open with that path as its current working directory.

This has the consequence that in order to open the requestor with a working directory and an empty filename, *filename* must have a trailing directory separator.

The encoding of *filename* is preferred GLib file name encoding, which may not be UTF-8. See `g-filename-from-utf8`.

filesel
: a `<gtk-file-selection>`.

filename
: a string to set as the default file name.

`gtk-file-selection-get-filename` (*self* `<gtk-file-selection>`) [Function]
⇒ (*ret* `mchars`)

`get-filename` [Method]

This function returns the selected filename in the GLib file name encoding. To convert to UTF-8, call `g-filename-to-utf8`. The returned string points to a statically allocated buffer and should be copied if you plan to keep it around.

If no file is selected then the selected directory path is returned.

filesel a `<gtk-file-selection>`

ret currently-selected filename in the on-disk encoding.

`gtk-file-selection-complete` (*self* `<gtk-file-selection>`) [Function]
(*pattern* `mchars`)

`complete` [Method]

Will attempt to match *pattern* to a valid filenames or subdirectories in the current directory. If a match can be made, the matched filename will appear in the text entry field of the file selection dialog. If a partial match can be made, the "Files" list will contain those file names which have been partially matched, and the "Folders" list those directories which have been partially matched.

filesel a `<gtk-file-selection>`.

pattern a string of characters which may or may not match any filenames in the current directory.

`gtk-file-selection-get-selections` (*self* `<gtk-file-selection>`) [Function]
⇒ (*ret* `<gchar**>`)

`get-selections` [Method]

Retrieves the list of file selections the user has made in the dialog box. This function is intended for use when the user can select multiple files in the file list.

The filenames are in the GLib file name encoding. To convert to UTF-8, call `g-filename-to-utf8` on each string.

filesel a `<gtk-file-selection>`

ret a newly-allocated '`#f`'-terminated array of strings. Use `g-strfreev` to free it.

81 GtkFileChooser

File chooser interface used by GtkFileChooserWidget and GtkFileChooserDialog

81.1 Overview

<gtk-file-chooser> is an interface that can be implemented by file selection widgets. In GTK+, the main objects that implement this interface are <gtk-file-chooser-widget>, <gtk-file-chooser-dialog>, and <gtk-file-chooser-button>. You do not need to write an object that implements the <gtk-file-chooser> interface unless you are trying to adapt an existing file selector to expose a standard programming interface.

<gtk-file-chooser> allows for shortcuts to various places in the filesystem. In the default implementation these are displayed in the left pane. It may be a bit confusing at first taht these shortcuts come from various sources and in various flavours, so lets explain the terminology here:

Shortcuts

Volumes

are created by the user, by dragging folders from the right pane to the left pane, or by using the "Add". Bookmarks can be renamed and deleted by the user.

can be provided by the application or by the underlying filesystem abstraction (e.g. both the gnome-vfs and the Windows filesystems provide "Desktop" shortcuts). Shortcuts cannot be modified by the user.

are provided by the underlying filesystem abstraction. They are the "roots" of the filesystem.

81.2 File Names and Encodings

When the user is finished selecting files in a <gtk-file-chooser>, your program can get the selected names either as filenames or as URIs. For URIs, the normal escaping rules are applied if the URI contains non-ASCII characters. However, filenames are *always* returned in the character set specified by the G_FILENAME_ENCODING environment variable. Please see the Glib documentation for more details about this variable.

This means that while you can pass the result of gtk-file-chooser-get-filename to open(2) or fopen(3), you may not be able to directly set it as the text of a <gtk-label> widget unless you convert it first to UTF-8, which all GTK+ widgets expect. You should use g-filename-to-utf8 to convert filenames into strings that can be passed to GTK+ widgets.

81.3 Adding a Preview Widget

You can add a custom preview widget to a file chooser and then get notification about when the preview needs to be updated. To install a preview widget, use gtk-file-chooser-set-preview-widget. Then, connect to the <gtk-file-chooser::update-preview> signal to get notified when you need to update the contents of the preview.

Your callback should use gtk-file-chooser-get-preview-filename to see what needs previewing. Once you have generated the preview for the corresponding file, you must

call `gtk-file-chooser-set-preview-widget-active` with a boolean flag that indicates whether your callback could successfully generate a preview.

```
{
  GtkImage *preview;

  ...

  preview = gtk_image_new ();

  gtk_file_chooser_set_preview_widget (my_file_chooser, preview);
  g_signal_connect (my_file_chooser, "update-preview",
    G_CALLBACK (update_preview_cb), preview);
}

static void
update_preview_cb (GtkFileChooser *file_chooser, gpointer data)
{
  GtkWidget *preview;
  char *filename;
  GdkPixbuf *pixbuf;
  gboolean have_preview;

  preview = GTK_WIDGET (data);
  filename = gtk_file_chooser_get_preview_filename (file_chooser);

  pixbuf = gdk_pixbuf_new_from_file_at_size (filename, 128, 128, NULL);
  have_preview = (pixbuf != NULL);
  g_free (filename);

  gtk_image_set_from_pixbuf (GTK_IMAGE (preview), pixbuf);
  if (pixbuf)
    gobject_unref (pixbuf);

  gtk_file_chooser_set_preview_widget_active (file_chooser, have_preview);
}
```

81.4 Adding Extra Widgets

You can add extra widgets to a file chooser to provide options that are not present in the default design. For example, you can add a toggle button to give the user the option to open a file in read-only mode. You can use `gtk-file-chooser-set-extra-widget` to insert additional widgets in a file chooser.

```
{
  GtkWidget *toggle;
```

```
  ...

  toggle = gtk_check_button_new_with_label ("Open file read-only");
  gtk_widget_show (toggle);
  gtk_file_chooser_set_extra_widget (my_file_chooser, toggle);
}
```

If you want to set more than one extra widget in the file chooser, you can a container such as a GtkVBox or a GtkTable and include your widgets in it. Then, set the container as the whole extra widget.

81.5 Key Bindings

Internally, GTK+ implements a file chooser's graphical user interface with the private . This widget has several key bindings and their associated signals. This section describes the available key binding signals.

The default keys that activate the key-binding signals in are as follows:

Both the individual key and the numeric keypad's "divide" key are supported.

Both the individual Up key and the numeric keypad's Up key are supported.

You can change these defaults to something else. For example, to add a modifier to a few of the default bindings, you can include the following fragment in your '`.gtkrc-2.0`' file:

```
binding "my-own-gtkfilechooser-bindings" {
bind "<Alt><Shift>Up" {
"up-folder" ()
}
bind "<Alt><Shift>Down" {
"down-folder" ()
}
bind "<Alt><Shift>Home" {
"home-folder" ()
}
}

class "GtkFileChooserDefault" binding "my-own-gtkfilechooser-bindings"
```

81.5.1 The "GtkFileChooserDefault::location-popup" signal

```
void user_function (GtkFileChooserDefault *chooser,
                    const char            *path,
                    gpointer user_data);
```

This is used to make the file chooser show a "Location" dialog which the user can use to manually type the name of the file he wishes to select. The *path* argument is a string that gets put in the text entry for the file name. By default this is bound to with a *path* string of "" (the empty string). It is also bound to with a *path* string of "'/'" (a slash): this lets you type and immediately type a path name. On Unix systems, this is bound to (tilde) with a *path* string of "~" itself for access to home directories.

chooser the object which received the signal.

path default contents for the text entry for the file name

user-data user data set when the signal handler was connected.

You can create your own bindings for the signal with custom *path* strings, and have a crude form of easily-to-type bookmarks. For example, say you access the path '/home/username/misc' very frequently. You could then create an shortcut by including the following in your '.gtkrc-2.0':

```
binding "misc-shortcut" {
bind "<Alt>M" {
"location-popup" ("/home/username/misc")
}
}

class "GtkFileChooserDefault" binding "misc-shortcut"
```

81.5.2 The "GtkFileChooserDefault::up-folder" signal

```
void user_function (GtkFileChooserDefault *chooser,
                    gpointer user_data);
```

This is used to make the file chooser go to the parent of the current folder in the file hierarchy. By default this is bound to and (the Up key in the numeric keypad also works).

chooser the object which received the signal.

user-data user data set when the signal handler was connected.

81.5.3 The "GtkFileChooserDefault::down-folder" signal

```
void user_function (GtkFileChooserDefault *chooser,
                    gpointer user_data);
```

This is used to make the file chooser go to a child of the current folder in the file hierarchy. The subfolder that will be used is displayed in the path bar widget of the file chooser. For example, if the path bar is showing "/foo/*bar*/baz", then this will cause the file chooser to switch to the "baz" subfolder. By default this is bound to (the Down key in the numeric keypad also works).

chooser the object which received the signal.

user-data user data set when the signal handler was connected.

81.5.4 The "GtkFileChooserDefault::home-folder" signal

```
void user_function (GtkFileChooserDefault *chooser,
                    gpointer user_data);
```

This is used to make the file chooser show the user's home folder in the file list. By default this is bound to (the Home key in the numeric keypad also works).

chooser the object which received the signal.

user-data user data set when the signal handler was connected.

81.5.5 The "GtkFileChooserDefault::desktop-folder" signal

```
void user_function (GtkFileChooserDefault *chooser,
                    gpointer user_data);
```

This is used to make the file chooser show the user's Desktop folder in the file list. By default this is bound to .

chooser the object which received the signal.

user-data user data set when the signal handler was connected.

81.5.6 The "GtkFileChooserDefault::quick-bookmark" signal

```
void user_function (GtkFileChooserDefault *chooser,
                    gint bookmark_index,
                    gpointer user_data);
```

This is used to make the file chooser switch to the bookmark specified in the *bookmark-index* parameter. For example, if you have three bookmarks, you can pass 0, 1, 2 to this signal to switch to each of them, respectively. By default this is bound to , , etc. until . Note that in the default binding, that is actually defined to switch to the bookmark at index 0, and so on successively; is defined to switch to the bookmark at index 10.

chooser the object which received the signal.

bookmark-indes
index of the bookmark to switch to; the indices start at 0.

user-data user data set when the signal handler was connected.

81.6 Usage

`<gtk-file-chooser>` [Class]

This `<gobject>` class defines the following properties:

`extra-widget`
: Application supplied widget for extra options.

`preview-widget`
: Application supplied widget for custom previews.

`action` The type of operation that the file selector is performing

`file-system-backend`
: Name of file system backend to use

`preview-widget-active`
: Whether the application supplied widget for custom previews should be shown.

`filter` The current filter for selecting which files are displayed

`show-hidden`
: Whether the hidden files and folders should be displayed

`use-preview-label`
: Whether to display a stock label with the name of the previewed file.

`do-overwrite-confirmation`
: Whether a file chooser in save mode will present an overwrite confirmation dialog if necessary.

`local-only`
: Whether the selected file(s) should be limited to local file: URLs

`select-multiple`
: Whether to allow multiple files to be selected

`current-folder-changed` [Signal on `<gtk-file-chooser>`]

This signal is emitted when the current folder in a `<gtk-file-chooser>` changes. This can happen due to the user performing some action that changes folders, such as selecting a bookmark or visiting a folder on the file list. It can also happen as a result of calling a function to explicitly change the current folder in a file chooser.

Normally you do not need to connect to this signal, unless you need to keep track of which folder a file chooser is showing.

See also: `gtk-file-chooser-set-current-folder`, `gtk-file-chooser-get-current-folder`, `gtk-file-chooser-set-current-folder-uri`, `gtk-file-chooser-get-current-folder-uri`.

`selection-changed` [Signal on `<gtk-file-chooser>`]

This signal is emitted when there is a change in the set of selected files in a `<gtk-file-chooser>`. This can happen when the user modifies the selection with the mouse or the keyboard, or when explicitly calling functions to change the selection.

Normally you do not need to connect to this signal, as it is easier to wait for the file chooser to finish running, and then to get the list of selected files using the functions mentioned below.

See also: `gtk-file-chooser-select-filename`, `gtk-file-chooser-unselect-filename`, `gtk-file-chooser-get-filename`, `gtk-file-chooser-get-filenames`, `gtk-file-chooser-select-uri`, `gtk-file-chooser-unselect-uri`, `gtk-file-chooser-get-uri`, `gtk-file-chooser-get-uris`.

`update-preview` [Signal on `<gtk-file-chooser>`]

This signal is emitted when the preview in a file chooser should be regenerated. For example, this can happen when the currently selected file changes. You should use this signal if you want your file chooser to have a preview widget.

Once you have installed a preview widget with `gtk-file-chooser-set-preview-widget`, you should update it when this signal is emitted. You can use the functions `gtk-file-chooser-get-preview-filename` or `gtk-file-chooser-get-preview-uri` to get the name of the file to preview. Your widget may not be able to preview all kinds of files; your callback must call `gtk-file-chooser-set-preview-wiget-active` to inform the file chooser about whether the preview was generated successfully or not.

Please see the example code in *(the missing figure, gtkfilechooser-preview.*

See also: `gtk-file-chooser-set-preview-widget`, `gtk-file-chooser-set-preview-widget-active`, `gtk-file-chooser-set-use-preview-label`, `gtk-file-chooser-get-preview-filename`, `gtk-file-chooser-get-preview-uri`.

`file-activated` [Signal on `<gtk-file-chooser>`]

This signal is emitted when the user "activates" a file in the file chooser. This can happen by double-clicking on a file in the file list, or by pressing .

Normally you do not need to connect to this signal. It is used internally by `<gtk-file-chooser-dialog>` to know when to activate the default button in the dialog.

See also: `gtk-file-chooser-get-filename`, `gtk-file-chooser-get-filenames`, `gtk-file-chooser-get-uri`, `gtk-file-chooser-get-uris`.

`confirm-overwrite` [Signal on `<gtk-file-chooser>`]

⇒ `<gtk-file-chooser-confirmation>`

This signal gets emitted whenever it is appropriate to present a confirmation dialog when the user has selected a file name that already exists. The signal only gets emitted when the file chooser is in `<gtk-file-chooser-action-save>` mode.

Most applications just need to turn on the do-overwrite-confirmation property (or call the `gtk-file-chooser-set-do-overwrite-confirmation` function), and they will automatically get a stock confirmation dialog. Applications which need to customize this behavior should do that, and also connect to the signal.

A signal handler for this signal must return a `<gtk-file-chooser-confirmation>` value, which indicates the action to take. If the handler determines that the user wants to select a different filename, it should return `<gtk-file-chooser-confirmation-select-again>`. If it determines that the user is satisfied with his choice of file

name, it should return <gtk-file-chooser-confirmation-accept-filename>. On the other hand, if it determines that the stock confirmation dialog should be used, it should return <gtk-file-chooser-confirmation-confirm>. The following example illustrates this.

gtk-file-chooser-set-action (*self* <gtk-file-chooser>) [Function]
(*action* <gtk-file-chooser-action>)

set-action [Method]

Sets the type of operation that the chooser is performing; the user interface is adapted to suit the selected action. For example, an option to create a new folder might be shown if the action is 'GTK_FILE_CHOOSER_ACTION_SAVE' but not if the action is 'GTK_FILE_CHOOSER_ACTION_OPEN'.

chooser a <gtk-file-chooser>

action the action that the file selector is performing

Since 2.4

gtk-file-chooser-get-action (*self* <gtk-file-chooser>) [Function]
⇒ (*ret* <gtk-file-chooser-action>)

get-action [Method]

Gets the type of operation that the file chooser is performing; see gtk-file-chooser-set-action.

chooser a <gtk-file-chooser>

ret the action that the file selector is performing

Since 2.4

gtk-file-chooser-set-local-only (*self* <gtk-file-chooser>) [Function]
(*local_only* bool)

set-local-only [Method]

Sets whether only local files can be selected in the file selector. If *local-only* is '#t' (the default), then the selected file are files are guaranteed to be accessible through the operating systems native file file system and therefore the application only needs to worry about the filename functions in <gtk-file-chooser>, like gtk-file-chooser-get-filename, rather than the URI functions like gtk-file-chooser-get-uri,

chooser a <gtk-file-chooser>

local-only '#t' if only local files can be selected

Since 2.4

gtk-file-chooser-get-local-only (*self* <gtk-file-chooser>) [Function]
⇒ (*ret* bool)

get-local-only [Method]

Gets whether only local files can be selected in the file selector. See gtk-file-chooser-set-local-only

chooser a <gtk-file-choosre>

ret '#t' if only local files can be selected.

Since 2.4

`gtk-file-chooser-set-show-hidden` (*self* `<gtk-file-chooser>`) (*show_hidden* `bool`) [Function]
`set-show-hidden` [Method]

Sets whether hidden files and folders are displayed in the file selector.

chooser a `<gtk-file-chooser>`

show-hidden '`#t`' if hidden files and folders should be displayed.

Since 2.6

`gtk-file-chooser-get-show-hidden` (*self* `<gtk-file-chooser>`) ⇒ (*ret* `bool`) [Function]
`get-show-hidden` [Method]

Gets whether hidden files and folders are displayed in the file selector. See `gtk-file-chooser-set-show-hidden`.

chooser a `<gtk-file-chooser>`

ret '`#t`' if hidden files and folders are displayed.

Since 2.6

`gtk-file-chooser-set-current-name` (*self* `<gtk-file-chooser>`) (*name* `mchars`) [Function]
`set-current-name` [Method]

Sets the current name in the file selector, as if entered by the user. Note that the name passed in here is a UTF-8 string rather than a filename. This function is meant for such uses as a suggested name in a "Save As..." dialog.

If you want to preselect a particular existing file, you should use `gtk-file-chooser-set-filename` or `gtk-file-chooser-set-uri` instead. Please see the documentation for those functions for an example of using `gtk-file-chooser-set-current-name` as well.

chooser a `<gtk-file-chooser>`

name the filename to use, as a UTF-8 string

Since 2.4

`gtk-file-chooser-get-filename` (*self* `<gtk-file-chooser>`) ⇒ (*ret* `mchars`) [Function]
`get-filename` [Method]

Gets the filename for the currently selected file in the file selector. If multiple files are selected, one of the filenames will be returned at random.

If the file chooser is in folder mode, this function returns the selected folder.

chooser a `<gtk-file-chooser>`

ret The currently selected filename, or '`#f`' if no file is selected, or the selected file can't be represented with a local filename. Free with `g-free`.

Since 2.4

`gtk-file-chooser-set-filename` (*self* `<gtk-file-chooser>`) [Function]
(*filename* `mchars`) ⇒ (*ret* `bool`)

`set-filename` [Method]

Sets *filename* as the current filename for the file chooser, by changing to the file's parent folder and actually selecting the file in list. If the *chooser* is in `<gtk-file-chooser-action-save>` mode, the file's base name will also appear in the dialog's file name entry.

If the file name isn't in the current folder of *chooser*, then the current folder of *chooser* will be changed to the folder containing *filename*. This is equivalent to a sequence of `gtk-file-chooser-unselect-all` followed by `gtk-file-chooser-select-filename`.

Note that the file must exist, or nothing will be done except for the directory change.

If you are implementing a dialog, you should use this function if you already have a file name to which the user may save; for example, when the user opens an existing file and then does on it. If you don't have a file name already — for example, if the user just created a new file and is saving it for the first time, do not call this function. Instead, use something similar to this:

```
if (document_is_new)
  {
    /* the user just created a new document */
    gtk_file_chooser_set_current_folder (chooser, default_folder_for_saving);
    gtk_file_chooser_set_current_name (chooser, "Untitled document");
  }
else
  {
    /* the user edited an existing document */
    gtk_file_chooser_set_filename (chooser, existing_filename);
  }
```

chooser a `<gtk-file-chooser>`

filename the filename to set as current

ret '`#t`' if both the folder could be changed and the file was selected successfully, '`#f`' otherwise.

Since 2.4

`gtk-file-chooser-select-filename` (*self* `<gtk-file-chooser>`) [Function]
(*filename* `mchars`) ⇒ (*ret* `bool`)

`select-filename` [Method]

Selects a filename. If the file name isn't in the current folder of *chooser*, then the current folder of *chooser* will be changed to the folder containing *filename*.

chooser a `<gtk-file-chooser>`

filename the filename to select

ret '`#t`' if both the folder could be changed and the file was selected successfully, '`#f`' otherwise.

Since 2.4

`gtk-file-chooser-unselect-filename` (*self* `<gtk-file-chooser>`) [Function]
(*filename* `mchars`)
`unselect-filename` [Method]

Unselects a currently selected filename. If the filename is not in the current directory, does not exist, or is otherwise not currently selected, does nothing.

chooser a `<gtk-file-chooser>`

filename the filename to unselect

Since 2.4

`gtk-file-chooser-select-all` (*self* `<gtk-file-chooser>`) [Function]
`select-all` [Method]

Selects all the files in the current folder of a file chooser.

chooser a `<gtk-file-chooser>`

Since 2.4

`gtk-file-chooser-unselect-all` (*self* `<gtk-file-chooser>`) [Function]
`unselect-all` [Method]

Unselects all the files in the current folder of a file chooser.

chooser a `<gtk-file-chooser>`

Since 2.4

`gtk-file-chooser-get-filenames` (*self* `<gtk-file-chooser>`) [Function]
⇒ (*ret* `gslist-of`)
`get-filenames` [Method]

Lists all the selected files and subfolders in the current folder of *chooser*. The returned names are full absolute paths. If files in the current folder cannot be represented as local filenames they will be ignored. (See `gtk-file-chooser-get-uris`)

chooser a `<gtk-file-chooser>`

ret a `<gs-list>` containing the filenames of all selected files and subfolders in the current folder. Free the returned list with `g-slist-free`, and the filenames with `g-free`.

Since 2.4

`gtk-file-chooser-set-current-folder` (*self* `<gtk-file-chooser>`) [Function]
(*filename* `mchars`) ⇒ (*ret* `bool`)
`set-current-folder` [Method]

Sets the current folder for *chooser* from a local filename. The user will be shown the full contents of the current folder, plus user interface elements for navigating to other folders.

chooser a `<gtk-file-chooser>`

filename the full path of the new current folder

ret '`#t`' if the folder could be changed successfully, '`#f`' otherwise.

Since 2.4

`gtk-file-chooser-get-current-folder` (*self* `<gtk-file-chooser>`) [Function]
⇒ (*ret* `mchars`)

`get-current-folder` [Method]

Gets the current folder of *chooser* as a local filename. See `gtk-file-chooser-set-current-folder`.

Note that this is the folder that the file chooser is currently displaying (e.g. "/home/username/Documents"), which is *not the same* as the currently-selected folder if the chooser is in `<gtk-file-chooser-select-folder>` mode (e.g. "/home/username/Documents/selected-folder/". To get the currently-selected folder in that mode, use `gtk-file-chooser-get-uri` as the usual way to get the selection.

chooser a `<gtk-file-chooser>`

ret the full path of the current folder, or '`#f`' if the current path cannot be represented as a local filename. Free with `g-free`. This function will also return '`#f`' if the file chooser was unable to load the last folder that was requested from it; for example, as would be for calling `gtk-file-chooser-set-current-folder` on a nonexistent folder.

Since 2.4

`gtk-file-chooser-get-uri` (*self* `<gtk-file-chooser>`) [Function]
⇒ (*ret* `mchars`)

`get-uri` [Method]

Gets the URI for the currently selected file in the file selector. If multiple files are selected, one of the filenames will be returned at random.

If the file chooser is in folder mode, this function returns the selected folder.

chooser a `<gtk-file-chooser>`

ret The currently selected URI, or '`#f`' if no file is selected. Free with `g-free`

Since 2.4

`gtk-file-chooser-set-uri` (*self* `<gtk-file-chooser>`) (*uri* `mchars`) [Function]
⇒ (*ret* `bool`)

`set-uri` [Method]

Sets the file referred to by *uri* as the current file for the file chooser, by changing to the URI's parent folder and actually selecting the URI in the list. If the *chooser* is `<gtk-file-chooser-action-save>` mode, the URI's base name will also appear in the dialog's file name entry.

If the URI isn't in the current folder of *chooser*, then the current folder of *chooser* will be changed to the folder containing *uri*. This is equivalent to a sequence of `gtk-file-chooser-unselect-all` followed by `gtk-file-chooser-select-uri`.

Note that the URI must exist, or nothing will be done except for the directory change. If you are implementing a dialog, you should use this function if you already have a file name to which the user may save; for example, when the user opens an existing

file and then does on it. If you don't have a file name already — for example, if the user just created a new file and is saving it for the first time, do not call this function. Instead, use something similar to this:

```
if (document_is_new)
  {
    /* the user just created a new document */
    gtk_file_chooser_set_current_folder_uri (chooser, default_folder_for_saving);
    gtk_file_chooser_set_current_name (chooser, "Untitled document");
  }
else
  {
    /* the user edited an existing document */
    gtk_file_chooser_set_uri (chooser, existing_uri);
  }
```

chooser a `<gtk-file-chooser>`

uri the URI to set as current

ret '`#t`' if both the folder could be changed and the URI was selected successfully, '`#f`' otherwise.

Since 2.4

`gtk-file-chooser-select-uri` (*self* `<gtk-file-chooser>`) [Function]
(*uri* `mchars`) ⇒ (*ret* `bool`)

`select-uri` [Method]

Selects the file to by *uri*. If the URI doesn't refer to a file in the current folder of *chooser*, then the current folder of *chooser* will be changed to the folder containing *filename*.

chooser a `<gtk-file-chooser>`

uri the URI to select

ret '`#t`' if both the folder could be changed and the URI was selected successfully, '`#f`' otherwise.

Since 2.4

`gtk-file-chooser-unselect-uri` (*self* `<gtk-file-chooser>`) [Function]
(*uri* `mchars`)

`unselect-uri` [Method]

Unselects the file referred to by *uri*. If the file is not in the current directory, does not exist, or is otherwise not currently selected, does nothing.

chooser a `<gtk-file-chooser>`

uri the URI to unselect

Since 2.4

`gtk-file-chooser-get-uris` (*self* `<gtk-file-chooser>`) [Function]
⇒ (*ret* `gslist-of`)
`get-uris` [Method]

Lists all the selected files and subfolders in the current folder of *chooser*. The returned names are full absolute URIs.

chooser a `<gtk-file-chooser>`

ret a `<gs-list>` containing the URIs of all selected files and subfolders in the current folder. Free the returned list with `g-slist-free`, and the filenames with `g-free`.

Since 2.4

`gtk-file-chooser-set-preview-widget` (*self* `<gtk-file-chooser>`) [Function]
(*preview_widget* `<gtk-widget>`)
`set-preview-widget` [Method]

Sets an application-supplied widget to use to display a custom preview of the currently selected file. To implement a preview, after setting the preview widget, you connect to the ::update-preview signal, and call `gtk-file-chooser-get-preview-filename` or `gtk-file-chooser-get-preview-uri` on each change. If you can display a preview of the new file, update your widget and set the preview active using `gtk-file-chooser-set-preview-widget-active`. Otherwise, set the preview inactive.

When there is no application-supplied preview widget, or the application-supplied preview widget is not active, the file chooser may display an internally generated preview of the current file or it may display no preview at all.

chooser a `<gtk-file-chooser>`

preview-widget
widget for displaying preview.

Since 2.4

`gtk-file-chooser-get-preview-widget` (*self* `<gtk-file-chooser>`) [Function]
⇒ (*ret* `<gtk-widget>`)
`get-preview-widget` [Method]

Gets the current preview widget; see `gtk-file-chooser-set-preview-widget`.

chooser a `<gtk-file-chooser>`

ret the current preview widget, or '`#f`'

Since 2.4

`gtk-file-chooser-get-preview-uri` (*self* `<gtk-file-chooser>`) [Function]
⇒ (*ret* `mchars`)
`get-preview-uri` [Method]

Gets the URI that should be previewed in a custom preview widget. See `gtk-file-chooser-set-preview-widget`.

chooser a `<gtk-file-chooser>`

ret the URI for the file to preview, or '`#f`' if no file is selected. Free with `g-free`.

Since 2.4

`gtk-file-chooser-set-extra-widget` (*self* `<gtk-file-chooser>`) [Function]
(*extra_widget* `<gtk-widget>`)
`set-extra-widget` [Method]

Sets an application-supplied widget to provide extra options to the user.

chooser a `<gtk-file-chooser>`

extra-widget
widget for extra options

Since 2.4

`gtk-file-chooser-get-extra-widget` (*self* `<gtk-file-chooser>`) [Function]
⇒ (*ret* `<gtk-widget>`)
`get-extra-widget` [Method]

Gets the current preview widget; see `gtk-file-chooser-set-extra-widget`.

chooser a `<gtk-file-chooser>`

ret the current extra widget, or '`#f`'

Since 2.4

`gtk-file-chooser-add-filter` (*self* `<gtk-file-chooser>`) [Function]
(*filter* `<gtk-file-filter*>`)
`add-filter` [Method]

Adds *filter* to the list of filters that the user can select between. When a filter is selected, only files that are passed by that filter are displayed.

Note that the *chooser* takes ownership of the filter, so you have to ref and sink it if you want to keep a reference.

chooser a `<gtk-file-chooser>`

filter a `<gtk-file-filter>`

Since 2.4

`gtk-file-chooser-remove-filter` (*self* `<gtk-file-chooser>`) [Function]
(*filter* `<gtk-file-filter*>`)
`remove-filter` [Method]

Removes *filter* from the list of filters that the user can select between.

chooser a `<gtk-file-chooser>`

filter a `<gtk-file-filter>`

Since 2.4

`gtk-file-chooser-list-filters` (*self* `<gtk-file-chooser>`) [Function]
⇒ (*ret* `gslist-of`)

`list-filters` [Method]

Lists the current set of user-selectable filters; see `gtk-file-chooser-add-filter`, `gtk-file-chooser-remove-filter`.

chooser a `<gtk-file-chooser>`

ret a `<gs-list>` containing the current set of user selectable filters. The contents of the list are owned by GTK+, but you must free the list itself with `g-slist-free` when you are done with it.

Since 2.4

`gtk-file-chooser-set-filter` (*self* `<gtk-file-chooser>`) [Function]
(*filter* `<gtk-file-filter*>`)

`set-filter` [Method]

Sets the current filter; only the files that pass the filter will be displayed. If the user-selectable list of filters is non-empty, then the filter should be one of the filters in that list. Setting the current filter when the list of filters is empty is useful if you want to restrict the displayed set of files without letting the user change it.

chooser a `<gtk-file-chooser>`

filter a `<gtk-file-filter>`

Since 2.4

`gtk-file-chooser-get-filter` (*self* `<gtk-file-chooser>`) [Function]
⇒ (*ret* `<gtk-file-filter*>`)

`get-filter` [Method]

Gets the current filter; see `gtk-file-chooser-set-filter`.

chooser a `<gtk-file-chooser>`

ret the current filter, or '`#f`'

Since 2.4

82 GtkFileChooserButton

A button to launch a file selection dialog

82.1 Overview

The <gtk-file-chooser-button> is a widget that lets the user select a file. It implements the <gtk-file-chooser> interface. Visually, it is a file name with a button to bring up a <gtk-file-chooser-dialog>. The user can then use that dialog to change the file associated with that button. This widget does not support setting the "select-multiple" property to '#t'.

```
{
  GtkWidget *button;

  button = gtk_file_chooser_button_new (_("Select a file"),
                                        GTK_FILE_CHOOSER_ACTION_OPEN);
  gtk_file_chooser_set_current_folder (GTK_FILE_CHOOSER (button),
                                       "/etc");
}
```

The <gtk-file-chooser-button> supports the <gtk-file-chooser-action>s 'GTK_FILE_CHOOSER_ACTION_OPEN' and 'GTK_FILE_CHOOSER_ACTION_SELECT_FOLDER'.

The <gtk-file-chooser-button> will ellipsize the label, and thus will thus request little horizontal space. To give the button more space, you should call gtk-widget-size-request, gtk-file-chooser-button-set-width-chars, or pack the button in such a way that other interface elements give space to the widget.

82.2 Usage

<gtk-file-chooser-button> [Class]

This <gobject> class defines the following properties:

dialog The file chooser dialog to use.

focus-on-click
Whether the button grabs focus when it is clicked with the mouse

title The title of the file chooser dialog.

width-chars
The desired width of the button widget, in characters.

file-set [Signal on <gtk-file-chooser-button>]

undocumented

gtk-file-chooser-button-new (*title* mchars) [Function]
(*action* <gtk-file-chooser-action>) ⇒ (*ret* <gtk-widget>)

Creates a new file-selecting button widget.

title the title of the browse dialog.

action the open mode for the widget.

ret a new button widget.

Since 2.6

`gtk-file-chooser-button-get-title` [Function]
(*self* `<gtk-file-chooser-button>`) ⇒ (*ret* `mchars`)

`get-title` [Method]

Retrieves the title of the browse dialog used by *button*. The returned value should not be modified or freed.

button the button widget to examine.

ret a pointer to the browse dialog's title.

Since 2.6

`gtk-file-chooser-button-set-title` [Function]
(*self* `<gtk-file-chooser-button>`) (*title* `mchars`)

`set-title` [Method]

Modifies the *title* of the browse dialog used by *button*.

button the button widget to modify.

title the new browse dialog title.

Since 2.6

83 GtkFileChooserDialog

A file chooser dialog, suitable for "File/Open" or "File/Save" commands

83.1 Overview

<gtk-file-chooser-dialog> is a dialog box suitable for use with "File/Open" or "File/Save as" commands. This widget works by putting a <gtk-file-chooser-widget> inside a <gtk-dialog>. It exposes the <gtk-file-chooser-iface> interface, so you can use all of the <gtk-file-chooser> functions on the file chooser dialog as well as those for <gtk-dialog>.

Note that <gtk-file-chooser-dialog> does not have any methods of its own. Instead, you should use the functions that work on a <gtk-file-chooser>.

In the simplest of cases, you can the following code to use <gtk-file-chooser-dialog> to select a file for opening:

```
GtkWidget *dialog;

dialog = gtk_file_chooser_dialog_new ("Open File",
      parent_window,
      GTK_FILE_CHOOSER_ACTION_OPEN,
      GTK_STOCK_CANCEL, GTK_RESPONSE_CANCEL,
      GTK_STOCK_OPEN, GTK_RESPONSE_ACCEPT,
      NULL);

if (gtk_dialog_run (GTK_DIALOG (dialog)) == GTK_RESPONSE_ACCEPT)
  {
    char *filename;

    filename = gtk_file_chooser_get_filename (GTK_FILE_CHOOSER (dialog));
    open_file (filename);
    g_free (filename);
  }

gtk_widget_destroy (dialog);
```

To use a dialog for saving, you can use this:

```
GtkWidget *dialog;

dialog = gtk_file_chooser_dialog_new ("Save File",
      parent_window,
      GTK_FILE_CHOOSER_ACTION_SAVE,
      GTK_STOCK_CANCEL, GTK_RESPONSE_CANCEL,
      GTK_STOCK_SAVE, GTK_RESPONSE_ACCEPT,
      NULL);
```

```
gtk_file_chooser_set_do_overwrite_confirmation (GTK_FILE_CHOOSER (dialog), TRUE);

if (user_edited_a_new_document)
  {
    gtk_file_chooser_set_current_folder (GTK_FILE_CHOOSER (dialog), default_folder_for
    gtk_file_chooser_set_current_name (GTK_FILE_CHOOSER (dialog), "Untitled document")
  }
else
  gtk_file_chooser_set_filename (GTK_FILE_CHOOSER (dialog), filename_for_existing_docu


if (gtk_dialog_run (GTK_DIALOG (dialog)) == GTK_RESPONSE_ACCEPT)
  {
    char *filename;

    filename = gtk_file_chooser_get_filename (GTK_FILE_CHOOSER (dialog));
    save_to_file (filename);
    g_free (filename);
  }

gtk_widget_destroy (dialog);
```

<gtk-file-chooser-dialog> inherits from <gtk-dialog>, so buttons that go in its action area have response codes such as <gtk-response-accept> and <gtk-response-cancel>. For example, you could call gtk-file-chooser-dialog-new as follows:

```
GtkWidget *dialog;

dialog = gtk_file_chooser_dialog_new ("Open File",
      parent_window,
      GTK_FILE_CHOOSER_ACTION_OPEN,
      GTK_STOCK_CANCEL, GTK_RESPONSE_CANCEL,
      GTK_STOCK_OPEN, GTK_RESPONSE_ACCEPT,
      NULL);
```

This will create buttons for "Cancel" and "Open" that use stock response identifiers from <gtk-response-type>. For most dialog boxes you can use your own custom response codes rather than the ones in <gtk-response-type>, but <gtk-file-chooser-dialog> assumes that its "accept"-type action, e.g. an "Open" or "Save" button, *will* have one of the following response codes:

This is because <gtk-file-chooser-dialog> must intercept responses and switch to folders if appropriate, rather than letting the dialog terminate — the implementation uses these known response codes to know which responses can be blocked if appropriate.

To summarize, make sure you use a stock response code when you use <gtk-file-chooser-dialog> to ensure proper operation.

83.2 Usage

`<gtk-file-chooser-dialog>` [Class]

This `<gobject>` class defines no properties, other than those defined by its superclasses.

84 GtkFileChooserWidget

File chooser widget that can be embedded in other widgets

84.1 Overview

<gtk-file-chooser-widget> is a widget suitable for selecting files. It is the main building block of a <gtk-file-chooser-dialog>. Most applications will only need to use the latter; you can use <gtk-file-chooser-widget> as part of a larger window if you have special needs.

Note that <gtk-file-chooser-widget> does not have any methods of its own. Instead, you should use the functions that work on a <gtk-file-chooser>.

84.2 Usage

<gtk-file-chooser-widget> [Class]

This <gobject> class defines no properties, other than those defined by its superclasses.

gtk-file-chooser-widget-new [Function]

(*action* <gtk-file-chooser-action>) ⇒ (*ret* <gtk-widget>)

Creates a new <gtk-file-chooser-widget>. This is a file chooser widget that can be embedded in custom windows, and it is the same widget that is used by <gtk-file-chooser-dialog>.

action Open or save mode for the widget

ret a new <gtk-file-chooser-widget>

Since 2.4

85 GtkFileFilter

A filter for selecting a file subset

85.1 Overview

A GtkFileFilter can be used to restrict the files being shown in a `<gtk-file-chooser>`. Files can be filtered based on their name (with `gtk-file-filter-add-pattern`), on their mime type (with `gtk-file-filter-add-mime-type`), or by a custom filter function (with `gtk-file-filter-add-custom`).

Filtering by mime types handles aliasing and subclassing of mime types; e.g. a filter for text/plain also matches a file with mime type application/rtf, since application/rtf is a subclass of text/plain. Note that `<gtk-file-filter>` allows wildcards for the subtype of a mime type, so you can e.g. filter for image/*.

Normally, filters are used by adding them to a `<gtk-file-chooser>`, see `gtk-file-chooser-add-filter`, but it is also possible to manually use a filter on a file with `gtk-file-filter-filter`.

85.2 Usage

`gtk-file-filter-new` ⇒ (*ret* `<gtk-file-filter*>`) [Function]

Creates a new `<gtk-file-filter>` with no rules added to it. Such a filter doesn't accept any files, so is not particularly useful until you add rules with `gtk-file-filter-add-mime-type`, `gtk-file-filter-add-pattern`, or `gtk-file-filter-add-custom`. To create a filter that accepts any file, use:

```
GtkFileFilter *filter = gtk_file_filter_new ();
gtk_file_filter_add_pattern (filter, "*");
```

ret a new `<gtk-file-filter>`

Since 2.4

`gtk-file-filter-set-name` (*self* `<gtk-file-filter*>`) [Function]
(*name* `mchars`)

Sets the human-readable name of the filter; this is the string that will be displayed in the file selector user interface if there is a selectable list of filters.

filter a `<gtk-file-filter>`

name the human-readable-name for the filter, or '`#f`' to remove any existing name.

Since 2.4

`gtk-file-filter-get-name` (*self* `<gtk-file-filter*>`) [Function]
⇒ (*ret* `mchars`)

Gets the human-readable name for the filter. See `gtk-file-filter-set-name`.

filter a `<gtk-file-filter>`

ret The human-readable name of the filter, or '`#f`'. This value is owned by GTK+ and must not be modified or freed.

Since 2.4

`gtk-file-filter-add-mime-type` (*self* `<gtk-file-filter*>`) [Function]
(*mime_type* `mchars`)
Adds a rule allowing a given mime type to *filter*.

filter A `<gtk-file-filter>`

mime-type
name of a MIME type

Since 2.4

`gtk-file-filter-add-pattern` (*self* `<gtk-file-filter*>`) [Function]
(*pattern* `mchars`)
Adds a rule allowing a shell style glob to a filter.

filter a `<gtk-file-filter>`

pattern a shell style glob

Since 2.4

`gtk-file-filter-add-pixbuf-formats` (*self* `<gtk-file-filter*>`) [Function]
Adds a rule allowing image files in the formats supported by GdkPixbuf.

filter a `<gtk-file-filter>`

Since 2.6

`gtk-file-filter-add-custom` (*self* `<gtk-file-filter*>`) [Function]
(*needed* `<gtk-file-filter-flags>`) (*func* `<gtk-file-filter-func>`)
(*data* `<gpointer>`) (*notify* `<g-destroy-notify>`)
Adds rule to a filter that allows files based on a custom callback function. The bitfield *needed* which is passed in provides information about what sorts of information that the filter function needs; this allows GTK+ to avoid retrieving expensive information when it isn't needed by the filter.

filter a `<gtk-file-filter>`

needed bitfield of flags indicating the information that the custom filter function needs.

func callback function; if the function returns '`#t`', then the file will be displayed.

data data to pass to *func*

notify function to call to free *data* when it is no longer needed.

Since 2.4

`gtk-file-filter-get-needed` (*self* `<gtk-file-filter*>`) [Function]
⇒ (*ret* `<gtk-file-filter-flags>`)

Gets the fields that need to be filled in for the structure passed to `gtk-file-filter-filter`

This function will not typically be used by applications; it is intended principally for use in the implementation of `<gtk-file-chooser>`.

filter a `<gtk-file-filter>`

ret bitfield of flags indicating needed fields when calling `gtk-file-filter-filter`

Since 2.4

`gtk-file-filter-filter` (*self* `<gtk-file-filter*>`) [Function]
(*filter_info* `<gtk-file-filter-info*>`) ⇒ (*ret* `bool`)

Tests whether a file should be displayed according to *filter*. The `<gtk-file-filter-info>` structure *filter-info* should include the fields returned from `gtk-file-filter-get-needed`.

This function will not typically be used by applications; it is intended principally for use in the implementation of `<gtk-file-chooser>`.

filter a `<gtk-file-filter>`

filter-info a `<gtk-file-filter-info>` structure containing information about a file.

ret '`#t`' if the file should be displayed

Since 2.4

86 GtkFontButton

A button to launch a font selection dialog

86.1 Overview

The <gtk-font-button> is a button which displays the currently selected font an allows to open a font selection dialog to change the font. It is suitable widget for selecting a font in a preference dialog.

86.2 Usage

<gtk-font-button> [Class]

This <gobject> class defines the following properties:

title The title of the font selection dialog

font-name
The name of the selected font

use-font Whether the label is drawn in the selected font

use-size Whether the label is drawn with the selected font size

show-style
Whether the selected font style is shown in the label

show-size
Whether selected font size is shown in the label

font-set [Signal on <gtk-font-button>]

The ::font-set signal is emitted when the user selects a font. When handling this signal, use gtk-font-button-get-font-name to find out which font was just selected.

Note that this signal is only emitted when the *user* changes the font. If you need to react to programmatic font changes as well, use the notify::font-name signal.

Since 2.4

gtk-font-button-new ⇒ (*ret* <gtk-widget>) [Function]

Creates a new font picker widget.

ret a new font picker widget.

Since 2.4

gtk-font-button-new-with-font (*fontname* mchars) [Function]
⇒ (*ret* <gtk-widget>)

Creates a new font picker widget.

fontname Name of font to display in font selection dialog

ret a new font picker widget.

Since 2.4

`gtk-font-button-set-font-name` (*self* `<gtk-font-button>`) [Function]
(*fontname* `mchars`) ⇒ (*ret* `bool`)

`set-font-name` [Method]

Sets or updates the currently-displayed font in font picker dialog.

font-button
a `<gtk-font-button>`

fontname Name of font to display in font selection dialog

ret Return value of `gtk-font-selection-dialog-set-font-name` if the font selection dialog exists, otherwise '`#f`'.

Since 2.4

`gtk-font-button-get-font-name` (*self* `<gtk-font-button>`) [Function]
⇒ (*ret* `mchars`)

`get-font-name` [Method]

Retrieves the name of the currently selected font.

font-button
a `<gtk-font-button>`

ret an internal copy of the font name which must not be freed.

Since 2.4

`gtk-font-button-set-show-style` (*self* `<gtk-font-button>`) [Function]
(*show_style* `bool`)

`set-show-style` [Method]

If *show-style* is '`#t`', the font style will be displayed along with name of the selected font.

font-button
a `<gtk-font-button>`

show-style '`#t`' if font style should be displayed in label.

Since 2.4

`gtk-font-button-get-show-style` (*self* `<gtk-font-button>`) [Function]
⇒ (*ret* `bool`)

`get-show-style` [Method]

Returns whether the name of the font style will be shown in the label.

font-button
a `<gtk-font-button>`

ret whether the font style will be shown in the label.

Since 2.4

`gtk-font-button-set-show-size` (*self* `<gtk-font-button>`) [Function]
(*show_size* `bool`)

`set-show-size` [Method]

If *show-size* is '`#t`', the font size will be displayed along with the name of the selected font.

font-button
a `<gtk-font-button>`

show-size '`#t`' if font size should be displayed in dialog.

Since 2.4

`gtk-font-button-get-show-size` (*self* `<gtk-font-button>`) [Function]
⇒ (*ret* `bool`)
`get-show-size` [Method]
Returns whether the font size will be shown in the label.

font-button
a `<gtk-font-button>`

ret whether the font size will be shown in the label.

Since 2.4

`gtk-font-button-set-use-font` (*self* `<gtk-font-button>`) [Function]
(*use_font* `bool`)
`set-use-font` [Method]
If *use-font* is '`#t`', the font name will be written using the selected font.

font-button
a `<gtk-font-button>`

use-font If '`#t`', font name will be written using font chosen.

Since 2.4

`gtk-font-button-get-use-font` (*self* `<gtk-font-button>`) [Function]
⇒ (*ret* `bool`)
`get-use-font` [Method]
Returns whether the selected font is used in the label.

font-button
a `<gtk-font-button>`

ret whether the selected font is used in the label.

Since 2.4

`gtk-font-button-set-use-size` (*self* `<gtk-font-button>`) [Function]
(*use_size* `bool`)
`set-use-size` [Method]
If *use-size* is '`#t`', the font name will be written using the selected size.

font-button
a `<gtk-font-button>`

use-size If '`#t`', font name will be written using the selected size.

Since 2.4

`gtk-font-button-get-use-size` (*self* `<gtk-font-button>`) [Function]
⇒ (*ret* `bool`)

`get-use-size` [Method]

Returns whether the selected size is used in the label.

font-button
a `<gtk-font-button>`

ret whether the selected size is used in the label.

Since 2.4

`gtk-font-button-set-title` (*self* `<gtk-font-button>`) [Function]
(*title* `mchars`)

`set-title` [Method]

Sets the title for the font selection dialog.

font-button
a `<gtk-font-button>`

title a string containing the font selection dialog title

Since 2.4

`gtk-font-button-get-title` (*self* `<gtk-font-button>`) [Function]
⇒ (*ret* `mchars`)

`get-title` [Method]

Retrieves the title of the font selection dialog.

font-button
a `<gtk-font-button>`

ret an internal copy of the title string which must not be freed.

Since 2.4

87 GtkFontSelection

A widget for selecting fonts

87.1 Overview

The <gtk-font-selection> widget lists the available fonts, styles and sizes, allowing the user to select a font. It is used in the <gtk-font-selection-dialog> widget to provide a dialog box for selecting fonts.

To set the font which is initially selected, use gtk-font-selection-set-font-name.

To get the selected font use gtk-font-selection-get-font-name.

To change the text which is shown in the preview area, use gtk-font-selection-set-preview-text.

87.2 Usage

<gtk-font-selection> [Class]

This <gobject> class defines the following properties:

font-name
: The X string that represents this font

font
: The GdkFont that is currently selected

preview-text
: The text to display in order to demonstrate the selected font

gtk-font-selection-new ⇒ (*ret* <gtk-widget>) [Function]

Creates a new <gtk-font-selection>.

ret
: a new <gtk-font-selection>.

gtk-font-selection-get-font-name (*self* <gtk-font-selection>) [Function]
⇒ (*ret* mchars)

get-font-name [Method]

Gets the currently-selected font name. Note that this can be a different string than what you set with gtk-font-selection-set-font-name, as the font selection widget may normalize font names and thus return a string with a different structure. For example, "Helvetica Italic Bold 12" could be normalized to "Helvetica Bold Italic 12". Use pango-font-description-equal if you want to compare two font descriptions.

fontsel
: a <gtk-font-selection>

ret
: A string with the name of the current font, or #f if no font is selected. You must free this string with g-free.

gtk-font-selection-set-font-name (*self* <gtk-font-selection>) [Function]
(*fontname* mchars) ⇒ (*ret* bool)

set-font-name [Method]

Sets the currently-selected font. Note that the *fontsel* needs to know the screen in which it will appear for this to work; this can be guaranteed by simply making sure that the *fontsel* is inserted in a toplevel window before you call this function.

fontsel a `<gtk-font-selection>`

fontname a font name like `"Helvetica 12"` or `"Times Bold 18"`

ret `#t` if the font could be set successfully; `#f` if no such font exists or if the *fontsel* doesn't belong to a particular screen yet.

`gtk-font-selection-get-preview-text` [Function]
(*self* `<gtk-font-selection>`) ⇒ (*ret* `mchars`)

`get-preview-text` [Method]

Gets the text displayed in the preview area.

fontsel a `<gtk-font-selection>`.

ret the text displayed in the preview area. This string is owned by the widget and should not be modified or freed.

`gtk-font-selection-set-preview-text` [Function]
(*self* `<gtk-font-selection>`) (*text* `mchars`)

`set-preview-text` [Method]

Sets the text displayed in the preview area.

fontsel a `<gtk-font-selection>`.

text the text to display in the preview area.

88 GtkFontSelectionDialog

A dialog box for selecting fonts

88.1 Overview

The `<gtk-font-selection-dialog>` widget is a dialog box for selecting a font.

To set the font which is initially selected, use `gtk-font-selection-dialog-set-font-name`.

To get the selected font use `gtk-font-selection-dialog-get-font-name`.

To change the text which is shown in the preview area, use `gtk-font-selection-dialog-set-preview-text`.

88.2 Usage

`<gtk-font-selection-dialog>` [Class]

This `<gobject>` class defines no properties, other than those defined by its superclasses.

`gtk-font-selection-dialog-new` (*title* `mchars`) [Function]
⇒ (*ret* `<gtk-widget>`)

Creates a new `<gtk-font-selection-dialog>`.

title the title of the dialog box.

ret a new `<gtk-font-selection-dialog>`.

89 GtkInputDialog

Configure devices for the XInput extension

89.1 Overview

NOTE this widget is considered too specialized/little-used for GTK+, and will in the future be moved to some other package. If your application needs this widget, feel free to use it, as the widget does work and is useful in some applications; it's just not of general interest. However, we are not accepting new features for the widget, and it will eventually move out of the GTK+ distribution.

`<gtk-input-dialog>` displays a dialog which allows the user to configure XInput extension devices. For each device, they can control the mode of the device (disabled, screen-relative, or window-relative), the mapping of axes to coordinates, and the mapping of the devices macro keys to key press events.

`<gtk-input-dialog>` contains two buttons to which the application can connect; one for closing the dialog, and one for saving the changes. No actions are bound to these by default. The changes that the user makes take effect immediately.

89.2 Usage

`<gtk-input-dialog>` [Class]
This `<gobject>` class defines no properties, other than those defined by its superclasses.

`enable-device` (*arg0* `<gdk-device>`) [Signal on `<gtk-input-dialog>`]
This signal is emitted when the user changes the mode of a device from `<gdk-mode-disabled>` to a `<gdk-mode-screen>` or `<gdk-mode-window>`.

`disable-device` (*arg0* `<gdk-device>`) [Signal on `<gtk-input-dialog>`]
This signal is emitted when the user changes the mode of a device from a `<gdk-mode-screen>` or `<gdk-mode-window>` to `<gdk-mode-enabled>`.

`gtk-input-dialog-new` ⇒ (*ret* `<gtk-widget>`) [Function]
Creates a new `<gtk-input-dialog>`.

ret the new `<gtk-input-dialog>`.

90 GtkAlignment

A widget which controls the alignment and size of its child

90.1 Overview

The `<gtk-alignment>` widget controls the alignment and size of its child widget. It has four settings: xscale, yscale, xalign, and yalign.

The scale settings are used to specify how much the child widget should expand to fill the space allocated to the `<gtk-alignment>`. The values can range from 0 (meaning the child doesn't expand at all) to 1 (meaning the child expands to fill all of the available space).

The align settings are used to place the child widget within the available area. The values range from 0 (top or left) to 1 (bottom or right). Of course, if the scale settings are both set to 1, the alignment settings have no effect.

90.2 Usage

`<gtk-alignment>` [Class]

This `<gobject>` class defines the following properties:

`xalign` Horizontal position of child in available space. 0.0 is left aligned, 1.0 is right aligned

`yalign` Vertical position of child in available space. 0.0 is top aligned, 1.0 is bottom aligned

`xscale` If available horizontal space is bigger than needed for the child, how much of it to use for the child. 0.0 means none, 1.0 means all

`yscale` If available vertical space is bigger than needed for the child, how much of it to use for the child. 0.0 means none, 1.0 means all

`top-padding`
The padding to insert at the top of the widget.

`bottom-padding`
The padding to insert at the bottom of the widget.

`left-padding`
The padding to insert at the left of the widget.

`right-padding`
The padding to insert at the right of the widget.

`gtk-alignment-new` (*xalign* `float`) (*yalign* `float`) (*xscale* `float`) (*yscale* `float`) ⇒ (*ret* `<gtk-widget>`) [Function]

Creates a new `<gtk-alignment>`.

xalign the horizontal alignment of the child widget, from 0 (left) to 1 (right).

yalign the vertical alignment of the child widget, from 0 (top) to 1 (bottom).

xscale the amount that the child widget expands horizontally to fill up unused space, from 0 to 1. A value of 0 indicates that the child widget should never expand. A value of 1 indicates that the child widget will expand to fill all of the space allocated for the `<gtk-alignment>`.

yscale the amount that the child widget expands vertically to fill up unused space, from 0 to 1. The values are similar to *xscale*.

ret the new `<gtk-alignment>`.

`gtk-alignment-set` (*self* `<gtk-alignment>`) (*xalign* `float`) (*yalign* `float`) (*xscale* `float`) (*yscale* `float`) [Function]

`set` [Method]

Sets the `<gtk-alignment>` values.

alignment a `<gtk-alignment>`.

xalign the horizontal alignment of the child widget, from 0 (left) to 1 (right).

yalign the vertical alignment of the child widget, from 0 (top) to 1 (bottom).

xscale the amount that the child widget expands horizontally to fill up unused space, from 0 to 1. A value of 0 indicates that the child widget should never expand. A value of 1 indicates that the child widget will expand to fill all of the space allocated for the `<gtk-alignment>`.

yscale the amount that the child widget expands vertically to fill up unused space, from 0 to 1. The values are similar to *xscale*.

`gtk-alignment-get-padding` (*self* `<gtk-alignment>`) [Function]
⇒ (*padding_top* `unsigned-int`) (*padding_bottom* `unsigned-int`) (*padding_left* `unsigned-int`) (*padding_right* `unsigned-int`)

`get-padding` [Method]

Gets the padding on the different sides of the widget. See `gtk-alignment-set-padding`.

alignment a `<gtk-alignment>`

padding-top
location to store the padding for the top of the widget, or '`#f`'

padding-bottom
location to store the padding for the bottom of the widget, or '`#f`'

padding-left
location to store the padding for the left of the widget, or '`#f`'

padding-right
location to store the padding for the right of the widget, or '`#f`'

Since 2.4

`gtk-alignment-set-padding` (*self* `<gtk-alignment>`) (*padding_top* `unsigned-int`) (*padding_bottom* `unsigned-int`) (*padding_left* `unsigned-int`) (*padding_right* `unsigned-int`) [Function]

`set-padding` [Method]

Sets the padding on the different sides of the widget. The padding adds blank space to the sides of the widget. For instance, this can be used to indent the child widget towards the right by adding padding on the left.

alignment a `<gtk-alignment>`

padding-top
the padding at the top of the widget

padding-bottom
the padding at the bottom of the widget

padding-left
the padding at the left of the widget

padding-right
the padding at the right of the widget.

Since 2.4

91 GtkAspectFrame

A frame that constrains its child to a particular aspect ratio

91.1 Overview

The <gtk-aspect-frame> is useful when you want pack a widget so that it can resize but always retains the same aspect ratio. For instance, one might be drawing a small preview of a larger image. <gtk-aspect-frame> derives from <gtk-frame>, so it can draw a label and a frame around the child. The frame will be "shrink-wrapped" to the size of the child.

91.2 Usage

<gtk-aspect-frame> [Class]

This <gobject> class defines the following properties:

xalign X alignment of the child

yalign Y alignment of the child

ratio Aspect ratio if obey_child is FALSE

obey-child
Force aspect ratio to match that of the frame's child

gtk-aspect-frame-new (*label* mchars) (*xalign* float) (*yalign* float) (*ratio* float) (*obey_child* bool) ⇒ (*ret* <gtk-widget>) [Function]

Create a new <gtk-aspect-frame>.

label Label text.

xalign Horizontal alignment of the child within the allocation of the <gtk-aspect-frame>. This ranges from 0.0 (left aligned) to 1.0 (right aligned)

yalign Vertical alignment of the child within the allocation of the <gtk-aspect-frame>. This ranges from 0.0 (left aligned) to 1.0 (right aligned)

ratio The desired aspect ratio.

obey-child If '#t', *ratio* is ignored, and the aspect ratio is taken from the requistion of the child.

ret the new <gtk-aspect-frame>.

gtk-aspect-frame-set (*self* <gtk-aspect-frame>) (*xalign* float) (*yalign* float) (*ratio* float) (*obey_child* bool) [Function]

set [Method]

Set parameters for an existing <gtk-aspect-frame>.

aspect-frame
a <gtk-aspect-frame>

xalign Horizontal alignment of the child within the allocation of the <gtk-aspect-frame>. This ranges from 0.0 (left aligned) to 1.0 (right aligned)

yalign Vertical alignment of the child within the allocation of the `<gtk-aspect-frame>`. This ranges from 0.0 (left aligned) to 1.0 (right aligned)

ratio The desired aspect ratio.

obey-child If '`#t`', *ratio* is ignored, and the aspect ratio is taken from the requistion of the child.

92 GtkHBox

A horizontal container box

92.1 Overview

GtkHBox is a container that organizes child widgets into a single row.

Use the `<gtk-box>` packing interface to determine the arrangement, spacing, width, and alignment of GtkHBox children.

All children are allocated the same height.

92.2 Usage

`<gtk-hbox>` [Class]

This `<gobject>` class defines no properties, other than those defined by its superclasses.

`gtk-hbox-new` (*homogeneous* `bool`) (*spacing* `int`) [Function]
⇒ (*ret* `<gtk-widget>`)

Creates a new GtkHBox.

homogeneous
: '`#t`' if all children are to be given equal space allotments.

spacing
: the number of pixels to place by default between children.

ret
: a new GtkHBox.

93 GtkVBox

A vertical container box

93.1 Overview

GtkVBox is a container that organizes child widgets into a single column.

Use the `<gtk-box>` packing interface to determine the arrangement, spacing, height, and alignment of GtkVBox children.

All children are allocated the same width.

93.2 Usage

`<gtk-vbox>` [Class]

This `<gobject>` class defines no properties, other than those defined by its superclasses.

`gtk-vbox-new` (*homogeneous* `bool`) (*spacing* `int`) [Function]
⇒ (*ret* `<gtk-widget>`)

Creates a new GtkVBox.

homogeneous
: '`#t`' if all children are to be given equal space allotments.

spacing
: the number of pixels to place by default between children.

ret
: a new GtkVBox.

94 GtkHButtonBox

A container for arranging buttons horizontally

94.1 Overview

A button box should be used to provide a consistent layout of buttons throughout your application. The layout/spacing can be altered by the programmer, or if desired, by the user to alter the 'feel' of a program to a small degree.

A `<gtk-hbutton-box>` is created with `gtk-hbutton-box-new`. Buttons are packed into a button box the same way widgets are added to any other container, using `gtk-container-add`. You can also use `gtk-box-pack-start` or `gtk-box-pack-end`, but for button boxes both these functions work just like `gtk-container-add`, ie., they pack the button in a way that depends on the current layout style and on whether the button has had `gtk-button-box-set-child-secondary` called on it.

The spacing between buttons can be set with `gtk-box-set-spacing`. The arrangement and layout of the buttons can be changed with `gtk-button-box-set-layout`.

94.2 Usage

`<gtk-hbutton-box>` [Class]

This `<gobject>` class defines no properties, other than those defined by its superclasses.

`gtk-hbutton-box-new` ⇒ (*ret* `<gtk-widget>`) [Function]

Creates a new horizontal button box.

ret a new button box `<gtk-widget>`.

95 GtkVButtonBox

A container for arranging buttons vertically

95.1 Overview

A button box should be used to provide a consistent layout of buttons throughout your application. The layout/spacing can be altered by the programmer, or if desired, by the user to alter the 'feel' of a program to a small degree.

A `<gtk-vbutton-box>` is created with `gtk-vbutton-box-new`. Buttons are packed into a button box the same way widgets are added to any other container, using `gtk-container-add`. You can also use `gtk-box-pack-start` or `gtk-box-pack-end`, but for button boxes both these functions work just like `gtk-container-add`, ie., they pack the button in a way that depends on the current layout style and on whether the button has had `gtk-button-box-set-child-secondary` called on it.

The spacing between buttons can be set with `gtk-box-set-spacing`. The arrangement and layout of the buttons can be changed with `gtk-button-box-set-layout`.

95.2 Usage

`<gtk-vbutton-box>` [Class]

This `<gobject>` class defines no properties, other than those defined by its superclasses.

`gtk-vbutton-box-new` ⇒ (*ret* `<gtk-widget>`) [Function]

Creates a new vertical button box.

ret a new button box `<gtk-widget>`.

96 GtkFixed

A container which allows you to position widgets at fixed coordinates

96.1 Overview

The <gtk-fixed> widget is a container which can place child widgets at fixed positions and with fixed sizes, given in pixels. <gtk-fixed> performs no automatic layout management.

For most applications, you should not use this container! It keeps you from having to learn about the other GTK+ containers, but it results in broken applications. With <gtk-fixed>, the following things will result in truncated text, overlapping widgets, and other display bugs:

Themes, which may change widget sizes.

Fonts other than the one you used to write the app will of course change the size of widgets containing text; keep in mind that users may use a larger font because of difficulty reading the default, or they may be using Windows or the framebuffer port of GTK+, where different fonts are available.

Translation of text into other languages changes its size. Also, display of non-English text will use a different font in many cases.

In addition, the fixed widget can't properly be mirrored in right-to-left languages such as Hebrew and Arabic. i.e. normally GTK+ will flip the interface to put labels to the right of the thing they label, but it can't do that with <gtk-fixed>. So your application will not be usable in right-to-left languages.

Finally, fixed positioning makes it kind of annoying to add/remove GUI elements, since you have to reposition all the other elements. This is a long-term maintenance problem for your application.

If you know none of these things are an issue for your application, and prefer the simplicity of <gtk-fixed>, by all means use the widget. But you should be aware of the tradeoffs.

96.2 Usage

<gtk-fixed> [Class]

This <gobject> class defines no properties, other than those defined by its superclasses.

gtk-fixed-new ⇒ (*ret* <gtk-widget>) [Function]

Creates a new <gtk-fixed>.

ret a new <gtk-fixed>.

gtk-fixed-put (*self* <gtk-fixed>) (*widget* <gtk-widget>) (*x* int) (*y* int) [Function]

put [Method]

Adds a widget to a <gtk-fixed> container at the given position.

fixed a <gtk-fixed>.

widget the widget to add.

x the horizontal position to place the widget at.

y the vertical position to place the widget at.

`gtk-fixed-move` (*self* `<gtk-fixed>`) (*widget* `<gtk-widget>`) (*x* `int`) (*y* `int`) [Function]

`move` [Method]

Moves a child of a `<gtk-fixed>` container to the given position.

fixed a `<gtk-fixed>`.

widget the child widget.

x the horizontal position to move the widget to.

y the vertical position to move the widget to.

`gtk-fixed-get-has-window` (*self* `<gtk-fixed>`) ⇒ (*ret* `bool`) [Function]

`get-has-window` [Method]

Gets whether the `<gtk-fixed>` has its own `<gdk-window>`. See `gdk-fixed-set-has-window`.

fixed a `<gtk-widget>`

ret '`#t`' if *fixed* has its own window.

`gtk-fixed-set-has-window` (*self* `<gtk-fixed>`) (*has_window* `bool`) [Function]

`set-has-window` [Method]

Sets whether a `<gtk-fixed>` widget is created with a separate `<gdk-window>` for *widget*`->`*window* or not. (By default, it will be created with no separate `<gdk-window>`). This function must be called while the `<gtk-fixed>` is not realized, for instance, immediately after the window is created.

This function was added to provide an easy migration path for older applications which may expect `<gtk-fixed>` to have a separate window.

fixed a `<gtk-fixed>`

has-window

'`#t`' if a separate window should be created

97 GtkHPaned

A container with two panes arranged horizontally

97.1 Overview

The HPaned widget is a container widget with two children arranged horizontally. The division between the two panes is adjustable by the user by dragging a handle. See `<gtk-paned>` for details.

97.2 Usage

`<gtk-hpaned>` [Class]
This `<gobject>` class defines no properties, other than those defined by its superclasses.

`gtk-hpaned-new` ⇒ (*ret* `<gtk-widget>`) [Function]
Create a new `<gtk-hpaned>`

ret the new `<gtk-hpaned>`

98 GtkVPaned

A container with two panes arranged vertically

98.1 Overview

The VPaned widget is a container widget with two children arranged vertically. The division between the two panes is adjustable by the user by dragging a handle. See `<gtk-paned>` for details.

98.2 Usage

`<gtk-vpaned>` [Class]
This `<gobject>` class defines no properties, other than those defined by its superclasses.

`gtk-vpaned-new` ⇒ (*ret* `<gtk-widget>`) [Function]
Create a new `<gtk-vpaned>`

ret the new `<gtk-vpaned>`

99 GtkLayout

Infinite scrollable area containing child widgets and/or custom drawing

99.1 Overview

`<gtk-layout>` is similar to `<gtk-drawing-area>` in that it's a "blank slate" and doesn't do anything but paint a blank background by default. It's different in that it supports scrolling natively (you can add it to a `<gtk-scrolled-window>`), and it can contain child widgets, since it's a `<gtk-container>`. However if you're just going to draw, a `<gtk-drawing-area>` is a better choice since it has lower overhead.

When handling expose events on a `<gtk-layout>`, you must draw to GTK_LAYOUT (layout)->bin_window, rather than to GTK_WIDGET (layout)->window, as you would for a drawing area.

99.2 Usage

`<gtk-layout>` [Class]

This `<gobject>` class defines the following properties:

`hadjustment`
: The GtkAdjustment for the horizontal position

`vadjustment`
: The GtkAdjustment for the vertical position

`width` The width of the layout

`height` The height of the layout

`set-scroll-adjustments` (*arg0* `<gtk-adjustment>`) (*arg1* `<gtk-adjustment>`) [Signal on `<gtk-layout>`]

`gtk-layout-new` (*hadjustment* `<gtk-adjustment>`) (*vadjustment* `<gtk-adjustment>`) ⇒ (*ret* `<gtk-widget>`) [Function]

Creates a new `<gtk-layout>`. Unless you have a specific adjustment you'd like the layout to use for scrolling, pass '`#f`' for *hadjustment* and *vadjustment*.

hadjustment
: horizontal scroll adjustment, or '`#f`'

vadjustment
: vertical scroll adjustment, or '`#f`'

ret a new `<gtk-layout>`

`gtk-layout-put` (*self* `<gtk-layout>`) (*child_widget* `<gtk-widget>`) (*x* `int`) (*y* `int`) [Function]

`put` [Method]

Adds *child-widget* to *layout*, at position (x,y). *layout* becomes the new parent container of *child-widget*.

layout a `<gtk-layout>`

child-widget
child widget

x X position of child widget

y Y position of child widget

`gtk-layout-move` (*self* `<gtk-layout>`) (*child_widget* `<gtk-widget>`) (*x* `int`) (*y* `int`) [Function]

`move` [Method]

Moves a current child of *layout* to a new position.

layout a `<gtk-layout>`

child-widget
a current child of *layout*

x X position to move to

y Y position to move to

`gtk-layout-set-size` (*self* `<gtk-layout>`) (*width* `unsigned-int`) (*height* `unsigned-int`) [Function]

`set-size` [Method]

Sets the size of the scrollable area of the layout.

layout a `<gtk-layout>`

width width of entire scrollable area

height height of entire scrollable area

`gtk-layout-get-size` (*self* `<gtk-layout>`) ⇒ (*width* `unsigned-int`) (*height* `unsigned-int`) [Function]

`get-size` [Method]

Gets the size that has been set on the layout, and that determines the total extents of the layout's scrollbar area. See `gtk-layout-set-size`.

layout a `<gtk-layout>`

width location to store the width set on *layout*, or '`#f`'

height location to store the height set on *layout*, or '`#f`'

`gtk-layout-get-hadjustment` (*self* `<gtk-layout>`) ⇒ (*ret* `<gtk-adjustment>`) [Function]

`get-hadjustment` [Method]

This function should only be called after the layout has been placed in a `<gtk-scrolled-window>` or otherwise configured for scrolling. It returns the `<gtk-adjustment>` used for communication between the horizontal scrollbar and *layout*.

See `<gtk-scrolled-window>`, `<gtk-scrollbar>`, `<gtk-adjustment>` for details.

layout a `<gtk-layout>`

ret horizontal scroll adjustment

`gtk-layout-get-vadjustment` (*self* `<gtk-layout>`) [Function]
⇒ (*ret* `<gtk-adjustment>`)

`get-vadjustment` [Method]

This function should only be called after the layout has been placed in a `<gtk-scrolled-window>` or otherwise configured for scrolling. It returns the `<gtk-adjustment>` used for communication between the vertical scrollbar and *layout*.

See `<gtk-scrolled-window>`, `<gtk-scrollbar>`, `<gtk-adjustment>` for details.

layout a `<gtk-layout>`

ret vertical scroll adjustment

`gtk-layout-set-hadjustment` (*self* `<gtk-layout>`) [Function]
(*adjustment* `<gtk-adjustment>`)

`set-hadjustment` [Method]

Sets the horizontal scroll adjustment for the layout.

See `<gtk-scrolled-window>`, `<gtk-scrollbar>`, `<gtk-adjustment>` for details.

layout a `<gtk-layout>`

adjustment
new scroll adjustment

`gtk-layout-set-vadjustment` (*self* `<gtk-layout>`) [Function]
(*adjustment* `<gtk-adjustment>`)

`set-vadjustment` [Method]

Sets the vertical scroll adjustment for the layout.

See `<gtk-scrolled-window>`, `<gtk-scrollbar>`, `<gtk-adjustment>` for details.

layout a `<gtk-layout>`

adjustment
new scroll adjustment

100 GtkNotebook

A tabbed notebook container

100.1 Overview

The `<gtk-notebook>` widget is a `<gtk-container>` whose children are pages that can be switched between using tab labels along one edge.

There are many configuration options for `<gtk-notebook>`. Among other things, you can choose on which edge the tabs appear (see `gtk-notebook-set-tab-pos`), whether, if there are too many tabs to fit the noteobook should be made bigger or scrolling arrows added (see gtk_notebook_set_scrollable), and whether there will be a popup menu allowing the users to switch pages. (see `gtk-notebook-enable-popup`, `gtk-noteobook-disable-popup`)

100.2 Usage

`<gtk-notebook>` [Class]

This `<gobject>` class defines the following properties:

`tab-pos` Which side of the notebook holds the tabs

`show-tabs`
: Whether tabs should be shown or not

`show-border`
: Whether the border should be shown or not

`scrollable`
: If TRUE, scroll arrows are added if there are too many tabs to fit

`tab-border`
: Width of the border around the tab labels

`tab-hborder`
: Width of the horizontal border of tab labels

`tab-vborder`
: Width of the vertical border of tab labels

`page` The index of the current page

`enable-popup`
: If TRUE, pressing the right mouse button on the notebook pops up a menu that you can use to go to a page

`group-id` Group ID for tabs drag and drop

`group` Group for tabs drag and drop

`homogeneous`
: Whether tabs should have homogeneous sizes

`switch-page` (*arg0* `<gpointer>`) (*arg1* `<guint>`) [Signal on `<gtk-notebook>`]

Emitted when the user or a function changes the current page.

`focus-tab` (*arg0* `<gtk-notebook-tab>`) ⇒ `<gboolean>` [Signal on `<gtk-notebook>`]

`select-page` (*arg0* `<gboolean>`) ⇒ `<gboolean>` [Signal on `<gtk-notebook>`]

`change-current-page` (*arg0* `<gint>`) ⇒ `<gboolean>` [Signal on `<gtk-notebook>`]

`move-focus-out` (*arg0* `<gtk-direction-type>`) [Signal on `<gtk-notebook>`]

`reorder-tab` (*arg0* `<gtk-direction-type>`) (*arg1* `<gboolean>`) ⇒ `<gboolean>` [Signal on `<gtk-notebook>`]

`page-reordered` (*arg0* `<gtk-widget>`) (*arg1* `<guint>`) [Signal on `<gtk-notebook>`]

the ::page-reordered signal is emitted in the notebook right after a page has been reordered.

Since 2.10

`page-removed` (*arg0* `<gtk-widget>`) (*arg1* `<guint>`) [Signal on `<gtk-notebook>`]

the ::page-removed signal is emitted in the notebook right after a page is removed from the notebook.

Since 2.10

`page-added` (*arg0* `<gtk-widget>`) (*arg1* `<guint>`) [Signal on `<gtk-notebook>`]

the ::page-added signal is emitted in the notebook right after a page is added to the notebook.

Since 2.10

`create-window` (*arg0* `<gtk-widget>`) (*arg1* `<gint>`) (*arg2* `<gint>`) ⇒ `<gtk-notebook>` [Signal on `<gtk-notebook>`]

undocumented

`gtk-notebook-new` ⇒ (*ret* `<gtk-widget>`) [Function]

Creates a new `<gtk-notebook>` widget with no pages.

ret the newly created `<gtk-notebook>`

`gtk-notebook-append-page` (*self* `<gtk-notebook>`) (*child* `<gtk-widget>`) (*tab_label* `<gtk-widget>`) ⇒ (*ret* `int`) [Function]

`append-page` [Method]

Appends a page to *notebook*.

notebook a `<gtk-notebook>`

child the `<gtk-widget>` to use as the contents of the page.

tab-label the `<gtk-widget>` to be used as the label for the page, or '`#f`' to use the default label, 'page N'.

ret the index (starting from 0) of the appended page in the notebook, or -1 if function fails

`gtk-notebook-append-page-menu` (*self* `<gtk-notebook>`) [Function]
(*child* `<gtk-widget>`) (*tab_label* `<gtk-widget>`)
(*menu_label* `<gtk-widget>`) ⇒ (*ret* `int`)

`append-page-menu` [Method]

Appends a page to *notebook*, specifying the widget to use as the label in the popup menu.

notebook a `<gtk-notebook>`

child the `<gtk-widget>` to use as the contents of the page.

tab-label the `<gtk-widget>` to be used as the label for the page, or '`#f`' to use the default label, 'page N'.

menu-label the widget to use as a label for the page-switch menu, if that is enabled. If '`#f`', and *tab-label* is a `<gtk-label>` or '`#f`', then the menu label will be a newly created label with the same text as *tab-label*; If *tab-label* is not a `<gtk-label>`, *menu-label* must be specified if the page-switch menu is to be used.

ret the index (starting from 0) of the appended page in the notebook, or -1 if function fails

`gtk-notebook-prepend-page` (*self* `<gtk-notebook>`) [Function]
(*child* `<gtk-widget>`) (*tab_label* `<gtk-widget>`) ⇒ (*ret* `int`)

`prepend-page` [Method]

Prepends a page to *notebook*.

notebook a `<gtk-notebook>`

child the `<gtk-widget>` to use as the contents of the page.

tab-label the `<gtk-widget>` to be used as the label for the page, or '`#f`' to use the default label, 'page N'.

ret the index (starting from 0) of the prepended page in the notebook, or -1 if function fails

`gtk-notebook-prepend-page-menu` (*self* `<gtk-notebook>`) [Function]
(*child* `<gtk-widget>`) (*tab_label* `<gtk-widget>`)
(*menu_label* `<gtk-widget>`) ⇒ (*ret* `int`)

`prepend-page-menu` [Method]

Prepends a page to *notebook*, specifying the widget to use as the label in the popup menu.

notebook a `<gtk-notebook>`

child the `<gtk-widget>` to use as the contents of the page.

tab-label the `<gtk-widget>` to be used as the label for the page, or '`#f`' to use the default label, 'page N'.

menu-label
: the widget to use as a label for the page-switch menu, if that is enabled. If '`#f`', and *tab-label* is a `<gtk-label>` or '`#f`', then the menu label will be a newly created label with the same text as *tab-label*; If *tab-label* is not a `<gtk-label>`, *menu-label* must be specified if the page-switch menu is to be used.

ret
: the index (starting from 0) of the prepended page in the notebook, or -1 if function fails

`gtk-notebook-insert-page` (*self* `<gtk-notebook>`) [Function]
(*child* `<gtk-widget>`) (*tab_label* `<gtk-widget>`) (*position* `int`) ⇒ (*ret* `int`)

`insert-page` [Method]

Insert a page into *notebook* at the given position.

notebook
: a `<gtk-notebook>`

child
: the `<gtk-widget>` to use as the contents of the page.

tab-label
: the `<gtk-widget>` to be used as the label for the page, or '`#f`' to use the default label, 'page N'.

position
: the index (starting at 0) at which to insert the page, or -1 to append the page after all other pages.

ret
: the index (starting from 0) of the inserted page in the notebook, or -1 if function fails

`gtk-notebook-insert-page-menu` (*self* `<gtk-notebook>`) [Function]
(*child* `<gtk-widget>`) (*tab_label* `<gtk-widget>`)
(*menu_label* `<gtk-widget>`) (*position* `int`) ⇒ (*ret* `int`)

`insert-page-menu` [Method]

Insert a page into *notebook* at the given position, specifying the widget to use as the label in the popup menu.

notebook
: a `<gtk-notebook>`

child
: the `<gtk-widget>` to use as the contents of the page.

tab-label
: the `<gtk-widget>` to be used as the label for the page, or '`#f`' to use the default label, 'page N'.

menu-label
: the widget to use as a label for the page-switch menu, if that is enabled. If '`#f`', and *tab-label* is a `<gtk-label>` or '`#f`', then the menu label will be a newly created label with the same text as *tab-label*; If *tab-label* is not a `<gtk-label>`, *menu-label* must be specified if the page-switch menu is to be used.

position
: the index (starting at 0) at which to insert the page, or -1 to append the page after all other pages.

ret
: the index (starting from 0) of the inserted page in the notebook

`gtk-notebook-remove-page` (*self* `<gtk-notebook>`) (*page_num* `int`) [Function]
`remove-page` [Method]

Removes a page from the notebook given its index in the notebook.

notebook a `<gtk-notebook>`.

page-num the index of a notebook page, starting from 0. If -1, the last page will be removed.

`gtk-notebook-page-num` (*self* `<gtk-notebook>`) (*child* `<gtk-widget>`) [Function]
⇒ (*ret* `int`)
`page-num` [Method]

Finds the index of the page which contains the given child widget.

notebook a `<gtk-notebook>`

child a `<gtk-widget>`

ret the index of the page containing *child*, or -1 if *child* is not in the notebook.

`gtk-notebook-next-page` (*self* `<gtk-notebook>`) [Function]
`next-page` [Method]

Switches to the next page. Nothing happens if the current page is the last page.

notebook a `<gtk-notebook>`

`gtk-notebook-prev-page` (*self* `<gtk-notebook>`) [Function]
`prev-page` [Method]

Switches to the previous page. Nothing happens if the current page is the first page.

notebook a `<gtk-notebook>`

`gtk-notebook-reorder-child` (*self* `<gtk-notebook>`) [Function]
(*child* `<gtk-widget>`) (*position* `int`)
`reorder-child` [Method]

Reorders the page containing *child*, so that it appears in position *position*. If *position* is greater than or equal to the number of children in the list or negative, *child* will be moved to the end of the list.

notebook a `<gtk-notebook>`

child the child to move

position the new position, or -1 to move to the end

`gtk-notebook-set-tab-pos` (*self* `<gtk-notebook>`) [Function]
(*pos* `<gtk-position-type>`)
`set-tab-pos` [Method]

Sets the edge at which the tabs for switching pages in the notebook are drawn.

notebook a `<gtk-notebook>`.

pos the edge to draw the tabs at.

`gtk-notebook-set-show-tabs` (*self* `<gtk-notebook>`) [Function]
(*show_tabs* `bool`)
`set-show-tabs` [Method]

Sets whether to show the tabs for the notebook or not.

notebook a `<gtk-notebook>`

show-tabs '`#t`' if the tabs should be shown.

`gtk-notebook-set-show-border` (*self* `<gtk-notebook>`) [Function]
(*show_border* `bool`)
`set-show-border` [Method]

Sets whether a bevel will be drawn around the notebook pages. This only has a visual effect when the tabs are not shown. See `gtk-notebook-set-show-tabs`.

notebook a `<gtk-notebook>`

show-border
'`#t`' if a bevel should be drawn around the notebook.

`gtk-notebook-set-scrollable` (*self* `<gtk-notebook>`) [Function]
(*scrollable* `bool`)
`set-scrollable` [Method]

Sets whether the tab label area will have arrows for scrolling if there are too many tabs to fit in the area.

notebook a `<gtk-notebook>`

scrollable '`#t`' if scroll arrows should be added

`gtk-notebook-popup-enable` (*self* `<gtk-notebook>`) [Function]
`popup-enable` [Method]

Enables the popup menu: if the user clicks with the right mouse button on the bookmarks, a menu with all the pages will be popped up.

notebook a `<gtk-notebook>`

`gtk-notebook-popup-disable` (*self* `<gtk-notebook>`) [Function]
`popup-disable` [Method]

Disables the popup menu.

notebook a `<gtk-notebook>`

`gtk-notebook-get-current-page` (*self* `<gtk-notebook>`) [Function]
⇒ (*ret* `int`)
`get-current-page` [Method]

Returns the page number of the current page.

notebook a `<gtk-notebook>`

ret the index (starting from 0) of the current page in the notebook. If the notebook has no pages, then -1 will be returned.

`gtk-notebook-get-menu-label` (*self* `<gtk-notebook>`) [Function]
(*child* `<gtk-widget>`) ⇒ (*ret* `<gtk-widget>`)
`get-menu-label` [Method]

Retrieves the menu label widget of the page containing *child*.

notebook a `<gtk-notebook>`

child a widget contained in a page of *notebook*

ret the menu label, or '`#f`' if the notebook page does not have a menu label other than the default (the tab label).

`gtk-notebook-get-nth-page` (*self* `<gtk-notebook>`) (*page_num* `int`) [Function]
⇒ (*ret* `<gtk-widget>`)
`get-nth-page` [Method]

Returns the child widget contained in page number *page-num*.

notebook a `<gtk-notebook>`

page-num the index of a page in the noteobok, or -1 to get the last page.

ret the child widget, or '`#f`' if *page-num* is out of bounds.

`gtk-notebook-get-n-pages` (*self* `<gtk-notebook>`) ⇒ (*ret* `int`) [Function]
`get-n-pages` [Method]

Gets the number of pages in a notebook.

notebook a `<gtk-notebook>`

ret the number of pages in the notebook.

Since 2.2

`gtk-notebook-get-tab-label` (*self* `<gtk-notebook>`) [Function]
(*child* `<gtk-widget>`) ⇒ (*ret* `<gtk-widget>`)
`get-tab-label` [Method]

Returns the tab label widget for the page *child*. '`#f`' is returned if *child* is not in *notebook* or if no tab label has specifically been set for *child*.

notebook a `<gtk-notebook>`

child the page

ret the tab label

`gtk-notebook-set-menu-label` (*self* `<gtk-notebook>`) [Function]
(*child* `<gtk-widget>`) (*menu_label* `<gtk-widget>`)
`set-menu-label` [Method]

Changes the menu label for the page containing *child*.

notebook a `<gtk-notebook>`

child the child widget

menu-label
the menu label, or NULL for default

`gtk-notebook-set-menu-label-text` (*self* `<gtk-notebook>`) [Function]
(*child* `<gtk-widget>`) (*menu_text* `mchars`)
`set-menu-label-text` [Method]

Creates a new label and sets it as the menu label of *child*.

notebook a `<gtk-notebook>`

child the child widget

menu-text the label text

`gtk-notebook-set-tab-label` (*self* `<gtk-notebook>`) [Function]
(*child* `<gtk-widget>`) (*tab_label* `<gtk-widget>`)
`set-tab-label` [Method]

Changes the tab label for *child*. If '`#f`' is specified for *tab-label*, then the page will have the label 'page N'.

notebook a `<gtk-notebook>`

child the page

tab-label the tab label widget to use, or '`#f`' for default tab label.

`gtk-notebook-set-tab-label-packing` (*self* `<gtk-notebook>`) [Function]
(*child* `<gtk-widget>`) (*expand* `bool`) (*fill* `bool`)
(*pack_type* `<gtk-pack-type>`)
`set-tab-label-packing` [Method]

Sets the packing parameters for the tab label of the page containing *child*. See `gtk-box-pack-start` for the exact meaning of the parameters.

notebook a `<gtk-notebook>`

child the child widget

expand whether to expand the bookmark or not

fill whether the bookmark should fill the allocated area or not

pack-type the position of the bookmark

`gtk-notebook-set-tab-label-text` (*self* `<gtk-notebook>`) [Function]
(*child* `<gtk-widget>`) (*tab_text* `mchars`)
`set-tab-label-text` [Method]

Creates a new label and sets it as the tab label for the page containing *child*.

notebook a `<gtk-notebook>`

child the page

tab-text the label text

`gtk-notebook-set-tab-reorderable` (*self* `<gtk-notebook>`) [Function]
(*child* `<gtk-widget>`) (*reorderable* `bool`)
`set-tab-reorderable` [Method]

Sets whether the notebook tab can be reordered via drag and drop or not.

notebook a `<gtk-notebook>`

child a child <gtk-widget>

reorderable
whether the tab is reorderable or not.

Since 2.10

gtk-notebook-set-tab-detachable (*self* <gtk-notebook>) (*child* <gtk-widget>) (*detachable* bool) [Function]

set-tab-detachable [Method]

Sets whether the tab can be detached from *notebook* to another notebook or widget.

Note that 2 notebooks must share a common group identificator (see gtk-notebook-set-group-id) to allow automatic tabs interchange between them.

If you want a widget to interact with a notebook through DnD (i.e.: accept dragged tabs from it) it must be set as a drop destination and accept the target "GTK_NOTEBOOK_TAB". The notebook will fill the selection with a GtkWidget** pointing to the child widget that corresponds to the dropped tab.

```
static void
on_drop_zone_drag_data_received (GtkWidget        *widget,
                                 GdkDragContext   *context,
                                 gint              x,
                                 gint              y,
                                 GtkSelectionData *selection_data,
                                 guint             info,
                                 guint             time,
                                 gpointer          user_data)
{
  GtkWidget *notebook;
  GtkWidget **child;

  notebook = gtk_drag_get_source_widget (context);
  child = (void*) selection_data->data;

  process_widget (*child);
  gtk_container_remove (GTK_CONTAINER (notebook), *child);
}
```

If you want a notebook to accept drags from other widgets, you will have to set your own DnD code to do it.

notebook a <gtk-notebook>

child a child <gtk-widget>

detachable
whether the tab is detachable or not

Since 2.10

`gtk-notebook-get-menu-label-text` (*self* `<gtk-notebook>`) [Function]
(*child* `<gtk-widget>`) ⇒ (*ret* `mchars`)
`get-menu-label-text` [Method]

Retrieves the text of the menu label for the page containing *child*.

notebook a `<gtk-notebook>`

child the child widget of a page of the notebook.

ret value: the text of the tab label, or '`#f`' if the widget does not have a menu label other than the default menu label, or the menu label widget is not a `<gtk-label>`. The string is owned by the widget and must not be freed.

`gtk-notebook-get-scrollable` (*self* `<gtk-notebook>`) ⇒ (*ret* `bool`) [Function]
`get-scrollable` [Method]

Returns whether the tab label area has arrows for scrolling. See `gtk-notebook-set-scrollable`.

notebook a `<gtk-notebook>`

ret '`#t`' if arrows for scrolling are present

`gtk-notebook-get-show-border` (*self* `<gtk-notebook>`) [Function]
⇒ (*ret* `bool`)
`get-show-border` [Method]

Returns whether a bevel will be drawn around the notebook pages. See `gtk-notebook-set-show-border`.

notebook a `<gtk-notebook>`

ret '`#t`' if the bevel is drawn

`gtk-notebook-get-show-tabs` (*self* `<gtk-notebook>`) ⇒ (*ret* `bool`) [Function]
`get-show-tabs` [Method]

Returns whether the tabs of the notebook are shown. See `gtk-notebook-set-show-tabs`.

notebook a `<gtk-notebook>`

ret '`#t`' if the tabs are shown

`gtk-notebook-get-tab-label-text` (*self* `<gtk-notebook>`) [Function]
(*child* `<gtk-widget>`) ⇒ (*ret* `mchars`)
`get-tab-label-text` [Method]

Retrieves the text of the tab label for the page containing *child*.

notebook a `<gtk-notebook>`

child a widget contained in a page of *notebook*

ret value: the text of the tab label, or '`#f`' if the tab label widget is not a `<gtk-label>`. The string is owned by the widget and must not be freed.

`gtk-notebook-get-tab-pos` (*self* `<gtk-notebook>`) [Function]
⇒ (*ret* `<gtk-position-type>`)
`get-tab-pos` [Method]

Gets the edge at which the tabs for switching pages in the notebook are drawn.

notebook a `<gtk-notebook>`

ret the edge at which the tabs are drawn

`gtk-notebook-get-tab-reorderable` (*self* `<gtk-notebook>`) [Function]
(*child* `<gtk-widget>`) ⇒ (*ret* `bool`)
`get-tab-reorderable` [Method]

Gets whether the tab can be reordered via drag and drop or not.

notebook a `<gtk-notebook>`

child a child `<gtk-widget>`

ret '`#t`' if the tab is reorderable.

Since 2.10

`gtk-notebook-get-tab-detachable` (*self* `<gtk-notebook>`) [Function]
(*child* `<gtk-widget>`) ⇒ (*ret* `bool`)
`get-tab-detachable` [Method]

Returns whether the tab contents can be detached from *notebook*.

notebook a `<gtk-notebook>`

child a child `<gtk-widget>`

ret TRUE if the tab is detachable.

Since 2.10

`gtk-notebook-set-current-page` (*self* `<gtk-notebook>`) [Function]
(*page_num* `int`)
`set-current-page` [Method]

Switches to the page number *page-num*.

Note that due to historical reasons, GtkNotebook refuses to switch to a page unless the child widget is visible. Therefore, it is recommended to show child widgets before adding them to a notebook.

notebook a `<gtk-notebook>`

page-num index of the page to switch to, starting from 0. If negative, the last page will be used. If greater than the number of pages in the notebook, nothing will be done.

`gtk-notebook-set-group-id` (*self* `<gtk-notebook>`) (*group_id* `int`) [Function]
`set-group-id` [Method]

Sets an group identificator for *notebook*, notebooks sharing the same group identificator will be able to exchange tabs via drag and drop. A notebook with group identificator -1 will not be able to exchange tabs with any other notebook.

notebook a `<gtk-notebook>`

group-id a group identificator, or -1 to unset it

Since 2.10

`gtk-notebook-get-group-id` (*self* `<gtk-notebook>`) ⇒ (*ret* `int`) [Function]
`get-group-id` [Method]

Gets the current group identificator for *notebook*.

notebook a `<gtk-notebook>`

ret the group identificator, or -1 if none is set.

Since 2.10

101 GtkTable

Pack widgets in regular patterns

101.1 Overview

The `<gtk-table>` functions allow the programmer to arrange widgets in rows and columns, making it easy to align many widgets next to each other, horizontally and vertically.

Tables are created with a call to `gtk-table-new`, the size of which can later be changed with `gtk-table-resize`.

Widgets can be added to a table using `gtk-table-attach` or the more convenient (but slightly less flexible) `gtk-table-attach-defaults`.

To alter the space next to a specific row, use `gtk-table-set-row-spacing`, and for a column, `gtk-table-set-col-spacing`.

The gaps between *all* rows or columns can be changed by calling `gtk-table-set-row-spacings` or `gtk-table-set-col-spacings` respectively.

`gtk-table-set-homogeneous`, can be used to set whether all cells in the table will resize themselves to the size of the largest widget in the table.

101.2 Usage

`<gtk-table>` [Class]

This `<gobject>` class defines the following properties:

`n-rows` The number of rows in the table

`n-columns`
The number of columns in the table

`column-spacing`
The amount of space between two consecutive columns

`row-spacing`
The amount of space between two consecutive rows

`homogeneous`
If TRUE, the table cells are all the same width/height

`gtk-table-new` (*rows* `unsigned-int`) (*columns* `unsigned-int`) (*homogeneous* `bool`) ⇒ (*ret* `<gtk-widget>`) [Function]

Used to create a new table widget. An initial size must be given by specifying how many rows and columns the table should have, although this can be changed later with `gtk-table-resize`. *rows* and *columns* must both be in the range 0 .. 65535.

rows The number of rows the new table should have.

columns The number of columns the new table should have.

homogeneous
If set to '`#t`', all table cells are resized to the size of the cell containing the largest widget.

ret A pointer to the the newly created table widget.

`gtk-table-resize` (*self* `<gtk-table>`) (*rows* `unsigned-int`) [Function]
(*columns* `unsigned-int`)
`resize` [Method]

If you need to change a table's size *after* it has been created, this function allows you to do so.

table The `<gtk-table>` you wish to change the size of.

rows The new number of rows.

columns The new number of columns.

`gtk-table-attach` (*self* `<gtk-table>`) (*child* `<gtk-widget>`) [Function]
(*left_attach* `unsigned-int`) (*right_attach* `unsigned-int`)
(*top_attach* `unsigned-int`) (*bottom_attach* `unsigned-int`)
(*xoptions* `<gtk-attach-options>`) (*yoptions* `<gtk-attach-options>`)
(*xpadding* `unsigned-int`) (*ypadding* `unsigned-int`)
`attach` [Method]

Adds a widget to a table. The number of 'cells' that a widget will occupy is specified by *left-attach*, *right-attach*, *top-attach* and *bottom-attach*. These each represent the leftmost, rightmost, uppermost and lowest column and row numbers of the table. (Columns and rows are indexed from zero).

table The `<gtk-table>` to add a new widget to.

child The widget to add.

left-attach the column number to attach the left side of a child widget to.

right-attach
the column number to attach the right side of a child widget to.

top-attach the row number to attach the top of a child widget to.

bottom-attach
the row number to attach the bottom of a child widget to.

xoptions Used to specify the properties of the child widget when the table is resized.

yoptions The same as xoptions, except this field determines behaviour of vertical resizing.

xpadding An integer value specifying the padding on the left and right of the widget being added to the table.

ypadding The amount of padding above and below the child widget.

`gtk-table-attach-defaults` (*self* `<gtk-table>`) [Function]
(*widget* `<gtk-widget>`) (*left_attach* `unsigned-int`)
(*right_attach* `unsigned-int`) (*top_attach* `unsigned-int`)
(*bottom_attach* `unsigned-int`)
`attach-defaults` [Method]

As there are many options associated with `gtk-table-attach`, this convenience function provides the programmer with a means to add children to a table with identical padding and expansion options. The values used for the `<gtk-attach-options>` are '`GTK_EXPAND | GTK_FILL`', and the padding is set to 0.

table The table to add a new child widget to.

widget The child widget to add.

left-attach The column number to attach the left side of the child widget to.

right-attach
The column number to attach the right side of the child widget to.

top-attach The row number to attach the top of the child widget to.

bottom-attach
The row number to attach the bottom of the child widget to.

`gtk-table-set-row-spacing` (*self* `<gtk-table>`) [Function]
(*row* `unsigned-int`) (*spacing* `unsigned-int`)
`set-row-spacing` [Method]
Changes the space between a given table row and the subsequent row.

table a `<gtk-table>` containing the row whose properties you wish to change.

row row number whose spacing will be changed.

spacing number of pixels that the spacing should take up.

`gtk-table-set-col-spacing` (*self* `<gtk-table>`) [Function]
(*column* `unsigned-int`) (*spacing* `unsigned-int`)
`set-col-spacing` [Method]
Alters the amount of space between a given table column and the following column.

table a `<gtk-table>`.

column the column whose spacing should be changed.

spacing number of pixels that the spacing should take up.

`gtk-table-set-row-spacings` (*self* `<gtk-table>`) [Function]
(*spacing* `unsigned-int`)
`set-row-spacings` [Method]
Sets the space between every row in *table* equal to *spacing*.

table a `<gtk-table>`.

spacing the number of pixels of space to place between every row in the table.

`gtk-table-set-col-spacings` (*self* `<gtk-table>`) [Function]
(*spacing* `unsigned-int`)
`set-col-spacings` [Method]
Sets the space between every column in *table* equal to *spacing*.

table a `<gtk-table>`.

spacing the number of pixels of space to place between every column in the table.

`gtk-table-set-homogeneous` (*self* `<gtk-table>`) (*homogeneous* `bool`) [Function]
`set-homogeneous` [Method]
Changes the homogenous property of table cells, ie. whether all cells are an equal size or not.

table The `<gtk-table>` you wish to set the homogeneous properties of.

homogeneous
Set to '`#t`' to ensure all table cells are the same size. Set to '`#f`' if this is not your desired behaviour.

`gtk-table-get-default-row-spacing` (*self* `<gtk-table>`) ⇒ (*ret* `unsigned-int`) [Function]
`get-default-row-spacing` [Method]
Gets the default row spacing for the table. This is the spacing that will be used for newly added rows. (See `gtk-table-set-row-spacings`)

table a `<gtk-table>`

ret value: the default row spacing

`gtk-table-get-homogeneous` (*self* `<gtk-table>`) ⇒ (*ret* `bool`) [Function]
`get-homogeneous` [Method]
Returns whether the table cells are all constrained to the same width and height. (See `gtk-table-set-homogenous`)

table a `<gtk-table>`

ret '`#t`' if the cells are all constrained to the same size

`gtk-table-get-row-spacing` (*self* `<gtk-table>`) (*row* `unsigned-int`) ⇒ (*ret* `unsigned-int`) [Function]
`get-row-spacing` [Method]
Gets the amount of space between row *row*, and row *row* + 1. See `gtk-table-set-row-spacing`.

table a `<gtk-table>`

row a row in the table, 0 indicates the first row

ret the row spacing

`gtk-table-get-col-spacing` (*self* `<gtk-table>`) (*column* `unsigned-int`) ⇒ (*ret* `unsigned-int`) [Function]
`get-col-spacing` [Method]
Gets the amount of space between column *col*, and column *col* + 1. See `gtk-table-set-col-spacing`.

table a `<gtk-table>`

column a column in the table, 0 indicates the first column

ret the column spacing

`gtk-table-get-default-col-spacing` (*self* `<gtk-table>`) ⇒ (*ret* `unsigned-int`) [Function]
`get-default-col-spacing` [Method]
Gets the default column spacing for the table. This is the spacing that will be used for newly added columns. (See `gtk-table-set-col-spacings`)

table a `<gtk-table>`

ret value: the default column spacing

102 GtkExpander

A container which can hide its child

102.1 Overview

A <gtk-expander> allows the user to hide or show its child by clicking on an expander triangle similar to the triangles used in a <gtk-tree-view>.

Normally you use an expander as you would use any other descendant of <gtk-bin>; you create the child widget and use gtk-container-add to add it to the expander. When the expander is toggled, it will take care of showing and hiding the child automatically.

There are situations in which you may prefer to show and hide the expanded widget yourself, such as when you want to actually create the widget at expansion time. In this case, create a <gtk-expander> but do not add a child to it. The expander widget has an 'expanded' property which can be used to monitor its expansion state. You should watch this property with a signal connection as follows:

```
expander = gtk_expander_new_with_mnemonic ("_More Options");
g_signal_connect (expander, "notify::expanded",
                  G_CALLBACK (expander_callback), NULL);

...

static void
expander_callback (GObject    *object,
                   GParamSpec *param_spec,
                   gpointer    user_data)
{
  GtkExpander *expander;

  expander = GTK_EXPANDER (object);

  if (gtk_expander_get_expanded (expander))
    {
      /* Show or create widgets */
    }
  else
    {
      /* Hide or destroy widgets */
    }
}
```

102.2 Usage

`<gtk-expander>` [Class]

This `<gobject>` class defines the following properties:

`expanded` Whether the expander has been opened to reveal the child widget

`label` Text of the expander's label

`use-underline`
If set, an underline in the text indicates the next character should be used for the mnemonic accelerator key

`use-markup`
The text of the label includes XML markup. See pango_parse_markup()

`spacing` Space to put between the label and the child

`label-widget`
A widget to display in place of the usual expander label

`activate` [Signal on `<gtk-expander>`]

`gtk-expander-new` (*label* `mchars`) ⇒ (*ret* `<gtk-widget>`) [Function]

Creates a new expander using *label* as the text of the label.

label the text of the label

ret a new `<gtk-expander>` widget.

Since 2.4

`gtk-expander-new-with-mnemonic` (*label* `mchars`) [Function]
⇒ (*ret* `<gtk-widget>`)

Creates a new expander using *label* as the text of the label. If characters in *label* are preceded by an underscore, they are underlined. If you need a literal underscore character in a label, use '__' (two underscores). The first underlined character represents a keyboard accelerator called a mnemonic. Pressing Alt and that key activates the button.

label the text of the label with an underscore in front of the mnemonic character

ret a new `<gtk-expander>` widget.

Since 2.4

`gtk-expander-set-expanded` (*self* `<gtk-expander>`) (*expanded* `bool`) [Function]
`set-expanded` [Method]

Sets the state of the expander. Set to '`#t`', if you want the child widget to be revealed, and '`#f`' if you want the child widget to be hidden.

expander a `<gtk-expander>`

expanded whether the child widget is revealed

Since 2.4

`gtk-expander-get-expanded` (*self* `<gtk-expander>`) ⇒ (*ret* `bool`) [Function]
`get-expanded` [Method]

Queries a `<gtk-expander>` and returns its current state. Returns '`#t`' if the child widget is revealed.

See `gtk-expander-set-expanded`.

expander a `<gtk-expander>`

ret the current state of the expander.

Since 2.4

`gtk-expander-set-spacing` (*self* `<gtk-expander>`) (*spacing* `int`) [Function]
`set-spacing` [Method]

Sets the spacing field of *expander*, which is the number of pixels to place between expander and the child.

expander a `<gtk-expander>`

spacing distance between the expander and child in pixels.

Since 2.4

`gtk-expander-get-spacing` (*self* `<gtk-expander>`) ⇒ (*ret* `int`) [Function]
`get-spacing` [Method]

Gets the value set by `gtk-expander-set-spacing`.

expander a `<gtk-expander>`

ret spacing between the expander and child.

Since 2.4

`gtk-expander-set-label` (*self* `<gtk-expander>`) (*label* `mchars`) [Function]
`set-label` [Method]

Sets the text of the label of the expander to *label*.

This will also clear any previously set labels.

expander a `<gtk-expander>`

label a string

Since 2.4

`gtk-expander-get-label` (*self* `<gtk-expander>`) ⇒ (*ret* `mchars`) [Function]
`get-label` [Method]

Fetches the text from the label of the expander, as set by `gtk-expander-set-label`. If the label text has not been set the return value will be '`#f`'. This will be the case if you create an empty button with `gtk-button-new` to use as a container.

expander a `<gtk-expander>`

ret The text of the label widget. This string is owned by the widget and must not be modified or freed.

Since 2.4

`gtk-expander-set-use-underline` (*self* `<gtk-expander>`) (*use_underline* `bool`) [Function]
`set-use-underline` [Method]

If true, an underline in the text of the expander label indicates the next character should be used for the mnemonic accelerator key.

expander a `<gtk-expander>`

use-underline '`#t`' if underlines in the text indicate mnemonics

Since 2.4

`gtk-expander-get-use-underline` (*self* `<gtk-expander>`) ⇒ (*ret* `bool`) [Function]
`get-use-underline` [Method]

Returns whether an embedded underline in the expander label indicates a mnemonic. See `gtk-expander-set-use-underline`.

expander a `<gtk-expander>`

ret '`#t`' if an embedded underline in the expander label indicates the mnemonic accelerator keys.

Since 2.4

`gtk-expander-set-use-markup` (*self* `<gtk-expander>`) (*use_markup* `bool`) [Function]
`set-use-markup` [Method]

Sets whether the text of the label contains markup in Pango's text markup language. See `gtk-label-set-markup`.

expander a `<gtk-expander>`

use-markup '`#t`' if the label's text should be parsed for markup

Since 2.4

`gtk-expander-get-use-markup` (*self* `<gtk-expander>`) ⇒ (*ret* `bool`) [Function]
`get-use-markup` [Method]

Returns whether the label's text is interpreted as marked up with the Pango text markup language. See `gtk-expander-set-use-markup`.

expander a `<gtk-expander>`

ret '`#t`' if the label's text will be parsed for markup

Since 2.4

`gtk-expander-set-label-widget` (*self* `<gtk-expander>`) (*label_widget* `<gtk-widget>`) [Function]
`set-label-widget` [Method]

Set the label widget for the expander. This is the widget that will appear embedded alongside the expander arrow.

expander a `<gtk-expander>`

label-widget
the new label widget

Since 2.4

`gtk-expander-get-label-widget` (*self* `<gtk-expander>`) [Function]
⇒ (*ret* `<gtk-widget>`)
`get-label-widget` [Method]

Retrieves the label widget for the frame. See `gtk-expander-set-label-widget`.

expander a `<gtk-expander>`

ret the label widget, or '`#f`' if there is none.

Since 2.4

103 GtkFrame

A bin with a decorative frame and optional label

103.1 Overview

The frame widget is a Bin that surrounds its child with a decorative frame and an optional label. If present, the label is drawn in a gap in the top side of the frame. The position of the label can be controlled with `gtk-frame-set-label-align`.

103.2 Usage

`<gtk-frame>` [Class]

This `<gobject>` class defines the following properties:

`label` Text of the frame's label

`label-xalign`
The horizontal alignment of the label

`label-yalign`
The vertical alignment of the label

`shadow` Deprecated property, use shadow_type instead

`shadow-type`
Appearance of the frame border

`label-widget`
A widget to display in place of the usual frame label

`gtk-frame-new` (*label* `mchars`) ⇒ (*ret* `<gtk-widget>`) [Function]

Creates a new `<gtk-frame>`, with optional label *label*. If *label* is '`#f`', the label is omitted.

label the text to use as the label of the frame

ret a new `<gtk-frame>` widget

`gtk-frame-set-label` (*self* `<gtk-frame>`) (*label* `mchars`) [Function]
`set-label` [Method]

Sets the text of the label. If *label* is '`#f`', the current label is removed.

frame a `<gtk-frame>`

label the text to use as the label of the frame

`gtk-frame-set-label-widget` (*self* `<gtk-frame>`) [Function]
(*label_widget* `<gtk-widget>`)
`set-label-widget` [Method]

Sets the label widget for the frame. This is the widget that will appear embedded in the top edge of the frame as a title.

frame a `<gtk-frame>`

label-widget
the new label widget

`gtk-frame-set-label-align` (*self* `<gtk-frame>`) (*xalign* `float`) (*yalign* `float`) [Function]
`set-label-align` [Method]

Sets the alignment of the frame widget's label. The default values for a newly created frame are 0.0 and 0.5.

frame a `<gtk-frame>`

xalign The position of the label along the top edge of the widget. A value of 0.0 represents left alignment; 1.0 represents right alignment.

yalign The y alignment of the label. A value of 0.0 aligns under the frame; 1.0 aligns above the frame.

`gtk-frame-set-shadow-type` (*self* `<gtk-frame>`) (*type* `<gtk-shadow-type>`) [Function]
`set-shadow-type` [Method]

Sets the shadow type for *frame*.

frame a `<gtk-frame>`

type the new `<gtk-shadow-type>`

`gtk-frame-get-label` (*self* `<gtk-frame>`) ⇒ (*ret* `mchars`) [Function]
`get-label` [Method]

If the frame's label widget is a `<gtk-label>`, returns the text in the label widget. (The frame will have a `<gtk-label>` for the label widget if a non-'`#f`' argument was passed to `gtk-frame-new`.)

frame a `<gtk-frame>`

ret the text in the label, or '`#f`' if there was no label widget or the lable widget was not a `<gtk-label>`. This string is owned by GTK+ and must not be modified or freed.

`gtk-frame-get-label-align` (*self* `<gtk-frame>`) ⇒ (*xalign* `float`) (*yalign* `float`) [Function]
`get-label-align` [Method]

Retrieves the X and Y alignment of the frame's label. See `gtk-frame-set-label-align`.

frame a `<gtk-frame>`

xalign location to store X alignment of frame's label, or '`#f`'

yalign location to store X alignment of frame's label, or '`#f`'

`gtk-frame-get-label-widget` (*self* `<gtk-frame>`) ⇒ (*ret* `<gtk-widget>`) [Function]
`get-label-widget` [Method]

Retrieves the label widget for the frame. See `gtk-frame-set-label-widget`.

frame a `<gtk-frame>`

ret the label widget, or '`#f`' if there is none.

`gtk-frame-get-shadow-type` (*self* `<gtk-frame>`) [Function]
⇒ (*ret* `<gtk-shadow-type>`)

`get-shadow-type` [Method]

Retrieves the shadow type of the frame. See `gtk-frame-set-shadow-type`.

frame a `<gtk-frame>`

ret the current shadow type of the frame.

104 GtkHSeparator

A horizontal separator

104.1 Overview

The <gtk-hseparator> widget is a horizontal separator, used to group the widgets within a window. It displays a horizontal line with a shadow to make it appear sunken into the interface.

The <gtk-hseparator> widget is not used as a separator within menus. To create a separator in a menu create an empty <gtk-separator-menu-item> widget using gtk-separator-menu-item-new and add it to the menu with gtk-menu-shell-append.

104.2 Usage

<gtk-hseparator> [Class]
This <gobject> class defines no properties, other than those defined by its superclasses.

gtk-hseparator-new ⇒ (*ret* <gtk-widget>) [Function]
Creates a new <gtk-hseparator>.

ret a new <gtk-hseparator>.

105 GtkVSeparator

A vertical separator

105.1 Overview

The <gtk-vseparator> widget is a vertical separator, used to group the widgets within a window. It displays a vertical line with a shadow to make it appear sunken into the interface.

105.2 Usage

<gtk-vseparator> [Class]

This <gobject> class defines no properties, other than those defined by its superclasses.

gtk-vseparator-new ⇒ (*ret* <gtk-widget>) [Function]

Creates a new <gtk-vseparator>.

ret a new <gtk-vseparator>.

106 GtkHScrollbar

A horizontal scrollbar

106.1 Overview

The <gtk-hscrollbar> widget is a widget arranged horizontally creating a scrollbar. See <gtk-scrollbar> for details on scrollbars. <gtk-adjustment> pointers may be added to handle the adjustment of the scrollbar or it may be left '#f' in which case one will be created for you. See <gtk-adjustment> for details.

106.2 Usage

<gtk-hscrollbar> [Class]

This <gobject> class defines no properties, other than those defined by its superclasses.

gtk-hscrollbar-new (*adjustment* <gtk-adjustment>) [Function]
⇒ (*ret* <gtk-widget>)

Creates a new horizontal scrollbar.

adjustment
the <gtk-adjustment> to use, or '#f' to create a new adjustment.

ret the new <gtk-hscrollbar>.

107 GtkVScrollbar

A vertical scrollbar

107.1 Overview

The <gtk-vscrollbar> widget is a widget arranged vertically creating a scrollbar. See <gtk-scrollbar> for details on scrollbars. <gtk-adjustment> pointers may be added to handle the adjustment of the scrollbar or it may be left '#f' in which case one will be created for you. See <gtk-adjustment> for details.

107.2 Usage

<gtk-vscrollbar> [Class]

This <gobject> class defines no properties, other than those defined by its superclasses.

gtk-vscrollbar-new (*adjustment* <gtk-adjustment>) [Function]
⇒ (*ret* <gtk-widget>)

Creates a new vertical scrollbar.

adjustment
the <gtk-adjustment> to use, or '#f' to create a new adjustment.

ret the new <gtk-vscrollbar>

108 GtkScrolledWindow

Adds scrollbars to its child widget

108.1 Overview

<gtk-scrolled-window> is a <gtk-bin> subclass: it's a container the accepts a single child widget. <gtk-scrolled-window> adds scrollbars to the child widget and optionally draws a beveled frame around the child widget.

The scrolled window can work in two ways. Some widgets have native scrolling support; these widgets have "slots" for <gtk-adjustment> objects. Widgets with native scroll support include <gtk-tree-view>, <gtk-text-view>, and <gtk-layout>.

The scrolled window installs <gtk-adjustment> objects in the child window's slots using the set_scroll_adjustments_signal, found in <gtk-widget-class>. (Conceptually, these widgets implement a "Scrollable" interface; because GTK+ 1.2 lacked interface support in the object system, this interface is hackily implemented as a signal in <gtk-widget-class>. The GTK+ 2.0 object system would allow a clean implementation, but it wasn't worth breaking the API.)

For widgets that lack native scrolling support, the <gtk-viewport> widget acts as an adaptor class, implementing scrollability for child widgets that lack their own scrolling capabilities. Use <gtk-viewport> to scroll child widgets such as <gtk-table>, <gtk-box>, and so on.

If a widget has native scrolling abilities, it can be added to the <gtk-scrolled-window> with gtk-container-add. If a widget does not, you must first add the widget to a <gtk-viewport>, then add the <gtk-viewport> to the scrolled window. The convenience function gtk-scrolled-window-add-with-viewport does exactly this, so you can ignore the presence of the viewport.

The position of the scrollbars is controlled by the scroll adjustments. See <gtk-adjustment> for the fields in an adjustment - for <gtk-scrollbar>, used by <gtk-scrolled-window>, the "value" field represents the position of the scrollbar, which must be between the "lower" field and "upper - page_size." The "page_size" field represents the size of the visible scrollable area. The "step_increment" and "page_increment" fields are used when the user asks to step down (using the small stepper arrows) or page down (using for example the PageDown key).

If a <gtk-scrolled-window> doesn't behave quite as you would like, or doesn't have exactly the right layout, it's very possible to set up your own scrolling with <gtk-scrollbar> and for example a <gtk-table>.

108.2 Usage

<gtk-scrolled-window> [Class]

This <gobject> class defines the following properties:

hadjustment
: The GtkAdjustment for the horizontal position

vadjustment
: The GtkAdjustment for the vertical position

`hscrollbar-policy`
When the horizontal scrollbar is displayed

`vscrollbar-policy`
When the vertical scrollbar is displayed

`window-placement`
Where the contents are located with respect to the scrollbars. This property only takes effect if "window-placement-set" is TRUE.

`window-placement-set`
Whether "window-placement" should be used to determine the location of the contents with respect to the scrollbars.

`shadow-type`
Style of bevel around the contents

`move-focus-out` [Signal on `<gtk-scrolled-window>`]
(*arg0* `<gtk-direction-type>`)

`scroll-child` (*arg0* `<gtk-scroll-type>`) [Signal on `<gtk-scrolled-window>`]
(*arg1* `<gboolean>`) ⇒ `<gboolean>`

`gtk-scrolled-window-new` (*hadjustment* `<gtk-adjustment>`) [Function]
(*vadjustment* `<gtk-adjustment>`) ⇒ (*ret* `<gtk-widget>`)
Creates a new scrolled window. The two arguments are the scrolled window's adjustments; these will be shared with the scrollbars and the child widget to keep the bars in sync with the child. Usually you want to pass '`#f`' for the adjustments, which will cause the scrolled window to create them for you.

hadjustment
Horizontal adjustment.

vadjustment
Vertical adjustment.

ret New scrolled window.

`gtk-scrolled-window-get-hadjustment` [Function]
(*self* `<gtk-scrolled-window>`) ⇒ (*ret* `<gtk-adjustment>`)
`get-hadjustment` [Method]
Returns the horizontal scrollbar's adjustment, used to connect the horizontal scrollbar to the child widget's horizontal scroll functionality.

scrolled-window
A `<gtk-scrolled-window>`.

ret The horizontal `<gtk-adjustment>`.

`gtk-scrolled-window-get-vadjustment` [Function]
(*self* `<gtk-scrolled-window>`) ⇒ (*ret* `<gtk-adjustment>`)
`get-vadjustment` [Method]
Returns the vertical scrollbar's adjustment, used to connect the vertical scrollbar to the child widget's vertical scroll functionality.

scrolled-window
A `<gtk-scrolled-window>`.

ret The vertical `<gtk-adjustment>`.

`gtk-scrolled-window-get-hscrollbar` [Function]
(*self* `<gtk-scrolled-window>`) ⇒ (*ret* `<gtk-widget>`)
`get-hscrollbar` [Method]
Returns the horizontal scrollbar of *scrolled-window*.

scrolled-window
a `<gtk-scrolled-window>`

ret the horizontal scrollbar of the scrolled window, or '`#f`' if it does not have one.

Since 2.8

`gtk-scrolled-window-get-vscrollbar` [Function]
(*self* `<gtk-scrolled-window>`) ⇒ (*ret* `<gtk-widget>`)
`get-vscrollbar` [Method]
Returns the vertical scrollbar of *scrolled-window*.

scrolled-window
a `<gtk-scrolled-window>`

ret the vertical scrollbar of the scrolled window, or '`#f`' if it does not have one.

Since 2.8

`gtk-scrolled-window-set-policy` (*self* `<gtk-scrolled-window>`) [Function]
(*hscrollbar_policy* `<gtk-policy-type>`)
(*vscrollbar_policy* `<gtk-policy-type>`)
`set-policy` [Method]
Sets the scrollbar policy for the horizontal and vertical scrollbars. The policy determines when the scrollbar should appear; it is a value from the `<gtk-policy-type>` enumeration. If '`GTK_POLICY_ALWAYS`', the scrollbar is always present; if '`GTK_POLICY_NEVER`', the scrollbar is never present; if '`GTK_POLICY_AUTOMATIC`', the scrollbar is present only if needed (that is, if the slider part of the bar would be smaller than the trough - the display is larger than the page size).

scrolled-window
A `<gtk-scrolled-window>`.

hscrollbar-policy
Policy for horizontal bar.

vscrollbar-policy
Policy for vertical bar.

`gtk-scrolled-window-set-placement` [Function]
(*self* `<gtk-scrolled-window>`) (*window_placement* `<gtk-corner-type>`)

`set-placement` [Method]

Sets the placement of the contents with respect to the scrollbars for the scrolled window.

See also `gtk-scrolled-window-get-placement` and `gtk-scrolled-window-unset-placement`.

Determines the location of the child widget with respect to the scrollbars. The default is '`GTK_CORNER_TOP_LEFT`', meaning the child is in the top left, with the scrollbars underneath and to the right. Other values in `<gtk-corner-type>` are '`GTK_CORNER_TOP_RIGHT`', '`GTK_CORNER_BOTTOM_LEFT`', and '`GTK_CORNER_BOTTOM_RIGHT`'.

scrolled-window
: a `<gtk-scrolled-window>`

window-placement
: Position of the child window.

`gtk-scrolled-window-unset-placement` [Function]
(*self* `<gtk-scrolled-window>`)

`unset-placement` [Method]

Unsets the placement of the contents with respect to the scrollbars for the scrolled window. If no window placement is set for a scrolled window, it obeys the "gtk-scrolled-window-placement" XSETTING.

See also `gtk-scrolled-window-set-placement` and `gtk-scrolled-window-get-placement`.

scrolled-window
: a `<gtk-scrolled-window>`

Since 2.10

`gtk-scrolled-window-set-shadow-type` [Function]
(*self* `<gtk-scrolled-window>`) (*type* `<gtk-shadow-type>`)

`set-shadow-type` [Method]

Changes the type of shadow drawn around the contents of *scrolled-window*.

scrolled-window
: a `<gtk-scrolled-window>`

type
: kind of shadow to draw around scrolled window contents

`gtk-scrolled-window-set-hadjustment` [Function]
(*self* `<gtk-scrolled-window>`) (*hadjustment* `<gtk-adjustment>`)

`set-hadjustment` [Method]

Sets the `<gtk-adjustment>` for the horizontal scrollbar.

scrolled-window
: A `<gtk-scrolled-window>`.

hadjustment
: Horizontal scroll adjustment.

`gtk-scrolled-window-set-vadjustment` [Function]
(*self* `<gtk-scrolled-window>`) (*vadjustment* `<gtk-adjustment>`)
`set-vadjustment` [Method]

Sets the `<gtk-adjustment>` for the vertical scrollbar.

scrolled-window
A `<gtk-scrolled-window>`.

vadjustment
Vertical scroll adjustment.

`gtk-scrolled-window-get-placement` [Function]
(*self* `<gtk-scrolled-window>`) ⇒ (*ret* `<gtk-corner-type>`)
`get-placement` [Method]

Gets the placement of the contents with respect to the scrollbars for the scrolled window. See `gtk-scrolled-window-set-placement`.

scrolled-window
a `<gtk-scrolled-window>`

ret the current placement value. See also `gtk-scrolled-window-set-placement` and `gtk-scrolled-window-unset-placement`.

`gtk-scrolled-window-get-policy` (*self* `<gtk-scrolled-window>`) [Function]
(*hscrollbar_policy* `<gtk-policy-type*>`)
(*vscrollbar_policy* `<gtk-policy-type*>`)
`get-policy` [Method]

Retrieves the current policy values for the horizontal and vertical scrollbars. See `gtk-scrolled-window-set-policy`.

scrolled-window
a `<gtk-scrolled-window>`

hscrollbar-policy
location to store the policy for the horizontal scrollbar, or '`#f`'.

vscrollbar-policy
location to store the policy for the horizontal scrollbar, or '`#f`'.

`gtk-scrolled-window-get-shadow-type` [Function]
(*self* `<gtk-scrolled-window>`) ⇒ (*ret* `<gtk-shadow-type>`)
`get-shadow-type` [Method]

Gets the shadow type of the scrolled window. See `gtk-scrolled-window-set-shadow-type`.

scrolled-window
a `<gtk-scrolled-window>`

ret the current shadow type

109 GtkPrintOperation

High-level Printing API

109.1 Overview

GtkPrintOperation is the high-level, portable printing API. It looks a bit different than other GTK+ dialogs such as the <gtk-file-chooser>, since some platforms don't expose enough infrastructure to implement a good print dialog. On such platforms, GtkPrintOperation uses the native print dialog. On platforms which do not provide a native print dialog, GTK+ uses its own, see <gtk-print-unix-dialog>.

The typical way to use the high-level printing API is to create a <gtk-print-operation> object with gtk-print-operation-new when the user selects to print. Then you set some properties on it, e.g. the page size, any <gtk-print-settings> from previous print operations, the number of pages, the current page, etc.

Then you start the print operation by calling gtk-print-operation-run. It will then show a dialog, let the user select a printer and options. When the user finished the dialog various signals will be emitted on the <gtk-print-operation>, the main one being ::draw-page, which you are supposed to catch and render the page on the provided <gtk-print-context> using Cairo.

```
static GtkPrintSettings *settings = NULL;

static void
do_print (void)
{
  GtkPrintOperation *print;
  GtkPrintOperationResult res;

  print = gtk_print_operation_new ();

  if (settings != NULL)
    gtk_print_operation_set_print_settings (print, settings);

  g_signal_connect (print, "begin_print", G_CALLBACK (begin_print), NULL);
  g_signal_connect (print, "draw_page", G_CALLBACK (draw_page), NULL);

  res = gtk_print_operation_run (print, GTK_PRINT_OPERATION_ACTION_PRINT_DIALOG,
                                 GTK_WINDOW (main_window), NULL);

  if (res == GTK_PRINT_OPERATION_RESULT_APPLY)
    {
      if (settings != NULL)
        g_object_unref (settings);
      settings = g_object_ref (gtk_print_operation_get_print_settings (print));
    }
```

```
  g_object_unref (print);
}
```

By default GtkPrintOperation uses an external application to do print preview. To implement a custom print preview, an application must connect to the preview signal. The functions `gtk-print-operation-print-preview-render-page`, `gtk-print-operation-preview-end-preview` and `gtk-print-operation-preview-is-selected` are useful when implementing a print preview.

Printing support was added in GTK+ 2.10.

109.2 Usage

`<gtk-print-operation>` [Class]

This `<gobject>` class defines the following properties:

`default-page-setup`
: The GtkPageSetup used by default

`print-settings`
: The GtkPrintSettings used for initializing the dialog

`job-name` A string used for identifying the print job.

`n-pages` The number of pages in the document.

`current-page`
: The current page in the document

`use-full-page`
: TRUE if the origin of the context should be at the corner of the page and not the corner of the imageable area

`track-print-status`
: TRUE if the print operation will continue to report on the print job status after the print data has been sent to the printer or print server.

`unit` The unit in which distances can be measured in the context

`show-progress`
: TRUE if a progress dialog is shown while printing.

`allow-async`
: TRUE if print process may run asynchronous.

`export-filename`
: Export filename

`status` The status of the print operation

`status-string`
: A human-readable description of the status

`custom-tab-label`
: Label for the tab containing custom widgets.

`done` (*arg0* `<gtk-print-operation-result>`) [Signal on `<gtk-print-operation>`]
Emitted when the print operation run has finished doing everything required for printing. *result* gives you information about what happened during the run. If *result* is '`GTK_PRINT_OPERATION_RESULT_ERROR`' then you can call `gtk-print-operation-get-error` for more information.

If you enabled print status tracking then `gtk-print-operation-is-finished` may still return '`#f`' after this was emitted.

Since 2.10

`begin-print` (*arg0* `<gtk-print-context>`) [Signal on `<gtk-print-operation>`]
Emitted after the user has finished changing print settings in the dialog, before the actual rendering starts.

A typical use for this signal is to use the parameters from the `<gtk-print-context>` and paginate the document accordingly, and then set the number of pages with `gtk-print-operation-set-n-pages`.

Since 2.10

`paginate` (*arg0* `<gtk-print-context>`) [Signal on `<gtk-print-operation>`]
⇒ `<gboolean>`
Emitted after the begin-print signal, but before the actual rendering starts. It keeps getting emitted until it returns '`#f`'.

This signal is intended to be used for paginating the document in small chunks, to avoid blocking the user interface for a long time. The signal handler should update the number of pages using `gtk-print-operation-set-n-pages`, and return '`#t`' if the document has been completely paginated.

If you don't need to do pagination in chunks, you can simply do it all in the begin-print handler, and set the number of pages from there.

Since 2.10

`request-page-setup` [Signal on `<gtk-print-operation>`]
(*arg0* `<gtk-print-context>`) (*arg1* `<gint>`) (*arg2* `<gtk-page-setup>`)
Emitted once for every page that is printed, to give the application a chance to modify the page setup. Any changes done to *setup* will be in force only for printing this page.

Since 2.10

`draw-page` (*arg0* `<gtk-print-context>`) [Signal on `<gtk-print-operation>`]
(*arg1* `<gint>`)
Emitted for every page that is printed. The signal handler must render the *page-nr*'s page onto the cairo context obtained from *context* using `gtk-print-context-get-cairo-context`.

```
static void
draw_page (GtkPrintOperation *operation,
           GtkPrintContext   *context,
           gint               page_nr,
           gpointer           user_data)
```

```
{
  cairo_t *cr;
  PangoLayout *layout;
  gdouble width, text_height;
  gint layout_height;
  PangoFontDescription *desc;

  cr = gtk_print_context_get_cairo_context (context);
  width = gtk_print_context_get_width (context);

  cairo_rectangle (cr, 0, 0, width, HEADER_HEIGHT);

  cairo_set_source_rgb (cr, 0.8, 0.8, 0.8);
  cairo_fill (cr);

  layout = gtk_print_context_create_pango_layout (context);

  desc = pango_font_description_from_string ("sans 14");
  pango_layout_set_font_description (layout, desc);
  pango_font_description_free (desc);

  pango_layout_set_text (layout, "some text", -1);
  pango_layout_set_width (layout, width);
  pango_layout_set_alignment (layout, PANGO_ALIGN_CENTER);

  pango_layout_get_size (layout, NULL, &layout_height);
  text_height = (gdouble)layout_height / PANGO_SCALE;

  cairo_move_to (cr, width / 2,  (HEADER_HEIGHT - text_height) / 2);
  pango_cairo_show_layout (cr, layout);

  g_object_unref (layout);
}
```

Use `gtk-print-operation-set-use-full-page` and `gtk-print-operation-set-unit` before starting the print operation to set up the transformation of the cairo context according to your needs.

Since 2.10

`end-print` (*arg0* `<gtk-print-context>`) [Signal on `<gtk-print-operation>`]

Emitted after all pages have been rendered. A handler for this signal can clean up any resources that have been allocated in the ::begin-print handler.

Since 2.10

`status-changed` [Signal on `<gtk-print-operation>`]

Emitted at between the various phases of the print operation. See `<gtk-print-status>` for the phases that are being discriminated. Use `gtk-print-operation-get-status` to find out the current status.

Since 2.10

`create-custom-widget` ⇒ `<gobject>` [Signal on `<gtk-print-operation>`]

Emitted when displaying the print dialog. If you return a widget in a handler for this signal it will be added to a custom tab in the print dialog. You typically return a container widget with multiple widgets in it.

The print dialog owns the returned widget, and its lifetime isn't controlled by the app. However, the widget is guaranteed to stay around until the custom-widget-apply signal is emitted on the operation. Then you can read out any information you need from the widgets.

Since 2.10

`custom-widget-apply` (*arg0* `<gtk-widget>`) [Signal on `<gtk-print-operation>`]

Emitted right before begin-print if you added a custom widget in the create-custom-widget handler. When you get this signal you should read the information from the custom widgets, as the widgets are not guaraneed to be around at a later time.

Since 2.10

`preview` [Signal on `<gtk-print-operation>`]
(*arg0* `<gtk-print-operation-preview>`) (*arg1* `<gtk-print-context>`) (*arg2* `<gtk-window>`) ⇒ `<gboolean>`

Gets emitted when a preview is requested from the native dialog.

The default handler for this signal uses an external viewer application to preview.

To implement a custom print preview, an application must return '`#t`' from its handler for this signal. In order to use the provided *context* for the preview implementation, it must be given a suitable cairo context with `gtk-print-context-set-cairo-context`.

The custom preview implementation can use `gtk-print-operation-preview-is-selected` and `gtk-print-operation-preview-render-page` to find pages which are selected for print and render them. The preview must be finished by calling `gtk-print-operation-preview-end-preview` (typically in response to the user clicking a close button).

Since 2.10

`gtk-print-operation-new` ⇒ (*ret* `<gtk-print-operation>`) [Function]

Creates a new `<gtk-print-operation>`.

ret a new `<gtk-print-operation>`

Since 2.10

`gtk-print-operation-set-allow-async` [Function]
(*self* `<gtk-print-operation>`) (*allow_async* `bool`)
`set-allow-async` [Method]

Sets whether the `gtk-print-operation-run` may return before the print operation is completed. Note that some platforms may not allow asynchronous operation.

op a `<gtk-print-operation>`

allow-async
: '`#t`' to allow asynchronous operation

Since 2.10

`gtk-print-operation-get-error` (*self* `<gtk-print-operation>`) [Function]
`get-error` [Method]

Call this when the result of a print operation is '`GTK_PRINT_OPERATION_RESULT_ERROR`', either as returned by `gtk-print-operation-run`, or in the ::done signal handler. The returned `<g-error>` will contain more details on what went wrong.

op a `<gtk-print-operation>`

error return location for the error

Since 2.10

`gtk-print-operation-set-job-name` (*self* `<gtk-print-operation>`) [Function]
(*job_name* `mchars`)
`set-job-name` [Method]

Sets the name of the print job. The name is used to identify the job (e.g. in monitoring applications like eggcups).

If you don't set a job name, GTK+ picks a default one by numbering successive print jobs.

op a `<gtk-print-operation>`

job-name a string that identifies the print job

Since 2.10

`gtk-print-operation-set-n-pages` (*self* `<gtk-print-operation>`) [Function]
(*n_pages* `int`)
`set-n-pages` [Method]

Sets the number of pages in the document.

This *must* be set to a positive number before the rendering starts. It may be set in a ::begin-print signal hander.

Note that the page numbers passed to the ::request-page-setup and ::draw-page signals are 0-based, i.e. if the user chooses to print all pages, the last ::draw-page signal will be for page *n-pages* - 1.

op a `<gtk-print-operation>`

n-pages the number of pages

Since 2.10

`gtk-print-operation-set-unit` (*self* `<gtk-print-operation>`) [Function]
(*unit* `<gtk-unit>`)
`set-unit` [Method]

Sets up the transformation for the cairo context obtained from `<gtk-print-context>` in such a way that distances are measured in units of *unit*.

op a `<gtk-print-operation>`

unit the unit to use

Since 2.10

`gtk-print-operation-run` (*self* `<gtk-print-operation>`) [Function]
(*action* `<gtk-print-operation-action>`) (*parent* `<gtk-window>`)
⇒ (*ret* `<gtk-print-operation-result>`)

`run` [Method]

Runs the print operation, by first letting the user modify print settings in the print dialog, and then print the document.

Normally that this function does not return until the rendering of all pages is complete. You can connect to the ::status-changed signal on *op* to obtain some information about the progress of the print operation. Furthermore, it may use a recursive mainloop to show the print dialog.

If you call `gtk-print-operation-set-allow-async` or set the allow-async property the operation will run asyncronously if this is supported on the platform. The ::done signal will be emitted with the operation results when the operation is done (i.e. when the dialog is canceled, or when the print succeeds or fails).

```
if (settings != NULL)
  gtk_print_operation_set_print_settings (print, settings);

if (page_setup != NULL)
  gtk_print_operation_set_default_page_setup (print, page_setup);

g_signal_connect (print, "begin-print",
                  G_CALLBACK (begin_print), &data);
g_signal_connect (print, "draw-page",
                  G_CALLBACK (draw_page), &data);

res = gtk_print_operation_run (print, GTK_PRINT_OPERATION_ACTION_PRINT_DIALOG, pa

if (res == GTK_PRINT_OPERATION_RESULT_ERROR)
 {
   error_dialog = gtk_message_dialog_new (GTK_WINDOW (parent),
                        GTK_DIALOG_DESTROY_WITH_PARENT,
     GTK_MESSAGE_ERROR,
     GTK_BUTTONS_CLOSE,
     "Error printing file:\n%s",
     error->message);
   g_signal_connect (error_dialog, "response",
                     G_CALLBACK (gtk_widget_destroy), NULL);
   gtk_widget_show (error_dialog);
   g_error_free (error);
 }
else if (res == GTK_PRINT_OPERATION_RESULT_APPLY)
 {
```

```
    if (settings != NULL)
  g_object_unref (settings);
    settings = g_object_ref (gtk_print_operation_get_print_settings (print));
 }
```

Note that `gtk-print-operation-run` can only be called once on a given `<gtk-print-operation>`.

op a `<gtk-print-operation>`

action the action to start

parent Transient parent of the dialog, or '`#f`'

error Return location for errors, or '`#f`'

ret the result of the print operation. A return value of '`GTK_PRINT_OPERATION_RESULT_APPLY`' indicates that the printing was completed successfully. In this case, it is a good idea to obtain the used print settings with `gtk-print-operation-get-print-settings` and store them for reuse with the next print operation. A value of '`GTK_PRINT_OPERATION_RESULT_IN_PROGRESS`' means the operation is running asynchronously, and will emit the ::done signal when done.

Since 2.10

`gtk-print-operation-cancel` (*self* `<gtk-print-operation>`) [Function]
`cancel` [Method]

Cancels a running print operation. This function may be called from a begin-print, paginate or draw-page signal handler to stop the currently running print operation.

op a `<gtk-print-operation>`

Since 2.10

`gtk-print-operation-get-status` (*self* `<gtk-print-operation>`) [Function]
⇒ (*ret* `<gtk-print-status>`)
`get-status` [Method]

Returns the status of the print operation. Also see `gtk-print-operation-get-status-string`.

op a `<gtk-print-operation>`

ret the status of the print operation

Since 2.10

`gtk-print-operation-is-finished` (*self* `<gtk-print-operation>`) [Function]
⇒ (*ret* `bool`)
`is-finished` [Method]

A convenience function to find out if the print operation is finished, either successfully ('`GTK_PRINT_STATUS_FINISHED`') or unsuccessfully ('`GTK_PRINT_STATUS_FINISHED_ABORTED`').

Note: when you enable print status tracking the print operation can be in a non-finished state even after done has been called, as the operation status then tracks the print job status on the printer.

op a `<gtk-print-operation>`

ret '`#t`', if the print operation is finished.

Since 2.10

`gtk-print-operation-get-error` (*self* `<gtk-print-operation>`) [Function]
`get-error` [Method]

Call this when the result of a print operation is '`GTK_PRINT_OPERATION_RESULT_ERROR`', either as returned by `gtk-print-operation-run`, or in the ::done signal handler. The returned `<g-error>` will contain more details on what went wrong.

op a `<gtk-print-operation>`

error return location for the error

Since 2.10

`gtk-print-run-page-setup-dialog` (*parent* `<gtk-window>`) [Function]
(*page_setup* `<gtk-page-setup*>`) (*settings* `<gtk-print-settings*>`)
⇒ (*ret* `<gtk-page-setup*>`)

Runs a page setup dialog, letting the user modify the values from *page-setup*. If the user cancels the dialog, the returned `<gtk-page-setup>` is identical to the passed in *page-setup*, otherwise it contains the modifications done in the dialog.

Note that this function may use a recursive mainloop to show the page setup dialog. See `gtk-print-run-page-setup-dialog-async` if this is a problem.

parent transient parent, or '`#f`'

page-setup an existing `<gtk-page-setup>`, or '`#f`'

settings a `<gtk-print-settings>`

ret a new `<gtk-page-setup>`

Since 2.10

110 GtkPrintContext

Encapsulates context for drawing pages

110.1 Overview

A GtkPrintContext encapsulates context information that is required when drawing pages for printing, such as the cairo context and important parameters like page size and resolution. It also lets you easily create <pango-layout> and <pango-context> objects that match the font metrics of the cairo surface.

GtkPrintContext objects gets passed to the ::begin-print, ::end-print, ::request-page-setup and ::draw-page signals on the <gtk-print-operation>.

```
static void
draw_page (GtkPrintOperation *operation,
   GtkPrintContext   *context,
   int                page_nr)
{
  cairo_t *cr;
  PangoLayout *layout;
  PangoFontDescription *desc;

  cr = gtk_print_context_get_cairo_context (context);

  /* Draw a red rectangle, as wide as the paper (inside the margins) */
  cairo_set_source_rgb (cr, 1.0, 0, 0);
  cairo_rectangle (cr, 0, 0, gtk_print_context_get_width (context), 50);

  cairo_fill (cr);

  /* Draw some lines */
  cairo_move_to (cr, 20, 10);
  cairo_line_to (cr, 40, 20);
  cairo_arc (cr, 60, 60, 20, 0, M_PI);
  cairo_line_to (cr, 80, 20);

  cairo_set_source_rgb (cr, 0, 0, 0);
  cairo_set_line_width (cr, 5);
  cairo_set_line_cap (cr, CAIRO_LINE_CAP_ROUND);
  cairo_set_line_join (cr, CAIRO_LINE_JOIN_ROUND);

  cairo_stroke (cr);

  /* Draw some text */
  layout = gtk_print_context_create_layout (context);
  pango_layout_set_text (layout, "Hello World! Printing is easy", -1);
  desc = pango_font_description_from_string ("sans 28");
```

```
  pango_layout_set_font_description (layout, desc);
  pango_font_description_free (desc);

  cairo_move_to (cr, 30, 20);
  pango_cairo_layout_path (cr, layout);

  /* Font Outline */
  cairo_set_source_rgb (cr, 0.93, 1.0, 0.47);
  cairo_set_line_width (cr, 0.5);
  cairo_stroke_preserve (cr);

  /* Font Fill */
  cairo_set_source_rgb (cr, 0, 0.0, 1.0);
  cairo_fill (cr);

  g_object_unref (layout);
}
```

Printing support was added in GTK+ 2.10.

110.2 Usage

`gtk-print-context-get-cairo-context` [Function]
(*self* `<gtk-print-context*>`) ⇒ (*ret* `cairo-t`)

Obtains the cairo context that is associated with the `<gtk-print-context>`.

context a `<gtk-print-context>`

ret the cairo context of *context*

Since 2.10

`gtk-print-context-set-cairo-context` [Function]
(*self* `<gtk-print-context*>`) (*cr* `cairo-t`) (*dpi_x* `double`) (*dpi_y* `double`)

Sets a new cairo context on a print context.

This function is intended to be used when implementing an internal print preview, it is not needed for printing, since GTK+ itself creates a suitable cairo context in that case.

context a `<gtk-print-context>`

cr the cairo context

dpi-x the horizontal resolution to use with *cr*

dpi-y the vertical resolution to use with *cr*

Since 2.10

`gtk-print-context-get-page-setup` (*self* `<gtk-print-context*>`) [Function]
⇒ (*ret* `<gtk-page-setup*>`)

Obtains the `<gtk-page-setup>` that determines the page dimensions of the `<gtk-print-context>`.

context a <gtk-print-context>

ret the page setup of *context*

Since 2.10

gtk-print-context-get-width (*self* <gtk-print-context*>) [Function]
⇒ (*ret* double)

Obtains the width of the <gtk-print-context>, in pixels.

context a <gtk-print-context>

ret the width of *context*

Since 2.10

gtk-print-context-get-height (*self* <gtk-print-context*>) [Function]
⇒ (*ret* double)

Obtains the height of the <gtk-print-context>, in pixels.

context a <gtk-print-context>

ret the height of *context*

Since 2.10

gtk-print-context-get-dpi-x (*self* <gtk-print-context*>) [Function]
⇒ (*ret* double)

Obtains the horizontal resolution of the <gtk-print-context>, in dots per inch.

context a <gtk-print-context>

ret the horizontal resolution of *context*

Since 2.10

gtk-print-context-get-dpi-y (*self* <gtk-print-context*>) [Function]
⇒ (*ret* double)

Obtains the vertical resolution of the <gtk-print-context>, in dots per inch.

context a <gtk-print-context>

ret the vertical resolution of *context*

Since 2.10

gtk-print-context-get-pango-fontmap [Function]
(*self* <gtk-print-context*>) ⇒ (*ret* <pango-font-map>)

Returns a <pango-font-map> that is suitable for use with the <gtk-print-context>.

context a <gtk-print-context>

ret the font map of *context*

Since 2.10

111 GtkPrintSettings

Stores print settings

111.1 Overview

A GtkPrintSettings object represents the settings of a print dialog in a system-independent way. The main use for this object is that once you've printed you can get a settings object that represents the settings the user chose, and the next time you print you can pass that object in so that the user doesn't have to re-set all his settings.

Its also possible to enumerate the settings so that you can easily save the settings for the next time your app runs, or even store them in a document. The predefined keys try to use shared values as much as possible so that moving such a document between systems still works.

Printing support was added in GTK+ 2.10.

111.2 Usage

`gtk-print-settings-new` ⇒ (*ret* `<gtk-print-settings*>`) [Function]

Creates a new `<gtk-print-settings>` object.

ret a new `<gtk-print-settings>` object

Since 2.10

`gtk-print-settings-has-key` (*self* `<gtk-print-settings*>`) [Function]
(*key* `mchars`) ⇒ (*ret* `bool`)

Returns '`#t`', if a value is associated with *key*.

settings a `<gtk-print-settings>`

key a key

ret '`#t`', if *key* has a value

Since 2.10

`gtk-print-settings-get` (*self* `<gtk-print-settings*>`) [Function]
(*key* `mchars`) ⇒ (*ret* `mchars`)

Looks up the string value associated with *key*.

settings a `<gtk-print-settings>`

key a key

ret the string value for *key*

Since 2.10

`gtk-print-settings-set` (*self* `<gtk-print-settings*>`) [Function]
(*key* `mchars`) (*value* `mchars`)

Associates *value* with *key*.

settings a `<gtk-print-settings>`

key a key

value a string value, or '`#f`'

Since 2.10

`gtk-print-settings-unset` (*self* `<gtk-print-settings*>`) [Function]
(*key* `mchars`)

Removes any value associated with *key*. This has the same effect as setting the value to '`#f`'.

settings a `<gtk-print-settings>`

key a key

Since 2.10

`gtk-print-settings-get-bool` (*self* `<gtk-print-settings*>`) [Function]
(*key* `mchars`) ⇒ (*ret* `bool`)

Returns the boolean represented by the value that is associated with *key*.

The string `"true"` represents '`#t`', any other string '`#f`'.

settings a `<gtk-print-settings>`

key a key

ret '`#t`', if *key* maps to a true value.

Since 2.10

`gtk-print-settings-set-bool` (*self* `<gtk-print-settings*>`) [Function]
(*key* `mchars`) (*value* `bool`)

Sets *key* to a boolean value.

settings a `<gtk-print-settings>`

key a key

value a boolean

Since 2.10

`gtk-print-settings-get-double` (*self* `<gtk-print-settings*>`) [Function]
(*key* `mchars`) ⇒ (*ret* `double`)

Returns the double value associated with *key*, or 0.

settings a `<gtk-print-settings>`

key a key

ret the double value of *key*

Since 2.10

`gtk-print-settings-set-double` (*self* `<gtk-print-settings*>`) [Function]
(*key* `mchars`) (*value* `double`)

Sets *key* to a double value.

settings a `<gtk-print-settings>`

key a key

value a double value

Since 2.10

`gtk-print-settings-get-length` (*self* `<gtk-print-settings*>`) [Function]
(*key* `mchars`) (*unit* `<gtk-unit>`) ⇒ (*ret* `double`)
Returns the value associated with *key*, interpreted as a length. The returned value is converted to *units*.

settings a `<gtk-print-settings>`

key a key

unit the unit of the return value

ret the length value of *key*, converted to *unit*

Since 2.10

`gtk-print-settings-set-length` (*self* `<gtk-print-settings*>`) [Function]
(*key* `mchars`) (*value* `double`) (*unit* `<gtk-unit>`)
Associates a length in units of *unit* with *key*.

settings a `<gtk-print-settings>`

key a key

value a length

unit the unit of *length*

Since 2.10

`gtk-print-settings-get-int` (*self* `<gtk-print-settings*>`) [Function]
(*key* `mchars`) ⇒ (*ret* `int`)
Returns the integer value of *key*, or 0.

settings a `<gtk-print-settings>`

key a key

ret the integer value of *key*

Since 2.10

`gtk-print-settings-set-int` (*self* `<gtk-print-settings*>`) [Function]
(*key* `mchars`) (*value* `int`)
Sets *key* to an integer value.

settings a `<gtk-print-settings>`

key a key

value an integer

Since 2.10

`gtk-print-settings-get-printer` (*self* `<gtk-print-settings*>`) [Function]
⇒ (*ret* `mchars`)

Convenience function to obtain the value of '`GTK_PRINT_SETTINGS_PRINTER`'.

settings a `<gtk-print-settings>`

ret the printer name

Since 2.10

`gtk-print-settings-set-printer` (*self* `<gtk-print-settings*>`) [Function]
(*printer* `mchars`)

Convenience function to set '`GTK_PRINT_SETTINGS_PRINTER`' to *printer*.

settings a `<gtk-print-settings>`

printer the printer name

Since 2.10

`gtk-print-settings-get-orientation` [Function]
(*self* `<gtk-print-settings*>`) ⇒ (*ret* `<gtk-page-orientation>`)

Get the value of '`GTK_PRINT_SETTINGS_ORIENTATION`', converted to a `<gtk-page-orientation>`.

settings a `<gtk-print-settings>`

ret the orientation

Since 2.10

`gtk-print-settings-set-orientation` [Function]
(*self* `<gtk-print-settings*>`) (*orientation* `<gtk-page-orientation>`)

Sets the value of '`GTK_PRINT_SETTINGS_ORIENTATION`'.

settings a `<gtk-print-settings>`

orientation
a page orientation

Since 2.10

`gtk-print-settings-get-paper-size` [Function]
(*self* `<gtk-print-settings*>`) ⇒ (*ret* `<gtk-paper-size*>`)

Gets the value of '`GTK_PRINT_SETTINGS_PAPER_FORMAT`', converted to a `<gtk-paper-size>`.

settings a `<gtk-print-settings>`

ret the paper size

Since 2.10

`gtk-print-settings-set-paper-size` [Function]
(*self* `<gtk-print-settings*>`) (*paper_size* `<gtk-paper-size*>`)

Sets the value of '`GTK_PRINT_SETTINGS_PAPER_FORMAT`', '`GTK_PRINT_SETTINGS_PAPER_WIDTH`' and '`GTK_PRINT_SETTINGS_PAPER_HEIGHT`'.

settings a `<gtk-print-settings>`

paper-size a paper size

Since 2.10

`gtk-print-settings-get-paper-width` [Function]
(*self* `<gtk-print-settings*>`) (*unit* `<gtk-unit>`) ⇒ (*ret* `double`)
Gets the value of '`GTK_PRINT_SETTINGS_PAPER_WIDTH`', converted to *unit*.

settings a `<gtk-print-settings>`

unit the unit for the return value

ret the paper width, in units of *unit*

Since 2.10

`gtk-print-settings-set-paper-width` [Function]
(*self* `<gtk-print-settings*>`) (*width* `double`) (*unit* `<gtk-unit>`)
Sets the value of '`GTK_PRINT_SETTINGS_PAPER_WIDTH`'.

settings a `<gtk-print-settings>`

width the paper width

unit the units of *width*

Since 2.10

`gtk-print-settings-get-paper-height` [Function]
(*self* `<gtk-print-settings*>`) (*unit* `<gtk-unit>`) ⇒ (*ret* `double`)
Gets the value of '`GTK_PRINT_SETTINGS_PAPER_HEIGHT`', converted to *unit*.

settings a `<gtk-print-settings>`

unit the unit for the return value

ret the paper height, in units of *unit*

Since 2.10

`gtk-print-settings-set-paper-height` [Function]
(*self* `<gtk-print-settings*>`) (*height* `double`) (*unit* `<gtk-unit>`)
Sets the value of '`GTK_PRINT_SETTINGS_PAPER_HEIGHT`'.

settings a `<gtk-print-settings>`

height the paper height

unit the units of *height*

Since 2.10

`gtk-print-settings-get-use-color` (*self* `<gtk-print-settings*>`) [Function]
⇒ (*ret* `bool`)
Gets the value of '`GTK_PRINT_SETTINGS_USE_COLOR`'.

settings a `<gtk-print-settings>`

ret whether to use color

Since 2.10

`gtk-print-settings-set-use-color` (*self* `<gtk-print-settings*>`) [Function]
(*use_color* `bool`)
Sets the value of '`GTK_PRINT_SETTINGS_USE_COLOR`'.

settings a `<gtk-print-settings>`

use-color whether to use color

Since 2.10

`gtk-print-settings-get-collate` (*self* `<gtk-print-settings*>`) [Function]
⇒ (*ret* `bool`)
Gets the value of '`GTK_PRINT_SETTINGS_COLLATE`'.

settings a `<gtk-print-settings>`

ret whether to collate the printed pages

Since 2.10

`gtk-print-settings-set-collate` (*self* `<gtk-print-settings*>`) [Function]
(*collate* `bool`)
Sets the value of '`GTK_PRINT_SETTINGS_COLLATE`'.

settings a `<gtk-print-settings>`

collate whether to collate the output

Since 2.10

`gtk-print-settings-get-reverse` (*self* `<gtk-print-settings*>`) [Function]
⇒ (*ret* `bool`)
Gets the value of '`GTK_PRINT_SETTINGS_REVERSE`'.

settings a `<gtk-print-settings>`

ret whether to reverse the order of the printed pages

Since 2.10

`gtk-print-settings-set-reverse` (*self* `<gtk-print-settings*>`) [Function]
(*reverse* `bool`)
Sets the value of '`GTK_PRINT_SETTINGS_REVERSE`'.

settings a `<gtk-print-settings>`

reverse whether to reverse the output

Since 2.10

`gtk-print-settings-get-duplex` (*self* `<gtk-print-settings*>`) [Function]
⇒ (*ret* `<gtk-print-duplex>`)
Gets the value of '`GTK_PRINT_SETTINGS_DUPLEX`'.

settings a `<gtk-print-settings>`

ret whether to print the output in duplex.

Since 2.10

`gtk-print-settings-set-duplex` (*self* `<gtk-print-settings*>`) [Function]
(*duplex* `<gtk-print-duplex>`)
Sets the value of '`GTK_PRINT_SETTINGS_DUPLEX`'.

settings a `<gtk-print-settings>`

duplex a `<gtk-print-duplex>` value

Since 2.10

`gtk-print-settings-get-quality` (*self* `<gtk-print-settings*>`) [Function]
⇒ (*ret* `<gtk-print-quality>`)
Gets the value of '`GTK_PRINT_SETTINGS_QUALITY`'.

settings a `<gtk-print-settings>`

ret the print quality

Since 2.10

`gtk-print-settings-set-quality` (*self* `<gtk-print-settings*>`) [Function]
(*quality* `<gtk-print-quality>`)
Sets the value of '`GTK_PRINT_SETTINGS_QUALITY`'.

settings a `<gtk-print-settings>`

quality a `<gtk-print-quality>` value

Since 2.10

`gtk-print-settings-get-n-copies` (*self* `<gtk-print-settings*>`) [Function]
⇒ (*ret* `int`)
Gets the value of '`GTK_PRINT_SETTINGS_N_COPIES`'.

settings a `<gtk-print-settings>`

ret the number of copies to print

Since 2.10

`gtk-print-settings-set-n-copies` (*self* `<gtk-print-settings*>`) [Function]
(*num_copies* `int`)
Sets the value of '`GTK_PRINT_SETTINGS_N_COPIES`'.

settings a `<gtk-print-settings>`

num-copies
the number of copies

Since 2.10

`gtk-print-settings-get-number-up` (*self* `<gtk-print-settings*>`) [Function]
⇒ (*ret* `int`)
Gets the value of '`GTK_PRINT_SETTINGS_NUMBER_UP`'.

settings a `<gtk-print-settings>`

ret the number of pages per sheet

Since 2.10

`gtk-print-settings-set-number-up` (*self* `<gtk-print-settings*>`) (*number_up* `int`) [Function]

Sets the value of '`GTK_PRINT_SETTINGS_NUMBER_UP`'.

settings a `<gtk-print-settings>`

number-up the number of pages per sheet

Since 2.10

`gtk-print-settings-get-resolution` (*self* `<gtk-print-settings*>`) ⇒ (*ret* `int`) [Function]

Gets the value of '`GTK_PRINT_SETTINGS_RESOLUTION`'.

settings a `<gtk-print-settings>`

ret the resolution in dpi

Since 2.10

`gtk-print-settings-set-resolution` (*self* `<gtk-print-settings*>`) (*resolution* `int`) [Function]

Sets the value of '`GTK_PRINT_SETTINGS_RESOLUTION`'.

settings a `<gtk-print-settings>`

resolution the resolution in dpi

Since 2.10

`gtk-print-settings-get-scale` (*self* `<gtk-print-settings*>`) ⇒ (*ret* `double`) [Function]

Gets the value of '`GTK_PRINT_SETTINGS_SCALE`'.

settings a `<gtk-print-settings>`

ret the scale in percent

Since 2.10

`gtk-print-settings-set-scale` (*self* `<gtk-print-settings*>`) (*scale* `double`) [Function]

Sets the value of '`GTK_PRINT_SETTINGS_SCALE`'.

settings a `<gtk-print-settings>`

scale the scale in percent

Since 2.10

`gtk-print-settings-get-print-pages` (*self* `<gtk-print-settings*>`) ⇒ (*ret* `<gtk-print-pages>`) [Function]

Gets the value of '`GTK_PRINT_SETTINGS_PRINT_PAGES`'.

settings a `<gtk-print-settings>`

ret which pages to print

Since 2.10

`gtk-print-settings-set-print-pages` [Function]
(*self* `<gtk-print-settings*>`) (*pages* `<gtk-print-pages>`)
Sets the value of '`GTK_PRINT_SETTINGS_PRINT_PAGES`'.

settings a `<gtk-print-settings>`

pages a `<gtk-print-pages>` value

Since 2.10

`gtk-print-settings-get-page-ranges` [Function]
(*self* `<gtk-print-settings*>`) ⇒ (*ret* `<gtk-page-range*>`)
(*num_ranges* `int`)
Gets the value of '`GTK_PRINT_SETTINGS_PAGE_RANGES`'.

settings a `<gtk-print-settings>`

num-ranges
return location for the length of the returned array

ret an array of `<gtk-page-range>`s. Use `g-free` to free the array when it is no longer needed.

Since 2.10

`gtk-print-settings-set-page-ranges` [Function]
(*self* `<gtk-print-settings*>`) (*page_ranges* `<gtk-page-range*>`)
(*num_ranges* `int`)
Sets the value of '`GTK_PRINT_SETTINGS_PAGE_RANGES`'.

settings a `<gtk-print-settings>`

page-ranges
an array of `<gtk-page-range>`s

num-ranges
the length of *page-ranges*

Since 2.10

`gtk-print-settings-get-page-set` (*self* `<gtk-print-settings*>`) [Function]
⇒ (*ret* `<gtk-page-set>`)
Gets the value of '`GTK_PRINT_SETTINGS_PAGE_SET`'.

settings a `<gtk-print-settings>`

ret the set of pages to print

Since 2.10

`gtk-print-settings-set-page-set` (*self* `<gtk-print-settings*>`) [Function]
(*page_set* `<gtk-page-set>`)
Sets the value of '`GTK_PRINT_SETTINGS_PAGE_SET`'.

settings a `<gtk-print-settings>`

page-set a `<gtk-page-set>` value

Since 2.10

`gtk-print-settings-get-media-type` [Function]
(*self* `<gtk-print-settings*>`) ⇒ (*ret* `mchars`)
Gets the value of '`GTK_PRINT_SETTINGS_MEDIA_TYPE`'.

The set of media types is defined in PWG 5101.1-2002 PWG.

settings a `<gtk-print-settings>`

ret the media type

Since 2.10

`gtk-print-settings-set-media-type` [Function]
(*self* `<gtk-print-settings*>`) (*media_type* `mchars`)
Sets the value of '`GTK_PRINT_SETTINGS_MEDIA_TYPE`'.

The set of media types is defined in PWG 5101.1-2002 PWG.

settings a `<gtk-print-settings>`

media-type
the media type

Since 2.10

`gtk-print-settings-get-dither` (*self* `<gtk-print-settings*>`) [Function]
⇒ (*ret* `mchars`)
Gets the value of '`GTK_PRINT_SETTINGS_DITHER`'.

settings a `<gtk-print-settings>`

ret the dithering that is used

Since 2.10

`gtk-print-settings-set-dither` (*self* `<gtk-print-settings*>`) [Function]
(*dither* `mchars`)
Sets the value of '`GTK_PRINT_SETTINGS_DITHER`'.

settings a `<gtk-print-settings>`

dither the dithering that is used

Since 2.10

`gtk-print-settings-get-finishings` [Function]
(*self* `<gtk-print-settings*>`) ⇒ (*ret* `mchars`)
Gets the value of '`GTK_PRINT_SETTINGS_FINISHINGS`'.

settings a `<gtk-print-settings>`

ret the finishings

Since 2.10

`gtk-print-settings-set-finishings` [Function]
(*self* `<gtk-print-settings*>`) (*finishings* `mchars`)
Sets the value of '`GTK_PRINT_SETTINGS_FINISHINGS`'.

settings a `<gtk-print-settings>`

finishings the finishings

Since 2.10

`gtk-print-settings-get-output-bin` [Function]

(*self* `<gtk-print-settings*>`) ⇒ (*ret* `mchars`)

Gets the value of '`GTK_PRINT_SETTINGS_OUTPUT_BIN`'.

settings a `<gtk-print-settings>`

ret the output bin

Since 2.10

`gtk-print-settings-set-output-bin` [Function]

(*self* `<gtk-print-settings*>`) (*output_bin* `mchars`)

Sets the value of '`GTK_PRINT_SETTINGS_OUTPUT_BIN`'.

settings a `<gtk-print-settings>`

output-bin
the output bin

Since 2.10

112 GtkPageSetup

Stores page setup information

112.1 Overview

A GtkPageSetup object stores the page size, orientation and margins. The idea is that you can get one of these from the page setup dialog and then pass it to the <gtk-print-operation> when printing. The benefit of splitting this out of the <gtk-print-settings> is that these affect the actual layout of the page, and thus need to be set long before user prints.

The margins specified in this object are the "print margins", i.e. the parts of the page that the printer cannot print on. These are different from the layout margins that a word processor uses; they are typically used to determine the *minimal* size for the layout margins.

To obtain a <gtk-page-setup> use gtk-page-setup-new to get the defaults, or use gtk-print-run-page-setup-dialog to show the page setup dialog and receive the resulting page setup.

```
static GtkPrintSettings *settings = NULL;
static GtkPageSetup *page_setup = NULL;

static void
do_page_setup (void)
{
  GtkPageSetup *new_page_setup;

  if (settings == NULL)
    settings = gtk_print_settings_new ();

  new_page_setup = gtk_print_run_page_setup_dialog (GTK_WINDOW (main_window),
                                                    page_setup, settings);

  if (page_setup)
    g_object_unref (page_setup);

  page_setup = new_page_setup;
}
```

Printing support was added in GTK+ 2.10.

112.2 Usage

gtk-page-setup-new ⇒ (*ret* <gtk-page-setup*>) [Function]

Creates a new <gtk-page-setup>.

ret a new <gtk-page-setup>.

Since 2.10

`gtk-page-setup-get-orientation` (*self* `<gtk-page-setup*>`) [Function]
⇒ (*ret* `<gtk-page-orientation>`)

Gets the page orientation of the `<gtk-page-setup>`.

setup a `<gtk-page-setup>`

ret the page orientation

Since 2.10

`gtk-page-setup-set-orientation` (*self* `<gtk-page-setup*>`) [Function]
(*orientation* `<gtk-page-orientation>`)

Sets the page orientation of the `<gtk-page-setup>`.

setup a `<gtk-page-setup>`

orientation a `<gtk-page-orientation>` value

Since 2.10

`gtk-page-setup-get-paper-size` (*self* `<gtk-page-setup*>`) [Function]
⇒ (*ret* `<gtk-paper-size*>`)

Gets the paper size of the `<gtk-page-setup>`.

setup a `<gtk-page-setup>`

ret the paper size

Since 2.10

`gtk-page-setup-set-paper-size` (*self* `<gtk-page-setup*>`) [Function]
(*size* `<gtk-paper-size*>`)

Sets the paper size of the `<gtk-page-setup>` without changing the margins. See `gtk-page-setup-set-paper-size-and-default-margins`.

setup a `<gtk-page-setup>`

size a `<gtk-paper-size>`

Since 2.10

`gtk-page-setup-get-top-margin` (*self* `<gtk-page-setup*>`) [Function]
(*unit* `<gtk-unit>`) ⇒ (*ret* `double`)

Gets the top margin in units of *unit*.

setup a `<gtk-page-setup>`

unit the unit for the return value

ret the top margin

Since 2.10

`gtk-page-setup-set-top-margin` (*self* `<gtk-page-setup*>`) [Function]
(*margin* `double`) (*unit* `<gtk-unit>`)

Sets the top margin of the `<gtk-page-setup>`.

setup a <gtk-page-setup>

margin the new top margin in units of *unit*

unit the units for *margin*

Since 2.10

gtk-page-setup-get-bottom-margin (*self* <gtk-page-setup*>) [Function]
(*unit* <gtk-unit>) ⇒ (*ret* double)

Gets the bottom margin in units of *unit*.

setup a <gtk-page-setup>

unit the unit for the return value

ret the bottom margin

Since 2.10

gtk-page-setup-set-bottom-margin (*self* <gtk-page-setup*>) [Function]
(*margin* double) (*unit* <gtk-unit>)

Sets the bottom margin of the <gtk-page-setup>.

setup a <gtk-page-setup>

margin the new bottom margin in units of *unit*

unit the units for *margin*

Since 2.10

gtk-page-setup-get-left-margin (*self* <gtk-page-setup*>) [Function]
(*unit* <gtk-unit>) ⇒ (*ret* double)

Gets the left margin in units of *unit*.

setup a <gtk-page-setup>

unit the unit for the return value

ret the left margin

Since 2.10

gtk-page-setup-set-left-margin (*self* <gtk-page-setup*>) [Function]
(*margin* double) (*unit* <gtk-unit>)

Sets the left margin of the <gtk-page-setup>.

setup a <gtk-page-setup>

margin the new left margin in units of *unit*

unit the units for *margin*

Since 2.10

`gtk-page-setup-get-right-margin` (*self* `<gtk-page-setup*>`) [Function]
(*unit* `<gtk-unit>`) ⇒ (*ret* `double`)

Gets the right margin in units of *unit*.

setup a `<gtk-page-setup>`

unit the unit for the return value

ret the right margin

Since 2.10

`gtk-page-setup-set-right-margin` (*self* `<gtk-page-setup*>`) [Function]
(*margin* `double`) (*unit* `<gtk-unit>`)

Sets the right margin of the `<gtk-page-setup>`.

setup a `<gtk-page-setup>`

margin the new right margin in units of *unit*

unit the units for *margin*

Since 2.10

`gtk-page-setup-get-paper-width` (*self* `<gtk-page-setup*>`) [Function]
(*unit* `<gtk-unit>`) ⇒ (*ret* `double`)

Returns the paper width in units of *unit*.

Note that this function takes orientation, but not margins into consideration. See `gtk-page-setup-get-page-width`.

setup a `<gtk-page-setup>`

unit the unit for the return value

ret the paper width.

Since 2.10

`gtk-page-setup-get-paper-height` (*self* `<gtk-page-setup*>`) [Function]
(*unit* `<gtk-unit>`) ⇒ (*ret* `double`)

Returns the paper height in units of *unit*.

Note that this function takes orientation, but not margins into consideration. See `gtk-page-setup-get-page-height`.

setup a `<gtk-page-setup>`

unit the unit for the return value

ret the paper height.

Since 2.10

`gtk-page-setup-get-page-width` (*self* `<gtk-page-setup*>`) [Function]
(*unit* `<gtk-unit>`) ⇒ (*ret* `double`)

Returns the page width in units of *unit*.

Note that this function takes orientation and margins into consideration. See `gtk-page-setup-get-paper-width`.

setup a `<gtk-page-setup>`

unit the unit for the return value

ret the page width.

Since 2.10

`gtk-page-setup-get-page-height` (*self* `<gtk-page-setup*>`) [Function]
(*unit* `<gtk-unit>`) ⇒ (*ret* `double`)

Returns the page height in units of *unit*.

Note that this function takes orientation and margins into consideration. See `gtk-page-setup-get-paper-height`.

setup a `<gtk-page-setup>`

unit the unit for the return value

ret the page height.

Since 2.10

113 GtkPaperSize

Support for named paper sizes

113.1 Overview

GtkPaperSize handles paper sizes. It uses the standard called "PWG 5101.1-2002 PWG: Standard for Media Standardized Names" to name the paper sizes (and to get the data for the page sizes). In addition to standard paper sizes, GtkPaperSize allows to construct custom paper sizes with arbitrary dimensions.

The `<gtk-paper-size>` object stores not only the dimensions (width and height) of a paper size and its name, it also provides default print margins.

Printing support has been added in GTK+ 2.10.

113.2 Usage

`gtk-paper-size-new` (*name* `mchars`) ⇒ (*ret* `<gtk-paper-size*>`) [Function]

Creates a new `<gtk-paper-size>` object by parsing a PWG 5101.1-2002 PWG paper name.

If *name* is '`#f`', the default paper size is returned, see `gtk-paper-size-get-default`.

name a paper size name, or '`#f`'

ret a new `<gtk-paper-size>`, use `gtk-paper-size-free` to free it

Since 2.10

`gtk-paper-size-new-from-ppd` (*ppd_name* `mchars`) [Function]
(*ppd_display_name* `mchars`) (*width* `double`) (*height* `double`)
⇒ (*ret* `<gtk-paper-size*>`)

Creates a new `<gtk-paper-size>` object by using PPD information.

If *ppd-name* is not a recognized PPD paper name, *ppd-display-name*, *width* and *height* are used to construct a custom `<gtk-paper-size>` object.

ppd-name a PPD paper name

ppd-display-name
the corresponding human-readable name

width the paper width, in points

height the paper height in points

ret a new `<gtk-paper-size>`, use `gtk-paper-size-free` to free it

Since 2.10

`gtk-paper-size-new-custom` (*name* `mchars`) (*display_name* `mchars`) [Function]
(*width* `double`) (*height* `double`) (*unit* `<gtk-unit>`)
⇒ (*ret* `<gtk-paper-size*>`)

Creates a new `<gtk-paper-size>` object with the given parameters.

name the paper name

display-name
: the human-readable name

width
: the paper width, in units of *unit*

height
: the paper height, in units of *unit*

unit
: the unit for *width* and *height*

ret
: a new `<gtk-paper-size>` object, use `gtk-paper-size-free` to free it

Since 2.10

`gtk-paper-size-is-equal` (*self* `<gtk-paper-size*>`) [Function]
(*size2* `<gtk-paper-size*>`) ⇒ (*ret* `bool`)

Compares two `<gtk-paper-size>` objects.

size1
: a `<gtk-paper-size>` object

size2
: another `<gtk-paper-size>` object

ret
: '`#t`', if *size1* and *size2* represent the same paper size

Since 2.10

`gtk-paper-size-get-name` (*self* `<gtk-paper-size*>`) [Function]
⇒ (*ret* `mchars`)

Gets the name of the `<gtk-paper-size>`.

size
: a `<gtk-paper-size>` object

ret
: the name of *size*

Since 2.10

`gtk-paper-size-get-display-name` (*self* `<gtk-paper-size*>`) [Function]
⇒ (*ret* `mchars`)

Gets the human-readable name of the `<gtk-paper-size>`.

size
: a `<gtk-paper-size>` object

ret
: the human-readable name of *size*

Since 2.10

`gtk-paper-size-get-ppd-name` (*self* `<gtk-paper-size*>`) [Function]
⇒ (*ret* `mchars`)

Gets the PPD name of the `<gtk-paper-size>`, which may be '`#f`'.

size
: a `<gtk-paper-size>` object

ret
: the PPD name of *size*

Since 2.10

`gtk-paper-size-get-width` (*self* `<gtk-paper-size*>`) [Function]
(*unit* `<gtk-unit>`) ⇒ (*ret* `double`)

Gets the paper width of the `<gtk-paper-size>`, in units of *unit*.

size a `<gtk-paper-size>` object

unit the unit for the return value

ret the paper width

Since 2.10

`gtk-paper-size-get-height` (*self* `<gtk-paper-size*>`) (*unit* `<gtk-unit>`) ⇒ (*ret* `double`) [Function]

Gets the paper height of the `<gtk-paper-size>`, in units of *unit*.

size a `<gtk-paper-size>` object

unit the unit for the return value

ret the paper height

Since 2.10

`gtk-paper-size-is-custom` (*self* `<gtk-paper-size*>`) ⇒ (*ret* `bool`) [Function]

Returns '`#t`' if *size* is not a standard paper size.

size a `<gtk-paper-size>` object

ret whether *size* is a custom paper size.

`gtk-paper-size-set-size` (*self* `<gtk-paper-size*>`) (*width* `double`) (*height* `double`) (*unit* `<gtk-unit>`) [Function]

Changes the dimensions of a *size* to *width* x *height*.

size a custom `<gtk-paper-size>` object

width the new width in units of *unit*

height the new height in units of *unit*

unit the unit for *width* and *height*

Since 2.10

`gtk-paper-size-get-default` ⇒ (*ret* `mchars`) [Function]

Returns the name of the default paper size, which depends on the current locale.

ret the name of the default paper size. The string is owned by GTK+ and should not be modified.

Since 2.10

114 GtkPrinter

Represents a printer

114.1 Overview

A <gtk-printer> object represents a printer. You only need to deal directly with printers if you use the non-portable <gtk-print-unix-dialog> API.

A <gtk-printer> allows to get status information about the printer, such as its description, its location, the number of queued jobs, etc. Most importantly, a <gtk-printer> object can be used to create a <gtk-print-job> object, which lets you print to the printer.

Printing support was added in GTK+ 2.10.

114.2 Usage

115 GtkPrintJob

Represents a print job

115.1 Overview

A <gtk-print-job> object represents a job that is sent to a printer. You only need to deal directly with print jobs if you use the non-portable <gtk-print-unix-dialog> API.

Use gtk-print-job-get-surface to obtain the cairo surface onto which the pages must be drawn. Use gtk-print-job-send to send the finished job to the printer. If you don't use cairo <gtk-print-job> also supports printing of manually generated postscript, via gtk-print-job-set-source-file.

Printing support was added in GTK+ 2.10.

115.2 Usage

116 GtkPrintUnixDialog

A print dialog

116.1 Overview

GtkPrintUnixDialog implements a print dialog for platforms which don't provide a native print dialog, like Unix. It can be used very much like any other GTK+ dialog, at the cost of the portability offered by the high-level printing API

In order to print something with `<gtk-print-unix-dialog>`, you need to use `gtk-print-unix-dialog-get-selected-printer` to obtain a `<gtk-printer>` object and use it to construct a `<gtk-print-job>` using `gtk-print-job-new`.

`<gtk-print-unix-dialog>` uses the following response values:

GTK_RESPONSE_APPLY
GTK_RESPONSE_CANCEL

for the "Print" button

for the "Preview" button

for the "Cancel" button

Printing support was added in GTK+ 2.10.

116.2 Usage

117 GtkPageSetupUnixDialog

A page setup dialog

117.1 Overview

GtkPageSetupUnixDialog implements a page setup dialog for platforms which don't provide a native page setup dialog, like Unix. It can be used very much like any other GTK+ dialog, at the cost of the portability offered by the high-level printing API

Printing support was added in GTK+ 2.10.

117.2 Usage

118 GtkAdjustment

A GtkObject representing an adjustable bounded value

118.1 Overview

The <gtk-adjustment> object represents a value which has an associated lower and upper bound, together with step and page increments, and a page size. It is used within several GTK+ widgets, including <gtk-spin-button>, <gtk-viewport>, and <gtk-range> (which is a base class for <gtk-hscrollbar>, <gtk-vscrollbar>, <gtk-hscale>, and <gtk-vscale>).

The <gtk-adjustment> object does not update the value itself. Instead it is left up to the owner of the <gtk-adjustment> to control the value.

The owner of the <gtk-adjustment> typically calls the gtk-adjustment-value-changed and gtk-adjustment-changed functions after changing the value and its bounds. This results in the emission of the "value_changed" or "changed" signal respectively.

118.2 Usage

<gtk-adjustment> [Class]

This <gobject> class defines the following properties:

value The value of the adjustment

lower The minimum value of the adjustment

upper The maximum value of the adjustment

step-increment
The step increment of the adjustment

page-increment
The page increment of the adjustment

page-size
The page size of the adjustment

changed [Signal on <gtk-adjustment>]

Emitted when one or more of the <gtk-adjustment> fields have been changed, other than the value field.

value-changed [Signal on <gtk-adjustment>]

Emitted when the <gtk-adjustment> value field has been changed.

gtk-adjustment-new (*value* double) (*lower* double) (*upper* double) [Function]
(*step_increment* double) (*page_increment* double) (*page_size* double)
⇒ (*ret* <gtk-object>)

Creates a new <gtk-adjustment>.

value the initial value.

lower the minimum value.

upper the maximum value.

step-increment
the step increment.

page-increment
the page increment.

page-size the page size.

ret a new `<gtk-adjustment>`.

`gtk-adjustment-get-value` (*self* `<gtk-adjustment>`) [Function]
⇒ (*ret* `double`)
`get-value` [Method]

Gets the current value of the adjustment. See `gtk-adjustment-set-value`.

adjustment
a `<gtk-adjustment>`

ret The current value of the adjustment.

`gtk-adjustment-set-value` (*self* `<gtk-adjustment>`) (*value* `double`) [Function]
`set-value` [Method]

Sets the `<gtk-adjustment>` value. The value is clamped to lie between '`adjustment->lower`' and '`adjustment->upper`'.

Note that for adjustments which are used in a `<gtk-scrollbar>`, the effective range of allowed values goes from '`adjustment->lower`' to '`adjustment->upper - adjustment->page_size`'.

adjustment
a `<gtk-adjustment>`.

value the new value.

`gtk-adjustment-clamp-page` (*self* `<gtk-adjustment>`) [Function]
(*lower* `double`) (*upper* `double`)
`clamp-page` [Method]

Updates the `<gtk-adjustment>`*value* to ensure that the range between *lower* and *upper* is in the current page (i.e. between *value* and *value* + *page-size*). If the range is larger than the page size, then only the start of it will be in the current page. A "changed" signal will be emitted if the value is changed.

adjustment
a `<gtk-adjustment>`.

lower the lower value.

upper the upper value.

`gtk-adjustment-changed` (*self* `<gtk-adjustment>`) [Function]
`changed` [Method]

Emits a "changed" signal from the `<gtk-adjustment>`. This is typically called by the owner of the `<gtk-adjustment>` after it has changed any of the `<gtk-adjustment>` fields other than the value.

adjustment
a `<gtk-adjustment>`

`gtk-adjustment-value-changed` (*self* `<gtk-adjustment>`) [Function]
`value-changed` [Method]
Emits a "value_changed" signal from the `<gtk-adjustment>`. This is typically called by the owner of the `<gtk-adjustment>` after it has changed the `<gtk-adjustment>` value field.

adjustment
a `<gtk-adjustment>`

119 GtkArrow

Displays an arrow

119.1 Overview

GtkArrow should be used to draw simple arrows that need to point in one of the four cardinal directions (up, down, left, or right). The style of the arrow can be one of shadow in, shadow out, etched in, or etched out. Note that these directions and style types may be ammended in versions of Gtk to come.

GtkArrow will fill any space alloted to it, but since it is inherited from `<gtk-misc>`, it can be padded and/or aligned, to fill exactly the space the programmer desires.

Arrows are created with a call to `gtk-arrow-new`. The direction or style of an arrow can be changed after creation by using `gtk-arrow-set`.

119.2 Usage

`<gtk-arrow>` [Class]

This `<gobject>` class defines the following properties:

`arrow-type`
: The direction the arrow should point

`shadow-type`
: Appearance of the shadow surrounding the arrow

`gtk-arrow-new` (*arrow_type* `<gtk-arrow-type>`) (*shadow_type* `<gtk-shadow-type>`) ⇒ (*ret* `<gtk-widget>`) [Function]

Creates a new arrow widget.

arrow-type
: a valid `<gtk-arrow-type>`.

shadow-type
: a valid `<gtk-shadow-type>`.

ret the new `<gtk-arrow>` widget.

`gtk-arrow-set` (*self* `<gtk-arrow>`) (*arrow_type* `<gtk-arrow-type>`) (*shadow_type* `<gtk-shadow-type>`) [Function]

`set` [Method]

Sets the direction and style of the `<gtk-arrow>`, *arrow*.

arrow a widget of type `<gtk-arrow>`.

arrow-type
: a valid `<gtk-arrow-type>`.

shadow-type
: a valid `<gtk-shadow-type>`.

120 GtkCalendar

Displays a calendar and allows the user to select a date

120.1 Overview

`<gtk-calendar>` is a widget that displays a calendar, one month at a time. It can be created with `gtk-calendar-new`.

The month and year currently displayed can be altered with `gtk-calendar-select-month`. The exact day can be selected from the displayed month using `gtk-calendar-select-day`.

To place a visual marker on a particular day, use `gtk-calendar-mark-day` and to remove the marker, `gtk-calendar-unmark-day`. Alternative, all marks can be cleared with `gtk-calendar-clear-marks`.

The way in which the calendar itself is displayed can be altered using `gtk-calendar-set-display-options`.

The selected date can be retrieved from a `<gtk-calendar>` using `gtk-calendar-get-date`.

120.2 Usage

`<gtk-calendar>` [Class]

This `<gobject>` class defines the following properties:

`year` The selected year

`month` The selected month (as a number between 0 and 11)

`day` The selected day (as a number between 1 and 31, or 0 to unselect the currently selected day)

`show-heading`
If TRUE, a heading is displayed

`show-day-names`
If TRUE, day names are displayed

`no-month-change`
If TRUE, the selected month cannot be changed

`show-week-numbers`
If TRUE, week numbers are displayed

`month-changed` [Signal on `<gtk-calendar>`]

Emitted when the user clicks a button to change the selected month on a calendar.

`day-selected` [Signal on `<gtk-calendar>`]

Emitted when the user selects a day.

`day-selected-double-click` [Signal on `<gtk-calendar>`]

`prev-month` [Signal on `<gtk-calendar>`]

`next-month` [Signal on `<gtk-calendar>`]

`prev-year` [Signal on `<gtk-calendar>`]

`next-year` [Signal on `<gtk-calendar>`]

`gtk-calendar-new` ⇒ (*ret* `<gtk-widget>`) [Function]

Creates a new calendar, with the current date being selected.

ret a newly `<gtk-calendar>` widget

`gtk-calendar-select-month` (*self* `<gtk-calendar>`) [Function]
(*month* `unsigned-int`) (*year* `unsigned-int`) ⇒ (*ret* `bool`)

`select-month` [Method]

Shifts the calendar to a different month.

calendar a `<gtk-calendar>`

month a month number between 0 and 11.

year the year the month is in.

ret '`#t`', always

`gtk-calendar-select-day` (*self* `<gtk-calendar>`) [Function]
(*day* `unsigned-int`)

`select-day` [Method]

Selects a day from the current month.

calendar a `<gtk-calendar>`.

day the day number between 1 and 31, or 0 to unselect the currently selected day.

`gtk-calendar-mark-day` (*self* `<gtk-calendar>`) (*day* `unsigned-int`) [Function]
⇒ (*ret* `bool`)

`mark-day` [Method]

Places a visual marker on a particular day.

calendar a `<gtk-calendar>`

day the day number to mark between 1 and 31.

ret '`#t`', always

`gtk-calendar-unmark-day` (*self* `<gtk-calendar>`) [Function]
(*day* `unsigned-int`) ⇒ (*ret* `bool`)

`unmark-day` [Method]

Removes the visual marker from a particular day.

calendar a `<gtk-calendar>`.

day the day number to unmark between 1 and 31.

ret '`#t`', always

`gtk-calendar-clear-marks` (*self* `<gtk-calendar>`) [Function]
`clear-marks` [Method]

Remove all visual markers.

calendar a `<gtk-calendar>`

`gtk-calendar-set-display-options` (*self* `<gtk-calendar>`) [Function]
(*flags* `<gtk-calendar-display-options>`)
`set-display-options` [Method]

Sets display options (whether to display the heading and the month headings).

calendar a `<gtk-calendar>`

flags the display options to set

Since 2.4

`gtk-calendar-get-date` (*self* `<gtk-calendar>`) [Function]
⇒ (*year* `unsigned-int`) (*month* `unsigned-int`) (*day* `unsigned-int`)
`get-date` [Method]

Obtains the selected date from a `<gtk-calendar>`.

calendar a `<gtk-calendar>`

year location to store the year number, or '`#f`'

month location to store the month number (between 0 and 11), or '`#f`'

day location to store the day number (between 1 and 31), or '`#f`'

`gtk-calendar-freeze` (*self* `<gtk-calendar>`) [Function]
`freeze` [Method]

'`gtk_calendar_freeze`' has been deprecated since version 2.8 and should not be used in newly-written code.

Does nothing. Previously locked the display of the calendar until it was thawed with `gtk-calendar-thaw`.

calendar a `<gtk-calendar>`

`gtk-calendar-thaw` (*self* `<gtk-calendar>`) [Function]
`thaw` [Method]

'`gtk_calendar_thaw`' has been deprecated since version 2.8 and should not be used in newly-written code.

Does nothing. Previously defrosted a calendar; all the changes made since the last `gtk-calendar-freeze` were displayed.

calendar a `<gtk-calendar>`

121 GtkDrawingArea

A widget for custom user interface elements

121.1 Overview

The <gtk-drawing-area> widget is used for creating custom user interface elements. It's essentially a blank widget; you can draw on 'widget->window'. After creating a drawing area, the application may want to connect to:

Mouse and button press signals to respond to input from the user. (Use gtk-widget-add-events to enable events you wish to receive.)

The "realize" signal to take any necessary actions when the widget is instantiated on a particular display. (Create GDK resources in response to this signal.)

The "configure_event" signal to take any necessary actions when the widget changes size.

The "expose_event" signal to handle redrawing the contents of the widget.

The following code portion demonstrates using a drawing area to display a circle in the normal widget foreground color. Note that GDK automatically clears the exposed area to the background color before sending the expose event, and that drawing is implicitly clipped to the exposed area.

```
gboolean
expose_event_callback (GtkWidget *widget, GdkEventExpose *event, gpointer data)
{
  gdk_draw_arc (widget->window,
                widget->style->fg_gc[GTK_WIDGET_STATE (widget)],
                TRUE,
                0, 0, widget->allocation.width, widget->allocation.height,
                0, 64 * 360);

  return TRUE;
}
[...]
  GtkWidget *drawing_area = gtk_drawing_area_new ();
  gtk_widget_set_size_request (drawing_area, 100, 100);
  g_signal_connect (G_OBJECT (drawing_area), "expose_event",
                    G_CALLBACK (expose_event_callback), NULL);
```

Expose events are normally delivered when a drawing area first comes onscreen, or when it's covered by another window and then uncovered (exposed). You can also force an expose event by adding to the "damage region" of the drawing area's window; gtk-widget-queue-draw-area and gdk-window-invalidate-rect are equally good ways to do this. You'll then get an expose event for the invalid region.

The available routines for drawing are documented on the GDK Drawing Primitives page. See also gdk-pixbuf-render-to-drawable for drawing a <gdk-pixbuf>.

To receive mouse events on a drawing area, you will need to enable them with gtk-widget-add-events. To receive keyboard events, you will need to set the <gtk-can-focus>

flag on the drawing area, and should probably draw some user-visible indication that the drawing area is focused. Use the `gtk-has-focus` macro in your expose event handler to decide whether to draw the focus indicator. See `gtk-paint-focus` for one way to draw focus.

121.2 Usage

`<gtk-drawing-area>` [Class]

This `<gobject>` class defines no properties, other than those defined by its superclasses.

`gtk-drawing-area-new` ⇒ (*ret* `<gtk-widget>`) [Function]

Creates a new drawing area.

ret a new `<gtk-drawing-area>`

122 GtkEventBox

A widget used to catch events for widgets which do not have their own window

122.1 Overview

The <gtk-event-box> widget is a subclass of <gtk-bin> which also has its own window. It is useful since it allows you to catch events for widgets which do not have their own window.

122.2 Usage

<gtk-event-box> [Class]

This <gobject> class defines the following properties:

visible-window

Whether the event box is visible, as opposed to invisible and only used to trap events.

above-child

Whether the event-trapping window of the eventbox is above the window of the child widget as opposed to below it.

gtk-event-box-new ⇒ (*ret* <gtk-widget>) [Function]

Creates a new <gtk-event-box>.

ret a new <gtk-event-box>.

gtk-event-box-set-above-child (*self* <gtk-event-box>) [Function]
(*above_child* bool)
set-above-child [Method]

Set whether the event box window is positioned above the windows of its child, as opposed to below it. If the window is above, all events inside the event box will go to the event box. If the window is below, events in windows of child widgets will first got to that widget, and then to its parents.

The default is to keep the window below the child.

event-box a <gtk-event-box>

above-child

'#t' if the event box window is above the windows of its child

Since 2.4

gtk-event-box-get-above-child (*self* <gtk-event-box>) [Function]
⇒ (*ret* bool)
get-above-child [Method]

Returns whether the event box window is above or below the windows of its child. See gtk-event-box-set-above-child for details.

event-box a <gtk-event-box>

ret '#t' if the event box window is above the window of its child.

Since 2.4

`gtk-event-box-set-visible-window` (*self* `<gtk-event-box>`) [Function]
(*visible_window* `bool`)

`set-visible-window` [Method]

Set whether the event box uses a visible or invisible child window. The default is to use visible windows.

In an invisible window event box, the window that that the event box creates is a '`GDK_INPUT_ONLY`' window, which means that it is invisible and only serves to receive events.

A visible window event box creates a visible ('`GDK_INPUT_OUTPUT`') window that acts as the parent window for all the widgets contained in the event box.

You should generally make your event box invisible if you just want to trap events. Creating a visible window may cause artifacts that are visible to the user, especially if the user is using a theme with gradients or pixmaps.

The main reason to create a non input-only event box is if you want to set the background to a different color or draw on it.

There is one unexpected issue for an invisible event box that has its window below the child. (See `gtk-event-box-set-above-child`.) Since the input-only window is not an ancestor window of any windows that descendent widgets of the event box create, events on these windows aren't propagated up by the windowing system, but only by GTK+. The practical effect of this is if an event isn't in the event mask for the descendant window (see `gtk-widget-add-events`), it won't be received by the event box.

This problem doesn't occur for visible event boxes, because in that case, the event box window is actually the ancestor of the descendant windows, not just at the same place on the screen.

event-box a `<gtk-event-box>`

visible-window
boolean value

Since 2.4

`gtk-event-box-get-visible-window` (*self* `<gtk-event-box>`) [Function]
⇒ (*ret* `bool`)

`get-visible-window` [Method]

Returns whether the event box has a visible window. See `gtk-event-box-set-visible-window` for details.

event-box a `<gtk-event-box>`

ret '`#t`' if the event box window is visible

Since 2.4

123 GtkHandleBox

a widget for detachable window portions

123.1 Overview

The <gtk-handle-box> widget allows a portion of a window to be "torn off". It is a bin widget which displays its child and a handle that the user can drag to tear off a separate window (the *float window*) containing the child widget. A thin *ghost* is drawn in the original location of the handlebox. By dragging the separate window back to its original location, it can be reattached.

When reattaching, the ghost and float window, must be aligned along one of the edges, the *snap edge*. This either can be specified by the application programmer explicitely, or GTK+ will pick a reasonable default based on the handle position.

To make detaching and reattaching the handlebox as minimally confusing as possible to the user, it is important to set the snap edge so that the snap edge does not move when the handlebox is deattached. For instance, if the handlebox is packed at the bottom of a VBox, then when the handlebox is detached, the bottom edge of the handlebox's allocation will remain fixed as the height of the handlebox shrinks, so the snap edge should be set to 'GTK_POS_BOTTOM'.

123.2 Usage

<gtk-handle-box> [Class]

This <gobject> class defines the following properties:

shadow Deprecated property, use shadow_type instead

shadow-type
Appearance of the shadow that surrounds the container

handle-position
Position of the handle relative to the child widget

snap-edge
Side of the handlebox that's lined up with the docking point to dock the handlebox

snap-edge-set
Whether to use the value from the snap_edge property or a value derived from handle_position

child-attached (*arg0* <gtk-widget>) [Signal on <gtk-handle-box>]

This signal is emitted when the contents of the handlebox are reattached to the main window.

child-detached (*arg0* <gtk-widget>) [Signal on <gtk-handle-box>]

This signal is emitted when the contents of the handlebox are detached from the main window.

`gtk-handle-box-new` ⇒ (*ret* `<gtk-widget>`) [Function]

Create a new handle box.

ret a new `<gtk-handle-box>`.

`gtk-handle-box-set-shadow-type` (*self* `<gtk-handle-box>`) [Function]
(*type* `<gtk-shadow-type>`)
`set-shadow-type` [Method]

Sets the type of shadow to be drawn around the border of the handle box.

handle-box
a `<gtk-handle-box>`

type the shadow type.

`gtk-handle-box-set-handle-position` (*self* `<gtk-handle-box>`) [Function]
(*position* `<gtk-position-type>`)
`set-handle-position` [Method]

Sets the side of the handlebox where the handle is drawn.

handle-box
a `<gtk-handle-box>`

position the side of the handlebox where the handle should be drawn.

`gtk-handle-box-set-snap-edge` (*self* `<gtk-handle-box>`) [Function]
(*edge* `<gtk-position-type>`)
`set-snap-edge` [Method]

Sets the snap edge of a handlebox. The snap edge is the edge of the detached child that must be aligned with the corresponding edge of the "ghost" left behind when the child was detached to reattach the torn-off window. Usually, the snap edge should be chosen so that it stays in the same place on the screen when the handlebox is torn off.

If the snap edge is not set, then an appropriate value will be guessed from the handle position. If the handle position is '`GTK_POS_RIGHT`' or '`GTK_POS_LEFT`', then the snap edge will be '`GTK_POS_TOP`', otherwise it will be '`GTK_POS_LEFT`'.

handle-box
a `<gtk-handle-box>`

edge the snap edge, or -1 to unset the value; in which case GTK+ will try to guess an appropriate value in the future.

`gtk-handle-box-get-handle-position` (*self* `<gtk-handle-box>`) [Function]
⇒ (*ret* `<gtk-position-type>`)
`get-handle-position` [Method]

Gets the handle position of the handle box. See `gtk-handle-box-set-handle-position`.

handle-box
a `<gtk-handle-box>`

ret the current handle position.

`gtk-handle-box-get-shadow-type` (*self* `<gtk-handle-box>`) [Function]
⇒ (*ret* `<gtk-shadow-type>`)

`get-shadow-type` [Method]

Gets the type of shadow drawn around the handle box. See `gtk-handle-box-set-shadow-type`.

handle-box
a `<gtk-handle-box>`

ret the type of shadow currently drawn around the handle box.

`gtk-handle-box-get-snap-edge` (*self* `<gtk-handle-box>`) [Function]
⇒ (*ret* `<gtk-position-type>`)

`get-snap-edge` [Method]

Gets the edge used for determining reattachment of the handle box. See `gtk-handle-box-set-snap-edge`.

handle-box
a `<gtk-handle-box>`

ret the edge used for determining reattachment, or (GtkPositionType)-1 if this is determined (as per default) from the handle position.

124 GtkIMContextSimple

An input method context supporting table-based input methods

124.1 Overview

124.2 Usage

`<gtk-im-context-simple>` [Class]

This `<gobject>` class defines no properties, other than those defined by its superclasses.

`gtk-im-context-simple-new` ⇒ (*ret* `<gtk-im-context*>`) [Function]

Creates a new `<gtk-im-context-simple>`.

ret a new `<gtk-im-context-simple>`.

`gtk-im-context-simple-add-table` [Function]
(*self* `<gtk-im-context-simple>`) (*max_seq_len* `int`) (*n_seqs* `int`)
⇒ (*data* `unsigned-int16`)

`add-table` [Method]

Adds an additional table to search to the input context. Each row of the table consists of *max-seq-len* key symbols followed by two `<guint16>` interpreted as the high and low words of a `<gunicode>` value. Tables are searched starting from the last added.

The table must be sorted in dictionary order on the numeric value of the key symbol fields. (Values beyond the length of the sequence should be zero.)

context-simple
A `<gtk-im-context-simple>`

data the table

max-seq-len
Maximum length of a sequence in the table (cannot be greater than `<gtk-max-compose-len>`)

n-seqs number of sequences in the table

125 GtkIMMulticontext

An input method context supporting multiple, loadable input methods

125.1 Overview

125.2 Usage

`<gtk-im-multicontext>` [Class]
This `<gobject>` class defines no properties, other than those defined by its superclasses.

`gtk-im-multicontext-new` ⇒ (*ret* `<gtk-im-context*>`) [Function]
Creates a new `<gtk-im-multicontext>`.

ret a new `<gtk-im-multicontext>`.

126 GtkSizeGroup

Grouping widgets so they request the same size

126.1 Overview

<gtk-size-group> provides a mechanism for grouping a number of widgets together so they all request the same amount of space. This is typically useful when you want a column of widgets to have the same size, but you can't use a <gtk-table> widget.

In detail, the size requested for each widget in a <gtk-size-group> is the maximum of the sizes that would have been requested for each widget in the size group if they were not in the size group. The mode of the size group (see gtk-size-group-set-mode) determines whether this applies to the horizontal size, the vertical size, or both sizes.

Note that size groups only affect the amount of space requested, not the size that the widgets finally receive. If you want the widgets in a <gtk-size-group> to actually be the same size, you need to pack them in such a way that they get the size they request and not more. For example, if you are packing your widgets into a table, you would not include the 'GTK_FILL' flag.

<gtk-size-group> objects are referenced by each widget in the size group, so once you have added all widgets to a <gtk-size-group>, you can drop the initial reference to the size group with g-object-unref. If the widgets in the size group are subsequently destroyed, then they will be removed from the size group and drop their references on the size group; when all widgets have been removed, the size group will be freed.

Widgets can be part of multiple size groups; GTK+ will compute the horizontal size of a widget from the horizontal requisition of all widgets that can be reached from the widget by a chain of size groups of type 'GTK_SIZE_GROUP_HORIZONTAL' or 'GTK_SIZE_GROUP_BOTH', and the vertical size from the vertical requisition of all widgets that can be reached from the widget by a chain of size groups of type 'GTK_SIZE_GROUP_VERTICAL' or 'GTK_SIZE_GROUP_BOTH'.

126.2 Usage

<gtk-size-group> [Class]

This <gobject> class defines the following properties:

`mode` The directions in which the size group affects the requested sizes of its component widgets

`ignore-hidden` If TRUE, unmapped widgets are ignored when determining the size of the group

gtk-size-group-new (*mode* <gtk-size-group-mode>) [Function]
⇒ (*ret* <gtk-size-group>)

Create a new <gtk-size-group>.

mode the mode for the new size group.

ret a newly created <gtk-size-group>

`gtk-size-group-set-mode` (*self* `<gtk-size-group>`) [Function]
(*mode* `<gtk-size-group-mode>`)
`set-mode` [Method]

Sets the `<gtk-size-group-mode>` of the size group. The mode of the size group determines whether the widgets in the size group should all have the same horizontal requisition (‘`GTK_SIZE_GROUP_MODE_HORIZONTAL`’) all have the same vertical requisition (‘`GTK_SIZE_GROUP_MODE_VERTICAL`’), or should all have the same requisition in both directions (‘`GTK_SIZE_GROUP_MODE_BOTH`’).

size-group a `<gtk-size-group>`

mode the mode to set for the size group.

`gtk-size-group-get-mode` (*self* `<gtk-size-group>`) [Function]
⇒ (*ret* `<gtk-size-group-mode>`)
`get-mode` [Method]

Gets the current mode of the size group. See `gtk-size-group-set-mode`.

size-group a `<gtk-size-group>`

ret the current mode of the size group.

`gtk-size-group-set-ignore-hidden` (*self* `<gtk-size-group>`) [Function]
(*ignore_hidden* `bool`)
`set-ignore-hidden` [Method]

Sets whether unmapped widgets should be ignored when calculating the size.

size-group a `<gtk-size-group>`

ignore-hidden
whether unmapped widgets should be ignored when calculating the size

Since 2.8

`gtk-size-group-get-ignore-hidden` (*self* `<gtk-size-group>`) [Function]
⇒ (*ret* `bool`)
`get-ignore-hidden` [Method]

Returns if invisible widgets are ignored when calculating the size.

size-group a `<gtk-size-group>`

ret ‘`#t`’ if invisible widgets are ignored.

Since 2.8

`gtk-size-group-add-widget` (*self* `<gtk-size-group>`) [Function]
(*widget* `<gtk-widget>`)
`add-widget` [Method]

Adds a widget to a `<gtk-size-group>`. In the future, the requisition of the widget will be determined as the maximum of its requisition and the requisition of the other widgets in the size group. Whether this applies horizontally, vertically, or in both directions depends on the mode of the size group. See `gtk-size-group-set-mode`.

size-group a `<gtk-size-group>`

widget the `<gtk-widget>` to add

`gtk-size-group-remove-widget` (*self* `<gtk-size-group>`) [Function]
(*widget* `<gtk-widget>`)
`remove-widget` [Method]

Removes a widget from a `<gtk-size-group>`.

size-group a `<gtk-size-grup>`

widget the `<gtk-widget>` to remove

127 GtkTooltips

Add tips to your widgets

127.1 Overview

Tooltips are the messages that appear next to a widget when the mouse pointer is held over it for a short amount of time. They are especially helpful for adding more verbose descriptions of things such as buttons in a toolbar.

An individual tooltip belongs to a group of tooltips. A group is created with a call to `gtk-tooltips-new`. Every tooltip in the group can then be turned off with a call to `gtk-tooltips-disable` and enabled with `gtk-tooltips-enable`.

The length of time the user must keep the mouse over a widget before the tip is shown, can be altered with `gtk-tooltips-set-delay`. This is set on a 'per group of tooltips' basis.

To assign a tip to a particular `<gtk-widget>`, `gtk-tooltips-set-tip` is used.

Tooltips can only be set on widgets which have their own X window and receive enter and leave events. To check if a widget has its own window use `gtk-widget-no-window`. To add a tooltip to a widget that doesn't have its own window, place the widget inside a `<gtk-event-box>` and add a tooltip to that instead.

The default appearance of all tooltips in a program is determined by the current GTK+ theme that the user has selected.

Information about the tooltip (if any) associated with an arbitrary widget can be retrieved using `gtk-tooltips-data-get`.

```
  GtkWidget *load_button, *save_button, *hbox;
  GtkTooltips *button_bar_tips;

  button_bar_tips = gtk_tooltips_new ();

  /* Create the buttons and pack them into a GtkHBox */
  hbox = gtk_hbox_new (TRUE, 2);

  load_button = gtk_button_new_with_label ("Load a file");
  gtk_box_pack_start (GTK_BOX (hbox), load_button, TRUE, TRUE, 2);
  gtk_widget_show (load_button);

  save_button = gtk_button_new_with_label ("Save a file");
  gtk_box_pack_start (GTK_BOX (hbox), save_button, TRUE, TRUE, 2);
  gtk_widget_show (save_button);
  gtk_widget_show (hbox);

  /* Add the tips */
  gtk_tooltips_set_tip (GTK_TOOLTIPS (button_bar_tips), load_button,
"Load a new document into this window",
"Requests the filename of a document.
 This will then be loaded into the current
```

```
 window, replacing the contents of whatever
 is already loaded.");
  gtk_tooltips_set_tip (GTK_TOOLTIPS (button_bar_tips), save_button,
"Saves the current document to a file",
"If you have saved the document previously,
 then the new version will be saved over the
 old one. Otherwise, you will be prompted for
 a filename.");
```

127.2 Usage

`<gtk-tooltips>` [Class]

This `<gobject>` class defines no properties, other than those defined by its superclasses.

`gtk-tooltips-new` ⇒ (*ret* `<gtk-tooltips>`) [Function]

Creates an empty group of tooltips. This function initialises a `<gtk-tooltips>` structure. Without at least one such structure, you can not add tips to your application.

ret a new `<gtk-tooltips>` group for you to use.

`gtk-tooltips-enable` (*self* `<gtk-tooltips>`) [Function]
`enable` [Method]

Allows the user to see your tooltips as they navigate your application.

tooltips a `<gtk-tooltips>`.

`gtk-tooltips-disable` (*self* `<gtk-tooltips>`) [Function]
`disable` [Method]

Causes all tooltips in *tooltips* to become inactive. Any widgets that have tips associated with that group will no longer display their tips until they are enabled again with `gtk-tooltips-enable`.

tooltips a `<gtk-tooltips>`.

`gtk-tooltips-set-tip` (*self* `<gtk-tooltips>`) [Function]
(*widget* `<gtk-widget>`) (*tip_text* `mchars`) (*tip_private* `mchars`)
`set-tip` [Method]

Adds a tooltip containing the message *tip-text* to the specified `<gtk-widget>`.

tooltips a `<gtk-tooltips>`.

widget the `<gtk-widget>` you wish to associate the tip with.

tip-text a string containing the tip itself.

tip-private
a string of any further information that may be useful if the user gets stuck.

`gtk-tooltips-force-window` (*self* `<gtk-tooltips>`) [Function]
`force-window` [Method]

Ensures that the window used for displaying the given *tooltips* is created.

Applications should never have to call this function, since GTK+ takes care of this.

tooltips a `<gtk-tool-tips>`

128 GtkViewport

An adapter which makes widgets scrollable

128.1 Overview

128.2 Usage

`<gtk-viewport>` [Class]

This `<gobject>` class defines the following properties:

`hadjustment`
: The GtkAdjustment that determines the values of the horizontal position for this viewport

`vadjustment`
: The GtkAdjustment that determines the values of the vertical position for this viewport

`shadow-type`
: Determines how the shadowed box around the viewport is drawn

`set-scroll-adjustments` (*arg0* `<gtk-adjustment>`) [Signal on `<gtk-viewport>`]
(*arg1* `<gtk-adjustment>`)

`gtk-viewport-new` (*hadjustment* `<gtk-adjustment>`) [Function]
(*vadjustment* `<gtk-adjustment>`) ⇒ (*ret* `<gtk-widget>`)

Creates a new `<gtk-viewport>` with the given adjustments.

hadjustment
: horizontal adjustment.

vadjustment
: vertical adjustment.

ret a new `<gtk-viewport>`.

`gtk-viewport-get-hadjustment` (*self* `<gtk-viewport>`) [Function]
⇒ (*ret* `<gtk-adjustment>`)

`get-hadjustment` [Method]

Returns the horizontal adjustment of the viewport.

viewport a `<gtk-viewport>`.

ret the horizontal adjustment of *viewport*.

`gtk-viewport-get-vadjustment` (*self* `<gtk-viewport>`) [Function]
⇒ (*ret* `<gtk-adjustment>`)

`get-vadjustment` [Method]

Returns the vertical adjustment of the viewport.

viewport a `<gtk-viewport>`.

ret the vertical adjustment of *viewport*.

`gtk-viewport-set-hadjustment` (*self* `<gtk-viewport>`) [Function]
(*adjustment* `<gtk-adjustment>`)
`set-hadjustment` [Method]
Sets the horizontal adjustment of the viewport.

viewport a `<gtk-viewport>`.

adjustment
a `<gtk-adjustment>`.

`gtk-viewport-set-vadjustment` (*self* `<gtk-viewport>`) [Function]
(*adjustment* `<gtk-adjustment>`)
`set-vadjustment` [Method]
Sets the vertical adjustment of the viewport.

viewport a `<gtk-viewport>`.

adjustment
a `<gtk-adjustment>`.

`gtk-viewport-set-shadow-type` (*self* `<gtk-viewport>`) [Function]
(*type* `<gtk-shadow-type>`)
`set-shadow-type` [Method]
Sets the shadow type of the viewport.

viewport a `<gtk-viewport>`.

type the new shadow type.

`gtk-viewport-get-shadow-type` (*self* `<gtk-viewport>`) [Function]
⇒ (*ret* `<gtk-shadow-type>`)
`get-shadow-type` [Method]
Gets the shadow type of the `<gtk-viewport>`. See `gtk-viewport-set-shadow-type`.

viewport a `<gtk-viewport>`

ret the shadow type

129 GtkAccessible

Accessibility support for widgets

129.1 Overview

129.2 Usage

`<gtk-accessible>` [Class]
This `<gobject>` class defines no properties, other than those defined by its superclasses.

130 GtkBin

A container with just one child

130.1 Overview

The <gtk-bin> widget is a container with just one child. It is not very useful itself, but it is useful for deriving subclasses, since it provides common code needed for handling a single child widget.

Many GTK+ widgets are subclasses of <gtk-bin>, including <gtk-window>, <gtk-button>, <gtk-frame>, <gtk-handle-box>, and <gtk-scrolled-window>.

130.2 Usage

<gtk-bin> [Class]

This <gobject> class defines no properties, other than those defined by its superclasses.

gtk-bin-get-child (*self* <gtk-bin>) ⇒ (*ret* <gtk-widget>) [Function]
get-child [Method]

Gets the child of the <gtk-bin>, or '#f' if the bin contains no child widget. The returned widget does not have a reference added, so you do not need to unref it.

bin a <gtk-bin>

ret pointer to child of the <gtk-bin>

131 GtkBox

Base class for box containers

131.1 Overview

GtkBox is an abstract widget which encapsulates functionallity for a particular kind of container, one that organizes a variable number of widgets into a rectangular area. GtkBox currently has two derived classes, `<gtk-hbox>` and `<gtk-vbox>`.

The rectangular area of a GtkBox is organized into either a single row or a single column of child widgets depending upon whether the box is of type `<gtk-hbox>` or `<gtk-vbox>`, respectively. Thus, all children of a GtkBox are allocated one dimension in common, which is the height of a row, or the width of a column.

GtkBox uses a notion of *packing*. Packing refers to adding widgets with reference to a particular position in a `<gtk-container>`. For a GtkBox, there are two reference positions: the *start* and the *end* of the box. For a `<gtk-vbox>`, the start is defined as the top of the box and the end is defined as the bottom. For a `<gtk-hbox>` the start is defined as the left side and the end is defined as the right side.

Use repeated calls to `gtk-box-pack-start` to pack widgets into a GtkBox from start to end. Use `gtk-box-pack-end` to add widgets from end to start. You may intersperse these calls and add widgets from both ends of the same GtkBox.

Use `gtk-box-pack-start-defaults` or `gtk-box-pack-end-defaults` to pack widgets into a GtkBox if you do not need to specify the , , or attributes of the child to be added.

Because GtkBox is a `<gtk-container>`, you may also use `gtk-container-add` to insert widgets into the box, and they will be packed as if with `gtk-box-pack-start-defaults`. Use `gtk-container-remove` to remove widgets from the GtkBox.

Use `gtk-box-set-homogeneous` to specify whether or not all children of the GtkBox are forced to get the same amount of space.

Use `gtk-box-set-spacing` to determine how much space will be minimally placed between all children in the GtkBox.

Use `gtk-box-reorder-child` to move a GtkBox child to a different place in the box.

Use `gtk-box-set-child-packing` to reset the , , and attributes of any GtkBox child. Use `gtk-box-query-child-packing` to query these fields.

131.2 Usage

`<gtk-box>` [Class]

This `<gobject>` class defines the following properties:

`spacing` The amount of space between children

`homogeneous`
Whether the children should all be the same size

`gtk-box-pack-start` (*self* `<gtk-box>`) (*child* `<gtk-widget>`) [Function]
(*expand* `bool`) (*fill* `bool`) (*padding* `unsigned-int`)

`pack-start` [Method]

Adds *child* to *box*, packed with reference to the start of *box*. The *child* is packed after any other child packed with reference to the start of *box*.

box a `<gtk-box>`.

child the `<gtk-widget>` to be added to *box*.

expand '`#t`' if the new child is to be given extra space allocated to *box*. The extra space will be divided evenly between all children of *box* that use this option.

fill '`#t`' if space given to *child* by the *expand* option is actually allocated to *child*, rather than just padding it. This parameter has no effect if *expand* is set to '`#f`'. A child is always allocated the full height of a `<gtk-hbox>` and the full width of a `<gtk-vbox>`. This option affects the other dimension.

padding extra space in pixels to put between this child and its neighbors, over and above the global amount specified by in `<gtk-box>`. If *child* is a widget at one of the reference ends of *box*, then *padding* pixels are also put between *child* and the reference edge of *box*.

`gtk-box-pack-end` (*self* `<gtk-box>`) (*child* `<gtk-widget>`) [Function]
(*expand* `bool`) (*fill* `bool`) (*padding* `unsigned-int`)

`pack-end` [Method]

Adds *child* to *box*, packed with reference to the end of *box*. The *child* is packed after (away from end of) any other child packed with reference to the end of *box*.

box a `<gtk-box>`.

child the `<gtk-widget>` to be added to *box*.

expand '`#t`' if the new child is to be given extra space allocated to *box*. The extra space will be divided evenly between all children of *box* that use this option.

fill '`#t`' if space given to *child* by the *expand* option is actually allocated to *child*, rather than just padding it. This parameter has no effect if *expand* is set to '`#f`'. A child is always allocated the full height of a `<gtk-hbox>` and the full width of a `<gtk-vbox>`. This option affects the other dimension.

padding extra space in pixels to put between this child and its neighbors, over and above the global amount specified by in `<gtk-box>`. If *child* is a widget at one of the reference ends of *box*, then *padding* pixels are also put between *child* and the reference edge of *box*.

`gtk-box-pack-start-defaults` (*self* `<gtk-box>`) [Function]
(*widget* `<gtk-widget>`)

`pack-start-defaults` [Method]

Adds *widget* to *box*, packed with reference to the start of *box*. The child is packed after any other child packed with reference to the start of *box*.

Parameters for how to pack the child *widget*, , , and in `<gtk-box-child>`, are given their default values, '`#t`', '`#t`', and 0, respectively.

box a `<gtk-box>`.

widget the `<gtk-widget>` to be added to *box*.

`gtk-box-pack-end-defaults` (*self* `<gtk-box>`) [Function]
(*widget* `<gtk-widget>`)
`pack-end-defaults` [Method]

Adds *widget* to *box*, packed with reference to the end of *box*. The child is packed after (away from end of) any other child packed with reference to the end of *box*.

Parameters for how to pack the child *widget*, , , and in `<gtk-box-child>`, are given their default values, '`#t`', '`#t`', and 0, respectively.

box a `<gtk-box>`.

widget the `<gtk-widget>` to be added to *box*.

`gtk-box-get-homogeneous` (*self* `<gtk-box>`) ⇒ (*ret* `bool`) [Function]
`get-homogeneous` [Method]

Returns whether the box is homogeneous (all children are the same size). See `gtk-box-set-homogeneous`.

box a `<gtk-box>`

ret '`#t`' if the box is homogeneous.

`gtk-box-set-homogeneous` (*self* `<gtk-box>`) (*homogeneous* `bool`) [Function]
`set-homogeneous` [Method]

Sets the field of `<gtk-box>`, controlling whether or not all children of *box* are given equal space in the box.

box a `<gtk-box>`.

homogeneous
a boolean value, '`#t`' to create equal allotments, '`#f`' for variable allotments.

`gtk-box-get-spacing` (*self* `<gtk-box>`) ⇒ (*ret* `int`) [Function]
`get-spacing` [Method]

Gets the value set by `gtk-box-set-spacing`.

box a `<gtk-box>`

ret spacing between children

`gtk-box-set-spacing` (*self* `<gtk-box>`) (*spacing* `int`) [Function]
`set-spacing` [Method]

Sets the field of `<gtk-box>`, which is the number of pixels to place between children of *box*.

box a `<gtk-box>`.

spacing the number of pixels to put between children.

`gtk-box-reorder-child` (*self* `<gtk-box>`) (*child* `<gtk-widget>`) (*position* `int`) [Function]

`reorder-child` [Method]

Moves *child* to a new *position* in the list of *box* children. The list is the field of `<gtk-box>`, and contains both widgets packed `<gtk-pack-start>` as well as widgets packed `<gtk-pack-end>`, in the order that these widgets were added to *box*.

A widget's position in the *box* children list determines where the widget is packed into *box*. A child widget at some position in the list will be packed just after all other widgets of the same packing type that appear earlier in the list.

box a `<gtk-box>`.

child the `<gtk-widget>` to move.

position the new position for *child* in the list of `<gtk-box>`, starting from 0. If negative, indicates the end of the list.

`gtk-box-query-child-packing` (*self* `<gtk-box>`) (*child* `<gtk-widget>`) (*pack_type* `<gtk-pack-type*>`) ⇒ (*expand* `bool`) (*fill* `bool`) (*padding* `unsigned-int`) [Function]

`query-child-packing` [Method]

Returns information about how *child* is packed into *box*.

box a `<gtk-box>`.

child the `<gtk-widget>` of the child to query.

expand the returned value of the field in `<gtk-box-child>`.

fill the returned value of the field in `<gtk-box-child>`.

padding the returned value of the field in `<gtk-box-child>`.

pack-type the returned value of the field in `<gtk-box-child>`.

`gtk-box-set-child-packing` (*self* `<gtk-box>`) (*child* `<gtk-widget>`) (*expand* `bool`) (*fill* `bool`) (*padding* `unsigned-int`) (*pack_type* `<gtk-pack-type>`) [Function]

`set-child-packing` [Method]

Sets the way *child* is packed into *box*.

box a `<gtk-box>`.

child the `<gtk-widget>` of the child to set.

expand the new value of the field in `<gtk-box-child>`.

fill the new value of the field in `<gtk-box-child>`.

padding the new value of the field in `<gtk-box-child>`.

pack-type the new value of the field in `<gtk-box-child>`.

132 GtkButtonBox

Base class for and

132.1 Overview

The primary purpose of this class is to keep track of the various properties of `<gtk-hbutton-box>` and `<gtk-vbutton-box>` widgets.

`gtk-button-box-get-child-size` retrieves the minimum width and height for widgets in a given button box. `gtk-button-box-set-child-size` allows those properties to be changed.

The internal padding of buttons can be retrieved and changed per button box using `gtk-button-box-get-child-ipadding` and `gtk-button-box-set-child-ipadding` respectively.

`gtk-button-box-get-spacing` and `gtk-button-box-set-spacing` retrieve and change default number of pixels between buttons, respectively.

`gtk-button-box-get-layout` and `gtk-button-box-set-layout` retrieve and alter the method used to spread the buttons in a button box across the container, respectively.

The main purpose of GtkButtonBox is to make sure the children have all the same size. Therefore it ignores the homogeneous property which it inherited from GtkBox, and always behaves as if homogeneous was '`#t`'.

132.2 Usage

`<gtk-button-box>` [Class]

This `<gobject>` class defines the following properties:

`layout-style`
: How to layout the buttons in the box. Possible values are default, spread, edge, start and end

`gtk-button-box-get-layout` (*self* `<gtk-button-box>`) ⇒ (*ret* `<gtk-button-box-style>`) [Function]

`get-layout` [Method]

Retrieves the method being used to arrange the buttons in a button box.

widget a `<gtk-button-box>`.

ret the method used to layout buttons in *widget*.

`gtk-button-box-get-child-secondary` (*self* `<gtk-button-box>`) (*child* `<gtk-widget>`) ⇒ (*ret* `bool`) [Function]

`get-child-secondary` [Method]

Returns whether *child* should appear in a secondary group of children.

widget a `<gtk-button-box>`

child a child of *widget*

ret whether *child* should appear in a secondary group of children.

Since 2.4

`gtk-button-box-set-layout` (*self* `<gtk-button-box>`) [Function]
(*layout_style* `<gtk-button-box-style>`)
`set-layout` [Method]

Changes the way buttons are arranged in their container.

widget a `<gtk-button-box>`.

layout-style
the new layout style.

`gtk-button-box-set-child-secondary` (*self* `<gtk-button-box>`) [Function]
(*child* `<gtk-widget>`) (*is_secondary* `bool`)
`set-child-secondary` [Method]

Sets whether *child* should appear in a secondary group of children. A typical use of a secondary child is the help button in a dialog.

This group appears after the other children if the style is '`GTK_BUTTONBOX_START`', '`GTK_BUTTONBOX_SPREAD`' or '`GTK_BUTTONBOX_EDGE`', and before the other children if the style is '`GTK_BUTTONBOX_END`'. For horizontal button boxes, the definition of before/after depends on direction of the widget (see `gtk-widget-set-direction`). If the style is '`GTK_BUTTONBOX_START`' or '`GTK_BUTTONBOX_END`', then the secondary children are aligned at the other end of the button box from the main children. For the other styles, they appear immediately next to the main children.

widget a `<gtk-button-box>`

child a child of *widget*

is-secondary
if '`#t`', the *child* appears in a secondary group of the button box.

133 GtkContainer

Base class for widgets which contain other widgets

133.1 Overview

A GTK+ user interface is constructed by nesting widgets inside widgets. Container widgets are the inner nodes in the resulting tree of widgets: they contain other widgets. So, for example, you might have a `<gtk-window>` containing a `<gtk-frame>` containing a GtkLabel. If you wanted an image instead of a textual label inside the frame, you might replace the `<gtk-label>` widget with a `<gtk-image>` widget.

There are two major kinds of container widgets in GTK+. Both are subclasses of the abstract `<gtk-container>` base class.

The first type of container widget has a single child widget and derives from `<gtk-bin>`. These containers are *decorators*, which add some kind of functionality to the child. For example, a `<gtk-button>` makes its child into a clickable button; a `<gtk-frame>` draws a frame around its child and a `<gtk-window>` places its child widget inside a top-level window.

The second type of container can have more than one child; its purpose is to manage *layout*. This means that these containers assign sizes and positions to their children. For example, a `<gtk-hbox>` arranges its children in a horizontal row, and a `<gtk-table>` arranges the widgets it contains in a two-dimensional grid.

To fulfill its task, a layout container must negotiate the size requirements with its parent and its children. This negotiation is carried out in two phases, *size requisition* and *size allocation*.

133.2 Size Requisition

The size requisition of a widget is it's desired width and height. This is represented by a `<gtk-requisition>`.

How a widget determines its desired size depends on the widget. A `<gtk-label>`, for example, requests enough space to display all its text. Container widgets generally base their size request on the requisitions of their children.

The size requisition phase of the widget layout process operates top-down. It starts at a top-level widget, typically a `<gtk-window>`. The top-level widget asks its child for its size requisition by calling `gtk-widget-size-request`. To determine its requisition, the child asks its own children for their requisitions and so on. Finally, the top-level widget will get a requisition back from its child.

133.3 Size Allocation

When the top-level widget has determined how much space its child would like to have, the second phase of the size negotiation, size allocation, begins. Depending on its configuration (see `gtk-window-set-resizable`), the top-level widget may be able to expand in order to satisfy the size request or it may have to ignore the size request and keep its fixed size. It then tells its child widget how much space it gets by calling `gtk-widget-size-allocate`. The child widget divides the space among its children and tells each child how much space

it got, and so on. Under normal circumstances, a <gtk-window> will always give its child the amount of space the child requested.

A child's size allocation is represented by a <gtk-allocation>. This struct contains not only a width and height, but also a position (i.e. X and Y coordinates), so that containers can tell their children not only how much space they have gotten, but also where they are positioned inside the space available to the container.

Widgets are required to honor the size allocation they receive; a size request is only a request, and widgets must be able to cope with any size.

133.4 Child properties

introduces *child properties* - these are object properties that are not specific to either the container or the contained widget, but rather to their relation. Typical examples of child properties are the position or pack-type of a widget which is contained in a <gtk-box>.

Use gtk-container-class-install-child-property to install child properties for a container class and gtk-container-class-find-child-property or gtk-container-class-list-child-properties to get information about existing child properties.

To set the value of a child property, use gtk-container-child-set-property, gtk-container-child-set or gtk-container-child-set-valist. To obtain the value of a child property, use gtk-container-child-get-property, gtk-container-child-get or gtk-container-child-get-valist. To emit notification about child property changes, use gtk-widget-child-notify.

133.5 Usage

<gtk-container> [Class]

This <gobject> class defines the following properties:

border-width
: The width of the empty border outside the containers children

resize-mode
: Specify how resize events are handled

child
: Can be used to add a new child to the container

add (*arg0* <gtk-widget>) [Signal on <gtk-container>]

remove (*arg0* <gtk-widget>) [Signal on <gtk-container>]

check-resize [Signal on <gtk-container>]

set-focus-child (*arg0* <gtk-widget>) [Signal on <gtk-container>]

gtk-container-add (*self* <gtk-container>) (*widget* <gtk-widget>) [Function]
add [Method]

Adds *widget* to *container*. Typically used for simple containers such as <gtk-window>, <gtk-frame>, or <gtk-button>; for more complicated layout containers such as <gtk-box> or <gtk-table>, this function will pick default packing parameters that may

not be correct. So consider functions such as `gtk-box-pack-start` and `gtk-table-attach` as an alternative to `gtk-container-add` in those cases. A widget may be added to only one container at a time; you can't place the same widget inside two different containers.

container a `<gtk-container>`

widget a widget to be placed inside *container*

`gtk-container-remove` (*self* `<gtk-container>`) [Function]
(*widget* `<gtk-widget>`)
`remove` [Method]

Removes *widget* from *container*. *widget* must be inside *container*. Note that *container* will own a reference to *widget*, and that this may be the last reference held; so removing a widget from its container can destroy that widget. If you want to use *widget* again, you need to add a reference to it while it's not inside a container, using `g-object-ref`. If you don't want to use *widget* again it's usually more efficient to simply destroy it directly using `gtk-widget-destroy` since this will remove it from the container and help break any circular reference count cycles.

container a `<gtk-container>`

widget a current child of *container*

`gtk-container-get-resize-mode` (*self* `<gtk-container>`) [Function]
⇒ (*ret* `<gtk-resize-mode>`)
`get-resize-mode` [Method]

Returns the resize mode for the container. See `gtk-container-set-resize-mode`.

container a `<gtk-container>`

ret the current resize mode

`gtk-container-set-resize-mode` (*self* `<gtk-container>`) [Function]
(*resize_mode* `<gtk-resize-mode>`)
`set-resize-mode` [Method]

Sets the resize mode for the container.

The resize mode of a container determines whether a resize request will be passed to the container's parent, queued for later execution or executed immediately.

container a `<gtk-container>`.

resize-mode
the new resize mode.

`gtk-container-check-resize` (*self* `<gtk-container>`) [Function]
`check-resize` [Method]

container

`gtk-container-get-children` (*self* `<gtk-container>`) [Function]
⇒ (*ret* `glist-of`)
`get-children` [Method]

Returns the container's non-internal children. See `gtk-container-forall` for details on what constitutes an "internal" child.

container a `<gtk-container>`.

ret a newly-allocated list of the container's non-internal children.

`gtk-container-set-focus-child` (*self* `<gtk-container>`) [Function]
(*child* `<gtk-widget>`)
`set-focus-child` [Method]

container

child

`gtk-container-get-focus-vadjustment` (*self* `<gtk-container>`) [Function]
⇒ (*ret* `<gtk-adjustment>`)
`get-focus-vadjustment` [Method]

Retrieves the vertical focus adjustment for the container. See `gtk-container-set-focus-vadjustment`.

container a `<gtk-container>`

ret the vertical focus adjustment, or '`#f`' if none has been set.

`gtk-container-set-focus-vadjustment` (*self* `<gtk-container>`) [Function]
(*adjustment* `<gtk-adjustment>`)
`set-focus-vadjustment` [Method]

Hooks up an adjustment to focus handling in a container, so when a child of the container is focused, the adjustment is scrolled to show that widget. This function sets the vertical alignment. See `gtk-scrolled-window-get-vadjustment` for a typical way of obtaining the adjustment and `gtk-container-set-focus-hadjustment` for setting the horizontal adjustment.

The adjustments have to be in pixel units and in the same coordinate system as the allocation for immediate children of the container.

container a `<gtk-container>`

adjustment
an adjustment which should be adjusted when the focus is moved among the descendents of *container*

`gtk-container-get-focus-hadjustment` (*self* `<gtk-container>`) [Function]
⇒ (*ret* `<gtk-adjustment>`)
`get-focus-hadjustment` [Method]

Retrieves the horizontal focus adjustment for the container. See `gtk-container-set-focus-hadjustment`.

container a `<gtk-container>`

ret the horizontal focus adjustment, or '`#f`' if none has been set.

`gtk-container-set-focus-hadjustment` (*self* `<gtk-container>`) [Function]
(*adjustment* `<gtk-adjustment>`)
`set-focus-hadjustment` [Method]

Hooks up an adjustment to focus handling in a container, so when a child of the container is focused, the adjustment is scrolled to show that widget. This function sets

the horizontal alignment. See `gtk-scrolled-window-get-hadjustment` for a typical way of obtaining the adjustment and `gtk-container-set-focus-vadjustment` for setting the vertical adjustment.

The adjustments have to be in pixel units and in the same coordinate system as the allocation for immediate children of the container.

container a `<gtk-container>`

adjustment
an adjustment which should be adjusted when the focus is moved among the descendents of *container*

`gtk-container-resize-children` (*self* `<gtk-container>`) [Function]
`resize-children` [Method]

container

`gtk-container-child-type` (*self* `<gtk-container>`) [Function]
⇒ (*ret* `<gtype>`)
`child-type` [Method]

Returns the type of the children supported by the container.

Note that this may return '`G_TYPE_NONE`' to indicate that no more children can be added, e.g. for a `<gtk-paned>` which already has two children.

container a `<gtk-container>`.

ret a `<g-type>`.

`gtk-container-child-get-property` (*self* `<gtk-container>`) [Function]
(*child* `<gtk-widget>`) (*property_name* `mchars`) (*value* `<gvalue>`)
`child-get-property` [Method]

Gets the value of a child property for *child* and *container*.

container a `<gtk-container>`

child a widget which is a child of *container*

property-name
the name of the property to get

value a location to return the value

`gtk-container-child-set-property` (*self* `<gtk-container>`) [Function]
(*child* `<gtk-widget>`) (*property_name* `mchars`) (*value* `<gvalue>`)
`child-set-property` [Method]

Sets a child property for *child* and *container*.

container a `<gtk-container>`

child a widget which is a child of *container*

property-name
the name of the property to set

value the value to set the property to

`gtk-container-get-border-width` (*self* `<gtk-container>`) [Function]
⇒ (*ret* `unsigned-int`)

`get-border-width` [Method]

Retrieves the border width of the container. See `gtk-container-set-border-width`.

container a `<gtk-container>`

ret the current border width

`gtk-container-set-border-width` (*self* `<gtk-container>`) [Function]
(*border_width* `unsigned-int`)

`set-border-width` [Method]

Sets the border width of the container.

The border width of a container is the amount of space to leave around the outside of the container. The only exception to this is `<gtk-window>`; because toplevel windows can't leave space outside, they leave the space inside. The border is added on all sides of the container. To add space to only one side, one approach is to create a `<gtk-alignment>` widget, call `gtk-widget-set-usize` to give it a size, and place it on the side of the container as a spacer.

container a `<gtk-container>`

border-width
amount of blank space to leave *outside* the container. Valid values are in the range 0-65535 pixels.

`gtk-container-propagate-expose` (*self* `<gtk-container>`) [Function]
(*child* `<gtk-widget>`) (*event* `<gdk-event-expose>`)

`propagate-expose` [Method]

When a container receives an expose event, it must send synthetic expose events to all children that don't have their own `<gdk-windows>`. This function provides a convenient way of doing this. A container, when it receives an expose event, calls `gtk-container-propagate-expose` once for each child, passing in the event the container received.

`gtk-container-propagate-expose` takes care of deciding whether an expose event needs to be sent to the child, intersecting the event's area with the child area, and sending the event.

In most cases, a container can simply either simply inherit the ::expose implementation from `<gtk-container>`, or, do some drawing and then chain to the ::expose implementation from `<gtk-container>`.

container a `<gtk-container>`

child a child of *container*

event a expose event sent to container

`gtk-container-get-focus-chain` (*self* `<gtk-container>`) [Function]
(*focusable_widgets* `<g-list**>`) ⇒ (*ret* `bool`)

`get-focus-chain` [Method]

Retrieves the focus chain of the container, if one has been set explicitly. If no focus chain has been explicitly set, GTK+ computes the focus chain based on the positions of the children. In that case, GTK+ stores '`#f`' in *focusable-widgets* and returns '`#f`'.

container a `<gtk-container>`

focusable-widgets
location to store the focus chain of the container, or '`#f`'. You should free this list using `g-list-free` when you are done with it, however no additional reference count is added to the individual widgets in the focus chain.

ret '`#t`' if the focus chain of the container has been set explicitly.

`gtk-container-set-focus-chain` (*self* `<gtk-container>`) [Function]
(*focusable_widgets* `glist-of`)
`set-focus-chain` [Method]

Sets a focus chain, overriding the one computed automatically by GTK+.

In principle each widget in the chain should be a descendant of the container, but this is not enforced by this method, since it's allowed to set the focus chain before you pack the widgets, or have a widget in the chain that isn't always packed. The necessary checks are done when the focus chain is actually traversed.

container a `<gtk-container>`.

focusable-widgets
the new focus chain.

`gtk-container-unset-focus-chain` (*self* `<gtk-container>`) [Function]
`unset-focus-chain` [Method]

Removes a focus chain explicitly set with `gtk-container-set-focus-chain`.

container a `<gtk-container>`.

134 GtkItem

Abstract base class for GtkMenuItem, GtkListItem and GtkTreeItem

134.1 Overview

The <gtk-item> widget is an abstract base class for <gtk-menu-item>, <gtk-list-item> and <gtk-tree-item>.

134.2 Usage

<gtk-item> [Class]
This <gobject> class defines no properties, other than those defined by its superclasses.

select [Signal on <gtk-item>]
Emitted when the item is selected.

deselect [Signal on <gtk-item>]
Emitted when the item is deselected.

toggle [Signal on <gtk-item>]
Emitted when the item is toggled.

gtk-item-select (*self* <gtk-item>) [Function]
select [Method]
Emits the "select" signal on the given item.

item a <gtk-item>.

gtk-item-deselect (*self* <gtk-item>) [Function]
deselect [Method]
Emits the "deselect" signal on the given item.

item a <gtk-item>.

gtk-item-toggle (*self* <gtk-item>) [Function]
toggle [Method]
Emits the "toggle" signal on the given item.

item a <gtk-item>.

135 GtkMisc

Base class for widgets with alignments and padding

135.1 Overview

The <gtk-misc> widget is an abstract widget which is not useful itself, but is used to derive subclasses which have alignment and padding attributes.

The horizontal and vertical padding attributes allows extra space to be added around the widget.

The horizontal and vertical alignment attributes enable the widget to be positioned within its allocated area. Note that if the widget is added to a container in such a way that it expands automatically to fill its allocated area, the alignment settings will not alter the widgets position.

135.2 Usage

<gtk-misc> [Class]

This <gobject> class defines the following properties:

- xalign — The horizontal alignment, from 0 (left) to 1 (right). Reversed for RTL layouts.
- yalign — The vertical alignment, from 0 (top) to 1 (bottom)
- xpad — The amount of space to add on the left and right of the widget, in pixels
- ypad — The amount of space to add on the top and bottom of the widget, in pixels

gtk-misc-set-alignment (*self* <gtk-misc>) (*xalign* float) (*yalign* float) [Function]
set-alignment [Method]

Sets the alignment of the widget.

- *misc* — a <gtk-misc>.
- *xalign* — the horizontal alignment, from 0 (left) to 1 (right).
- *yalign* — the vertical alignment, from 0 (top) to 1 (bottom).

gtk-misc-set-padding (*self* <gtk-misc>) (*xpad* int) (*ypad* int) [Function]
set-padding [Method]

Sets the amount of space to add around the widget.

- *misc* — a <gtk-misc>.
- *xpad* — the amount of space to add on the left and right of the widget, in pixels.
- *ypad* — the amount of space to add on the top and bottom of the widget, in pixels.

`gtk-misc-get-alignment` (*self* `<gtk-misc>`) ⇒ (*xalign* `float`) (*yalign* `float`) [Function]

`get-alignment` [Method]

Gets the X and Y alignment of the widget within its allocation. See `gtk-misc-set-alignment`.

misc a `<gtk-misc>`

xalign location to store X alignment of *misc*, or '`#f`'

yalign location to store Y alignment of *misc*, or '`#f`'

`gtk-misc-get-padding` (*self* `<gtk-misc>`) ⇒ (*xpad* `int`) (*ypad* `int`) [Function]

`get-padding` [Method]

Gets the padding in the X and Y directions of the widget. See `gtk-misc-set-padding`.

misc a `<gtk-misc>`

xpad location to store padding in the X direction, or '`#f`'

ypad location to store padding in the Y direction, or '`#f`'

136 GtkObject

The base class of the GTK+ type hierarchy

136.1 Overview

136.2 Description

<gtk-object> is the base class for all widgets, and for a few non-widget objects such as <gtk-adjustment>. <gtk-object> predates <gobject>; non-widgets that derive from <gtk-object> rather than <gobject> do so for backward compatibility reasons.

<gtk-object>s are created with a "floating" reference count. This means that the initial reference is not owned by anyone. Calling g-object-unref on a newly-created <gtk-object> is incorrect, the floating reference has to be removed first. This can be done by anyone at any time, by calling g-object-ref-sink to convert the floating reference into a regular reference. g-object-ref-sink returns a new reference if an object is already sunk (has no floating reference).

When you add a widget to its parent container, the parent container will do this: This means that the container now owns a reference to the child widget and the child widget has no floating reference.

```
g_object_ref_sink (G_OBJECT (child_widget));
```

The purpose of the floating reference is to keep the child widget alive until you add it to a parent container:

```
button = gtk_button_new ();
/* button has one floating reference to keep it alive */
gtk_container_add (GTK_CONTAINER (container), button);
/* button has one non-floating reference owned by the container */
```

<gtk-window> is a special case, because GTK+ itself will ref/sink it on creation. That is, after calling gtk-window-new, the <gtk-window> will have one reference which is owned by GTK+, and no floating references.

One more factor comes into play: the "destroy" signal, emitted by the gtk-object-destroy method. The "destroy" signal asks all code owning a reference to an object to release said reference. So, for example, if you call gtk-object-destroy on a <gtk-window>, GTK+ will release the reference count that it owns; if you call gtk-object-destroy on a <gtk-button>, then the button will be removed from its parent container and the parent container will release its reference to the button. Because these references are released, calling gtk-object-destroy should result in freeing all memory associated with an object, unless some buggy code fails to release its references in response to the "destroy" signal. Freeing memory (referred to as *finalization* only happens if the reference count reaches zero.

Some simple rules for handling <gtk-object:>

Never call g-object-unref unless you have previously called g-object-ref, even if you created the <gtk-object>. (Note: this is *not* true for <gobject>; for <gobject>, the creator of the object owns a reference.)

Call `gtk-object-destroy` to get rid of most objects in most cases. In particular, widgets are almost always destroyed in this way.

Because of the floating reference count, you don't need to worry about reference counting for widgets and toplevel windows, unless you explicitly call `g-object-ref` yourself.

136.3 Usage

`<gtk-object>` [Class]

This `<gobject>` class defines the following properties:

`user-data`

Anonymous User Data Pointer

`destroy` [Signal on `<gtk-object>`]

Signals that all holders of a reference to the `<gtk-object>` should release the reference that they hold. May result in finalization of the object if all references are released.

137 GtkPaned

Base class for widgets with two adjustable panes

137.1 Overview

<gtk-paned> is the base class for widgets with two panes, arranged either horizontally (<gtk-hpaned>) or vertically (<gtk-vpaned>). Child widgets are added to the panes of the widget with gtk-paned-pack1 and gtk-paned-pack2. The division beween the two children is set by default from the size requests of the children, but it can be adjusted by the user.

A paned widget draws a separator between the two child widgets and a small handle that the user can drag to adjust the division. It does not draw any relief around the children or around the separator. (The space in which the separator is called the gutter.) Often, it is useful to put each child inside a <gtk-frame> with the shadow type set to 'GTK_SHADOW_IN' so that the gutter appears as a ridge.

Each child has two options that can be set, *resize* and *shrink*. If *resize* is true, then when the <gtk-paned> is resized, that child will expand or shrink along with the paned widget. If *shrink* is true, then when that child can be made smaller than its requisition by the user. Setting *shrink* to '#f' allows the application to set a minimum size. If *resize* is false for both children, then this is treated as if *resize* is true for both children.

The application can set the position of the slider as if it were set by the user, by calling gtk-paned-set-position.

```
GtkWidget *hpaned = gtk_hpaned_new ();
GtkWidget *frame1 = gtk_frame_new (NULL);
GtkWidget *frame2 = gtk_frame_new (NULL);
gtk_frame_set_shadow_type (GTK_FRAME (frame1), GTK_SHADOW_IN);
gtk_frame_set_shadow_type (GTK_FRAME (frame2), GTK_SHADOW_IN);

gtk_widget_set_size_request (hpaned, 200 + GTK_PANED (hpaned)->gutter_size, -1);

gtk_paned_pack1 (GTK_PANED (hpaned), frame1, TRUE, FALSE);
gtk_widget_set_size_request (frame1, 50, -1);

gtk_paned_pack2 (GTK_PANED (hpaned), frame2, FALSE, FALSE);
gtk_widget_set_size_request (frame2, 50, -1);
```

137.2 Usage

<gtk-paned> [Class]

This <gobject> class defines the following properties:

position Position of paned separator in pixels (0 means all the way to the left/top)

position-set
TRUE if the Position property should be used

`min-position`
: Smallest possible value for the "position" property

`max-position`
: Largest possible value for the "position" property

`cycle-child-focus` (*arg0* `<gboolean>`) ⇒ `<gboolean>` [Signal on `<gtk-paned>`]

`toggle-handle-focus` ⇒ `<gboolean>` [Signal on `<gtk-paned>`]

`move-handle` (*arg0* `<gtk-scroll-type>`) ⇒ `<gboolean>` [Signal on `<gtk-paned>`]

`cycle-handle-focus` (*arg0* `<gboolean>`) ⇒ `<gboolean>` [Signal on `<gtk-paned>`]

`accept-position` ⇒ `<gboolean>` [Signal on `<gtk-paned>`]

`cancel-position` ⇒ `<gboolean>` [Signal on `<gtk-paned>`]

`gtk-paned-add1` (*self* `<gtk-paned>`) (*child* `<gtk-widget>`) [Function]
`add1` [Method]

Adds a child to the top or left pane with default parameters. This is equivalent to '`gtk_paned_pack1 (paned, child, FALSE, TRUE)`'.

paned a paned widget

child the child to add

`gtk-paned-add2` (*self* `<gtk-paned>`) (*child* `<gtk-widget>`) [Function]
`add2` [Method]

Adds a child to the bottom or right pane with default parameters. This is equivalent to '`gtk_paned_pack2 (paned, child, TRUE, TRUE)`'.

paned a paned widget

child the child to add

`gtk-paned-pack1` (*self* `<gtk-paned>`) (*child* `<gtk-widget>`) [Function]
(*resize* `bool`) (*shrink* `bool`)
`pack1` [Method]

Adds a child to the top or left pane.

paned a paned widget

child the child to add

resize should this child expand when the paned widget is resized.

shrink can this child be made smaller than its requisition.

`gtk-paned-pack2` (*self* `<gtk-paned>`) (*child* `<gtk-widget>`) [Function]
(*resize* `bool`) (*shrink* `bool`)
`pack2` [Method]

Adds a child to the bottom or right pane.

paned a paned widget

child the child to add

resize should this child expand when the paned widget is resized.

shrink can this child be made smaller than its requisition.

`gtk-paned-get-child1` (*self* `<gtk-paned>`) ⇒ (*ret* `<gtk-widget>`) [Function]
`get-child1` [Method]

Obtains the first child of the paned widget.

paned a `<gtk-paned>` widget

ret first child, or '`#f`' if it is not set.

Since 2.4

`gtk-paned-get-child2` (*self* `<gtk-paned>`) ⇒ (*ret* `<gtk-widget>`) [Function]
`get-child2` [Method]

Obtains the second child of the paned widget.

paned a `<gtk-paned>` widget

ret second child, or '`#f`' if it is not set.

Since 2.4

`gtk-paned-set-position` (*self* `<gtk-paned>`) (*position* `int`) [Function]
`set-position` [Method]

Sets the position of the divider between the two panes.

paned a `<gtk-paned>` widget

position pixel position of divider, a negative value means that the position is unset.

`gtk-paned-get-position` (*self* `<gtk-paned>`) ⇒ (*ret* `int`) [Function]
`get-position` [Method]

Obtains the position of the divider between the two panes.

paned a `<gtk-paned>` widget

ret position of the divider

138 GtkRange

Base class for widgets which visualize an adjustment

138.1 Overview

138.2 Usage

`<gtk-range>` [Class]

This `<gobject>` class defines the following properties:

`update-policy`
How the range should be updated on the screen

`adjustment`
The GtkAdjustment that contains the current value of this range object

`inverted` Invert direction slider moves to increase range value

`lower-stepper-sensitivity`
The sensitivity policy for the stepper that points to the adjustment's lower side

`upper-stepper-sensitivity`
The sensitivity policy for the stepper that points to the adjustment's upper side

`show-fill-level`
Whether to display a fill level indicator graphics on trough.

`restrict-to-fill-level`
Whether to restrict the upper boundary to the fill level.

`fill-level`
The fill level.

`value-changed` [Signal on `<gtk-range>`]

Emitted when the range value changes.

`adjust-bounds` (*arg0* `<gdouble>`) [Signal on `<gtk-range>`]

`move-slider` (*arg0* `<gtk-scroll-type>`) [Signal on `<gtk-range>`]

Virtual function that moves the slider. Used for keybindings.

`change-value` (*arg0* `<gtk-scroll-type>`) [Signal on `<gtk-range>`]
(*arg1* `<gdouble>`) ⇒ `<gboolean>`

The ::change-value signal is emitted when a scroll action is performed on a range. It allows an application to determine the type of scroll event that occurred and the resultant new value. The application can handle the event itself and return '`#t`' to prevent further processing. Or, by returning '`#f`', it can pass the event to other handlers until the default GTK+ handler is reached.

The value parameter is unrounded. An application that overrides the ::change-value signal is responsible for clamping the value to the desired number of decimal digits; the default GTK+ handler clamps the value based on *range->round-digits*.

It is not possible to use delayed update policies in an overridden ::change-value handler.

Since 2.6

`gtk-range-get-adjustment` (*self* `<gtk-range>`) [Function]
⇒ (*ret* `<gtk-adjustment>`)

`get-adjustment` [Method]

Get the `<gtk-adjustment>` which is the "model" object for `<gtk-range>`. See `gtk-range-set-adjustment` for details. The return value does not have a reference added, so should not be unreferenced.

range a `<gtk-range>`

ret a `<gtk-adjustment>`

`gtk-range-set-update-policy` (*self* `<gtk-range>`) [Function]
(*policy* `<gtk-update-type>`)

`set-update-policy` [Method]

Sets the update policy for the range. `<gtk-update-continuous>` means that anytime the range slider is moved, the range value will change and the value_changed signal will be emitted. `<gtk-update-delayed>` means that the value will be updated after a brief timeout where no slider motion occurs, so updates are spaced by a short time rather than continuous. `<gtk-update-discontinuous>` means that the value will only be updated when the user releases the button and ends the slider drag operation.

range a `<gtk-range>`

policy update policy

`gtk-range-set-adjustment` (*self* `<gtk-range>`) [Function]
(*adjustment* `<gtk-adjustment>`)

`set-adjustment` [Method]

Sets the adjustment to be used as the "model" object for this range widget. The adjustment indicates the current range value, the minimum and maximum range values, the step/page increments used for keybindings and scrolling, and the page size. The page size is normally 0 for `<gtk-scale>` and nonzero for `<gtk-scrollbar>`, and indicates the size of the visible area of the widget being scrolled. The page size affects the size of the scrollbar slider.

range a `<gtk-range>`

adjustment
a `<gtk-adjustment>`

`gtk-range-get-inverted` (*self* `<gtk-range>`) ⇒ (*ret* `bool`) [Function]

`get-inverted` [Method]

Gets the value set by `gtk-range-set-inverted`.

range a `<gtk-range>`

ret '`#t`' if the range is inverted

`gtk-range-set-inverted` (*self* `<gtk-range>`) (*setting* `bool`) [Function]
`set-inverted` [Method]
Ranges normally move from lower to higher values as the slider moves from top to bottom or left to right. Inverted ranges have higher values at the top or on the right rather than on the bottom or left.

range a `<gtk-range>`

setting '`#t`' to invert the range

`gtk-range-get-update-policy` (*self* `<gtk-range>`) [Function]
⇒ (*ret* `<gtk-update-type>`)
`get-update-policy` [Method]
Gets the update policy of *range*. See `gtk-range-set-update-policy`.

range a `<gtk-range>`

ret the current update policy

`gtk-range-get-value` (*self* `<gtk-range>`) ⇒ (*ret* `double`) [Function]
`get-value` [Method]
Gets the current value of the range.

range a `<gtk-range>`

ret current value of the range.

`gtk-range-set-increments` (*self* `<gtk-range>`) (*step* `double`) [Function]
(*page* `double`)
`set-increments` [Method]
Sets the step and page sizes for the range. The step size is used when the user clicks the `<gtk-scrollbar>` arrows or moves `<gtk-scale>` via arrow keys. The page size is used for example when moving via Page Up or Page Down keys.

range a `<gtk-range>`

step step size

page page size

`gtk-range-set-range` (*self* `<gtk-range>`) (*min* `double`) (*max* `double`) [Function]
`set-range` [Method]
Sets the allowable values in the `<gtk-range>`, and clamps the range value to be between *min* and *max*. (If the range has a non-zero page size, it is clamped between *min* and *max* - page-size.)

range a `<gtk-range>`

min minimum range value

max maximum range value

`gtk-range-set-value` (*self* `<gtk-range>`) (*value* `double`) [Function]
`set-value` [Method]

Sets the current value of the range; if the value is outside the minimum or maximum range values, it will be clamped to fit inside them. The range emits the "value_changed" signal if the value changes.

range a `<gtk-range>`

value new value of the range

139 GtkScale

Base class for GtkHScale and GtkVScale

139.1 Overview

A <gtk-scale> is a slider control used to select a numeric value. To use it, you'll probably want to investigate the methods on its base class, <gtk-range>, in addition to the methods for <gtk-scale> itself. To set the value of a scale, you would normally use gtk-range-set-value. To detect changes to the value, you would normally use the "value_changed" signal.

The <gtk-scale> widget is an abstract class, used only for deriving the subclasses <gtk-hscale> and <gtk-vscale>. To create a scale widget, call gtk-hscale-new-with-range or gtk-vscale-new-with-range.

139.2 Usage

<gtk-scale> [Class]

This <gobject> class defines the following properties:

digits The number of decimal places that are displayed in the value

draw-value
Whether the current value is displayed as a string next to the slider

value-pos
The position in which the current value is displayed

format-value (*arg0* <gdouble>) ⇒ <gchararray> [Signal on <gtk-scale>]

Signal which allows you to change how the scale value is displayed. Connect a signal handler which returns an allocated string representing *value*. That string will then be used to display the scale's value. Here's an example signal handler which displays a value 1.0 as with "–>1.0<–".

```
static gchar*
format_value_callback (GtkScale *scale,
                       gdouble   value)
{
  return g_strdup_printf ("-->%0.*g<--",
                           gtk_scale_get_digits (scale), value);
}
```

gtk-scale-set-digits (*self* <gtk-scale>) (*digits* int) [Function]
set-digits [Method]

Sets the number of decimal places that are displayed in the value. Also causes the value of the adjustment to be rounded off to this number of digits, so the retrieved value matches the value the user saw.

scale a <gtk-scale>.

digits the number of decimal places to display, e.g. use 1 to display 1.0, 2 to display 1.00 etc.

`gtk-scale-set-draw-value` (*self* `<gtk-scale>`) (*draw_value* `bool`) [Function]
`set-draw-value` [Method]
Specifies whether the current value is displayed as a string next to the slider.

scale a `<gtk-scale>`.

draw-value
a boolean.

`gtk-scale-set-value-pos` (*self* `<gtk-scale>`) [Function]
(*pos* `<gtk-position-type>`)
`set-value-pos` [Method]
Sets the position in which the current value is displayed.

scale a `<gtk-scale>`.

pos the position in which the current value is displayed.

`gtk-scale-get-digits` (*self* `<gtk-scale>`) ⇒ (*ret* `int`) [Function]
`get-digits` [Method]
Gets the number of decimal places that are displayed in the value.

scale a `<gtk-scale>`.

ret the number of decimal places that are displayed.

`gtk-scale-get-draw-value` (*self* `<gtk-scale>`) ⇒ (*ret* `bool`) [Function]
`get-draw-value` [Method]
Returns whether the current value is displayed as a string next to the slider.

scale a `<gtk-scale>`.

ret whether the current value is displayed as a string.

`gtk-scale-get-value-pos` (*self* `<gtk-scale>`) [Function]
⇒ (*ret* `<gtk-position-type>`)
`get-value-pos` [Method]
Gets the position in which the current value is displayed.

scale a `<gtk-scale>`.

ret the position in which the current value is displayed.

`gtk-scale-get-layout` (*self* `<gtk-scale>`) [Function]
⇒ (*ret* `<pango-layout*>`)
`get-layout` [Method]
Gets the `<pango-layout>` used to display the scale. The returned object is owned by the scale so does not need to be freed by the caller.

scale A `<gtk-scale>`

ret the `<pango-layout>` for this scale, or '`#f`' if the draw_value property is '`#f`'.

Since 2.4

`gtk-scale-get-layout-offsets` (*self* `<gtk-scale>`) ⇒ (*x* `int`) (*y* `int`) [Function]

`get-layout-offsets` [Method]

Obtains the coordinates where the scale will draw the `<pango-layout>` representing the text in the scale. Remember when using the `<pango-layout>` function you need to convert to and from pixels using `pango-pixels` or `<pango-scale>`.

If the draw_value property is '`#f`', the return values are undefined.

scale a `<gtk-scale>`

x location to store X offset of layout, or '`#f`'

y location to store Y offset of layout, or '`#f`'

Since 2.4

140 GtkScrollbar

Base class for GtkHScrollbar and GtkVScrollbar

140.1 Overview

The <gtk-scrollbar> widget is an abstract base class for <gtk-hscrollbar> and <gtk-vscrollbar>. It is not very useful in itself.

The position of the thumb in a scrollbar is controlled by the scroll adjustments. See <gtk-adjustment> for the fields in an adjustment - for <gtk-scrollbar>, the "value" field represents the position of the scrollbar, which must be between the "lower" field and "upper - page_size." The "page_size" field represents the size of the visible scrollable area. The "step_increment" and "page_increment" fields are used when the user asks to step down (using the small stepper arrows) or page down (using for example the PageDown key).

140.2 Usage

<gtk-scrollbar> [Class]

This <gobject> class defines no properties, other than those defined by its superclasses.

141 GtkSeparator

Base class for and

141.1 Overview

The <gtk-separator> widget is an abstract class, used only for deriving the subclasses <gtk-hseparator> and <gtk-vseparator>.

141.2 Usage

<gtk-separator> [Class]
: This <gobject> class defines no properties, other than those defined by its superclasses.

142 GtkWidget

Base class for all widgets

142.1 Overview

introduces *style properties* - these are basically object properties that are stored not on the object, but in the style object associated to the widget. Style properties are set in resource files. This mechanism is used for configuring such things as the location of the scrollbar arrows through the theme, giving theme authors more control over the look of applications without the need to write a theme engine in C.

Use `gtk-widget-class-install-style-property` to install style properties for a widget class, `gtk-widget-class-find-style-property` or `gtk-widget-class-list-style-properties` to get information about existing style properties and `gtk-widget-style-get-property`, `gtk-widget-style-get` or `gtk-widget-style-get-valist` to obtain the value of a style property.

142.2 Usage

`<gtk-widget>` [Class]

This `<gobject>` class defines the following properties:

`name` The name of the widget

`parent` The parent widget of this widget. Must be a Container widget

`width-request`
Override for width request of the widget, or -1 if natural request should be used

`height-request`
Override for height request of the widget, or -1 if natural request should be used

`visible` Whether the widget is visible

`sensitive`
Whether the widget responds to input

`app-paintable`
Whether the application will paint directly on the widget

`can-focus`
Whether the widget can accept the input focus

`has-focus`
Whether the widget has the input focus

`is-focus` Whether the widget is the focus widget within the toplevel

`can-default`
Whether the widget can be the default widget

`has-default`
: Whether the widget is the default widget

`receives-default`
: If TRUE, the widget will receive the default action when it is focused

`composite-child`
: Whether the widget is part of a composite widget

`style` The style of the widget, which contains information about how it will look (colors etc)

`events` The event mask that decides what kind of GdkEvents this widget gets

`extension-events`
: The mask that decides what kind of extension events this widget gets

`no-show-all`
: Whether gtk_widget_show_all() should not affect this widget

`has-tooltip`
: Whether this widget has a tooltip

`tooltip-markup`
: The contents of the tooltip for this widget

`tooltip-text`
: The contents of the tooltip for this widget

`composited-changed` [Signal on `<gtk-widget>`]

`show` [Signal on `<gtk-widget>`]

`hide` [Signal on `<gtk-widget>`]

`map` [Signal on `<gtk-widget>`]

`unmap` [Signal on `<gtk-widget>`]

`realize` [Signal on `<gtk-widget>`]

`unrealize` [Signal on `<gtk-widget>`]

`size-request` (*arg0* `<gtk-requisition>`) [Signal on `<gtk-widget>`]

`size-allocate` (*arg0* `<gdk-rectangle>`) [Signal on `<gtk-widget>`]

`state-changed` (*arg0* `<gtk-state-type>`) [Signal on `<gtk-widget>`]

`parent-set` (*arg0* `<gtk-widget>`) [Signal on `<gtk-widget>`]
The parent-set signal is emitted when a new parent has been set on a widget.

`hierarchy-changed` (*arg0* `<gtk-widget>`) [Signal on `<gtk-widget>`]
Emitted when there is a chance in the hierarchy to which a widget belong. More precisely, a widget is *anchored* when its toplevel ancestor is a `<gtk-window>`. This signal is emitted when a widget changes from un-anchored to anchored or vice-versa.

`style-set` (*arg0* `<gtk-style>`) [Signal on `<gtk-widget>`]
The style-set signal is emitted when a new style has been set on a widget. Note that style-modifying functions like `gtk-widget-modify-base` also cause this signal to be emitted.

`direction-changed` (*arg0* `<gtk-text-direction>`) [Signal on `<gtk-widget>`]

`grab-notify` (*arg0* `<gboolean>`) [Signal on `<gtk-widget>`]
The ::grab-notify signal is emitted when a widget becomes shadowed by a GTK+ grab (not a pointer or keyboard grab) on another widget, or when it becomes unshadowed due to a grab being removed.

A widget is shadowed by a `gtk-grab-add` when the topmost grab widget in the grab stack of its window group is not its ancestor.

`child-notify` (*arg0* `<gparam>`) [Signal on `<gtk-widget>`]
The ::child-notify signal is emitted for each child property that has changed on an object. The signal's detail holds the property name.

`mnemonic-activate` (*arg0* `<gboolean>`) ⇒ `<gboolean>` [Signal on `<gtk-widget>`]

`grab-focus` [Signal on `<gtk-widget>`]

`focus` (*arg0* `<gtk-direction-type>`) ⇒ `<gboolean>` [Signal on `<gtk-widget>`]

`move-focus` (*arg0* `<gtk-direction-type>`) [Signal on `<gtk-widget>`]
undocumented

`event` (*arg0* `<gdk-event>`) ⇒ `<gboolean>` [Signal on `<gtk-widget>`]

`event-after` (*arg0* `<gdk-event>`) [Signal on `<gtk-widget>`]

`button-press-event` (*arg0* `<gdk-event>`) [Signal on `<gtk-widget>`]
⇒ `<gboolean>`

`button-release-event` (*arg0* `<gdk-event>`) [Signal on `<gtk-widget>`]
⇒ `<gboolean>`

`scroll-event` (*arg0* `<gdk-event>`) ⇒ `<gboolean>` [Signal on `<gtk-widget>`]

`motion-notify-event` (*arg0* `<gdk-event>`) [Signal on `<gtk-widget>`]
⇒ `<gboolean>`

`keynav-failed` (*arg0* `<gtk-direction-type>`) [Signal on `<gtk-widget>`]
⇒ `<gboolean>`
undocumented

`delete-event` (*arg0* `<gdk-event>`) ⇒ `<gboolean>` [Signal on `<gtk-widget>`]
The ::delete-event signal is emitted if a user requests that a toplevel window is closed. The default handler for this signal destroys the window. Connecting `gtk-widget-hide-on-delete` to this signal will cause the window to be hidden instead, so that it can later be shown again without reconstructing it.

`destroy-event` (*arg0* `<gdk-event>`) ⇒ `<gboolean>` [Signal on `<gtk-widget>`]
The ::destroy-event signal is emitted when a `<gdk-window>` is destroyed. You rarely get this signal, because most widgets disconnect themselves from their window before they destroy it, so no widget owns the window at destroy time.

`expose-event` (*arg0* `<gdk-event>`) ⇒ `<gboolean>` [Signal on `<gtk-widget>`]

`key-press-event` (*arg0* `<gdk-event>`) ⇒ `<gboolean>` [Signal on `<gtk-widget>`]

`key-release-event` (*arg0* `<gdk-event>`) ⇒ `<gboolean>` [Signal on `<gtk-widget>`]

`enter-notify-event` (*arg0* `<gdk-event>`) ⇒ `<gboolean>` [Signal on `<gtk-widget>`]

`leave-notify-event` (*arg0* `<gdk-event>`) ⇒ `<gboolean>` [Signal on `<gtk-widget>`]

`configure-event` (*arg0* `<gdk-event>`) ⇒ `<gboolean>` [Signal on `<gtk-widget>`]

`focus-in-event` (*arg0* `<gdk-event>`) ⇒ `<gboolean>` [Signal on `<gtk-widget>`]

`focus-out-event` (*arg0* `<gdk-event>`) ⇒ `<gboolean>` [Signal on `<gtk-widget>`]

`map-event` (*arg0* `<gdk-event>`) ⇒ `<gboolean>` [Signal on `<gtk-widget>`]

`unmap-event` (*arg0* `<gdk-event>`) ⇒ `<gboolean>` [Signal on `<gtk-widget>`]

`property-notify-event` (*arg0* `<gdk-event>`) ⇒ `<gboolean>` [Signal on `<gtk-widget>`]

`selection-clear-event` (*arg0* `<gdk-event>`) ⇒ `<gboolean>` [Signal on `<gtk-widget>`]

`selection-request-event` (*arg0* `<gdk-event>`) ⇒ `<gboolean>` [Signal on `<gtk-widget>`]

`selection-notify-event` (*arg0* `<gdk-event>`) ⇒ `<gboolean>` [Signal on `<gtk-widget>`]

`selection-received` (*arg0* `<gtk-selection-data>`) (*arg1* `<guint>`) [Signal on `<gtk-widget>`]

`selection-get` (*arg0* `<gtk-selection-data>`) (*arg1* `<guint>`) (*arg2* `<guint>`) [Signal on `<gtk-widget>`]

`proximity-in-event` (*arg0* `<gdk-event>`) ⇒ `<gboolean>` [Signal on `<gtk-widget>`]

`proximity-out-event` (*arg0* `<gdk-event>`) ⇒ `<gboolean>` [Signal on `<gtk-widget>`]

`drag-leave` (*arg0* `<gdk-drag-context>`) (*arg1* `<guint>`) [Signal on `<gtk-widget>`]

The ::drag-leave signal is emitted on the drop site when the cursor leaves the widget. A typical reason to connect to this signal is to undo things done in ::drag-motion, e.g. undo highlighting with `gtk-drag-unhighlight`

`drag-begin` (*arg0* `<gdk-drag-context>`) [Signal on `<gtk-widget>`]

The ::drag-begin signal is emitted on the drag source when a drag is started. A typical reason to connect to this signal is to set up a custom drag icon with `gtk-drag-source-set-icon`.

`drag-end` (*arg0* `<gdk-drag-context>`) [Signal on `<gtk-widget>`]
The ::drag-end signal is emitted on the drag source when a drag is finished. A typical reason to connect to this signal is to undo things done in ::drag-begin.

`drag-data-delete` (*arg0* `<gdk-drag-context>`) [Signal on `<gtk-widget>`]
The ::drag-data-delete signal is emitted on the drag source when a drag with the action '`GDK_ACTION_MOVE`' is successfully completed. The signal handler is responsible for deleting the data that has been dropped. What "delete" means, depends on the context of the drag operation.

`drag-failed` (*arg0* `<gdk-drag-context>`) [Signal on `<gtk-widget>`]
(*arg1* `<gtk-drag-result>`) ⇒ `<gboolean>`
undocumented

`drag-motion` (*arg0* `<gdk-drag-context>`) [Signal on `<gtk-widget>`]
(*arg1* `<gint>`) (*arg2* `<gint>`) (*arg3* `<guint>`) ⇒ `<gboolean>`
The ::drag-motion signal is emitted on the drop site when the user moves the cursor over the widget during a drag. The signal handler must determine whether the cursor position is in a drop zone or not. If it is not in a drop zone, it returns '`#f`' and no further processing is necessary. Otherwise, the handler returns '`#t`'. In this case, the handler is responsible for providing the necessary information for displaying feedback to the user, by calling `gdk-drag-status`. If the decision whether the drop will be accepted or rejected can't be made based solely on the cursor position and the type of the data, the handler may inspect the dragged data by calling `gtk-drag-get-data` and defer the `gdk-drag-status` call to the ::drag-data-received handler.

Note that there is no ::drag-enter signal. The drag receiver has to keep track of whether he has received any ::drag-motion signals since the last ::drag-leave and if not, treat the ::drag-motion signal as an "enter" signal. Upon an "enter", the handler will typically highlight the drop site with `gtk-drag-highlight`.

```
static void
drag_motion (GtkWidget *widget,
         GdkDragContext *context,
             gint x,
             gint y,
             guint time)
{
  GdkAtom target;

  PrivateData *private_data = GET_PRIVATE_DATA (widget);

  if (!private_data->drag_highlight)
   {
     private_data->drag_highlight = 1;
     gtk_drag_highlight (widget);
   }
```

```
  target = gtk_drag_dest_find_target (widget, context, NULL);
  if (target == GDK_NONE)
    gdk_drag_status (context, 0, time);
  else
   {
     private_data->pending_status = context->suggested_action;
     gtk_drag_get_data (widget, context, target, time);
   }

  return TRUE;
}

static void
drag_data_received (GtkWidget        *widget,
                    GdkDragContext   *context,
                    gint              x,
                    gint              y,
                    GtkSelectionData *selection_data,
                    guint             info,
                    guint             time)
{
  PrivateData *private_data = GET_PRIVATE_DATA (widget);

  if (private_data->suggested_action)
   {
     private_data->suggested_action = 0;

    /* We are getting this data due to a request in drag_motion,
     * rather than due to a request in drag_drop, so we are just
     * supposed to call gdk_drag_status(), not actually paste in
     * the data.
     */
     str = gtk_selection_data_get_text (selection_data);
     if (!data_is_acceptable (str))
       gdk_drag_status (context, 0, time);
     else
       gdk_drag_status (context, private_data->suggested_action, time);
   }
  else
   {
     /* accept the drop */
   }
}
```

`drag-drop` (*arg0* `<gdk-drag-context>`) (*arg1* `<gint>`) [Signal on `<gtk-widget>`]
(*arg2* `<gint>`) (*arg3* `<guint>`) ⇒ `<gboolean>`

The ::drag-drop signal is emitted on the drop site when the user drops the data onto the widget. The signal handler must determine whether the cursor position is in a drop zone or not. If it is not in a drop zone, it returns '`#f`' and no further processing is necessary. Otherwise, the handler returns '`#t`'. In this case, the handler must ensure that `gtk-drag-finish` is called to let the source know that the drop is done. The call to `gtk-drag-finish` can be done either directly or in a ::drag-data-received handler which gets triggered by calling `gtk-drop-get-data` to receive the data for one or more of the supported targets.

`drag-data-get` (*arg0* `<gdk-drag-context>`) [Signal on `<gtk-widget>`]
(*arg1* `<gtk-selection-data>`) (*arg2* `<guint>`) (*arg3* `<guint>`)

The ::drag-data-get signal is emitted on the drag source when the drop site requests the data which is dragged. It is the responsibility of the signal handler to fill *data* with the data in the format which is indicated by *info*. See `gtk-selection-data-set` and `gtk-selection-data-set-text`.

`drag-data-received` (*arg0* `<gdk-drag-context>`) [Signal on `<gtk-widget>`]
(*arg1* `<gint>`) (*arg2* `<gint>`) (*arg3* `<gtk-selection-data>`)
(*arg4* `<guint>`) (*arg5* `<guint>`)

The ::drag-data-received signal is emitted on the drop site when the dragged data has been received. If the data was received in order to determine whether the drop will be accepted, the handler is expected to call `gdk-drag-status` and *not* finish the drag. If the data was received in response to a ::drag-drop signal (and this is the last target to be received), the handler for this signal is expected to process the received data and then call `gtk-drag-finish`, setting the *success* parameter depending on whether the data was processed successfully.

The handler may inspect and modify *drag-context->action* before calling `gtk-drag-finish`, e.g. to implement '`GDK_ACTION_ASK`' as shown in the following example:

```
void
drag_data_received (GtkWidget          *widget,
                    GdkDragContext     *drag_context,
                    gint                x,
                    gint                y,
                    GtkSelectionData   *data,
                    guint               info,
                    guint               time)
{
  if ((data->length >= 0) && (data->format == 8))
    {
      if (drag_context->action == GDK_ACTION_ASK)
        {
          GtkWidget *dialog;
          gint response;
```

```
                dialog = gtk_message_dialog_new (NULL,
                                                 GTK_DIALOG_MODAL |
                                                 GTK_DIALOG_DESTROY_WITH_PARENT,
                                                 GTK_MESSAGE_INFO,
                                                 GTK_BUTTONS_YES_NO,
                                                 "Move the data ?\n");
                response = gtk_dialog_run (GTK_DIALOG (dialog));
                gtk_widget_destroy (dialog);

                if (response == GTK_RESPONSE_YES)
                  drag_context->action = GDK_ACTION_MOVE;
                else
                  drag_context->action = GDK_ACTION_COPY;
               }

              gtk_drag_finish (drag_context, TRUE, FALSE, time);
              return;
             }

            gtk_drag_finish (drag_context, FALSE, FALSE, time);
           }
```

`visibility-notify-event` (*arg0* `<gdk-event>`) ⇒ `<gboolean>` [Signal on `<gtk-widget>`]

`client-event` (*arg0* `<gdk-event>`) ⇒ `<gboolean>` [Signal on `<gtk-widget>`]

`no-expose-event` (*arg0* `<gdk-event>`) ⇒ `<gboolean>` [Signal on `<gtk-widget>`]

`window-state-event` (*arg0* `<gdk-event>`) ⇒ `<gboolean>` [Signal on `<gtk-widget>`]

`grab-broken-event` (*arg0* `<gdk-event>`) ⇒ `<gboolean>` [Signal on `<gtk-widget>`]

Emitted when a pointer or keyboard grab on a window belonging to *widget* gets broken.

On X11, this happens when the grab window becomes unviewable (i.e. it or one of its ancestors is unmapped), or if the same application grabs the pointer or keyboard again.

Since 2.8

`query-tooltip` (*arg0* `<gint>`) (*arg1* `<gint>`) (*arg2* `<gboolean>`) (*arg3* `<gtk-tooltip>`) ⇒ `<gboolean>` [Signal on `<gtk-widget>`]

undocumented

`popup-menu` ⇒ `<gboolean>` [Signal on `<gtk-widget>`]

This signal gets emitted whenever a widget should pop up a context-sensitive menu. This usually happens through the standard key binding mechanism; by pressing a certain key while a widget is focused, the user can cause the widget to pop up a menu. For example, the `<gtk-entry>` widget creates a menu with clipboard commands. See *(the missing figure, checklist-popup-menu* for an example of how to use this signal.

`show-help` (*arg0* `<gtk-widget-help-type>`) ⇒ `<gboolean>` [Signal on `<gtk-widget>`]

`accel-closures-changed` [Signal on `<gtk-widget>`]

`screen-changed` (*arg0* `<gdk-screen>`) [Signal on `<gtk-widget>`]

`can-activate-accel` (*arg0* `<guint>`) ⇒ `<gboolean>` [Signal on `<gtk-widget>`]

Determines whether an accelerator that activates the signal identified by *signal-id* can currently be activated. This signal is present to allow applications and derived widgets to override the default `<gtk-widget>` handling for determining whether an accelerator can be activated.

`<gtk-requisition>` [Class]

`<gtk-selection-data>` [Class]

`gtk-widget-destroy` (*self* `<gtk-widget>`) [Function]
`destroy` [Method]

Destroys a widget. Equivalent to `gtk-object-destroy`, except that you don't have to cast the widget to `<gtk-object>`. When a widget is destroyed, it will break any references it holds to other objects. If the widget is inside a container, the widget will be removed from the container. If the widget is a toplevel (derived from `<gtk-window>`), it will be removed from the list of toplevels, and the reference GTK+ holds to it will be removed. Removing a widget from its container or the list of toplevels results in the widget being finalized, unless you've added additional references to the widget with `g-object-ref`.

In most cases, only toplevel widgets (windows) require explicit destruction, because when you destroy a toplevel its children will be destroyed as well.

widget a `<gtk-widget>`

`gtk-widget-unparent` (*self* `<gtk-widget>`) [Function]
`unparent` [Method]

This function is only for use in widget implementations. Should be called by implementations of the remove method on `<gtk-container>`, to dissociate a child from the container.

widget a `<gtk-widget>`

`gtk-widget-show` (*self* `<gtk-widget>`) [Function]
`show` [Method]

Flags a widget to be displayed. Any widget that isn't shown will not appear on the screen. If you want to show all the widgets in a container, it's easier to call `gtk-widget-show-all` on the container, instead of individually showing the widgets.

Remember that you have to show the containers containing a widget, in addition to the widget itself, before it will appear onscreen.

When a toplevel container is shown, it is immediately realized and mapped; other shown widgets are realized and mapped when their toplevel container is realized and mapped.

widget a `<gtk-widget>`

`gtk-widget-show-now` (*self* `<gtk-widget>`) [Function]
`show-now` [Method]

Shows a widget. If the widget is an unmapped toplevel widget (i.e. a `<gtk-window>` that has not yet been shown), enter the main loop and wait for the window to actually be mapped. Be careful; because the main loop is running, anything can happen during this function.

widget a `<gtk-widget>`

`gtk-widget-hide` (*self* `<gtk-widget>`) [Function]
`hide` [Method]

Reverses the effects of `gtk-widget-show`, causing the widget to be hidden (invisible to the user).

widget a `<gtk-widget>`

`gtk-widget-show-all` (*self* `<gtk-widget>`) [Function]
`show-all` [Method]

Recursively shows a widget, and any child widgets (if the widget is a container).

widget a `<gtk-widget>`

`gtk-widget-hide-all` (*self* `<gtk-widget>`) [Function]
`hide-all` [Method]

Recursively hides a widget and any child widgets.

widget a `<gtk-widget>`

`gtk-widget-map` (*self* `<gtk-widget>`) [Function]
`map` [Method]

This function is only for use in widget implementations. Causes a widget to be mapped if it isn't already.

widget a `<gtk-widget>`

`gtk-widget-unmap` (*self* `<gtk-widget>`) [Function]
`unmap` [Method]

This function is only for use in widget implementations. Causes a widget to be unmapped if it's currently mapped.

widget a `<gtk-widget>`

`gtk-widget-realize` (*self* `<gtk-widget>`) [Function]
`realize` [Method]

Creates the GDK (windowing system) resources associated with a widget. For example, *widget->window* will be created when a widget is realized. Normally realization happens implicitly; if you show a widget and all its parent containers, then the widget will be realized and mapped automatically.

Realizing a widget requires all the widget's parent widgets to be realized; calling `gtk-widget-realize` realizes the widget's parents in addition to *widget* itself. If a widget is not yet inside a toplevel window when you realize it, bad things will happen.

This function is primarily used in widget implementations, and isn't very useful otherwise. Many times when you think you might need it, a better approach is to connect to a signal that will be called after the widget is realized automatically, such as "expose_event". Or simply `g-signal-connect-after` to the "realize" signal.

widget a `<gtk-widget>`

`gtk-widget-unrealize` (*self* `<gtk-widget>`) [Function]
`unrealize` [Method]

This function is only useful in widget implementations. Causes a widget to be unrealized (frees all GDK resources associated with the widget, such as *widget->window*).

widget a `<gtk-widget>`

`gtk-widget-queue-draw` (*self* `<gtk-widget>`) [Function]
`queue-draw` [Method]

Equivalent to calling `gtk-widget-queue-draw-area` for the entire area of a widget.

widget a `<gtk-widget>`

`gtk-widget-queue-resize` (*self* `<gtk-widget>`) [Function]
`queue-resize` [Method]

This function is only for use in widget implementations. Flags a widget to have its size renegotiated; should be called when a widget for some reason has a new size request. For example, when you change the text in a `<gtk-label>`, `<gtk-label>` queues a resize to ensure there's enough space for the new text.

widget a `<gtk-widget>`

`gtk-widget-queue-resize-no-redraw` (*self* `<gtk-widget>`) [Function]
`queue-resize-no-redraw` [Method]

This function works like `gtk-widget-queue-resize`, except that the widget is not invalidated.

widget a `<gtk-widget>`

Since 2.4

`gtk-widget-size-request` (*self* `<gtk-widget>`) [Function]
(*requisition* `<gtk-requisition>`)
`size-request` [Method]

This function is typically used when implementing a `<gtk-container>` subclass. Obtains the preferred size of a widget. The container uses this information to arrange its child widgets and decide what size allocations to give them with `gtk-widget-size-allocate`.

You can also call this function from an application, with some caveats. Most notably, getting a size request requires the widget to be associated with a screen, because font information may be needed. Multihead-aware applications should keep this in mind.

Also remember that the size request is not necessarily the size a widget will actually be allocated.

See also `gtk-widget-get-child-requisition`.

widget a `<gtk-widget>`

requisition a `<gtk-requisition>` to be filled in

`gtk-widget-get-child-requisition` (*self* `<gtk-widget>`) [Function]
(*requisition* `<gtk-requisition>`)

`get-child-requisition` [Method]

This function is only for use in widget implementations. Obtains *widget*->*requisition*, unless someone has forced a particular geometry on the widget (e.g. with `gtk-widget-set-usize`), in which case it returns that geometry instead of the widget's requisition.

This function differs from `gtk-widget-size-request` in that it retrieves the last size request value from *widget*->*requisition*, while `gtk-widget-size-request` actually calls the "size_request" method on *widget* to compute the size request and fill in *widget*->*requisition*, and only then returns *widget*->*requisition*.

Because this function does not call the "size_request" method, it can only be used when you know that *widget*->*requisition* is up-to-date, that is, `gtk-widget-size-request` has been called since the last time a resize was queued. In general, only container implementations have this information; applications should use `gtk-widget-size-request`.

widget a `<gtk-widget>`

requisition a `<gtk-requisition>` to be filled in

`gtk-widget-size-allocate` (*self* `<gtk-widget>`) [Function]
(*allocation* `<gtk-allocation*>`)

`size-allocate` [Method]

This function is only used by `<gtk-container>` subclasses, to assign a size and position to their child widgets.

widget a `<gtk-widget>`

allocation position and size to be allocated to *widget*

`gtk-widget-add-accelerator` (*self* `<gtk-widget>`) [Function]
(*accel_signal* `mchars`) (*accel_group* `<gtk-accel-group>`)
(*accel_key* `unsigned-int`) (*accel_mods* `<gdk-modifier-type>`)
(*accel_flags* `<gtk-accel-flags>`)

`add-accelerator` [Method]

Installs an accelerator for this *widget* in *accel-group* that causes *accel-signal* to be emitted if the accelerator is activated. The *accel-group* needs to be added to the widget's toplevel via `gtk-window-add-accel-group`, and the signal must be of type '`G_RUN_ACTION`'. Accelerators added through this function are not user changeable during runtime. If you want to support accelerators that can be changed by the user, use `gtk-accel-map-add-entry` and `gtk-widget-set-accel-path` or `gtk-menu-item-set-accel-path` instead.

widget widget to install an accelerator on

accel-signal
widget signal to emit on accelerator activation

accel-group
: accel group for this widget, added to its toplevel

accel-key
: GDK keyval of the accelerator

accel-mods
: modifier key combination of the accelerator

accel-flags
: flag accelerators, e.g. '`GTK_ACCEL_VISIBLE`'

`gtk-widget-remove-accelerator` (*self* `<gtk-widget>`) [Function]
(*accel_group* `<gtk-accel-group>`) (*accel_key* `unsigned-int`)
(*accel_mods* `<gdk-modifier-type>`) ⇒ (*ret* `bool`)

`remove-accelerator` [Method]

Removes an accelerator from *widget*, previously installed with `gtk-widget-add-accelerator`.

widget
: widget to install an accelerator on

accel-group
: accel group for this widget

accel-key
: GDK keyval of the accelerator

accel-mods
: modifier key combination of the accelerator

ret
: whether an accelerator was installed and could be removed

`gtk-widget-set-accel-path` (*self* `<gtk-widget>`) [Function]
(*accel_path* `mchars`) (*accel_group* `<gtk-accel-group>`)

`set-accel-path` [Method]

Given an accelerator group, *accel-group*, and an accelerator path, *accel-path*, sets up an accelerator in *accel-group* so whenever the key binding that is defined for *accel-path* is pressed, *widget* will be activated. This removes any accelerators (for any accelerator group) installed by previous calls to `gtk-widget-set-accel-path`. Associating accelerators with paths allows them to be modified by the user and the modifications to be saved for future use. (See `gtk-accel-map-save`.)

This function is a low level function that would most likely be used by a menu creation system like `<gtk-item-factory>`. If you use `<gtk-item-factory>`, setting up accelerator paths will be done automatically.

Even when you you aren't using `<gtk-item-factory>`, if you only want to set up accelerators on menu items `gtk-menu-item-set-accel-path` provides a somewhat more convenient interface.

widget
: a `<gtk-widget>`

accel-path
: path used to look up the accelerator

accel-group
: a `<gtk-accel-group>`.

`gtk-widget-list-accel-closures` (*self* `<gtk-widget>`) [Function]
⇒ (*ret* `glist-of`)

`list-accel-closures` [Method]

Lists the closures used by *widget* for accelerator group connections with `gtk-accel-group-connect-by-path` or `gtk-accel-group-connect`. The closures can be used to monitor accelerator changes on *widget*, by connecting to the ::accel_changed signal of the `<gtk-accel-group>` of a closure which can be found out with `gtk-accel-group-from-accel-closure`.

widget widget to list accelerator closures for

ret a newly allocated `<g-list>` of closures

`gtk-widget-can-activate-accel` (*self* `<gtk-widget>`) [Function]
(*signal_id* `unsigned-int`) ⇒ (*ret* `bool`)

`can-activate-accel` [Method]

Determines whether an accelerator that activates the signal identified by *signal-id* can currently be activated. This is done by emitting the GtkWidget::can-activate-accel signal on *widget*; if the signal isn't overridden by a handler or in a derived widget, then the default check is that the widget must be sensitive, and the widget and all its ancestors mapped.

widget a `<gtk-widget>`

signal-id the ID of a signal installed on *widget*

ret '`#t`' if the accelerator can be activated.

Since 2.4

`gtk-widget-event` (*self* `<gtk-widget>`) (*event* `<gdk-event>`) [Function]
⇒ (*ret* `bool`)

`event` [Method]

Rarely-used function. This function is used to emit the event signals on a widget (those signals should never be emitted without using this function to do so). If you want to synthesize an event though, don't use this function; instead, use `gtk-main-do-event` so the event will behave as if it were in the event queue. Don't synthesize expose events; instead, use `gdk-window-invalidate-rect` to invalidate a region of the window.

widget a `<gtk-widget>`

event a `<gdk-event>`

ret return from the event signal emission ('`#t`' if the event was handled)

`gtk-widget-activate` (*self* `<gtk-widget>`) ⇒ (*ret* `bool`) [Function]

`activate` [Method]

For widgets that can be "activated" (buttons, menu items, etc.) this function activates them. Activation is what happens when you press Enter on a widget during key navigation. If *widget* isn't activatable, the function returns '`#f`'.

widget a `<gtk-widget>` that's activatable

ret '`#t`' if the widget was activatable

`gtk-widget-reparent` (*self* `<gtk-widget>`) [Function]
(*new_parent* `<gtk-widget>`)
`reparent` [Method]

Moves a widget from one `<gtk-container>` to another, handling reference count issues to avoid destroying the widget.

widget a `<gtk-widget>`

new-parent
a `<gtk-container>` to move the widget into

`gtk-widget-intersect` (*self* `<gtk-widget>`) (*area* `<gdk-rectangle>`) [Function]
(*intersection* `<gdk-rectangle>`) ⇒ (*ret* `bool`)
`intersect` [Method]

Computes the intersection of a *widget*'s area and *area*, storing the intersection in *intersection*, and returns '`#t`' if there was an intersection. *intersection* may be '`#f`' if you're only interested in whether there was an intersection.

widget a `<gtk-widget>`

area a rectangle

intersection
rectangle to store intersection of *widget* and *area*

ret '`#t`' if there was an intersection

`gtk-widget-is-focus` (*self* `<gtk-widget>`) ⇒ (*ret* `bool`) [Function]
`is-focus` [Method]

Determines if the widget is the focus widget within its toplevel. (This does not mean that the '`HAS_FOCUS`' flag is necessarily set; '`HAS_FOCUS`' will only be set if the toplevel widget additionally has the global input focus.)

widget a `<gtk-widget>`

ret '`#t`' if the widget is the focus widget.

`gtk-widget-grab-focus` (*self* `<gtk-widget>`) [Function]
`grab-focus` [Method]

Causes *widget* to have the keyboard focus for the `<gtk-window>` it's inside. *widget* must be a focusable widget, such as a `<gtk-entry>`; something like `<gtk-frame>` won't work. (More precisely, it must have the '`GTK_CAN_FOCUS`' flag set.)

widget a `<gtk-widget>`

`gtk-widget-grab-default` (*self* `<gtk-widget>`) [Function]
`grab-default` [Method]

Causes *widget* to become the default widget. *widget* must have the '`GTK_CAN_DEFAULT`' flag set; typically you have to set this flag yourself by calling '`GTK_WIDGET_SET_FLAGS (widget, GTK_CAN_DEFAULT)`'. The default widget is activated when the user presses Enter in a window. Default widgets must be activatable, that is, `gtk-widget-activate` should affect them.

widget a `<gtk-widget>`

`gtk-widget-set-name` (*self* `<gtk-widget>`) (*name* `mchars`) [Function]
`set-name` [Method]

Widgets can be named, which allows you to refer to them from a gtkrc file. You can apply a style to widgets with a particular name in the gtkrc file. See the documentation for gtkrc files (on the same page as the docs for `<gtk-rc-style>`).

Note that widget names are separated by periods in paths (see `gtk-widget-path`), so names with embedded periods may cause confusion.

widget a `<gtk-widget>`

name name for the widget

`gtk-widget-get-name` (*self* `<gtk-widget>`) ⇒ (*ret* `mchars`) [Function]
`get-name` [Method]

Retrieves the name of a widget. See `gtk-widget-set-name` for the significance of widget names.

widget a `<gtk-widget>`

ret name of the widget. This string is owned by GTK+ and should not be modified or freed

`gtk-widget-set-state` (*self* `<gtk-widget>`) [Function]
(*state* `<gtk-state-type>`)
`set-state` [Method]

This function is for use in widget implementations. Sets the state of a widget (insensitive, prelighted, etc.) Usually you should set the state using wrapper functions such as `gtk-widget-set-sensitive`.

widget a `<gtk-widget>`

state new state for *widget*

`gtk-widget-set-sensitive` (*self* `<gtk-widget>`) (*sensitive* `bool`) [Function]
`set-sensitive` [Method]

Sets the sensitivity of a widget. A widget is sensitive if the user can interact with it. Insensitive widgets are "grayed out" and the user can't interact with them. Insensitive widgets are known as "inactive", "disabled", or "ghosted" in some other toolkits.

widget a `<gtk-widget>`

sensitive '`#t`' to make the widget sensitive

`gtk-widget-set-parent` (*self* `<gtk-widget>`) (*parent* `<gtk-widget>`) [Function]
`set-parent` [Method]

This function is useful only when implementing subclasses of `<gtk-container>`. Sets the container as the parent of *widget*, and takes care of some details such as updating the state and style of the child to reflect its new location. The opposite function is `gtk-widget-unparent`.

widget a `<gtk-widget>`

parent parent container

`gtk-widget-set-parent-window` (*self* `<gtk-widget>`) [Function]
(*parent_window* `<gdk-window*>`)
`set-parent-window` [Method]

Sets a non default parent window for *widget*.

widget a `<gtk-widget>`.

parent-window
the new parent window.

`gtk-widget-get-parent-window` (*self* `<gtk-widget>`) [Function]
⇒ (*ret* `<gdk-window*>`)
`get-parent-window` [Method]

Gets *widget*'s parent window.

widget a `<gtk-widget>`.

ret the parent window of *widget*.

`gtk-widget-set-events` (*self* `<gtk-widget>`) [Function]
(*events* `<gdk-event-mask>`)
`set-events` [Method]

Sets the event mask (see `<gdk-event-mask>`) for a widget. The event mask determines which events a widget will receive. Keep in mind that different widgets have different default event masks, and by changing the event mask you may disrupt a widget's functionality, so be careful. This function must be called while a widget is unrealized. Consider `gtk-widget-add-events` for widgets that are already realized, or if you want to preserve the existing event mask. This function can't be used with `<gtk-no-window>` widgets; to get events on those widgets, place them inside a `<gtk-event-box>` and receive events on the event box.

widget a `<gtk-widget>`

events event mask

`gtk-widget-add-events` (*self* `<gtk-widget>`) [Function]
(*events* `<gdk-event-mask>`)
`add-events` [Method]

Adds the events in the bitfield *events* to the event mask for *widget*. See `gtk-widget-set-events` for details.

widget a `<gtk-widget>`

events an event mask, see `<gdk-event-mask>`

`gtk-widget-set-extension-events` (*self* `<gtk-widget>`) [Function]
(*mode* `<gdk-extension-mode>`)
`set-extension-events` [Method]

Sets the extension events mask to *mode*. See `<gdk-extension-mode>` and `gdk-input-set-extension-events`.

widget a `<gtk-widget>`

mode bitfield of extension events to receive

`gtk-widget-get-extension-events` (*self* `<gtk-widget>`) [Function]
⇒ (*ret* `<gdk-extension-mode>`)

`get-extension-events` [Method]

Retrieves the extension events the widget will receive; see `gdk-input-set-extension-events`.

widget a `<gtk-widget>`

ret extension events for *widget*

`gtk-widget-get-toplevel` (*self* `<gtk-widget>`) [Function]
⇒ (*ret* `<gtk-widget>`)

`get-toplevel` [Method]

This function returns the topmost widget in the container hierarchy *widget* is a part of. If *widget* has no parent widgets, it will be returned as the topmost widget. No reference will be added to the returned widget; it should not be unreferenced.

Note the difference in behavior vs. `gtk-widget-get-ancestor`; '`gtk_widget_get_ancestor (widget, GTK_TYPE_WINDOW)`' would return '`#f`' if *widget* wasn't inside a toplevel window, and if the window was inside a `<gtk-window-derived>` widget which was in turn inside the toplevel `<gtk-window>`. While the second case may seem unlikely, it actually happens when a `<gtk-plug>` is embedded inside a `<gtk-socket>` within the same application.

To reliably find the toplevel `<gtk-window>`, use `gtk-widget-get-toplevel` and check if the '`TOPLEVEL`' flags is set on the result.

```
GtkWidget *toplevel = gtk_widget_get_toplevel (widget);
if (GTK_WIDGET_TOPLEVEL (toplevel))
  {
    [ Perform action on toplevel. ]
  }
```

widget a `<gtk-widget>`

ret the topmost ancestor of *widget*, or *widget* itself if there's no ancestor.

`gtk-widget-get-ancestor` (*self* `<gtk-widget>`) [Function]
(*widget_type* `<gtype>`) ⇒ (*ret* `<gtk-widget>`)

`get-ancestor` [Method]

Gets the first ancestor of *widget* with type *widget-type*. For example, '`gtk_widget_get_ancestor (widget, GTK_TYPE_BOX)`' gets the first `<gtk-box>` that's an ancestor of *widget*. No reference will be added to the returned widget; it should not be unreferenced. See note about checking for a toplevel `<gtk-window>` in the docs for `gtk-widget-get-toplevel`.

Note that unlike `gtk-widget-is-ancestor`, `gtk-widget-get-ancestor` considers *widget* to be an ancestor of itself.

widget a `<gtk-widget>`

widget-type
ancestor type

ret the ancestor widget, or '`#f`' if not found

`gtk-widget-get-colormap` (*self* `<gtk-widget>`) [Function]
⇒ (*ret* `<gdk-colormap>`)
`get-colormap` [Method]
Gets the colormap that will be used to render *widget*. No reference will be added to the returned colormap; it should not be unreferenced.

widget a `<gtk-widget>`

ret the colormap used by *widget*

`gtk-widget-set-colormap` (*self* `<gtk-widget>`) [Function]
(*colormap* `<gdk-colormap>`)
`set-colormap` [Method]
Sets the colormap for the widget to the given value. Widget must not have been previously realized. This probably should only be used from an `init` function (i.e. from the constructor for the widget).

widget a `<gtk-widget>`

colormap a colormap

`gtk-widget-get-visual` (*self* `<gtk-widget>`) [Function]
⇒ (*ret* `<gdk-visual*>`)
`get-visual` [Method]
Gets the visual that will be used to render *widget*.

widget a `<gtk-widget>`

ret the visual for *widget*

`gtk-widget-get-events` (*self* `<gtk-widget>`) ⇒ (*ret* `int`) [Function]
`get-events` [Method]
Returns the event mask for the widget (a bitfield containing flags from the `<gdk-event-mask>` enumeration). These are the events that the widget will receive.

widget a `<gtk-widget>`

ret event mask for *widget*

`gtk-widget-get-pointer` (*self* `<gtk-widget>`) ⇒ (*x* `int`) (*y* `int`) [Function]
`get-pointer` [Method]
Obtains the location of the mouse pointer in widget coordinates. Widget coordinates are a bit odd; for historical reasons, they are defined as *widget->window* coordinates for widgets that are not `<gtk-no-window>` widgets, and are relative to *widget->allocation.x*, *widget->allocation.y* for widgets that are `<gtk-no-window>` widgets.

widget a `<gtk-widget>`

x return location for the X coordinate, or '`#f`'

y return location for the Y coordinate, or '`#f`'

`gtk-widget-is-ancestor` (*self* `<gtk-widget>`) [Function]
(*ancestor* `<gtk-widget>`) ⇒ (*ret* `bool`)
`is-ancestor` [Method]

Determines whether *widget* is somewhere inside *ancestor*, possibly with intermediate containers.

widget a `<gtk-widget>`

ancestor another `<gtk-widget>`

ret '`#t`' if *ancestor* contains *widget* as a child, grandchild, great grandchild, etc.

`gtk-widget-translate-coordinates` (*self* `<gtk-widget>`) [Function]
(*dest_widget* `<gtk-widget>`) (*src_x* `int`) (*src_y* `int`) ⇒ (*ret* `bool`)
(*dest_x* `int`) (*dest_y* `int`)
`translate-coordinates` [Method]

Translate coordinates relative to *src-widget*'s allocation to coordinates relative to *dest-widget*'s allocations. In order to perform this operation, both widgets must be realized, and must share a common toplevel.

src-widget a `<gtk-widget>`

dest-widget
a `<gtk-widget>`

src-x X position relative to *src-widget*

src-y Y position relative to *src-widget*

dest-x location to store X position relative to *dest-widget*

dest-y location to store Y position relative to *dest-widget*

ret '`#f`' if either widget was not realized, or there was no common ancestor. In this case, nothing is stored in **dest-x* and **dest-y*. Otherwise '`#t`'.

`gtk-widget-hide-on-delete` (*self* `<gtk-widget>`) ⇒ (*ret* `bool`) [Function]
`hide-on-delete` [Method]

Utility function; intended to be connected to the "delete_event" signal on a `<gtk-window>`. The function calls `gtk-widget-hide` on its argument, then returns '`#t`'. If connected to "delete_event", the result is that clicking the close button for a window (on the window frame, top right corner usually) will hide but not destroy the window. By default, GTK+ destroys windows when "delete_event" is received.

widget a `<gtk-widget>`

ret '`#t`'

`gtk-widget-set-style` (*self* `<gtk-widget>`) (*style* `<gtk-style>`) [Function]
`set-style` [Method]

Sets the `<gtk-style>` for a widget (*widget->style*). You probably don't want to use this function; it interacts badly with themes, because themes work by replacing the `<gtk-style>`. Instead, use `gtk-widget-modify-style`.

widget a `<gtk-widget>`

style a `<gtk-style>`, or '`#f`' to remove the effect of a previous `gtk-widget-set-style` and go back to the default style

`gtk-widget-ensure-style` (*self* `<gtk-widget>`) [Function]
`ensure-style` [Method]

Ensures that *widget* has a style (*widget->style*). Not a very useful function; most of the time, if you want the style, the widget is realized, and realized widgets are guaranteed to have a style already.

widget a `<gtk-widget>`

`gtk-widget-get-style` (*self* `<gtk-widget>`) ⇒ (*ret* `<gtk-style>`) [Function]
`get-style` [Method]

Simply an accessor function that returns *widget->style*.

widget a `<gtk-widget>`

ret the widget's `<gtk-style>`

`gtk-widget-reset-rc-styles` (*self* `<gtk-widget>`) [Function]
`reset-rc-styles` [Method]

Reset the styles of *widget* and all descendents, so when they are looked up again, they get the correct values for the currently loaded RC file settings.

This function is not useful for applications.

widget a `<gtk-widget>`.

`gtk-widget-push-colormap` (*cmap* `<gdk-colormap>`) [Function]

Pushes *cmap* onto a global stack of colormaps; the topmost colormap on the stack will be used to create all widgets. Remove *cmap* with `gtk-widget-pop-colormap`. There's little reason to use this function.

cmap a `<gdk-colormap>`

`gtk-widget-pop-colormap` [Function]

Removes a colormap pushed with `gtk-widget-push-colormap`.

`gtk-widget-set-default-colormap` (*colormap* `<gdk-colormap>`) [Function]

Sets the default colormap to use when creating widgets. `gtk-widget-push-colormap` is a better function to use if you only want to affect a few widgets, rather than all widgets.

colormap a `<gdk-colormap>`

`gtk-widget-get-default-style` ⇒ (*ret* `<gtk-style>`) [Function]

Returns the default style used by all widgets initially.

ret the default style. This `<gtk-style>` object is owned by GTK+ and should not be modified or freed.

`gtk-widget-get-default-colormap` ⇒ (*ret* `<gdk-colormap>`) [Function]

Obtains the default colormap used to create widgets.

ret default widget colormap

`gtk-widget-get-default-visual` ⇒ (*ret* `<gdk-visual*>`) [Function]

Obtains the visual of the default colormap. Not really useful; used to be useful before `gdk-colormap-get-visual` existed.

ret visual of the default colormap

`gtk-widget-set-direction` (*self* `<gtk-widget>`) [Function]
(*dir* `<gtk-text-direction>`)
`set-direction` [Method]

Sets the reading direction on a particular widget. This direction controls the primary direction for widgets containing text, and also the direction in which the children of a container are packed. The ability to set the direction is present in order so that correct localization into languages with right-to-left reading directions can be done. Generally, applications will let the default reading direction present, except for containers where the containers are arranged in an order that is explicitely visual rather than logical (such as buttons for text justification).

If the direction is set to '`GTK_TEXT_DIR_NONE`', then the value set by `gtk-widget-set-default-direction` will be used.

widget a `<gtk-widget>`

dir the new direction

`gtk-widget-get-direction` (*self* `<gtk-widget>`) [Function]
⇒ (*ret* `<gtk-text-direction>`)
`get-direction` [Method]

Gets the reading direction for a particular widget. See `gtk-widget-set-direction`.

widget a `<gtk-widget>`

ret the reading direction for the widget.

`gtk-widget-set-default-direction` (*dir* `<gtk-text-direction>`) [Function]

Sets the default reading direction for widgets where the direction has not been explicitly set by `gtk-widget-set-direction`.

dir the new default direction. This cannot be '`GTK_TEXT_DIR_NONE`'.

`gtk-widget-get-default-direction` [Function]
⇒ (*ret* `<gtk-text-direction>`)

Obtains the current default reading direction. See `gtk-widget-set-default-direction`.

ret the current default direction.

`gtk-widget-shape-combine-mask` (*self* `<gtk-widget>`) [Function]
(*shape_mask* `<gdk-bitmap*>`) (*offset_x* `int`) (*offset_y* `int`)
`shape-combine-mask` [Method]

Sets a shape for this widget's GDK window. This allows for transparent windows etc., see `gdk-window-shape-combine-mask` for more information.

widget a `<gtk-widget>`.

shape-mask
shape to be added, or '`#f`' to remove an existing shape.

offset-x X position of shape mask with respect to *window*.

offset-y Y position of shape mask with respect to *window*.

`gtk-widget-input-shape-combine-mask` (*self* `<gtk-widget>`) [Function]
(*shape_mask* `<gdk-bitmap*>`) (*offset_x* `int`) (*offset_y* `int`)

`input-shape-combine-mask` [Method]
Sets an input shape for this widget's GDK window. This allows for windows which react to mouse click in a nonrectangular region, see `gdk-window-input-shape-combine-mask` for more information.

widget a `<gtk-widget>`.

shape-mask
shape to be added, or '`#f`' to remove an existing shape.

offset-x X position of shape mask with respect to *window*.

offset-y Y position of shape mask with respect to *window*.

Since 2.10

`gtk-widget-path` (*self* `<gtk-widget>`) [Function]
⇒ (*path_length* `unsigned-int`) (*path* `mchars`) (*path_reversed* `mchars`)

`path` [Method]
Obtains the full path to *widget*. The path is simply the name of a widget and all its parents in the container hierarchy, separated by periods. The name of a widget comes from `gtk-widget-get-name`. Paths are used to apply styles to a widget in gtkrc configuration files. Widget names are the type of the widget by default (e.g. `"GtkButton"`) or can be set to an application-specific value with `gtk-widget-set-name`. By setting the name of a widget, you allow users or theme authors to apply styles to that specific widget in their gtkrc file. *path-reversed-p* fills in the path in reverse order, i.e. starting with *widget*'s name instead of starting with the name of *widget*'s outermost ancestor.

widget a `<gtk-widget>`

path-length
location to store length of the path, or '`#f`'

path location to store allocated path string, or '`#f`'

path-reversed
location to store allocated reverse path string, or '`#f`'

`gtk-widget-class-path` (*self* `<gtk-widget>`) [Function]
⇒ (*path_length* `unsigned-int`) (*path* `mchars`) (*path_reversed* `mchars`)

`class-path` [Method]
Same as `gtk-widget-path`, but always uses the name of a widget's type, never uses a custom name set with `gtk-widget-set-name`.

widget a `<gtk-widget>`

path-length
location to store the length of the class path, or '`#f`'

path location to store the class path as an allocated string, or '`#f`'

path-reversed
location to store the reverse class path as an allocated string, or '`#f`'

`gtk-widget-get-composite-name` (*self* `<gtk-widget>`) [Function]
⇒ (*ret* `mchars`)
`get-composite-name` [Method]
Obtains the composite name of a widget.

widget a `<gtk-widget>`.

ret the composite name of *widget*, or '`#f`' if *widget* is not a composite child. The string should not be freed when it is no longer needed.

`gtk-widget-modify-style` (*self* `<gtk-widget>`) [Function]
(*style* `<gtk-rc-style>`)
`modify-style` [Method]
Modifies style values on the widget. Modifications made using this technique take precedence over style values set via an RC file, however, they will be overriden if a style is explicitely set on the widget using `gtk-widget-set-style`. The `<gtk-rc-style>` structure is designed so each field can either be set or unset, so it is possible, using this function, to modify some style values and leave the others unchanged.

Note that modifications made with this function are not cumulative with previous calls to `gtk-widget-modify-style` or with such functions as `gtk-widget-modify-fg`. If you wish to retain previous values, you must first call `gtk-widget-get-modifier-style`, make your modifications to the returned style, then call `gtk-widget-modify-style` with that style. On the other hand, if you first call `gtk-widget-modify-style`, subsequent calls to such functions `gtk-widget-modify-fg` will have a cumulative effect with the initial modifications.

widget a `<gtk-widget>`

style the `<gtk-rc-style>` holding the style modifications

`gtk-widget-get-modifier-style` (*self* `<gtk-widget>`) [Function]
⇒ (*ret* `<gtk-rc-style>`)
`get-modifier-style` [Method]
Returns the current modifier style for the widget. (As set by `gtk-widget-modify-style`.) If no style has previously set, a new `<gtk-rc-style>` will be created with all values unset, and set as the modifier style for the widget. If you make changes to this rc style, you must call `gtk-widget-modify-style`, passing in the returned rc style, to make sure that your changes take effect.

Caution: passing the style back to `gtk-widget-modify-style` will normally end up destroying it, because `gtk-widget-modify-style` copies the passed-in style and sets the copy as the new modifier style, thus dropping any reference to the old modifier style. Add a reference to the modifier style if you want to keep it alive.

widget a `<gtk-widget>`

ret the modifier style for the widget. This rc style is owned by the widget. If you want to keep a pointer to value this around, you must add a refcount using `g-object-ref`.

`gtk-widget-modify-fg` (*self* `<gtk-widget>`) [Function]
(*state* `<gtk-state-type>`) (*color* `<gdk-color>`)
`modify-fg` [Method]

Sets the foreground color for a widget in a particular state. All other style values are left untouched. See also `gtk-widget-modify-style`.

widget a `<gtk-widget>`.

state the state for which to set the foreground color.

color the color to assign (does not need to be allocated), or '`#f`' to undo the effect of previous calls to of `gtk-widget-modify-fg`.

`gtk-widget-modify-bg` (*self* `<gtk-widget>`) [Function]
(*state* `<gtk-state-type>`) (*color* `<gdk-color>`)
`modify-bg` [Method]

Sets the background color for a widget in a particular state. All other style values are left untouched. See also `gtk-widget-modify-style`.

Note that `"no window"` widgets (which have the '`GTK_NO_WINDOW`' flag set) draw on their parent container's window and thus may not draw any background themselves. This is the case for e.g. `<gtk-label>`. To modify the background of such widgets, you have to set the background color on their parent; if you want to set the background of a rectangular area around a label, try placing the label in a `<gtk-event-box>` widget and setting the background color on that.

widget a `<gtk-widget>`.

state the state for which to set the background color.

color the color to assign (does not need to be allocated), or '`#f`' to undo the effect of previous calls to of `gtk-widget-modify-bg`.

`gtk-widget-modify-text` (*self* `<gtk-widget>`) [Function]
(*state* `<gtk-state-type>`) (*color* `<gdk-color>`)
`modify-text` [Method]

Sets the text color for a widget in a particular state. All other style values are left untouched. The text color is the foreground color used along with the base color (see `gtk-widget-modify-base`) for widgets such as `<gtk-entry>` and `<gtk-text-view>`. See also `gtk-widget-modify-style`.

widget a `<gtk-widget>`.

state the state for which to set the text color.

color the color to assign (does not need to be allocated), or '`#f`' to undo the effect of previous calls to of `gtk-widget-modify-text`.

`gtk-widget-modify-base` (*self* `<gtk-widget>`) [Function]
(*state* `<gtk-state-type>`) (*color* `<gdk-color>`)
`modify-base` [Method]

Sets the base color for a widget in a particular state. All other style values are left untouched. The base color is the background color used along with the text color (see `gtk-widget-modify-text`) for widgets such as `<gtk-entry>` and `<gtk-text-view>`. See also `gtk-widget-modify-style`.

Note that `"no window"` widgets (which have the '`GTK_NO_WINDOW`' flag set) draw on their parent container's window and thus may not draw any background themselves. This is the case for e.g. `<gtk-label>`. To modify the background of such widgets, you have to set the base color on their parent; if you want to set the background of a rectangular area around a label, try placing the label in a `<gtk-event-box>` widget and setting the base color on that.

widget a `<gtk-widget>`.

state the state for which to set the base color.

color the color to assign (does not need to be allocated), or '`#f`' to undo the effect of previous calls to of `gtk-widget-modify-base`.

`gtk-widget-modify-font` (*self* `<gtk-widget>`) [Function]
(*font_desc* `<pango-font-description>`)
`modify-font` [Method]

Sets the font to use for a widget. All other style values are left untouched. See also `gtk-widget-modify-style`.

widget a `<gtk-widget>`

font-desc the font description to use, or '`#f`' to undo the effect of previous calls to `gtk-widget-modify-font`.

`gtk-widget-create-pango-context` (*self* `<gtk-widget>`) [Function]
⇒ (*ret* `<pango-context*>`)
`create-pango-context` [Method]

Creates a new `<pango-context>` with the appropriate font map, font description, and base direction for drawing text for this widget. See also `gtk-widget-get-pango-context`.

widget a `<gtk-widget>`

ret the new `<pango-context>`

`gtk-widget-get-pango-context` (*self* `<gtk-widget>`) [Function]
⇒ (*ret* `<pango-context*>`)
`get-pango-context` [Method]

Gets a `<pango-context>` with the appropriate font map, font description, and base direction for this widget. Unlike the context returned by `gtk-widget-create-pango-context`, this context is owned by the widget (it can be used until the screen for the widget changes or the widget is removed from its toplevel), and will be updated to match any changes to the widget's attributes.

If you create and keep a <pango-layout> using this context, you must deal with changes to the context by calling pango-layout-context-changed on the layout in response to the ::style-set and ::direction-changed signals for the widget.

widget a <gtk-widget>

ret the <pango-context> for the widget.

gtk-widget-create-pango-layout (*self* <gtk-widget>) [Function]
(*text* mchars) ⇒ (*ret* <pango-layout*>)

create-pango-layout [Method]

Creates a new <pango-layout> with the appropriate font map, font description, and base direction for drawing text for this widget.

If you keep a <pango-layout> created in this way around, in order to notify the layout of changes to the base direction or font of this widget, you must call pango-layout-context-changed in response to the ::style-set and ::direction-changed signals for the widget.

widget a <gtk-widget>

text text to set on the layout (can be '#f')

ret the new <pango-layout>

gtk-widget-render-icon (*self* <gtk-widget>) (*stock_id* mchars) [Function]
(*size* <gtk-icon-size>) (*detail* mchars) ⇒ (*ret* <gdk-pixbuf>)

render-icon [Method]

A convenience function that uses the theme engine and RC file settings for *widget* to look up *stock-id* and render it to a pixbuf. *stock-id* should be a stock icon ID such as <gtk-stock-open> or <gtk-stock-ok>. *size* should be a size such as <gtk-icon-size-menu>. *detail* should be a string that identifies the widget or code doing the rendering, so that theme engines can special-case rendering for that widget or code.

The pixels in the returned <gdk-pixbuf> are shared with the rest of the application and should not be modified. The pixbuf should be freed after use with g-object-unref.

widget a <gtk-widget>

stock-id a stock ID

size a stock size. A size of (GtkIconSize)-1 means render at the size of the source and don't scale (if there are multiple source sizes, GTK+ picks one of the available sizes).

detail render detail to pass to theme engine

ret a new pixbuf, or '#f' if the stock ID wasn't known

gtk-widget-pop-composite-child [Function]

Cancels the effect of a previous call to gtk-widget-push-composite-child.

`gtk-widget-push-composite-child` [Function]

Makes all newly-created widgets as composite children until the corresponding `gtk-widget-pop-composite-child` call.

A composite child is a child that's an implementation detail of the container it's inside and should not be visible to people using the container. Composite children aren't treated differently by GTK (but see `gtk-container-foreach` vs. `gtk-container-forall`), but e.g. GUI builders might want to treat them in a different way.

Here is a simple example:

```
gtk_widget_push_composite_child ();
scrolled_window->hscrollbar = gtk_hscrollbar_new (hadjustment);
gtk_widget_set_composite_name (scrolled_window->hscrollbar, "hscrollbar");
gtk_widget_pop_composite_child ();
gtk_widget_set_parent (scrolled_window->hscrollbar,
                       GTK_WIDGET (scrolled_window));
g_object_ref (scrolled_window->hscrollbar);
```

`gtk-widget-queue-draw-area` (*self* `<gtk-widget>`) (*x* `int`) (*y* `int`) (*width* `int`) (*height* `int`) [Function]

`queue-draw-area` [Method]

Invalidates the rectangular area of *widget* defined by *x*, *y*, *width* and *height* by calling `gdk-window-invalidate-rect` on the widget's window and all its child windows. Once the main loop becomes idle (after the current batch of events has been processed, roughly), the window will receive expose events for the union of all regions that have been invalidated.

Normally you would only use this function in widget implementations. You might also use it, or `gdk-window-invalidate-rect` directly, to schedule a redraw of a `<gtk-drawing-area>` or some portion thereof.

Frequently you can just call `gdk-window-invalidate-rect` or `gdk-window-invalidate-region` instead of this function. Those functions will invalidate only a single window, instead of the widget and all its children.

The advantage of adding to the invalidated region compared to simply drawing immediately is efficiency; using an invalid region ensures that you only have to redraw one time.

widget	a `<gtk-widget>`
x	x coordinate of upper-left corner of rectangle to redraw
y	y coordinate of upper-left corner of rectangle to redraw
width	width of region to draw
height	height of region to draw

`gtk-widget-reset-shapes` (*self* `<gtk-widget>`) [Function]

`reset-shapes` [Method]

Recursively resets the shape on this widget and its descendants.

widget	a `<gtk-widget>`.

`gtk-widget-set-app-paintable` (*self* `<gtk-widget>`) [Function]
(*app_paintable* `bool`)
`set-app-paintable` [Method]

Sets whether the application intends to draw on the widget in an ::expose-event handler.

This is a hint to the widget and does not affect the behavior of the GTK+ core; many widgets ignore this flag entirely. For widgets that do pay attention to the flag, such as `<gtk-event-box>` and `<gtk-window>`, the effect is to suppress default themed drawing of the widget's background. (Children of the widget will still be drawn.) The application is then entirely responsible for drawing the widget background.

Note that the background is still drawn when the widget is mapped. If this is not suitable (e.g. because you want to make a transparent window using an RGBA visual), you can work around this by doing:

```
gtk_widget_realize (window);
gdk_window_set_back_pixmap (window->window, NULL, FALSE);
gtk_widget_show (window);
```

widget a `<gtk-widget>`

app-paintable
'`#t`' if the application will paint on the widget

`gtk-widget-set-double-buffered` (*self* `<gtk-widget>`) [Function]
(*double_buffered* `bool`)
`set-double-buffered` [Method]

Widgets are double buffered by default; you can use this function to turn off the buffering. "Double buffered" simply means that `gdk-window-begin-paint-region` and `gdk-window-end-paint` are called automatically around expose events sent to the widget. `gdk-window-begin-paint` diverts all drawing to a widget's window to an offscreen buffer, and `gdk-window-end-paint` draws the buffer to the screen. The result is that users see the window update in one smooth step, and don't see individual graphics primitives being rendered.

In very simple terms, double buffered widgets don't flicker, so you would only use this function to turn off double buffering if you had special needs and really knew what you were doing.

Note: if you turn off double-buffering, you have to handle expose events, since even the clearing to the background color or pixmap will not happen automatically (as it is done in `gdk-window-begin-paint`).

widget a `<gtk-widget>`

double-buffered
'`#t`' to double-buffer a widget

`gtk-widget-set-redraw-on-allocate` (*self* `<gtk-widget>`) [Function]
(*redraw_on_allocate* `bool`)

`set-redraw-on-allocate` [Method]

Sets whether the entire widget is queued for drawing when its size allocation changes. By default, this setting is '`#t`' and the entire widget is redrawn on every size change. If your widget leaves the upper left unchanged when made bigger, turning this setting on will improve performance.

Note that for '`NO_WINDOW`' widgets setting this flag to '`#f`' turns off all allocation on resizing: the widget will not even redraw if its position changes; this is to allow containers that don't draw anything to avoid excess invalidations. If you set this flag on a '`NO_WINDOW`' widget that *does* draw on *widget->window*, you are responsible for invalidating both the old and new allocation of the widget when the widget is moved and responsible for invalidating regions newly when the widget increases size.

widget a `<gtk-widget>`

redraw-on-allocate
if '`#t`', the entire widget will be redrawn when it is allocated to a new size. Otherwise, only the new portion of the widget will be redrawn.

`gtk-widget-set-composite-name` (*self* `<gtk-widget>`) [Function]
(*name* `mchars`)

`set-composite-name` [Method]

Sets a widgets composite name. The widget must be a composite child of its parent; see `gtk-widget-push-composite-child`.

widget a `<gtk-widget>`.

name the name to set.

`gtk-widget-set-scroll-adjustments` (*self* `<gtk-widget>`) [Function]
(*hadjustment* `<gtk-adjustment>`) (*vadjustment* `<gtk-adjustment>`)
⇒ (*ret* `bool`)

`set-scroll-adjustments` [Method]

For widgets that support scrolling, sets the scroll adjustments and returns '`#t`'. For widgets that don't support scrolling, does nothing and returns '`#f`'. Widgets that don't support scrolling can be scrolled by placing them in a `<gtk-viewport>`, which does support scrolling.

widget a `<gtk-widget>`

hadjustment
an adjustment for horizontal scrolling, or '`#f`'

vadjustment
an adjustment for vertical scrolling, or '`#f`'

ret '`#t`' if the widget supports scrolling

`gtk-widget-mnemonic-activate` (*self* `<gtk-widget>`) [Function]
(*group_cycling* `bool`) ⇒ (*ret* `bool`)

`mnemonic-activate` [Method]

widget

group-cycling

ret

`gtk-widget-region-intersect` (*self* `<gtk-widget>`) [Function]
(*region* `<gdk-region*>`) ⇒ (*ret* `<gdk-region*>`)

`region-intersect` [Method]

Computes the intersection of a *widget*'s area and *region*, returning the intersection. The result may be empty, use `gdk-region-empty` to check.

widget a `<gtk-widget>`

region a `<gdk-region>`, in the same coordinate system as *widget->allocation*. That is, relative to *widget->window* for '`NO_WINDOW`' widgets; relative to the parent window of *widget->window* for widgets with their own window.

ret A newly allocated region holding the intersection of *widget* and *region*. The coordinates of the return value are relative to *widget->window* for '`NO_WINDOW`' widgets, and relative to the parent window of *widget->window* for widgets with their own window.

`gtk-widget-send-expose` (*self* `<gtk-widget>`) (*event* `<gdk-event>`) [Function]
⇒ (*ret* `int`)

`send-expose` [Method]

Very rarely-used function. This function is used to emit an expose event signals on a widget. This function is not normally used directly. The only time it is used is when propagating an expose event to a child '`NO_WINDOW`' widget, and that is normally done using `gtk-container-propagate-expose`.

If you want to force an area of a window to be redrawn, use `gdk-window-invalidate-rect` or `gdk-window-invalidate-region`. To cause the redraw to be done immediately, follow that call with a call to `gdk-window-process-updates`.

widget a `<gtk-widget>`

event a expose `<gdk-event>`

ret return from the event signal emission ('`#t`' if the event was handled)

`gtk-widget-style-get-property` (*self* `<gtk-widget>`) [Function]
(*property_name* `mchars`) (*value* `<gvalue>`)

`style-get-property` [Method]

Gets the value of a style property of *widget*.

widget a `<gtk-widget>`

property-name
the name of a style property

value location to return the property value

`gtk-widget-get-accessible` (*self* `<gtk-widget>`) [Function]
⇒ (*ret* `<atk-object>`)

`get-accessible` [Method]

Returns the accessible object that describes the widget to an assistive technology.

If no accessibility library is loaded (i.e. no ATK implementation library is loaded via `GTK_MODULES` or via another application library, such as libgnome), then this

`<atk-object>` instance may be a no-op. Likewise, if no class-specific `<atk-object>` implementation is available for the widget instance in question, it will inherit an `<atk-object>` implementation from the first ancestor class for which such an implementation is defined.

The documentation of the ATK library contains more information about accessible objects and their uses.

widget a `<gtk-widget>`

ret the `<atk-object>` associated with *widget*

`gtk-widget-child-focus` (*self* `<gtk-widget>`) [Function]
(*direction* `<gtk-direction-type>`) ⇒ (*ret* `bool`)

`child-focus` [Method]

This function is used by custom widget implementations; if you're writing an app, you'd use `gtk-widget-grab-focus` to move the focus to a particular widget, and `gtk-container-set-focus-chain` to change the focus tab order. So you may want to investigate those functions instead.

`gtk-widget-child-focus` is called by containers as the user moves around the window using keyboard shortcuts. *direction* indicates what kind of motion is taking place (up, down, left, right, tab forward, tab backward). `gtk-widget-child-focus` invokes the `"focus"` signal on `<gtk-widget>`; widgets override the default handler for this signal in order to implement appropriate focus behavior.

The `"focus"` default handler for a widget should return '`#t`' if moving in *direction* left the focus on a focusable location inside that widget, and '`#f`' if moving in *direction* moved the focus outside the widget. If returning '`#t`', widgets normally call `gtk-widget-grab-focus` to place the focus accordingly; if returning '`#f`', they don't modify the current focus location.

This function replaces `gtk-container-focus` from GTK+ 1.2. It was necessary to check that the child was visible, sensitive, and focusable before calling `gtk-container-focus`. `gtk-widget-child-focus` returns '`#f`' if the widget is not currently in a focusable state, so there's no need for those checks.

widget a `<gtk-widget>`

direction direction of focus movement

ret '`#t`' if focus ended up inside *widget*

`gtk-widget-child-notify` (*self* `<gtk-widget>`) [Function]
(*child_property* `mchars`)

`child-notify` [Method]

Emits a `"child-notify"` signal for the child property*child-property* on *widget*.

This is the analogue of `g-object-notify` for child properties.

widget a `<gtk-widget>`

child-property
the name of a child property installed on the class of *widget*'s parent.

`gtk-widget-freeze-child-notify` (*self* `<gtk-widget>`) [Function]
`freeze-child-notify` [Method]

Stops emission of "child-notify" signals on *widget*. The signals are queued until `gtk-widget-thaw-child-notify` is called on *widget*.

This is the analogue of `g-object-freeze-notify` for child properties.

widget a `<gtk-widget>`

`gtk-widget-get-child-visible` (*self* `<gtk-widget>`) ⇒ (*ret* `bool`) [Function]
`get-child-visible` [Method]

Gets the value set with `gtk-widget-set-child-visible`. If you feel a need to use this function, your code probably needs reorganization.

This function is only useful for container implementations and never should be called by an application.

widget a `<gtk-widget>`

ret '`#t`' if the widget is mapped with the parent.

`gtk-widget-get-parent` (*self* `<gtk-widget>`) ⇒ (*ret* `<gtk-widget>`) [Function]
`get-parent` [Method]

Returns the parent container of *widget*.

widget a `<gtk-widget>`

ret the parent container of *widget*, or '`#f`'

`gtk-widget-get-settings` (*self* `<gtk-widget>`) [Function]
⇒ (*ret* `<gtk-settings>`)
`get-settings` [Method]

Gets the settings object holding the settings (global property settings, RC file information, etc) used for this widget.

Note that this function can only be called when the `<gtk-widget>` is attached to a toplevel, since the settings object is specific to a particular `<gdk-screen>`.

widget a `<gtk-widget>`

ret the relevant `<gtk-settings>` object

`gtk-widget-get-clipboard` (*self* `<gtk-widget>`) [Function]
(*selection* `<gdk-atom>`) ⇒ (*ret* `<gtk-clipboard*>`)
`get-clipboard` [Method]

Returns the clipboard object for the given selection to be used with *widget*. *widget* must have a `<gdk-display>` associated with it, so must be attached to a toplevel window.

widget a `<gtk-widget>`

selection a `<gdk-atom>` which identifies the clipboard to use. '`GDK_SELECTION_CLIPBOARD`' gives the default clipboard. Another common value is '`GDK_SELECTION_PRIMARY`', which gives the primary X selection.

ret the appropriate clipboard object. If no clipboard already exists, a new one will be created. Once a clipboard object has been created, it is persistent for all time.

Since 2.2

`gtk-widget-get-display` (*self* `<gtk-widget>`) [Function]
⇒ (*ret* `<gdk-display>`)

`get-display` [Method]

Get the `<gdk-display>` for the toplevel window associated with this widget. This function can only be called after the widget has been added to a widget hierarchy with a `<gtk-window>` at the top.

In general, you should only create display specific resources when a widget has been realized, and you should free those resources when the widget is unrealized.

widget a `<gtk-widget>`

ret the `<gdk-display>` for the toplevel for this widget.

Since 2.2

`gtk-widget-get-root-window` (*self* `<gtk-widget>`) [Function]
⇒ (*ret* `<gdk-window*>`)

`get-root-window` [Method]

Get the root window where this widget is located. This function can only be called after the widget has been added to a widget heirarchy with `<gtk-window>` at the top.

The root window is useful for such purposes as creating a popup `<gdk-window>` associated with the window. In general, you should only create display specific resources when a widget has been realized, and you should free those resources when the widget is unrealized.

widget a `<gtk-widget>`

ret the `<gdk-window>` root window for the toplevel for this widget.

Since 2.2

`gtk-widget-get-screen` (*self* `<gtk-widget>`) ⇒ (*ret* `<gdk-screen>`) [Function]

`get-screen` [Method]

Get the `<gdk-screen>` from the toplevel window associated with this widget. This function can only be called after the widget has been added to a widget hierarchy with a `<gtk-window>` at the top.

In general, you should only create screen specific resources when a widget has been realized, and you should free those resources when the widget is unrealized.

widget a `<gtk-widget>`

ret the `<gdk-screen>` for the toplevel for this widget.

Since 2.2

`gtk-widget-has-screen` (*self* `<gtk-widget>`) ⇒ (*ret* `bool`) [Function]
`has-screen` [Method]

Checks whether there is a `<gdk-screen>` is associated with this widget. All toplevel widgets have an associated screen, and all widgets added into a heirarchy with a toplevel window at the top.

widget a `<gtk-widget>`

ret '`#t`' if there is a `<gdk-screen>` associcated with the widget.

Since 2.2

`gtk-widget-get-size-request` (*self* `<gtk-widget>`) ⇒ (*width* `int`) (*height* `int`) [Function]
`get-size-request` [Method]

Gets the size request that was explicitly set for the widget using `gtk-widget-set-size-request`. A value of -1 stored in *width* or *height* indicates that that dimension has not been set explicitly and the natural requisition of the widget will be used intead. See `gtk-widget-set-size-request`. To get the size a widget will actually use, call `gtk-widget-size-request` instead of this function.

widget a `<gtk-widget>`

width return location for width, or '`#f`'

height return location for height, or '`#f`'

`gtk-widget-set-child-visible` (*self* `<gtk-widget>`) (*is_visible* `bool`) [Function]
`set-child-visible` [Method]

Sets whether *widget* should be mapped along with its when its parent is mapped and *widget* has been shown with `gtk-widget-show`.

The child visibility can be set for widget before it is added to a container with `gtk-widget-set-parent`, to avoid mapping children unnecessary before immediately unmapping them. However it will be reset to its default state of '`#t`' when the widget is removed from a container.

Note that changing the child visibility of a widget does not queue a resize on the widget. Most of the time, the size of a widget is computed from all visible children, whether or not they are mapped. If this is not the case, the container can queue a resize itself.

This function is only useful for container implementations and never should be called by an application.

widget a `<gtk-widget>`

is-visible if '`#t`', *widget* should be mapped along with its parent.

`gtk-widget-set-size-request` (*self* `<gtk-widget>`) (*width* `int`) (*height* `int`) [Function]
`set-size-request` [Method]

Sets the minimum size of a widget; that is, the widget's size request will be *width* by *height*. You can use this function to force a widget to be either larger or smaller than it normally would be.

In most cases, `gtk-window-set-default-size` is a better choice for toplevel windows than this function; setting the default size will still allow users to shrink the window. Setting the size request will force them to leave the window at least as large as the size request. When dealing with window sizes, `gtk-window-set-geometry-hints` can be a useful function as well.

Note the inherent danger of setting any fixed size - themes, translations into other languages, different fonts, and user action can all change the appropriate size for a given widget. So, it's basically impossible to hardcode a size that will always be correct.

The size request of a widget is the smallest size a widget can accept while still functioning well and drawing itself correctly. However in some strange cases a widget may be allocated less than its requested size, and in many cases a widget may be allocated more space than it requested.

If the size request in a given direction is -1 (unset), then the "natural" size request of the widget will be used instead.

Widgets can't actually be allocated a size less than 1 by 1, but you can pass 0,0 to this function to mean "as small as possible."

widget a `<gtk-widget>`

width width *widget* should request, or -1 to unset

height height *widget* should request, or -1 to unset

`gtk-widget-thaw-child-notify` (*self* `<gtk-widget>`) [Function]
`thaw-child-notify` [Method]

Reverts the effect of a previous call to `gtk-widget-freeze-child-notify`. This causes all queued "child-notify" signals on *widget* to be emitted.

widget a `<gtk-widget>`

`gtk-widget-set-no-show-all` (*self* `<gtk-widget>`) [Function]
(*no_show_all* `bool`)
`set-no-show-all` [Method]

Sets the "no_show_all" property, which determines whether calls to `gtk-widget-show-all` and `gtk-widget-hide-all` will affect this widget.

This is mostly for use in constructing widget hierarchies with externally controlled visibility, see `<gtk-ui-manager>`.

widget a `<gtk-widget>`

no-show-all
the new value for the "no_show_all" property

Since 2.4

`gtk-widget-get-no-show-all` (*self* `<gtk-widget>`) ⇒ (*ret* `bool`) [Function]
`get-no-show-all` [Method]

Returns the current value of the "no_show_all" property, which determines whether calls to `gtk-widget-show-all` and `gtk-widget-hide-all` will affect this widget.

widget a <gtk-widget>

ret the current value of the "no_show_all" property.

Since 2.4

`gtk-widget-list-mnemonic-labels` (*self* <gtk-widget>) ⇒ (*ret* glist-of) [Function]

`list-mnemonic-labels` [Method]

Returns a newly allocated list of the widgets, normally labels, for which this widget is a the target of a mnemonic (see for example, `gtk-label-set-mnemonic-widget`).

The widgets in the list are not individually referenced. If you want to iterate through the list and perform actions involving callbacks that might destroy the widgets, you *must* call '`g_list_foreach (result, (GFunc)g_object_ref, NULL)`' first, and then unref all the widgets afterwards.

widget a <gtk-widget>

ret the list of mnemonic labels; free this list with `g-list-free` when you are done with it.

Since 2.4

`gtk-widget-add-mnemonic-label` (*self* <gtk-widget>) (*label* <gtk-widget>) [Function]

`add-mnemonic-label` [Method]

Adds a widget to the list of mnemonic labels for this widget. (See `gtk-widget-list-mnemonic-labels`). Note the list of mnemonic labels for the widget is cleared when the widget is destroyed, so the caller must make sure to update its internal state at this point as well, by using a connection to the ::destroy signal or a weak notifier.

widget a <gtk-widget>

label a <gtk-widget> that acts as a mnemonic label for *widget*.

Since 2.4

`gtk-widget-remove-mnemonic-label` (*self* <gtk-widget>) (*label* <gtk-widget>) [Function]

`remove-mnemonic-label` [Method]

Removes a widget from the list of mnemonic labels for this widget. (See `gtk-widget-list-mnemonic-labels`). The widget must have previously been added to the list with `gtk-widget-add-mnemonic-label`.

widget a <gtk-widget>

label a <gtk-widget> that was previously set as a mnemnic label for *widget* with `gtk-widget-add-mnemonic-label`.

Since 2.4

`gtk-widget-get-action` (*self* <gtk-widget>) ⇒ (*ret* <gtk-action>) [Function]

`get-action` [Method]

Returns the <gtk-action> that *widget* is a proxy for. See also `gtk-action-get-proxies`.

widget a `<gtk-widget>`

ret the action that a widget is a proxy for, or '`#f`', if it is not attached to an action.

Since 2.10

`gtk-widget-is-composited` (*self* `<gtk-widget>`) ⇒ (*ret* `bool`) [Function]
`is-composited` [Method]

Whether *widget* can rely on having its alpha channel drawn correctly. On X11 this function returns whether a compositing manager is running for *widget*'s screen

widget a `<gtk-widget>`

ret '`#t`' if the widget can rely on its alpha channel being drawn correctly.

Since 2.10

`gtk-requisition-copy` (*self* `<gtk-requisition>`) [Function]
⇒ (*ret* `<gtk-requisition>`)

Copies a `<gtk-requisition>`.

requisition a `<gtk-requisition>`.

ret a copy of *requisition*.

143 GtkIMContext

Base class for input method contexts

143.1 Overview

143.2 Usage

`gtk-im-context-set-client-window` (*self* `<gtk-im-context*>`) [Function]
(*window* `<gdk-window*>`)

Set the client window for the input context; this is the `<gdk-window>` in which the input appears. This window is used in order to correctly position status windows, and may also be used for purposes internal to the input method.

context a `<gtk-im-context>`

window the client window. This may be '`#f`' to indicate that the previous client window no longer exists.

`gtk-im-context-get-preedit-string` (*self* `<gtk-im-context*>`) [Function]
(*attrs* `<pango-attr-list**>`) ⇒ (*str* `mchars`) (*cursor_pos* `int`)

Retrieve the current preedit string for the input context, and a list of attributes to apply to the string. This string should be displayed inserted at the insertion point.

context a `<gtk-im-context>`

str location to store the retrieved string. The string retrieved must be freed with `g-free`.

attrs location to store the retrieved attribute list. When you are done with this list, you must unreference it with `pango-attr-list-unref`.

cursor-pos location to store position of cursor (in characters) within the preedit string.

`gtk-im-context-filter-keypress` (*self* `<gtk-im-context*>`) [Function]
(*event* `<gdk-event-key>`) ⇒ (*ret* `bool`)

Allow an input method to internally handle key press and release events. If this function returns '`#t`', then no further processing should be done for this key event.

context a `<gtk-im-context>`

event the key event

ret '`#t`' if the input method handled the key event.

`gtk-im-context-focus-in` (*self* `<gtk-im-context*>`) [Function]

Notify the input method that the widget to which this input context corresponds has gained focus. The input method may, for example, change the displayed feedback to reflect this change.

context a `<gtk-im-context>`

`gtk-im-context-focus-out` (*self* `<gtk-im-context*>`) [Function]
Notify the input method that the widget to which this input context corresponds has lost focus. The input method may, for example, change the displayed feedback or reset the contexts state to reflect this change.

context a `<gtk-im-context>`

`gtk-im-context-reset` (*self* `<gtk-im-context*>`) [Function]
Notify the input method that a change such as a change in cursor position has been made. This will typically cause the input method to clear the preedit state.

context a `<gtk-im-context>`

`gtk-im-context-set-cursor-location` (*self* `<gtk-im-context*>`) [Function]
(*area* `<gdk-rectangle>`)
Notify the input method that a change in cursor position has been made. The location is relative to the client window.

context a `<gtk-im-context>`

area new location

`gtk-im-context-set-use-preedit` (*self* `<gtk-im-context*>`) [Function]
(*use_preedit* `bool`)
Sets whether the IM context should use the preedit string to display feedback. If *use-preedit* is FALSE (default is TRUE), then the IM context may use some other method to display feedback, such as displaying it in a child of the root window.

context a `<gtk-im-context>`

use-preedit
whether the IM context should use the preedit string.

`gtk-im-context-set-surrounding` (*self* `<gtk-im-context*>`) [Function]
(*text* `mchars`) (*len* `int`) (*cursor_index* `int`)
Sets surrounding context around the insertion point and preedit string. This function is expected to be called in response to the GtkIMContext::retrieve_surrounding signal, and will likely have no effect if called at other times.

context a `<gtk-im-context>`

text text surrounding the insertion point, as UTF-8. the preedit string should not be included within *text*.

len the length of *text*, or -1 if *text* is nul-terminated

cursor-index
the byte index of the insertion cursor within *text*.

`gtk-im-context-get-surrounding` (*self* `<gtk-im-context*>`) [Function]
⇒ (*ret* `bool`) (*text* `mchars`) (*cursor_index* `int`)
Retrieves context around the insertion point. Input methods typically want context in order to constrain input text based on existing text; this is important for languages such as Thai where only some sequences of characters are allowed.

This function is implemented by emitting the GtkIMContext::retrieve_surrounding signal on the input method; in response to this signal, a widget should provide as much context as is available, up to an entire paragraph, by calling `gtk-im-context-set-surrounding`. Note that there is no obligation for a widget to respond to the ::retrieve_surrounding signal, so input methods must be prepared to function without context.

context a `<gtk-im-context>`

text location to store a UTF-8 encoded string of text holding context around the insertion point. If the function returns '`#t`', then you must free the result stored in this location with `g-free`.

cursor-index location to store byte index of the insertion cursor within *text*.

ret '`#t`' if surrounding text was provided; in this case you must free the result stored in *text.

`gtk-im-context-delete-surrounding` (*self* `<gtk-im-context*>`) [Function]
(*offset* `int`) (*n_chars* `int`) ⇒ (*ret* `bool`)

Asks the widget that the input context is attached to to delete characters around the cursor position by emitting the GtkIMContext::delete_surrounding signal. Note that *offset* and *n-chars* are in characters not in bytes which differs from the usage other places in `<gtk-im-context>`.

In order to use this function, you should first call `gtk-im-context-get-surrounding` to get the current context, and call this function immediately afterwards to make sure that you know what you are deleting. You should also account for the fact that even if the signal was handled, the input context might not have deleted all the characters that were requested to be deleted.

This function is used by an input method that wants to make subsitutions in the existing text in response to new input. It is not useful for applications.

context a `<gtk-im-context>`

offset offset from cursor position in chars; a negative value means start before the cursor.

n-chars number of characters to delete.

ret '`#t`' if the signal was handled.

144 GtkPlug

Toplevel for embedding into other processes

144.1 Overview

Together with `<gtk-socket>`, `<gtk-plug>` provides the ability to embed widgets from one process into another process in a fashion that is transparent to the user. One process creates a `<gtk-socket>` widget and, passes the ID of that widgets window to the other process, which then creates a `<gtk-plug>` with that window ID. Any widgets contained in the `<gtk-plug>` then will appear inside the first applications window.

The `<gtk-plug>` and `<gtk-socket>` widgets are currently not available on all platforms supported by GTK+.

144.2 Usage

`<gtk-plug>` [Class]

This `<gobject>` class defines the following properties:

`embedded` Whether or not the plug is embedded

`embedded` [Signal on `<gtk-plug>`]

`gtk-plug-construct` (*self* `<gtk-plug>`) (*socket_id* `unsigned-long`) [Function]
`construct` [Method]

Finish the initialization of *plug* for a given `<gtk-socket>` identified by *socket-id*. This function will generally only be used by classes deriving from `<gtk-plug>`.

plug a `<gtk-plug>`.

socket-id the XID of the socket's window.

`gtk-plug-construct-for-display` (*self* `<gtk-plug>`) [Function]
(*display* `<gdk-display>`) (*socket_id* `unsigned-long`)
`construct-for-display` [Method]

Finish the initialization of *plug* for a given `<gtk-socket>` identified by *socket-id* which is currently displayed on *display*. This function will generally only be used by classes deriving from `<gtk-plug>`.

plug a `<gtk-plug>`.

display the `<gdk-display>` associated with *socket-id*'s `<gtk-socket>`.

socket-id the XID of the socket's window.

Since 2.2

`gtk-plug-new` (*socket_id* `unsigned-long`) ⇒ (*ret* `<gtk-widget>`) [Function]

Creates a new plug widget inside the `<gtk-socket>` identified by *socket-id*. If *socket-id* is 0, the plug is left `"unplugged"` and can later be plugged into a `<gtk-socket>` by `gtk-socket-add-id`.

socket-id the window ID of the socket, or 0.

ret the new `<gtk-plug>` widget.

`gtk-plug-new-for-display` (*display* `<gdk-display>`) [Function]
(*socket_id* `unsigned-long`) ⇒ (*ret* `<gtk-widget>`)

Create a new plug widget inside the `<gtk-socket>` identified by socket_id.

display the `<gdk-display>` on which *socket-id* is displayed

socket-id the XID of the socket's window.

ret the new `<gtk-plug>` widget.

Since 2.2

`gtk-plug-get-id` (*self* `<gtk-plug>`) ⇒ (*ret* `unsigned-long`) [Function]
`get-id` [Method]

Gets the window ID of a `<gtk-plug>` widget, which can then be used to embed this window inside another window, for instance with `gtk-socket-add-id`.

plug a `<gtk-plug>`.

ret the window ID for the plug

145 GtkSocket

Container for widgets from other processes

145.1 Overview

Together with <gtk-plug>, <gtk-socket> provides the ability to embed widgets from one process into another process in a fashion that is transparent to the user. One process creates a <gtk-socket> widget and, passes the that widget's window ID to the other process, which then creates a <gtk-plug> with that window ID. Any widgets contained in the <gtk-plug> then will appear inside the first applications window.

The socket's window ID is obtained by using gtk-socket-get-id. Before using this function, the socket must have been realized, and for hence, have been added to its parent.

```
GtkWidget *socket = gtk_socket_new ();
gtk_widget_show (socket);
gtk_container_add (GTK_CONTAINER (parent), socket);

/* The following call is only necessary if one of
 * the ancestors of the socket is not yet visible.
 */
gtk_widget_realize (socket);
g_print ("The ID of the sockets window is %x\n",
         gtk_socket_get_id (socket));
```

Note that if you pass the window ID of the socket to another process that will create a plug in the socket, you must make sure that the socket widget is not destroyed until that plug is created. Violating this rule will cause unpredictable consequences, the most likely consequence being that the plug will appear as a separate toplevel window. You can check if the plug has been created by examining the field of the <gtk-socket> structure. If this field is non-'#f', then the plug has been successfully created inside of the socket.

When GTK+ is notified that the embedded window has been destroyed, then it will destroy the socket as well. You should always, therefore, be prepared for your sockets to be destroyed at any time when the main event loop is running.

The communication between a <gtk-socket> and a <gtk-plug> follows the XEmbed protocol. This protocol has also been implemented in other toolkits, e.g. , allowing the same level of integration when embedding a widget in GTK or vice versa.

A socket can also be used to swallow arbitrary pre-existing top-level windows using gtk-socket-steal, though the integration when this is done will not be as close as between a <gtk-plug> and a <gtk-socket>.

The <gtk-plug> and <gtk-socket> widgets are currently not available on all platforms supported by GTK+.

145.2 Usage

<gtk-socket> [Class]

This <gobject> class defines no properties, other than those defined by its superclasses.

`plug-added` [Signal on `<gtk-socket>`]

This signal is emitted when a client is successfully added to the socket.

`plug-removed` ⇒ `<gboolean>` [Signal on `<gtk-socket>`]

This signal is emitted when a client is removed from the socket. The default action is to destroy the `<gtk-socket>` widget, so if you want to reuse it you must add a signal handler that returns '`#t`'.

`gtk-socket-new` ⇒ (*ret* `<gtk-widget>`) [Function]

Create a new empty `<gtk-socket>`.

ret the new `<gtk-socket>`.

`gtk-socket-add-id` (*self* `<gtk-socket>`) (*window_id* `unsigned-long`) [Function]
`add-id` [Method]

Adds an XEMBED client, such as a `<gtk-plug>`, to the `<gtk-socket>`. The client may be in the same process or in a different process.

To embed a `<gtk-plug>` in a `<gtk-socket>`, you can either create the `<gtk-plug>` with '`gtk_plug_new (0)`', call `gtk-plug-get-id` to get the window ID of the plug, and then pass that to the `gtk-socket-add-id`, or you can call `gtk-socket-get-id` to get the window ID for the socket, and call `gtk-plug-new` passing in that ID.

The `<gtk-socket>` must have already be added into a toplevel window before you can make this call.

socket a `<gtk-socket>`

window-id the window ID of a client participating in the XEMBED protocol.

`gtk-socket-get-id` (*self* `<gtk-socket>`) ⇒ (*ret* `unsigned-long`) [Function]
`get-id` [Method]

Gets the window ID of a `<gtk-socket>` widget, which can then be used to create a client embedded inside the socket, for instance with `gtk-plug-new`.

The `<gtk-socket>` must have already be added into a toplevel window before you can make this call.

socket a `<gtk-socket>`.

ret the window ID for the socket

146 GtkCurve

Allows direct editing of a curve

146.1 Overview

This widget is considered too specialized/little-used for GTK+, and will in the future be moved to some other package. If your application needs this widget, feel free to use it, as the widget does work and is useful in some applications; it's just not of general interest. However, we are not accepting new features for the widget, and it will eventually move out of the GTK+ distribution.

The `<gtk-curve>` widget allows the user to edit a curve covering a range of values. It is typically used to fine-tune color balances in graphics applications like the Gimp.

The `<gtk-curve>` widget has 3 modes of operation - spline, linear and free. In spline mode the user places points on the curve which are automatically connected together into a smooth curve. In linear mode the user places points on the curve which are connected by straight lines. In free mode the user can draw the points of the curve freely, and they are not connected at all.

146.2 Usage

`<gtk-curve>` [Class]

This `<gobject>` class defines the following properties:

`curve-type`
: Is this curve linear, spline interpolated, or free-form

`min-x`
: Minimum possible value for X

`max-x`
: Maximum possible X value

`min-y`
: Minimum possible value for Y

`max-y`
: Maximum possible value for Y

`curve-type-changed` [Signal on `<gtk-curve>`]

Emitted when the curve type has been changed. The curve type can be changed explicitly with a call to `gtk-curve-set-curve-type`. It is also changed as a side-effect of calling `gtk-curve-reset` or `gtk-curve-set-gamma`.

`gtk-curve-new` ⇒ (*ret* `<gtk-widget>`) [Function]

Creates a new `<gtk-curve>`.

ret
: a new `<gtk-curve>`.

`gtk-curve-reset` (*self* `<gtk-curve>`) [Function]
`reset` [Method]

Resets the curve to a straight line from the minimum x and y values to the maximum x and y values (i.e. from the bottom-left to the top-right corners). The curve type is not changed.

curve
: a `<gtk-curve>`.

`gtk-curve-set-gamma` (*self* `<gtk-curve>`) (*gamma* `float`) [Function]
`set-gamma` [Method]

Recomputes the entire curve using the given gamma value. A gamma value of 1 results in a straight line. Values greater than 1 result in a curve above the straight line. Values less than 1 result in a curve below the straight line. The curve type is changed to '`GTK_CURVE_TYPE_FREE`'. FIXME: Needs a more precise definition of gamma.

curve a `<gtk-curve>`.

gamma the gamma value.

`gtk-curve-set-range` (*self* `<gtk-curve>`) (*min_x* `float`) [Function]
(*max_x* `float`) (*min_y* `float`) (*max_y* `float`)
`set-range` [Method]

Sets the minimum and maximum x and y values of the curve. The curve is also reset with a call to `gtk-curve-reset`.

curve a `<gtk-curve>`.

min-x the minimum x value.

max-x the maximum x value.

min-y the minimum y value.

max-y the maximum y value.

`gtk-curve-get-vector` (*self* `<gtk-curve>`) (*veclen* `int`) [Function]
(*vector* `<gfloat*>`)
`get-vector` [Method]

Returns a vector of points representing the curve.

curve a `<gtk-curve>`.

veclen the number of points to calculate.

vector returns the points.

`gtk-curve-set-vector` (*self* `<gtk-curve>`) (*veclen* `int`) [Function]
(*vector* `<gfloat*>`)
`set-vector` [Method]

Sets the vector of points on the curve. The curve type is set to '`GTK_CURVE_TYPE_FREE`'.

curve a `<gtk-curve>`.

veclen the number of points.

vector the points on the curve.

`gtk-curve-set-curve-type` (*self* `<gtk-curve>`) [Function]
(*type* `<gtk-curve-type>`)
`set-curve-type` [Method]

Sets the type of the curve. The curve will remain unchanged except when changing from a free curve to a linear or spline curve, in which case the curve will be changed as little as possible.

curve	a `<gtk-curve>`.
type	the type of the curve.

147 GtkGammaCurve

a subclass of for editing gamma curves.

147.1 Overview

This widget is considered too specialized/little-used for GTK+, and will in the future be moved to some other package. If your application needs this widget, feel free to use it, as the widget does work and is useful in some applications; it's just not of general interest. However, we are not accepting new features for the widget, and it will eventually move out of the GTK+ distribution.

The `<gtk-gamma-curve>` widget is a variant of `<gtk-curve>` specifically for editing gamma curves, which are used in graphics applications such as the Gimp.

The `<gtk-gamma-curve>` widget shows a curve which the user can edit with the mouse just like a `<gtk-curve>` widget. On the right of the curve it also displays 5 buttons, 3 of which change between the 3 curve modes (spline, linear and free), and the other 2 set the curve to a particular gamma value, or reset it to a straight line.

147.2 Usage

`<gtk-gamma-curve>` [Class]

This `<gobject>` class defines no properties, other than those defined by its superclasses.

`gtk-gamma-curve-new` ⇒ (*ret* `<gtk-widget>`) [Function]

Creates a new `<gtk-gamma-curve>`.

ret a new `<gtk-gamma-curve>`.

148 GtkRuler

Base class for horizontal or vertical rulers

148.1 Overview

This widget is considered too specialized/little-used for GTK+, and will in the future be moved to some other package. If your application needs this widget, feel free to use it, as the widget does work and is useful in some applications; it's just not of general interest. However, we are not accepting new features for the widget, and it will eventually move out of the GTK+ distribution.

The GTKRuler widget is a base class for horizontal and vertical rulers. Rulers are used to show the mouse pointer's location in a window. The ruler can either be horizontal or vertical on the window. Within the ruler a small triangle indicates the location of the mouse relative to the horizontal or vertical ruler. See `<gtk-hruler>` to learn how to create a new horizontal ruler. See `<gtk-vruler>` to learn how to create a new vertical ruler.

148.2 Usage

`<gtk-ruler>` [Class]

This `<gobject>` class defines the following properties:

`lower`	Lower limit of ruler
`upper`	Upper limit of ruler
`position`	Position of mark on the ruler
`max-size`	Maximum size of the ruler
`metric`	The metric used for the ruler

`gtk-ruler-set-metric` (*self* `<gtk-ruler>`) [Function]
(*metric* `<gtk-metric-type>`)

`set-metric` [Method]

This calls the `<gtk-metric-type>` to set the ruler to units defined. Available units are GTK_PIXELS, GTK_INCHES, or GTK_CENTIMETERS. The default unit of measurement is GTK_PIXELS.

ruler	the gtkruler
metric	the unit of measurement

`gtk-ruler-set-range` (*self* `<gtk-ruler>`) (*lower* `double`) [Function]
(*upper* `double`) (*position* `double`) (*max_size* `double`)

`set-range` [Method]

This sets the range of the ruler using gfloat lower, gfloat upper, gfloat position, and gfloat max_size.

ruler	the gtkruler
lower	the lower limit of the ruler

upper the upper limit of the ruler

position the mark on the ruler

max-size the maximum size of the ruler

`gtk-ruler-get-metric` (*self* `<gtk-ruler>`) [Function]
⇒ (*ret* `<gtk-metric-type>`)

`get-metric` [Method]

Gets the units used for a `<gtk-ruler>`. See `gtk-ruler-set-metric`.

ruler a `<gtk-ruler>`

ret the units currently used for *ruler*

`gtk-ruler-get-range` (*self* `<gtk-ruler>`) ⇒ (*lower* `double`) [Function]
(*upper* `double`) (*position* `double`) (*max_size* `double`)

`get-range` [Method]

Retrieves values indicating the range and current position of a `<gtk-ruler>`. See `gtk-ruler-set-range`.

ruler a `<gtk-ruler>`

lower location to store lower limit of the ruler, or '`#f`'

upper location to store upper limit of the ruler, or '`#f`'

position location to store the current position of the mark on the ruler, or '`#f`'

max-size location to store the maximum size of the ruler used when calculating the space to leave for the text, or '`#f`'.

149 GtkHRuler

A horizontal ruler.

149.1 Overview

This widget is considered too specialized/little-used for GTK+, and will in the future be moved to some other package. If your application needs this widget, feel free to use it, as the widget does work and is useful in some applications; it's just not of general interest. However, we are not accepting new features for the widget, and it will eventually move out of the GTK+ distribution.

The HRuler widget is a widget arranged horizontally creating a ruler that is utilized around other widgets such as a text widget. The ruler is used to show the location of the mouse on the window and to show the size of the window in specified units. The available units of measurement are GTK_PIXELS, GTK_INCHES and GTK_CENTIMETERS. GTK_PIXELS is the default. rulers.

149.2 Usage

`<gtk-hruler>` [Class]

This `<gobject>` class defines no properties, other than those defined by its superclasses.

`gtk-hruler-new` ⇒ (*ret* `<gtk-widget>`) [Function]

Creates a new horizontal ruler.

ret a new `<gtk-hruler>`.

150 GtkVRuler

A vertical ruler.

150.1 Overview

This widget is considered too specialized/little-used for GTK+, and will in the future be moved to some other package. If your application needs this widget, feel free to use it, as the widget does work and is useful in some applications; it's just not of general interest. However, we are not accepting new features for the widget, and it will eventually move out of the GTK+ distribution.

The VRuler widget is a widget arranged vertically creating a ruler that is utilized around other widgets such as a text widget. The ruler is used to show the location of the mouse on the window and to show the size of the window in specified units. The available units of measurement are GTK_PIXELS, GTK_INCHES and GTK_CENTIMETERS. GTK_PIXELS is the default. rulers.

150.2 Usage

`<gtk-vruler>` [Class]

This `<gobject>` class defines no properties, other than those defined by its superclasses.

`gtk-vruler-new` ⇒ (*ret* `<gtk-widget>`) [Function]

Creates a new vertical ruler

ret a new `<gtk-vruler>`.

151 GtkRecentManager

Managing Recently Used Files

151.1 Overview

<gtk-recent-manager> provides a facility for adding, removing and looking up recently used files. Each recently used file is identified by its URI, and has meta-data associated to it, like the names and command lines of the applications that have registered it, the number of time each application has registered the same file, the mime type of the file and whether the file should be displayed only by the applications that have registered it.

The <gtk-recent-manager> acts like a database of all the recently used files. You can create new <gtk-recent-manager> objects, but it is more efficient to use the standard recent manager for the <gdk-screen> so that informations about the recently used files is shared with other people using them. In case the default screen is being used, adding a new recently used file is as simple as:

```
GtkRecentManager *manager;

manager = gtk_recent_manager_get_default ();
gtk_recent_manager_add_item (manager, file_uri);
```

```
GtkRecentManager *manager;
GtkRecentInfo *info;
GError *error = NULL;

manager = gtk_recent_manager_get_default ();
info = gtk_recent_manager_lookup_item (manager, file_uri, &error);
if (error)
  {
    g_warning ("Could not find the file: %s", error->message);
    g_error_free (error);
  }
else
  {
    /* Use the info object */
    gtk_recent_info_unref (info);
  }
```

Recently used files are supported since GTK+ 2.10.

151.2 Usage

<gtk-recent-manager> [Class]

This <gobject> class defines the following properties:

`filename` The full path to the file to be used to store and read the list

`limit` The maximum number of items to be returned by gtk_recent_manager_get_items()

`size` The size of the recently used resources list

`changed` [Signal on `<gtk-recent-manager>`]

Emitted when the current recently used resources manager changes its contents.

Since 2.10

`gtk-recent-manager-new` ⇒ (*ret* `<gtk-recent-manager>`) [Function]

Creates a new recent manager object. Recent manager objects are used to handle the list of recently used resources. A `<gtk-recent-manager>` object monitors the recently used resources list, and emits the "changed" signal each time something inside the list changes.

`<gtk-recent-manager>` objects are expensive: be sure to create them only when needed. You should use the `gtk-recent-manager-new-for-screen` or the `gtk-recent-manager-get-default` functions instead.

ret A newly created `<gtk-recent-manager>` object.

Since 2.10

`gtk-recent-manager-get-default` ⇒ (*ret* `<gtk-recent-manager>`) [Function]

Gets the recent manager for the default screen. See `gtk-recent-manager-get-for-screen`.

ret A unique `<gtk-recent-manager>` associated with the default screen. This recent manager is associated with the screen and can be used as long as the screen is open. Do not ref or unref it.

Since 2.10

`gtk-recent-manager-get-for-screen` (*screen* `<gdk-screen>`) [Function]
⇒ (*ret* `<gtk-recent-manager>`)

Gets the recent manager object associated with *screen*; if this function has not previously been called for the given screen, a new recent manager object will be created and associated with the screen. Recent manager objects are fairly expensive to create, so using this function is usually a better choice than calling `gtk-recent-manager-new` and setting the screen yourself; by using this function a single recent manager object will be shared between users.

screen a `<gdk-screen>`

ret A unique `<gtk-recent-manager>` associated with the given screen. This recent manager is associated to the with the screen and can be used as long as the screen is open. Do not ref or unref it.

Since 2.10

`gtk-recent-manager-set-screen` (*self* `<gtk-recent-manager>`) [Function]
(*screen* `<gdk-screen>`)

`set-screen` [Method]

Sets the screen for a recent manager; the screen is used to track the user's currently configured recently used documents storage.

manager a `<gtk-recent-manager>`

screen a `<gdk-screen>`

Since 2.10

`gtk-recent-manager-add-item` (*self* `<gtk-recent-manager>`) [Function]
(*uri* `mchars`) ⇒ (*ret* `bool`)

`add-item` [Method]

Adds a new resource, pointed by *uri*, into the recently used resources list.

This function automatically retrieving some of the needed metadata and setting other metadata to common default values; it then feeds the data to `gtk-recent-manager-add-full`.

See `gtk-recent-manager-add-full` if you want to explicitely define the metadata for the resource pointed by *uri*.

manager a `<gtk-recent-manager>`

uri a valid URI

ret '`#t`' if the new item was successfully added to the recently used resources list

Since 2.10

`gtk-recent-manager-remove-item` (*self* `<gtk-recent-manager>`) [Function]
(*uri* `mchars`) ⇒ (*ret* `bool`)

`remove-item` [Method]

Removes a resource pointed by *uri* from the recently used resources list handled by a recent manager.

manager a `<gtk-recent-manager>`

uri the URI of the item you wish to remove

error return location for a `<g-error>`, or '`#f`'

ret '`#t`' if the item pointed by *uri* has been successfully removed by the recently used resources list, and '`#f`' otherwise.

Since 2.10

`gtk-recent-manager-lookup-item` (*self* `<gtk-recent-manager>`) [Function]
(*uri* `mchars`) ⇒ (*ret* `<gtk-recent-info*>`)

`lookup-item` [Method]

Searches for a URI inside the recently used resources list, and returns a structure containing informations about the resource like its MIME type, or its display name.

manager a `<gtk-recent-manager>`

uri a URI

error a return location for a `<g-error>`, or '`#f`'

ret a `<gtk-recent-info>` structure containing information about the resource pointed by *uri*, or '`#f`' if the URI was not registered in the recently used resources list. Free with `gtk-recent-info-unref`.

Since 2.10

`gtk-recent-manager-has-item` (*self* `<gtk-recent-manager>`) [Function]
(*uri* `mchars`) ⇒ (*ret* `bool`)

`has-item` [Method]

Checks whether there is a recently used resource registered with *uri* inside the recent manager.

manager a `<gtk-recent-manager>`

uri a URI

ret '`#t`' if the resource was found, '`#f`' otherwise.

Since 2.10

`gtk-recent-manager-move-item` (*self* `<gtk-recent-manager>`) [Function]
(*uri* `mchars`) (*new_uri* `mchars`) ⇒ (*ret* `bool`)

`move-item` [Method]

Changes the location of a recently used resource from *uri* to *new-uri*.

Please note that this function will not affect the resource pointed by the URIs, but only the URI used in the recently used resources list.

manager a `<gtk-recent-manager>`

uri the URI of a recently used resource

new-uri the new URI of the recently used resource, or '`#f`' to remove the item pointed by *uri* in the list

error a return location for a `<g-error>`, or '`#f`'

ret '`#t`' on success.

Since 2.10

`gtk-recent-manager-get-limit` (*self* `<gtk-recent-manager>`) [Function]
⇒ (*ret* `int`)

`get-limit` [Method]

Gets the maximum number of items that the `gtk-recent-manager-get-items` function should return.

manager a `<gtk-recent-manager>`

ret the number of items to return, or -1 for every item.

Since 2.10

`gtk-recent-manager-set-limit` (*self* `<gtk-recent-manager>`) [Function]
(*limit* `int`)

`set-limit` [Method]

Sets the maximum number of item that the `gtk-recent-manager-get-items` function should return. If *limit* is set to -1, then return all the items.

manager a `<gtk-recent-manager>`

limit the maximum number of items to return, or -1.

Since 2.10

`gtk-recent-manager-get-items` (*self* `<gtk-recent-manager>`) [Function]
⇒ (*ret* `glist-of`)

`get-items` [Method]

Gets the list of recently used resources.

manager a `<gtk-recent-manager>`

ret a list of newly allocated `<gtk-recent-info>` objects. Use `gtk-recent-info-unref` on each item inside the list, and then free the list itself using `g-list-free`.

Since 2.10

`gtk-recent-manager-purge-items` (*self* `<gtk-recent-manager>`) [Function]
⇒ (*ret* `int`)

`purge-items` [Method]

Purges every item from the recently used resources list.

manager a `<gtk-recent-manager>`

error a return location for a `<g-error>`, or '`#f`'

ret the number of items that have been removed from the recently used resources list.

Since 2.10

`gtk-recent-info-get-uri` (*self* `<gtk-recent-info*>`) [Function]
⇒ (*ret* `mchars`)

Gets the URI of the resource.

info a `<gtk-recent-info>`

ret the URI of the resource. The returned string is owned by the recent manager, and should not be freed.

Since 2.10

`gtk-recent-info-get-display-name` (*self* `<gtk-recent-info*>`) [Function]
⇒ (*ret* `mchars`)

Gets the name of the resource. If none has been defined, the basename of the resource is obtained.

info a `<gtk-recent-info>`

ret the display name of the resource. The returned string is owned by the recent manager, and should not be freed.

Since 2.10

`gtk-recent-info-get-description` (*self* `<gtk-recent-info*>`) [Function]
⇒ (*ret* `mchars`)

Gets the (short) description of the resource.

info a `<gtk-recent-info>`

ret the description of the resource. The returned string is owned by the recent manager, and should not be freed.

Since 2.10

`gtk-recent-info-get-mime-type` (*self* `<gtk-recent-info*>`) [Function]
⇒ (*ret* `mchars`)

Gets the MIME type of the resource.

info a `<gtk-recent-info>`

ret the MIME type of the resource. The returned string is owned by the recent manager, and should not be freed.

Since 2.10

`gtk-recent-info-get-added` (*self* `<gtk-recent-info*>`) [Function]
⇒ (*ret* `long`)

Gets the timestamp (seconds from system's Epoch) when the resource was added to the recently used resources list.

info a `<gtk-recent-info>`

ret the number of seconds elapsed from system's Epoch when the resource was added to the list, or -1 on failure.

Since 2.10

`gtk-recent-info-get-modified` (*self* `<gtk-recent-info*>`) [Function]
⇒ (*ret* `long`)

Gets the timestamp (seconds from system's Epoch) when the resource was last modified.

info a `<gtk-recent-info>`

ret the number of seconds elapsed from system's Epoch when the resource was last modified, or -1 on failure.

Since 2.10

`gtk-recent-info-get-visited` (*self* `<gtk-recent-info*>`) [Function]
⇒ (*ret* `long`)

Gets the timestamp (seconds from system's Epoch) when the resource was last visited.

info a `<gtk-recent-info>`

ret the number of seconds elapsed from system's Epoch when the resource was last visited, or -1 on failure.

Since 2.10

`gtk-recent-info-get-private-hint` (*self* `<gtk-recent-info*>`) [Function]
⇒ (*ret* `bool`)

Gets the value of the `"private"` flag. Resources in the recently used list that have this flag set to '`#t`' should only be displayed by the applications that have registered them.

info a `<gtk-recent-info>`

ret '`#t`' if the private flag was found, '`#f`' otherwise.

Since 2.10

`gtk-recent-info-last-application` (*self* `<gtk-recent-info*>`) [Function]
⇒ (*ret* `mchars`)

Gets the name of the last application that have registered the recently used resource represented by *info*.

info a `<gtk-recent-info>`

ret an application name. Use `g-free` to free it.

Since 2.10

`gtk-recent-info-has-group` (*self* `<gtk-recent-info*>`) [Function]
(*group_name* `mchars`) ⇒ (*ret* `bool`)

Checks whether *group-name* appears inside the groups registered for the recently used item *info*.

info a `<gtk-recent-info>`

group-name
name of a group

ret '`#t`' if the group was found.

Since 2.10

`gtk-recent-info-has-application` (*self* `<gtk-recent-info*>`) [Function]
(*app_name* `mchars`) ⇒ (*ret* `bool`)

Checks whether an application registered this resource using *app-name*.

info a `<gtk-recent-info>`

app-name a string containing an application name

ret '`#t`' if an application with name *app-name* was found, '`#f`' otherwise.

Since 2.10

`gtk-recent-info-get-icon` (*self* `<gtk-recent-info*>`) (*size* `int`) [Function]
⇒ (*ret* `<gdk-pixbuf>`)

Retrieves the icon of size *size* associated to the resource MIME type.

info a `<gtk-recent-info>`

size the size of the icon in pixels

ret a `<gdk-pixbuf>` containing the icon, or '`#f`'.

Since 2.10

`gtk-recent-info-get-short-name` (*self* `<gtk-recent-info*>`) [Function]
⇒ (*ret* `mchars`)

Computes a valid UTF-8 string that can be used as the name of the item in a menu or list. For example, calling this function on an item that refers to "file:///foo/bar.txt" will yield "bar.txt".

info an `<gtk-recent-info>`

ret A newly-allocated string in UTF-8 encoding; free it with `g-free`.

Since 2.10

`gtk-recent-info-get-uri-display` (*self* `<gtk-recent-info*>`) [Function]
⇒ (*ret* `mchars`)

Gets a displayable version of the resource's URI.

info a `<gtk-recent-info>`

ret a UTF-8 string containing the resource's URI or '`#f`'

Since 2.10

`gtk-recent-info-get-age` (*self* `<gtk-recent-info*>`) ⇒ (*ret* `int`) [Function]

Gets the number of days elapsed since the last update of the resource pointed by *info*.

info a `<gtk-recent-info>`

ret a positive integer containing the number of days elapsed since the time this resource was last modified.

Since 2.10

`gtk-recent-info-is-local` (*self* `<gtk-recent-info*>`) [Function]
⇒ (*ret* `bool`)

Checks whether the resource is local or not by looking at the scheme of its URI.

info a `<gtk-recent-info>`

ret '`#t`' if the resource is local.

Since 2.10

`gtk-recent-info-exists` (*self* `<gtk-recent-info*>`) ⇒ (*ret* `bool`) [Function]

Checks whether the resource pointed by *info* still exists. At the moment this check is done only on resources pointing to local files.

info a `<gtk-recent-info>`

ret '`#t`' if the resource exists

Since 2.10

`gtk-recent-info-match` (*self* `<gtk-recent-info*>`) [Function]
(*info_b* `<gtk-recent-info*>`) ⇒ (*ret* `bool`)

Checks whether two `<gtk-recent-info>` structures point to the same resource.

info-a a `<gtk-recent-info>`

info-b a `<gtk-recent-info>`

ret '`#t`' if both `<gtk-recent-info>` structures point to se same resource, '`#f`' otherwise.

Since 2.10

152 GtkRecentChooser

Interface implemented by GtkRecentChooserWidget, GtkRecentChooserMenu and GtkRecentChooserDialog

152.1 Overview

`<gtk-recent-chooser>` is an interface that can be implemented by widgets displaying the list of recently used files. In GTK+, the main objects that implement this interface are `<gtk-recent-chooser-widget>`, `<gtk-recent-chooser-dialog>` and `<gtk-recent-chooser-menu>`.

Recently used files are supported since GTK+ 2.10.

152.2 Usage

`gtk-recent-chooser-set-show-private` [Function]
(*self* `<gtk-recent-chooser*>`) (*show_private* `bool`)

Whether to show recently used resources marked registered as private.

chooser a `<gtk-recent-chooser>`

show-private
'`#t`' to show private items, '`#f`' otherwise

Since 2.10

`gtk-recent-chooser-get-show-private` [Function]
(*self* `<gtk-recent-chooser*>`) ⇒ (*ret* `bool`)

Returns whether *chooser* should display recently used resources registered as private.

chooser a `<gtk-recent-chooser>`

ret '`#t`' if the recent chooser should show private items, '`#f`' otherwise.

Since 2.10

`gtk-recent-chooser-set-show-icons` [Function]
(*self* `<gtk-recent-chooser*>`) (*show_icons* `bool`)

Sets whether *chooser* should show an icon near the resource when displaying it.

chooser a `<gtk-recent-chooser>`

show-icons
whether to show an icon near the resource

Since 2.10

`gtk-recent-chooser-get-show-icons` [Function]
(*self* `<gtk-recent-chooser*>`) ⇒ (*ret* `bool`)

Retrieves whether *chooser* should show an icon near the resource.

chooser a `<gtk-recent-chooser>`

ret '`#t`' if the icons should be displayed, '`#f`' otherwise.

Since 2.10

`gtk-recent-chooser-set-local-only` [Function]
(*self* `<gtk-recent-chooser*>`) (*local_only* `bool`)

Sets whether only local resources, that is resources using the file:// URI scheme, should be shown in the recently used resources selector. If *local-only* is '`#t`' (the default) then the shown resources are guaranteed to be accessible through the operating system native file system.

chooser a `<gtk-recent-chooser>`

local-only '`#t`' if only local files can be shown

Since 2.10

`gtk-recent-chooser-get-local-only` [Function]
(*self* `<gtk-recent-chooser*>`) ⇒ (*ret* `bool`)

Gets whether only local resources should be shown in the recently used resources selector. See `gtk-recent-chooser-set-local-only`

chooser a `<gtk-recent-chooser>`

ret '`#t`' if only local resources should be shown.

Since 2.10

`gtk-recent-chooser-set-limit` (*self* `<gtk-recent-chooser*>`) [Function]
(*limit* `int`)

Sets the number of items that should be returned by `gtk-recent-chooser-get-items` and `gtk-recent-chooser-get-uris`.

chooser a `<gtk-recent-chooser>`

limit a positive integer, or -1 for all items

Since 2.10

`gtk-recent-chooser-get-limit` (*self* `<gtk-recent-chooser*>`) [Function]
⇒ (*ret* `int`)

Gets the number of items returned by `gtk-recent-chooser-get-items` and `gtk-recent-chooser-get-uris`.

chooser a `<gtk-recent-chooser>`

ret A positive integer, or -1 meaning that all items are returned.

Since 2.10

`gtk-recent-chooser-set-show-tips` (*self* `<gtk-recent-chooser*>`) [Function]
(*show_tips* `bool`)

Sets whether to show a tooltips on the widget.

chooser a `<gtk-recent-chooser>`

show-tips '`#t`' if tooltips should be shown

Since 2.10

`gtk-recent-chooser-get-show-tips` (*self* `<gtk-recent-chooser*>`) ⇒ (*ret* `bool`) [Function]

Gets whether *chooser* should display tooltips.

chooser a `<gtk-recent-chooser>`

ret '`#t`' if the recent chooser should show tooltips, '`#f`' otherwise.

Since 2.10

`gtk-recent-chooser-set-show-numbers` (*self* `<gtk-recent-chooser*>`) (*show_numbers* `bool`) [Function]

Whether to show recently used resources prepended by a unique number.

Do not use this function: use `gtk-recent-chooser-menu-set-show-numbers` instead.

chooser a `<gtk-recent-chooser>`

show-numbers '`#t`' to show numbers, '`#f`' otherwise

Since 2.10

`gtk-recent-chooser-get-show-numbers` (*self* `<gtk-recent-chooser*>`) ⇒ (*ret* `bool`) [Function]

Returns whether *chooser* should display recently used resources prepended by a unique number.

Do not use this function: use `gtk-recent-chooser-menu-get-show-numbers` instead.

chooser a `<gtk-recent-chooser>`

ret '`#t`' if the recent chooser should show display numbers, '`#f`' otherwise.

Since 2.10

`gtk-recent-chooser-set-sort-type` (*self* `<gtk-recent-chooser*>`) (*sort_type* `<gtk-recent-sort-type>`) [Function]

Changes the sorting order of the recently used resources list displayed by *chooser*.

chooser a `<gtk-recent-chooser>`

sort-type sort order that the chooser should use

Since 2.10

`gtk-recent-chooser-get-sort-type` (*self* `<gtk-recent-chooser*>`) ⇒ (*ret* `<gtk-recent-sort-type>`) [Function]

Gets the value set by `gtk-recent-chooser-set-sort-type`.

chooser a `<gtk-recent-chooser>`

ret the sorting order of the *chooser*.

Since 2.10

`gtk-recent-chooser-set-sort-func` (*self* `<gtk-recent-chooser*>`) [Function]
(*sort_func* `<gtk-recent-sort-func>`) (*sort_data* `<gpointer>`)
(*data_destroy* `<g-destroy-notify>`)

Sets the comparison function used when sorting to be *sort-func*. If the *chooser* has the sort type set to `<gtk-recent-sort-custom>` then the chooser will sort using this function.

To the comparison function will be passed two `<gtk-recent-info>` structs and *sort-data*; *sort-func* should return a positive integer if the first item comes before the second, zero if the two items are equal and a negative integer if the first item comes after the second.

chooser a `<gtk-recent-chooser>`

sort-func the comparison function

sort-data user data to pass to *sort-func*, or '`#f`'

data-destroy
destroy notifier for *sort-data*, or '`#f`'

Since 2.10

`gtk-recent-chooser-set-current-uri` [Function]
(*self* `<gtk-recent-chooser*>`) (*uri* `mchars`) ⇒ (*ret* `bool`)

Sets *uri* as the current URI for *chooser*.

chooser a `<gtk-recent-chooser>`

uri a URI

error return location for a `<g-error>`, or '`#f`'

ret '`#t`' if the URI was found.

Since 2.10

`gtk-recent-chooser-get-current-uri` [Function]
(*self* `<gtk-recent-chooser*>`) ⇒ (*ret* `mchars`)

Gets the URI currently selected by *chooser*.

chooser a `<gtk-recent-chooser>`

ret a newly allocated string holding a URI.

Since 2.10

`gtk-recent-chooser-get-current-item` [Function]
(*self* `<gtk-recent-chooser*>`) ⇒ (*ret* `<gtk-recent-info*>`)

Gets the `<gtk-recent-info>` currently selected by *chooser*.

chooser a `<gtk-recent-chooser>`

ret a `<gtk-recent-info>`. Use `gtk-recent-info-unref` when when you have finished using it.

Since 2.10

`gtk-recent-chooser-select-uri` (*self* `<gtk-recent-chooser*>`) [Function]
(*uri* `mchars`) ⇒ (*ret* `bool`)

Selects *uri* inside *chooser*.

chooser a `<gtk-recent-chooser>`

uri a URI

error return location for a `<g-error>`, or '`#f`'

ret '`#t`' if *uri* was found.

Since 2.10

`gtk-recent-chooser-unselect-uri` (*self* `<gtk-recent-chooser*>`) [Function]
(*uri* `mchars`)

Unselects *uri* inside *chooser*.

chooser a `<gtk-recent-chooser>`

uri a URI

Since 2.10

`gtk-recent-chooser-select-all` (*self* `<gtk-recent-chooser*>`) [Function]

Selects all the items inside *chooser*, if the *chooser* supports multiple selection.

chooser a `<gtk-recent-chooser>`

Since 2.10

`gtk-recent-chooser-unselect-all` (*self* `<gtk-recent-chooser*>`) [Function]

Unselects all the items inside *chooser*.

chooser a `<gtk-recent-chooser>`

Since 2.10

`gtk-recent-chooser-get-items` (*self* `<gtk-recent-chooser*>`) [Function]
⇒ (*ret* `glist-of`)

Gets the list of recently used resources in form of `<gtk-recent-info>` objects.

The return value of this function is affected by the "sort-type" and "limit" properties of *chooser*.

chooser a `<gtk-recent-chooser>`

ret A newly allocated list of `<gtk-recent-info>` objects. You should use `gtk-recent-info-unref` on every item of the list, and then free the list itself using `g-list-free`.

Since 2.10

`gtk-recent-chooser-get-uris` (*self* `<gtk-recent-chooser*>`) [Function]
⇒ (*ret* `<gchar**>`) (*length* `size_t`)

Gets the URI of the recently used resources.

The return value of this function is affected by the "sort-type" and "limit" properties of *chooser*.

Since the returned array is '`#f`' terminated, *length* may be '`#f`'.

chooser a `<gtk-recent-chooser>`

length return location for a the length of the URI list, or '`#f`'

ret A newly allocated, '`#f`' terminated array of strings. Use `g-strfreev` to free it.

Since 2.10

`gtk-recent-chooser-add-filter` (*self* `<gtk-recent-chooser*>`) [Function]
(*filter* `<gtk-recent-filter*>`)

Adds *filter* to the list of `<gtk-recent-filter>` objects held by *chooser*.

If no previous filter objects were defined, this function will call `gtk-recent-chooser-set-filter`.

chooser a `<gtk-recent-chooser>`

filter a `<gtk-recent-filter>`

Since 2.10

`gtk-recent-chooser-remove-filter` (*self* `<gtk-recent-chooser*>`) [Function]
(*filter* `<gtk-recent-filter*>`)

Removes *filter* from the list of `<gtk-recent-filter>` objects held by *chooser*.

chooser a `<gtk-recent-chooser>`

filter a `<gtk-recent-filter>`

Since 2.10

`gtk-recent-chooser-list-filters` (*self* `<gtk-recent-chooser*>`) [Function]
⇒ (*ret* `gslist-of`)

Gets the `<gtk-recent-filter>` objects held by *chooser*.

chooser a `<gtk-recent-chooser>`

ret A singly linked list of `<gtk-recent-filter>` objects. You should just free the returned list using `g-slist-free`.

Since 2.10

`gtk-recent-chooser-set-filter` (*self* `<gtk-recent-chooser*>`) [Function]
(*filter* `<gtk-recent-filter*>`)

Sets *filter* as the current `<gtk-recent-filter>` object used by *chooser* to affect the displayed recently used resources.

chooser a `<gtk-recent-chooser>`

filter a `<gtk-recent-filter>`

Since 2.10

`gtk-recent-chooser-get-filter` (*self* `<gtk-recent-chooser*>`) [Function]
⇒ (*ret* `<gtk-recent-filter*>`)

Gets the `<gtk-recent-filter>` object currently used by *chooser* to affect the display of the recently used resources.

chooser a `<gtk-recent-chooser>`

ret a `<gtk-recent-filter>` object.

Since 2.10

153 GtkRecentChooserDialog

Displays recently used files in a dialog

153.1 Overview

<gtk-recent-chooser-dialog> is a dialog box suitable for displaying the recently used documents. This widgets works by putting a <gtk-recent-chooser-widget> inside a <gtk-dialog>. It exposes the <gtk-recent-chooser-iface> interface, so you can use all the <gtk-recent-chooser> functions on the recent chooser dialog as well as those for <gtk-dialog>.

Note that <gtk-recent-chooser-dialog> does not have any methods of its own. Instead, you should use the functions that work on a <gtk-recent-chooser>.

In the simplest of cases, you can use the following code to use a <gtk-recent-chooser-dialog> to select a recently used file:

```
GtkWidget *dialog;

dialog = gtk_recent_chooser_dialog_new ("Recent Documents",
parent_window,
GTK_STOCK_CANCEL, GTK_RESPONSE_CANCEL,
GTK_STOCK_OPEN, GTK_RESPONSE_ACCEPT,
NULL);

if (gtk_dialog_run (GTK_DIALOG (dialog)) == GTK_RESPONSE_ACCEPT)
  {
    GtkRecentInfo *info;

    info = gtk_recent_chooser_get_current_item (GTK_RECENT_CHOOSER (dialog));
    open_file (gtk_recent_info_get_uri (info));
    gtk_recent_info_unref (info);
  }

gtk_widget_destroy (dialog);
```

Recently used files are supported since GTK+ 2.10.

153.2 Usage

<gtk-recent-chooser-dialog> [Class]

This <gobject> class defines no properties, other than those defined by its superclasses.

154 GtkRecentChooserMenu

Displays recently used files in a menu

154.1 Overview

<gtk-recent-chooser-menu> is a widget suitable for displaying recently used files inside a menu. It can be used to set a sub-menu of a <gtk-menu-item> using gtk-menu-item-set-submenu, or as the menu of a <gtk-menu-tool-button>.

Note that <gtk-recent-chooser-menu> does not have any methods of its own. Instead, you should use the functions that work on a <gtk-recent-chooser>.

Note also that <gtk-recent-chooser-menu> does not support multiple filters, as it has no way to let the user choose between them as the <gtk-recent-chooser-widget> and <gtk-recent-chooser-dialog> widgets do. Thus using gtk-recent-chooser-add-filter on a <gtk-recent-chooser-menu> widget will yield the same effects as using gtk-recent-chooser-set-filter, replacing any currently set filter with the supplied filter; gtk-recent-chooser-remove-filter will remove any currently set <gtk-recent-filter> object and will unset the current filter; gtk-recent-chooser-list-filters will return a list containing a single <gtk-recent-filter> object.

Recently used files are supported since GTK+ 2.10.

154.2 Usage

<gtk-recent-chooser-menu> [Class]

This <gobject> class defines the following properties:

show-numbers

Whether the items should be displayed with a number

gtk-recent-chooser-menu-new ⇒ (*ret* <gtk-widget>) [Function]

Creates a new <gtk-recent-chooser-menu> widget.

This kind of widget shows the list of recently used resources as a menu, each item as a menu item. Each item inside the menu might have an icon, representing its MIME type, and a number, for mnemonic access.

This widget implements the <gtk-recent-chooser> interface.

This widget creates its own <gtk-recent-manager> object. See the gtk-recent-chooser-menu-new-for-manager function to know how to create a <gtk-recent-chooser-menu> widget bound to another <gtk-recent-manager> object.

ret a new <gtk-recent-chooser-menu>

Since 2.10

155 GtkRecentChooserWidget

Displays recently used files

155.1 Overview

<gtk-recent-chooser-widget> is a widget suitable for selecting recently used files. It is the main building block of a <gtk-recent-chooser-dialog>. Most applications will only need to use the latter; you can use <gtk-recent-chooser-widget> as part of a larger window if you have special needs.

Note that <gtk-recent-chooser-widget> does not have any methods of its own. Instead, you should use the functions that work on a <gtk-recent-chooser>.

Recently used files are supported since GTK+ 2.10.

155.2 Usage

<gtk-recent-chooser-widget> [Class]
This <gobject> class defines no properties, other than those defined by its superclasses.

gtk-recent-chooser-widget-new ⇒ (*ret* <gtk-widget>) [Function]
Creates a new <gtk-recent-chooser-widget> object. This is an embeddable widget used to access the recently used resources list.

ret a new <gtk-recent-chooser-widget>

Since 2.10

156 GtkRecentFilter

A filter for selecting a subset of recently used files

156.1 Overview

A <gtk-recent-filter> can be used to restrict the files being shown in a <gtk-recent-chooser>. Files can be filtered based on their name (with gtk-recent-filter-add-pattern), on their mime type (with gtk-file-filter-add-mime-type), on the application that has registered them (with gtk-recent-filter-add-application), or by a custom filter function (with gtk-recent-filter-add-custom).

Filtering by mime type handles aliasing and subclassing of mime types; e.g. a filter for text/plain also matches a file with mime type application/rtf, since application/rtf is a subclass of text/plain. Note that <gtk-recent-filter> allows wildcards for the subtype of a mime type, so you can e.g. filter for image/*.

Normally, filters are used by adding them to a <gtk-recent-chooser>, see gtk-recent-chooser-add-filter, but it is also possible to manually use a filter on a file with gtk-recent-filter-filter.

Recently used files are supported since GTK+ 2.10.

156.2 Usage

gtk-recent-filter-new ⇒ (*ret* <gtk-recent-filter*>) [Function]

Creates a new <gtk-recent-filter> with no rules added to it. Such filter does not accept any recently used resources, so is not particularly useful until you add rules with gtk-recent-filter-add-pattern, gtk-recent-filter-add-mime-type, gtk-recent-filter-add-application, gtk-recent-filter-add-age. To create a filter that accepts any recently used resource, use:

```
GtkRecentFilter *filter = gtk_recent_filter_new ();
gtk_recent_filter_add_pattern (filter, "*");
```

ret a new <gtk-recent-filter>

Since 2.10

gtk-recent-filter-get-name (*self* <gtk-recent-filter*>) [Function]
⇒ (*ret* mchars)

Gets the human-readable name for the filter. See gtk-recent-filter-set-name.

filter a <gtk-recent-filter>

ret the name of the filter, or '#f'. The returned string is owned by the filter object and should not be freed.

Since 2.10

gtk-recent-filter-set-name (*self* <gtk-recent-filter*>) [Function]
(*name* mchars)

Sets the human-readable name of the filter; this is the string that will be displayed in the recently used resources selector user interface if there is a selectable list of filters.

filter a `<gtk-recent-filter>`

name then human readable name of *filter*

Since 2.10

`gtk-recent-filter-add-mime-type` (*self* `<gtk-recent-filter*>`) [Function]
(*mime_type* `mchars`)

Adds a rule that allows resources based on their registered MIME type.

filter a `<gtk-recent-filter>`

mime-type
a MIME type

Since 2.10

`gtk-recent-filter-add-pattern` (*self* `<gtk-recent-filter*>`) [Function]
(*pattern* `mchars`)

Adds a rule that allows resources based on a pattern matching their display name.

filter a `<gtk-recent-filter>`

pattern a file pattern

Since 2.10

`gtk-recent-filter-add-application` (*self* `<gtk-recent-filter*>`) [Function]
(*application* `mchars`)

Adds a rule that allows resources based on the name of the application that has registered them.

filter a `<gtk-recent-filter>`

application
an application name

Since 2.10

`gtk-recent-filter-add-group` (*self* `<gtk-recent-filter*>`) [Function]
(*group* `mchars`)

Adds a rule that allows resources based on the name of the group to which they belong

filter a `<gtk-recent-filter>`

group a group name

Since 2.10

`gtk-recent-filter-add-age` (*self* `<gtk-recent-filter*>`) [Function]
(*days* `int`)

Adds a rule that allows resources based on their age - that is, the number of days elapsed since they were last modified.

filter a `<gtk-recent-filter>`

days number of days

Since 2.10

`gtk-recent-filter-add-custom` (*self* `<gtk-recent-filter*>`) [Function]
(*needed* `<gtk-recent-filter-flags>`)
(*func* `<gtk-recent-filter-func>`) (*data* `<gpointer>`)
(*data_destroy* `<g-destroy-notify>`)

Adds a rule to a filter that allows resources based on a custom callback function. The bitfield *needed* which is passed in provides information about what sorts of information that the filter function needs; this allows GTK+ to avoid retrieving expensive information when it isn't needed by the filter.

filter a `<gtk-recent-filter>`

needed bitfield of flags indicating the information that the custom filter function needs.

func callback function; if the function returns '`#t`', then the file will be displayed.

data data to pass to *func*

data-destroy
function to call to free *data* when it is no longer needed.

Since 2.10

`gtk-recent-filter-get-needed` (*self* `<gtk-recent-filter*>`) [Function]
⇒ (*ret* `<gtk-recent-filter-flags>`)

Gets the fields that need to be filled in for the structure passed to `gtk-recent-filter-filter`

This function will not typically be used by applications; it is intended principally for use in the implementation of `<gtk-recent-chooser>`.

filter a `<gtk-recent-filter>`

ret bitfield of flags indicating needed fields when calling `gtk-recent-filter-filter`

Since 2.10

`gtk-recent-filter-filter` (*self* `<gtk-recent-filter*>`) [Function]
(*filter_info* `<gtk-recent-filter-info*>`) ⇒ (*ret* `bool`)

Tests whether a file should be displayed according to *filter*. The `<gtk-recent-filter-info>` structure *filter-info* should include the fields returned from `gtk-recent-filter-get-needed`.

This function will not typically be used by applications; it is intended principally for use in the implementation of `<gtk-recent-chooser>`.

filter a `<gtk-recent-filter>`

filter-info a `<gtk-recent-filter-info>` structure containing information about a recently used resource

ret '`#t`' if the file should be displayed

Since 2.10

Concept Index

(Index is nonexistent)

Function Index

A

B

C

D

E

F

G

H

I

K

L

M

N

O

P

Q

R

S

T

U

V

W