

Package ‘RGtk2’

January 31, 2010

Version 2.12.18

Title R bindings for Gtk 2.8.0 and above

Author Michael Lawrence <michafla@gene.com> and Duncan Temple Lang
<duncan@wald.ucdavis.edu>

Depends R (>= 2.5.0)

SystemRequirements Cairo (>= 1.0.0), ATK (>= 1.10.0), Pango (>= 1.10.0), GTK+ (>= 2.8.0), GLib (>= 2.8.0)

Maintainer Michael Lawrence <michafla@gene.com>

Description Facilities in the R language for programming graphical interfaces using Gtk, the Gimp Tool Kit.

License GPL

URL <http://www.ggobi.org/rgtk2>, <http://www.omegahat.org>

Encoding UTF-8

Repository CRAN

Date/Publication 2010-01-31 10:08:17

R topics documented:

assertions	2
ATK	3
CAIRO	4
checkGTK	5
classes	6
enums-and-flags	8
GDK	9
GDK-Pixbuf	10
GMainLoop	11
GObject	12

GSignal	14
GTK	16
GType	22
Libglade	23
Pango	23
RGtk	24
RGtkDataFrame	25
RGtkObject	27
transparent-type	28

Index	29
--------------	-----------

assertions	<i>RGtk2 Type Assertion</i>
------------	-----------------------------

Description

Assert that an object is of a particular type

Usage

```
checkPtrType(w, klass = "GtkWidget", nullOk = FALSE, critical = TRUE)
implements(obj, interface)
```

Arguments

<code>w</code>	An object whose type is to be verified.
<code>klass</code>	The type the object is expected to be.
<code>nullOk</code>	Whether the object is allowed to be <code>NULL</code> .
<code>critical</code>	Whether to stop if the object is not of the specified type. If this is a character vector, then the function will stop on mismatch and report that string as the error message.
<code>obj</code>	A <code>GObject</code> .
<code>interface</code>	The interface that <code>obj</code> is expected to implement.

Details

All `RGtk2` functions check that the arguments are of the correct type, if possible. The `checkPtrType` function is most useful to the user when it is not known if an object is of the required type. A good example is the *user data* argument of a callback function. To see if a `GObject` implements a certain interface, use `implements`.

Author(s)

Michael Lawrence and Duncan Temple Lang

ATK

ATK

Description

ATK is the Accessibility Toolkit. It provides a set of generic interfaces allowing accessibility technologies to interact with a graphical user interface. For example, a screen reader uses ATK to discover the text in an interface and read it to blind users. GTK+ widgets have built-in support for accessibility using the ATK framework.

Details

The RGtk binding to the ATK library consists of the following components:

AtkAction The ATK interface provided by UI components which the user can activate/interact with,

AtkComponent The ATK interface provided by UI components which occupy a physical area on the screen.

AtkDocument The ATK interface which represents the toplevel container for document content.

AtkEditableText The ATK interface implemented by components containing user-editable text content.

AtkGObjectAccessible This object class is derived from `AtkObject` and can be used as a basis implementing accessible objects.

AtkHyperlink An ATK object which encapsulates a link or set of links in a hypertext document.

AtkHypertext The ATK interface which provides standard mechanism for manipulating hyperlinks.

AtkImage The ATK Interface implemented by components which expose image or pixmap content on-screen.

atk-AtkMisc *undocumented*

AtkNoOpObject An `AtkObject` which purports to implement all ATK interfaces.

AtkNoOpObjectFactory The `AtkObjectFactory` which creates an `AtkNoOpObject`.

AtkObject The base object class for the Accessibility Toolkit API.

AtkObjectFactory The base object class for a factory used to create accessible objects for objects of a specific `GType`.

AtkRegistry An object used to store the `GType` of the factories used to create an accessible object for an object of a particular `GType`.

AtkRelation An object used to describe a relation between a object and one or more other objects.

AtkRelationSet A set of `AtkRelations`, normally the set of `AtkRelations` which an `AtkObject` has.

AtkSelection The ATK interface implemented by container objects whose children can be selected.

atk-AtkState An `AtkState` describes a component's particular state.

atk-AtkStateSet An `AtkStateSet` determines a component's state set.

- AtkStreamableContent** The ATK interface which provides access to streamable content.
- AtkTable** The ATK interface implemented for UI components which contain tabular or row/column information.
- AtkText** The ATK interface implemented by components with text content.
- AtkUtil** A set of ATK utility functions for event and toolkit support.
- AtkValue** The ATK interface implemented by valuator and components which display or select a value from a bounded range of values.

Author(s)

Derived by RGtkGen from GTK+ documentation

References

<http://developer.gnome.org/doc/API/2.0/atk>

 CAIRO

 CAIRO

Description

Cairo is a 2D graphics library with support for multiple output devices. Currently supported output targets include the X Window System, win32, and image buffers.

Details

The RGtk binding to the CAIRO library consists of the following components:

- cairo-cairo-font-face-t** Base class for font faces
- cairo-Font-Options** How a font should be rendered
- cairo-cairo-font-face-t** Base class for fonts
- cairo-Image-Surfaces** Rendering to memory buffers
- cairo-cairo-matrix-t** Generic matrix operations
- cairo-Paths** Creating paths and manipulating path data
- cairo-Patterns** Sources for drawing
- cairo-PDF-Surfaces** Rendering PDF documents
- cairo-PNG-Support** Reading and writing PNG images
- cairo-PostScript-Surfaces** Rendering PostScript documents
- cairo-Scaled-Fonts** Font face at particular size and options
- cairo-Error-Handling** Decoding cairo's status
- cairo-cairo-surface-t** Base class for surfaces
- cairo-SVG-Surfaces** Rendering SVG documents

- cairo-Text** Rendering text and glyphs
- cairo-Transformations** Manipulating the current transformation matrix
- cairo-Types** Generic data types
- cairo-Version-Information** Compile-time and run-time version checks.
- cairo-cairo-t** The cairo drawing context

Author(s)

Derived by RGtkGen from GTK+ documentation

References

<http://www.cairographics.org/manual>

checkGTK

Bound versions

Description

These functions are for querying (*bound**) and checking (*check**) the bound versions of the libraries (GTK, Pango and Cairo).

Usage

```
checkGTK (version)
checkPango (version)
checkCairo (version)
boundGTKVersion ()
boundPangoVersion ()
boundCairoVersion ()
```

Arguments

`version` Version description to compare to the bound version, as in: `compareVersion (boundGTKVersion (version))`.

Value

The *check** functions return the result of the comparison as an integer, as from `compareVersion`.

The *bound** functions return a character vector representation of the bound library version.

Author(s)

Michael Lawrence

See Also

[compareVersion](#)

Examples

```
checkGTK("2.12.0")
# same as
compareVersion(boundGTKVersion(), "2.12.0")
```

classes

Custom GObject classes

Description

Highly experimental support for constructing new `GObject` classes entirely from with R.

Usage

```
gClass(name, parent = "GObject", ..., abstract = FALSE)
parentHandler(method, obj = NULL, ...)
assignProp(obj, pspec, value)
getProp(obj, pspec)
registerVirtuals(virtuals)
unregisterVirtuals(virtuals)
```

Arguments

name	The name of the new class
parent	The name of the parent class
abstract	If TRUE, the class should not be instantiable.
method	The name of the method to invoke in the parent
obj	A GObject
...	Additional arguments. For <code>parentHandler()</code> , arguments to pass to the parent method. For <code>gClass()</code> , arguments specifying the class definition (see Details).
pspec	A GParamSpec describing the property
value	The value to set on the property
virtuals	An environment containing lists where each list contains the names of the virtual methods for the class matching the name of the list.

Details

The bulk of the class definition (everything except the name and the parent) is passed through additional arguments to the `gClass` function. This information includes:

Methods R functions that override virtual methods in a `GObject` class. Functions overriding methods in the same class are grouped together in a list and are named according to the virtual they override. Each list is passed as a separate parameter to the `class_def` list and bears the name of the corresponding class.

Signals Signals that are emitted by the class, in addition to those of the superclasses. Each signal definition is a list containing the following elements: signal name, vector of type names of signal arguments, type name of signal return value, and a vector of values from the `GSignalFlags` enumeration. The list of signal definitions is passed as a parameter named `.signals` to the `gClass`.

Properties Properties defined by the class. This is a list of lists, each corresponding to a `GParamSpec`, as created by `gParamSpec`. The list is passed under the name `.props` to `gClass`. The property values are stored in a private environment. To override that behavior or to be notified (first) upon property changes, simply override the `set_property` and `get_property` virtuals in the `GObject` class. To override the implementation of properties defined by an ancestor class, specify their names in a separate vector passed as the `.prop_overrides` parameter. If you override the setting or getting of properties, you can use `assignProp` or `getProp` to conveniently directly assign or get the value of a property to or from the low-level data structure, respectively. These functions differ from the normal property accessor mechanism in that they bypass the property system, thus avoiding recursion. They should only be used when overriding property handling.

Initializer Upon instance creation, the function named `.initialize` (in the parameters passed to `gClass`) will be called with the instance as the only argument.

New members It is possible to define new public, protected, and private fields and methods inside an R class, by passing them to `gClass` within lists named `.public`, `.protected`, or `.private`, respectively. The encapsulation works much the same as Java. Any protected and public functions may be overridden in a class derived from the defining class. All public fields are immutable. All function bindings are locked except for private ones. This means private functions can be replaced.

The above may seem complicated, and it is. Please see the `alphaSliderClass` for an example. Also note that the `local` function is convenient for defining static namespaces on the fly. For calling parent virtuals, use `parentHandler`.

`assignProp` and `getProp` are low-level functions; they should not be used in place of the conventional `GObject` property mechanism, except in the case mentioned above.

`registerVirtuals` and `unregisterVirtuals` are meant for use by packages that bind C `GObject` classes to R using the `RGtk2` system. An example of such a package is `rgobi`.

Value

For `gClass`, the `GType` of the new class. For `getProp`, the value of the property.

Note

This functionality is not for casual users. If you don't know what you're doing you will break things. Otherwise, have fun.

Author(s)

Michael Lawrence

enums-and-flags *Enums and Flags*

Description

Convenience functions and operators for operating on bitflags and enums

Usage

```
as.flag(x)
[.flags(x, value)
|.flag(x, y)
&.flag(x, y)
!.flag(x)
==.enum(x, y)
```

Arguments

x	Numeric value to coerce to a <code>flag</code> , an object of class <code>flags</code> , or the left hand operand
y	Right hand operand
value	The character id or index for a particular flag in a <code>flags</code> vector

Details

The libraries bound by RGtk2 often return numeric values that are either bitflags or enumerations. In order to facilitate operations on these types (especially bitflags), several methods have been defined corresponding to conventional operators for performing bitwise operations and comparisons.

RGtk2 defines all of the enum and flag types from the API's as vectors of class `flags` or `enums` with their names corresponding to the nicknames of the values. The `[` operator on the `flags` class retrieves a value as a `flag`. This is only necessary for the bitwise ops and thus is not necessary for `enums`.

The `==.enum` method compares a `enum` with either a character or numeric representation of an enum value.

Value

A `flag` for `as.flag`, `[.flags`, and the bitwise operators. A logical value for `==.enum`.

Note

Sometimes the API does not return a value specifically as a `flag`. In this case, it is a generic numeric value and should be coerced with `as .flag`.

Author(s)

Michael Lawrence

 GDK

 GDK

Description

GDK is the abstraction layer that allows GTK+ to support multiple windowing systems. GDK provides drawing and window system facilities on X11, Windows, and the Linux framebuffer device.

Details

The RGtk binding to the GDK library consists of the following components:

- gdk-Cairo-Interaction** Functions to support using Cairo
- gdk-Colormaps-and-Colors** Manipulation of colors and colormaps
- gdk-Cursors** Standard and pixmap cursors
- gdk-Drag-and-Drop** Functions for controlling drag and drop handling
- gdk-Drawing-Primitives** Functions for drawing points, lines, arcs, and text
- gdk-Event-Structures** Data structures specific to each type of event
- gdk-Events** Functions for handling events from the window system
- gdk-Fonts** Loading and manipulating fonts
- gdk-Graphics-Contexts** Objects to encapsulate drawing properties
- GdkDisplay** Controls the keyboard/mouse pointer grabs and a set of s
- GdkDisplayManager** Maintains a list of all open s
- GdkScreen** Object representing a physical screen
- gdk-General** Library initialization and miscellaneous functions
- gdk-Images** A client-side area for bit-mapped graphics
- gdk-Input-Devices** Functions for handling extended input devices
- gdk-Keyboard-Handling** Functions for manipulating keyboard codes
- gdk-Pango-Interaction** Using Pango in GDK
- gdk-Pixbufs** Functions for rendering pixbufs on drawables
- gdk-Bitmaps-and-Pixmaps** Offscreen drawables
- gdk-Properties-and-Atoms** Functions to manipulate properties on windows
- gdk-Points-Rectangles-and-Regions** Simple graphical data types
- gdk-GdkRGB** Renders RGB, grayscale, or indexed image data to a GdkDrawable
- gdk-Visuals** Low-level display hardware information
- gdk-Windows** Onscreen display areas in the target window system

Author(s)

Derived by RGtkGen from GTK+ documentation

References

<http://developer.gnome.org/doc/API/2.0/gdk>

GDK-Pixbuf

GDK-Pixbuf

Description

This is a small library which allows you to create `GdkPixbuf` ('pixel buffer') objects from image data or image files. Use a `GdkPixbuf` in combination with `GtkImage` to display images.

Details

The RGtk binding to the GDK-Pixbuf library consists of the following components:

`gdk-pixbuf-animation` Animated images.

`gdk-pixbuf-creating` Creating a pixbuf from image data that is already in memory.

`gdk-pixbuf-file-loading` Loading a pixbuf from a file.

`gdk-pixbuf-file-saving` Saving a pixbuf to a file.

`GdkPixbufLoader` Application-driven progressive image loading.

`gdk-pixbuf-gdk-pixbuf` Information that describes an image.

`gdk-pixbuf-Versioning` Library version numbers.

`gdk-pixbuf-Module-Interface` Extending

`gdk-pixbuf-scaling` Scaling pixbufs and scaling and compositing pixbufs

`gdk-pixbuf-util` Utility and miscellaneous convenience functions.

Author(s)

Derived by RGtkGen from GTK+ documentation

References

<http://developer.gnome.org/doc/API/2.0/gdk-pixbuf>

Description

GLib provides an event-loop to all GLib-based libraries and applications. RGtk2 is one such library.

Usage

```
gTimeoutAdd(interval, f, data = NULL)
gIdleAdd(f, data = NULL)
gSourceRemove(id)
```

Arguments

<code>interval</code>	The time interval which determines the frequency of the handler call
<code>f</code>	An R function that is called by the loop
<code>data</code>	Any R object that is passed to the R function as the last parameter
<code>id</code>	The source id obtained when adding a handler

Details

The RGtk2 user has limited control over the event loop, but it is still possible to register handlers as either timeout or idle tasks. A handler may be any R function, though it must return `TRUE` as long as it wants to stay connected to the loop.

Timeout tasks are performed once per some specified interval of time. Use `gTimeoutAdd` to register such a handler.

When the event loop is idle (not busy) it will execute the idle handlers, which may be registered with `gIdleAdd`.

If one needs to externally remove a handler from the loop, `gSourceRemove` will serve this purpose.

Value

`gIdleAdd` and `gTimeoutAdd` both return a source id that may be used to remove the handler later.

Author(s)

Michael Lawrence

References

<http://developer.gnome.org/doc/API/2.0/glib/glib-The-Main-Event-Loop.html>

Description

GObject is the fundamental type providing the common attributes and methods for all object types in GTK+, Pango and other libraries based on GObject. It provides facilities for object construction, properties, and signals.

Usage

```
gObjectGet(obj, ..., drop = T)
"[.GObject"(obj, value, ...)
gObjectSet(obj, ...)
"[<-.GObject"(obj, propName, value)
"[[.GObject"(obj, member, where = parent.frame())
"[[<-.GObject"(obj, member, value)
"$GObject"(x, member)
"$<-.GObject"(obj, member, value)
gObject(type, ...)
gObjectNew(type, ...)
gObjectSetData(obj, key, data = NULL)
gObjectGetData(obj, key)
gObjectGetSignals(obj)
gObjectGetPropInfo(obj, parents = TRUE, collapse = TRUE)
gTypeGetPropInfo(type)
names.GObject(x)
interface.GObject(obj)
gObjectParentClass(obj)
```

Arguments

obj	an instance of a GObject
drop	when retrieving the value of a single property, TRUE to return the element from the list, instead of the list with just that element.
member	the name of a member in an R-defined (custom) GObject class
type	the type of GObject
key	the unique identifier under which the data is stored
data	the data to store with the GObject
...	named arguments of properties to set or names of properties to retrieve
propNames	the names properties to set or get
value	a value with which to set a property
parents	whether to include the parents when retrieving property info
collapse	whether to collapse the properties over the parents
x	The GObject for which the property names are to be retrieved
where	The environment in which to look for the field accessor function

Details

Every `GObject` has a type, known as a `GType`. Like all object-oriented paradigms, types may be (in this case singly) inherited. Thus, every `GObject` has a type that descends from the common `GObject` type. `GObjects` may also implement interfaces. The interfaces implemented by a particular object may be found in the `interfaces` attribute of an R object representing a `GObject`, for which, as you might expect, `inherits("GObject")` returns `TRUE`. To conveniently access this attribute, use `interface.GObject`.

A `GObject` is usually constructed with the constructor belonging to a particular subtype (for example, `gtkWindowNew` constructs a `GtkWindow`). It is also possible to use `gObjectNew` to construct an instance of `GObject` with the given type and properties.

The properties of a `GObject` are name-value pairs that may be retrieved and set using `gObjectGet` and `gObjectSet`, respectively. Whenever specifying properties as arguments in `RGtk2`, name the arguments with the property name and give the desired property value as the actual argument. For example, `gObjectSet(window, modal = T)` to make a window modal. For convenience, the `[.GObject` and `[<-.GObject` functions may be used to get and set properties, respectively. For example, `window["modal"] <- T`. Properties help describe the state of an object and are convenient for many reasons, including the ability to register handlers that are invoked when a property changes. They are also associated with metadata that describe their purpose and allow runtime checking of constraints, such as the data type or range in the case of a numeric type.

This notification occurs via `GObject` signals, which are named hooks for which callbacks may be registered. The event driven system of `GTK+` depends on signals for coordinating objects in response to both user and programmatic events. You can use `gSignalConnect` to connect an R function to a signal.

When new `GObject` classes are defined in R, they may provide additional fields and methods. `[.GObject` and `[<-.GObject` get and set, respectively, those members, depending on permissions: private members are only available to methods of the defining class, and protected only to subclasses of the defining class. If `[` fails to find an R-defined member, it searches for a C field and then a `GObject` property. `[<-` first tries to set a `GObject` property before looking for an R member to ensure that properties are set through the proper channel. Note that the bindings of public fields and public and protected methods are locked, so they cannot be changed using `[<-`. `$.GObject` serves as a synonym of `[<-.GObject`, but `$.GObject` first checks for a function (see `$.RGtkObject`) before falling back to the behavior of `[.GObject`.

Finally, arbitrary R objects can be stored in a `GObject` under a specific key for later retrieval. This can be achieved with `gObjectSetData` and `gObjectGetData`, respectively. This is similar to attributes in R, with a major difference being that changes occur in the external `GObject`, transcending the local R object.

`GObjects` also offer some introspection capabilities. `gObjectGetPropInfo` and `gObjectGetSignals` provide a list of supported properties and signals, respectively. `names.GObject` lists the available properties for an object. It is hoped that in the future methods and fields may also be introspected.

Value

Properties and data for `gObjectGet` and `gObjectGetData`, respectively. `gObjectNew` returns a new instance of the specified type. `gObjectGetPropInfo` and `gTypeGetPropInfo`

return a named list (or list of lists if `collapse` is `FALSE`) of properties (`GParamSpecs`) belonging to the `GObject` type and its parents (unless `parents` is `FALSE`). `gObjectGetSignals` gets a list of signal ids with names for the signals supported by the object. `gObjectParentClass` returns a pointer to the parent class of the object.

Author(s)

Michael Lawrence

References

<http://developer.gnome.org/doc/API/2.0/gobject/gobject-The-Base-Object-Type.html>

See Also

[GType GSignal](#)

GSignal

The GSignal API

Description

The basic concept of the signal system is that of the emission of a signal. Signals are introduced per-type and are identified through strings. Signals introduced for a parent type are available in derived types as well, so basically they are a per-type facility that is inherited.

Usage

```
gSignalConnect(obj, signal, f, data = NULL, after = FALSE, user.data.first = FALSE)
gSignalHandlerDisconnect(obj, id)
gSignalHandlerBlock(obj, id)
gSignalHandlerUnblock(obj, id)
gSignalEmit(obj, signal, ..., detail = NULL)
gSignalStopEmission(obj, signal, detail = NULL)
gSignalGetInfo(sig)
```

Arguments

<code>obj</code>	The object that owns the signal
<code>signal</code>	The detailed name of the signal
<code>f</code>	The R function to connect as a callback
<code>data</code>	Arbitrary "user data" that will be passed to the callback <code>f</code>
<code>after</code>	Whether <code>f</code> will be called before or after the default handler
<code>user.data.first</code>	Whether the <code>data</code> is the first or last argument to the callback

<code>id</code>	The signal handler id obtained upon connection to the signal
<code>...</code>	Arguments to pass to the signal handlers
<code>detail</code>	Optional separate argument for the <i>detail</i> portion of the signal
<code>sig</code>	A signal id provided by <code>gObjectGetSignals</code> .

Details

A signal emission mainly involves invocation of a certain set of callbacks in precisely defined manner. There are two main categories of such callbacks, per-object ones and user provided ones. The per-object callbacks are most often referred to as "object method handler" or "default (signal) handler", while user provided callbacks are usually just called "signal handler". The object method handler is provided at signal creation time (this most frequently happens at the end of an object class' creation), while user provided handlers are frequently connected and disconnected to/from a certain signal on certain object instances.

A signal emission consists of five stages, unless prematurely stopped:

1. Invocation of the object method handler for `G_SIGNAL_RUN_FIRST` signals
2. Invocation of normal user-provided signal handlers (after flag `FALSE`)
3. Invocation of the object method handler for `G_SIGNAL_RUN_LAST` signals
4. Invocation of user provided signal handlers, connected with an `after` flag of `TRUE`
5. Invocation of the object method handler for `G_SIGNAL_RUN_CLEANUP` signals

The user-provided signal handlers are called in the order they were connected in. All handlers may prematurely stop a signal emission, and any number of handlers may be connected, disconnected, blocked or unblocked during a signal emission. There are certain criteria for skipping user handlers in stages 2 and 4 of a signal emission. First, user handlers may be blocked, blocked handlers are omitted during callback invocation, to return from the "blocked" state, a handler has to get unblocked exactly the same amount of times it has been blocked before. Second, upon emission of a `G_SIGNAL_DETAILED` signal, an additional "detail" argument passed in to `gSignalEmit` has to match the detail argument of the signal handler currently subject to invocation. Specification of no detail argument for signal handlers (omission of the detail part of the signal specification upon connection) serves as a wildcard and matches any detail argument passed in to emission.

Most of the time, the RGtk2 user will be connecting to signals using `gSignalConnect`. This attaches an R function (and, optionally, some arbitrary "user data") to a specific `GObject` as a listener to the named signal.

`gSignalHandlerBlock` and `gSignalHandlerUnblock` provide facilities for (temporarily) blocking and unblocking the calling of an R function in response to some signal. To permanently disconnect the handler from the object and signal, use `gSignalHandlerDisconnect`.

A signal may be manually emitted with `gSignalEmit`. The emission of a signal may be killed prematurely with `gSignalStopEmission`.

Detailed information about a signal may be introspected with `gSignalGetInfo` using ids obtained with `gObjectGetSignals`.

Value

`gSignalConnect` returns a numeric id for the signal handler. It is used for blocking and disconnecting the handler.

`gSignalGetInfo` returns detailed information about a signal. The returned list contains the following elements:

<code>returnType</code>	The return GType id of the signal
<code>signal</code>	The signal id
<code>parameters</code>	A list of GType ids for the parameters
<code>objectType</code>	The GType id owning the signal
<code>runFlags</code>	The flags determining behavior of the signal, see reference

Author(s)

Adapted from GSignal documentation by Michael Lawrence

References

<http://developer.gnome.org/doc/API/2.0/gobject/gobject-Signals.html>

See Also

[GObject](#)

 GTK

 GTK

Description

The GTK+ library itself contains widgets, that is, GUI components such as [GtkButton](#) or [GtkTextView](#).

Details

The RGtk binding to the GTK library consists of the following components:

- GtkAboutDialog** Display information about an application
- gtk-Keybaord-Accelerators** Groups of global keyboard accelerators for an entire [GtkWindow](#)
- GtkAccelLabel** A label which displays an accelerator key on the right of the text
- gtk-Accelerator-Maps** Loadable keyboard accelerator specifications
- GtkAccessible** Accessibility support for widgets
- GtkAction** An action which can be triggered by a menu or toolbar item
- GtkActionGroup** A group of actions
- GtkAdjustment** A [GtkObject](#) representing an adjustable bounded value
- GtkAlignment** A widget which controls the alignment and size of its child

GtkArrow Displays an arrow

GtkAspectFrame A frame that constrains its child to a particular aspect ratio

GtkAssistant A widget used to guide users through multi-step operations

GtkButtonBox Base class for GtkHButtonBox and GtkVButtonBox

GtkBin A container with just one child

GtkBox Base class for box containers

gtk-gtkbuildable Interface for objects that can be built by GtkBuilder

GtkBuilder Build an interface from an XML UI definition

GtkButton A widget that creates a signal when clicked on

GtkCalendar Displays a calendar and allows the user to select a date

GtkCellEditable Interface for widgets which can be used for editing cells

GtkCellLayout An interface for packing cells

GtkCellRenderer An object for rendering a single cell on a GdkDrawable

GtkCellRendererAccel Renders a keyboard accelerator in a cell

GtkCellRendererCombo Renders a combobox in a cell

GtkCellRendererPixbuf Renders a pixbuf in a cell

GtkCellRendererProgress Renders numbers as progress bars

GtkCellRendererSpin Renders a spin button in a cell

GtkCellRendererText Renders text in a cell

GtkCellRendererToggle Renders a toggle button in a cell

GtkCellView A widget displaying a single row of a GtkTreeModel

GtkCheckButton Create widgets with a discrete toggle button

GtkCheckMenuItem A menu item with a check box

gtk-Clipboards Storing data on clipboards

GtkCList A multi-columned scrolling list widget

GtkColorButton A button to launch a color selection dialog

GtkColorSelection A widget used to select a color

GtkColorSelectionDialog A standard dialog box for selecting a color

GtkCombo A text entry field with a dropdown list

GtkComboBox A widget used to choose from a list of items

GtkComboBoxEntry A text entry field with a dropdown list

GtkContainer Base class for widgets which contain other widgets

GtkCTree A widget displaying a hierarchical tree

GtkCurve Allows direct editing of a curve

GtkDialog Create popup windows

gtk-Drag-and-Drop Functions for controlling drag and drop handling

GtkDrawingArea A widget for custom user interface elements

GtkEditable Interface for text-editing widgets

GtkEntry A single line text entry field

GtkEntryCompletion Completion functionality for GtkEntry

gtk-Standard-Enumerations Public enumerated types used throughout GTK+

GtkEventBox A widget used to catch events for widgets which do not have their own window

GtkExpander A container which can hide its child

GtkFileChooser File chooser interface used by GtkFileChooserWidget and GtkFileChooserDialog

GtkFileChooserButton A button to launch a file selection dialog

GtkFileChooserDialog A file chooser dialog, suitable for "File/Open" or "File/Save" commands

GtkFileChooserWidget File chooser widget that can be embedded in other widgets

gtk-gtkfilefilter A filter for selecting a file subset

GtkFileSelection Prompt the user for a file or directory name

GtkFixed A container which allows you to position widgets at fixed coordinates

GtkFontButton A button to launch a font selection dialog

GtkFontSelection A widget for selecting fonts

GtkFontSelectionDialog A dialog box for selecting fonts

GtkFrame A bin with a decorative frame and optional label

GtkGammaCurve A subclass of GtkCurve for editing gamma curves

gtk-Graphics-Contexts A shared pool of GdkGC objects

GtkHandleBox a widget for detachable window portions

GtkHButtonBox A container for arranging buttons horizontally

GtkHBox A horizontal container box

GtkHPaned A container with two panes arranged horizontally

GtkHRuler A horizontal ruler

GtkHScale A horizontal slider widget for selecting a value from a range

GtkHScrollbar A horizontal scrollbar

GtkHSeparator A horizontal separator

gtk-Themeable-Stock-Images Manipulating stock icons

GtkIconTheme Looking up icons by name

GtkIconView A widget which displays a list of icons in a grid

GtkImage A widget displaying an image

GtkImageMenuItem A menu item with an icon

GtkIMContext Base class for input method contexts

GtkIMContextSimple An input method context supporting table-based input methods

GtkIMMulticontext An input method context supporting multiple, loadable input methods

GtkInputDialog Configure devices for the XInput extension

GtkInvisible A widget which is not displayed

GtkItem Abstract base class for GtkMenuItem, GtkListItem and GtkTreeItem

GtkItemFactory A factory for menus

GtkLabel A widget that displays a small to medium amount of text

GtkLayout Infinite scrollable area containing child widgets and/or custom drawing

GtkLinkButton Create buttons bound to a URL

GtkList Widget for packing a list of selectable items

GtkListItem An item in a GtkList

GtkListStore A list-like data structure that can be used with the GtkTreeView

gtk-General Library initialization, main event loop, and events

GtkMenu A menu widget

GtkMenuBar A subclass widget for GtkMenuShell which holds GtkMenuItem widgets

GtkMenuItem The widget used for item in menus

GtkMenuShell A base class for menu objects

GtkMenuToolButton A GtkToolItem containing a button with an additional dropdown menu

GtkMessageDialog A convenient message window

GtkMisc Base class for widgets with alignments and padding

GtkNotebook A tabbed notebook container

GtkOldEditable Base class for text-editing widgets

GtkOptionMenu A widget used to choose from a list of valid choices

GtkPageSetup Stores page setup information

GtkPaned Base class for widgets with two adjustable panes

gtk-GtkPaperSize Support for named paper sizes

GtkPixmap A widget displaying a graphical image or icon

GtkPlug Toplevel for embedding into other processes

GtkPreview A widget to display RGB or grayscale data

GtkPrintContext Encapsulates context for drawing pages

gtk-High-level-Printing-API High-level Printing API

GtkPrintSettings Stores print settings

GtkProgress Base class for GtkProgressBar

GtkProgressBar A widget which indicates progress visually

GtkRadioAction An action of which only one in a group can be active

GtkRadioButton A choice from multiple check buttons

GtkRadioMenuItem A choice from multiple check menu items

GtkRadioToolButton A toolbar item that contains a radio button

GtkRange Base class for widgets which visualize an adjustment

gtk-Resource-Files Routines for handling resource files

GtkRecentAction An action of which represents a list of recently used files

GtkRecentChooser Interface implemented by widgets displaying recently used files

GtkRecentChooserDialog Displays recently used files in a dialog

GtkRecentChooserMenu Displays recently used files in a menu

GtkRecentChooserWidget Displays recently used files

GtkRecentFilter A filter for selecting a subset of recently used files

GtkRecentManager Managing Recently Used Files

GtkRuler Base class for horizontal or vertical rulers

GtkScale Base class for GtkHScale and GtkVScale

GtkScaleButton A button which pops up a scale

GtkScrollbar Base class for GtkHScrollbar and GtkVScrollbar

GtkScrolledWindow Adds scrollbars to its child widget

gtk-Selections Functions for handling inter-process communication via selections

GtkSeparator Base class for GtkHSeparator and GtkVSeparator

GtkSeparatorMenuItem A separator used in menus

GtkSeparatorToolItem A toolbar item that separates groups of other toolbar items

GtkSettings Sharing settings between applications

GtkSizeGroup Grouping widgets so they request the same size

GtkSocket Container for widgets from other processes

GtkSpinButton Retrieve an integer or floating-point number from the user

GtkStatusbar Report messages of minor importance to the user

GtkStatusIcon Display an icon in the system tray

gtk-Stock-Items Prebuilt common menu/toolbar items and corresponding icons

GtkStyle Functions for drawing widget parts

GtkTable Pack widgets in regular patterns

GtkTearoffMenuItem A menu item used to tear off and reattach its menu

GtkTextBuffer Stores attributed text for display in a GtkTextView

gtk-GtkTextIter Text buffer iterator

GtkTextMark A position in the buffer preserved across buffer modifications

GtkTextTag A tag that can be applied to text in a GtkTextBuffer

GtkTextTagTable Collection of tags that can be used together

GtkTextView Widget that displays a GtkTextBuffer

GtkTipsQuery Displays help about widgets in the user interface

GtkToggleAction An action which can be toggled between two states

GtkToggleButton Create buttons which retain their state

GtkToggleToolButton A GtkToolItem containing a toggle button

GtkToolbar Create bars of buttons and other widgets

GtkToolButton A GtkToolItem subclass that displays buttons

- GtkToolItem** The base class of widgets that can be added to GtkToolbar
- GtkTooltip** Add tips to your widgets
- GtkTooltips** Add tips to your widgets
- gtk-GtkTreeView-drag-and-drop** Interfaces for drag-and-drop support in GtkTreeView
- GtkTreeModel** The tree interface used by GtkTreeView
- GtkTreeModelFilter** A GtkTreeModel which hides parts of an underlying tree model
- GtkTreeModelSort** A GtkTreeModel which makes an underlying tree model sortable
- GtkTreeSelection** The selection object for GtkTreeView
- GtkTreeSortable** The interface for sortable models used by GtkTreeView
- GtkTreeStore** A tree-like data structure that can be used with the GtkTreeView
- GtkTreeView** A widget for displaying both trees and lists
- GtkTreeViewColumn** A visible column in a GtkTreeView widget
- GtkUIManager** Constructing menus and toolbars from an XML description
- GtkVButtonBox** A container for arranging buttons vertically
- GtkVBox** A vertical container box
- GtkViewPort** An adapter which makes widgets scrollable
- GtkVolumeButton** A button which pops up a volume control
- GtkVPaned** A container with two panes arranged vertically
- GtkVRuler** A vertical ruler
- GtkVScale** A vertical slider widget for selecting a value from a range
- GtkVScrollbar** *undocumented*
- GtkVSeparator** A vertical separator
- GtkWidget** Base class for all widgets
- GtkWindow** Toplevel which can contain other widgets
- GtkWindowGroup** Limit the effect of grabs

Author(s)

Derived by RGtkGen from GTK+ documentation

References

<http://developer.gnome.org/doc/API/2.0/gtk>

GType

*The GType system***Description**

"The GType API is the foundation of the GObject system. It provides the facilities for registering and managing all fundamental data types, user-defined object and interface types." - GObject documentation

Usage

```
gTypeGetAncestors (type)
gTypeGetInterfaces (type)
gTypeFromName (name)
gTypeGetClass (type)
```

Arguments

<code>type</code>	The GType, either its name or numeric value, see below
<code>name</code>	The name of a GType

Details

The GType system supports inheritance and interfaces, enabling the psuedo-object-oriented system known as [GObject](#). However, they also encompass all fundamental (primitive) types.

A GType is considered a [transparent-type](#) in RGtk2, since you may specify one as either the type name or the numeric value retrieved from some API function like `gTypeFromName`. The GType system obviously names primitive types different from the corresponding types in R, but this is automatically taken care of for you, so you can use R type names (ie, "character", "logical", etc) when specifying a GType. This means that `gTypeFromName` is not that useful to the RGtk2 programmer.

All R objects representing external RGtk2 objects have their hierarchy stored in the `class` attribute. Everything descends from "RGtkObject", then, for example, "GObject", etc. The types do not necessarily correspond to GTypes, but they do for all `GObjects` and others. Thus, `gTypeGetAncestors` is also of little use unless one is working with pure GTypes.

Value

`gTypeGetAncestors` returns a vector of type names from which `type` inherits. `gTypeGetInterfaces` names the interfaces implemented by `type`. `gTypeFromName` retrieves the numeric value of a type from its name. `gTypeGetClass` returns the class instance for the type, for example `GtkWidgetClass`.

Author(s)

Michael Lawrence

References

<http://developer.gnome.org/doc/API/2.0/gobject/gobject-Type-Information.html>

See Also

[GObject](#)

Libglade

Libglade

Description

Libglade loads and parses XML descriptions of user interfaces at runtime. It also provides functions that can be used to connect signal handlers to parts of the interface.

Details

The RGtk binding to the Libglade library consists of the following components:

GladeXML Allows dynamic loading of user interfaces from XML descriptions.

Author(s)

Derived by RGtkGen from GTK+ documentation

References

<http://developer.gnome.org/doc/API/2.0/libglade>

Pango

Pango

Description

Pango is a library for internationalized text handling. It centers around the [PangoLayout](#) object, representing a paragraph of text. Pango provides the engine for [GtkTextView](#), [GtkLabel](#), [GtkEntry](#), and other widgets that display text.

Details

The RGtk binding to the Pango library consists of the following components:

- pango-Coverage-Maps** Unicode character range coverage storage
- pango-Fonts** Structures representing abstract fonts
- pango-Glyph-Storage** Structures for storing information about glyphs
- pango-Layout-Objects** High-level layout driver objects
- pango-Text-Processing** Functions to run the rendering pipeline
- PangoRenderer** Rendering driver base class
- pango-Version-Checking** Tools for checking Pango version at compile- and run-time.
- pango-Cairo-Rendering** Rendering with the Cairo backend
- pango-Scripts** Identifying writing systems
- pango-Tab-Stops** Structures for storing tab stops
- pango-Text-Attributes** Font and other attributes for annotating text
- pango-Vertical-Text** Laying text out in vertical directions

Author(s)

Derived by RGtkGen from GTK+ documentation

References

<http://developer.gnome.org/doc/API/2.0/pango>

RGtk

The RGtk2 package

Description

RGtk2 provides a set of bindings between R and the GTK+ library and several of its dependent libraries. It allows the user to construct full-featured GUI's completely from within R.

Details

RGtk2 binds to the following libraries:

- ATK** ATK is the Accessibility Toolkit. It provides a set of generic interfaces allowing accessibility technologies to interact with a graphical user interface. For example, a screen reader uses ATK to discover the text in an interface and read it to blind users. GTK+ widgets have built-in support for accessibility using the ATK framework.
- Pango** Pango is a library for internationalized text handling. It centers around the `PangoLayout` object, representing a paragraph of text. Pango provides the engine for `GtkTextView`, `GtkLabel`, `GtkEntry`, and other widgets that display text.

GDK GDK is the abstraction layer that allows GTK+ to support multiple windowing systems. GDK provides drawing and window system facilities on X11, Windows, and the Linux framebuffer device.

GTK The GTK+ library itself contains widgets, that is, GUI components such as `GtkButton` or `GtkTextView`.

GDK-Pixbuf This is a small library which allows you to create `GdkPixbuf` ('pixel buffer') objects from image data or image files. Use a `GdkPixbuf` in combination with `GtkImage` to display images.

Cairo Cairo is a 2D graphics library with support for multiple output devices. Currently supported output targets include the X Window System, win32, and image buffers.

Libglade Libglade loads and parses XML descriptions of user interfaces at runtime. It also provides functions that can be used to connect signal handlers to parts of the interface.

RGtk2 also partially binds some lower-level libraries in order to support the bindings to the others. These include `GObject` and `GMainLoop`.

R objects passed between the user and RGtk2 are either primitive types (`character`, `logical`, etc) or external objects (`externalptr`). All R objects wrapping external objects extend the `RGtkObject` class.

Note

As described above, RGtk2 binds many libraries beyond GTK+ itself. Thus, it can serve many purposes besides GUI construction. For example, `GDKPixbuf` and `Cairo` allow the R user to produce arbitrary high-quality graphics.

Author(s)

Michael Lawrence, with excerpts from library documentation

RGtkDataFrame

The RGtkDataFrame model

Description

A `GtkTreeModel` implementation backed by an R data frame

Usage

```
rGtkDataFrame(frame = data.frame())
rGtkDataFrameNew(frame = data.frame())
rGtkDataFrameAppendColumns(x, ...)
rGtkDataFrameAppendRows(x, ...)
as.data.frame.RGtkDataFrame(x, ...)
rGtkDataFrameSetFrame(x, frame = data.frame())
"[.RGtkDataFrame"(x, i, j, drop = T)
"[<- .RGtkDataFrame"(x, i, j, value)
```

```
dim.RGtkDataFrame(x, ...)
dimnames.RGtkDataFrame(x, ...)
"dimnames<-RGtkDataFrame"(x, value)
```

Arguments

<code>frame</code>	The frame to use as the backing store of the model
<code>x</code>	An <code>RGtkDataFrame</code> object
<code>i</code>	Row index
<code>j</code>	Column index
<code>value</code>	An R object similar to that accepted by <code>[<-data.frame</code> or the <code>dimnames</code> for the data frame
<code>drop</code>	Whether to 'drop' the result to the simplest structure
<code>...</code>	Items to append as columns or rows or just additional arguments

Details

The `RGtk2` interface carries a lot of overhead, slowing down operations that require large numbers of function calls, such as loading a `GtkTreeModel`. Under the assumption that R programmers will store large datasets as data frames, a new `GtkTreeModel` was implemented that draws data directly from an R data frame. This offers not only a dramatic performance gain but also allows efficient addition of columns to a model, which the default GTK implementations do not allow.

The `RGtkDataFrame` is constructed with a delegate data frame, which can be empty, via either `rGtkDataFrameNew` or `rGtkDataFrame` for short. The subset and replacement methods work much the same as for normal data frames, except one should note that removing columns (ie by replacing columns with `NULLs`) is not supported. Note that even if the initial data frame is empty, one should ensure that the empty vectors representing the column are of the desired types. If one wants to simply replace the backing frame with a new one, then there are two options: create a new `RGtkDataFrame` and connect it to the views of the old model, or use `rGtkDataFrameSetFrame`.

The `rGtkDataFrameAppendColumns` and `rGtkDataFrameAppendRows` methods allow appending columns and rows, respectively. Note that these are a lot shorter if using the `object$appendColumns(...)` syntax.

The `as.data.frame` method retrieves the backing data frame from the model, so that one can perform any data frame operation on the data. Of course, any changes are *not* propagated back to the model, so it may take some work to efficiently merge any changes, if necessary.

For convenience, one can access the dimensions and dimension names using `dim.RGtkDataframe` and `dimnames.RGtkDataFrame`, respectively. It is possible to set the dimension names using the conventional replacement function. Note that `rownames` mean nothing to GTK.

Value

The constructors return instances of `RGtkDataFrame`. `as.data.frame.RGtkDataFrame` returns the data frame backing the model. `[.RGtkDataFrame` returns the result of the `[` method on the backing frame.

Note

It is not yet clear how to encode a tree structure with a data frame, so this is only currently useful for flat tables.

Author(s)

Michael Lawrence

 RGtkObject

The base object of RGtk2

Description

RGtkObject identifies an external object as being owned by RGtk. Practically, it allows convenience operators to be specified for any external object.

Usage

```

[[.RGtkObject"(x, field, where = parent.frame())
".$.RGtkObject"(x, member)
"==.RGtkObject"(x, y)

```

Arguments

x	The RGtkObject to which the method or field belongs or the left hand of a comparison
field	The name of the field whose value will be retrieved
member	The name of the member (eg method) that will be retrieved
y	The right hand operand of a comparison
where	The environment in which to look for the field accessor function

Details

The functions `[[.RGtkObject` and `$.RGtkObject` both expand to an RGtk function that accesses external objects. The `[[` operator looks for a field from an external C structure by expanding `objectOfClassName[[fieldName]]` to `classNameGetFieldName()`. External "methods" are expanded by the `$` operator to form `classNameMethodName(objectOfClassName, ...)` from the Java-like `objectOfClassName$methodName(...)`. The long and short mechanisms give the same result, but the shortcut is obviously more convenient. If the method does not exist, `$` will fall back to other types of members, like properties (for `GObjects`) and fields.

The `==` operator compares two RGtkObjects on the basis of their internal pointer value. This should rarely be useful for users.

Value

A context-dependent value resulting from the specified API call.

Author(s)

Michael Lawrence

transparent-type *Transparent types in RGtk2*

Description

A *transparent type* in RGtk2 is a non-primitive type passed between the user and the API as an ordinary R object (usually a list with a defined structure).

Details

The RGtk2 documentation defines the public structure of every object. Some of these have been tagged as being *transparent*, indicating that the R programmer need not obtain an external object but rather simply create the analogous structure in R. *Transparent types* are usually simple types that would be created inline in C code for convenience, instead of invoking a function with a large number of arguments. RGtk2 emulates this in R.

Usually these structures are constructed as R lists, with optionally named elements. The lists elements are matched up to structure fields according to the same logic as function calls to function definitions (see [match.call](#)).

Author(s)

Michael Lawrence

See Also

[GParamSpec](#) [GtkFileFilterInfo](#) [GtkTargetEntry](#) [AtkAttribute](#)
[GtkSettingsValue](#) [GClosure](#) [GType](#)
[GtkStockItem](#) [GtkItemFactoryEntry](#) [GtkAllocation](#) [GdkAtom](#) [GTimeVal](#)
[PangoRectangle](#) [GdkRectangle](#) [AtkAttributeSet](#) [GdkRgbCmap](#) [GdkKeymapKey](#)
[GdkGCValues](#) [GdkGeometry](#)
[GdkPoint](#) [GdkSegment](#) [GdkColor](#) [GdkNativeWindow](#) [GError](#) [GdkWindowAttr](#) [GdkTrapezoid](#)
[GtkActionEntry](#) [GtkToggleActionEntry](#) [GtkRadioActionEntry](#) [CairoPath](#) [CairoGlyph](#)
[CairoPathData](#) [AtkTextRectangle](#) [AtkTextRange](#) [GdkSpan](#) [GdkTimeCoord](#)

Index

!.flag (*enums-and-flags*), 8

*Topic **interface**

- ATK, 2
- CAIRO, 4
- checkGTK, 5
- classes, 6
- enums-and-flags, 8
- GDK, 9
- GDK-Pixbuf, 10
- GMainLoop, 10
- GObject, 11
- GSignal, 14
- GTK, 16
- GType, 21
- Libglade, 23
- Pango, 23
- RGtk, 24
- RGtkDataFrame, 25
- RGtkObject, 26
- transparent-type, 27

*Topic **misc**

- assertions, 2
- ==.RGtkObject (*RGtkObject*), 26
- ==.enum (*enums-and-flags*), 8
- [.GObject (*GObject*), 11
- [.RGtkDataFrame (*RGtkDataFrame*), 25
- [.flags (*enums-and-flags*), 8
- [<-.GObject (*GObject*), 11
- [<-.RGtkDataFrame (*RGtkDataFrame*), 25
- [[.GObject (*GObject*), 11
- [[.RGtkObject (*RGtkObject*), 26
- [[<-.GObject (*GObject*), 11
- \$.GObject (*GObject*), 11
- \$.RGtkObject, 13
- \$.RGtkObject (*RGtkObject*), 26
- \$<-.GObject (*GObject*), 11
- &.flag (*enums-and-flags*), 8
- |.flag (*enums-and-flags*), 8
- as.data.frame.RGtkDataFrame (*RGtkDataFrame*), 25
- as.flag (*enums-and-flags*), 8
- as.struct (*transparent-type*), 27
- assertions, 2
- assignProp (*classes*), 6
- ATK, 2, 24
- atk-AtkMisc, 3
- atk-AtkState, 3
- atk-AtkStateSet, 3
- AtkAction, 3
- AtkAttribute, 28
- AtkAttributeSet, 28
- AtkComponent, 3
- AtkDocument, 3
- AtkEditableText, 3
- AtkGObjectAccessible, 3
- AtkHyperlink, 3
- AtkHypertext, 3
- AtkImage, 3
- AtkNoOpObject, 3
- AtkNoOpObjectFactory, 3
- AtkObject, 3
- AtkObjectFactory, 3
- AtkRegistry, 3
- AtkRelation, 3
- AtkRelationSet, 3
- AtkSelection, 3
- AtkStreamableContent, 3
- AtkTable, 3
- AtkText, 3
- AtkTextRange, 28
- AtkTextRectangle, 28
- AtkUtil, 3
- AtkValue, 3
- boundCairoVersion (*checkGTK*), 5
- boundGTKVersion (*checkGTK*), 5

- boundPangoVersion (*checkGTK*), 5
- CAIRO, 4
- Cairo, 24, 25
- cairo-cairo-font-face-t, 4
- cairo-cairo-matrix-t, 4
- cairo-cairo-surface-t, 4
- cairo-cairo-t, 4
- cairo-Error-Handling, 4
- cairo-Font-Options, 4
- cairo-Image-Surfaces, 4
- cairo-Paths, 4
- cairo-Patterns, 4
- cairo-PDF-Surfaces, 4
- cairo-PNG-Support, 4
- cairo-PostScript-Surfaces, 4
- cairo-Scaled-Fonts, 4
- cairo-SVG-Surfaces, 4
- cairo-Text, 4
- cairo-Transformations, 4
- cairo-Types, 4
- cairo-Version-Information, 4
- CairoGlyph, 28
- CairoPath, 28
- CairoPathData, 28
- checkArrType (*assertions*), 2
- checkCairo (*checkGTK*), 5
- checkGTK, 5
- checkPango (*checkGTK*), 5
- checkPtrType (*assertions*), 2
- classes, 6
- compareVersion, 5
- connectSignal (*GSignal*), 14
- dim.RGtkDataFrame
 (*RGtkDataFrame*), 25
- dimnames.RGtkDataFrame
 (*RGtkDataFrame*), 25
- dimnames<- .RGtkDataFrame
 (*RGtkDataFrame*), 25
- enums-and-flags, 8
- gClass (*classes*), 6
- GClosure, 28
- GConnectFlags (*GSignal*), 14
- GDK, 9, 24
- gdk-Bitmaps-and-Pixmaps, 9
- gdk-Cairo-Interaction, 9
- gdk-Colormaps-and-Colors, 9
- gdk-Cursors, 9
- gdk-Drag-and-Drop, 9
- gdk-Drawing-Primitives, 9
- gdk-Event-Structures, 9
- gdk-Events, 9
- gdk-Fonts, 9
- gdk-GdkRGB, 9
- gdk-General, 9
- gdk-Graphics-Contexts, 9
- gdk-Images, 9
- gdk-Input-Devices, 9
- gdk-Keyboard-Handling, 9
- gdk-Pango-Interaction, 9
- GDK-Pixbuf, 10, 24
- gdk-pixbuf-animation, 10
- gdk-pixbuf-creating, 10
- gdk-pixbuf-file-loading, 10
- gdk-pixbuf-file-saving, 10
- gdk-pixbuf-gdk-pixbuf, 10
- gdk-pixbuf-Module-Interface, 10
- gdk-pixbuf-scaling, 10
- gdk-pixbuf-util, 10
- gdk-pixbuf-Versioning, 10
- gdk-Pixbufs, 9
- gdk-Points-Rectangles-and-Regions,
 9
- gdk-Properties-and-Atoms, 9
- gdk-Visuals, 9
- gdk-Windows, 9
- GdkAtom, 28
- GdkColor, 28
- GdkDisplay, 9
- GdkDisplayManager, 9
- GdkGCValues, 28
- GdkGeometry, 28
- GdkKeymapKey, 28
- GdkNativeWindow, 28
- GDKPixbuf, 25
- GdkPixbuf, 10, 24
- GdkPixbufLoader, 10
- GdkPoint, 28
- GdkRectangle, 28
- GdkRgbCmap, 28
- GdkScreen, 9
- GdkSegment, 28
- GdkSpan, 28
- GdkTimeCoord, 28

- GdkTrapezoid, 28
- GdkWindowAttr, 28
- GError, 28
- getProp (*classes*), 6
- gIdleAdd (*GMainLoop*), 10
- GladeXML, 23
- GMainLoop, 10, 24
- GObject, 6, 7, 11, 16, 22, 24, 27
- gObject (*GObject*), 11
- gObjectGet (*GObject*), 11
- gObjectGetData (*GObject*), 11
- gObjectGetPropInfo (*GObject*), 11
- gObjectGetSignals, 14, 15
- gObjectGetSignals (*GObject*), 11
- gObjectNew, 13
- gObjectNew (*GObject*), 11
- gObjectParentClass (*GObject*), 11
- gObjectSet (*GObject*), 11
- gObjectSetData (*GObject*), 11
- GParamSpec, 6, 13, 28
- gParamSpec, 7
- GSignal, 14, 14
- gSignalConnect, 13
- gSignalConnect (*GSignal*), 14
- gSignalEmit (*GSignal*), 14
- GSignalFlags, 6
- GSignalFlags (*GSignal*), 14
- gSignalGetInfo (*GSignal*), 14
- gSignalHandlerBlock (*GSignal*), 14
- gSignalHandlerDisconnect
 (*GSignal*), 14
- gSignalHandlerUnblock (*GSignal*),
 14
- gSignalStopEmission (*GSignal*), 14
- gSourceRemove (*GMainLoop*), 10
- gTimeoutAdd (*GMainLoop*), 10
- GTimeVal, 28
- GTK, 16, 24
- gtk-Accelerator-Maps, 16
- gtk-Clipboards, 17
- gtk-Drag-and-Drop, 17
- gtk-General, 19
- gtk-Graphics-Contexts, 18
- gtk-gtkbuildable, 16
- gtk-gtkfilefilter, 18
- gtk-GtkPaperSize, 19
- gtk-GtkTextIter, 20
- gtk-GtkTreeView-drag-and-drop, 20
- gtk-High-level-Printing-API, 19
- gtk-Keybaord-Accelerators, 16
- gtk-Resource-Files, 19
- gtk-Selections, 20
- gtk-Standard-Enumerations, 17
- gtk-Stock-Items, 20
- gtk-Themeable-Stock-Images, 18
- GtkAboutDialog, 16
- GtkAccelLabel, 16
- GtkAccessible, 16
- GtkAction, 16
- GtkActionEntry, 28
- GtkActionGroup, 16
- GtkAdjustment, 16
- GtkAlignment, 16
- GtkAllocation, 28
- GtkArrow, 16
- GtkAspectFrame, 16
- GtkAssistant, 16
- GtkBin, 16
- GtkBox, 16
- GtkBuilder, 17
- GtkButton, 16, 17, 24
- GtkButtonBox, 16
- GtkCalendar, 17
- GtkCellEditable, 17
- GtkCellLayout, 17
- GtkCellRenderer, 17
- GtkCellRendererAccel, 17
- GtkCellRendererCombo, 17
- GtkCellRendererPixbuf, 17
- GtkCellRendererProgress, 17
- GtkCellRendererSpin, 17
- GtkCellRendererText, 17
- GtkCellRendererToggle, 17
- GtkCellView, 17
- GtkCheckButton, 17
- GtkCheckMenuItem, 17
- GtkCList, 17
- GtkColorButton, 17
- GtkColorSelection, 17
- GtkColorSelectionDialog, 17
- GtkCombo, 17
- GtkComboBox, 17
- GtkComboBoxEntry, 17
- GtkContainer, 17
- GtkCTree, 17
- GtkCurve, 17

- GtkDialog, 17
- GtkDrawingArea, 17
- GtkEditable, 17
- GtkEntry, 17, 23, 24
- GtkEntryCompletion, 17
- GtkEventBox, 17
- GtkExpander, 17
- GtkFileChooser, 17
- GtkFileChooserButton, 18
- GtkFileChooserDialog, 18
- GtkFileChooserWidget, 18
- GtkFileFilterInfo, 28
- GtkFileSelection, 18
- GtkFixed, 18
- GtkFontButton, 18
- GtkFontSelection, 18
- GtkFontSelectionDialog, 18
- GtkFrame, 18
- GtkGammaCurve, 18
- GtkHandleBox, 18
- GtkHBox, 18
- GtkHButtonBox, 18
- GtkHPaned, 18
- GtkHRuler, 18
- GtkHScale, 18
- GtkHScrollbar, 18
- GtkHSeparator, 18
- GtkIconTheme, 18
- GtkIconView, 18
- GtkImage, 10, 18, 24
- GtkImageMenuItem, 18
- GtkIMContext, 18
- GtkIMContextSimple, 18
- GtkIMMulticontext, 18
- GtkInputDialog, 18
- GtkInvisible, 18
- GtkItem, 18
- GtkItemFactory, 18
- GtkItemFactoryEntry, 28
- GtkLabel, 18, 23, 24
- GtkLayout, 18
- GtkLinkButton, 18
- GtkList, 18
- GtkListItem, 18
- GtkListStore, 19
- GtkMenu, 19
- GtkMenuBar, 19
- GtkMenuItem, 19
- GtkMenuShell, 19
- GtkMenuToolButton, 19
- GtkMessageDialog, 19
- GtkMisc, 19
- GtkNotebook, 19
- gtkObject (*GObject*), 11
- gtkObjectNew (*GObject*), 11
- GtkOldEditable, 19
- GtkOptionMenu, 19
- GtkPageSetup, 19
- GtkPaned, 19
- GtkPixmap, 19
- GtkPlug, 19
- GtkPreview, 19
- GtkPrintContext, 19
- GtkPrintSettings, 19
- GtkProgress, 19
- GtkProgressBar, 19
- GtkRadioAction, 19
- GtkRadioActionEntry, 28
- GtkRadioButton, 19
- GtkRadioMenuItem, 19
- GtkRadioToolButton, 19
- GtkRange, 19
- GtkRecentAction, 19
- GtkRecentChooser, 19
- GtkRecentChooserDialog, 19
- GtkRecentChooserMenu, 19
- GtkRecentChooserWidget, 19
- GtkRecentFilter, 19
- GtkRecentManager, 19
- GtkRuler, 19
- GtkScale, 20
- GtkScaleButton, 20
- GtkScrollbar, 20
- GtkScrolledWindow, 20
- GtkSeparator, 20
- GtkSeparatorMenuItem, 20
- GtkSeparatorToolItem, 20
- GtkSettings, 20
- GtkSettingsValue, 28
- GtkSizeGroup, 20
- GtkSocket, 20
- GtkSpinButton, 20
- GtkStatusbar, 20
- GtkStatusIcon, 20
- GtkStockItem, 28
- GtkStyle, 20

- GtkTable, 20
- GtkTargetEntry, 28
- GtkTearoffMenuItem, 20
- GtkTextBuffer, 20
- GtkTextMark, 20
- GtkTextTag, 20
- GtkTextTagTable, 20
- GtkTextView, 16, 20, 23, 24
- GtkTipsQuery, 20
- GtkToggleAction, 20
- GtkToggleActionEntry, 28
- GtkToggleButton, 20
- GtkToggleToolButton, 20
- GtkToolbar, 20
- GtkToolButton, 20
- GtkToolItem, 20
- GtkTooltip, 20
- GtkTooltips, 20
- GtkTreeModel, 20, 25, 26
- GtkTreeModelFilter, 20
- GtkTreeModelSort, 20
- GtkTreeSelection, 21
- GtkTreeSortable, 21
- GtkTreeStore, 21
- GtkTreeView, 21
- GtkTreeViewColumn, 21
- GtkUIManager, 21
- GtkVBox, 21
- GtkVButtonBox, 21
- GtkViewport, 21
- GtkVolumeButton, 21
- GtkVPaned, 21
- GtkVRuler, 21
- GtkVScale, 21
- GtkVScrollbar, 21
- GtkVSeparator, 21
- GtkWidget, 21
- GtkWidgetClass, 22
- GtkWindow, 13, 21
- GtkWindowGroup, 21
- gtkWindowNew, 13
- GType, 7, 12, 14–16, 21, 28
 - gTypeFromName (*GType*), 21
 - gTypeGetAncestors (*GType*), 21
 - gTypeGetClass (*GType*), 21
 - gTypeGetInterfaces (*GType*), 21
 - gTypeGetPropInfo (*GObject*), 11
 - gTypeGetSignals (*GType*), 21
- implements (*assertions*), 2
- interface.*GObject* (*GObject*), 11
- Libglade, 23, 24
- match.call, 28
- names.*GObject* (*GObject*), 11
- Pango, 23, 24
 - pango-Cairo-Rendering, 23
 - pango-Coverage-Maps, 23
 - pango-Fonts, 23
 - pango-Glyph-Storage, 23
 - pango-Layout-Objects, 23
 - pango-Scripts, 23
 - pango-Tab-Stops, 23
 - pango-Text-Attributes, 23
 - pango-Text-Processing, 23
 - pango-Version-Checking, 23
 - pango-Vertical-Text, 23
- PangoLayout, 23, 24
- PangoRectangle, 28
- PangoRenderer, 23
- parentHandler (*classes*), 6
- registerVirtuals (*classes*), 6
- RGtk, 24, 26
- RGtkDataFrame, 25
- rGtkDataFrame (*RGtkDataFrame*), 25
 - rGtkDataFrameAppendColumns (*RGtkDataFrame*), 25
 - rGtkDataFrameAppendRows (*RGtkDataFrame*), 25
 - rGtkDataFrameNew (*RGtkDataFrame*), 25
 - rGtkDataFrameSetFrame (*RGtkDataFrame*), 25
- RGtkObject, 24, 26
- transparent-type, 22
- transparent-type, 27
- unregisterVirtuals (*classes*), 6