Package ‘RGtk2’

May 10, 2017

Version 2.20.33
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 (>= 3.4.0)
Imports methods
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
Encoding UTF-8
NeedsCompilation yes
Repository CRAN
Date/Publication 2017-05-10 12:19:03 UTC

R topics documented:

assertions ....................................................... 2
ATK .......................................................... 3
CAIRO ......................................................... 4
checkGTK ...................................................... 5
classes .......................................................... 6
enums-and-flags .................................................. 8
GDK .............................................................. 9
GDK-Pixbuf .................................................... 10
GIO .............................................................. 11
giocon .......................................................... 13
GMainLoop ...................................................... 13
assertions

RGtk2 Type Assertion

Description

Assert that an object is of a particular type

Usage

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

Arguments

w An object whose type is to be verified.
klass The type the object is expected to be.
nullOk Whether the object is allowed to be NULL.
critical 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.
obj A GObject.
interface The interface that obj 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
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.
- **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 valuators and components which display or select a value from a bounded range of values.

Author(s)
Derived by RGtkGen from GTK+ documentation

References
https://developer.gnome.org/atk

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-font-face Base class for font faces
cairo-font-options How a font should be rendered
cairo-image-surface Rendering to memory buffers
cairo-matrix Generic matrix operations
cairo-paths Creating paths and manipulating path data
cairo-pattern Sources for drawing
cairo-pdf-surface Rendering PDF documents
cairo-png-functions Reading and writing PNG images
cairo-ps-surface Rendering PostScript documents
cairo-scaled-font Font face at particular size and options
cairo-error-status Decoding cairo’s status
cairo-surface Base class for surfaces
cairo-svg-surface Rendering SVG documents
cairo-text Rendering text and glyphs
**cairo-transformation**s Manipulating the current transformation matrix  
**cairo-types** Generic data types  
**cairo-user-font** Font support with font data provided by the user  
**cairo-version-info** Compile-time and run-time version checks.  
**cairo-context** The cairo drawing context  

**Author(s)**  
Derived by RGtkGen from GTK+ documentation  

**References**  

---

**checkGTK**  
**Bound versions**

---

**Description**

These functions are for querying (bound*) and checking (check*) the bound versions of the libraries (GTK, Pango and Cairo). As of RGtk2 2.20.19, the check* functions are deprecated in favor of the more explicit boundVersion() >= version syntax.

**Usage**

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

**Arguments**

version Version description to compare to the bound version, as in: boundGTKVersion() >= version.

**Value**

The check* functions return TRUE if version is satisfied, otherwise FALSE.  
The bound* functions return a numeric_version representation of the bound library version.

**Author(s)**

Michael Lawrence
Examples

```r
## instead of
# checkGTK("2.12.0")
## do this:
boundGTKVersion() >= "2.12.0"
```

---

**Custom GObject classes**

**Description**

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

**Usage**

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

**Arguments**

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

**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 virtuals 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 rggobi.

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

**Description**

Convenience functions and operators for operating on bitflags and enums

**Usage**

```r
as.flag(x)
## S3 method for class 'flags'
x[value]
## S3 method for class 'flag'
x | y
## S3 method for class 'flag'
x & y
## S3 method for class 'flag'
!x
## S3 method for class 'enum'
x == y
```

**Arguments**

- `x` Numeric value to coerce to a flag, an object of class `flags`, or the left hand operand
- `y` Right hand operand
- `value` The character id or index for a particular flag in a `flags` 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` or `enums` class retrieves a value as a flag or enum, respectively.

The `==.enum` method compares an 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`. 
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
- **gdk-Application-launching** Startup notification for applications
- **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-Testing** Test utilities
- **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-Bitmap-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
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

https://developer.gnome.org/gdk2

GdkPixbuf  GDK-Pixbuf
Description

GIO is a modern, easy-to-use VFS API

Details

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

- gio-Extension-Points: Extension Points
- GAppInfo: Application information and launch contexts
- GAsyncInitable: Asynchronously failable object initialization interface
- GAsyncResult: Asynchronous Function Results
- GBufferedInputStream: Buffered Input Stream
- GBufferedOutputStream: Buffered Output Stream
- GCancellable: Thread-safe Operation Cancellation Stack
- gio-GContentType: Platform-specific content typing
- GDataInputStream: Data Input Stream
- GDataOutputStream: Data Output Stream
- GDrive: Drive management
- GEmblem: An object for emblems
- GEmblemedIcon: Icon with emblems
- GFile: File and Directory Handling
- gio-GFileAttribute: Key-Value Paired File Attributes
- GFileEnumerator: Enumerated Files Routines
- GFileIcon: Icons pointing to an image file
- GFileInfo: File Information and Attributes
- GFileInputStream: File input streaming operations
- GFileOutputStream: File read and write streaming operations
- GFileMonitor: File Monitor
- GFilenameCompleter: Filename Completer
- GFileOutputStream: File output streaming operations
- GFilterInputStream: Filter Input Stream
- GFilterOutputStream: Filter Output Stream
- GIcon: Interface for icons
- GInetAddress: An IPv4/IPv6 address
- GInetSocketAddress: Internet GSocketAddress
**GInitable** Failable object initialization interface

**GInputStream** Base class for implementing streaming input

**gio-GIOError** Error helper functions

**GIOModule** Loadable GIO Modules

**gio-GIOScheduler** I/O Scheduler

**GIOStream** Base class for implementing read/write streams

**GLoadableIcon** Loadable Icons

**GMemoryInputStream** Streaming input operations on memory chunks

**GMemoryOutputStream** Streaming output operations on memory chunks

**GMount** Mount management

**GMountOperation** Object used for authentication and user interaction

**GNetworkAddress** A GSocketConnectable for resolving hostnames

**GNetworkService** A GSocketConnectable for resolving SRV records

**GOutputStream** Base class for implementing streaming output

**GResolver** Asynchronous and cancellable DNS resolver

**GSeekable** Stream seeking interface

**GSimpleAsyncResult** Simple asynchronous results implementation

**GSocket** Low-level socket object

**GSocketAddress** Abstract base class representing endpoints for socket communication

**GSocketClient** Helper for connecting to a network service

**GSocketConnectable** Interface for potential socket endpoints

**GSocketConnection** A socket connection

**GSocketControlMessage** A GSocket control message

**GSocketListener** Helper for accepting network client connections

**GSocketService** Make it easy to implement a network service

**GSrvTarget** DNS SRV record target

**GThemedIcon** Icon theming support

**GThreadedSocketService** A threaded GSocketService

**GVfs** Virtual File System

**GVolume** Volume management

**GVolumeMonitor** Volume Monitor

**Author(s)**

Derived by RGtkGen from GTK+ documentation

**References**

https://developer.gnome.org/gio
giocon

R connections for GIO streams

Description

Creates a regular R connection based on a GIO stream. The integration happens directly in C. The connection can be used anywhere connections are supported. See the httpd demo for examples.

Usage

giocon(stream, binary = FALSE, blocking = TRUE)

Arguments

stream The GIO endpoint, such as a GInputStream, GOutputStream or GIOStream.
binary Whether the data are binary or character.
blocking Whether the connection should block when reading and writing.

Value

An R connection

Author(s)

Michael Lawrence

GMainLoop

The GLib Main Loop

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

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

The RGtk2 user has limited control over the event loop, but it 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


GObject

The GObject system in RGtk2

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)
## S3 method for class 'GObject'
obj[value, ...]
bagaiSet(obj, ...)
## S3 replacement method for class 'GObject'
obj[propNames] <- value
## S3 method for class 'GObject'
obj[[member, where = parent.frame()]]
## S3 replacement method for class 'GObject'
obj[[member]] <- value
## S3 method for class 'GObject'
x$member
## S3 replacement method for class 'GObject'
GObject

```r
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)
## S3 method for class 'GObject'
names(x)
interface(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.

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


See Also

GType GSignal
**GQuark**

**Description**

A GQuark is a unique identifier used for internalizing strings in GLib. RGtk2 will automatically coerce R strings to GQuarks as needed, but see `as.GQuark` for explicit coercion.

**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**

```r
gsignalconnectHobjL signalL fL data \] nullL after \] falseL userNdataNfirst \] falseI
gsignalhandlerdisconnectHobjL idI
gsignalhandlerblockHobjL idI
gsignalhandlerunblockHobjL idI
gsignalemitHobjL signalL NNNL detail \] nullI
gsignalstopemissionHobjL signalL detail \] nullI
gsignalgetinfoHsigI
```

**Arguments**

- **obj** The object that owns the signal
- **signal** The detailed name of the signal
- **f** The R function to connect as a callback
- **data** Arbitrary "user data" that will be passed to the callback `f`
- **after** Whether `f` will be called before or after the default handler
- **user.data.first** Whether the data is the first or last argument to the callback
- **id** The signal handler id obtained upon connection to the signal
- **...** Arguments to pass to the signal handlers
- **detail** Optional separate argument for the `detail` portion of the signal
- **sig** A signal id provided by `gObjectGetSignals`. 
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 gSignalEmitt 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 gSignalEmitt. 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:

- returnType: The return GType id of the signal
- signal: The signal id
parameters  A list of GType ids for the parameters
objectType  The GType id owning the signal
runFlags  The flags determining behavior of the signal, see reference

Author(s)
Adapted from GSignal documentation by Michael Lawrence

References

See Also
GObject

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:

- chap-drawing-model  The GTK+ drawing model in detail
- gtk-Filesystem-utilities  Functions for working with GIO
- GtkAboutDialog  Display information about an application
- gtk-Keyboard-Accelerators  Groups of global keyboard accelerators for an entire GtkWidget
- 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
- GtkActivatable  An interface for activatable widgets
- GtkAdjustment  A GtkWidget representing an adjustable bounded value
- GtkAlignment  A widget which controls the alignment and size of its child
- GtkArrow  Displays an arrow
- GtkAspectRatio  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 are 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  
**GtkCellRendererSpinner**  Renders a spinning animation 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  
**gtk-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  
**GtkEntryBuffer**  Text buffer for GtkEntry
GTK

**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

**GtkHSV**  A 'color wheel' widget

**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

**GtkInfoBar**  Report important messages to the user

**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
GtkOffscreenWindow A toplevel container widget used to manage offscreen rendering of child widgets.
GtkOldEditable Base class for text-editing widgets
GtkOptionMenu A widget used to choose from a list of valid choices
gtk-Orientable An interface for flippable widgets
GtkPageSetup Stores page setup information
GtkPaned Base class for widgets with two adjustable panes
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
GTK

**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

**GtkSpinner** Show a spinner animation

**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

**gtk-Testing** Utilities for testing GTK+ applications

**GtkTextBuffer** Stores attributed text for display in a GtkTextView

**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 GtkToolShell
GtkToolItemGroup  A sub container used in a tool palette
GtkToolPalette  A tool palette with categories
GtkToolShell  Interface for containers containing GtkToolItem widgets
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  A vertical scrollbar
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

https://developer.gnome.org/gtk
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)
- gTypeGetSignals(type)

Arguments

- type: The GType, either its name or numeric value, see below
- name: The name of a GType

Details

The GType system supports inheritance and interfaces, enabling the pseudo-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.
- gTypeGetSignals returns a list of signal ids with names for the signals supported by the type.
Author(s)

Michael Lawrence

References

https://developer.gnome.org/gobject/stable/gobject-Type-Information.html

See Also

GObject

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-Bidirectional-Text**  Types and functions to help with handling bidirectional text
- **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-and-Languages**  Identifying writing systems and languages
- **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

https://developer.gnome.org/pango
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 `GtkWidget` 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.

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
References


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, ...)
## S3 method for class 'RGtkDataFrame'
as.data.frame(x, ...)
RGtkDataFrameSetFrame(x, frame = data.frame())
## S3 method for class 'RGtkDataFrame'
x[i, j, drop = T]
## S3 replacement method for class 'RGtkDataFrame'
x[i, j] <- value
## S3 method for class 'RGtkDataFrame'
dim(x, ...)
## S3 method for class 'RGtkDataFrame'
dimnames(x, ...)
## S3 replacement method for class 'RGtkDataFrame'
dimnames(x) <- value

Arguments

frame The frame to use as the backing store of the model
x An RGtkDataFrame object
i Row index
j Column index
value An R object similar to that accepted by [<-.data.frame or the dimnames for the data frame
drop Whether to 'drop' the result to the simplest structure
... 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
Usage

```r
# S3 method for class 'RGtkObject'
x[[field, where = parent.frame()]]
# S3 method for class 'RGtkObject'
x$member
# S3 method for class '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 (e.g., 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 `GObject`s) and fields.

The `==` operator compares two `RGtkObject`s 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 GtkWidgetFactoryEntry GtkWidgetAllocation GdkAtom GTimeVal
PangoRectangle GdkRectangle AtkAttributeSet GdkRgbCmap GdkKeymapKey GdkGCVValues GdkGeometry
GdkPoint GdkSegment GdkColor GdkNativeWindow GError GdkWindowAttr GdkTrapezoid
GtkActionEntry GtkToggleActionEntry GtkRadioActionEntry CairoPath CairoGlyph
CairoPathData AtkTextRectangle AtkTextRange GdkSpan GdkTimeCoord
Index

+Topic ANflag (enums-and-flags), 8
  +Topic interface
  ATK, 3
  CAIRO, 4
  checkGTK, 5
  classes, 6
  enums-and-flags, 8
  GDK, 9
  GDK-Pixbuf, 10
  GIO, 11
  GMainLoop, 13
  Object (GObject), 14
  GSignal, 17
  GTK, 19
  GType, 25
  Pango, 26
  RTK, 27
  RTkDataFrame, 28
  RTkObject, 29
  transparent-type, 30
+Topic misc
  assertions, 2
  =. RTkObject (RTkObject), 29
  =. enum (enums-and-flags), 8
  [. GObject (GObject), 14
  [. RTkDataFrame (RTkDataFrame), 28
  [. enums (enums-and-flags), 8
  [. flags (enums-and-flags), 8
  [<-. GObject (GObject), 14
  [<-. RTkDataFrame (RTkDataFrame), 28
  [[. GObject (GObject), 14
  [[. RTkObject (RTkObject), 29
  [[<-. GObject (GObject), 14
  $. GObject (GObject), 14
  $. RTkObject, 16
  $. RTkObject (RTkObject), 29
  $<-. GObject (GObject), 14
  &. flag (enums-and-flags), 8
  as.data.frame.RTkDataFrame (RTkDataFrame), 28
  as.flag (enums-and-flags), 8
  as.QQuark, 17
  as.struct (transparent-type), 30
  assertions, 2
  assignProp (classes), 6
  ATK, 3, 27
  atk-AtkMisc, 3
  atk-AtkState, 3
  AtkAction, 3
  AtkAttribute, 31
  AtkAttributeSet, 31
  AtkComponent, 3
  AtkDocument, 3
  AtkEditorableText, 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
  AtkStateSet, 3
  AtkStreamableContent, 4
  AtkTable, 4
  AtkText, 4
  AtkTextRange, 31
  AtkTextRectangle, 31
  AtkUtil, 4
  AtkValue, 4
  boundCairoVersion (checkGTK), 5
  boundGTKVersion (checkGTK), 5
  boundPangoVersion (checkGTK), 5

32
INDEX

GdkPoint, 31
GdkRectangle, 31
GdkRgbCmap, 31
GdkScreen, 9
GdkSegment, 31
GdkSpan, 31
GdkTimeCoord, 31
GdkTrapezoid, 31
GdkWindowAttr, 31
GDrive, 11
GEmblem, 11
GEmblemedIcon, 11
GError, 31
getProp (classes), 6
GFile, 11
GFileEnumerator, 11
GFileIcon, 11
GFileInfo, 11
GFileInputStream, 11
GFileInputStream, 11
GFileMonitor, 11
GFilenameCompleter, 11
GFileOutputStream, 11
GFilterInputStream, 11
GFilterOutputStream, 11
GIcon, 11
GIcon
GObject (GObject), 14
GObjectGet (GObject), 14
GObjectGetData (GObject), 14
GObjectGetPropertyInfo (GObject), 14
GObjectGetSignals (GObject), 14
GObjectNew, 15
GObjectNew (GObject), 14
GObjectParentClass (GObject), 14
GObjectSet (GObject), 14
GObjectSetData (GObject), 14
GOutputStream, 12
GParamSpec, 6, 16, 31
GParamSpec, 7
GQuark, 17
GResolver, 12
GSeekable, 12
GSignal, 16, 17
GSignalConnect, 16
GSignalConnect (GSignal), 17
GSignalEmit (GSignal), 17
GSignalFlags, 7
GSignalFlags (GSignal), 17
GSignalGetInfo (GSignal), 17
GSignalHandlerBlock (GSignal), 17
GSignalHandlerDisconnect (GSignal), 17
GSignalHandlerUnblock (GSignal), 17
GSignalStopEmission (GSignal), 17
GSimpleAsyncResult, 12
GSocket, 12
GSocketAddress, 12
GSocketClient, 12
GSocketConnectable, 12
GSocketConnection, 12
GSocketControlMessage, 12
GSocketListener, 12
GSocketService, 12
GSourceRemove (GMainLoop), 13
GSrvTarget, 12
GThemedIcon, 12
GThreadedSocketService, 12
GTimeoutAdd (GMainLoop), 13
GTimeVal, 31
GTK, 19, 27
gtk-Accelerator-Maps, 19
gtk-Clipboards, 20
gtk-Drag-and-Drop, 20

GObject, 6, 7, 14, 19, 25–27, 30
GObject (GObject), 14
GObjectGet (GObject), 14
GObjectGetData (GObject), 14
GObjectGetPropertyInfo (GObject), 14
GObjectGetSignals, 17, 18
GObjectGetSignals (GObject), 14
GObjectNew, 15
GObjectNew (GObject), 14
GObjectParentClass (GObject), 14
GObjectSet (GObject), 14
GObjectSetData (GObject), 14
GOutputStream, 12
GParamSpec, 6, 16, 31
GParamSpec, 7
GQuark, 17
GResolver, 12
GSeekable, 12
GSignal, 16, 17
GSignalConnect, 16
GSignalConnect (GSignal), 17
GSignalEmit (GSignal), 17
GSignalFlags, 7
GSignalFlags (GSignal), 17
GSignalGetInfo (GSignal), 17
GSignalHandlerBlock (GSignal), 17
GSignalHandlerDisconnect (GSignal), 17
GSignalHandlerUnblock (GSignal), 17
GSignalStopEmission (GSignal), 17
GSimpleAsyncResult, 12
GSocket, 12
GSocketAddress, 12
GSocketClient, 12
GSocketConnectable, 12
GSocketConnection, 12
GSocketControlMessage, 12
GSocketListener, 12
GSocketService, 12
GSourceRemove (GMainLoop), 13
GSrvTarget, 12
GThemedIcon, 12
GThreadedSocketService, 12
GTimeoutAdd (GMainLoop), 13
GTimeVal, 31
GTK, 19, 27
gtk-Accelerator-Maps, 19
gtk-Clipboards, 20
gtk-Drag-and-Drop, 20
<table>
<thead>
<tr>
<th>gtk-Mfilesystem-utilities, 19</th>
<th>GtKList, 20</th>
</tr>
</thead>
<tbody>
<tr>
<td>gtk-General, 22</td>
<td>GtKColorButton, 20</td>
</tr>
<tr>
<td>gtk-Graphics-Contexts, 21</td>
<td>GtKColorSelection, 20</td>
</tr>
<tr>
<td>gtk-gtkbuildable, 20</td>
<td>GtKColorSelectionDialog, 20</td>
</tr>
<tr>
<td>gtk-gtkcheckmenuitem, 20</td>
<td>GtKCombo, 20</td>
</tr>
<tr>
<td>gtk-gtkfilefilter, 21</td>
<td>GtKComboBox, 20</td>
</tr>
<tr>
<td>gtk-GtkTreeView-drag-and-drop, 24</td>
<td>GtKComboBoxEntry, 20</td>
</tr>
<tr>
<td>gtk-High-level-Printing-API, 22</td>
<td>GtKContainer, 20</td>
</tr>
<tr>
<td>gtk-Keyboard-Accelerators, 19</td>
<td>GtKCTree, 20</td>
</tr>
<tr>
<td>gtk-Orientable, 22</td>
<td>GtKCurve, 20</td>
</tr>
<tr>
<td>gtk-Resource-Files, 23</td>
<td>GtKDialog, 20</td>
</tr>
<tr>
<td>gtk-Selections, 23</td>
<td>GtKDrawingArea, 20</td>
</tr>
<tr>
<td>gtk-Standard-Enumerations, 21</td>
<td>GtKEditable, 20</td>
</tr>
<tr>
<td>gtk-Stock-Items, 23</td>
<td>GtKEntry, 20, 26, 27</td>
</tr>
<tr>
<td>gtk-Testing, 23</td>
<td>GtKEntryBuffer, 20</td>
</tr>
<tr>
<td>gtk-Themeable-Stock-Images, 21</td>
<td>GtKEntryCompletion, 21</td>
</tr>
<tr>
<td>GtKAboutDialog, 19</td>
<td>GtKEventBox, 21</td>
</tr>
<tr>
<td>GtKAccellabel, 19</td>
<td>GtKExpander, 21</td>
</tr>
<tr>
<td>GtKAccessible, 19</td>
<td>GtKFileChooser, 21</td>
</tr>
<tr>
<td>GtKAction, 19</td>
<td>GtKFileChooserButton, 21</td>
</tr>
<tr>
<td>GtKActionEntry, 31</td>
<td>GtKFileChooserDialog, 21</td>
</tr>
<tr>
<td>GtKActionGroup, 19</td>
<td>GtKFileChooserWidget, 21</td>
</tr>
<tr>
<td>GtKActivatable, 19</td>
<td>GtKFileFilterInfo, 31</td>
</tr>
<tr>
<td>GtKAdjustment, 19</td>
<td>GtKFileSelection, 21</td>
</tr>
<tr>
<td>GtKAlignment, 19</td>
<td>GtKFixed, 21</td>
</tr>
<tr>
<td>GtKAllocation, 31</td>
<td>GtKFontButton, 21</td>
</tr>
<tr>
<td>GtKArrow, 19</td>
<td>GtKFontSelection, 21</td>
</tr>
<tr>
<td>GtKAspectFrame, 19</td>
<td>GtKFontSelectionDialog, 21</td>
</tr>
<tr>
<td>GtKAssistant, 19</td>
<td>GtKFrame, 21</td>
</tr>
<tr>
<td>GtKBin, 20</td>
<td>GtKGammaCurve, 21</td>
</tr>
<tr>
<td>GtKBox, 20</td>
<td>GtKHandleBox, 21</td>
</tr>
<tr>
<td>GtKBuilder, 20</td>
<td>GtKHBox, 21</td>
</tr>
<tr>
<td>GtKButton, 19, 20, 27</td>
<td>GtKHButtonBox, 21</td>
</tr>
<tr>
<td>GtKButtonBox, 19</td>
<td>GtKHPaned, 21</td>
</tr>
<tr>
<td>GtKCalendar, 20</td>
<td>GtKHRuler, 21</td>
</tr>
<tr>
<td>GtKCellEditable, 20</td>
<td>GtKHScale, 21</td>
</tr>
<tr>
<td>GtKCellLayout, 20</td>
<td>GtKHSeparator, 21</td>
</tr>
<tr>
<td>GtKCellRenderer, 20</td>
<td>GtKHSeparator, 21</td>
</tr>
<tr>
<td>GtKCellRendererAccel, 20</td>
<td>GtKHSV, 21</td>
</tr>
<tr>
<td>GtKCellRendererCombo, 20</td>
<td>GtKIconTheme, 21</td>
</tr>
<tr>
<td>GtKCellRendererPixbuf, 20</td>
<td>GtKIconView, 21</td>
</tr>
<tr>
<td>GtKCellRendererProgress, 20</td>
<td>GtKImage, 10, 21, 27</td>
</tr>
<tr>
<td>GtKCellRendererSpin, 20</td>
<td>GtKImageMenuitem, 21</td>
</tr>
<tr>
<td>GtKCellRendererSpinner, 20</td>
<td>GtKIMContext, 21</td>
</tr>
<tr>
<td>GtKCellRendererText, 20</td>
<td>GtKIMContextSimple, 21</td>
</tr>
<tr>
<td>GtKCellRendererToggle, 20</td>
<td>GtKIMMulticontext, 21</td>
</tr>
<tr>
<td>GtKCellValue, 20</td>
<td>GtKInfoBar, 21</td>
</tr>
<tr>
<td>GtKCheckButton, 20</td>
<td>GtKInputDialog, 21</td>
</tr>
</tbody>
</table>
INDEX

GtkInvisible, 21
GtkItem, 22
GtkItemFactory, 22
GtkItemFactoryEntry, 31
GtkLabel, 22, 26, 27
GtkLayout, 22
GtkLinkButton, 22
GtkList, 22
GtkListItem, 22
GtkListStore, 22
GtkMenu, 22
GtkMenuBar, 22
GtkMenuItem, 22
GtkMenuShell, 22
GtkMenuToolButton, 22
GtkMessageDialog, 22
GtkMisc, 22
GtkNotebook, 22
GtkWidget (GObject), 14
GtkObjectNew (GObject), 14
GtkOffscreenWindow, 22
GtkOldEditable, 22
GtkOptionMenu, 22
GtkPageSetup, 22
GtkPaned, 22
GtkPaperSize, 22
GtkPixmap, 22
GtkPlug, 22
GtkPreview, 22
GtkPrintContext, 22
GtkPrintSettings, 22
GtkProgress, 22
GtkProgressBar, 22
GtkRadioAction, 22
GtkRadioActionEntry, 31
GtkRadioButton, 22
GtkRadioMenuEntry, 22
GtkRadioToolButton, 22
GtkRange, 23
GtkRecentAction, 23
GtkRecentChooser, 23
GtkRecentChooserDialog, 23
GtkRecentChooserMenu, 23
GtkRecentChooserWidget, 23
GtkRecentFilter, 23
GtkRecentManager, 23
GtkRuler, 23
GtkScale, 23

GtkScaleButton, 23
GtkScrollbar, 23
GtkScrolledWindow, 23
GtkSeparator, 23
GtkSeparatorMenuItem, 23
GtkSeparatorToolItem, 23
GtkSettings, 23
GtkSettingsValue, 31
GtkSizeGroup, 23
GtkSocket, 23
GtkSpinButton, 23
GtkSpinner, 23
GtkStatusbar, 23
GtkStatusIcon, 23
GtkStockItem, 31
GtkStyle, 23
GtkTable, 23
GtkTargetEntry, 31
GtkTearoffMenu, 23
GtkTextBuffer, 23
GtkTextIter, 23
GtkTextMark, 23
GtkTextTag, 23
GtkTextTagTable, 23
GtkTextView, 19, 23, 26, 27
GtkTipsQuery, 23
GtkToggleAction, 24
GtkToggleActionEntry, 31
GtkToggleButton, 24
GtkToggleToolButton, 24
GtkToolbar, 24
GtkToolButton, 24
GtkToolGroup, 24
GtkToolItemGroup, 24
GtkToolPalette, 24
GtkToolShell, 24
GtkTooltip, 24
GtkTooltips, 24
GtkTreeModel, 24, 28, 29
GtkTreeModelFilter, 24
GtkTreeModelSort, 24
GtkTreeSelection, 24
GtkTreeSortable, 24
GtkTreeStore, 24
GtkTreeView, 24
GtkTreeViewColumn, 24
GtkUIManager, 24
GtkVBox, 24
<table>
<thead>
<tr>
<th>Class/Function</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>gtkvbuttonbox</td>
<td>24</td>
</tr>
<tr>
<td>gtkviewport</td>
<td>24</td>
</tr>
<tr>
<td>gtkvolumebutton</td>
<td>24</td>
</tr>
<tr>
<td>gtkvpaned</td>
<td>24</td>
</tr>
<tr>
<td>gtkvruler</td>
<td>24</td>
</tr>
<tr>
<td>gtkvscrollbar</td>
<td>24</td>
</tr>
<tr>
<td>gtkvseparator</td>
<td>24</td>
</tr>
<tr>
<td>gtkwidget</td>
<td>24</td>
</tr>
<tr>
<td>gtkwidgetclass</td>
<td>25</td>
</tr>
<tr>
<td>gtkwindow</td>
<td>15, 24</td>
</tr>
<tr>
<td>gobject</td>
<td>14</td>
</tr>
<tr>
<td>match.call</td>
<td>31</td>
</tr>
<tr>
<td>names.GObject</td>
<td>14</td>
</tr>
<tr>
<td>numeric_version</td>
<td>5</td>
</tr>
<tr>
<td>GType</td>
<td>7, 15, 16, 18, 19, 25, 31</td>
</tr>
<tr>
<td>gtypefromname</td>
<td>25</td>
</tr>
<tr>
<td>gtypegetancestors</td>
<td>25</td>
</tr>
<tr>
<td>gtypegetclass</td>
<td>25</td>
</tr>
<tr>
<td>gtypegetinterfaces</td>
<td>25</td>
</tr>
<tr>
<td>gtypegetpropinfo</td>
<td>14</td>
</tr>
<tr>
<td>gtypegetsignals</td>
<td>25</td>
</tr>
<tr>
<td>GVfs</td>
<td>12</td>
</tr>
<tr>
<td>GVolume</td>
<td>12</td>
</tr>
<tr>
<td>GVolumeMonitor</td>
<td>12</td>
</tr>
<tr>
<td>implements (assertions)</td>
<td>2</td>
</tr>
<tr>
<td>interface (GObject)</td>
<td>14</td>
</tr>
<tr>
<td>printCallbackID (GMainLoop)</td>
<td>13</td>
</tr>
<tr>
<td>print.enum (enums-and-flags)</td>
<td>8</td>
</tr>
<tr>
<td>print.flags (enums-and-flags)</td>
<td>8</td>
</tr>
<tr>
<td>print.GType (GType)</td>
<td>25</td>
</tr>
<tr>
<td>registerVirtuals (classes)</td>
<td>6</td>
</tr>
<tr>
<td>RGtk</td>
<td>27, 29</td>
</tr>
<tr>
<td>RGtkDataFrame</td>
<td>28</td>
</tr>
<tr>
<td>RGtkDataFrameAppendColumns</td>
<td>28</td>
</tr>
<tr>
<td>RGtkDataFrameAppendRows</td>
<td>28</td>
</tr>
<tr>
<td>RGtkDataFrameNew (RGtkDataFrame)</td>
<td>28</td>
</tr>
<tr>
<td>RGtkDataFrameSetFrame (RGtkDataFrame)</td>
<td>28</td>
</tr>
<tr>
<td>RGtkObject</td>
<td>27, 29</td>
</tr>
<tr>
<td>transparent-type</td>
<td>30</td>
</tr>
<tr>
<td>Pango</td>
<td>26, 27</td>
</tr>
<tr>
<td>pango-Bidirectional-Text</td>
<td>26</td>
</tr>
<tr>
<td>pango-Cairo-Rendering</td>
<td>26</td>
</tr>
<tr>
<td>pango-Coverage-Maps</td>
<td>26</td>
</tr>
<tr>
<td>pango-Fonts</td>
<td>26</td>
</tr>
<tr>
<td>pango-Glyph-Storage</td>
<td>26</td>
</tr>
<tr>
<td>pango-Layout-Objects</td>
<td>26</td>
</tr>
<tr>
<td>pango-Scripts-and-Languages</td>
<td>26</td>
</tr>
<tr>
<td>pango-Tab-Stops</td>
<td>26</td>
</tr>
<tr>
<td>pango-Text-Attributes</td>
<td>26</td>
</tr>
<tr>
<td>pango-Text-Processing</td>
<td>26</td>
</tr>
<tr>
<td>pango-Version-Checking</td>
<td>26</td>
</tr>
<tr>
<td>pango-Vertical-Text</td>
<td>26</td>
</tr>
<tr>
<td>PangoLayout</td>
<td>26, 27</td>
</tr>
<tr>
<td>PangoRectangle</td>
<td>31</td>
</tr>
<tr>
<td>PangoRenderer</td>
<td>26</td>
</tr>
<tr>
<td>parentHandler (classes)</td>
<td>6</td>
</tr>
</tbody>
</table>