Package ‘RGtk2’

February 19, 2015

Version 2.20.31
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
Encoding UTF-8
NeedsCompilation yes
Repository CRAN
Date/Publication 2014-07-19 16:30:55

R topics documented:

assertions ................................................................. 2
ATK ................................................................. 3
CAIRO ................................................................. 4
checkGTK ................................................................. 5
classes ................................................................. 6
enums-and-flags ........................................................ 8
GDK ................................................................. 9
GDK-Pixbuf ............................................................... 10
GIO ................................................................. 11
GMainLoop .............................................................. 13
GObject ............................................................... 14
GQuark ............................................................... 16
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

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>w</td>
<td>An object whose type is to be verified.</td>
</tr>
<tr>
<td>klass</td>
<td>The type the object is expected to be.</td>
</tr>
<tr>
<td>nullOk</td>
<td>Whether the object is allowed to be NULL.</td>
</tr>
<tr>
<td>critical</td>
<td>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.</td>
</tr>
<tr>
<td>obj</td>
<td>A GObject.</td>
</tr>
<tr>
<td>interface</td>
<td>The interface that obj is expected to implement.</td>
</tr>
</tbody>
</table>

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 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
http://library.gnome.org/devel//atk

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-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
checkGTK

**cairo-transformations**  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**


---

<table>
<thead>
<tr>
<th>checkGTK</th>
<th>Bound versions</th>
</tr>
</thead>
</table>

**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**

```c
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**

- **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 parentHandler(), arguments to pass to the parent method. For gClass(), 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 overriden 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
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` class retrieves a value as a flag. This 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

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-Bitmaps-and-Pixmaps** Offscreen drawables
**GDK-Pixbuf**

### GDK-Pixbuf

**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-Visicals** 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://library.gnome.org/devel/gdk](http://library.gnome.org/devel/gdk)

---

**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://library.gnome.org/devel/gdk-pixbuf](http://library.gnome.org/devel/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
- **GFileIOStream**  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

**GInputStream**  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**

http://library.gnome.org/devel//gio
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 Hinterval, 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 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
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

```r
 TypeInfo objectGet(obj, ..., drop = T)
## S3 method for class 'TypeInfo'
obj[|value, ...|
TypeInfo objectSet(obj, ...)
## S3 replacement method for class 'TypeInfo'
obj[[|propNames] <- value
## S3 method for class 'TypeInfo'
obj[[|member, where = parent.frame()]]
## S3 replacement method for class 'TypeInfo'
obj[[|member]] <- value
## S3 method for class 'TypeInfo'
x$member
## S3 replacement method for class 'TypeInfo'
obj$member <- value
TypeInfo object(type, ...)
TypeInfo objectNew(type, ...)
TypeInfo objectGetData(obj, key, data = NULL)
TypeInfo objectGetData(obj, key)
TypeInfo objectGetSignals(obj)
TypeInfo objectGetPropInfo(obj, parents = TRUE, collapse = TRUE)
TypeInfo getTypePropInfo(type)
## S3 method for class 'TypeInfo'
names(x)
interface(obj)
TypeInfo objectParentClass(obj)
```

Arguments

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>obj</td>
<td>an instance of a GObject</td>
</tr>
<tr>
<td>drop</td>
<td>when retrieving the value of a single property, TRUE to return the element from the list, instead of the list with just that element.</td>
</tr>
<tr>
<td>member</td>
<td>the name of a member in an R-defined (custom) GObject class</td>
</tr>
<tr>
<td>type</td>
<td>the type of GObject</td>
</tr>
<tr>
<td>key</td>
<td>the unique identifier under which the data is stored</td>
</tr>
</tbody>
</table>
GObject

<table>
<thead>
<tr>
<th>data</th>
<th>the data to store with the GObject</th>
</tr>
</thead>
<tbody>
<tr>
<td>...</td>
<td>named arguments of properties to set or names of properties to retrieve</td>
</tr>
<tr>
<td>propNames</td>
<td>the names properties to set or get</td>
</tr>
<tr>
<td>value</td>
<td>a value with which to set a property</td>
</tr>
<tr>
<td>parents</td>
<td>whether to include the parents when retrieving property info</td>
</tr>
<tr>
<td>collapse</td>
<td>whether to collapse the properties over the parents</td>
</tr>
<tr>
<td>x</td>
<td>The GObject for which the property names are to be retrieved</td>
</tr>
<tr>
<td>where</td>
<td>The environment in which to look for the field accessor function</td>
</tr>
</tbody>
</table>

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 GtkWidget). 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. `namesGObject` 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`

---

**GQuark**

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

```c
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

- **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 gsige
mit has to match the detail argument of the signal handler currently subject to invocation. Specifica
on 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 disconnect-
ing the handler.

gSignalGetInfo returns detailed information about a signal. The returned list contains the follow-
ing 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

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

See Also

GObject
Description

The GTK+ library itself contains widgets, that is, GUI components such as \texttt{GtkButton} or \texttt{GtkTextView}.

Details

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

- \texttt{chap-drawing-model} The GTK+ drawing model in detail
- \texttt{gtk-Filesystem-utilities} Functions for working with GIO
- \texttt{GtkAboutDialog} Display information about an application
- \texttt{gtk-Keyboard-Accelerators} Groups of global keyboard accelerators for an entire \texttt{GtkWindow}
- \texttt{GtkAccelLabel} A label which displays an accelerator key on the right of the text
- \texttt{gtk-Accelerator-Maps} Loadable keyboard accelerator specifications
- \texttt{GtkAccessible} Accessibility support for widgets
- \texttt{GtkAction} An action which can be triggered by a menu or toolbar item
- \texttt{GtkActionGroup} A group of actions
- \texttt{GtkActivatable} An interface for activatable widgets
- \texttt{GtkAdjustment} A \texttt{GtkObject} representing an adjustable bounded value
- \texttt{GtkAlignment} A widget which controls the alignment and size of its child
- \texttt{GtkArrow} Displays an arrow
- \texttt{GtkAspectFrame} A frame that constrains its child to a particular aspect ratio
- \texttt{GtkAssistant} A widget used to guide users through multi-step operations
- \texttt{GtkButtonBox} Base class for \texttt{GtkHButtonBox} and \texttt{GtkVButtonBox}
- \texttt{GtkBin} A container with just one child
- \texttt{GtkBox} Base class for box containers
- \texttt{gtk-gtkbuildable} Interface for objects that can be built by \texttt{GtkBuilder}
- \texttt{GtkBuilder} Build an interface from an XML UI definition
- \texttt{GtkButton} A widget that creates a signal when clicked on
- \texttt{GtkCalendar} Displays a calendar and allows the user to select a date
- \texttt{GtkCellEditable} Interface for widgets which can be used for editing cells
- \texttt{GtkCellLayout} An interface for packing cells
- \texttt{GtkCellRenderer} An object for rendering a single cell on a GdkDrawable
- \texttt{GtkCellRendererAccel} Renders a keyboard accelerator in a cell
- \texttt{GtkCellRendererCombo} Renders a combobox in a cell
- \texttt{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
GtkCellValue  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 forGtkEntry
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

**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 holdsGtkMenuItem 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
GtkPageSize  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
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
**GType**

- **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
- **GtkWidget** Toplevel which can contain other widgets
- **GtkWindowGroup** Limit the effect of grabs

**Author(s)**

Derived by RGtkGen from GTK+ documentation

**References**

http://library.gnome.org/devel/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
GType

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 (i.e., "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 `GObject`s 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

- [http://developer.gnome.org/doc/API/2.0/gobject/gobject-Type-Information.html](http://developer.gnome.org/doc/API/2.0/gobject/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

[http://library.gnome.org/devel//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**

```r
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 (i.e., 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

```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 `classNameGetField(tmp)`. External "methods" are expanded by the `$` operator to form `classNameMethodName(objectOfClassName, ...)`. The `[[` and `$` 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 GtkWidgetFileInfo GtkWidgetEntry AtkAttribute
GtkSettingsValue GClosure GType
GtkStockItem GtkWidgetFactoryEntry GtkWidgetAllocation GdkAtom GTimeVal
PangoRectangle GdkRectangle AtkAttributeSet GdkRgbCmap GdkKeymapKey GdkGCValues GdkGeometry
GdkPoint GdkSegment GdkColor GdkNativeWindow GdkError GdkWindowAttr GdkTrapezoid
GtkWidgetEntry GtkWidgetToggleActionEntry GtkWidgetRadioActionEntry CairoPath CairoGlyph
CairoPathData AtkTextRectangle AtkTextRange GdkSpan GdkTimeCoord
Index

*Topic interface
  ATK, \[3\]
  CAIRO, \[4\]
  checkGTK, \[5\]
  classes, \[6\]
  enums-and-flags, \[8\]
  GDK, \[9\]
  GDK-Pixbuf, \[10\]
  GIO, \[11\]
  GMainLoop, \[13\]
  GObject, \[14\]
  GSignal, \[16\]
  GTK, \[19\]
  GType, \[24\]
  Pango, \[26\]
  RGtk, \[27\]
  RGtkDataFrame, \[28\]
  RGtkObject, \[29\]
  transparent-type, \[30\]

*Topic misc
  assertions, \[2\]
  as\-.RGtkObject (RGtkObject), \[29\]
  as\-.enum (enums-and-flags), \[8\]
  as\-.GObject (GObject), \[14\]
  as\-.RGtkDataFrame (RGtkDataFrame), \[28\]
  as\-.flags (enums-and-flags), \[8\]
  as\-.RGtkObject (RGtkObject), \[29\]
  as\-.GObject (GObject), \[14\]
  as\-.RGtkObject (RGtkObject), \[29\]
  as\-.GObject (GObject), \[14\]
  as\-.flags (enums-and-flags), \[8\]
  as\-.data-frame (RGtkDataFrame)
  (RGtkDataFrame), \[28\]

as\-.flag (enums-and-flags), \[8\]
as\-.GQuark, \[16\]
as\-.struct (transparent-type), \[30\]
assertions, \[2\]
assignProp (classes), \[6\]
ATK, \[3, 27\]
 ATK-Misc, \[3\]
 ATK-State, \[3\]
 ATK-Action, \[3\]
 ATK-Attribute, \[31\]
 ATK-AttributeSet, \[31\]
 ATK-Component, \[3\]
 ATK-Document, \[3\]
 ATK-EditableText, \[3\]
 ATK-GObjectAccessible, \[3\]
 ATK-Hyperlink, \[3\]
 ATK-Hypertext, \[3\]
 ATK-Image, \[3\]
 ATK-NoObject, \[3\]
 ATK-NoObjectFactory, \[3\]
 ATK-Object, \[3\]
 ATK-ObjectFactory, \[3\]
 ATK-Registry, \[3\]
 ATK-Relation, \[3\]
 ATK-RelationSet, \[3\]
 ATK-Selection, \[3\]
 ATK-StateSet, \[3\]
 ATK-StreamableContent, \[4\]
 ATK-Table, \[4\]
 ATK-Text, \[4\]
 ATK-TextRange, \[31\]
 ATK-TextRectangle, \[31\]
 ATK-Util, \[4\]
 ATK-Value, \[4\]

boundCairoVersion (checkGTK), \[5\]
boundGTKVersion (checkGTK), \[5\]
boundPangoVersion (checkGTK), \[5\]

CAIRO, \[4\]
INDEX

Cairo, 27
cairo-context, 5
cairo-error-status, 4
cairo-font-face, 4
cairo-font-options, 4
cairo-image-surface, 4
cairo-matrix, 4
cairo-paths, 4
cairo-pattern, 4
cairo-pdf-surface, 4
cairo-png-functions, 4
cairo-ps-surface, 4
cairo-scaled-font, 4
cairo-surface, 4
cairo-svg-surface, 4
cairo-text, 4
cairo-transformations, 5
cairo-types, 5
cairo-user-font, 5
cairo-version-info, 5
CairoGlyph, 31
CairoPath, 31
CairoPathData, 31
chap-drawing-model, 19
checkArrType (assertions), 2
checkcairo (checkGTK), 5
checkGTK, 5
checkPango (checkGTK), 5
checkPtrType (assertions), 2
classes, 6
connectSignal (GSignal), 16
dim.RGtkDataFrame (RGtkDataFrame), 28
dimnames.RGtkDataFrame (RGtkDataFrame), 28
dimnames<-.RGtkDataFrame
(GGtkDataFrame), 28
enums-and-flags, 8
GAppInfo, 11
GAsyncInitable, 11
GAsyncResult, 11
GBufferedInputStream, 11
GBufferedOutputStream, 11
GCancellable, 11
gClass (classes), 6
GClosure, 31
GConnectFlags (GSignal), 16
GDataInputStream, 11
GDataOutputStream, 11
GDK, 9, 27
gdk-Application-launching, 9
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, 10
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, 27
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, 10
gdk-Properties-and-Atoms, 10
gdk-Testing, 9
gdk-Visuals, 10
gdk-Windows, 10
GdkAtom, 31
GdkColor, 31
GdkDisplay, 9
GdkDisplayManager, 9
GdkGCValues, 31
GdkGeometry, 31
GdkKeymapKey, 31
GdkNativeWindow, 31
GDKPixbuf, 27
GdkPixbuf, 10, 27
GdkPixbufLoader, 10
GdkPoint, 31
gtk-graphics-contexts, 21
gtk-gtkbuildable, 19
gtk-gtkcheckmenuitem, 20
gtk-gtkfilefilter, 20
gtk-GtkTreeView-drag-and-drop, 23
gtk-high-level-printing-API, 22
gtk-keyboard-accelerators, 19
gtk-orientable, 22
gtk-resource-files, 22
gtk-selections, 23
gtk-standard-enumerations, 20
gtk-stock-items, 23
gtk-testing, 23
gtk-themeable-stock-images, 21
GtkAboutDialog, 19
GtkAccellLabel, 19
GtkAccessible, 19
GtkAction, 19
GtkActionEntry, 31
GtkActionGroup, 19
GtkActivatable, 19
GtkAdjustment, 19
GtkAlignment, 19
GtkAllocation, 31
GtkArrow, 19
GtkAspectFrame, 19
GtkAssistant, 19
GtkBin, 19
GtkBox, 19
GtkBuilder, 19
GtkButton, 19, 27
GtkButtonBox, 19
GtkCalendar, 19
GtkCellEditable, 19
GtkCellLayout, 19
GtkCellRenderer, 19
GtkCellRendererAccel, 19
GtkCellRendererCombo, 19
GtkCellRendererPixbuf, 19
GtkCellRendererProgress, 20
GtkCellRendererSpin, 20
GtkCellRendererSpinner, 20
GtkCellRendererText, 20
GtkCellRendererToggle, 20
GtkCellValue, 20
GtkCheckButton, 20
GtkClist, 20
GtkColorButton, 20
GtkColorSelection, 20
GtkColorSelectionDialog, 20
GtkCombo, 20
GtkComboBox, 20
GtkComboBoxEntry, 20
GtkContainer, 20
GtkCTree, 20
GtkCurve, 20
GtkDialog, 20
GtkDrawingArea, 20
GtkEditable, 20
GtkEntry, 20, 26, 27
GtkEntryBuffer, 20
GtkEntryCompletion, 20
GtkEventBox, 20
GtkExpander, 20
GtkFileChooser, 20
GtkFileChooserButton, 20
GtkFileChooserDialog, 20
GtkFileChooserWidget, 20
GtkFileFilterInfo, 31
GtkFileSelection, 20
GtkFixed, 20
GtkFontButton, 20
GtkFontSelection, 21
GtkFontSelectionDialog, 21
GtkFrame, 21
GtkGammaCurve, 21
GtkHandleBox, 21
GtkHBox, 21
GtkHButtonBox, 21
GtkHRaned, 21
GtkHRuler, 21
GtkHScale, 21
GtkHScrollbar, 21
GtkHSeparator, 21
GtkHSV, 21
GtkIconTheme, 21
GtkIconView, 21
GtkImage, 10, 21, 27
GtkImageMenuitem, 21
GtkIMContext, 21
GtkIMContextSimple, 21
GtkIMMulticontext, 21
GtkInfoBar, 21
GtkInputDialog, 21
GtkInvisible, 21
GtkItem, 21
GtkVolumeButton, 24
GtkVPaned, 24
GtkVSlider, 24
GtkVScale, 24
GtkVScrollbar, 24
GtkVSeparator, 24
GtkWidget, 24
GtkWidgetClass, 25
GtkWidgetClass, 25
GtkWindow, 15, 24
GtkWidgetGroup, 24
printenums (enums-and-flags), 8
printflag (enums-and-flags), 8
printflags (enums-and-flags), 8
printGType (GType), 24
registerVirtuals (classes), 6
RGtk, 27, 29
RGtkDataFrame, 28
rgtkDataFrame (RGtkDataFrame), 28
rgtkDataFrameAppendColumns
(RGtkDataFrame), 28
rgtkDataFrameAppendRows
(RGtkDataFrame), 28
rgtkDataFrameNew (RGtkDataFrame), 28
rgtkDataFrameSetFrame (RGtkDataFrame), 28
RGtkObject, 27, 29
transparent-type, 30

implements (assertions), 2
interface (GObject), 14
match.call, 31
names (GObject), 14
numeric_version, 5

Pango, 26, 27
pango-Bidirectional-Text, 26
pango-Cairo-Rendering, 26
pango-Coverage-Maps, 26
pango-Fonts, 26
pango-Glyph-Storage, 26
pango-Layout-Objects, 26
pango-Scripts-and-Languages, 26
pango-Tab-Stops, 26
pango-Text-Attributes, 26
pango-Text-Processing, 26
pango-Version-Checking, 26
pango-Vertical-Text, 26
PangoLayout, 26, 27
PangoRectangle, 31
PangoRenderer, 26
parentHandler (classes), 6
printCallbackID (GMainLoop), 13
printenum (enums-and-flags), 8