Package 'rkwarddev'

October 19, 2014

Type Package

Title A collection of tools for RKWard plugin development

Author m.eik michalke <meik.michalke@hhu.de>

Maintainer m.eik michalke <meik.michalke@hhu.de>

Depends R (>= 2.9.0),methods,XiMpLe (>= 0.03-21),rkward (>= 0.5.7)

Enhances rkward

Description Provides functions to create plugin skeletons and XML structures for RKWard.

License GPL (>= 3)

Encoding UTF-8

LazyLoad yes

URL http://rkward.sourceforge.net

Version 0.06-5

Date 2014-10-19

Collate '00_class_01_rk.JS.arr.R' '00_class_02_rk.JS.var.R' '00_class_03_rk.JS.ite.R'
'00_class_04_rk.JS.opt.R' '00_class_05_rk.JS.oset.R' 'rk.rkh.doc.R' 'rk.XML.plugin.R'
'rk.JS.doc.R' '00_class_06_rk.plug.comp.R' 'echo.R' 'i18n.R' 'id.R' 'ite.R' 'join.R' 'qp.R'
'rk.comment.R' 'rk-internal.R' 'rk.JS.array.R' 'rk.JS.options.R' 'rk.JS.optionset.R' 'rk.JS.saveobj.R'
'rk.JS.scan.R' 'rk.JS.vars.R' 'rk.XML.about.R' 'rk.XML.attribute.R' 'rk.XML.browser.R'
'rk.XML.cbox.R' 'rk.XML.code.R' 'rk.XML.col.R' 'rk.XML.component.R'
'rk.XML.components.R' 'rk.XML.connect.R' 'rk.XML.context.R' 'rk.XML.convert.R'
'rk.XML.copy.R' 'rk.XML.dependencies.R' 'rk.XML.dependency_check.R' 'rk.XML.dialog.R'
'rk.XML.dropdown.R' 'rk.XML.embed.R' 'rk.XML.entry.R' 'rk.XML.external.R'
'rk.XML.formula.R' 'rk.XML.frame.R' 'rk.XML.help.R' 'rk.XML.hierarchy.R'
'rk.XML.include.R' 'rk.XML.input.R' 'rk.XML.insert.R' 'rk.XML.logic.R' 'rk.XML.matrix.R'
'rk.XML.menu.R' 'rk.XML.option.R' 'rk.XML.optioncolumn.R' 'rk.XML.optiondisplay.R'
'rk.XML.optionset.R' 'rk.XML.page.R' 'rk.XML.pluginmap.R' 'rk.XML.preview.R'
'rk.XML.radio.R' 'rk.XML.require.R' 'rk.XML.row.R' 'rk.XML.saveobj.R' 'rk.XML.select.R'

'rk.XML.set.R' 'rk.XML.snippet.R' 'rk.XML.snippets.R' 'rk.XML.spinbox.R' 'rk.XML.stretch.R'
'rk.XML.switch.R' 'rk.XML.tabbook.R' 'rk.XML.text.R' 'rk.XML.values.R'
'rk.XML.valueselector.R' 'rk.XML.valueslot.R' 'rk.XML.vars.R' 'rk.XML.varselector.R'
'rk.XML.varslot.R' 'rk.XML.wizard.R' 'rk.build.plugin.R' 'rk.get.comp.R' 'rk.get.language.R'
'rk.get.rkh.prompter.R' 'rk.paste.JS.R' 'rk.paste.JS.graph.R' 'rk.plotOptions.R'
'rk.plugin.component.R' 'rk.plugin.skeleton.R' 'rk.rkh.caption.R' 'rk.rkh.link.R' 'rk.rkh.related.R'
'rk.rkh.scan.R' 'rk.rkh.section.R' 'rk.rkh.setting.R' 'rk.rkh.settings.R' 'rk.rkh.summary.R'
'rk.rkh.technical.R' 'rk.rkh.title.R' 'rk.rkh.usage.R' 'rk.set.comp.R' 'rk.set.language.R'
'rk.set.rkh.prompter.R' 'rk.testsuite.doc.R' 'rk.uniqueIDs.R' 'rkwarddev-desc-internal.R'
'rkwarddev-package.R' 'rkwarddev.required.R' 'show-methods.R' 'tf.R' 'zzz.rk.plot.opts-class.R'

R topics documented:

`rkwarddev-package` *The rkwarddev Package*

Description

A collection of tools for RKWard plugin development.

Details

Package:	rkwarddev
Type:	Package
Version:	0.06-5
Date:	2014-10-19
Depends:	R (>= 2.9.0),methods,XiMpLe (>= 0.03-21),rkward (>= 0.5.7)
Enhances:	rkward
Encoding:	UTF-8
License:	GPL (>= 3)
LazyLoad:	yes
URL:	http://rkward.sourceforge.net

Provides functions to create plugin skeletons and XML structures for RKWard.

Author(s)

Meik Michalke

echo *Generate JavaScript echo command call*

Description

This function will take several elements, either character strings, or objects of class `XiMpLe.node` which hold an XML node of some plugin GUI definition, or objects of classes `rk.JS.arr` or `rk.JS.opt`. From those, it will generate a ready-to-run JavaScript `echo();` call from it.

Usage

```
echo(..., newline = "")
```

Arguments

`...`	One or several character strings and/or `XiMpLe.node` objects with plugin nodes, and/or objects of classes `rk.JS.arr` or `rk.JS.opt`, simply separated by comma.
`newline`	Character string, can be set to e.g. `"\n"` to force a newline after the call.

Value

A character string.

See Also

`rk.JS.vars`, `rk.JS.array`, `rk.JS.options`, `ite`, `id`, `qp`, and the Introduction to Writing Plugins for RKWard

Examples

```
cbox1 <- rk.XML.cbox(label="foo", value="foo1", id.name="CheckboxFoo.ID")
echo("bar <- \"", cbox1, "\"")
```

i18n *Translate parts of a plugin*

Description

Takes a list of entries named after abbreviated languages, and returns either the one matching the language set with `rk.set.language`, or the first entry if no language was set at all or the set language cannot be found in `...`.

Usage

```
i18n(..., lang = rk.get.language())
```

Arguments

...	Comma separated, named elements, see description.
lang	Character string, the language to return.

Details

If used in an rkwarddev script, this can be used to toggle the generation of plugins in a certain language.

Examples

```
rk.set.language("en", c("en_EN", "en_US"))
(var.select <- rk.XML.varselector(label=i18n(en="Select data", de="Wähle Daten")))

# now try the same with the alternate language
rk.set.language("de", "de_DE")
(var.select <- rk.XML.varselector(label=i18n(en="Select data", de="Wähle Daten")))
```

---

id *Replace XiMpLe.node objects with their ID value*

Description

This function is intended to be used for generating JavaScript code for RKWard plugins. Its sole purpose is to replace objects of class XiMpLe.node which hold an XML node of some plugin GUI definition, and objects of classes rk.JS.arr, rk.JS.opt or rk.JS.var with their ID (or JS variable name), and combine these replacements with character strings.

Usage

```
id(..., quote = FALSE, collapse = "", js = TRUE)
```

Arguments

...	One or several character strings and/or XiMpLe.node objects with plugin nodes, and/or objects of classes rk.JS.arr, rk.JS.opt or rk.JS.var, simply separated by comma.
quote	Logical, it the character strings sould be deparsed, so they come out "as-is" when written to files, e.g. by cat.
collapse	Character string, defining if and how the individual elements should be glued together.
js	Logical, if TRUE returns JavaScript varaible names for XiMpLe.node objects. Otherwise their actual ID is returned.

Value

A character string.

See Also

rk.JS.vars, rk.JS.array, rk.JS.options, echo, qp (a shortcut for id with different defaults), and the Introduction to Writing Plugins for RKWard

Examples

```
# an example checkbox XML node
cbox1 <- rk.XML.cbox(label="foo", value="foo1", id.name="CheckboxFoo.ID")
id("The variable name is: ", cbox1, "!")
```

---

ite *Generate JavaScript if/then/else constructs*

Description

Generate JavaScript if/then/else constructs

Usage

```
ite(ifjs, thenjs, elsejs = NULL)
```

Arguments

ifjs	Either a character string to be placed in the brackets if an if() statement, or an object of class XiMpLe.node. rk.JS.arr or rk.JS.opt (whose identifier will be used).
thenjs	Either a character string, the code to be executed in case the if() statement is true, or an object of class XiMpLe.node. rk.JS.arr or rk.JS.opt (whose identifier will be used). The latter is especially useful in combination with rk.JS.options. You can also give another object of class rk.JS.ite.
elsejs	Like thenjs, the code to be executed in case the if() statement is not true.

Value

An object of class rk.JS.ite

See Also

rk.paste.JS, rk.JS.vars, rk.JS.array, rk.JS.options, echo, id, qp, and the Introduction to Writing Plugins for RKWard

Examples

```
# first create an example checkbox XML node
cbox1 <- rk.XML.cbox(label="foo", value="foo1", id.name="CheckboxFoo.ID")
# now some JavaScript generation
ite(cbox1, echo("bar <- \"", cbox1, "\""), echo("bar <- NULL"))
```

join *Generate JavaScript to join an array object*

Description

This function pastes an object of class `rk.JS.arr` similar to `rk.paste.JS`, but was specifically written for elements like `<optionset>` or `<matrix>`, whose values must be queried by `getList()` rather than `getValue()`. This means, the resulting variable is already an array an merely needs to be joined in as R code output (e.g., an `<optioncolumn>`).

Usage

```
join(var, by = "\", \"")
```

Arguments

`var`	Either a character string (the name of the variable to combine to a vector or list), or an object of class `XiMpLe.node` (whose ID will be extracted and used). Also accepts objects of class `rk.JS.arr`.
`by`	Character string by which the values ought to be joined.

Value

An object of class `rk.JS.echo`.

See Also

`rk.paste.JS`, `rk.JS.options`, `rk.JS.vars`, `echo`, `id`, and the Introduction to Writing Plugins for RKWard

qp *Replace XiMpLe.node objects with their ID value*

Description

This function is a shortcut for `id` which sets some useful defaults (`quote=TRUE, collapse=" + ", js=TRUE`). The abbreviation stands for "quote + plus".

Usage

```
qp(...)
```

Arguments

`...`	One or several character strings and/or `XiMpLe.node` objects with plugin nodes, and/or objects of classes `rk.JS.arr` or `rk.JS.opt`, simply separated by comma.

Value

A character string.

See Also

rk.JS.vars, rk.JS.array, rk.JS.options, echo, id, and the Introduction to Writing Plugins for RKWard

Examples

```
# an example checkbox XML node
cbox1 <- rk.XML.cbox(label="foo", value="foo1", id.name="CheckboxFoo.ID")
qp("The variable name is: ", cbox1, "!")
```

---

rk.build.plugin *Build an RKWard plugin package*

Description

Build an RKWard plugin package

Usage

```
rk.build.plugin(plugin, check = FALSE, install = FALSE, R.libs = NULL)
```

Arguments

plugin	A character string, path to the plugin package root directory (hint: it's the directory with the DESCRIPTION file in it).
check	Logical, whether the package should be checked for errors. Always do this before you publish a package!
install	Logical, whether the built package should also be installed locally.
R.libs	A character string, path to local R packages, used by install to figure out where to install to.

See Also

Introduction to Writing Plugins for RKWard

Examples

```
## Not run:
plugin.dir <- rk.plugin.skeleton("MyPlugin", dialog=full.dialog, wizard=full.wizard)
rk.build.plugin(plugin.dir, R.libs="~/R", check=TRUE)

## End(Not run)
```

rk.comment *Create comment for RKWard plugin code*

Description

Create comment for RKWard plugin code

Usage

```
rk.comment(text)
```

Arguments

text	Character string, the text to be displayed.

Value

An object of class XiMpLe.node.

Examples

```
test.comment <- rk.comment("Added this text.")
cat(pasteXML(test.comment))
```

rk.get.comp *Get the name of the component you're currently working on*

Description

This is the "get"-equivalent to rk.set.comp. It is used by functions like, e.g., rk.XML.cbox, to add text for .rkh pages automatically to the current plugin component.

Usage

```
rk.get.comp()
```

rk.get.language *Get plugin language for internationalisation*

Description

Get plugin language for internationalisation

Usage

```
rk.get.language(locales = FALSE)
```

Arguments

locales	Logical, whether to query for language or locales set.

Examples

```
rk.get.language()
```

rk.get.rkh.prompter *Get .rkh related information stored internally*

Description

Get .rkh related information stored internally

Usage

```
rk.get.rkh.prompter(component = NULL, id = NULL)
```

Arguments

component	Character string, the name under which you stored information. If NULL, returns all information stored in the internal rkh.prompter list.
id	Character string, the node ID if a given component to search for. If NULL, returns the full list of the given component, otherwise only the help information for the requested node.

Examples

```
rk.get.rkh.prompter("rk.myPlugin", "someID")
```

rk.JS.array *Create a simple JavaScript array*

Description

If you need to combine multiple options (like values of several checkboxes) into one vector or list, this function can help with that task. All relevant variables will become part of an array and then joined into the desired argument type.

Usage

```
rk.JS.array(option, variables = list(), funct = "c", var.prefix = NULL,
  quote = FALSE)
```

Arguments

option	A character string, naming, e.g., an option of an R function which should be constructed from several variables.
variables	A list with either character strings (the names of the variables to combine to a vector or list), or objects of class XiMpLe.node with plugin XML nodes (whose ID will be extracted and used).
funct	Character string, name of the R function to be called to combine the options, e.g. "list" for list(), or "c" for c().
var.prefix	A character string. sets a global string to be used as a prefix for the JS variable names.
quote	Logical, if TRUE, the values will be quoted in the resulting R code (might be neccessary for character values).

Value

An object of class rk.JS.arr.

See Also

rk.paste.JS, rk.JS.options, rk.JS.vars, echo, id, and the Introduction to Writing Plugins for RKWard

Examples

```
# create three checkboxes for independent options
checkA <- rk.XML.cbox(label="Run Test A", value="A")
checkB <- rk.XML.cbox(label="Run Test B", value="B")
checkC <- rk.XML.cbox(label="Run Test C", value="C")
# combine them into one list of options via JavaScript
rk.JS.array("run.tests", variables=list(checkA, checkB, checkC), funct="list")
```

rk.JS.doc *Create JavaScript outline from RKWard plugin XML*

Description

Create JavaScript outline from RKWard plugin XML

Usage

```
rk.JS.doc(require = c(), variables = NULL, globals = NULL,
  results.header = NULL, preprocess = NULL, calculate = NULL,
  printout = NULL, doPrintout = NULL, load.silencer = NULL,
  gen.info = TRUE, indent.by = "\t")
```

Arguments

require	A character vector with names of R packages that the dialog depends on.
variables	Either a character string to be included to read in all needed variables from the dialog (see rk.JS.scan), or an object of class rk.JS.var which will be coerced into character. These variables will be defined in the calculate() and/or doPrintout() functions.
globals	Like variables, but these variables will be defined globally. If variables is set as well, the function tries to remove duplicate definitions.
results.header	A character string to headline the printed results. Include escapes quotes (\") if needed. Set to FALSE or "" if you need more control and want to define the header section in printout.
preprocess	A character string to be included in the preprocess() function. This string will be pasted as-is, after require has been evaluated.
calculate	A character string to be included in the calculate() function. This string will be pasted as-is, after variables has been evaluated.
printout	A character string to be included in the printout() function. This string will be pasted as-is, after results.header has been evaluated. Ignored if doPrintout is set.
doPrintout	A character string to be included in the doPrintout() function. This string will be pasted as-is. You don't need to define a preview() function, as this will be added automatically. Use ite("full", ...) style JavaScript code to include headers etc.
load.silencer	Either a character string (ID of probably a checkbox), or an object of class XiMpLe.node. This defines a switch you can add to your plugin, to set the require() call inside suppressMessages(), hence suppressing all load messages (except for warnings and errors) of required packages in the output.
gen.info	Logical, if TRUE a comment note will be written into the document, that it was generated by rkwarddev and changes should be done to the script.
indent.by	A character string defining how indentation should be done.

Value

A character string.

Note

The JavaScript

See Also

rk.paste.JS, rk.JS.vars, rk.JS.array, ite, echo, id, and the Introduction to Writing Plugins for RKWard

---

rk.JS.options *Combine several options in one JavaScript variable*

Description

Combine several options in one JavaScript variable

Usage

```
rk.JS.options(var, ..., collapse = ", ", option = NULL, funct = NULL,
  array = TRUE)
```

Arguments

var	Character string, name of the JavaScript variable to use in the script code.
...	A list of objects of class rk.JS.ite (see ite). Use the thenjs element to define only the value to add to the option, without commas (e.g., "paired=TRUE" or qp("conf.level=\"", conflevel, "\"")).
collapse	Character string, how all options should be concatenated on the R code level (if array=FALSE), or how option should be added to the generated R code. Hint: To place each option in a new line with tab indentation, set collapse=",\n\t".
option	A character string, naming, e.g., an option of an R function which should be constructed from several variables. Only used if array=TRUE.
funct	Character string, name of the R function to be called to combine the options, e.g. "list" for list(), or "c" for c(). Set to NULL to drop. Only used if array=TRUE.
array	Logical, if TRUE will generate the options as an array, otherwise in one concatenated character string (probably only useful for mandatory options).

Value

An object of class rk.JS.opt, use rk.paste.JS on that.

See Also

rk.JS.array, and the Introduction to Writing Plugins for RKWard

Examples

```
# create two checkboxes for independent options
checkA <- rk.XML.cbox(label="Run Test A", value="A")
checkB <- rk.XML.cbox(label="Run it fast", value="true")
# combine them into one JavaScript options variable
rk.JS.options("optionsTestA",
  ite(checkA, qp("test=\"", checkA, "\"")),
  ite(checkB, "fast=TRUE")
)
```

---

rk.JS.optionset *Evaluate optionset objects in plugin JavaScript*

Description

This function scans an object generated by rk.XML.optionset, extract IDs of all optioncolumn objects and nest the JavaScript code you define via ... inside a for loop that iterates through all columns. Inside ..., you can use the column objects of rk.XML.optioncolumn to refer to the respective column, rk.JS.optionset will use appropriate variables.

Usage

```
rk.JS.optionset(optionset, ..., loopvar = "i", collapse = ",\\n\\t",
  vars = FALSE, guess.getter = TRUE)
```

Arguments

optionset	A XiMpLe.node object, the full <optionset> node.
...	The JavaScript code, optionally including the optioncolumn objects. This will become the body of the for loop.
loopvar	Character, name of the index variable used in the for loop.
collapse	Character string, how all optioncolumns should be concatenated on the R code level Hint: To place each one on a new line with tab indentation, set collapse=", \n\t".
vars	Logical, if TRUE all optioncolumn varaibles will be defined first. This is not needed if rk.JS.scan was already called.
guess.getter	Logical, if TRUE try to get a good default getter function for JavaScript variable values. Only relevant if vars=TRUE.

Details

In case you simply want to define the variables, but not run the loop yet, set vars=TRUE and leave ... empty.

See Also

rk.XML.optionset, rk.XML.optioncolumn

Examples

```
# this example is taken from the plugin skeleton script
# first set up an optionset object
dep.optionset.packages <- rk.XML.optionset(
  content=rk.XML.frame(rk.XML.stretch(before=list(
    rk.XML.row(
      dep.pckg.name <- rk.XML.input("Package"),
      dep.pckg.min <- rk.XML.input("min"),
      dep.pckg.max <- rk.XML.input("max"),
      dep.pckg.repo <- rk.XML.input("Repository")
    )
  )), label="Depends on R packages"),
  optioncolumn=list(
    dep.optioncol.pckg.name <- rk.XML.optioncolumn(connect=dep.pckg.name,
      modifier="text"),
   dep.optioncol.pckg.min <- rk.XML.optioncolumn(connect=dep.pckg.min, modifier="text"),
   dep.optioncol.pckg.max <- rk.XML.optioncolumn(connect=dep.pckg.max, modifier="text"),
   dep.optioncol.pckg.repo <- rk.XML.optioncolumn(connect=dep.pckg.repo, modifier="text")
  )
)

# now translate it to JavaScript for loop
JS.optionset <- rk.JS.optionset(dep.optionset.packages,
  echo("c("),
  echo("name=\"", dep.optioncol.pckg.name, "\""),
  ite(dep.optioncol.pckg.min, echo(", min=\"", dep.optioncol.pckg.min, "\"")),
  ite(dep.optioncol.pckg.max, echo(", max=\"", dep.optioncol.pckg.max, "\"")),
  ite(dep.optioncol.pckg.repo, echo(", repository=\"", dep.optioncol.pckg.repo, "\"")),
  echo(")")
)
```

---

rk.JS.saveobj *Create JavaScript saveobject code from plugin XML*

Description

Create JavaScript saveobject code from plugin XML

Usage

```
rk.JS.saveobj(pXML, R.objects = "initial", vars = TRUE,
  add.abbrev = FALSE, indent.by = "\t")
```

Arguments

pXML	Either an object of class XiMpLe.doc or XiMpLe.node, or path to a plugin XML file.
R.objects	Character vector, the names of the internal R objects to be saved. If not empty must have the same length as <saveobject> nodes in the document, or be the keyword "initial", in which case the intital attribute values of the nodes are used.
vars	Logocal, whether the variables needed should also be defined in the JavaScript code.
add.abbrev	Logical, if TRUE the JavaScript variables will all have a prefix with an three letter abbreviation of the XML tag type to improve the readability of the code. But it's probably better to add this in the XML code in the first place.
indent.by	Character string used to indent each entry if js=TRUE.

Value

A character vector.

See Also

Introduction to Writing Plugins for RKWard

---

rk.JS.scan *Create JavaScript variables from plugin XML*

Description

Create JavaScript variables from plugin XML

Usage

```
rk.JS.scan(pXML, js = TRUE, add.abbrev = FALSE, guess.getter = FALSE,
  indent.by = "\t")
```

Arguments

pXML	Either an object of class XiMpLe.doc or XiMpLe.node, or path to a plugin XML file.
js	Logical, if TRUE usable JavaScript code will be returned, otherwise a character vector with only the relevant ID names.
add.abbrev	Logical, if TRUE the JavaScript variables will all have a prefix with an three letter abbreviation of the XML tag type to improve the readability of the code. But it's probably better to add this in the XML code in the first place.

guess.getter	Logical, if TRUE try to get a good default getter function for JavaScript variable values. This will use some functions which were added with RKWard 0.6.1, and therefore raise the dependencies for your plugin/component accordingly. Nonetheless, it's recommended.
indent.by	Character string used to indent each entry if js=TRUE.

Value

A character vector.

See Also

Introduction to Writing Plugins for RKWard

`rk.JS.vars` *Define variables in JavaScript code*

Description

Define variables in JavaScript code

Usage

```
rk.JS.vars(..., var.prefix = NULL, modifiers = NULL, default = FALSE,
  join = "", check.modifiers = TRUE, getter = "getValue",
  guess.getter = FALSE)
```

Arguments

...	Either one or more character strings (the names of the variables to define), or objects of class XiMpLe.node with plugin XML nodes (whose ID will be extracted and used).
var.prefix	A character string. will be used as a prefix for the JS variable names.
modifiers	A character vector with modifiers you'd like to apply to the XML node property.
default	Logical, if TRUE the default value (no special modifier) of the node will also be defined. Does nothing if modifiers=NULL.
join	A character string, useful for GUI elements which accept multiple objects (e.g., multi-varslots). If join is something other than "", these objects will be collapsed into one string when pasted, joined by this string.
check.modifiers	Logical, if TRUE the given modifiers will be checked for validity. Should only be turned off if you know what you're doing.
getter	A character string, naming the JavaScript function which should be used to get the values in the actual plugin. Depending on the XML element, "getString", "getBool" or "getList" can be useful alternatives. For backwards compatibility, the default is set to "getValue".
guess.getter	Logical, if TRUE try to get a good default getter function for JavaScript variable values.

Value

An object of class rk.JS.var.

Note

To get a list of the implemented modifiers in this package, call rkwarddev:::all.valid.modifiers.

See Also

rk.JS.array, echo, id, and the Introduction to Writing Plugins for RKWard

Examples

```
# create three checkboxes
checkA <- rk.XML.cbox(label="Run Test A", value="A")
checkB <- rk.XML.cbox(label="Run Test B", value="B")
checkC <- rk.XML.cbox(label="Run Test C", value="C")
# define them by their ID in JavaScript
cat(rk.paste.JS(rk.JS.vars(list(checkA, checkB, checkC))))
```

---

rk.paste.JS *Paste JavaScript objects and character strings*

Description

Paste JavaScript objects and character strings

Usage

```
rk.paste.JS(..., level = 2, indent.by = "\t", funct = NULL,
  array = NULL, var.prefix = NULL, modifiers = NULL, default = NULL,
  join = NULL, getter = NULL, empty.e = FALSE)
```

Arguments

...	Objects of class rk.JS.ite, rk.JS.arr, rk.JS.opt, rk.JS.oset or character. Another special case is XiMpLe nodes created by rk.comment(), which will be turned into JavaScript comments (i.e., lines starting with "//").
level	Integer, which indentation level to use, minimum is 1.
indent.by	A character string defining the indentation string to use.
funct	For rk.JS.arr and rk.JS.opt objects only: Character string, name of the R function to be called to combine the options, e.g. "list" for list(), or "c" for c().
array	For rk.JS.opt objects only: Logical, whether the options should be collected in an array or a concatenated character string.
var.prefix	For rk.JS.var objects only: A character string. will be used as a prefix for the JS variable names.

`modifiers` For `rk.JS.var` objects only: A character vector with modifiers you'd like to apply the XML node's property.

`default` For `rk.JS.var` objects only: Logical, if `TRUE` the default value (no special modifier) of the node will also be defined. Does nothing if `modifiers=NULL`.

`join` For `rk.JS.var` objects only: A character string, useful for GUI elements which accept multiple objects (e.g., multi-varslots). If `join` is something other than `""`, these objects will be collapsed into one string when pasted, joined by this string.

`getter` For `rk.JS.var` objects only: A character string, naming the JavaScript function which should be used to get the values in the actual plugin. Depending on the XML element, `"getString"`, `"getBool"` or `"getList"` can be useful alternatives. For backwards compatibility, the default is set to `"getValue"`.

`empty.e` For `rk.JS.ite` objects only: Logical, if `TRUE` will force to add empty `else {}` brackets when there is no `else` statement defined, which is considered to enhance code readability by some.

Value

A character string.

Note

To get a list of the implemented modifiers in this package, call `rkwarddev:::all.valid.modifiers`.

See Also

`rk.JS.array`, `rk.JS.options`, `rk.JS.optionset`, `rk.JS.vars`, `ite`, and the Introduction to Writing Plugins for RKWard

---

rk.paste.JS.graph *Paste simple JavaScript plot code*

Description

This function is similar to `rk.paste.JS`, but adds some code parts to its output which are commonly used to generate plots with RKWard.

Usage

```
rk.paste.JS.graph(..., plotOpts = NULL, printoutObj = NULL, level = 2,
  indent.by = "\t", empty.e = FALSE)
```

Arguments

...	The actual plot code, passed through to rk.paste.JS.
plotOpts	An object generated by rk.XML.embed or rk.plotOptions, i.e. embedded plot options.
printoutObj	An rk.JS.var object fetching the "code.printout" modifier of plotOpts (see examples below!). If NULL and plotOpts is of class rk.plot.opts (as returned by rk.plotOptions), will be fetched from plotOpts automatically.
level	Integer, which indentation level to use, minimum is 1.
indent.by	A character string defining the indentation string to use.
empty.e	For rk.JS.ite objects only: Logical, if TRUE will force to add empty else {} brackets when there is no else statement defined, which is considered to enhance code readability by some.

Details

The contents of the ... argument are evaluated by rk.paste.JS and encapsulated between if(full){rk.graph.on()} try and }) if(full){rk.graph.off()}. If generic plot options are supplied, their "code.preprocess" and "code.calculate" modifiers are also automatically taken care of, so you only need to include "code.printout" inside of ....

Value

A character string.

See Also

rk.paste.JS

Examples

```
tmp.var.selectVars <- rk.XML.varselector(label="Select data")
tmp.var.x <- rk.XML.varslot(label="My data", source=tmp.var.selectVars, required=TRUE)
# let this be the embedded generic plot options in yout plot dialog
tmp.plot.options <- rk.plotOptions()

# you can now generate the plot code using generic plot options
js.prnt <- rk.paste.JS.graph(
 echo("\t\tplot("),
 echo("\n\t\t\tx=", tmp.var.x),
 echo(tmp.plot.options),
 echo(")"),
 plotOpts=tmp.plot.options)

cat(js.prnt)
```

`rk.plotOptions` *Create an object for plot options in RKWard plugins*

Description

Generates XML and JavaScript code snippets by calling `rk.XML.embed` and `rk.JS.vars` with useful presets. The resulting object can be used inside the dialog XML object (to place the plot options button and disable certain tabs), as well as in the JS object (to then insert the actual plot options).

Usage

```
rk.plotOptions(label = "Generic plot options",
  embed = "rkward::plot_options", button = TRUE, id.name = "auto")
```

Arguments

`label`	A character string, text label for the button (only used if `button=TRUE`).
`embed`	A character string, registered name (`id` in pluginmap file) of the plot options component to be embedded.
`button`	Logical, whether the plot options should be embedded as a button and appear if it's pressed.
`id.name`	Character string, a unique ID for this plugin element. If `"auto"`, an ID will be generated automatically from the label and component strings.

Value

An object of class `rk.plot.opts`.

See Also

Introduction to Writing Plugins for RKWard

Examples

```
test.plotOptions <- rk.plotOptions()

# see how differently this object class is treated
# e.g., in the XML context
rk.XML.dialog(test.plotOptions)
# use this in the logic section to disable the "type" slot
rk.XML.set(test.plotOptions, set="allow_type", to=FALSE)

# now in JS context
# manually define the variable
cat(rk.paste.JS(test.plotOptions))
# this is usually not necessary, as rk.paste.JS.graph() can
# define variables automatically
cat(
```

```
  rk.paste.JS.graph(
    echo("plot(", test.plotOptions, ")"),
    plotOpts=test.plotOptions
  )
)

# as you can also see in the above example, echo() just
# fills in the JS varaible
echo(test.plotOptions)
```

`rk.plugin.component` *Generate RKWard plugin components*

Description

Generate RKWard plugin components

Usage

```
rk.plugin.component(about, xml = list(), js = list(), rkh = list(),
  provides = c("logic", "dialog"), scan = c("var", "saveobj", "settings"),
  guess.getter = FALSE, hierarchy = "test", include = NULL,
  create = c("xml", "js", "rkh"), hints = TRUE, gen.info = TRUE,
  indent.by = "\t")
```

Arguments

about	Either a character string with the name of this plugin component, or an object of class `XiMpLe.node` with further descriptive information on it, like its authors and dependencies (see `link[XiMpLe:rk.XML.about]{rk.XML.about}` for details). This is only useful for information that differs from the <about> section of the `.pluginmap` file.
xml	A named list of options to be forwarded to `rk.XML.plugin`, to generate the GUI XML file. Not all options are supported because some don't make sense in this context. Valid options are: `"dialog"`, `"wizard"`, `"logic"` and `"snippets"`. If not set, their default values are used. See `rk.XML.plugin` for details.
js	A named list of options to be forwarded to `rk.JS.doc`, to generate the JavaScript file. Not all options are supported because some don't make sense in this context. Valid options are: `"require"`, `"results.header"`, `"variables"`, `"globals"`, `"preprocess"`, `"calculate"`, `"printout"`, `"doPrintout"` and `"load.silencer"`. If not set, their default values are used. See `rk.JS.doc` for details.
rkh	A named list of options to be forwarded to `rk.rkh.doc`, to generate the help file. Not all options are supported because some don't make sense in this context. Valid options are: `"summary"`, `"usage"`, `"sections"`, `"settings"`, `"related"` and `"technical"`. If not set, their default values are used. See `rk.rkh.doc` for details.

`provides` Character vector with possible entries of `"logic"`, `"dialog"` or `"wizard"`, defining what sections the GUI XML file should provide even if `dialog`, `wizard` and `logic` are `NULL`. These sections must be edited manually and some parts are therefore commented out.

`scan` A character vector to trigger various automatic scans of the generated GUI XML file. Valid enties are:

`"var"` Calls `rk.JS.scan` to define all needed variables in the `calculate()` function of the JavaScript file. These variables will be added to variables defined by the `js` option, if any (see below).

`"saveobj"` Calls `rk.JS.saveobj` to generate code to save R results in the `printout()` function of the JavaScript file. This code will be added to the code defined by the `js` option, if any (see below).

`"settings"` Calls `rk.rkh.scan` to generate `<setting>` sections for each relevant GUI element in the `<settings>` section of the help file. This option will be overruled if you provide that section manually by the `rkh` option (see below).

`guess.getter` Logical, if `TRUE` try to get a good default getter function for JavaScript variable values (if `scan` is active). This will use some functions which were added with RKWard 0.6.1, and therefore raise the dependencies for your plugin/component accordingly. Nonetheless, it's recommended.

`hierarchy` A character vector with instructions where to place this component in the menu hierarchy, one list or string. Valid single values are `"file"`, `"edit"`, `"view"`, `"workspace"`, `"run"`, `"data"`, `"analysis"`, `"plots"`, `"distributions"`, `"windows"`, `"settings"` and `"help"`, anything else will place it in a "test" menu. If `hierarchy` is a list, each entry represents the label of a menu level.

`include` Character string or vector, relative path(s) to other file(s), which will then be included in the head of the GUI XML document.

`create` A character vector with one or more of these possible entries:

`"xml"` Create the plugin `.xml` XML file skeleton.

`"js"` Create the plugin `.js` JavaScript file skeleton.

`"rkh"` Create the plugin `.rkh` help file skeleton.

`hints` Logical, if `TRUE` and you leave optional entries empty (like `rkh=list()`), dummy sections will be added.

`gen.info` Logical, if `TRUE` comment notes will be written into the genrated documents, that they were generated by `rkwarddev` and changes should be done to the script.

`indent.by` A character string defining the indentation string to use.

Value

An object of class `rk.plug.comp`.

See Also

Introduction to Writing Plugins for RKWard

Examples

```
## Not run:
test.dropdown <- rk.XML.dropdown("mydrop",
  options=list("First Option"=c(val="val1"),
  "Second Option"=c(val="val2", chk=TRUE)))
test.checkboxes <- rk.XML.row(rk.XML.col(
  list(test.dropdown,
    rk.XML.cbox(label="foo", val="foo1", chk=TRUE),
    rk.XML.cbox(label="bar", val="bar2"))
  ))
test.vars <- rk.XML.vars("select some vars", "vars go here")
test.tabbook <- rk.XML.dialog(rk.XML.tabbook("My Tabbook",
  tabs=c("First Tab"=test.checkboxes, "Second Tab"=test.vars)))

rk.plugin.component("Square the Circle",
  xml=list(dialog=test.tabbook))

## End(Not run)
```

`rk.plugin.skeleton` *Generate skeletons for RKWard plugins*

Description

With this function you can write everything from a basic skeleton structure to a complete functional plugin, including several components/dialogs. You should always define one main component (by `xml`, `js`, `rkh` etc.) before you provide additional features by `components`.

Usage

```
rk.plugin.skeleton(about, path = tempdir(), provides = c("logic", "dialog"),
  scan = c("var", "saveobj", "settings"), guess.getter = FALSE,
  xml = list(), js = list(), pluginmap = list(), rkh = list(),
  overwrite = FALSE, tests = TRUE, lazyLoad = TRUE, create = c("pmap",
  "xml", "js", "rkh", "desc", "clog"), suggest.required = TRUE,
  components = list(), dependencies = NULL, edit = FALSE, load = FALSE,
  show = FALSE, gen.info = TRUE, hints = TRUE, indent.by = "\t",
  lang = rk.get.language())
```

Arguments

`about`	Either an object of class `XiMpLe.node` with descriptive information on the plugin and its authors (see `link[XiMpLe:rk.XML.about]{rk.XML.about}` for details), or a character string with the name of the plugin package. If the latter, no `DESCRIPTION` file will be created.
`path`	Character sting, path to the main directory where the skeleton should be created.

provides — Character vector with possible entries of "logic", "dialog" or "wizard", defining what sections the GUI XML file should provide even if dialog, wizard and logic are NULL. These sections must be edited manually and some parts are therefore commented out.

scan — A character vector to trigger various automatic scans of the generated GUI XML file. Valid enties are:

"var" Calls rk.JS.scan to define all needed variables in the calculate() function of the JavaScript file. These variables will be added to variables defined by the js option, if any (see below).

"saveobj" Calls rk.JS.saveobj to generate code to save R results in the printout() function of the JavaScript file. This code will be added to the code defined by the js option, if any (see below).

"settings" Calls rk.rkh.scan to generate <setting> sections for each relevant GUI element in the <settings> section of the help file. This option will be overruled if you provide that section manually by the rkh option (see below).

guess.getter — Logical, if TRUE try to get a good default getter function for JavaScript variable values (if scan is active). This will use some functions which were added with RKWard 0.6.1, and therefore raise the dependencies for your plugin/component accordingly. Nonetheless, it's recommended.

xml — A named list of options to be forwarded to rk.XML.plugin, to generate the GUI XML file. Not all options are supported because some don't make sense in this context. Valid options are: "dialog", "wizard", "logic" and "snippets". If not set, their default values are used. See rk.XML.plugin for details.

js — A named list of options to be forwarded to rk.JS.doc, to generate the JavaScript file. Not all options are supported because some don't make sense in this context. Valid options are: "require", "results.header", "variables", "globals", "preprocess", "calculate", "printout", "doPrintout" and "load.silencer". If not set, their default values are used. See rk.JS.doc for details.

pluginmap — A named list of options to be forwarded to rk.XML.pluginmap, to generate the pluginmap file. Not all options are supported because some don't make sense in this context. Valid options are: "name", "hierarchy" and "require". If not set, their default values are used. See rk.XML.pluginmap for details.

rkh — A named list of options to be forwarded to rk.rkh.doc, to generate the help file. Not all options are supported because some don't make sense in this context. Valid options are: "summary", "usage", "sections", "settings", "related" and "technical". If not set, their default values are used. See rk.rkh.doc for details.

overwrite — Logical, whether existing files should be replaced. Defaults to FALSE.

tests — Logical, whether directories and files for plugin tests should be created. Defaults to TRUE. A new testsuite file will only be generated if none is present (overwrite is ignored).

lazyLoad — Logical, whether the package should be prepared for lazy loading or not. Should be left TRUE, unless you have very good reasons not to.

create — A character vector with one or more of these possible entries:

"pmap" Create the .pluginmap file.

"xml" Create the plugin .xml XML file skeleton.

"js" Create the plugin .js JavaScript file skeleton.

"rkh" Create the plugin .rkh help file skeleton.

"desc" Create the DESCRIPTION file.

"clog" Create the ChangeLog file (only if none exists).

Default is to create all of these files. Existing files will only be overwritten if overwrite=TRUE.

suggest.required
: Logical, if TRUE R package dependencies in about will be added to the Suggests: field of the DESCRIPTION file, otherwise to the Depends: field.

components
: A list of plugin components. See rk.XML.component for details.

dependencies
: An object of class XiMpLe.node to be pasted as the <dependencies> section, See rk.XML.dependencies for details. Skipped if NULL.

edit
: Logical, if TRUE RKWard will automatically open the created files for editing, by calling rk.edit.files. This applies to all files defined in create.

load
: Logical, if TRUE and "pmap" in create, RKWard will automatically add the created .pluginmap file to its menu structure by calling rk.load.pluginmaps. You can then try the plugin immediately.

show
: Logical, if TRUE and "pmap" in create, RKWard will automatically call the created plugin after it was loaded (i.e., this implies and also sets load=TRUE). This will only work on the main component, though.

gen.info
: Logical, if TRUE comment notes will be written into the genrated documents, that they were generated by rkwarddev and changes should be done to the script.

hints
: Logical, if TRUE and you leave out optional entries (like dependencies=NULL), dummy sections will be added as comments.

indent.by
: A character string defining the indentation string to use.

lang
: Character string, the language of the plugin. See i18n for details.

Value

Character string with the path to the plugin root directory.

See Also

Introduction to Writing Plugins for RKWard

Examples

```
## Not run:
# a simple example with only basic information
about.info <- rk.XML.about(
  name="Square the circle",
  author=c(
    person(given="E.A.", family="Dölle",
      email="doelle@eternalwondermaths.example.org", role="aut"),
```

```
    person(given="A.", family="Assistant",
      email="alterego@eternalwondermaths.example.org", role=c("cre","ctb"))
    ))

rk.plugin.skeleton(about.info)

# a more complex example, already including some dialog elements
about.info <- rk.XML.about(
  name="Square the circle",
  author=c(
    person(given="E.A.", family="Dölle",
      email="doelle@eternalwondermaths.example.org", role="aut"),
    person(given="A.", family="Assistant",
      email="alterego@eternalwondermaths.example.org", role=c("cre","ctb"))
    ),
  about=list(
    desc="Squares the circle using Heisenberg compensation.",
    version="0.1-3",
    date=Sys.Date(),
    url="http://eternalwondermaths.example.org/23/stc.html",
    license="GPL",
    category="Geometry"),
  dependencies=list(
    rkward.min="0.5.3",
    rkward.max="",
    R.min="2.10",
    R.max=""),
  package=list(
    c(name="heisenberg", min="0.11-2", max="",
      repository="http://rforge.r-project.org"),
    c(name="DreamsOfPi", min="0.2", max="", repository="")),
  pluginmap=list(
    c(name="heisenberg.pluginmap", url="http://eternalwondermaths.example.org/hsb"))
)

test.dropdown <- rk.XML.dropdown("mydrop",
  opts=list("First Option"=c(val="val1"),
  "Second Option"=c(val="val2", chk=TRUE)))
test.checkboxes <- rk.XML.row(rk.XML.col(
  list(test.dropdown,
    rk.XML.cbox(label="foo", val="foo1", chk=TRUE),
    rk.XML.cbox(label="bar", val="bar2"))
  ))
test.vars <- rk.XML.vars("select some vars", "vars go here")
test.tabbook <- rk.XML.dialog(rk.XML.tabbook("My Tabbook", tab.labels=c("First Tab",
  "Second Tab"), children=list(test.checkboxes, test.vars)))

rk.plugin.skeleton(about.info, xml=list(dialog=test.tabbook),
  overwrite=TRUE)

## End(Not run)
```

rk.rkh.caption *Create XML "caption" node for RKWard help pages*

Description

This function will create a caption node for settings sections in RKWard help files.

Usage

```
rk.rkh.caption(id, title = NULL)
```

Arguments

id	Either a character string (the id of the XML element to explain), or an object of class XiMpLe.node (whose id will be extracted and used).
title	Character string, title to be displayed. If NULL, the label of the element will be shown.

Value

An object of class XiMpLe.node.

See Also

rk.rkh.doc, rk.rkh.settings and the Introduction to Writing Plugins for RKWard

Examples

```
# define a sample frame
test.dropdown <- rk.XML.dropdown("mydrop",
  options=list("First Option"=c(val="val1"),
  "Second Option"=c(val="val2", chk=TRUE)))
test.frame <- rk.XML.frame(test.dropdown, label="Some options")
# create the caption
test.caption <- rk.rkh.caption(test.frame)
cat(pasteXML(test.caption))
```

rk.rkh.doc *Create RKWard help file skeleton*

Description

Create RKWard help file skeleton

Usage

```
rk.rkh.doc(summary = NULL, usage = NULL, sections = NULL,
  settings = NULL, related = NULL, technical = NULL, title = NULL,
  hints = TRUE, gen.info = TRUE, lang = rk.get.language())
```

Arguments

summary	An object of class XiMpLe.node to be pasted as the <summary> section. See rk.rkh.summary for details.
usage	An object of class XiMpLe.node to be pasted as the <usage> section. See rk.rkh.usage for details.
sections	A (list of) objects of class XiMpLe.node to be pasted as <section> sections. See rk.rkh.section for details.
settings	An object of class XiMpLe.node to be pasted as the <settings> section. See rk.rkh.settings for details. Refer to rk.rkh.scan for a function to create this from an existing plugin XML file.
related	An object of class XiMpLe.node to be pasted as the <related> section. See rk.rkh.related for details.
technical	An object of class XiMpLe.node to be pasted as the <technical> section. See rk.rkh.technical for details.
title	An object of class XiMpLe.node to be pasted as the <title> section. See rk.rkh.title for details.
hints	Logical, if TRUE and you leave out optional entries (like technical=NULL), empty dummy sections will be added.
gen.info	Logical, if TRUE a comment note will be written into the document, that it was generated by rkwarddev and changes should be done to the script.
lang	Character string, the language of the help page. See i18n for details.

Value

An object of class XiMpLe.doc.

See Also

rk.rkh.summary, rk.rkh.usage, rk.rkh.settings, rk.rkh.scan, rk.rkh.related, rk.rkh.technical and the Introduction to Writing Plugins for RKWard

`rk.rkh.link` *Create XML "link" node for RKWard help pages*

Description

Create XML "link" node for RKWard help pages

Usage

```
rk.rkh.link(href, text = NULL, type = "R")
```

Arguments

`href`	Character string, either the URL to link to, name of an R package or ID of another plugin (see `type`).
`text`	Character string, optional link text.
`type`	Character string, one of the following valid entries:

- `"url"href` is assumend to be the actual URL.
- `"R"href` is assumend to be the name of an R package, i.e., the link generated will look like `rkward://rhelp/<href>`.
- `"RK"href` is assumend to be the ID of another RKWard plugin, i.e., the link generated will look like `rkward://component/<href>`.

Value

An object of class `XiMpLe.node`.

See Also

`rk.rkh.doc` and the Introduction to Writing Plugins for RKWard

Examples

```
package.link <- rk.rkh.link("Spice")
cat(pasteXML(package.link))
```

rk.rkh.related *Create XML "related" node for RKWard help pages*

Description

Create XML "related" node for RKWard help pages

Usage

```
rk.rkh.related(..., text = NULL)
```

Arguments

...	Objects of class XiMpLe.node. They must all have the name "link".
text	Character string, the text to be displayed.

Value

An object of class XiMpLe.node.

See Also

rk.rkh.doc and the Introduction to Writing Plugins for RKWard

Examples

```
package.link <- rk.rkh.link("Spice")
plugin.related <- rk.rkh.related(package.link)
cat(pasteXML(plugin.related))
```

rk.rkh.scan *Create RKWard help nodes from plugin XML*

Description

Create RKWard help nodes from plugin XML

Usage

```
rk.rkh.scan(pXML, help = TRUE, captions = TRUE, component = NULL)
```

Arguments

pXML	Either an object of class XiMpLe.doc or XiMpLe.node, or path to a plugin XML file.
help	Logical, if TRUE a list of XiMpLe.node objects will be returned, otherwise a character vector with only the relevant ID names.
captions	Logical, if TRUE captions will be generated for all "page", "tab" and "frame" nodes.
component	Character string, name of the scanned component. Only needed if you want to search for help text provided by rk.set.rkh.prompter.

Value

A character vector or a list of XiMpLe.node objects. Returns NULL if no documentable nodes are found.

See Also

Introduction to Writing Plugins for RKWard

---

rk.rkh.section *Create XML "section" node for RKWard help pages*

Description

This function will create a section node for settings sections in RKWard help files.

Usage

```
rk.rkh.section(title, text = NULL, short = NULL, id.name = "auto")
```

Arguments

title	Character string, title to be displayed.
text	Character string, the text to be displayed.
short	Character string, short title for the menu for links to this section.
id.name	Character string, a unique ID for this element. If "auto", an ID will be generated automatically from the title value.

Value

An object of class XiMpLe.node.

See Also

rk.rkh.doc and the Introduction to Writing Plugins for RKWard

Examples

```
test.section <- rk.rkh.section("General background", text="Some important notes...",
short="Background")
cat(pasteXML(test.section))
```

---

rk.rkh.setting *Create XML "setting" node for RKWard help pages*

Description

This function will create a setting node for settings sections in RKWard help files.

Usage

```
rk.rkh.setting(id, text = NULL, title = NULL)
```

Arguments

id	Either a character string (the id of the XML element to explain), or an object of class XiMpLe.node (whose id will be extracted and used).
text	Character string, the text to be displayed.
title	Character string, title to be displayed. If NULL, the label of the element will be shown.

Value

An object of class XiMpLe.node.

See Also

rk.rkh.doc, rk.rkh.settings and the Introduction to Writing Plugins for RKWard

Examples

```
test.checkbox <- rk.XML.cbox(label="foo", value="foo1", chk=TRUE)
# explain the option
test.setting <- rk.rkh.setting(test.checkbox, text="Check this to do Foo.")
cat(pasteXML(test.setting))
```

rk.rkh.settings *Create XML "settings" node for RKWard help pages*

Description

This function will create a settings node for the document section, with optional child nodes "setting" and "caption".

Usage

```
rk.rkh.settings(...)
```

Arguments

...	Objects of class XiMpLe.node. They must all have the name "setting" or "caption".

Value

An object of class XiMpLe.node.

See Also

rk.rkh.doc, rk.rkh.setting, rk.rkh.caption, and the Introduction to Writing Plugins for RKWard

Examples

```
# define a sample frame
test.dropdown <- rk.XML.dropdown("mydrop",
  options=list("First Option"=c(val="val1"),
  "Second Option"=c(val="val2", chk=TRUE)))
test.frame <- rk.XML.frame(test.dropdown, label="Some options")
# create the caption
test.caption <- rk.rkh.caption(test.frame)
test.setting <- rk.rkh.setting(test.dropdown, text="Chose one of the options.")
test.settings <- rk.rkh.settings(list(test.caption, test.setting))
```

rk.rkh.summary *Create XML "summary" node for RKWard help pages*

Description

Create XML "summary" node for RKWard help pages

Usage

```
rk.rkh.summary(text = NULL)
```

Arguments

text	Character string, the text to be displayed.

Value

An object of class XiMpLe.node.

See Also

rk.rkh.doc and the Introduction to Writing Plugins for RKWard

Examples

```
plugin.summary <- rk.rkh.summary("This plugin folds space, using the spice package.")
cat(pasteXML(plugin.summary))
```

---

rk.rkh.technical *Create XML "technical" node for RKWard help pages*

Description

Create XML "technical" node for RKWard help pages

Usage

```
rk.rkh.technical(text = NULL)
```

Arguments

text	Character string, the text to be displayed.

Value

An object of class XiMpLe.node.

See Also

rk.rkh.doc and the Introduction to Writing Plugins for RKWard

Examples

```
plugin.technical <- rk.rkh.technical("<b>TODO</b>: Implement sandworm detector.")
cat(pasteXML(plugin.technical))
```

rk.rkh.title *Create XML "title" node for RKWard help pages*

Description

Create XML "title" node for RKWard help pages

Usage

```
rk.rkh.title(text = NULL)
```

Arguments

text	Character string, the text to be displayed.

Value

An object of class XiMpLe.node.

See Also

rk.rkh.doc and the Introduction to Writing Plugins for RKWard

Examples

```
plugin.title <- rk.rkh.title("Spice")
cat(pasteXML(plugin.title))
```

rk.rkh.usage *Create XML "usage" node for RKWard help pages*

Description

Create XML "usage" node for RKWard help pages

Usage

```
rk.rkh.usage(text = NULL)
```

Arguments

text	Character string, the text to be displayed.

Value

An object of class XiMpLe.node.

See Also

rk.rkh.doc and the Introduction to Writing Plugins for RKWard

Examples

```
plugin.usage <- rk.rkh.usage("First do this, then do that ...")
```

---

rk.set.comp *Define the component you're currently working on*

Description

This small tool let's you set a component name as kind of "active", which simply means it will be returned by rk.get.comp. This can be used by functions like, e.g., rk.XML.cbox, to add text for .rkh pages automatically to the current plugin component.

Usage

```
rk.set.comp(component = NULL)
```

Arguments

component	Character string, name of the component to set. If NULL, no component will be set as default, and rk.get.comp will return NULL subsequently.

---

rk.set.language *Set plugin language for internationalisation*

Description

Stores the given language in an internal environment, so functions like i18n can use it.

Usage

```
rk.set.language(lang = NULL, locales = NULL)
```

Arguments

lang	Character string, abbreviated language to use with i18n, e.g. "en".
locales	Character vector, all the locales this translation covers, e.g. c("en_EN", "en_US").

Examples

```
rk.set.language("en", c("en_EN", "en_US"))
```

rk.set.rkh.prompter *Set up an environment to store .rkh related information*

Description

By using an environment like this, you are able to write information for RKWard help files directly into your script code of certrain functions, like for radio buttons or checkboxes.

Usage

```
rk.set.rkh.prompter(component = NULL, id = NULL, help = NULL,
  rm = FALSE)
```

Arguments

component	Character string, should be a unique name to identify the current plugin/component. If NULL, this function quits silently without any action.
id	Either a character string (the id of the node to store the help information for), or an object of class XiMpLe.node (whose id will be extracted and used).
help	Character string, will be used as the text value for a setting node in the .rkh file.
rm	Logical, If TRUE will remove all information stored by the name of component (if id=NULL) or of the given id=NULL, respectively.

Details

The information is temporarily stored in an internal environment, using the plugin/component name you specify. Each entry is named after the ID of its respective node. If you later call rk.plugin.component (or it is called by other functions) and you activate the scan option for rkh files, the scanning process will try to find a match for each relevant XML node. That is, the info which is stored in the environment will magically be written into the help file.

Examples

```
rk.set.rkh.prompter("rk.myPlugin", "someID", "CLick this to feel funny.")
```

rk.testsuite.doc *Create testsuite outline to test an RKWard plugin*

Description

Create testsuite outline to test an RKWard plugin

Usage

```
rk.testsuite.doc(name = NULL)
```

Arguments

name	A character string, name of the plugin/dialog.

Value

A character string.

See Also

Introduction to Writing Plugins for RKWard

---

rk.uniqueIDs *Check plugin dialogs for duplicate IDs*

Description

A plugin must not have duplicated IDs to work properly. This function cannot automatically correct duplicates, but it will warn you about it, so you can correct your plugin script manually

Usage

```
rk.uniqueIDs(obj, bool = FALSE, warning = TRUE)
```

Arguments

obj	An XML object of class XiMpLe.node or XiMpLe.doc.
bool	Logical, if TRUE this function will return a logical value.
warning	Logical, if TRUE will throw a warning if duplicates were found, listing the findings.

Value

A vector with duplicate IDs, if any were found. If bool=TRUE returns a logical value.

rk.XML.about *Create XML node "about" for RKWard pluginmaps*

Description

Create XML node "about" for RKWard pluginmaps

Usage

```
rk.XML.about(name, author, about = list(desc = "SHORT_DESCRIPTION", version =
  "0.01-0", date = Sys.Date(), url = "http://EXAMPLE.com", license =
  "GPL (>= 3)", long.desc = NULL), dependencies = NULL, package = NULL,
  pluginmap = NULL)
```

Arguments

name	A character string with the plugin name.
author	A vector of objects of class person with these elements (mandatory): **given** Author given name **family** Author family name **email** Author mail address (can be omitted if role does not include "cre") **role** This person's specific role, e.g. "aut" for actual author, "cre" for maintainer or "ctb" for contributor. See person for more details on this, especially for valid roles.
about	A named list with these elements: **desc** A short description (mandatory) **version** Plugin version (mandatory) **date** Release date (mandatory) **url** URL for the plugin (optional) **license** License the plugin is distributed under (mandatory) **category** A category for this plugin (optional) **long.desc** A long description (optional, defaults to desc)
dependencies	A named list with these elements: **rkward.min** Minimum RKWard version needed for this plugin (optional) **rkward.max** Maximum RKWard version needed for this plugin (optional) **R.min** Minimum R version needed for this plugin (optional) **R.max** Maximum R version needed for this plugin (optional)
package	A list of named character vectors, each with these elements: **name** Name of a package this plugin depends on (optional) **min** Minimum version of the package (optional) **max** Maximum version of the package (optional) **repository** Repository to download the package (optional)

`pluginmap` A named list with these elements:

name Identifier of a pluginmap this plugin depends on (optional)

url URL to get the pluginmap (optional)

See Also

Introduction to Writing Plugins for RKWard

Examples

```
about.node <- rk.XML.about(
  name="Square the circle",
  author=c(
    person(given="E.A.", family="Dölle",
      email="doelle@eternalwondermaths.example.org", role="aut"),
    person(given="A.", family="Assistant",
      email="alterego@eternalwondermaths.example.org", role=c("cre","ctb"))
    ),
  about=list(
    desc="Squares the circle using Heisenberg compensation.",
    version="0.1-3",
    date=Sys.Date(),
    url="http://eternalwondermaths.example.org/23/stc.html",
    license="GPL",
    category="Geometry"),
  dependencies=list(
    rkward.min="0.5.3",
    rkward.max="",
    R.min="2.10",
    R.max=""),
  package=list(
    c(name="heisenberg", min="0.11-2", max="",
      repository="http://rforge.r-project.org"),
    c(name="DreamsOfPi", min="0.2", max="", repository="")),
  pluginmap=list(
    c(name="heisenberg.pluginmap", url="http://eternalwondermaths.example.org/hsb"))
)

cat(pasteXML(about.node, shine=2))
```

`rk.XML.attribute` *Create XML "attribute" node for RKWard plugins*

Description

This function will create a attribute node for component sections in .pluginmap files. Only meaningful for import plugins.

Usage

```
rk.XML.attribute(id, value = NULL, label = NULL)
```

Arguments

id	Either a character string (the id of the property whose attribute should be set), or an object of class XiMpLe.node (whose id will be extracted and used).
value	Character string, new value for the attribute.
label	Character string, label associated with the attribute.

Value

An object of class XiMpLe.node.

See Also

rk.XML.components, and the Introduction to Writing Plugins for RKWard

Examples

```
test.checkbox <- rk.XML.cbox(label="foo", value="foo1", chk=TRUE)
# re-set the attribute
test.attribute <- rk.XML.attribute(test.checkbox, value="bar2", label="bar")
cat(pasteXML(test.attribute))
```

---

rk.XML.browser *Create XML node "browser" for RKWard plugins*

Description

Create XML node "browser" for RKWard plugins

Usage

```
rk.XML.browser(label, type = "file", initial = NULL, urls = FALSE,
  filter = NULL, required = TRUE, id.name = "auto", help = NULL,
  component = rk.get.comp())
```

Arguments

label	Character string, a text label for this plugin element.
type	Character string, valid values are "dir", "file" and "savefile" (i.e., an non-existing file).
initial	Character string, if not NULL will be used as the initial value of the browser.
urls	Logical, whether non-local URLs are permitted or not.

filter	Character vector, file type filter, e.g. filter=c("*.txt", "*.csv") for .txt and .csv files. Try not to induce limits unless absolutely needed, though.
required	Logical, whether an entry is mandatory or not.
id.name	Character string, a unique ID for this plugin element. If "auto" and a label was provided, an ID will be generated automatically from the label.
help	Character string, will be used as the text value for a setting node in the .rkh file. If set to FALSE, rk.rkh.scan will ignore this node. Also needs component to be set accordingly!
component	Character string, name of the component this node belongs to. Only needed if you want to use the scan features for automatic help file generation; needs help to be set accordingly, too!

Value

An object of class XiMpLe.node.

See Also

Introduction to Writing Plugins for RKWard

Examples

```
test.browser <- rk.XML.browser("Browse here:")
cat(pasteXML(test.browser))
```

---

rk.XML.cbox *Create XML node "checkbox" for RKWard plugins*

Description

Create XML node "checkbox" for RKWard plugins

Usage

```
rk.XML.cbox(label, value = "true", un.value = NULL, chk = FALSE,
  id.name = "auto", help = NULL, component = rk.get.comp())
```

Arguments

label	Character string, a text label for this plugin element.
value	Character string, the value to submit if the element is checked.
un.value	Character string, an optional value for the unchecked option.
chk	Logical, whether this element should be checked by default.
id.name	Character string, a unique ID for this plugin element. If "auto", an ID will be generated automatically from the label.

help	Character string, will be used as the text value for a setting node in the .rkh file. If set to FALSE, rk.rkh.scan will ignore this node. Also needs component to be set accordingly!
component	Character string, name of the component this node belongs to. Only needed if you want to use the scan features for automatic help file generation; needs help to be set accordingly, too!

Value

An object of class XiMpLe.node.

Note

There's also a simple wrapper function rk.XML.checkbox.

See Also

Introduction to Writing Plugins for RKWard

Examples

```
test.checkboxes <- rk.XML.row(rk.XML.col(
  list(
    rk.XML.cbox(label="foo", value="foo1", chk=TRUE),
    rk.XML.cbox(label="bar", value="bar2"))))
cat(pasteXML(test.checkboxes))
```

---

rk.XML.code *Create XML node "code" for RKWard plugins*

Description

Create XML node "code" for RKWard plugins

Usage

```
rk.XML.code(file)
```

Arguments

file	A character string, the JavaScript file name to be included.

Value

An object of class XiMpLe.node.

See Also

Introduction to Writing Plugins for RKWard

Examples

```
test.code <- rk.XML.code("some_file.js")
cat(pasteXML(test.code))
```

`rk.XML.col` *Create XML node "column" for RKWard plugins*

Description

Create XML node "column" for RKWard plugins

Usage

```
rk.XML.col(..., id.name = "auto")
```

Arguments

`...`	Objects of class `XiMpLe.node`.
`id.name`	Character string, a unique ID for this plugin element. If `"auto"`, an ID will be generated automatically from the objects in `...`. If `NULL`, no ID will be given.

Value

An object of class `XiMpLe.node`.

See Also

Introduction to Writing Plugins for RKWard

Examples

```
test.checkboxes <- rk.XML.row(rk.XML.col(
    rk.XML.cbox(label="foo", val="foo1", chk=TRUE),
    rk.XML.cbox(label="bar", val="bar2")))
cat(pasteXML(test.checkboxes))
```

rk.XML.component *Create XML "component" node for RKWard plugins*

Description

This function will create a component node for components sections of .pluginmap files.

Usage

```
rk.XML.component(label, file, id.name = "auto", type = "standard",
  dependencies = NULL)
```

Arguments

label	Character string, a label for the component.
file	Character string, file name of a plugin XML file defining the GUI.
id.name	Character string, a unique ID for this plugin element. If "auto", an ID will be generated automatically from the label.
type	Character string, type of component. As of now, only "standard" is supported. The option is just implemented for completeness.
dependencies	An object of class XiMpLe.node to define <dependencies> for this component. See rk.XML.dependencies for details. Skipped if NULL.

Value

An object of class XiMpLe.node.

See Also

rk.XML.components, rk.XML.dependencies, and the Introduction to Writing Plugins for RKWard

Examples

```
test.component <- rk.XML.component("My GUI dialog", "plugins/MyGUIdialog.xml")
```

rk.XML.components *Create XML "components" node for RKWard plugins*

Description

This function will create a components node for a .pluginmap file, with mandatory child nodes "component".

Usage

```
rk.XML.components(...)
```

Arguments

...	Objects of class XiMpLe.node. They must all have the name "component".

Value

A list of objects of class XiMpLe.node.

See Also

rk.XML.pluginmap, rk.XML.component, and the Introduction to Writing Plugins for RKWard

Examples

```
test.component <- rk.XML.component("My GUI dialog", "plugins/MyGUIdialog.xml")
test.components <- rk.XML.components(test.component)
cat(pasteXML(test.components))
```

rk.XML.connect *Create XML node "connect" for RKWard plugins*

Description

If you define a XiMpLe.node object as governor which is not a <convert> node and not=FALSE, the function will automatically append to its id.

Usage

```
rk.XML.connect(governor, client, get = "state", set = "enabled",
  not = FALSE, reconcile = FALSE)
```

Arguments

governor	Either a character string (the id of the property whose state should control the client), or an object of class XiMpLe.node (whose id will be extracted and used). Usually a <convert> node defined earlier (see rk.XML.convert).
client	Character string, the id if the element to be controlled by governor.
get	Character string, a valid modifier for the node property of governor, often the ".state" value of some apropriate node.
set	Character string, a valid modifier for the node property of client, usually one of "enabled", "visible" or "required".
not	Logical, if TRUE, the state of governor (TRUE/FALSE) will be inversed.
reconcile	Logical, forces the governor to only accept values which are valid for the client as well.

Value

An object of class XiMpLe.node.

Note

To get a list of the implemented modifiers in this package, call rkwarddev:::all.valid.modifiers.

See Also

rk.XML.convert, rk.XML.external, rk.XML.logic, rk.XML.set, rk.XML.switch, and the Introduction to Writing Plugins for RKWard

Examples

```
test.connect <- rk.XML.connect(governor="lgc_foobar", client="frame_bar")
cat(pasteXML(test.connect))
```

---

rk.XML.context *Create XML "context" node for RKWard plugins*

Description

This function will create a context node for .pluginmap files, with mandatory child nodes "menu".

Usage

```
rk.XML.context(..., id = "x11")
```

Arguments

...	Objects of class XiMpLe.node, must all be "menu".
id	Character string, either "x11" or "import".

Value

An object of class XiMpLe.node.

See Also

rk.XML.menu, rk.XML.entry, rk.XML.component, and the Introduction to Writing Plugins for RKWard

Examples

```
test.component <- rk.XML.component("My GUI dialog", "plugins/MyGUIdialog.xml")
test.entry <- rk.XML.entry(test.component)
test.menu <- rk.XML.menu("Analysis", nodes=test.entry, id.name="analysis")
test.context <- rk.XML.context(test.menu)
cat(pasteXML(test.context))
```

---

rk.XML.convert *Create XML node convert for RKWard plugins*

Description

If sources holds XiMpLe.node objects, the validity of modifiers is automatically checked for that tag.

Usage

```
rk.XML.convert(sources, mode = c(), required = FALSE, id.name = "auto")
```

Arguments

sources	A list with at least one value, either resembling the id of an existing element to be queried as a character string, or a previously defined object of class XiMpLe.node (whose id will be extracted and used). If you want to examine e.g. the state or string value specificly, just name the value accoringly, e.g., sources=list("vars0", string="input1"
mode	A named vector with either exactly one of the following elements: • equalsTrue if sources equals this value. • notequalsTrue if sources differs from this value. • andTrue if all sources are true. The sources must be boolean, and the actual value here is irrelevant, so mode=c(and="") is valid. • orTrue if any of the sources is true. The sources must be boolean, and the actual value here is irrelevant, so mode=c(or="") is valid. or at least one of these elemets: • minTrue if sources is at least this value. They must be numeric. • maxTrue if sources is below this value. They must be numeric.

required	Logical, sets the state of the required_true attribute. If TRUE, the plugin submit button is only enabled if this property is true.
id.name	Character string, a unique ID for this plugin element. If "auto", an ID will be generated automatically from the sources and mode value.

Value

An object of class XiMpLe.node.

Note

To get a list of the implemented modifiers for sources in this package, call rkwarddev:::all.valid.modifiers.

See Also

rk.XML.connect, rk.XML.external, rk.XML.logic, rk.XML.set, rk.XML.switch, and the Introduction to Writing Plugins for RKWard

Examples

```
test.convert <- rk.XML.convert(list(string="foo"), mode=c(notequals="bar"))
cat(pasteXML(test.convert))
```

---

rk.XML.copy *Create XML copy node for RKWard plugins*

Description

Create XML copy node for RKWard plugins

Usage

```
rk.XML.copy(id, as = NULL)
```

Arguments

id	Either a character string (the id of the property to be copied), or an object of class XiMpLe.node (whose id will be extracted and used).
as	A character string resembling the copy_element_tag_name value. I.e., must be a valid tag name. Will cause a change of tag name of the id (e.g. "tab") to as (e.g. "page").

Value

An object of class XiMpLe.node.

See Also

rk.XML.plugin, rk.plugin.skeleton, and the Introduction to Writing Plugins for RKWard

Examples

```
# define a checkbox for the actual dialog
test.cbox1 <- rk.XML.cbox(label="More than 30 subjects", val="true")
# define the wizard
test.text <- rk.XML.text("Did you test more than 30 subjects?")
test.copy <- rk.XML.copy(id=test.cbox1)
test.wizard <- rk.XML.wizard(rk.XML.page(list(test.text, test.copy)))
cat(pasteXML(test.wizard))
```

---

rk.XML.dependencies *Create XML node "dependencies" for RKWard pluginmaps*

Description

Create XML node "dependencies" for RKWard pluginmaps

Usage

```
rk.XML.dependencies(dependencies = NULL, package = NULL, pluginmap = NULL,
  hints = FALSE)
```

Arguments

dependencies	A named list with these elements: **rkward.min** Minimum RKWard version needed for this plugin (optional) **rkward.max** Maximum RKWard version needed for this plugin (optional) **R.min** Minimum R version needed for this plugin (optional) **R.max** Maximum R version needed for this plugin (optional)
package	A list of named character vectors, each with these elements: **name** Name of a package this plugin depends on (optional) **min** Minimum version of the package (optional) **max** Maximum version of the package (optional) **repository** Repository to download the package (optional)
pluginmap	A named list with these elements: **name** Identifier of a pluginmap this plugin depends on (optional) **url** URL to get the pluginmap (optional)
hints	Logical, if TRUE, NULL values will be replaced with example text.

Note

The <dependencies> node was introduced with RKWard 0.6.1, please set the dependencies of your component/plugin accordingly.

See Also

rk.XML.dependency_check, and the Introduction to Writing Plugins for RKWard

Examples

```
dependencies.node <- rk.XML.dependencies(
  dependencies=list(
    rkward.min="0.5.3",
    rkward.max="",
    R.min="2.10",
    R.max=""),
  package=list(
    c(name="heisenberg", min="0.11-2", max="",
      repository="http://rforge.r-project.org"),
    c(name="DreamsOfPi", min="0.2", max="", repository="")),
  pluginmap=list(
    c(name="heisenberg.pluginmap", url="http://eternalwondermaths.example.org/hsb"))
)
```

---

rk.XML.dependency_check

Create XML node "dependency_check" for RKWard pluginmaps

Description

Create XML node "dependency_check" for RKWard pluginmaps

Usage

```
rk.XML.dependency_check(id.name, dependencies = NULL, package = NULL,
  pluginmap = NULL, hints = FALSE)
```

Arguments

id.name	Character string, a unique ID for this plugin element.
dependencies	A named list with these elements: **rkward.min** Minimum RKWard version needed for this plugin (optional) **rkward.max** Maximum RKWard version needed for this plugin (optional) **R.min** Minimum R version needed for this plugin (optional) **R.max** Maximum R version needed for this plugin (optional)
package	A list of named character vectors, each with these elements: **name** Name of a package this plugin depends on (optional) **min** Minimum version of the package (optional) **max** Maximum version of the package (optional) **repository** Repository to download the package (optional)

pluginmap	A named list with these elements: **name** Identifier of a pluginmap this plugin depends on (optional) **url** URL to get the pluginmap (optional)
hints	Logical, if TRUE, NULL values will be replaced with example text.

Note

The <dependency_check> node was introduced with RKWard 0.6.1, please set the dependencies of your component/plugin accordingly.

See Also

rk.XML.dependencies, and the Introduction to Writing Plugins for RKWard

Examples

```
dependency_check.node <- rk.XML.dependency_check(
  id.name="dep_check",
  dependencies=list(
    rkward.min="0.5.3",
    rkward.max="",
    R.min="2.10",
    R.max=""),
  package=list(
    c(name="heisenberg", min="0.11-2", max="",
      repository="http://rforge.r-project.org"),
    c(name="DreamsOfPi", min="0.2", max="", repository="")),
  pluginmap=list(
    c(name="heisenberg.pluginmap", url="http://eternalwondermaths.example.org/hsb"))
)
```

---

rk.XML.dialog *Create XML dialog section for RKWard plugins*

Description

This function will create a dialog section with optional child nodes "browser", "checkbox", "column", "copy", "dropdown", "embed", "formula", "frame", "include", "input", "insert", "preview", "radio", "row", "saveobject", "select", "spinbox", "stretch", "tabbook", "text", "valueselector", "valueslot", "varselector" and "varslot".

Usage

```
rk.XML.dialog(..., label = NULL, recommended = FALSE)
```

Arguments

...	Objects of class XiMpLe.node.
label	Character string, a text label for this plugin element.
recommended	Logical, whether the dialog should be the recommended interface (unless the user has configured RKWard to default to a specific interface). This attribute currently has no effect, as it is implicitly "true", unless the wizard is recommended.

Value

An object of class XiMpLe.node.

See Also

rk.XML.plugin, rk.plugin.skeleton, and the Introduction to Writing Plugins for RKWard

Examples

```
# define an input field and two checkboxes
test.input <- rk.XML.input("Type some text")
test.cbox1 <- rk.XML.cbox(label="Want to type?", val="true")
test.cbox2 <- rk.XML.cbox(label="Are you shure?", val="true")
test.dialog <- rk.XML.dialog(rk.XML.col(test.input, test.cbox1, test.cbox2))
cat(pasteXML(test.dialog))
```

---

rk.XML.dropdown *Create XML node "dropdown" for RKWard plugins*

Description

Create XML node "dropdown" for RKWard plugins

Usage

```
rk.XML.dropdown(label, options = list(label = c(val = "", chk = FALSE)),
  id.name = "auto", help = NULL, component = rk.get.comp())
```

Arguments

label	Character string, a text label for this plugin element.
options	A named list with options to choose from. The names of the list elements will become labels of the options, val defines the value to submit if the option is checked, and chk=TRUE should be set in the one option which is checked by default. Objects generated with rk.XML.option are accepted as well.
id.name	Character string, a unique ID for this plugin element. If "auto" and a label was provided, an ID will be generated automatically from the label.

help	Character string, will be used as the text value for a setting node in the .rkh file. If set to FALSE, rk.rkh.scan will ignore this node. Also needs component to be set accordingly!
component	Character string, name of the component this node belongs to. Only needed if you want to use the scan features for automatic help file generation; needs help to be set accordingly, too!

Value

An object of class XiMpLe.node.

See Also

Introduction to Writing Plugins for RKWard

Examples

```
test.dropdown <- rk.XML.dropdown("mydrop",
  options=list("First Option"=c(val="val1"),
  "Second Option"=c(val="val2", chk=TRUE)))
cat(pasteXML(test.dropdown))
```

---

rk.XML.embed *Create XML node "embed" for RKWard plugins*

Description

Create XML node "embed" for RKWard plugins

Usage

```
rk.XML.embed(component, button = FALSE, label = "Options",
  id.name = "auto")
```

Arguments

component	A character string, registered name (id in pluginmap file) of the component to be embedded.
button	Logical, whether the plugin should be embedded as a button and appear if it's pressed.
label	A character string, text label for the button (only used if button=TRUE).
id.name	Character string, a unique ID for this plugin element. If "auto", an ID will be generated automatically from the label and component strings.

Value

An object of class XiMpLe.node.

See Also

Introduction to Writing Plugins for RKWard

Examples

```
test.embed <- rk.XML.embed("someComponent")
cat(pasteXML(test.embed))
```

`rk.XML.entry` *Create XML "entry" node for RKWard plugins*

Description

This function will create a entry node for menu sections in .pluginmap files.

Usage

```
rk.XML.entry(component, index = -1)
```

Arguments

`component`	A "component" object of class `XiMpLe.node`, or an ID.
`index`	Integer number to influence the level of menu placement.

Value

An object of class `XiMpLe.node`.

See Also

`rk.XML.menu`, `rk.XML.hierarchy`, `rk.XML.component`, `rk.XML.components`, and the Introduction to Writing Plugins for RKWard

Examples

```
test.component <- rk.XML.component("My GUI dialog", "plugins/MyGUIdialog.xml")
test.entry <- rk.XML.entry(test.component)
cat(pasteXML(test.entry))
```

rk.XML.external *Create XML node "external" for RKWard plugins*

Description

Create XML node "external" for RKWard plugins

Usage

```
rk.XML.external(id, default = NULL)
```

Arguments

id	Character string, the ID of the new property.
default	Character string, initial value of the property if not connected.

Value

An object of class XiMpLe.node.

See Also

rk.XML.connect, rk.XML.convert, rk.XML.logic, rk.XML.set, rk.XML.switch, and the Introduction to Writing Plugins for RKWard

Examples

```
test.external <- rk.XML.external(id="ext_property", default="none")
cat(pasteXML(test.external))
```

rk.XML.formula *Create XML node "formula" for RKWard plugins*

Description

If fixed or dependent are objects of class XiMpLe.node, their id will be extracted and used.

Usage

```
rk.XML.formula(fixed, dependent, id.name = "auto")
```

Arguments

fixed	The id of the varslot holding the selected fixed factors.
dependent	The id of the varslot holding the selected dependent variable.
id.name	Character string, a unique ID for this plugin element. If "auto", an ID will be generated automatically from the fixed and dependent value.

Value

An object of class XiMpLe.node.

See Also

rk.XML.varselector, rk.XML.varslot, rk.XML.vars (a wrapper, including formula), and the Introduction to Writing Plugins for RKWard

Examples

```
test.varselector <- rk.XML.varselector("Select some vars")
test.varslot1 <- rk.XML.varslot("Fixed factors", source=test.varselector)
test.varslot2 <- rk.XML.varslot("Dependent variables", source=test.varselector)
test.formula <- rk.XML.formula(fixed=test.varslot1, dependent=test.varslot2)
cat(pasteXML(test.formula))
```

---

rk.XML.frame *Create XML node "frame" for RKWard plugins*

Description

Create XML node "frame" for RKWard plugins

Usage

```
rk.XML.frame(..., label = NULL, checkable = FALSE, chk = TRUE,
  id.name = "auto")
```

Arguments

...	Objects of class XiMpLe.node.
label	Character string, a text label for this plugin element.
checkable	Logical, if TRUE the frame can be switched on and off.
chk	Logical, if TRUE and checkable=TRUE the frame is checkable and active by default.
id.name	Character string, a unique ID for this plugin element. If "auto" and a label was provided, an ID will be generated automatically from the label if presen, otherwise from the objects in the frame. If NULL, no ID will be given.

Value

An object of class XiMpLe.node.

See Also

Introduction to Writing Plugins for RKWard

Examples

```
test.dropdown <- rk.XML.dropdown("mydrop",
  options=list("First Option"=c(val="val1"),
  "Second Option"=c(val="val2", chk=TRUE)))
cat(pasteXML(rk.XML.frame(test.dropdown, label="Some options")))
```

---

rk.XML.help *Create XML node "help" for RKWard plugins*

Description

Create XML node "help" for RKWard plugins

Usage

```
rk.XML.help(file)
```

Arguments

file	A character string, the file name to be included as reference.

Value

An object of class XiMpLe.node.

See Also

Introduction to Writing Plugins for RKWard

Examples

```
test.help <- rk.XML.help("some_file.rkh")
cat(pasteXML(test.help))
```

---

rk.XML.hierarchy *Create XML hierarchy section for RKWard plugins*

Description

This function will create a hierarchy section for .pluginmap files, with mandatory child nodes "menu".

Usage

```
rk.XML.hierarchy(...)
```

Arguments

...	Objects of class XiMpLe.node, must all be "menu".

Value

An object of class XiMpLe.node.

See Also

rk.XML.menu, rk.XML.entry, rk.XML.component, rk.XML.components, and the Introduction to Writing Plugins for RKWard

Examples

```
test.component <- rk.XML.component("My GUI dialog", "plugins/MyGUIdialog.xml")
test.entry <- rk.XML.entry(test.component)
test.menu <- rk.XML.menu("Analysis", nodes=test.entry, id.name="analysis")
test.hierarchy <- rk.XML.hierarchy(test.menu)
cat(pasteXML(test.hierarchy))
```

---

rk.XML.include *Create XML node "include" for RKWard plugins*

Description

Create XML node "include" for RKWard plugins

Usage

```
rk.XML.include(file)
```

Arguments

file	A character string, the file name to be included.

Value

An object of class XiMpLe.node.

See Also

Introduction to Writing Plugins for RKWard

Examples

```
test.include <- rk.XML.include("../some_file.xml")
cat(pasteXML(test.include))
```

`rk.XML.input` *Create XML node "input" for RKWard plugins*

Description

Create XML node "input" for RKWard plugins

Usage

```
rk.XML.input(label, initial = NULL, size = "medium", required = FALSE,
  id.name = "auto", help = NULL, component = rk.get.comp())
```

Arguments

`label`	Character string, a text label for this plugin element.
`initial`	Character string, if not `NULL` will be used as the initial value of the input field.
`size`	One value of either "small", "medium" or "large".
`required`	Logical, whether an entry is mandatory or not.
`id.name`	Character string, a unique ID for this plugin element. If `"auto"`, an ID will be generated automatically from the label.
`help`	Character string, will be used as the `text` value for a setting node in the .rkh file. If set to `FALSE`, `rk.rkh.scan` will ignore this node. Also needs `component` to be set accordingly!
`component`	Character string, name of the component this node belongs to. Only needed if you want to use the scan features for automatic help file generation; needs `help` to be set accordingly, too!

Value

An object of class `XiMpLe.node`.

See Also

Introduction to Writing Plugins for RKWard

Examples

```
test.input <- rk.XML.input("Type some text")
cat(pasteXML(test.input))
```

rk.XML.insert *Create XML node "insert" for RKWard plugins*

Description

This function creates an insert node to use snippets.

Usage

```
rk.XML.insert(snippet)
```

Arguments

snippet	Either a character string (the id of the snippet to be inserted), or an object of class XiMpLe.node (whose id will be extracted and used; must be a snippet!).

Value

An object of class XiMpLe.node.

See Also

rk.XML.snippets, rk.XML.snippet, and the Introduction to Writing Plugins for RKWard

Examples

```
# define a formula section with varselector and varslots
test.formula <- rk.XML.vars("Variables", "Fixed", formula.dependent="Dependent")
# define the snippet
test.snippet <- rk.XML.snippet(test.formula)
# now to insert the snippet
test.insert <- rk.XML.insert(test.snippet)
cat(pasteXML(test.insert))
```

rk.XML.logic *Create XML logic section for RKWard plugins*

Description

This function will create a logic section with "convert", "connect", "include", "insert", "external" and "set" nodes. You can also include JavaScript code to use the locig scripting features of RKWard, if you place it in a comment with rk.comment: Its contents will automatically be placed inside a <script><![CDATA[ ]]></script> node.

Usage

```
rk.XML.logic(...)
```

Arguments

...	Objects of class XiMpLe.node.

Value

An object of class XiMpLe.node.

See Also

rk.XML.convert, rk.XML.connect, rk.XML.external, rk.XML.set, rk.XML.switch, and the Introduction to Writing Plugins for RKWard

Examples

```
# define an input field and two checkboxes
test.input <- rk.XML.input("Type some text")
test.cbox1 <- rk.XML.cbox(label="Want to type?", value="true")
test.cbox2 <- rk.XML.cbox(label="Are you shure?", value="true")
# now create some logic so that the input field is only enabled when both boxes are checked
test.convert <- rk.XML.convert(c(state=test.cbox1,state=test.cbox2), mode=c(and=""))
test.connect <- rk.XML.connect(governor=test.convert, client=test.input, set="enabled")
test.logic <- rk.XML.logic(test.convert, test.connect)
cat(pasteXML(test.logic))

# with only one checkbox, you can directly query if it's checked
test.connect2 <- rk.XML.connect(governor=test.cbox1, client=test.input, set="enabled")
test.logic2 <- rk.XML.logic(test.connect2)
cat(pasteXML(test.logic2))
```

---

rk.XML.matrix *Create XML "matrix" node for RKWard plugins*

Description

Create XML "matrix" node for RKWard plugins

Usage

```
rk.XML.matrix(label, mode = "real", rows = 2, columns = 2, min = NULL,
  max = NULL, allow_missings = FALSE, allow_user_resize_columns = TRUE,
  allow_user_resize_rows = TRUE, fixed_width = FALSE,
  fixed_height = FALSE, horiz_headers = NULL, vert_headers = NULL,
  id.name = "auto", help = NULL, component = rk.get.comp())
```

Arguments

label	Character string, a label for the matrix.
mode	Character string, one of "integer", "real" or "string". The type of data that will be accepted in the table (required)
rows	Number of rows in the matrix. Has no effect if allow_user_resize_rows=TRUE.
columns	Number of columns in the matrix. Has no effect if allow_user_resize_columns=TRUE.
min	Minimum acceptable value (if type is "integer" or "real"). Defaults to the smallest representable value.
max	Maximum acceptable value (if type is "integer" or "real"). Defaults to the largest representable value.
allow_missings	Logical, whether missing (empty) values are allowed in the matrix (if type is "string").
allow_user_resize_columns	Logical, if TRUE, the user can add columns by typing on the rightmost (inactive) cells.
allow_user_resize_rows	Logical, if TRUE, the user can add rows by typing on the bottommost (inactive) cells.
fixed_width	Logical, force the GUI element to stay at its initial width. Do not use in combindation with matrices, where the number of columns may change in any way. Useful, esp. when creating a vector input element (rows="1").
fixed_height	Logical, force the GUI element to stay at its initial height. Do not use in combindation with matrices, where the number of rows may change in any way. Useful, esp. when creating a vector input element (columns="1").
horiz_headers	Character vector to use for the horiztonal header. Defaults to column number.
vert_headers	Character vector to use for the vertical header. Defaults to row number.
id.name	Character string, a unique ID for this plugin element. If "auto", an ID will be generated automatically from the label.
help	Character string, will be used as the text value for a setting node in the .rkh file. If set to FALSE, rk.rkh.scan will ignore this node. Also needs component to be set accordingly!
component	Character string, name of the component this node belongs to. Only needed if you want to use the scan features for automatic help file generation; needs help to be set accordingly, too!

Value

An object of class XiMpLe.node.

Note

The <matrix> node was introduced with RKWard 0.6.1, please set the dependencies of your component/plugin accordingly.

See Also

and the Introduction to Writing Plugins for RKWard

Examples

```
test.matrix <- rk.XML.matrix("A matrix")
```

`rk.XML.menu` *Create XML "menu" node for RKWard plugins*

Description

This function will create a menu node for hierarchy sections. Use same `id` values to place entries in the same menu.

Usage

```
rk.XML.menu(label, ..., index = -1, id.name = "auto")
```

Arguments

`label`	Character string, a label for the menu.
`...`	Eithet objects of class `XiMpLe.node`, must be either "menu" or "entry", or a list of character strings representing the menu path, with the last element being the `component` value for `rk.XML.entry`.
`index`	Integer number to influence the level of menu placement. If `...` is a list, `index` can also be a vector of the same length + 1, so indices will be set in the same order to the menu levels, the last value is for the entry.
`id.name`	Character, a unique ID for this plugin element. If `"auto"`, an ID will be generated automatically from the label. Otherwise, if `...` is a list, `id.name` must have the same length and will be set in the same order to the menu levels. Used to place the menu in the global menu hierarchy.

Value

An object of class `XiMpLe.node`.

See Also

`rk.XML.hierarchy`, `rk.XML.entry`, `rk.XML.component`, `rk.XML.components`, and the Introduction to Writing Plugins for RKWard

Examples

```
test.component <- rk.XML.component("My GUI dialog", "plugins/MyGUIdialog.xml")
test.entry <- rk.XML.entry(test.component)
test.menu <- rk.XML.menu("Analysis", test.entry, id.name="analysis")
cat(pasteXML(test.menu))
# manual definition of a menu path by a list:
test.menu <- rk.XML.menu("Analysis", list("Level 1", "Level 2", test.component))
```

`rk.XML.option` *Create XML node "option" for RKWard plugins*

Description

Create XML node "option" for RKWard plugins

Usage

```
rk.XML.option(label, val = NULL, chk = FALSE, id.name = NULL)
```

Arguments

`label`	Character string, a text label for this plugin element.
`val`	Character string, defines the value to submit if the option is checked.
`chk`	Logical, should be set `TRUE` in the one option which is checked by default.
`id.name`	Character string, a unique ID for this plugin element. If `"auto"` and a label was provided, an ID will be generated automatically from the label.

Value

An object of class `XiMpLe.node`.

Note

You will rarely need this function, as options can be defined directly as a list in applicable functions like `rk.XML.radio`. The main purpose for having this function is to set an ID for a particular option, e.g. to be able to hide it by logic rules.

To address such an option in your logic section, the `id` you need is a combination of `<parent id>.<option id>`. That is, you must always prefix it with the parent's `id`. If you use the object an object generated by this function inside a parent node, both IDs will automatically be stored internally, so that the correct prefix will be added if needed whenever you apply logic rules to the option object.

See Also

Introduction to Writing Plugins for RKWard

Examples

```
test.radio <- rk.XML.radio("Chose one",
  options=list(
    "First Option"=c(val="val1", chk=TRUE),
    test.radio.opt2 <- rk.XML.option("Second Option", val="val2", id.name="auto"),
    "third Option"=c(val="val3"))
)
cat(pasteXML(test.radio))
```

`rk.XML.optioncolumn` *Create XML node "optioncolumn" for RKWard plugins*

Description

These nodes are valid only inside `<optionset>` nodes.

Usage

```
rk.XML.optioncolumn(connect, modifier = NULL, label = TRUE,
  external = FALSE, default = NULL, id.name = "auto", help = NULL,
  component = rk.get.comp())
```

Arguments

`connect`	Either a character string (the `id` of the property to connect this optioncolumn to), or an object of class XiMpLe.node (whose `id` will be extracted and used). For external `<optioncolumn>`s, the corresponding value will be set to the externally set value. For regular (non-external) `<optioncolumn>`s, the corresponding row of the `<optioncolumn>` property, will be set when the property changes inside the content-area.
`modifier`	Character string, the modifier of the property to connect to, will be appended to the `id` of `connect`.
`label`	Either logical or a character string. If given, the optioncolumn will be displayed in the `<optiondisplay>` in a column by that label. If set to `TRUE` and you provide a XiMpLe node object to `connect`, the label will be extracted from that node.
`external`	Logical, set to `TRUE` if the optioncolumn is controlled from outside the optionset.
`default`	Character string, only for external columns: The value to assume for this column, if no value is known for an entry. Rarely useful.
`id.name`	Character string, a unique ID for this plugin element. If `"auto"`, an ID will be generated automatically from the `connect` object.
`help`	Character string, will be used as the `text` value for a setting node in the .rkh file. If set to `FALSE`, `rk.rkh.scan` will ignore this node. Also needs `component` to be set accordingly!
`component`	Character string, name of the component this node belongs to. Only needed if you want to use the scan features for automatic help file generation; needs `help` to be set accordingly, too!

Value

An object of class XiMpLe.node.

Note

The <optionset> node was introduced with RKWard 0.6.1, please set the dependencies of your component/plugin accordingly.

See Also

rk.XML.optionset, rk.XML.optiondisplay, and the Introduction to Writing Plugins for RKWard

Examples

```
myInput <- rk.XML.input(label="Given name(s)", size="small")
myOptCol <- rk.XML.optioncolumn(myInput, modifier="text")
```

---

rk.XML.optiondisplay *Create XML node "optiondisplay" for RKWard plugins*

Description

This node is only allowed once insinde the <content> node of an <optionset>.

Usage

```
rk.XML.optiondisplay(index = TRUE, id.name = NULL)
```

Arguments

index	Logical, whether to show a column with a numeric index in the optiondisplay.
id.name	Character string, a unique ID for this plugin element (optional).

Value

An object of class XiMpLe.node.

Note

The <optionset> node was introduced with RKWard 0.6.1, please set the dependencies of your component/plugin accordingly.

See Also

rk.XML.optionset, rk.XML.optioncolumn, and the Introduction to Writing Plugins for RKWard

`rk.XML.optionset` *Create XML node "optionset" for RKWard plugins*

Description

Note that if you want to refer to the optioncolumns in your JavaScript code, the `id` you need is a combination of `<optionset id>.<optioncolumn id>.<modifier>`. that is, you must always prefix it with the sets' `id`. For JavaScript code generating with `rkwarddev`, the easiest way to get to results is to use `rk.JS.optionset`. It will automatically place your code fragments into a for loop and iterate through all available rows of the set.

Usage

```
rk.XML.optionset(content, optioncolumn, min_rows = 0, min_rows_if_any = 0,
  max_rows = 0, keycolumn = NULL, logic = NULL, optiondisplay = TRUE,
  id.name = "auto")
```

Arguments

`content`	A list of XiMpLe.nodes to be placed inside the `<content>` node of this `<optionset>`.
`optioncolumn`	A list of `<optioncolumn>` XiMpLe.nodes.
`min_rows`	Numeric (integer), if specified, the set will be marked invalid, unless it has at least this number of rows. Ignored if set to 0.
`min_rows_if_any`	Numeric (integer), like min_rows, but will only be tested, if there is at least one row. Ignored if set to 0.
`max_rows`	Numeric (integer), if specified, the set will be marked invalid, unless it has at most this number of rows. Ignored if set to 0.
`keycolumn`	Character
`logic`	A valid `<logic>` node.
`optiondisplay`	Logical value, can be used to automatically add an `<optiondisplay>` node on top of the `<content>` section. Depending on whether it's `TRUE` or `FALSE`, its `index` argument will be set to `"true"` or `"false"`, respectively. Set to `NULL` to deactivate.
`id.name`	Character string, a unique ID for this plugin element. If `"auto"`, an ID will be generated automatically from the <content> nodes.

Details

If this isn't flexible enough for your needs, you can also use the ID that functions like `id` return, because the JavaScript variable name will only contain a constant prefix ("ocol") and the column ID.

Value

An object of class `XiMpLe.node`.

Note

The <optionset> node was introduced with RKWard 0.6.1, please set the dependencies of your component/plugin accordingly.

See Also

rk.XML.optioncolumn, rk.XML.optiondisplay, rk.JS.optionset, and the Introduction to Writing Plugins for RKWard

Examples

```
firstname <- rk.XML.input("Given name(s)")
lastname <- rk.XML.input("Family name")
genderselect <- rk.XML.radio("Gender", options=list(
  Male = c(val="m"),
  Female = c(val="f")))
(myOptionset <- rk.XML.optionset(
  content = list(
    rk.XML.row(
      firstname,
      lastname,
      genderselect)),
  optioncolumn = list(
    rk.XML.optioncolumn(firstname, modifier="text"),
    rk.XML.optioncolumn(lastname, modifier="text"),
    rk.XML.optioncolumn(genderselect)
  )
))
```

---

rk.XML.page *Create XML "page" node for RKWard plugins*

Description

This function will create a page node for wizard sections, with optional child nodes "browser", "checkbox", "column", "copy", "dropdown", "formula", "frame", "input", "page", "radio", "row", "saveobject", "select", "spinbox", "stretch", "tabbook", "text", "valueselector", "valueslot", "varselector" and "varslot".

Usage

```
rk.XML.page(..., id.name = "auto")
```

Arguments

...	Objects of class XiMpLe.node.
id.name	Character string, a unique ID for this plugin element. If "auto", an ID will be generated automatically from the objects in .... If NULL, no ID will be given.

Value

An object of class XiMpLe.node.

See Also

rk.XML.wizard, and the Introduction to Writing Plugins for RKWard

Examples

```
# define a checkbox for the actual dialog
test.cbox1 <- rk.XML.cbox(label="More than 30 subjects", val="true")
# define the wizard
test.text <- rk.XML.text("Did you test more than 30 subjects?")
test.copy <- rk.XML.copy(id=test.cbox1)
test.wizard <- rk.XML.wizard(rk.XML.page(test.text, test.copy))
cat(pasteXML(test.wizard))
```

---

rk.XML.plugin *Create XML document for RKWard plugins*

Description

Create XML document for RKWard plugins

Usage

```
rk.XML.plugin(name, dialog = NULL, wizard = NULL, logic = NULL,
  snippets = NULL, provides = NULL, help = TRUE, include = NULL,
  label = NULL, clean.name = TRUE, about = NULL, dependencies = NULL,
  gen.info = TRUE, lang = rk.get.language())
```

Arguments

name	Character string, the name of the plugin. Will be used for the names of the JavaScript and help files to be included, following the scheme *"<name>.<ext>"*.
dialog	An object of class XiMpLe.node to be pasted as the <dialog> section. See rk.XML.dialog for details.
wizard	An object of class XiMpLe.node to be pasted as the <wizard> section. See rk.XML.wizard for details.
logic	An object of class XiMpLe.node to be pasted as the <logic> section. See rk.XML.logic for details.
snippets	An object of class XiMpLe.node to be pasted as the <snippets> section. See rk.XML.snippets for details.
provides	Character vector with possible entries of "logic", "dialog" or "wizard", defining what sections the document should provide even if dialog, wizard and logic are NULL. These sections must be edited manually and some parts are therefore commented out.

help	Logical, if TRUE an include tag for a help file named *"<name>.rkh"* will be added to the header.
include	Character string or vector, relative path(s) to other file(s), which will then be included in the head of the GUI XML document.
label	Character string, a text label for the plugin's top level, i.e. the window title of the dialog. Will only be used if dialog or wizard are NULL.
clean.name	Logical, if TRUE, all non-alphanumeric characters except the underscore ("_") will be removed from name.
about	An object of class XiMpLe.node with descriptive information on the plugin and its authors, see link[XiMpLe:rk.XML.about]{rk.XML.about} for details. Only useful for information that differs from the <about> section of the .pluginmap file. Skipped if NULL.
dependencies	An object of class XiMpLe.node to be pasted as the <dependencies> section, See rk.XML.dependencies for details. Skipped if NULL.
gen.info	Logical, if TRUE a comment note will be written into the document, that it was generated by rkwarddev and changes should be done to the script.
lang	Character string, the language of the plugin. See i18n for details.

Value

An object of class XiMpLe.doc.

See Also

Introduction to Writing Plugins for RKWard

Examples

```
## Not run:
test.checkboxes <- rk.XML.row(rk.XML.col(
  list(
    rk.XML.cbox(label="foo", val="foo1", chk=TRUE),
    rk.XML.cbox(label="bar", val="bar2"))))
test.dropdown <- rk.XML.dropdown("mydrop",
  options=list("First Option"=c(val="val1"),
  "Second Option"=c(val="val2", chk=TRUE)))
# combine the above into a tabbook
test.tabbook <- rk.XML.tabbook("My Tabbook", tabs=c(
  "First Tab"=test.checkboxes, "Second Tab"=test.dropdown))
# make a plugin with that tabbook
test.plugin <- rk.XML.plugin("My test", dialog=rk.XML.dialog(test.tabbook))

## End(Not run)
```

`rk.XML.pluginmap` *Write a pluginmap file for RKWard*

Description

Write a pluginmap file for RKWard

Usage

```
rk.XML.pluginmap(name, about = NULL, components, hierarchy = "test",
  require = NULL, x11.context = NULL, import.context = NULL,
  clean.name = TRUE, hints = FALSE, gen.info = TRUE,
  dependencies = NULL, namespace = name, priority = "medium",
  id.name = "auto", lang = rk.get.language())
```

Arguments

`name`	Character string, name of the plugin.
`about`	An object of class `XiMpLe.node` to be pasted as the <about> section, See `link[XiMpLe:rk.XML.about]{` for details. Skipped if `NULL`.
`components`	Either an object of class `XiMpLe.node` to be pasted as the <components> section (see `rk.XML.components` for details). Or a character vector with at least one plugin component file name, relative path from the pluginmap file and ending with ".xml". Can be set to `NULL` if `require` is used accordingly.
`hierarchy`	Either an object of class `XiMpLe.node` to be pasted as the <hierarchy> section (see `rk.XML.hierarchy` for details). Or a character vector with instructions where to place the plugin in the menu hierarchy, one list or string for each included component. Valid single values are `"file"`, `"edit"`, `"view"`, `"workspace"`, `"run"`, `"data"`, `"analysis"`, `"plots"`, `"distributions"`, `"windows"`, `"settings"` and `"help"`, anything else will place it in a "test" menu. If `hierarchy` is a list, each entry represents the label of a menu level. Can be set to `NULL` if `require` is used accordingly.
`require`	Either a (list of) objects of class `XiMpLe.node` to be pasted as a <require> section (see `rk.XML.require` for details). Or a character vector with at least one .pluginmap filename to be included in this one.
`x11.context`	An object of class `XiMpLe.node` to be pasted as a <context id="x11"> section, see `rk.XML.context` for details.
`import.context`	An object of class `XiMpLe.node` to be pasted as the <context id="import"> section, see `rk.XML.context` for details.
`clean.name`	Logical, if `TRUE`, all non-alphanumeric characters except the underscore (`"_"`) will be removed from `name`.
`hints`	Logical, if `TRUE` and you leave out optional entries (like `about=NULL`), dummy sections will be added as comments.
`gen.info`	Logical, if `TRUE` a comment note will be written into the document, that it was generated by `rkwarddev` and changes should be done to the script.

dependencies	An object of class XiMpLe.node to be pasted as the <dependencies> section, See rk.XML.dependencies for details. Skipped if NULL.
namespace	Character string, the namespace attribute of the <document> node, defaults to the plugin name (which you probably shouldn't touch...)
priority	Character string, the priority attribute of the <document> node. Must be either "hidden", "low", "medium", or "high", defaults to "medium".
lang	Character string, the language of the document. See i18n for details.

Value

An object of class XiMpLe.node.

See Also

Introduction to Writing Plugins for RKWard

---

rk.XML.preview *Create XML node "preview" for RKWard plugins*

Description

Create XML node "preview" for RKWard plugins

Usage

```
rk.XML.preview(label = "Preview")
```

Arguments

label	A character string, text label for the preview checkbox.

Value

An object of class XiMpLe.node.

See Also

Introduction to Writing Plugins for RKWard

Examples

```
test.preview <- rk.XML.preview("See a preview?")
cat(pasteXML(test.preview))
```

rk.XML.radio *Create XML node "radio" for RKWard plugins*

Description

Create XML node "radio" for RKWard plugins

Usage

```
rk.XML.radio(label, options = list(label = c(val = NULL, chk = FALSE)),
  id.name = "auto", help = NULL, component = rk.get.comp())
```

Arguments

label	Character string, a text label for this plugin element.
options	A named list with options to choose from. The names of the list elements will become labels of the options, val defines the value to submit if the option is checked, and chk=TRUE should be set in the one option which is checked by default. Objects generated with rk.XML.option are accepted as well.
id.name	Character string, a unique ID for this plugin element. If "auto" and a label was provided, an ID will be generated automatically from the label.
help	Character string, will be used as the text value for a setting node in the .rkh file. Also needs component to be set accordingly!
component	Character string, name of the component this node belongs to. Only needed if you want to use the scan features for automatic help file generation; needs help to be set accordingly, too!

Value

An object of class XiMpLe.node.

Note

It is also possible to address a particular option by giving it an ID, probably useful in logic sections. Have a look at rk.XML.option for details.

See Also

rk.XML.option, Introduction to Writing Plugins for RKWard

Examples

```
test.radio <- rk.XML.radio("Chose one",
  options=list("First Option"=c(val="val1"),
  "Second Option"=c(val="val2", chk=TRUE)))
cat(pasteXML(test.radio))
```

`rk.XML.require` *Create XML "require" node for RKWard plugins*

Description

This function will create a require node for .pluginmap files.

Usage

```
rk.XML.require(file = NULL, map = NULL, localized = FALSE)
```

Arguments

`file`	Character string, file name of another .pluginmap file to be included. Should be preferred over `map` if that file is in the same package.
`map`	Character string, should be `"namespace::id"` of another .pluginmap to be included. Can be used to address plugin maps which are not part of the same plugin package.
`localized`	Logical, only useful for plugins with static internationalisation. Adds the `localized` attribute if set to `TRUE`; `file` must then point to the default pluginmap in a subdirectory containing all localised pluginmaps.

Details

Note that only one of the values can be set at a time. `file` should be preferred whenever possible.

Value

An object of class `XiMpLe.node`.

See Also

Introduction to Writing Plugins for RKWard

Examples

```
test.require <- rk.XML.require("another.pluginmap")
cat(pasteXML(test.require))
```

rk.XML.row *Create XML node "row" for RKWard plugins*

Description

Create XML node "row" for RKWard plugins

Usage

```
rk.XML.row(..., id.name = "auto")
```

Arguments

...	Objects of class XiMpLe.node.
id.name	Character string, a unique ID for this plugin element. If "auto", an ID will be generated automatically from the objects in ... If NULL, no ID will be given.

Value

An object of class XiMpLe.node.

See Also

Introduction to Writing Plugins for RKWard

Examples

```
test.checkboxes <- rk.XML.row(rk.XML.col(
    rk.XML.cbox(label="foo", val="foo1", chk=TRUE),
    rk.XML.cbox(label="bar", val="bar2")))
cat(pasteXML(test.checkboxes))
```

rk.XML.saveobj *Create XML node "saveobject" for RKWard plugins*

Description

Create XML node "saveobject" for RKWard plugins

Usage

```
rk.XML.saveobj(label, chk = FALSE, checkable = TRUE, initial = "auto",
  required = FALSE, id.name = "auto", help = NULL,
  component = rk.get.comp())
```

Arguments

label	Character string, a text label for this plugin element.
chk	Logical, if TRUE and checkable=TRUE the option is checkable and active by default.
checkable	Logical, if TRUE the option can be switched on and off.
initial	Character string, the default name for the object should be saved to. If "auto" and a label was provided, an name will be generated automatically from the label.
required	Logical, whether an entry is mandatory or not.
id.name	Character string, a unique ID for this plugin element. If "auto" and a label was provided, an ID will be generated automatically from the label.
help	Character string, will be used as the text value for a setting node in the .rkh file. If set to FALSE, rk.rkh.scan will ignore this node. Also needs component to be set accordingly!
component	Character string, name of the component this node belongs to. Only needed if you want to use the scan features for automatic help file generation; needs help to be set accordingly, too!

Value

An object of class XiMpLe.node.

See Also

Introduction to Writing Plugins for RKWard

Examples

```
test.saveobj <- rk.XML.saveobj("Save the results")
cat(pasteXML(test.saveobj))
```

---

rk.XML.select *Create XML node "select" for RKWard plugins*

Description

Create XML node "select" for RKWard plugins

Usage

```
rk.XML.select(label, options = list(label = c(val = "", chk = FALSE)),
  id.name = "auto", help = NULL, component = rk.get.comp())
```

Arguments

label	Character string, a text label for this plugin element.
options	A named list with options to choose from. The names of the list elements will become labels of the options, val defines the value to submit if the option is selected, and chk=TRUE should be set in the one option which is selected by default. Objects generated with rk.XML.option are accepted as well.
id.name	Character string, a unique ID for this plugin element. If "auto" and a label was provided, an ID will be generated automatically from the label.
help	Character string, will be used as the text value for a setting node in the .rkh file. If set to FALSE, rk.rkh.scan will ignore this node. Also needs component to be set accordingly!
component	Character string, name of the component this node belongs to. Only needed if you want to use the scan features for automatic help file generation; needs help to be set accordingly, too!

Value

An object of class XiMpLe.node.

See Also

Introduction to Writing Plugins for RKWard

Examples

```
test.select <- rk.XML.select("myselect",
  options=list("First Option"=c(val="val1"),
  "Second Option"=c(val="val2", chk=TRUE)))
cat(pasteXML(test.select))
```

---

rk.XML.set *Create XML node "set" for RKWard plugins*

Description

Create XML node "set" for RKWard plugins

Usage

```
rk.XML.set(id, set = NULL, to, check.modifiers = TRUE)
```

Arguments

id	Either a character string (the id of the property whose value should be set), or an object of class XiMpLe.node (whose id will be extracted and used).
set	Character string, a valid modifier.
to	Character string or logical, the value the property should be set to.
check.modifiers	Logical, if TRUE the given modifiers will be checked for validity. Should only be turned off if you know what you're doing.

Value

An object of class XiMpLe.node.

See Also

rk.XML.connect, rk.XML.external, rk.XML.logic, rk.XML.set, rk.XML.switch, and the Introduction to Writing Plugins for RKWard

Examples

```
test.set <- rk.XML.set(id="input_foo", set="required", to=TRUE)
cat(pasteXML(test.set))
```

---

rk.XML.snippet *Create XML "snippet" node for RKWard plugins*

Description

This function will create a snippet node for snippets sections.

Usage

```
rk.XML.snippet(..., id.name = "auto")
```

Arguments

...	Objects of class XiMpLe.node.
id.name	Character string, a unique ID for this plugin element. If "auto", an ID will be generated automatically from the tag names and IDs of the given nodes.

Value

An object of class XiMpLe.node.

See Also

rk.XML.snippets, and the Introduction to Writing Plugins for RKWard

Examples

```
# define a formula section with varselector and varslots
test.formula <- rk.XML.vars("Variables", "Fixed", formula.dependent="Dependent")
# define the snippet
test.snippet <- rk.XML.snippet(test.formula)
cat(pasteXML(test.snippet))
```

`rk.XML.snippets` *Create XML "snippets" node for RKWard plugins*

Description

This function will create a snippets node for the document section, with optional child nodes "snippet".

Usage

```
rk.XML.snippets(...)
```

Arguments

`...`	Objects of class `XiMpLe.node`. They must all have the name "snippet".

Value

An object of class `XiMpLe.node`.

See Also

`rk.XML.plugin` `rk.XML.snippet`, and the Introduction to Writing Plugins for RKWard

Examples

```
# define a formula section with varselector and varslots
test.formula <- rk.XML.vars("Variables", "Fixed", formula.dependent="Dependent")
# define the snippets section
test.snippet <- rk.XML.snippet(test.formula)
test.snippets <- rk.XML.snippets(test.snippet)
cat(pasteXML(test.snippets))
```

rk.XML.spinbox *Create XML node "spinbox" for RKWard plugins*

Description

Create XML node "spinbox" for RKWard plugins

Usage

```
rk.XML.spinbox(label, min = NULL, max = NULL, initial = 0, real = TRUE,
  precision = 2, max.precision = 8, id.name = "auto", help = NULL,
  component = rk.get.comp())
```

Arguments

label	Character string, a text label for this plugin element.
min	Numeric, the lowest value allowed. Defaults to the lowest value technically representable in the spinbox.
max	Numeric, the largest value allowed. Defaults to the highest value technically representable in the spinbox.
initial	Numeric, will be used as the initial value.
real	Logical, whether values should be real or integer numbers.
precision	Numeric, if real=TRUE defines the default number of decimal places shown in the spinbox.
max.precision	Numeric, maximum number of digits that can be meaningfully represented.
id.name	Character string, a unique ID for this plugin element. If "auto" and a label was provided, an ID will be generated automatically from the label.
help	Character string, will be used as the text value for a setting node in the .rkh file. If set to FALSE, rk.rkh.scan will ignore this node. Also needs component to be set accordingly!
component	Character string, name of the component this node belongs to. Only needed if you want to use the scan features for automatic help file generation; needs help to be set accordingly, too!

Value

An object of class XiMpLe.node.

See Also

Introduction to Writing Plugins for RKWard

Examples

```
test.spinbox <- rk.XML.spinbox("Spin this:")
cat(pasteXML(test.spinbox))
```

rk.XML.stretch *Create XML empty node "stretch" for RKWard plugins*

Description

The simplest way to use `rk.XML.stretch` is to call it without arguments. If you provide `before` and/or `after`, a "<stretch />" will be put between the XML elements defined there.

Usage

```
rk.XML.stretch(before = NULL, after = NULL)
```

Arguments

`before`	A list of objects of class `XiMpLe.node`.
`after`	A list of objects of class `XiMpLe.node`.

Value

An object of class `XiMpLe.node`.

See Also

Introduction to Writing Plugins for RKWard

Examples

```
cat(pasteXML(rk.XML.stretch()))
```

rk.XML.switch *Create XML node "switch" for RKWard plugins*

Description

This node can only be used in `<logic>` sections. If the provided property is logical, in the `cases` list you must also provide lists called `true` and `false`. If not, you must provide at least one list called `case`.

Usage

```
rk.XML.switch(condition, cases, modifier = NULL, id.name = "auto")
```

Arguments

condition	Either a character string (the id of the property whose state should be queried), or an object of class XiMpLe.node (whose id will be extracted and used).
cases	A named list of named lists. The lists contained must either be called true and false, setting the return values if condition is logical, or case and optionally default. You can provide as many case lists as you need, setting a return value for each condition == case respectively. Each list must contain either a fixed_value or a dynamic_value element. In addition, each case list must also have one standard element.
modifier	Character string, an optional modifier to be appended to condition.
id.name	Character string, a unique ID for this property. If "auto", IDs will be generated automatically from the condition ID.

Details

The values to be returned can be either fixed_value or dynamic_value. A fixed_value must be a character string which will be returned if the condition is met. Whereas a dynamic_value is the id of another property, an can be provided as either a character string or an object of class XiMpLe.node.

Value

An object of class XiMpLe.node.

Note

The <switch> node was introduced with RKWard 0.6.1, please set the dependencies of your component/plugin accordingly.

See Also

rk.XML.connect, rk.XML.convert, rk.XML.external, rk.XML.logic, rk.XML.set, and the Introduction to Writing Plugins for RKWard

Examples

```
# example for a boolean switch
myCheckbox <- rk.XML.cbox("foo")
rk.XML.switch(myCheckbox, cases=list(
  true=list(fixed_value="foo"),
  false=list(fixed_value="bar"))
)

# example for a case switch
MyRadio <- rk.XML.radio("Chose one",
  options=list(
    "First Option"=c(val="val1"),
    "Second Option"=c(val="val2", chk=TRUE))
)
```

```
rk.XML.switch(MyRadio, modifier="string", cases=list(
  case=list(standard="val1", fixed_value="foo"),
  case=list(standard="val2", fixed_value="bar"))
)
```

`rk.XML.tabbook` *Create XML node "tabbook" for RKWard plugins*

Description

Create XML node "tabbook" for RKWard plugins

Usage

```
rk.XML.tabbook(label = NULL, tabs = list(), id.name = "auto")
```

Arguments

`label`	Character string, a text label for this plugin element.
`tabs`	An optional named list with objects of class `XiMpLe.node` (or a list of these objects). You must provide one named element for each tab. Use `NULL` for tabs without predefined children.
`id.name`	Character vector, unique IDs for the tabbook (first entry) and all tabs. If `"auto"`, IDs will be generated automatically from the labels. If `NULL`, no IDs will be given.

Value

An object of class `XiMpLe.node`.

See Also

Introduction to Writing Plugins for RKWard

Examples

```
test.checkboxes <- rk.XML.row(rk.XML.col(
  rk.XML.cbox(label="foo", val="foo1", chk=TRUE),
  rk.XML.cbox(label="bar", val="bar2")))
test.dropdown <- rk.XML.dropdown("mydrop",
  options=list("First Option"=c(val="val1"),
  "Second Option"=c(val="val2", chk=TRUE)))
# combine the above into a tabbook
test.tabbook <- rk.XML.tabbook("My Tabbook",
  tabs=list("First Tab"=test.checkboxes, "Second Tab"=test.dropdown))
cat(pasteXML(test.tabbook))
```

rk.XML.text *Create XML node "text" for RKWard plugins*

Description

Create XML node "text" for RKWard plugins

Usage

```
rk.XML.text(text, type = "normal", id.name = "auto")
```

Arguments

text	Character string, the text to be displayed.
type	One value of either "normal", "warning" or "error".
id.name	Character string, a unique ID for this plugin element. If "auto", an ID will be generated automatically from text. If NULL, no ID will be given.

Value

An object of class XiMpLe.node.

See Also

Introduction to Writing Plugins for RKWard

Examples

```
test.text <- rk.XML.text("Added this text.")
cat(pasteXML(test.text))
```

rk.XML.values *Create a value selector for RKWard plugins*

Description

This function will create a <frame> node including a <valueselector> and a <valueslot> node. It is actually a wrapper for rk.XML.valueslot and rk.XML.valueselector, since you usually won't define one without the other.

Usage

```
rk.XML.values(label, slot.text, required = FALSE, multi = FALSE, min = 1,
  any = 1, max = 0, horiz = TRUE, add.nodes = NULL,
  frame.label = NULL, id.name = "auto", help = NULL,
  component = rk.get.comp())
```

Arguments

label	Character string, a text label for the value browser.
slot.text	Character string, a text label for the value selection slot.
required	Logical, whether the selection of values is mandatory or not.
multi	Logical, whether the valueslot holds only one or several objects.
min	If multi=TRUE defines how many objects must be selected.
any	If multi=TRUE defines how many objects must be selected at least if any are selected at all.
max	If multi=TRUE defines how many objects can be selected in total (0 means any number).
horiz	Logical. If TRUE, the valueslot will be placed next to the selector, if FALSE below it.
add.nodes	A list of objects of class XiMpLe.node to be placed after the valueslot.
frame.label	Character string, a text label for the whole frame.
id.name	Character vector, unique IDs for the frame (first entry), the valueselector (second entry) and valueslot (third entry). If formula.dependent is not NULL, a fourth and fifth entry is needed as well, for the dependent valueslot and the formula node, respectively. If "auto", IDs will be generated automatically from label and slot.text.
help	Character string, will be used as the text value for a setting node in the .rkh file. If set to FALSE, rk.rkh.scan will ignore this node. Also needs component to be set accordingly!
component	Character string, name of the component this node belongs to. Only needed if you want to use the scan features for automatic help file generation; needs help to be set accordingly, too!

Value

An object of class XiMpLe.node.

See Also

rk.XML.valueslot, rk.XML.valueselector, and the Introduction to Writing Plugins for RKWard

Examples

```
test.values <- rk.XML.values("Select some values", "Vars go here")
cat(pasteXML(test.values))
```

rk.XML.valueselector *Create node "valueselector" for RKWard plugins*

Description

Create node "valueselector" for RKWard plugins

Usage

```
rk.XML.valueselector(label = NULL, options = list(label = c(val = NULL, chk
  = FALSE)), id.name = "auto")
```

Arguments

label	Character string, a text label for the value selection slot. Must be set if id.name="auto".
options	A named list with string values to choose from. The names of the list elements will become labels of the options, val defines the value to submit if the value is selected, and chk=TRUE should be set in the one option which is checked by default. Objects generated with rk.XML.option are accepted as well.
id.name	Character vector, unique ID for this element.

Value

An object of class XiMpLe.node.

See Also

rk.XML.valueslot, rk.XML.values, rk.XML.option, and the Introduction to Writing Plugins for RKWard

Examples

```
test.valueselector <- rk.XML.valueselector("Select some values")
cat(pasteXML(test.valueselector))
```

rk.XML.valueslot *Create a XML node "valueslot" for RKWard plugins*

Description

Create a XML node "valueslot" for RKWard plugins

Usage

```
rk.XML.valueslot(label, source, required = FALSE, multi = FALSE, min = 1,
  any = 1, max = 0, id.name = "auto", help = NULL,
  component = rk.get.comp())
```

Arguments

label	Character string, a text label for the valueslot.
source	Either a character string (the id name of the valueselector to select values from), or an object of class XiMpLe.node (whose id will be extracted and used, must be a <valueselector> node).
required	Logical, whether the selection of values is mandatory or not.
multi	Logical, whether the valueslot holds only one or several objects.
min	If multi=TRUE defines how many objects must be selected. Sets multi=TRUE.
any	If multi=TRUE defines how many objects must be selected at least if any are selected at all. Sets multi=TRUE.
max	If multi=TRUE defines how many objects can be selected in total (0 means any number). Sets multi=TRUE.
id.name	Character vector, unique ID for the valueslot. If "auto", the ID will be generated automatically from label.
help	Character string, will be used as the text value for a setting node in the .rkh file. If set to FALSE, rk.rkh.scan will ignore this node. Also needs component to be set accordingly!
component	Character string, name of the component this node belongs to. Only needed if you want to use the scan features for automatic help file generation; needs help to be set accordingly, too!

Value

An object of class XiMpLe.node.

See Also

rk.XML.values, rk.XML.valueselector, and the Introduction to Writing Plugins for RKWard

Examples

```
## Not run:
test.valueselector <- rk.XML.valueselector("Select some values")
test.valueslot <- rk.XML.valueslot("Vars go here", source=test.valueselector)
cat(pasteXML(test.valueslot))

## End(Not run)
```

rk.XML.vars *Create a variable selector for RKWard plugins*

Description

This function will create a <frame> node including a <varselector> and a <varslot> node. It is actually a wrapper for rk.XML.varslot and rk.XML.varselector, since you usually won't define one without the other.

Usage

```
rk.XML.vars(label, slot.text, required = FALSE, multi = FALSE, min = 1,
  any = 1, max = 0, dim = 0, min.len = 0, max.len = NULL,
  classes = NULL, types = NULL, horiz = TRUE, add.nodes = NULL,
  frame.label = NULL, formula.dependent = NULL, dep.options = list(),
  id.name = "auto", help = NULL, component = rk.get.comp())
```

Arguments

label	Character string, a text label for the variable browser.
slot.text	Character string, a text label for the variable selection slot.
required	Logical, whether the selection of variables is mandatory or not.
multi	Logical, whether the varslot holds only one or several objects.
min	If multi=TRUE defines how many objects must be selected.
any	If multi=TRUE defines how many objects must be selected at least if any are selected at all.
max	If multi=TRUE defines how many objects can be selected in total (0 means any number).
dim	The number of dimensions, an object needs to have. If dim=0 any number of dimensions is acceptable.
min.len	The minimum length, an object needs to have.
max.len	The maximum length, an object needs to have. If NULL, defaults to the largest integer number representable on the system.
classes	An optional character vector, defining class names to which the selection must be limited.
types	If you specify one or more variables types here, the varslot will only accept objects of those types. Valid types are "unknown", "number", "string", "factor", "invalid". Optional, use with great care, the user should not be prevented from making valid choices, and rkward does not always know the type of a variable!
horiz	Logical. If TRUE, the varslot will be placed next to the selector, if FALSE below it.
add.nodes	A list of objects of class XiMpLe.node to be placed after the varslot.
frame.label	Character string, a text label for the whole frame.

formula.dependent
: Character string, if not NULL will cause the addition of a second varslot for the dependent variable(s), using the text of formula.dependent as its label. Also a <formula> node will be added, using both varslots for fixed_factors and dependent respectively.

dep.options
: A named list with optional attributes for the dependent varslot, if formula.dependent is not NULL. Valid options are required, multi, min, any, max, dim, min.len, max.len, classes and types. If an options is undefined, it defaults to the same values like the main options of this function.

id.name
: Character vector, unique IDs for the frame (first entry), the varselector (second entry) and varslot (third entry). If formula.dependent is not NULL, a fourth and fifth entry is needed as well, for the dependent varslot and the formula node, respectively. If "auto", IDs will be generated automatically from label and slot.text.

help
: Character string, will be used as the text value for a setting node in the .rkh file. If set to FALSE, rk.rkh.scan will ignore this node. Also needs component to be set accordingly!

component
: Character string, name of the component this node belongs to. Only needed if you want to use the scan features for automatic help file generation; needs help to be set accordingly, too!

Value

An object of class XiMpLe.node.

See Also

rk.XML.varslot, rk.XML.varselector, and the Introduction to Writing Plugins for RKWard

Examples

```
test.vars <- rk.XML.vars("Select some vars", "Vars go here")
cat(pasteXML(test.vars))
```

---

rk.XML.varselector *Create node "varselector" for RKWard plugins*

Description

Create node "varselector" for RKWard plugins

Usage

```
rk.XML.varselector(label = NULL, id.name = "auto")
```

Arguments

label	Character string, a text label for the variable selection slot. Must be set if id.name="auto".
id.name	Character vector, unique ID for this element.

Value

An object of class XiMpLe.node.

See Also

rk.XML.varslot, rk.XML.vars, and the Introduction to Writing Plugins for RKWard

Examples

```
test.varselector <- rk.XML.varselector("Select some vars")
cat(pasteXML(test.varselector))
```

---

rk.XML.varslot *Create a XML node "varslot" for RKWard plugins*

Description

Create a XML node "varslot" for RKWard plugins

Usage

```
rk.XML.varslot(label, source, required = FALSE, multi = FALSE, min = 1,
  any = 1, max = 0, dim = 0, min.len = 0, max.len = NULL,
  classes = NULL, types = NULL, id.name = "auto", help = NULL,
  component = rk.get.comp())
```

Arguments

label	Character string, a text label for the varslot.
source	Either a character string (the id name of the varselector to select variables from), or an object of class XiMpLe.node (whose id will be extracted and used, must be a <varselector> node).
required	Logical, whether the selection of variables is mandatory or not.
multi	Logical, whether the varslot holds only one or several objects.
min	If multi=TRUE defines how many objects must be selected. Sets multi=TRUE.
any	If multi=TRUE defines how many objects must be selected at least if any are selected at all. Sets multi=TRUE.
max	If multi=TRUE defines how many objects can be selected in total (0 means any number). Sets multi=TRUE.

dim	The number of dimensions, an object needs to have. If dim=0 any number of dimensions is acceptable.
min.len	The minimum length, an object needs to have.
max.len	The maximum length, an object needs to have. If NULL, defaults to the largest integer number representable on the system.
classes	An optional character vector, defining class names to which the selection must be limited.
types	If you specify one or more variables types here, the varslot will only accept objects of those types. Valid types are "unknown", "number", "string", "factor", "invalid". Optional, use with great care, the user should not be prevented from making valid choices, and rkward does not always know the type of a variable!
id.name	Character vector, unique ID for the varslot. If "auto", the ID will be generated automatically from label.
help	Character string, will be used as the text value for a setting node in the .rkh file. If set to FALSE, rk.rkh.scan will ignore this node. Also needs component to be set accordingly!
component	Character string, name of the component this node belongs to. Only needed if you want to use the scan features for automatic help file generation; needs help to be set accordingly, too!

Value

An object of class XiMpLe.node.

See Also

rk.XML.vars, rk.XML.varselector, and the Introduction to Writing Plugins for RKWard

Examples

```
## Not run:
test.varselector <- rk.XML.varselector("Select some vars")
test.varslot <- rk.XML.varslot("Vars go here", source=test.varselector)
cat(pasteXML(test.varslot))

## End(Not run)
```

---

rk.XML.wizard *Create XML wizard section for RKWard plugins*

Description

This function will create a wizard section with optional child nodes "browser", "checkbox", "column", "copy", "dropdown", "embed", "formula", "frame", "include", "input", "insert", "page", "preview", "radio", "row", "saveobject", "select", "spinbox", "stretch", "tabbook", "text", "valueselector", "valueslot", "varselector" and "varslot".

Usage

```
rk.XML.wizard(..., label = NULL, recommended = FALSE)
```

Arguments

...	Objects of class XiMpLe.node
label	Character string, a text label for this plugin element.
recommended	Logical, whether the wizard should be the recommended interface (unless the user has configured RKWard to default to a specific interface).

Value

An object of class XiMpLe.node.

See Also

rk.XML.plugin, rk.plugin.skeleton, and the Introduction to Writing Plugins for RKWard

Examples

```
# define a checkbox for the actual dialog
test.cbox1 <- rk.XML.cbox(label="More than 30 subjects", val="true")
# define the wizard
test.text <- rk.XML.text("Did you test more than 30 subjects?")
test.copy <- rk.XML.copy(id=test.cbox1)
test.wizard <- rk.XML.wizard(rk.XML.page(list(test.text, test.copy)))
cat(pasteXML(test.wizard))
```

---

rkwarddev.required *Check for rkwarddev package version requirements*

Description

Check for rkwarddev package version requirements

Usage

```
rkwarddev.required(min = "0.06-5", lib.loc = NULL)
```

Arguments

min	The minimum version number of rkwarddev that is required to run this script.
lib.loc	The lib.loc argument passed over to packageVersion.

Value

The function has no return value, but wil stop with an error if the specified version requirement is not met.

Examples

```
rkwarddev.required(min="0.06-5")
```

---

show *Show methods for S4 objects of class* rk.JS.*

Description

Show methods for S4 objects of class rk.JS.*

Usage

```
show(object)

## S4 method for signature 'rk.JS.arr'
show(object)

## S4 method for signature 'rk.JS.ite'
show(object)

## S4 method for signature 'rk.JS.opt'
show(object)

## S4 method for signature 'rk.JS.oset'
show(object)

## S4 method for signature 'rk.JS.var'
show(object)

## S4 method for signature 'rk.JS.echo'
show(object)
```

Arguments

object	An object of class rk.JS.*

---

tf *Replace checkbox XML objects with JavaScript code*

Description

This function is a basically shortcut for ite with some assumptions. It's thought to be used when a checkbox should turn an option of an R function to a specified value, by default TRUE or FALSE (hence the name, abbreviated "true or false"). The same result can be obtained with ite, but for most common cases tf is much quicker.

Usage

```
tf(cbox, true = TRUE, not = FALSE, ifelse = FALSE, false = FALSE,
  opt = NULL, prefix = ",\n", level = 3, indent.by = "\t")
```

Arguments

cbox	An object of class XiMpLe.node containing a <checkbox> node, as generated by rk.XML.cbox.
true	Logical or character, the value the option should get. E.g., if true=TRUE then the option will be set to TRUE if the box is checked, or in case not=TRUE, if the box is not checked.
not	Logical, inverses the checked status of the checkbox. In other words, set this to TRUE if you want the option to be set if the box is not checked.
ifelse	Logical, whether the options should be set anyway. By default, the option will only be set in one condition. If ifelse=TRUE, it will get the inverse value in case of the alternative condition, e.g. it will be set to either not=TRUE or not=FALSE if the box is checked or unchecked.
false	Logical or character, the value the option should, only used get if ifelse=TRUE as well. E.g., if false=FALSE then the option will be set to FALSE if the box is not checked, or in case not=TRUE, if the box is checked.
opt	A character string, naming the R option o be set. If NULL, the XML ID of the checkbox node will be used.
prefix	A character string, what should be pasted before the actual option string. Default is a comma and a newline.
level	Integer, which indentation level to use, minimum is 1.
indent.by	A character string defining the indentation string to use. This refers to the genrated R code, not the JavaScript code. Indentation is added after the prefix and before the option string.

Value

An object of class rk.JS.ite.

See Also

ite, echo, id, and the Introduction to Writing Plugins for RKWard

Examples

```
# an example checkbox XML node
cbox1 <- rk.XML.cbox(label="foo", value="foo1", id.name="foo_option")
tf(cbox1)
```

Index